FUJITSU

# FUJITSU Software
# Technical Computing Suite V2.0

# Job Operation Software
# End-user's Guide

J2UL-1837-02ENZ0(00)
January 2016

# Preface

## Purpose of This Manual

This manual describes the job operation procedures of "Job Operation Software" included in Technical Computing Suite V2.0.

## Intended Readers

This manual is intended the administrators who operate and manage jobs with Job Operation Software, and the end-user to actually operate job.

The manual assumes that readers as follows.

- Linux basic knowledge is required.
- Understand the overview of Job Operation Software in the "Job Operation Software First Step Guide."

## Organization of This Manual

This manual is organized as follows.

Chapter 1 Job Mechanism

This chapter describes the mechanism of jobs.

Chapter 2 Job Operation Procedures

This chapter describes job operation procedures.

Appendix A Job Information

This appendix describes the outputs of the command concerning information on the job.

Appendix B Notification Message Related to Job Execution

This appendix describes the messages that are notified to user by e-mail, if a job terminates abnormally.

Appendix C Executing programs of MPI processing system other than Technical Computing Language

This appendix describes how to execute MPI programs of the MPI processing system other than Technical Computing Language on the job operation software and the notes on executing them.

Appendix D Notes on Systems That Use Emergency Jobs [FX100]

This appendix describes the influence on the user program and notes on making the user program when the emergency job is used.

Appendix E Operations on Jobs

This appendix describes the relationship between job states and possibility of operations on jobs.

Appendix F References

The appendix describes how to refer the man pages of commands used by end-users and their messages.

Glossary

The glossary describes key terms related to the setup function.

## Notation Used in This Manual

### Notation of users

The users of the job operation software include the administrators responsible for system management and job operations and the end users who use the system to run programs. Unless otherwise noted, "user" in this manual means an end user.

### Representation of units

The following table lists the prefixes used to represent units in this manual. Basically, disk size is represented as a power of 10, and memory size is represented as a power of 2. Be careful about specifying them when displaying or entering commands.

| Prefix | Value | Prefix | Value |
|---|---|---|---|
| K (kilo) | $10^3$ | Ki (kibi) | $2^{10}$ |
| M (mega) | $10^6$ | Mi (mebi) | $2^{20}$ |
| G (giga) | $10^9$ | Gi (gibi) | $2^{30}$ |
| T (tera) | $10^{12}$ | Ti (tebi) | $2^{40}$ |
| P (peta) | $10^{15}$ | Pi (pebi) | $2^{50}$ |

### Notation of model names

In this manual, "PRIMEHPC FX100" is abbreviated as "FX100", "PRIMEHPC FX10" as "FX10".
Also, specifications of some of the functions described in the manual are different depending on the target model (FX100, FX10 or PRIMERGY). In the description of such a function, the target model is represented by its abbreviation as follows:
[FX100]: The description applies to FX100 functions.
[FX10]: The description applies to FX10 functions.
[FX100/FX10]: The description applies to FX100 and FX10 functions.
[PG]: The description applies to PRIMERGY functions.

### Path names of the commands

In the examples of the operations, the path names of the commands in the directory /bin, /usr/bin, /sbin or /usr/sbin might not be represented by absolute path.

### Symbols in this manual

This manual uses the following symbols.

**Note**

The Note symbol indicates an item requiring special care. Be sure to read these items.

**See**

The See symbol indicates the written reference source of detailed information.

**Information**

The Information symbol indicates a reference note related to Job Operation Software.

## Export Controls

Exportation/release of this document may require necessary procedures in accordance with the regulations of your resident country and/or US export control laws.

## Trademarks

- Linux is a registered trademark or trademark of Linus Torvalds in the U.S. and other countries.
- Other company and product names are trademarks or registered trademarks of their respective owners.

## Date of publication and Version

| Version | Manual Code |
|---|---|
| January 2016, 2nd Version | J2UL-1837-02ENZ0(00) |
| March 2015, Version 1.1 | J2UL-1837-01ENZ0(01) |

| Version | Manual Code |
|---|---|
| October 2014, 1st Version | J2UL-1837-01ENZ0(00) |



# Update history

| Changes | Location | Version |
|---|---|---|
| Added a note about node-exclusive jobs possibly being limited by the number of concurrently used nodes or the number of concurrently used CPU cores, according to a job operation setting. | 2.2.2 | 2nd Version |
| Added limit values (vnode=) for the number of virtual nodes by job to the display item "pjsub option parameters" of the pjacl command. | 2.2.2 | |
| Provided a method that uses the same format for the FX100/FX10 and PRIMERGY to specify an amount of memory per virtual node or CPU core and the number of CPU cores per virtual node.<br>In association with this, modified the descriptions and examples for the parameters vnode-core, core, vnode-mem, mem, and core-mem. | 2.3.2.1<br>2.3.2.2<br>2.3.2.3<br>2.3.4.1<br>2.3.4.5<br>A.1<br>F.7.1 | |
| Added a description about rough upper limits of the maximum virtual memory size and maximum stack size involved in executing a POSIX thread program. | 2.3.2.1 | |
| Clarified that the upper limit on the number of sub jobs handled in one bulk job is 65535. | 2.3.3.1 | |
| Added a description about how to batch submit multiple sub jobs for a step job.<br>Also added a description about how to submit a step job by specifying a job name. | 2.3.3.2 | |
| Improved descriptions about the PJM code INVALID HOSTFILE. | A.1<br>A.2 | |
| Added CPUTIME LIMIT EXCEEDED (Exceeded limit of the CPU time) to the REASON codes of job statistical information. | A.1<br>A.2 | |
| Clarified the cases where the value of job statistical information PERF COUNT *n* is 0 or cannot be obtained. | A.2 | |
| Added a note that the job will be aborted if the job swap couldn't complete within 60 minutes. | D.2.2 | |
| Added the following command messages:<br>* pjstat command : Message ID 0297<br>* pjacl command: Message ID 3295<br>* pjsub command: Message ID 0018 | F.4.1<br>F.4.2<br>F.5.1 | |
| Also fixed errata and modified descriptions. | - | |
| Added a clear description indicating that the individual sub jobs of a step job can be executed in different resource groups or resource units. | 1.2.1.3<br>2.3.3.2 | Version 1.1 |
| Provided support for non-contiguous mode as a method of FX100 or FX10 compute node allocation. | 1.2.5<br>1.5<br>1.7.2.1<br>2.2.1<br>2.2.2<br>2.3.2.2<br>2.3.2.5<br>2.3.3.5 | |

| Changes | Location | Version |
|---|---|---|
| | Appendix A<br>D.2.2 | |
| Added a clear description indicating that only sub jobs can be in the RUNNING-P or RUNNING-E state for a bulk job.<br>And added a description that the status of the step job is same as the status of running sub job or the status of the sub job to be executed next. | 1.3 | |
| Added PJM_APPNAME and PJM_FSNAME as environment variables that can be referenced in jobs. | 2.1.2 | |
| Added a description of how to skip the check for the stage-in target file when submitting a sub job of a step job. | 2.3.3.2 | |
| Added descriptions of the effects of node failures during job execution. | 2.7 | |
| Provided support for the -v 3 option of the pjshowrsc command that displays the nodes used by the job as a communication path. | 2.4.3 | |
| Added the code 27 of the job exit code which stands for the error in the resource management exit function | A.1<br>A.2 | |
| Added a clear description indicating that ELAPSE TIME (USE) in job statistical information is a time rounded up to an integral number of seconds. | A.2 | |
| The specification of the action for the code 3 of the job statistical information PRO EXIT CODE and EPI EXIT CODE was changed. For the code 3, the job is only held. | A.2 | |
| The name of the job statistical information RESERVE DATE was changed to SPECIFIED JOB START DATE. | A.2 | |
| Added job statistical information output items.<br>PERF COUNT 1 onwards in Tables A.3 and A.4 | A.2 | |
| Showed the formula to calculate performance information of the job from the job statistical information. | A.4 | |
| Modified the lists of job execution-related messages in notification by e-mail. | Appendix B | |
| Added a note that MPI programs of Intel MPI cannot be executed on the virtual node of the job using Multipurpose Daemon (MPD). | Appendix C | |
| Modified the link option for the cross compiler which enables the job swap on the user program. | D.2.1 | |
| Showed the relationship between job states and the possibility of operations on jobs. | Appendix E | |
| Added or changed the following messages of the pjshowrsc command:<br>- Added: Message ID 3226, 3228, 3227<br>- Changed: Message ID 3205 | F.4.3 | |
| Added or changed the following messages of the pjsub command:<br>- Added: Message ID 0056, 0068, 0089<br>- Changed: The message and description of Message ID 0080 was changed.<br>- Changed: Message IDs 0095 to 0100 were changed to 0062 to 0067 and their message text was modified. | F.5.1 | |
| Changed the following message of the pjdel command:<br>- Message ID 0170 | F.5.2 | |
| Changed the following message of the pjhold command:<br>- Message ID 0370 | F.5.3 | |
| Added or changed the following messages of the pjalter command:<br>- Added: Message ID 0502, 0572, 0589<br>- Changed: Message ID 0505, 0570, 0573<br>Modified the descriptions of the following messages:<br>- Message ID 0505, 0513, 0515, 0570, 0573 | F.5.7 | |

| Changes | Location | Version |
|---|---|---|
| Modified the description of the following message of the internal command plexec:<br>- Message ID 0015, 0022 | F.7.1 | |
| Also fixed errata and modified descriptions. | - | |

# Contents

# Chapter 1 Job Mechanism

The job operation software processes programs, created by users, in units called "jobs." This chapter describes jobs.

## 1.1 What Are Jobs?

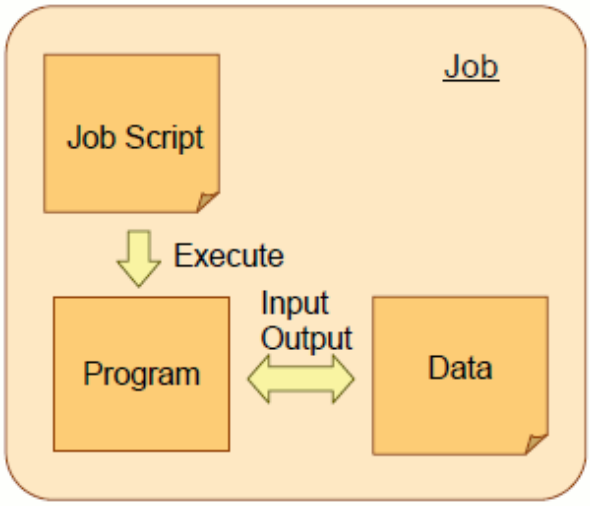
On the system where the job operation software is installed, users do not execute programs directly. Instead, they request the job operation management function of the job operation software to execute programs. Upon receiving a program execution request, the job operation management function reserves the computer resources required for executing the program, and executes it.

The unit of processing of such a program is a job. A job consists of a program, its input/output data, and the shell script with the written program execution procedure.

Table 1.1 Job components

| Component | Description |
|---|---|
| Program | A program is an executable program prepared by a user. An example is an execution file compiled by Technical Computing Language, which is related software. |
| Data | Data is program inputs and outputs. Examples include a data file that stores input parameters for the program, an output file that stores processing results, and program display results. |
| Job script | A job script is a shell script with the written program execution procedure. The job operation software executes a job based on this job script. |

Figure 1.1 Job configuration




The workflow for job execution is as follows.

1. The user prepares the files required for executing a job and places them on the login node.
2. The user requests the job operation management function to execute the job. This is called "job submission."

   For job submission, specify a job script in the pjsub command.
3. The job operation management function allocates computer resources to the job and executes the job script.
4. The function executes the program according to the contents of the job script, and outputs the execution results.
5. The user is notified by e-mail (only if the job submission includes an e-mail notification instruction) when the job script ends.
6. The user checks the job execution results on the login node.

See

The above description covers the general workflow for job execution. For details, see "Chapter 2 Job Operation Procedures."

# 1.2 Types of Job

Jobs are classified into several types, such as according to the required computer resource or the program type.

Jobs in the job operation software are classified into the following types.

Table 1.2 Types of job

| Classification | Type of job |
|---|---|
| Classification by job structure (job model) | Normal job |
| | Bulk job |
| | Step job |
| | Workflow job |
| Classification by job execution format (job type) | Batch job |
| | Interactive job |
| Classification by parallelism | Sequential job |
| | Parallel job |
| Classification by required computer resources (node resources) | Single node job |
| | Multinode job |
| Classification based on whether node is used exclusively or shared [FX100/FX10] | Node-exclusive job |
| | Node-sharing job |
| Classification by urgency [FX100] | Emergency job |
| | Non-emergency job |

Note

Unless otherwise noted, "node" in this manual means a compute node that executes a job.

The following sections describe each job classification and each type of job.

## 1.2.1 Job model

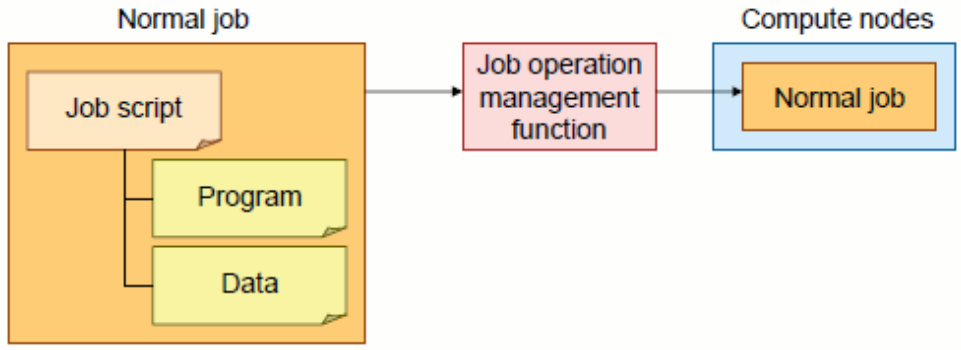
A classification by job structure is called a "job model." The following sections describe features of each job model.

### 1.2.1.1 Normal job

A normal job has the simplest structure.

One job script is executed for the job. The job ends when the job script ends.

Figure 1.2 Normal job




### 1.2.1.2 Bulk job

A bulk job consists of multiple instances of the same normal job submitted at the same time for execution.

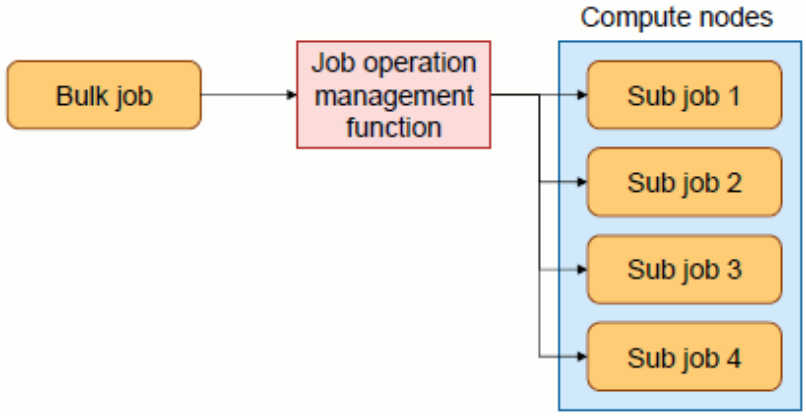
For example, suppose the user wants to change the job parameters and check the execution results for each change. The user would need to submit one normal job for each change. However, by using a bulk job, the user can submit multiple patterns at one time for one job.

The unit of processing of the job script executed at the same time in a bulk job is called a "sub job." Each sub job of a bulk job has a serial number (0 to 999999) set in the job. This number is called a "bulk number."

**Note**

Only a job that is batch and normal job can be a sub job of a bulk job.

Figure 1.3 Bulk job




### 1.2.1.3 Step job

A step job is a group of jobs that have an execution order or dependency relationship.

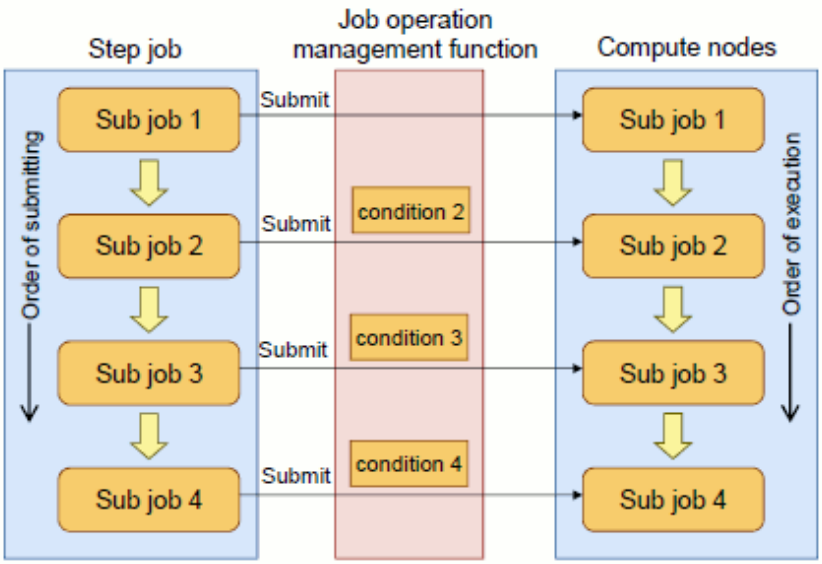
For example, suppose that whether a job is executed depends on the execution results of another job. If the job is a normal job, the user determines the job as such and submits the job. However, if the job is a step job, the user can specify the execution conditions and execution order for automatic processing according to the execution results of a specific job, when submitting the job.

Each job associated as a step job is called a "sub job." Each sub job of a step job has a serial number (0 to 65535) set in the job. This number is called a "step number."

**Note**

Only a job that is batch and normal job can be a sub job of a step job.

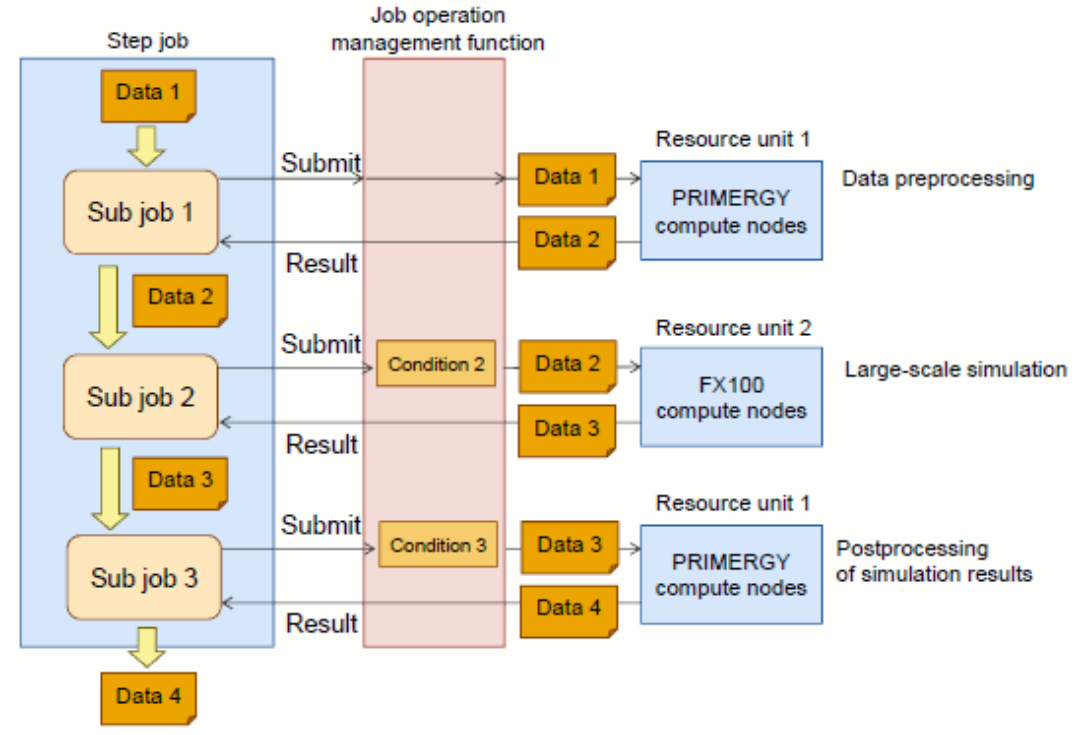
Figure 1.4 Step job




When submitting a step job, you can specify different resource units or groups for its sub jobs.
For example, suppose there is a cluster consisting of a resource unit of an FX100 compute node and a resource unit of a PRIMERGY compute node. You can submit applications, each of which is optimized for each architecture, as the sub jobs of a step job and execute them as a series of processes, as indicated below.

1. Sub job 1
   Preprocesses data for a large-scale simulation on a PRIMERGY compute node.
2. Sub job 2
   Takes the preprocessed data as its input and executes an application for a large-scale simulation on an FX100 compute node.
3. Sub job 3
   Takes the results of the large-scale simulation as its input and postprocesses the results (data analysis, visualization, etc.) on a PRIMERGY compute node.

Figure 1.5 Example of submitting individual sub jobs of a step job to different resource units



### 1.2.1.4 Workflow job

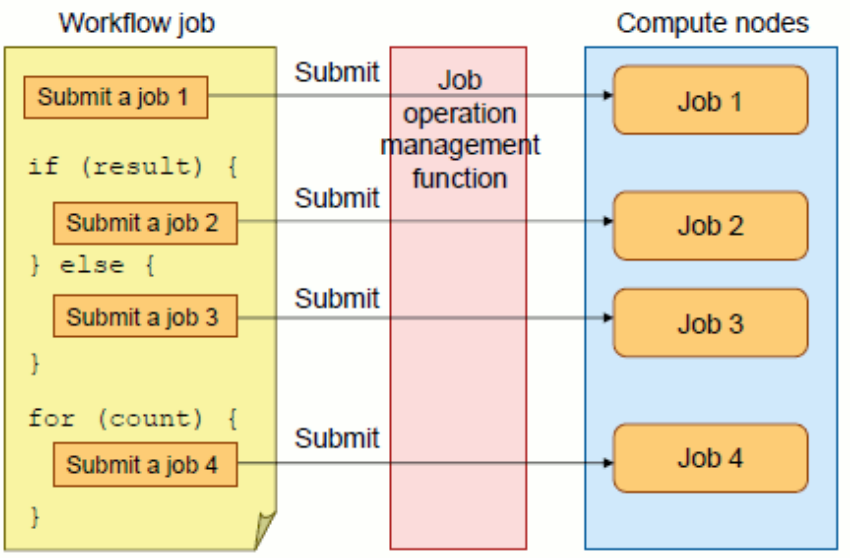
A workflow job is a group of jobs with the execution controlled (conditional branching and repetition) for each job.

Like with a step job, the next job to be executed depends on the execution results of another job. This execution control is more flexible than with a step job, depending on the contents of the shell script used to control execution.

**Information**

Specifically, a workflow job is not a type of job but a method using a shell script for user control over the submission of multiple jobs.

Figure 1.6 Workflow job




## 1.2.2 Batch job and interactive job

Jobs are classified into batch jobs and interactive jobs according to their execution format. They are called "job types."

Table 1.3 Batch job and interactive job

| Type of job | Description |
|---|---|
| Batch job | A job which is executed un-interactively.<br>The standard output and standard error of a job are output to files. |
| Interactive job | A job in which the interactive operation is possible for the program.<br>The standard input/output of a job is the standard input/output of the shell performing the job execution request operation. Interactive jobs are mainly used to debug jobs and programs. |

## 1.2.3 Sequential job and parallel job

Jobs are classified into sequential jobs and parallel jobs according to the program parallelism.

Table 1.4 Sequential job and parallel job

| Type of job | Description |
|---|---|
| Sequential job | A sequential job is a job for executing a sequential process (single thread, single process).<br>Example: Single-thread or single-process program |
| Parallel job | A parallel job is a job for executing a parallel processing program using multiple threads and/or multiple processes.<br>Parallel processing with multiple processes may be executed within a single node or across multiple nodes.<br>Example: Thread parallel processing program (automatic parallelization program, OpenMP |

| Type of job | Description |
|---|---|
| | program), process parallel processing program (MPI program), hybrid parallel processing program<br>A job that executes an MPI program may be called an "MPI job." |

### Information

Program parallelization in the job operation software is classified into the following types, which include thread parallel processing and process parallel processing.

Table 1.5 Types of program parallelization

| Parallelization type | Description |
|---|---|
| Thread parallel processing | Thread parallel processing is a method to divide one process into multiple threads and execute them in parallel on multiple CPUs (or multiple cores on a multicore CPU).<br>Since each thread shares the memory space of the process as it runs, programming is relatively easy. However, the upper limit on CPUs used is the number of CPUs in the node, so a significant performance improvement cannot be expected.<br>Automatic parallelization by a compiler and OpenMP are examples of thread parallel processing. For details, see the manual provided with Technical Computing Language. |
| Process parallel processing | Process parallel processing is a parallel processing method using multiple processes. The required data for parallel processing is transferred in communication between processes. Since this method can execute individual processes on different nodes, a significant performance improvement can be expected from an increase in the number of nodes. However, it requires advanced programming techniques.<br>Each of multiple processes for one node may run on a single thread. This is called flat parallel processing.<br>An MPI program that uses MPI (message passing interface) is an example of process parallel processing. For details on MPI, see the manual provided with Technical Computing Language. |
| Hybrid parallel processing | Hybrid parallel processing is a method using both thread parallel processing and process parallel processing. |

## 1.2.4 Single node job and multinode job

Jobs are classified into single node jobs and multinode jobs according to the required number of nodes.

Table 1.6 Single node job and multinode job

| Type of job | Description |
|---|---|
| Single node job | A single node job is a job that requires only one node for execution.<br>A sequential job is a single node job.<br>An example for a parallel job is one that uses multiple processes or threads within a node. |
| Multinode job | A multinode job is a job that requires multiple nodes for execution.<br>An example is a process parallel processing program that spans nodes or a hybrid parallel processing program. |

### Information

If more than one node is allocated to a single node job, the job operation software processes the job as a multinode job.

The following table lists the specific expressions in this manual for classifications by job parallelism and classifications by the required number of nodes.

Table 1.7 Single node jobs and multinode jobs

| Required number of nodes | Parallelization technique | Job name |
|---|---|---|
| 1 node | Sequential job | Single node (sequential) job |
| | Thread parallel processing | Single node (thread parallel processing) job |
| | Process parallel processing | Single node (process parallel processing) job |
| | Hybrid parallel processing | Single node (hybrid parallel processing) job |
| Multiple nodes | Process parallel processing | Multinode (process parallel processing) job |
| | Hybrid parallel processing | Multinode (hybrid parallel processing) job |

## 1.2.5 Node-exclusive jobs and node-sharing jobs [FX100/FX10]

Jobs submitted to an FX100 or FX10 compute node are classified into two types based on whether the job uses the node exclusively or shares the node with other jobs.

Table 1.8 Node-exclusive jobs and node-sharing jobs

| Type of job | Description |
|---|---|
| Node-exclusive job | Computer resources are allocated to the job in units of nodes for use exclusively by this single job.<br>There are three allocation methods: mesh mode, non-contiguous mode, and torus mode. The modes have different minimum units: one node for mesh mode and non-contiguous mode, and one Tofu (12 nodes) for torus mode.<br>Mesh mode and non-contiguous mode require resource allocation to fewer nodes than torus mode. Therefore, job execution in mesh mode may start earlier than in torus mode.<br>However, in non-contiguous mode, a communicating job may interfere with another communicating job. If communication performance is a primary consideration, use torus mode or mesh mode. |
| Node-sharing job | Computer resources are allocated to the job in units called virtual nodes. A virtual node is a set of CPU cores and memory.<br>The node may have more than one job because the CPU cores and memory within one node are shared among multiple jobs |

## 1.2.6 Emergency Job [FX100]

A job that must be executed with the highest priority during operation is called an "emergency job."

For example, if you want to execute a specific job at a predetermined time every day, you need an operation to make resources available beforehand. However, in situations where jobs are constantly being executed to keep system utilization high, a scheduled job may not be executed because the necessary resources are not available at a scheduled time.

In such a case, a specific job can be executed with priority higher than other jobs by submitting it as an emergency job.

The Job Operation Software handles an emergency job as follows:

- Among the jobs waiting for execution, the emergency job is placed first in the execution order.
- If there are insufficient resources to execute the emergency job because of the jobs currently executed, those jobs are stopped by being canceled or by job swap to allocate the resources to the emergency job.

### Information

- About job swap

  Job swap is a function that stops a running job while maintaining its state to temporarily release resources (CPU or memory). When the stopped job is restarted, it continues from the state in which it was stopped.
  The operation that stops a job and releases resources is called "swap-out"; the operation that allocates the resources again and restarts that job is called "swap-in." The period between the start of swap-out and the completion of swap-in is called the "swap period."

Whether abort or job swap is used to stop a job is determined automatically based on the job type and job state.
Also, administrators may explicitly swap out a specific job.

- About emergency jobs in FX10 and PRIMERGY compute nodes

  The execution of emergency jobs that use job swaps is supported only for FX100 compute nodes.
  Depending on the system, having the administrator control the allocation of computer resources and execution of jobs can serve as mechanisms for achieving the execution of jobs with a high degree of emergency. Since this is achieved by the operation method, for details, make an inquiry of the administrator.
  The "emergency jobs" described in this manual are those for FX100 compute nodes.

## 1.3 Job States

The following table lists the changing states of jobs, from job execution request to job end.

Table 1.9 Job states

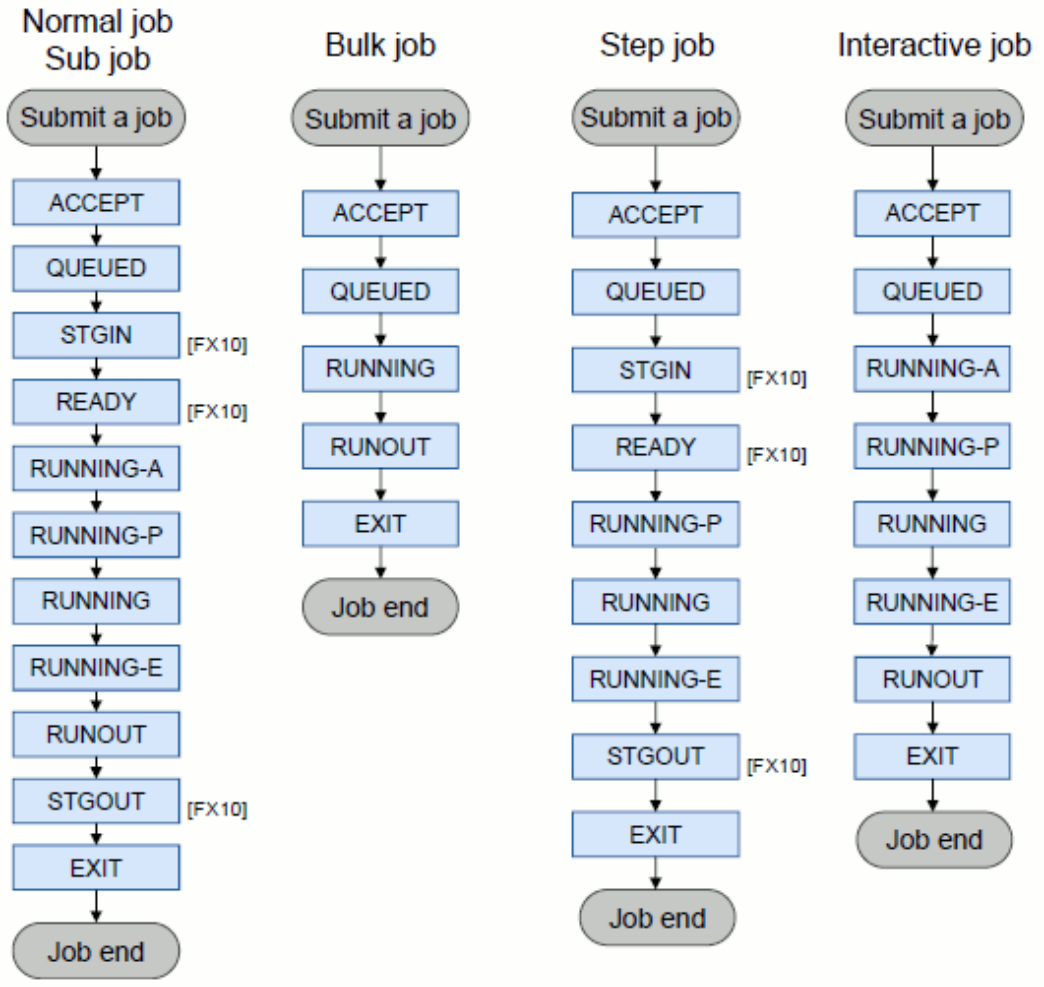
| State name | Description |
|---|---|
| ACCEPT | Checking whether the job satisfies the acceptance conditions (items restricted with the job ACL function) |
| QUEUED | Waiting for a turn to execute the job, which has been accepted |
| STGIN [FX10] | Waiting for completion of the stage-in process, which has started |
| READY [FX10] | Waiting to execute the job, with the stage-in process already completed |
| RUNNING-A | Reserving the resources required for job execution |
| RUNNING-P | Executing the prologue process |
| RUNNING | Executing the job |
| RUNNING-E | Executing the epilogue process |
| RUNOUT | Waiting for the stage-out process, following the end of job execution.<br>If staging is not used, job end processing is in progress. |
| STGOUT [FX10] | Waiting for completion of the stage-out process, which has started |
| EXIT | Job end |
| REJECT | Job acceptance rejected |
| CANCEL | Job stopped by an instruction from the job submitter or the administrator |
| HOLD | Keeping the state of the submitted job from changing, when job execution has stopped |
| ERROR | Job stopped because of an error detected by the job operation management function |
| SWAPOUT [FX100] | During swap-out processing |
| SWAPPED [FX100] | Swap-out completed |
| SWAPIN [FX100] | During swap-in processing |

(Note 1) For descriptions of "staging," "stage-in," and "stage-out," see "1.5 Job Staging [FX10]."

(Note 2) For the detail of "prologue" and "epilogue", see "1.11 Prologue and Epilogue Function."

The basic state transitions of jobs depend on the type of job as shown below.

Figure 1.7 Basic state transitions of jobs




## Note

- The bulk job and step job can have a state as one job and a state as a subjob. The "bulk job state" or "step job state" means the former. The bulk job and state job state transitions as well as their subjob state transitions differ as shown in the figure above.
- The step job and the running sub job have the same state (the next sub job, if there are no running sub job).
  However, if the sub job state is RUNNING-A, the step job state is QUEUED or READY. Also, if the sub job state is RUNOUT, the step job state is RUNNING or RUNNING-E.

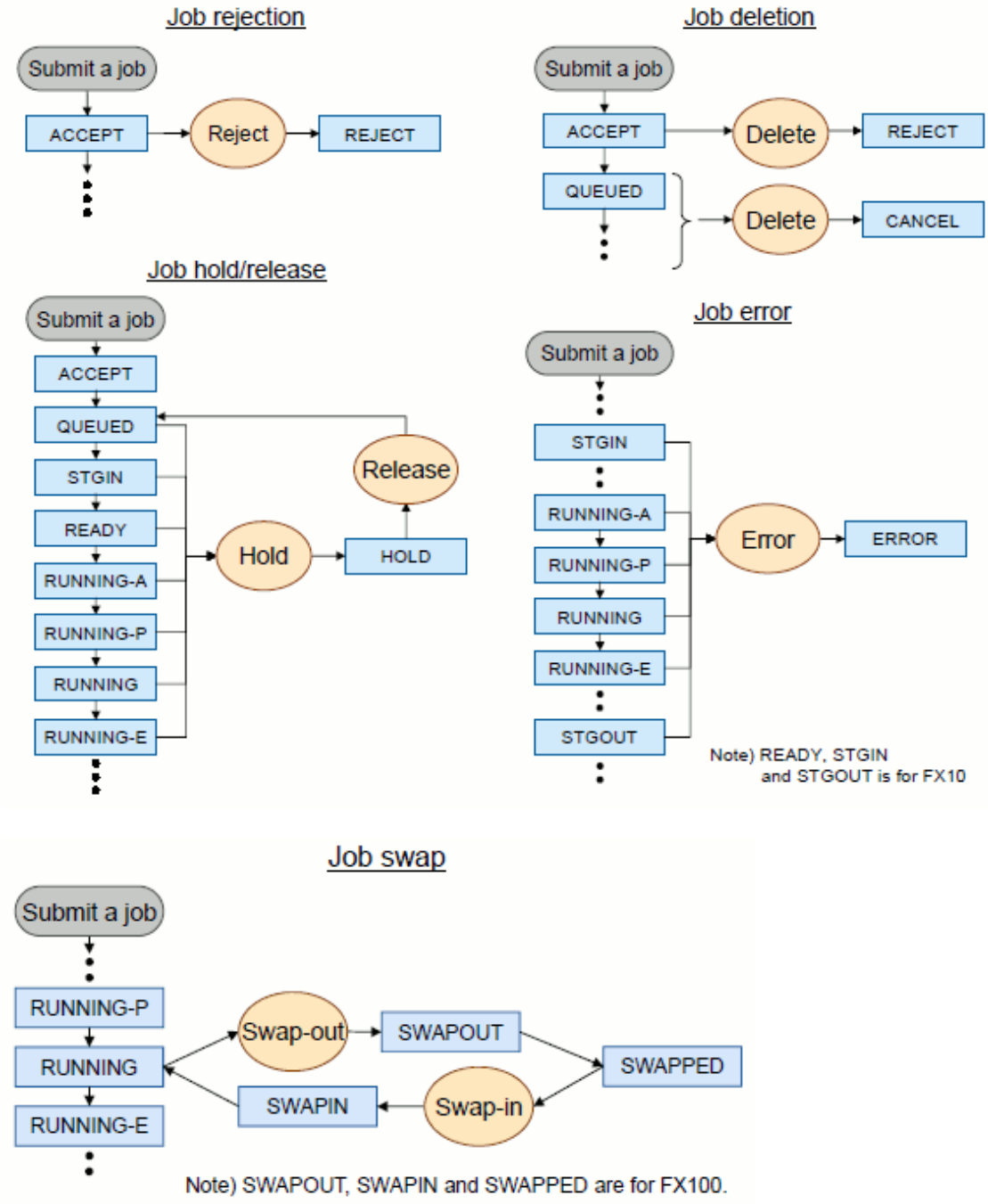
Job state transitions also depend on the events occurring after job submission, as described below.

Table 1.10 Events and job state transitions

| Event | Job state transition |
|---|---|
| Job rejection | The job submitted by a user enters the ACCEPT state, indicating tentative acceptance. If the job does not satisfy the acceptance conditions (job ACL), it enters the REJECT state, indicating job rejection.<br>For details on the job ACL, see "1.9 Job ACL Function." |
| Job deletion | The operation to cancel a submitted job is called "job deletion." When a job in the ACCEPT state is deleted, it enters the REJECT state. When a job in another state is deleted, it enters the CANCEL state. If the job is running, it is stopped.<br>For details on the job deletion operation, see "2.5.1 Deleting a job." |
| Job hold and release | The operation to stop job execution but not change the job state is called "job hold." A job in the QUEUED, STGIN, READY, RUNNING-A, RUNNING-P, RUNNING, or RUNNING-E state can be held, entering the |

| Event | Job state transition |
|---|---|
| | HOLD state as a result. Canceling the HOLD state restores the QUEUED state.<br>For details on the job hold and release operations, see "2.5.3 Holding a job and canceling the hold." |
| Job error | If an error occurs in the job operation management function for a job between the STGIN and STGOUT states, the job enters the ERROR state. |
| Job swap | When a swap-out occurs for a job in the RUNNING state, the job state becomes SWAPOUT. Once the swap-out processing completes, the state becomes SWAPPED.<br>When the swap-in processing of a job that has been swapped (SWAPPED) starts, the job state becomes SWAPIN.<br>When the swap-in processing completes, the state becomes RUNNING, and execution of the job continues. |

Figure 1.8 State transitions due to job rejection, error occurrence, etc.



## 1.4 Job ID and Sub Job ID

Each job is automatically assigned a "job ID," which is a unique identification number in the system. A job ID is a value ranging from 0 to 2147483647.

Among jobs, each bulk job and step job is also assigned a "sub job ID," which is a unique number identifying a sub job. A sub job ID is a character string consisting of a bulk number or step number added to a job ID. A bulk number is a value ranging from 0 to 999999, a step number is from 0 to 65535.

The sub job ID of a bulk job is written as "*job ID*[*bulk number*]". The sub job ID of a step job is written as "*job ID*_*step number*".

```
[Example]
Job ID                       : 123456
Sub job ID of bulk job   : 123456[1234]      Job ID 123456 and bulk number 1234 of sub job
Sub job ID of step job   : 123456_1234       Job ID 123456 and step number 1234 of sub job
```

## 1.5 Job Staging [FX10]

Compute nodes that execute programs must be able to reference the programs and data created by users and placed on the login node.

The two methods of doing so are to use a shared file system and to place the file on the local file system of each node.

The use of a shared file system in an environment for executing numerous jobs may lead to input/output conflicts between individual jobs and a deterioration in job input/output performance. In such cases, set the local file system of each compute node as the file system used by jobs to reduce input/output conflicts.

However, to use the local file system of each compute node, it is necessary to transfer the programs and input/output data required for executing jobs between the login node and the compute node before and after a job is executed. This is called "staging." Furthermore, data transfer from the login node to the local disk of a compute node is called "stage-in," and transfer of execution results from the local disk of a compute node to the login node is called "stage-out."

The job operation management function supports a function for automatic staging at the job start and end times.

To use this staging function, the user must place the target file at a specific location on the login node and write instructions in the job script about the staging target. The place is different depending on the composition of the system.

For details on how to write instructions about the staging target file in a job script, see "2.1.1 How to create a job script."

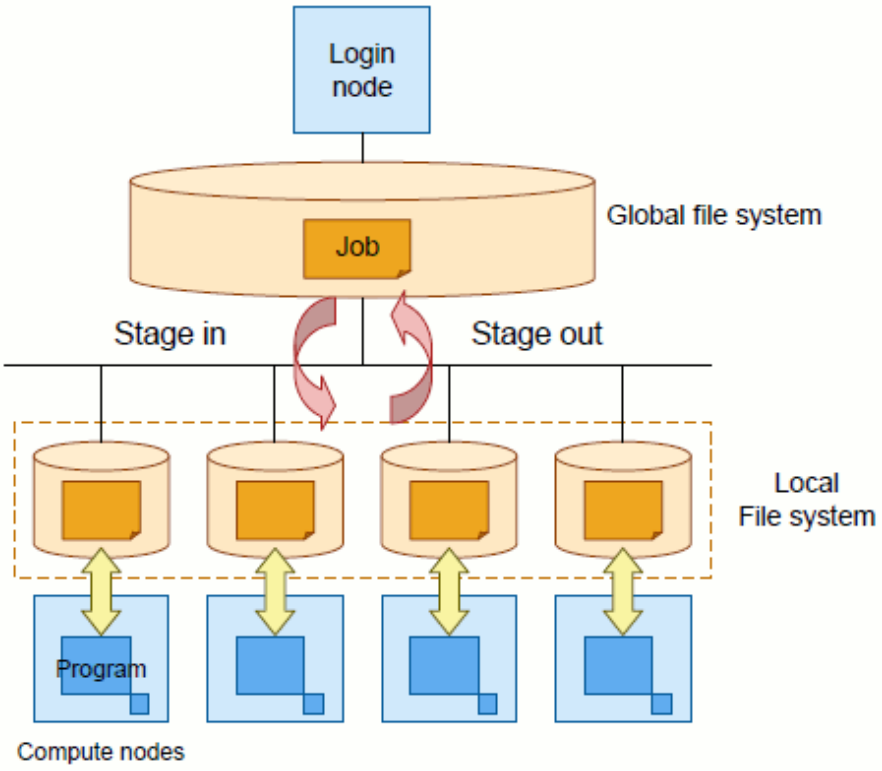
### Note

- The staging function can be used for all job models of the batch job. And, all job models can be used for the batch job.
  Note, however, that the staging function cannot be used if mesh mode or non-contiguous mode (see "1.7.2.1 Node allocation in units of nodes or Tofus") is used as the method of FX10 compute node allocation or if the job type is node-sharing job.
- The job operation management function schedules staging of multiple jobs and processes them in order. Therefore, staging of a job may not start immediately after the job is submitted.

### Information

In the system that uses the staging function as understood from the purpose of the staging function, the job cannot usually be accessed to the global file system by the compute node.
The end-user must specify the option at job submission to access the global file system in the job when the staging function is used. For details, see "2.3.2.5 Specifying staging [FX10]."

Figure 1.9 Job staging




# 1.6 Job Output

The standard output and standard error output of a job are output as separate files to the current directory at the job submission time.

For the file names, see "2.3.2.9 Specifying the standard output and standard error output files of a batch job."

# 1.7 Node Resource Allocation

To execute a job, the node resource (CPU cores and memory) must be allocated for executing the job.

Nodes in the system are grouped hierarchically in resource units. A resource unit is a unit for job operations in a cluster. A cluster is an operational unit of the system. Furthermore, a resource unit is divided into units for resource allocation, which are called resource groups. Multiple resource groups are prepared in accordance with the job operation policy. (For details, see the First Step Guide.)

A job is executed in a resource group. If the administrator has not set the default resource group with the job ACL function, it is necessary to specify a resource group at the time of job submission. The job process monopolizes the allocated CPU core. CPU core binding can be controlled by using the Technical Computing Language.

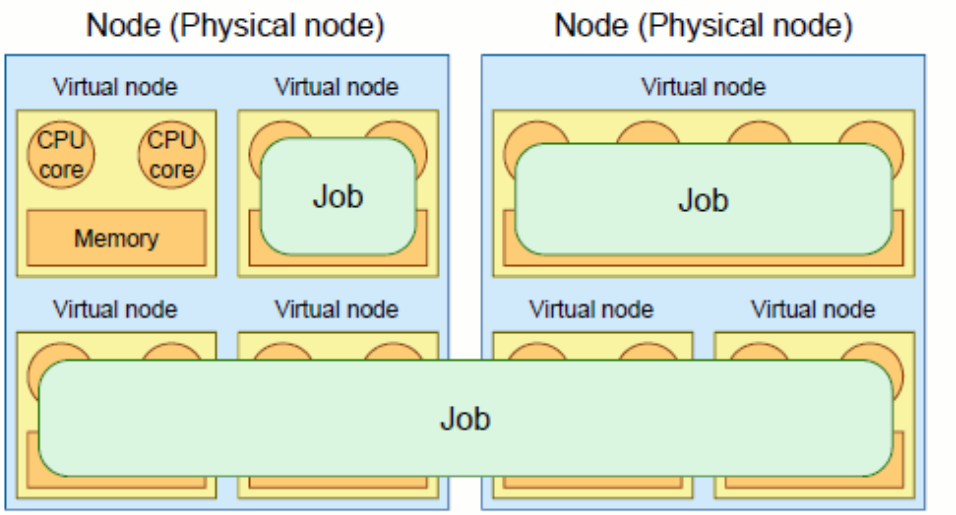
When submitting a job, the user specifies how to allocate the node resource in the resource group. Since the FX100 or FX10, and PRIMERGY have different architectures, their concepts on how to allocate the node resource are also different.

The following sections describe the node resource allocation concepts for the respective compute nodes of the FX100 or FX10, and PRIMERGY.

## 1.7.1 Nodes and virtual nodes

There are two types of nodes according to the concept of node resources managed by the job operation software: physical nodes and virtual nodes, each of which is a set of CPU cores and memory.
The user defines virtual nodes by selecting a quantity of resources from the available physical nodes to satisfy their requirements. Use of virtual nodes helps prevent system node resources from being wasted, thus improving system throughput and utilization.

Figure 1.10 Virtual nodes




## Information

The amount of resources (the number of cores and amount of memory) per virtual node must be within the range of one physical node. You cannot combine multiple physical nodes into one virtual node.

For the FX100 and FX10 compute node, you can select whether to allocate resources in units of physical nodes or virtual nodes. For a PRIMERGY compute node, resources are always allocated in units of virtual nodes.

## 1.7.2 Node allocation on FX100 and FX10 compute nodes

For the FX100 and FX10 compute node, the following two types of node allocation are available:

- Allocation in units of nodes or Tofus (for node-exclusive jobs)
- Virtual node allocation (for node-sharing jobs)

The following sections describe each type of allocation.

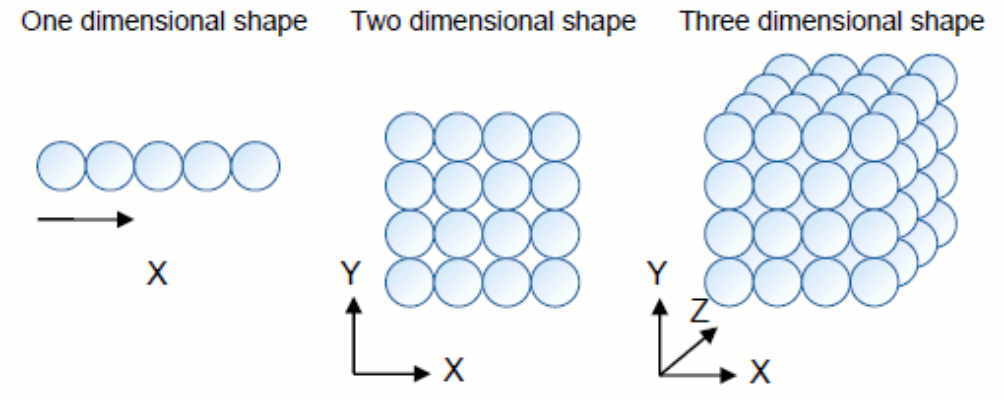
### 1.7.2.1 Node allocation in units of nodes or Tofus

You can specify the following for node allocation to node-exclusive jobs in units of nodes or Tofus:

- Node shape and number of nodes to allocate
- Node allocation rules for executing an MPI program

Specify the node to allocate as a shape placed in a one-dimensional, two-dimensional, or three-dimensional virtual space.

Figure 1.11 Node shapes (conceptual image)




There are three node allocation methods: torus mode, mesh mode, and non-contiguous mode.

Table 1.11 Node allocation methods

| Allocation mode | Description |
|---|---|
| Torus mode | The minimum node allocation unit is one Tofu (12 nodes).<br>Nodes selected for allocation are mutually adjacent in the Tofu coordinate space. |
| Mesh mode | The minimum node allocation unit is one node.<br>Nodes selected for allocation are mutually adjacent in the Tofu coordinate space. |
| Non-contiguous mode | The minimum node allocation unit is one node.<br>Nodes are selected for allocation so that they are mutually adjacent in the Tofu coordinate space as much as possible.<br>In the following cases, the selected nodes are not mutually adjacent:<br>- There are no free nodes that are mutually adjacent.<br>- Selecting nodes that are not mutually adjacent speeds up job execution start. |

The characteristics of these allocation methods are as follows:

- Job communication performance
  In terms of job communication performance, torus mode is advantageous.
  In either torus mode or mesh mode, communication of one job does not interfere with communication of another job. Note, however, that in mesh mode, communication performance may not always be constant because the lengths of inter-node communication paths may vary depending on the node allocation result.
  In non-contiguous mode, communication of one job is likely to interfere with communication of another job because the allocated nodes are not always mutually adjacent.

- Ease of node allocation
  In terms of ease of node allocation, mesh mode and non-contiguous mode are advantageous because the minimum node allocation unit is one node.
  In other words, job execution may start earlier in mesh mode or non-contiguous mode than in torus mode.
  In addition, in mesh mode, the shape of the allocated nodes needs to fulfill certain requirements, although allocation is made in node units (see "Table 2.5 Size of nodes that can be allocated [FX100/FX10]" in "2.2.1 Checking resource units and resource groups").
  Therefore, node allocation is easier in non-contiguous mode than in mesh mode because the former mode places no restrictions on the node shape.

- Tolerance to communication path failure
  In terms of the tolerance to communication path failure, torus mode and mesh mode are advantageous.
  In non-contiguous mode, nodes are allocated so that they are mutually adjacent as much as possible, but they may not always be mutually adjacent. As a result, the Tofu interconnect on a node that is not allocated to a job may be used as a communication path for multiple jobs. Therefore, if the Tofu interconnect on the node used as a communication path becomes faulty, the multiple jobs are likely to be affected by this fault.

## Note

- When a single node job with a one-dimensional shape is submitted in Torus mode, the allocated node is one node. In other words, a number of single node jobs with a one-dimensional shape can be executed in one Tofu node unit.
  Depending on the setting of the resource group to be executed by the administrator, a multinode job with up to 11 nodes allocated can undergo multiple executions in one Tofu unit.
  In Torus mode, when a single node job is submitted with a two-dimensional shape (1 x 1) or a three-dimensional shape (1 x 1 x 1), the allocated nodes are rounded up in Tofu units.
- If you specify nodes with a two-dimensional shape in mesh mode, the allocated node shape is rounded up to the minimum unit of 3 x 1.
- Suppose that a job in torus or mesh mode and a job in non-contiguous mode are mixed when executed within a resource unit. The job in non-contiguous mode may degrade the communication performance of the job in torus or mesh mode.
  Ask the administrator whether such jobs, including other users' jobs, can be mixed when executed within a resource unit.

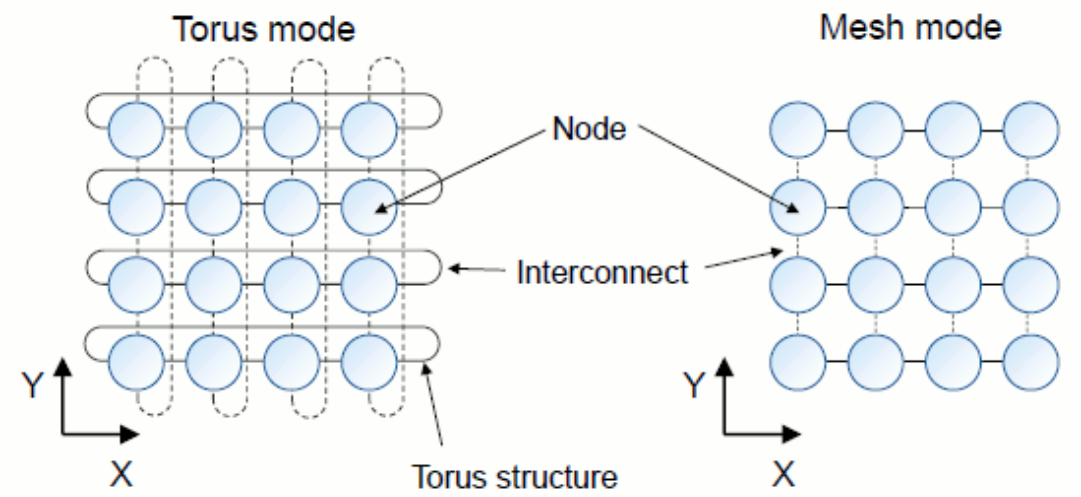
## See

For details on how to specify the nodes to allocate, see "2.3 Submitting a Job."

In torus mode, an interconnect that connects nodes consists of torus, each of which is along one axis. A torus is one of the shapes (topologies) for connecting nodes. It is a loop-shaped structure of nodes connected by interconnects. Each node at both ends is connected as the neighboring nodes along one axis of the node shape.

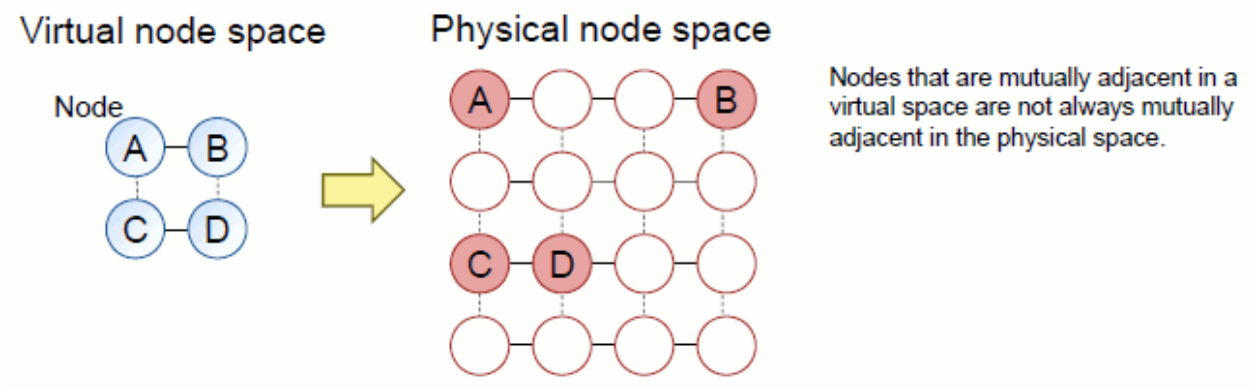
- Torus along the X-axis for a one-dimensional shape
- Torus along the X-axis and Y-axis for a two-dimensional shape
- Torus along the X-axis, Y-axis, and Z-axis for a three-dimensional shape

Figure 1.12 Connection between nodes (example of a two-dimensional shape)




In torus or mesh mode, nodes are allocated so that they will be mutually adjacent in the physical space. In contrast, in non-contiguous mode, allocated nodes are not guaranteed to be mutually adjacent in the physical space.
In other words, in non-contiguous mode, the inter-node distances in the virtual node space assumed for a job may not match the inter-node distances in the physical space.

Figure 1.13 Image of node allocation in non-contiguous mode




In torus mode and mesh mode, the maximum dimensions that can be specified for a node shape depend on the physical node configuration. The administrator specifies the maximum dimensions for each resource unit and resource group, which are units for job operations.

If the specified node shape exceeds the maximum dimensions, node resources become insufficient, and jobs are not executed. Users are requested to check the maximum dimensions for a node shape before submitting their jobs. For details on how to check it, see "2.2.1 Checking resource units and resource groups."

In torus mode and mesh mode, if a two-dimensional or three-dimensional node shape is specified and does not fit within the maximum dimensions as is, it may fit when rotated. If so, it is automatically adjusted in that way.
For example, suppose the maximum node dimensions are X x Y x Z = 2 x 3 x 32. You can specify a node shape of 6 x 3 x 2 even though it does not fit as specified. This is because the node shape fits within the maximum dimensions of 2 x 3 x 32 when rotated to 2 x 3 x 6.
In contrast, a node shape of 4 x 4 x 4 does not fit within the maximum dimensions of 2 x 3 x 32 no matter how it is rotated. For this shape, node resources would be insufficient, and the job would not be executed.

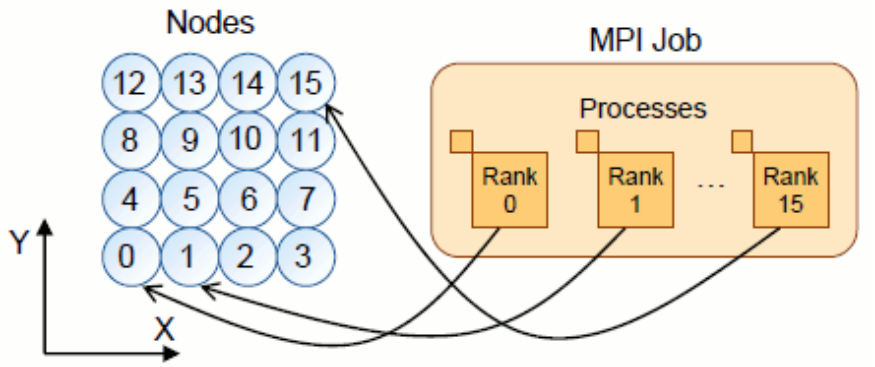
The automatic rotation of the node shape can be controlled. For details, see "2.3.2.2 Specifying a node resource."

It is necessary to also consider the following when determining the shape of a node group and the number of nodes.

- The node shape affects the cost of communication between nodes in a parallel program.
- An MPI program in the MPMD (multiple program multiple data) model handles the processing between multiple, different programs. It is necessary to consider the number of nodes required by each of these programs.

In MPI, a number called "rank" for process management is assigned to each process in the order of generation. You can specify which node is allocated to which rank in the job operation software, when submitting a job.

Figure 1.14 Example of ranks and node allocation




Users allocate nodes by writing either a rule or an instruction for one-to-one correspondence between ranks and nodes. For details, see "2.3.5 Submitting an MPI job."

For MPI jobs in torus mode and mesh mode, it is better to use a two-dimensional or three-dimensional shape than a one-dimensional shape to assign ranks so that the nodes are adjacent to one another. The communication distance between ranks is shorter, and job performance can thus be expected to improve.

### Note

In torus mode and mesh mode, job execution with a specified two-dimensional or three-dimensional shape may start with later allocation of resources than with a specified one-dimensional shape.

### 1.7.2.2 Allocation in units of virtual nodes

You can specify the following for node allocation to node-sharing jobs in units of virtual nodes:

- Number of virtual nodes to allocate
- Number of CPU cores and amount of memory per virtual node
- Amount of memory per CPU core

### Note

At present, only one virtual node can be allocated to the FX100 and FX10 compute node.

### Information

For details on how to specify the virtual nodes to allocate, see "2.3 Submitting a Job."

### 1.7.2.3 NUMA allocation policy [FX100][PG]

For NUMA architecture compute nodes, the cost of accessing memory shared among multiple CPU cores is not uniform. Depending on the CPU core allocated to the job, this may cause variation in or deterioration of the execution performance. Therefore, the administrator can select the job allocation rules (NUMA allocation policy) for CPU core groups that have the same cost of accessing memory.

Table 1.12 NUMA allocation policy

| Policy | Description |
|---|---|
| pack | The job is allocated so that it fits in one NUMA node as much as possible. (default) |
| unpack | The job is allocated so that it spans NUMA nodes. |

End users cannot select the NUMA allocation policy. However, end users can confirm which policy is set for the system in use by checking the setting contents of the job ACL function (see "1.9 Job ACL Function").

## 1.7.3 Virtual node allocation on PRIMERGY compute nodes

On PRIMERGY compute nodes, you can specify the following for virtual node allocation:

- Number of virtual nodes to allocate
- Number of CPU cores and amount of memory per virtual node
- Node allocation rules

### Information

Unlike FX100 or FX10 compute nodes, PRIMERGY compute nodes have no concept of shape for allocated nodes.

### 1.7.3.1 Node allocation concepts

Both nodes and virtual nodes must be considerations in node allocation on PRIMERGY compute nodes.

The following table lists node allocation concepts.

Table 1.13 Node allocation concepts

| Method | | Description |
|---|---|---|
| Relationship between nodes and virtual nodes | Virtual node placement policy | Concept of placement of virtual nodes as opposed to nodes |
| | Rank map | Concept of setting virtual node IDs for the virtual nodes placed at nodes |
| Relationship between jobs and nodes | Node selection method | Concept of selecting the nodes used by jobs |
| | Priority control of allocated nodes | Concept of selecting nodes according to their own set priority |
| Relationship with other jobs | Execution mode policy | Concept related to occupation of nodes |

This section describes these concepts.

### Virtual node placement policy

A virtual node placement policy indicates how to allocate virtual nodes, for selected nodes.
The following table lists the policies.

Table 1.14 Types of virtual node placement policy

| Method | Description |
|---|---|
| Absolutely PACK | All the virtual nodes are placed on one node. If they cannot be placed there, the job is not executed. |
| PACK | Virtual nodes are placed on as few nodes as possible. |
| Absolutely UNPACK | Only one virtual node is placed on one node. If there are more virtual nodes than nodes, the job is not executed. |
| UNPACK | If at all possible, only one virtual node is placed on one node. If there are more virtual nodes than nodes which are allocated to a job, multiple virtual nodes are placed on one node. |

### Rank map

A rank map describes rules on setting virtual node IDs for the virtual nodes placed in accordance with the virtual node placement policy. This corresponds to the specification of rank in MPI.
The following table lists the two methods for rank maps.

Table 1.15 Types of allocation methods with a rank map

| Method | Description |
|---|---|
| rank-map-bynode | Virtual node IDs are set one by one according to the node ID order of a node (round robin method). |
| rank-map-bychip | Virtual node IDs are set sequentially starting with the virtual nodes within the same node. |

### Node selection method

The node selection method specifies whether to distribute the allocation of nodes to jobs or concentrate allocation to a few nodes. However, users cannot specify the node selection method. The contents set by the administrator with the job ACL function are applied.

For distribution of nodes that are to be selected, the nodes not being used (nodes with many available cores) have priority for selection. For concentration on a few nodes, a selection is made from the nodes already being used (nodes with fewer cores left available).

### Priority control of allocated nodes

In the priority control of allocated nodes, the system selects nodes starting with that with higher priority according to the allocation priority set for each node by the administrator.

### Execution mode policy

An execution mode policy is a method of specification taking into regard the occupation of nodes by jobs. The following table lists methods called execution mode policies.

Table 1.16 Types of execution mode policy

| Method | Description |
|---|---|
| SIMPLEX | One job occupies a node.<br>The node where the SIMPLEX specified job is running is not allocated to other jobs even if it has available CPU cores. |
| SHARE | A node is shared with other jobs that have the SHARE specification. |

See

For specific methods of specification and illustrations of allocation based on each node allocation concept, see "2.3.2.2 Specifying a node resource."

# 1.8 Job Execution Order

Submitted jobs are evaluated based on several elements to determine the execution order. The elements and criteria for determining the execution order are specified by the administrator. This set of elements and criteria is called the job selection policy.

Table 1.17 Elements that determine the job execution order

| Element | Description |
|---|---|
| Privileged jobs | Emergency jobs and jobs for re-execution are assigned higher execution priority.<br>An emergency job submitted when compute resources are insufficient is executed after one or more running jobs are stopped. |
| Submission time | Jobs are executed in the order in which they were submitted. This is called fcfs (first-come first-serve). |
| Number of requested nodes [FX100/FX10] | A job is evaluated by the number of nodes requested by it. |
| Group priority and user priority | Jobs belonging to a group or user are evaluated by the job priority set for the group or user. |
| Resource group priority | A job is evaluated based on the priority assigned to its resource group in the resource unit. |
| Priority of jobs of the same user | Multiple jobs of the same user are evaluated based on the priority assigned to them. |
| Job priority in a resource unit | A job is evaluated based on its priority in the resource unit. |
| Priority based on job execution results | Jobs are evaluated based on the past job execution results so that users can use the system on an equal basis.<br>This is called the "fair share function." |
| Elapsed time limit value | A job is evaluated by its elapsed time limit value. |
| Elapsed time limit value x number of requested nodes [FX100/FX10] | A job is evaluated by the product of its elapsed time limit value and the number of nodes requested by it. |
| Requested disk capacity [FX10] | A job is evaluated based on the disk capacity requested by it for the staging function. |
| Stage-in file size [FX10] | A job is evaluated based on its stage-in target file size. |
| Specification of execution start time | A job is evaluated based on whether its job execution start time is specified at the time of job submission. |
| Specification as an interactive job | A job is evaluated based on whether it is submitted as an interactive job. |

Note

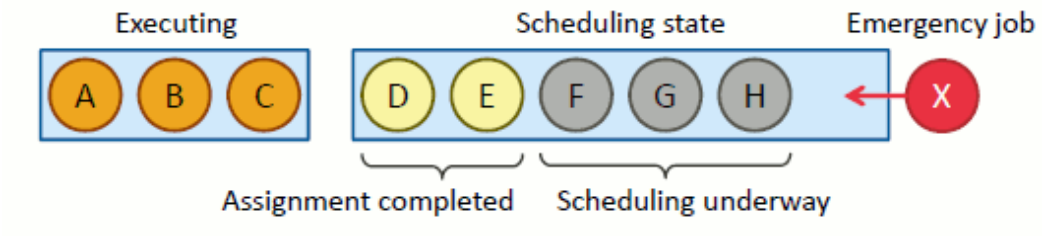
The job execution order is determined based on the above elements. However, if there are free resources available, some jobs may be executed earlier than in the normal execution order. This function is called the "backfill function." The administrator sets whether this function is to be enabled or disabled.

The following sections describe details of the elements which determine the job priorities.

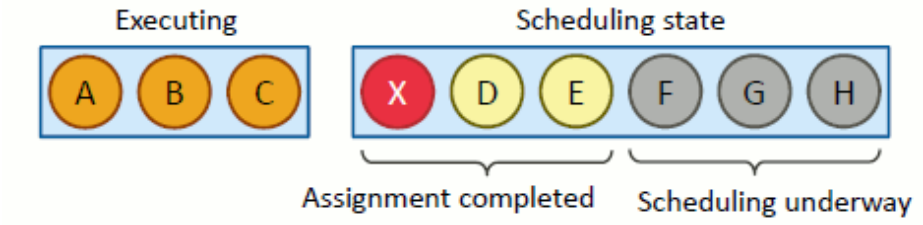
## 1.8.1 Emergency job

Emergency jobs have the highest priority in processing, which stops running jobs to release resources.

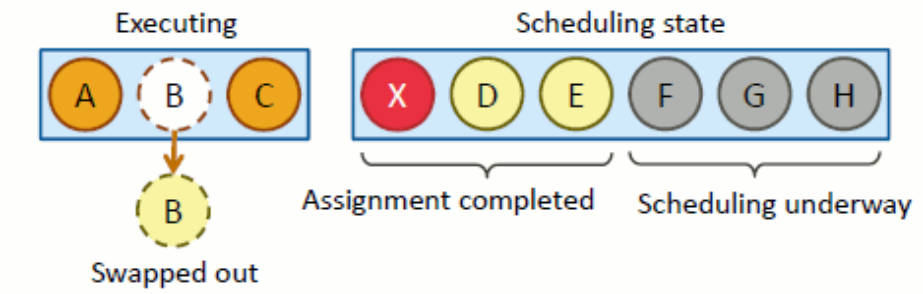
1. An emergency job is submitted.




2. The emergency job is assigned the highest priority.

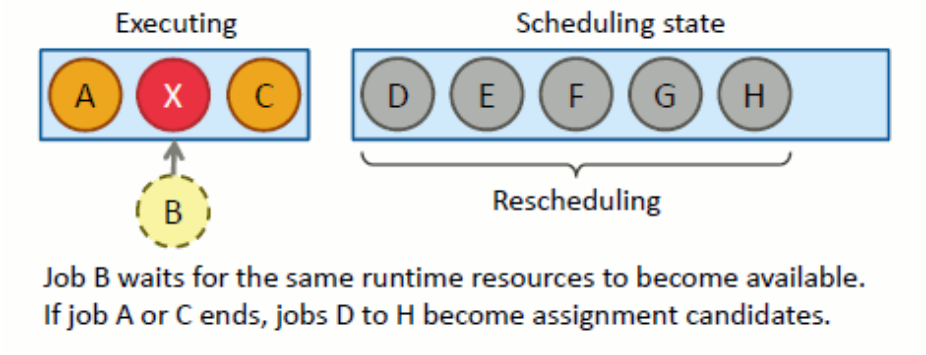



3. If the resources are not available because of other jobs currently executed, those jobs are stopped.

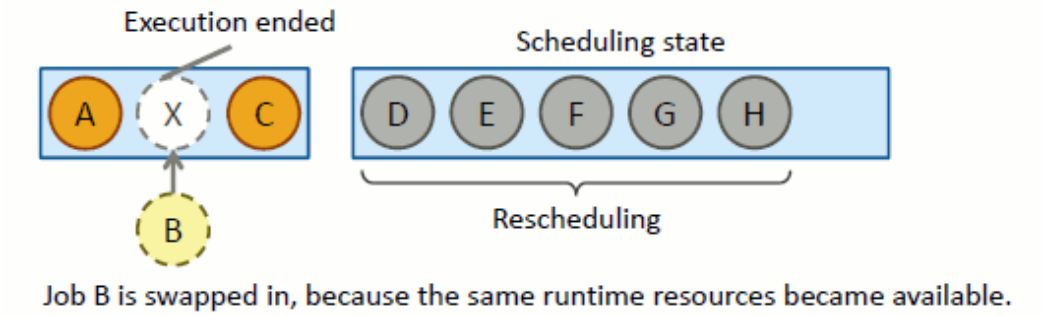



(*)The job is not swap out but might stopped, and be reexecuted.

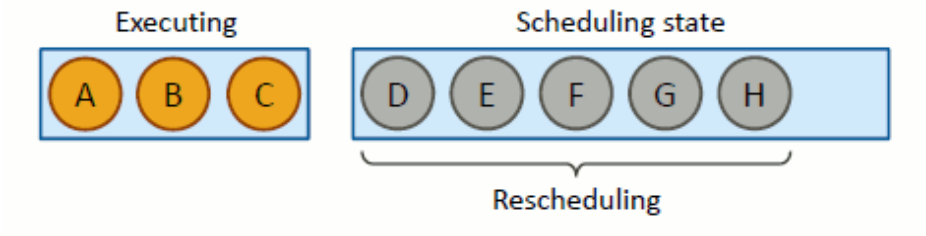
4. The emergency job is executed.




5. The emergency job ends.




6. The stopped job is restarted.




## 1.8.2 Group, user, and resource group priorities

In some cases, job execution priorities are assigned to groups or users in the OS. These priorities can only be configured by the administrator using the job ACL function (see "1.9 Job ACL Function").
You can check priorities assigned to groups or users by using the pjacl command (see "2.2.2 Checking restriction information").

Furthermore, job execution priorities may also be assigned to resource groups in the Job Operation Software.
These priorities can only be configured by the administrator as well. To obtain information on the priorities assigned to resource groups, contact the administrator.

## 1.8.3 Job priorities

Users can set the execution priorities of only their own jobs. The top priority is 255, and the lowest priority is 0.

You can specify the job priority at the job submission time by -p option of the pjsub command. After a job is submitted, you can change the job priority by the -p option of the pjalter command.

### See

For details on how to specify the priority at the job submission time, see "2.3.2.8 Specifying a job priority."

Each job is assigned two types of priorities: user-specific priorities and priorities applicable to all jobs within the resource unit. Priorities of the latter type can only be configured and changed by the administrator.

### 1.8.4 Fair share function

The fair share function determines execution priorities according to the results of computer resource usage for executed jobs, so that users and groups each have fair use of the system.
If the fair share function is enabled, the priority of a job decreases immediately after it is executed. The priority is restored at a constant rate as time elapses.
For this reason, if a large-scale job is executed or jobs are executed frequently, the subsequent jobs may not be immediately executed.
To find out whether the fair share function is enabled on your system, ask the administrator.

### 1.8.5 Job execution start time

Users can specify the job execution start time when submitting a job. To specify the job execution start time, use the --at option of the pjsub command when submitting a job. However, an interactive job cannot have a specified execution start time.
If the staging function is used, the stage-in process completes before the execution start time.

Job execution does not necessarily start at the specified time. It depends on the availability of computer resources and job selection policy. If execution cannot start at the specified time, it starts afterward as soon as computer resources can be reserved.

For details on how to specify the job execution start time, see "2.3.2.7 Specifying an execution start time."

## 1.9 Job ACL Function

The job operation software provides the "job ACL (Access Control List) function" to set upper limits on the memory and CPU time used by a job. The job ACL function also limits the functions available to users in accordance with the job operation policy.

The job ACL function has settings available for each cluster, resource unit, resource group, group, and user. The administrator configures settings appropriate to the job operation. The user can use the system within the restrictions by the job ACL function. For example, any jobs requiring resources or functions that exceed a restriction by the ACL function are rejected. Any job that reaches the job execution time limit is forcibly terminated.

You can use the pjacl command to check the details on the items restricted with the job ACL function.

For details on how to use the pjacl command to check the restrictions of the job ACL function and the restricted items, see "2.2.2 Checking restriction information."

## 1.10 Job Statistical Information

Information such as the amount of resources used by a job and various limit values is called "job statistical information." The job operation software provides a function to output this job statistical information. You can analyze the job execution situation after the fact by referencing job statistical information.

Users obtain job statistical information as job output results by using the pjsub command when submitting a job. You can also check the statistical information on a running job by using the pjstat command.

See

For details on how to check job statistical information by using the pjstat command, see "2.4.2 Displaying job statistical information." For details on how to output job statistical information by using the pjsub command, see "2.3.2.4 Specifying job statistical information output."

For details on the items included in job statistical information and their meanings, see "Appendix A Job Information."

## 1.11 Prologue and Epilogue Function

The administrator can set a specific script to be executed immediately before the start of and/or immediately after the execution of each job in accordance with the job operation policy. An example is a common process for all jobs, such as a process to set a specific environment variable or to prepare a work directory before job execution and delete it at job end.

This is called the "prologue and epilogue function."

Though users cannot change the prologue and epilogue function processes, keep the following in mind since they are executed as a part of jobs.

- The standard output and standard error output of the prologue and epilogue processes are output as job execution results.
- The resource limit values for jobs also apply to the prologue and epilogue processes.
- The contents added by the prologue and epilogue processes become job statistical information.
  However, whether the elapsed time of job execution includes the prologue and epilogue processes depends on the system settings. For details, contact the administrator.

# Chapter 2 Job Operation Procedures

This chapter describes the operations from creating a job to checking execution results.

Users create jobs, submit jobs, and check the execution results in the following workflow:

1. Creating a job
2. Confirming job control information
3. Submitting the job
4. Displaying job information
5. Deleting the job (Suspending job )
6. Confirming job results

Note

Unless otherwise noted, the operation examples in the following sections are performed on the login node.
Administrators can perform these operations not only on the login node but also with commands executed on other nodes. For details, see "F.8 Command List."

## 2.1 How to Create a Job

This section describes procedures related to creating and placing a job in an appropriate location.

### 2.1.1 How to create a job script

A job script is, in essence, a shell script.

The following example shows the descriptions in a job script.

```
#!/bin/bash                                          (1)
#PJM -L "node=2"                                     (2)
# comment                                            (3)
# comment
export PATH=/opt/FJSVfxlang/<version>/bin:$PATH      (4)
#PJM -L "elapse=86400"                               (5)

./a.out                                              (6)
```

(1) bash executes the job script.
(2) The fields delimited by a space after #PJM are handled like pjsub command arguments.
(3) Any line not beginning with #PJM is simply judged to be a shell comment.
(4) Environment variable settings, etc. (The example is for FX10)
(5) Once a line other than a comment line appears, the subsequent lines are simply judged to be comment lines.
(6) The command executes the program a.out.

Note the following about writing job scripts.

- Job scripts are executed by the shells listed below. Ask the administrator about the shells used by your system.
  - Specific shell that is set by the administrator
  - User login shell
  - Shell specified after "#!" on the first line of the job script

    Default shell that is set by the administrator, if no shell is specified
- You can write the arguments of the pjsub command for submitting a job, on a line beginning with "#PJM " in a job script. The arguments of the pjsub command has priority over that specified in a job script.

- Once a line other than a comment line appears in a job script, "#PJM" on the subsequent lines is ignored with the lines simply treated as comments.
- For a job script, the user who submits the job requires read authority. Execution authority is not required.
- Do not redirect to /dev/stdout or /dev/stderr in a job script. If the script redirects to either, the standard output file or standard error output file is overwritten from the beginning.

The following sections describe points for users to take into consideration when creating a job script.

### Information

You can write the options of the pjsub command for submitting a job, as described in "2.3 Submitting a Job," as a shell comment line in a shell script. For this reason, the following sections also describe the options of the pjsub command.

## 2.1.1.1 Execution directory when the staging function is used [FX10]

If the staging function is used, the job execution directory is created on the local file system. The directory is different from the job submission directory. Also, the path of the execution directory varies depending on the job execution format.

Input/output for files with the same parallel processes, like in an MPI-IO parallel job, uses the shared directory on the local file system. (Shared model)

In contrast, input/output for files with different parallel processes in a job, like in an MPI parallel job, uses a directory on the local file system of each compute node executing the parallel process to distribute load. (Unshared model)

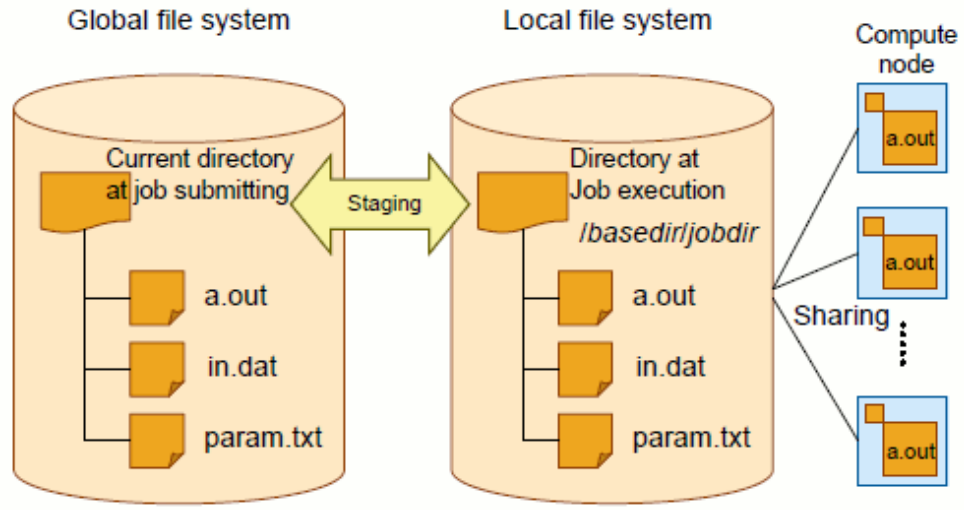
Therefore, for file access in a job when the staging function is used, we recommend not using an absolute path but using a relative path based on the execution directory.

The differences between the unshared and shared models regarding the job execution directory are described below.

1. Shared model

   In the shared model, each compute node shares the execution directory on the local file system.

   Figure 2.1 Staging with the shared model

2. Unshared model

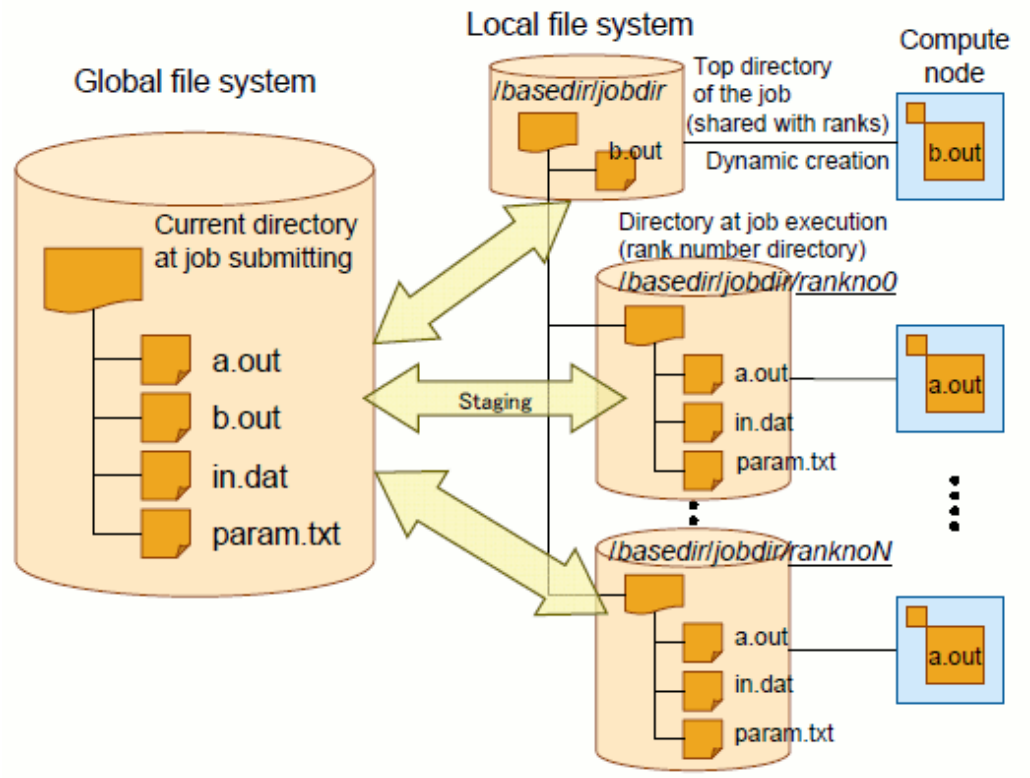
   There is a concept called rank, which corresponds to each parallel process in an MPI program. (The descriptions below do not include any details, which are beyond the scope of this document.)

   During input/output for a file by an MPI program, individual ranks use respectively different file systems for the input/output. This can be expected to prevent input/output conflicts and improve input/output performance. For this reason, a directory called a rank number directory is prepared for each rank number and used as the execution directory of the respective rank.

   Rank number directories are not shared between ranks, so the files required by ranks must be staged to their respective rank number directories.

When the rank number directories are used, the files required by process which is created dynamically must be stage-in into the top directory of the job (shared with ranks).

Figure 2.2 Staging with the unshared model




## Information

To use a rank number directory, the user needs to specify it when submitting the job. If no rank number directory is specified for use, the job uses the shared model. For details, see "2.3.5.8 Using a rank number directory [FX10]."

The following table lists the paths of execution directories for the shared and unshared models.

Table 2.1 Job execution directories

| Type of job | Shared/unshared model | directory at execution | Example |
|---|---|---|---|
| Normal job | Shared model | *<job name>.<job ID>* | JobA.123 |
| | Unshared model | *<job name>.<job ID>/<rank number>* | JobA.123/0 |
| Bulk job | Shared model | *<job name>.<sub job ID>* | JobA.123[0] |
| | Unshared model | *<job name>.<sub job ID>/<rank number>* | JobA.123[0]/0 |
| Step job | Shared model | *<job name>.<sub job ID>* | JobA.123_0 |
| | Unshared model | *<job name>.<sub job ID>/<rank number>* | JobA.123_0/0 |

## See

For details on how to specify the file transfer destination for staging, see "2.3.2.5 Specifying staging [FX10]."

An unsuitable description for the staging function will cause an error during staging. Before submitting a job, you can use the pjstgchk command to check whether the format is correct.

The following example shows flaws in the staging-related description in a created job script.

```
$ pjstgchk job.sh
--stgin "rank=* ./in.dat %r:./" "/home/usrname/in.dat -" Error: No such file or directory
--stgin "rank=* /work/bin/a.out %r:./" "/work/bin/a.out -" Error: Permission denied
```

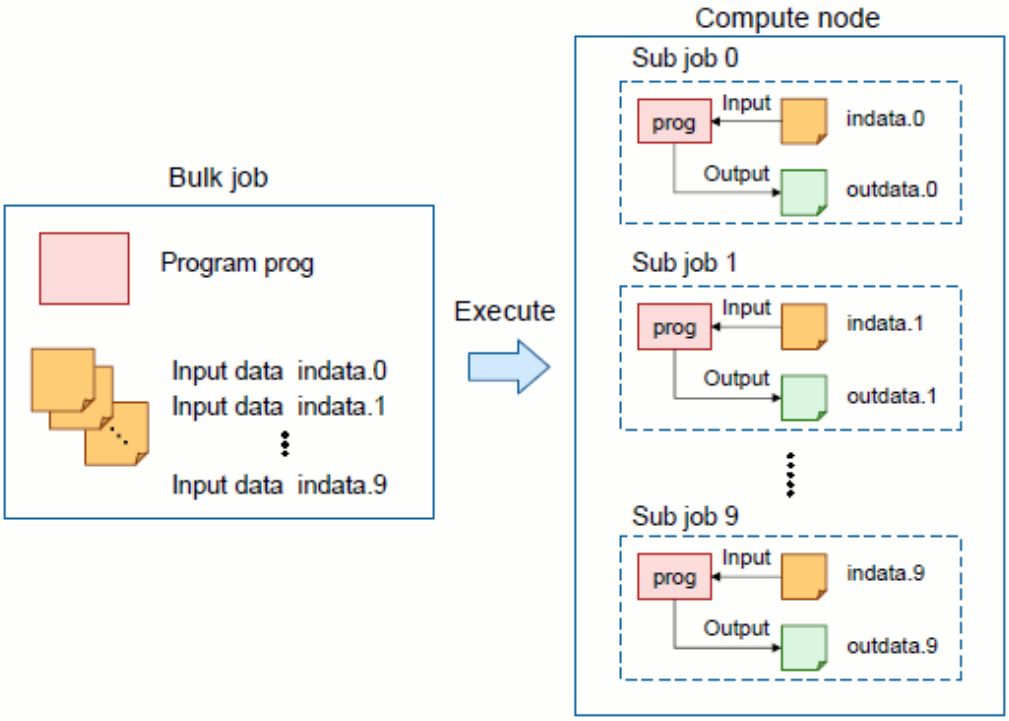
### 2.1.1.2 Creating a bulk job

A job script for a bulk job is designed such that input parameters of the job can be changed for each sub job.

For this reason, the bulk job uses the bulk number that is set for the sub job. The bulk number is set in the environment variable PJM_BULKNUM in the sub job.

The following example shows a bulk job.

Figure 2.3 Example of a bulk job




The following example shows a description in the job script of the above job.

```
#!/bin/sh
IN_DATA=./indata.${PJM_BULKNUM}              (1)
OUT_DATA=./outdata.${PJM_BULKNUM}            (2)

prog -i ${IN_DATA} -o ${OUT_DATA}            (3)

* The options at the job submission time (e.g., specifying the amount of resources)
  are omitted here to simplify the description.
```

(1) Determines the input data file name with the bulk number.
(2) Determines the output data file name with the bulk number.
(3) Specifies and executes the input/output data files with arguments of the program prog.

### 2.1.1.3 Creating a step job

A step job determines whether to execute a new sub job according to the results of past completed sub jobs. Therefore, consider the following points when creating a job.

- Consideration of the end code of a sub job

  In a step job, the end codes of sub jobs executed earlier change the behavior of later sub jobs. For this reason, set as many end codes of sub jobs as required.

- Consideration of stage-out of a dependent sub job [FX10]

  If a sub job requires the output file of the preceding sub job as input, execution of the sub job cannot start until stage-out of the preceding sub job is completed.

In the step job, specifying the condition for execution of next job, depending on the end code of previous job, is important. For details, see "2.3.3.2 How to submit a step job."

### 2.1.1.4 Creating a workflow job

In a workflow job, the user controls job submission with a shell script.

An example is a shell script that automatically submits a job after a specific job ends or selects the next job to submit based on the results of another job.

The user can thereby control job submission with the pjwait command provided by the job operation software.

- Wait for job end

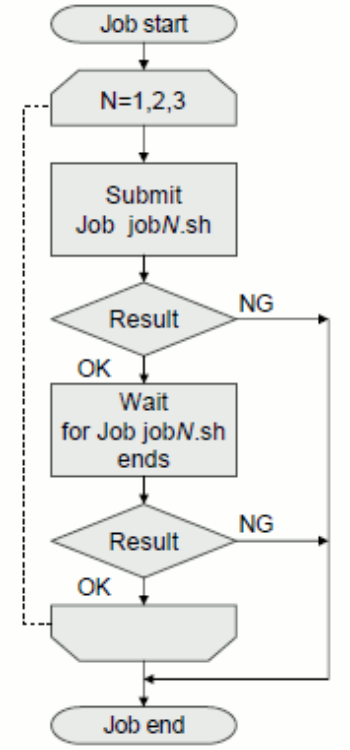
  You can use the pjwait command to wait for one or more jobs to end in a shell script.

- Result of job execution

  You can use the pjwait command to obtain the job exit code (PJM code; result code of the job manager), the exit code of a job script and the signal number when the job script ends.

The following example shows a workflow job.

Figure 2.4 Example of a workflow job




The following shows a description in the shell script of the above workflow job.

```
#!/bin/sh

for no in 1 2 3                          (1)
do
```

```
    JID=`pjsub -z jid job${no}.sh`        (2)
    if [ $? -ne 0 ]; then                 (3)
        exit 1
    fi
    set -- `pjwait $JID`                  (4)
    if [ $2 != "0" -o $3 != "0"]; then    (5)
        exit 1
    fi
done                                      (6)
```

(1) Submits job scripts job1.sh, job2.sh, and job3.sh in this order.
(2) Displays the job ID with the -z option, and assigns it to the shell variable JID.
(3) Ends the workflow job when the job submission result is other than 0.
(4) Waits for job end, and assigns the output to the positional parameter.

It is necessary to devise it so that the output result might become a multi-line according to the kind of the job and a method of specifying the job ID. For detail of the output, see the man page of pjwait command.

(5) Ends the workflow job when the PJM code ($2) or the job end code ($3) is other than 0.
(6) Goes to execution of the next job.

### 2.1.1.5 Creating a job for executing an MPI program

To execute an MPI program as a job, the node resources required for the MPI program must be allocated properly. Therefore, the user needs to exercise caution when writing these specifications in the job script.

For details, see "2.3.5 Submitting an MPI job."

Generally, MPI programs other than Technical Computing Language execute the process remotely on the nodes by using rsh and the ssh command. However, use the pjrsh command instead of them to execute the MPI program on Job Operation Software.
For details on how to use the pjrsh command instead of the rsh or ssh command, see "Appendix C Executing programs of MPI processing system other than Technical Computing Language".

Furthermore, some cases may require the setting of an environment variable specific to the MPI program. For details, see the "MPI User's Guide", which is a Technical Computing Language manual. Technical Computing Language is related software for creating MPI programs.

### 2.1.1.6 Execution a sequential program on the virtual nodes all at once [PG]

To execute a sequential program on multiple virtual nodes all at once, the pjexe command can be used.

The following example shows the execution of the sequential program prog on eight virtual nodes.

```
pjexe --vnode 8 prog
```

### 2.1.1.7 Creating a job that uses PAPI library or strace command [FX100/FX10]

To execute the following programs in a job to be submitted to the FX100 or FX10 compute node, the xospastop command (/usr/bin/xospastop) must be executed before executing the program.

- Program linked to the PAPI (Performance Application Program Interface) library of the open source software (including a program using, for example, TAU or Scalasca of the open source software)
- Program using the ptrace() function in the way that the strace command uses it

[CODING EXAMPLE]

```
#!/bin/bash
xospastop           <- Added description
strace ./a.out
```

If the xospastop command is not executed, the programs described above may not run normally.

When any of the following functions of Technical Computing Language is used, it meets the above condition; however, executing the xospastop command is not required.

- Profiler

- Function for outputting information at execution
- Debugger

### Information

The xospastop command is a command provided by the operating system for FX100 and FX10.
This command stops collecting the enhanced CPU statistical information of a job. There is no option or message output. The end status is always 0.

## 2.1.2 Job-related environment variables

The job operation software sets the following environment variables in jobs.

Table 2.2 Environment variables that may be referenced in jobs

| Environment variable name | Details |
|---|---|
| PJM_DPREFIX | Prefix indicating a command line from a job script to the job operation software.<br>Normally, it is as set by "#PJM", but you can change it with the -C or --dir-prefix option of the pjsub command. |
| PJM_ENVIRONMENT | "BATCH" for a batch job, and "INTERACT" for an interactive job |
| PJM_JOBID | Job ID |
| PJM_JOBNAME | Job name |
| PJM_JOBDIR | Path name of the directory that can be shared in a job in the case of using rank number directory.<br>Or, path name of the current directory of the beginning to execute job script in the case of not using rank number directory. |
| PJM_O_HOME | Environment variable HOME of the user who executed the pjsub command |
| PJM_O_HOST | Name of the host that executed the pjsub command |
| PJM_O_LANG | Environment variable LANG of the user who executed the pjsub command |
| PJM_O_LOGNAME | Environment variable LOGNAME of the user who executed the pjsub command |
| PJM_O_MAIL | Environment variable MAIL of the user who executed the pjsub command |
| PJM_O_PATH | Environment variable PATH of the user who executed the pjsub command |
| PJM_O_SHELL | Environment variable SHELL of the user who executed the pjsub command |
| PJM_O_TZ | Environment variable TZ of the user who executed the pjsub command |
| PJM_O_WORKDIR | Current directory at the pjsub command execution time |
| PJM_O_NODEINF **[PG]** | Path of the allocated node list file<br>Each line in the node list file describes the IP address of a node with an allocated virtual node. |
| PJM_COMMENT | Character string specified in the --comment option of the pjsub command |
| PJM_MAILSENDTO | User specified as the e-mail destination in the --mail-list option of the pjsub command |
| PJM_STEPNUM | Step number (set only for a step job) |
| PJM_BULKNUM | Bulk number (set only for a bulk job) |
| PJM_SUBJOBID | Sub job ID (set only for a step job or a bulk job) |
| PJM_APPNAME | Character string specified in the --appname option of the pjsub command |
| PJM_FSNAME | Character string specified in the --fs option of the pjsub command |

Values for these environment variables are determined with the following priority:

1. Specification as a shell executable within a job script (e.g., export PJM_O_MAIL=...)
2. Specification with -x option of the pjsub command (e.g., pjsub -x PJM_O_MAIL=...)

3. Specification with the -x option on the #PJM line within a job script (e.g., #PJM -x PJM_O_MAIL=...)
4. Value shown in "Table 2.2 Environment variables that may be referenced in jobs"

Unless instructed otherwise, nothing within a job inherits the environment variables set at the job submission time. To have environment variables inherited inside a job, specify the -X option of the pjsub command.

Note

Certain OS-defined environment variables (such as LD_LIBRARY_PATH) are not inherited within a job even if the -X option of the pjsub command is specified. Therefore, you will need to set these OS-defined environment variables again within the job script.

### 2.1.3 How to create a user program for execution

For details on how to create programs, see the manual provided with Technical Computing Language. For details on how to create MPI programs to be executed within jobs, automatic parallelization programs with the compiler, and other such programs, see the same manual.

In systems that use emergency jobs, when creating the user program, consideration must be given to emergency jobs. For details see "Appendix D Notes on Systems That Use Emergency Jobs [FX100]."

### 2.1.4 Location for placing a job

A created job must be accessible from the login node that submits the job and from the compute node that executes the job. Either a shared file system or the staging function may be used for this purpose.

- How to use a shared file system through the global file system

  Place the job on the file system shared by the login node and compute node.

  This style is always applied for the PRIMERGY computing node.

- How to use the staging function [FX10]

  Place the job on the file system that is the staging target on the login node.

Ask the administrator about how to place jobs with the system used by users and the path names of job locations, since they vary depending on the system configuration.

## 2.2 Checking Job-related Information

This section describes the information that users need to confirm before submitting their jobs.

### 2.2.1 Checking resource units and resource groups

To submit a job, the user checks whether the resource groups and resource units for executing the job satisfy size and shape requirements.

The user also checks the status of control of job submission to the resource units and resource groups, since the administrator may have such control.

You can check information about resource units and resource groups by using the --rsc option of the pjstat command.

```
$ pjstat --rsc
RSCUNIT                    RSCUNIT_SIZE  RSCGRP                    RSCGRP_SIZE
unit1[ENABLE,START]        20x9x16       group1[ENABLE,START]      20x3x16
unit1[ENABLE,START]        20x9x16       group2[ENABLE,STOP]       20x6x16
unit2[ENABLE,STOP]         20x9x16       group1[ENABLE,START]      1500
unit3[DISABLE,STOP]        4x18x16       group1[ENABLE,START]      1152
```

Table 2.3 pjstat command display items (1)

| Item | Description |
|---|---|
| RSCUNIT | Name and state of a resource unit<br>Format: *resource unit name*[*state*,...] |

| Item | Description |
|---|---|
| | "*state*" means the following.<br>ENABLE: Job can be submitted<br>DIABLE: Job cannot be submitted<br>START: Job can be executed<br>STOP: Job cannot be executed |
| RSCUNIT_SIZE | Size of a resource unit<br>[FX100/FX10]<br>The size is expressed as a cuboid with each dimension representing the number of Tofu units along the X, Y, or Z axis.<br>Format: *X*x*Y*x*Z*<br><br>[PG]<br>The number of nodes *N* that compose the resource unit is displayed. |
| RSCGRP | Name and state of a resource group<br>Format: *resource group name*[*state*,...]<br><br>The meaning of "*state*" is the same as for the previous item, RSCUNIT. |
| RSCGRP_SIZE | Size of a resource group<br>[FX100/FX10]<br>The size is expressed by a cuboid, with each dimension representing a number of nodes, or by a number of nodes.<br>Format: *X*x*Y*x*Z* (cuboid) or *N* (number of nodes)<br><br>[PG]<br>The number of nodes *N* that compose the resource group is displayed. |

Ask the administrator about the compute node models for resource units and resource groups.

The number of nodes allocated to the job and the node shape must fit within the resource group size (RSCGRP_SIZE shown above). If it exceeds this size, there will not be enough nodes to execute the job.

However, not all of the nodes shown by the RSCGRP_SIZE item in a resource group of FX100 or FX10 compute nodes may be allocable to the job, depending on the configuration.
The largest shape of nodes that can be allocated to the job can be obtained from the MAX_SIZE item, which is displayed by the pjstat command with the --shape option. See "Table 2.5 Size of nodes that can be allocated [FX100/FX10]."

```
$ pjstat --rsc --shape
RSCUNIT                      RSCUNIT_SIZE    RSCGRP                    RSCGRP_SIZE   MAX_SIZE
rscunit000[ENABLE,START]     3x2x9           rscgroup1[ENABLE,START]   540           4x6x18
rscunit000[ENABLE,START]     3x2x9           rscgroup2[ENABLE,START]   2x3x18        2x3x18
rscunit000[ENABLE,START]     3x2x9           rscgroup3[ENABLE,START]   2x3x18        2x3x18
rscunit000[ENABLE,START]     3x2x9           rscgroup4[ENABLE,START]   540           4x6x18

* The above is an example for a resource group of FX10 compute nodes.
```

Table 2.4 pjstat command display items (2)

| Item | Description |
|---|---|
| MAX_SIZE | Maximum size of nodes that can be allocated to a job<br><br>[FX100/FX10]<br>The size is expressed by a cuboid, *X*x*Y*x*Z*, with each side representing the number of nodes.<br>If there are multiple largest shapes, the command displays all of them, delimited by a comma.<br><br>[PG]<br>The displayed number of nodes, *N*, is equal to the RSCGRP_SIZE item. |

Table 2.5 Size of nodes that can be allocated [FX100/FX10]

| Node shape | Size of allocable nodes |
|---|---|
| One-dimensional shape | - In torus mode or mesh mode<br>The number of nodes must be up to the product (*X* Y*Z*) of the shape indicated by the item MAX_SIZE.<br>If the RSCGRP_SIZE item is displayed as the number of nodes, *N*, the number of nodes must be *N* or less.<br>- In non-contiguous mode<br>The number of nodes must be up to the product (*X* Y*Z*) of the shape indicated by the item RSCGRP_SIZE or up to the specified number (*N*) of nodes.<br>The item MAX_SIZE is not applicable in the case of non-contiguous mode. |
| Two-dimensional shape | - In torus mode or mesh mode<br>The nodes must fit within the shape (*X**3)x(*Y*Z*/3) obtained from the item MAX_SIZE (*X*x*Y*x*Z*), or any shape obtained as the result of rotating the shape.<br>If the RSCGRP_SIZE item is displayed as the number of nodes, *N*, the number of nodes must be *N* or less.<br>- In non-contiguous mode<br>The number of nodes must be up to the product (*X* Y*Z*) of the shape indicated by the item RSCGRP_SIZE or up to the specified number (*N*) of nodes.<br>The item MAX_SIZE is not applicable in the case of non-contiguous mode. |
| Three-dimensional shape | - In torus mode or mesh mode<br>The nodes must fit within the shape *X*x*Y*x*Z* indicated by the item MAX_SIZE, or any shape obtained as the result of rotating the shape.<br>If the RSCGRP_SIZE item is displayed as the number of nodes, *N*, the number of nodes must be *N* or less.<br>- In non-contiguous mode<br>The number of nodes must be up to the product (*X* Y*Z*) of the shape indicated by the item RSCGRP_SIZE or up to the specified number (*N*) of nodes.<br>The item MAX_SIZE is not applicable in the case of non-contiguous mode. |

[Example]

In the above display example, the number of nodes in the resource group rscgroup1 is 540. However, the maximum size of nodes that can be allocated is obtained from the MAX_SIZE item (4 x 6 x 18) as follows:

- For one-dimensional shape
  In torus mode or mesh mode, the number of nodes must be up to RSCGRP_SIZE = 540 and up to MAX_SIZE = 4 x 6 x 18 = 432. That is to say, the maximum number of nodes is 432.
  In non-contiguous mode, the number of nodes must be up to RSCGRP_SIZE = 540.

- For two-dimensional shape
  In torus mode or mesh mode, nodes must fit within the maximum shape of (4*3) x (6*18/3) = 12 x 36 or any shape obtained as the result of rotating the shape: 36x12. In addition, the number of nodes must be up to RSCGRP_SIZE = 540. That is to say, the maximum number of nodes, which is restricted by the maximum shape, is 12 x 36 = 432.
  In non-contiguous mode, the number of nodes must be up to RSCGRP_SIZE = 540.

- For three-dimensional shape
  In torus mode or mesh mode, nodes must fit within the maximum shape of 4 x 6 x 18 or any of the following shapes obtained as a result of rotating the maximum shape: 4 x 18 x 6, 6 x 4 x 18, 6 x 18 x 4, 18 x 6 x 4, or 18 x 4 x 6. (If ":strict" is specified, the nodes must fit within 4 x 6 x 18 only.) In addition, the number of nodes must be up to RSCGRP_SIZE = 540. That is to say, the maximum number of nodes, which is restricted by the maximum shape, is 4 x 6 x 18 = 432.
  In non-contiguous mode, the number of nodes must be up to RSCGRP_SIZE = 540.

## 2.2.2 Checking restriction information

The job ACL function restricts use of the system by users. (See "1.9 Job ACL Function.") Users need to submit and execute jobs within the restrictions.

You can use the pjacl command to check the information on restrictions that apply to a user.

```
$ pjacl
================== [PH] ====================   <- FX100 and FX10 compute node information
#                                                (PH: Stands for "PRIMEHPC")
#
# JOBACL information
#
user    user1                                                 (1)
group   group1                                                (2)

defines                                                       (3)
    default rscunit             rscunit1                      (4)
    default rscgroup            rscgroup1                     (5)
    default rscunit(interact)   rscunit1                      (6)
    default rscgroup(interact)  rscgroup1                     (7)
    default mesh                off                           (8)
    default allocation-mode     torus                         (9)
    user priority               127                           (10)
    fairshare init              0                             (11)
    fairshare recovery          100                           (12)
    ingroup priority            128                           (13)
    ingroup fairshare init      100                           (14)
    ingroup fairshare recovery  1                             (15)
    numa-policy                 pack                          (16)

groups                                                        (17)
    group priority              127                           (18)
    group fairshare init        0                             (19)
    group fairshare recovery    100                           (20)

pjsub option parameters                                       (21)
    (-L/--rsc-list)             upper           default       (22)
        (node-cpu=)             unlimited       unlimited     (23)
        (elapse=)               24:00:00        01:00:00      (24)
        (node=)                 2147483647      2             (25)
        (node-mem=)             unlimited       unlimited     (26)
        (node-quota=)           unlimited       unlimited     (27)
        (proc-core=)            unlimited       0             (28)
        (proc-cpu=)             unlimited       unlimited     (29)
        (proc-crproc=)          512             512           (30)
        (proc-data=)            unlimited       unlimited     (31)
        (proc-lockm=)           unlimited       unlimited     (32)
        (proc-msgq=)            unlimited       unlimited     (33)
        (proc-openfd=)          unlimited       1024          (34)
        (proc-psig=)            unlimited       unlimited     (35)
        (proc-filesz=)          unlimited       unlimited     (36)
        (proc-stack=)           unlimited       unlimited     (37)
        (proc-vmem=)            unlimited       unlimited     (38)
        (vnode-core=)           2147483647      1             (39)
        (vnode-mem=)            unlimited       unlimited     (40)
        (vnode=)                2147483647      1             (41)

    (--interact -L)             upper           default       (42)
        (node-cpu=)             unlimited       unlimited
        (elapse=)               24:00:00        01:00:00
        (node=)                 2147483647      1
        (node-mem=)             unlimited       unlimited
        (node-quota=)           unlimited       unlimited
        (proc-core=)            unlimited       0
        (proc-cpu=)             unlimited       unlimited
        (proc-crproc=)          512             512
        (proc-data=)            unlimited       unlimited
```

```
        (proc-lockm=)           unlimited       unlimited
        (proc-msgq=)            unlimited       unlimited
        (proc-openfd=)          unlimited       1024
        (proc-psig=)            unlimited       unlimited
        (proc-filesz=)          unlimited       unlimited
        (proc-stack=)           unlimited       unlimited
        (proc-vmem=)            unlimited       unlimited
        (vnode-core=)           2147483647      1
        (vnode-mem=)            unlimited       unlimited
        (vnode=)                2147483647      1

    (-p)                        upper           default     (43)
                                255             127

    (--bulk --sparam)           upper           default     (44)
        (number of subjob)      unlimited       -

execute                                                     (45)
    pjsub                       enable                      (46)
    pjsub(--no-stging)          enable                      (47)
    pjsub(--use-global)         enable                      (48)
    pjsub(--interact)           enable                      (49)
    pjsub(--step)               enable                      (50)
    pjsub(--bulk)               enable                      (51)
    pjstat                      enable                      (52)
    pjdel                       enable                      (53)
    pjhold                      enable                      (54)
    pjrls                       enable                      (55)
    pjwait                      enable                      (56)
    pjalter                     enable                      (57)
    pjsig                       enable                      (58)
    pjacl                       enable                      (59)
    pjsub(mesh-torus)           enable                      (60)
    pjsub(torus)                enable                      (61)
    pjsub(mesh)                 enable                      (62)
    pjsub(noncont)              enable                      (63)
    pjdel(--enforce)            disable                     (64)
    pjhold(--enforce)           disable                     (65)

permit                                                      (66)
    pjsub           allow own                               (67)
    pjstat          allow own                               (68)
    pjdel           allow own                               (69)
    pjhold          allow own                               (70)
    pjrls           allow own                               (71)
    pjwait          allow own                               (72)
    pjalter         allow own                               (73)
    pjsig           allow own                               (74)
    pjacl           allow own                               (75)
    pmerls          deny all                                (76)
    pmalter         deny all                                (77)

limit in rscunit (each users)                               (78)
    acceptable job              unlimited                   (79)
    running job                 unlimited                   (80)
    running bulk-subjob         unlimited                   (81)
    use node                    unlimited                   (82)
    acceptable job(interact)    unlimited                   (83)
    running job(interact)       unlimited                   (84)
    use node(interact)          unlimited                   (85)
    use core                    unlimited                   (86)
    use core(interact)          unlimited                   (87)
```

```
limit in rscunit (total users in samegroup)                 (88)
    acceptable job              unlimited
    running job                 unlimited
    running bulk-subjob         unlimited
    use node                    unlimited
    acceptable job(interact)    unlimited
    running job(interact)       unlimited
    use node(interact)          unlimited
    use core                    unlimited
    use core(interact)          unlimited

limit in rscunit (total all users)                          (89)
    acceptable job              unlimited
    running job                 unlimited
    running bulk-subjob         unlimited
    use node                    unlimited
    acceptable job(interact)    unlimited
    running job(interact)       unlimited
    use node(interact)          unlimited
    use core                    unlimited
    use core(interact)          unlimited

limit in rscgroup (each users)                              (90)
    acceptable job              unlimited
    running job                 unlimited
    running bulk-subjob         unlimited
    use node                    unlimited
    acceptable job(interact)    unlimited
    running job(interact)       unlimited
    use node(interact)          unlimited
    use core                    unlimited
    use core(interact)          unlimited

limit in rscgroup (total users in samegroup)                (91)
    acceptable job              unlimited
    running job                 unlimited
    running bulk-subjob         unlimited
    use node                    unlimited
    acceptable job(interact)    unlimited
    running job(interact)       unlimited
    use node(interact)          unlimited
    use core                    unlimited
    use core(interact)          unlimited

limit in rscgroup (total all users)                         (92)
    acceptable job              unlimited
    running job                 unlimited
    running bulk-subjob         unlimited
    use node                    unlimited
    acceptable job(interact)    unlimited
    running job(interact)       unlimited
    use node(interact)          unlimited
    use core                    unlimited
    use core(interact)          unlimited

================== [PG] ====================   <- PG compute node information
#
# JOBACL information
#
user    user1
group   group1

defines
```

```
    default rscunit             rscunit2
    default rscgroup            rscgroup2
    default rscunit(interact)   rscunit2
    default rscgroup(interact)  rscgroup2
    user priority               127
    fairshare init              0
    fairshare recovery          100
    ingroup priority            127
    ingroup fairshare init      0
    ingroup fairshare recovery  100
    load-policy                 balancing                   (93)
    vn-policy                   pack                        (94)
    exec-policy                 share                       (95)
    node-priority               31                          (96)
    numa-policy                 pack

groups
    group priority              127
    group fairshare init        0
    group fairshare recovery    100

pjsub option parameters
    (-L/--rsc-list)             upper           default
        (node-cpu=)             unlimited       unlimited
        (elapse=)               24:00:00        01:00:00
        (node=)                 2147483647      2
        (vnode-core=)           2147483647      1
        (vnode-mem=)            unlimited       unlimited
        (vnode=)                2147483647      1
        (proc-core=)            unlimited       0
        (proc-cpu=)             unlimited       unlimited
        (proc-crproc=)          512             512
        (proc-data=)            unlimited       unlimited
        (proc-lockm=)           unlimited       unlimited
        (proc-msgq=)            unlimited       unlimited
        (proc-openfd=)          unlimited       1024
        (proc-psig=)            unlimited       unlimited
        (proc-filesz=)          unlimited       unlimited
        (proc-stack=)           unlimited       unlimited
        (proc-vmem=)            unlimited       unlimited

    (--interact -L)             upper           default
        (node-cpu=)             unlimited       unlimited
        (elapse=)               24:00:00        01:00:00
        (node=)                 2147483647      2
        (vnode-core=)           2147483647      1
        (vnode-mem=)            unlimited       unlimited
        (vnode=)                2147483647      1
        (proc-core=)            unlimited       0
        (proc-cpu=)             unlimited       unlimited
        (proc-crproc=)          512             512
        (proc-data=)            unlimited       unlimited
        (proc-lockm=)           unlimited       unlimited
        (proc-msgq=)            unlimited       unlimited
        (proc-openfd=)          unlimited       1024
        (proc-psig=)            unlimited       unlimited
        (proc-filesz=)          unlimited       unlimited
        (proc-stack=)           unlimited       unlimited
        (proc-vmem=)            unlimited       unlimited

    (-p)                        upper           default
                                255             127
```

```
    (--bulk --sparam)           upper           default
        (number of subjob)      unlimited       -

execute
    pjsub                       enable
    pjsub(--interact)           enable
    pjsub(--step)               enable
    pjsub(--bulk)               enable
    pjstat                      enable
    pjdel                       enable
    pjhold                      enable
    pjrls                       enable
    pjwait                      enable
    pjalter                     enable
    pjsig                       enable
    pjacl                       enable
    pjsub(-P vn-policy)         enable                      (97)
    pjsub(-P exec-policy)       enable                      (98)
    pjdel(--enforce)            disable
    pjhold(--enforce)           disable

permit
    pjsub           allow own
    pjstat          allow own
    pjdel           allow own
    pjhold          allow own
    pjrls           allow own
    pjwait          allow own
    pjalter         allow own
    pjsig           allow own
    pjacl           allow own
    pmerls          deny all
    pmalter         deny all

limit in rscunit (each users)
    acceptable job              unlimited
    running job                 unlimited
    running bulk-subjob         unlimited
    use core                    unlimited
    acceptable job(interact)    unlimited
    running job(interact)       unlimited
    use core(interact)          unlimited

limit in rscunit (total users in samegroup)
    acceptable job              unlimited
    running job                 unlimited
    running bulk-subjob         unlimited
    use core                    unlimited
    acceptable job(interact)    unlimited
    running job(interact)       unlimited
    use core(interact)          unlimited

limit in rscunit (total all users)
    acceptable job              unlimited
    running job                 unlimited
    running bulk-subjob         unlimited
    use core                    unlimited
    acceptable job(interact)    unlimited
    running job(interact)       unlimited
    use core(interact)          unlimited

limit in rscgroup (each users)
    acceptable job              unlimited
```

```
    running job                 unlimited
    running bulk-subjob         unlimited
    use core                    unlimited
    acceptable job(interact)    unlimited
    running job(interact)       unlimited
    use core(interact)          unlimited

limit in rscgroup (total users in samegroup)
    acceptable job              unlimited
    running job                 unlimited
    running bulk-subjob         unlimited
    use core                    unlimited
    acceptable job(interact)    unlimited
    running job(interact)       unlimited
    use core(interact)          unlimited

limit in rscgroup (total all users)
    acceptable job              unlimited
    running job                 unlimited
    running bulk-subjob         unlimited
    use core                    unlimited
    acceptable job(interact)    unlimited
    running job(interact)       unlimited
    use core(interact)          unlimited
```

(1) Execution user name
(2) Execution group name
(3) User-related setting values
(4) Name of the default resource unit for submitting a batch job
(5) Name of the default resource group for submitting a batch job
(6) Name of the default resource unit for submitting an interactive job
(7) Name of the default resource group for submitting an interactive job
(8) Node-exclusive job (mesh mode) [FX100/FX10]
(9) Node allocation method [FX100/FX10]
(10) User priority
(11) Initial fair share value of the user
(12) Fair share value recovery rate of the user
(13) User priority in the group
(14) User's initial fair share value in the group
(15) User's fair share value recovery rate in the group
(16) NUMA allocation policy [FX100][PG]
(17) Group-related setting values
(18) Group priority
(19) Initial fair share value of the group
(20) Fair share value recovery rate of the group
(21) Upper limit and default values specified in pjsub command options
(22) Upper limit and default values in a batch job
(23) Limit on CPU time per node
(24) Limit on executable time per job
(25) Limit on the number of nodes per job
(26) Limit on the memory used per node [FX100/FX10]
(27) Limit on files allocated per node [FX10]
(28) Limit on the core file size per process
(29) Limit on CPU time per process
(30) Limit on the number of processes generated per process
(31) Limit on the data segment size per process
(32) Limit on the lock memory size per process
(33) Limit on the message queue size per process
(34) Limit on the number of file descriptors per process
(35) Limit on the number of pending signals per process

(36) Limit on the file size per process
(37) Limit on the stack size per process
(38) Limit on the virtual memory size per process
(39) Limit on the number of cores per a virtual node
(40) Limit on the memory size allocated to a virtual node
(41) Limit on the number of virtual nodes per job
(42) Upper limit and default values in an interactive job
(43) Upper limit and default values for job priority
(44) Limit on the number of sub jobs of a bulk job
(45) Execution privileges enabled or disabled
(46) Privilege to execute the pjsub command
(47) Privilege to submit without staging [FX10]
(48) Privilege to use the global file system [FX10]
(49) Privilege to submit interactive jobs
(50) Privilege to submit step jobs
(51) Privilege to submit bulk jobs
(52) Privilege to execute the pjstat command
(53) Privilege to execute the pjdel command
(54) Privilege to execute the pjhold command
(55) Privilege to execute the pjrls command
(56) Privilege to execute the pjwait command
(57) Privilege to execute the pjalter command
(58) Privilege to execute the pjsig command
(59) Privilege to execute the pjacl command
(60) Privilege to submit jobs in mesh or torus mode [FX100/FX10]
(61) Privilege to submit jobs in torus mode [FX100/FX10]
(62) Privilege to submit jobs in mesh mode [FX100/FX10]
(63) Privilege to submit jobs in non-contiguous mode [FX100/FX10]
(64) Privilege to cancel prologue and epilogue scripts in pjdel
(65) Privilege to cancel prologue and epilogue scripts in pjhold
(66) Permission for operated objects
(67) Permission allowing execution by the group with the pjsub command
(68) Target job of the pjstat command
(69) Target job of the pjdel command
(70) Target job of the pjhold command
(71) Target job of the pjrls command
(72) Target job of the pjwait command
(73) Target job of the pjalter command
(74) Target job of the pjsig command
(75) Target job of the pjacl command
(76) Target job of the pmerls command
(77) Target job of the pmalter command
(78) Limit values in a resource unit (each user)
(79) Limit on concurrent acceptance for a batch job
(80) Limit on concurrent execution for a batch job
(81) Limit on concurrent execution for sub jobs of a bulk job
(82) Limit on concurrent node usage for a batch job [FX100/FX10]
(83) Limit on concurrent acceptance for an interactive job
(84) Limit on concurrent execution for an interactive job
(85) Limit on concurrent node usage for an interactive job [FX100/FX10]
(86) Limit on the number of CPU cores that can be used concurrently by batch jobs
(87) Limit on the number of CPU cores that can be used concurrently by interactive jobs
(88) Limit values in a resource unit (total values in a group).
(89) Limit values in a resource unit (total values for all users).
(90) Limit values in a resource group (each user).
(91) Limit values in a resource group (total values in a group).
(92) Limit values in a resource group (total values for all users).
(93) Distribution or concentration [PG]

(94) Virtual node placement policy[PG]
(95) Execution mode [PG]
(96) Node priority [PG]
(97) Privilege to specify a virtual node placement policy [PG]
(98) Privilege to specify an execution mode policy [PG]

As shown above for the defined item "permit," "allow" indicates operation permitted, and "deny" indicates operation denied. The following table lists the meanings of the operated objects following "allow" or "deny."

Table 2.6 Notation of operated objects in the pjacl command

| Notation | Description |
|---|---|
| own | The target is only one's own submitted jobs. |
| all | The target is the jobs of all users. |
| g(g1, g2, ...) | The target group names are g1, g2, and so on. |
| u(u1, u2, ...) | The target user names are u1, u2, and so on. |

## Note

- The job ACL function may not permit a user to execute the job operation software commands described in this manual. Check the output results of the pjacl command.
  The user might not have a permission to execute the pjacl command.
- According to a job operation setting, node-exclusive jobs may be limited by the number of concurrently used nodes or the number of concurrently used CPU cores. Ask the administrator about which one of the above is set for job operations.
  If set to be limited by the number of concurrently used CPU cores, the number of CPU cores concurrently used by node-exclusive jobs is "number of mounted CPU cores per node x number of requested nodes."

Using the --limit option of the pjstat command, users can check the limit values and current usage amounts concerning submission of their jobs in the contents displayed by the pjacl command.

- Resource unit of FX100 or FX10 compute nodes

```
$ pjstat --limit --rscunit rscunit1
 System Resource Information:
 RSCUNIT: rscunit1
 USER: user1
  LIMIT-NAME                          LIMIT         ALLOC
  ru-accept                       unlimited             5      (1)
  ru-run-job                      unlimited             3      (2)
  ru-run-subjob                   unlimited             0      (3)
  ru-use-node                     unlimited          1011      (4)
  ru-interact-accept              unlimited             0      (5)
  ru-interact-run-job             unlimited             0      (6)
  ru-interact-use-node            unlimited             0      (7)
  ru-use-core                     unlimited            30      (8)
  ru-interact-use-core            unlimited             0      (9)
 GROUP: group1
  LIMIT-NAME                          LIMIT         ALLOC
  ru-accept                       unlimited             5
  ru-run-job                      unlimited             3
  ru-run-subjob                   unlimited             0
  ru-use-node                     unlimited          1011
  ru-interact-accept              unlimited             0
  ru-interact-run-job             unlimited             0
  ru-interact-use-node            unlimited             0
  ru-use-core                     unlimited            30
  ru-interact-use-core            unlimited             0
ALL:
  LIMIT-NAME                          LIMIT         ALLOC
```

```
  ru-accept                      unlimited             5
  ru-run-job                     unlimited             3
  ru-run-subjob                  unlimited             0
  ru-use-node                    unlimited          1011
  ru-interact-accept             unlimited             0
  ru-interact-run-job            unlimited             0
  ru-interact-use-node           unlimited             0
  ru-use-core                    unlimited            30
  ru-interact-use-core           unlimited             0
```

(1) Limit on concurrent acceptance for a batch job
(2) Limit on concurrent execution for a batch job
(3) Limit on concurrent execution for sub jobs of a bulk job
(4) Limit on concurrent node usage for batch job
(5) Limit on concurrent acceptance for an interactive job
(6) Limit on concurrent execution for an interactive job
(7) Limit on concurrent node usage for an interactive job
(8) Limit on the number of CPU cores that can be used concurrently by batch jobs
(9) Limit on the number of CPU cores that can be used concurrently by interactive jobs

- Resource unit of PRIMERGY compute nodes

```
$ pjstat --limit --rscunit rscunit2
 System Resource Information:
 RSCUNIT: rscunit2
 USER: user1
  LIMIT-NAME                         LIMIT         ALLOC
  ru-accept                      unlimited             0
  ru-run-job                     unlimited             0
  ru-run-subjob                  unlimited             0
  ru-use-core                    unlimited             0
  ru-interact-accept             unlimited             0
  ru-interact-run-job            unlimited             0
  ru-interact-use-core           unlimited             0
 GROUP: group1
  LIMIT-NAME                         LIMIT         ALLOC
  ru-accept                      unlimited             0
  ru-run-job                     unlimited             0
  ru-run-subjob                  unlimited             0
  ru-use-core                    unlimited             0
  ru-interact-accept             unlimited             0
  ru-interact-run-job            unlimited             0
  ru-interact-use-core           unlimited             0
 ALL:
  LIMIT-NAME                         LIMIT         ALLOC
  ru-accept                      unlimited             0
  ru-run-job                     unlimited             0
  ru-run-subjob                  unlimited             0
  ru-use-core                    unlimited             0
  ru-interact-accept             unlimited             0
  ru-interact-run-job            unlimited             0
  ru-interact-use-core           unlimited             0
```

The following table lists the items displayed by the --limit option of the pjstat command and their meanings.

| Item | Description |
|---|---|
| LIMIT-NAME | Limit value name |
| RSCUNIT | Resource unit name |
| GROUP | Group name in the operating system |
| LIMIT | Limit value |

| Item | Description |
|---|---|
| ALLOC | Current allocation value |

## 2.2.3 Checking resources

Use the pjshowrsc command to check the usage of resources in the system.

You can use the --rscunit option to display the status of resources of the resource group executing the job. You can use the -E option to display a lower layer.

```
$ pjshowrsc --rscunit -E
[ CLST: clst ]
[ RSCUNIT: unit1 ]
    NODEID     CPU                    MEM                    LFS
             TOTAL    FREE  ALLOC   TOTAL    FREE   ALLOC   TOTAL    FREE   ALLOC
0x01010010       8       0       8    14Gi       0    14Gi    130G    130G       0
0x01010011       8       0       8    14Gi       0    14Gi    130G    130G       0
0x01010012       8       0       8    14Gi       0    14Gi    130G    130G       0
0x01010013       8       0       8    14Gi       0    14Gi    130G    130G       0
0x01010014       8       8       0    14Gi    14Gi       0    130G    130G       0
```

The meanings of the displayed items are as follows.

| Item | | Meaning |
|---|---|---|
| CPU | TOTAL | Number of CPU cores mounted on nodes |
| | FREE | Number of CPU cores not allocated to jobs |
| | ALLOC | Number of CPU cores already allocated to jobs |
| MEM | TOTAL | Amount of memory mounted on nodes |
| | FREE | Amount of memory not allocated to jobs |
| | ALLOC | Amount of memory already allocated to jobs |
| LFS [FX10] | TOTAL | Size of local file system that can be used on nodes |
| | FREE | Size of local file system not allocated to jobs |
| | ALLOC | Size of local file system already allocated to jobs |

The pjshowrsc command displays the resource units for which the user has privileges. However, to check the default resource group for the user, use the --rsc option of the pjstat command. (See "2.2.1 Checking resource units and resource groups.")

Note

In environments in which a local file system does not exist, all values of item LFS of the pjshowrsc command are "-".

# 2.3 Submitting a Job

This section describes how to submit a job.

## 2.3.1 Basic methods of submitting a job

Use the pjsub command to submit a job.

The following is a simple example of submitting the job script job.sh.

- Submit a job to FX100 or FX10 compute nodes

```
$ pjsub job.sh
```

- Submit a job to PRIMERGY compute nodes

  The user must specify the number of virtual nodes (-L "vnode=*num*").

```
$ pjsub -L "vnode=8" job.sh
```

The following message appears when the job is accepted normally.

```
[INFO] PJM 0000 pjsub Job jobid submitted.
```

*jobid* is set as the job ID for the submitted job.

Also, "job name" is set for the job. The default is the job script name. You can use the -N or --name option to set a job name.

## Note

- On some systems, a single cluster may contain multiple resource units or groups that differ in terms of job operation policy, resource quantity, and/or compute node model.
  If necessary on such systems, specify the resource unit or group to which to submit jobs. (See "2.3.2.1 Specifying resources.")
  You can use the pjacl command to check the default resource unit or group to which to submit jobs. (See "2.2.1 Checking resource units and resource groups.")
- Keep the following in mind about the directory for executing the pjsub command for submitting a job to FX10 compute nodes.
  - Without using the staging function

    On systems that can use the staging function, jobs are staged by default. Use the --no-stging option of the pjsub command when you do not use the staging function.

```
$ pjsub --no-stging job.sh
```

    The current directory where the pjsub command was executed is the current directory of the job on a compute node. Note the relationship between the current directory and the location of the program and data to be accessed within the job script.

  - Using the staging function

    The current directory where the pjsub command was executed is the base directory of staging. Note the relationship between the current directory and how the staging target file is specified. For details, see "2.1.1.1 Execution directory when the staging function is used [FX10]".

  To submit a job to PRIMERGY compute nodes, the job must be located in the directory that is shared with the login nodes and the compute nodes, and it must be accessed with same path name on each node.

If no job script is specified, the job contents are read from the standard input of the pjsub command.

```
$ pjsub
#!/bin/sh                                          (1)
...
<CTRL-d>                                           (2)
[INFO] PJM 0000 pjsub Job 1000 submitted.          (3)
```

(1) The user enters the job contents to the standard input of pjsub.
(2) The user closes the standard input with the <CTRL-d> keys.
(3) The job is accepted.

During job execution, you can specify the following information in the options of the pjsub command:

- Type of job (job model or job type)
- Number of nodes allocated to the job and the node shape
- Upper limits on memory and executable time
- MPI program-specific information

```
$ pjsub option job.sh
```

### Information

You can write the options of the pjsub command not only on the command line but also in a job script.
The specifications with the options of the pjsub command have priority over the specifications in a job script.

The options of the pjsub command include many options appropriate to certain types of job, purposes, etc. For details, see its man page.

### Note

The pjsub command ignores the options of the FX100 or FX10 compute node for the PRIMERGY compute nodes. When the options of the PRIMERGY compute node are specified to the pjsub command for the FX100 or FX10 compute nodes, an error occurs.

## 2.3.2 Options at the job submission time

This section describes the pjsub command options for specifying resource allocation for a job and job operations.

### 2.3.2.1 Specifying resources

You can specify the resources to be allocated to a job by using the -L option or --rsc-list option of the pjsub command.

```
{-L | --rsc-list } item=value
```

You can specify the following items and values.

Table 2.7 Resource specification formats

| Format | Description |
|---|---|
| node=*shape*[:strict][:*node_allocation*]<br>[FX100/FX10] | Number and shape of nodes allocated to a job, and how to allocate nodes to a job (for node-exclusive job)<br>For details, see "2.3.2.2 Specifying a node resource." |
| vnode=*num*[*CPU cores or memory size*] | Number of the virtual nodes and CPU cores, or amount of memory allocated for node-sharing job or the job on the PRIMERGY.<br>For details, see "2.3.2.2 Specifying a node resource." |
| node-cpu=*limit* | Upper limit of CPU usage time spent per node [FX100/FX10], or per virtual node [PG] |
| elapse=*limit* | Maximum executable time for a job |
| node-mem=*limit*<br>[FX100/FX10] | Upper limit of the amount of memory used per node (for node-exclusive job)<br>The minimum value is 1 MiB (mebibyte = $2^{20}$ bytes).<br>The upper limit of the memory use amount is set by rounding up the specified value in the following units.<br>FX100 compute node: 32Mi bytes<br>FX10 compute node: 256Mi bytes |
| node-quota=*limit*<br>[FX10] | Upper limit of the disk usage amount per node |
| rscunit=*name* | Resource unit name for submitting a job<br>*name* is a character string with up to 63 characters. |
| rscgrp=*name* | Resource group name for submitting a job<br>*name* is a character string with up to 63 characters. |
| proc-core=*limit* | Maximum core file size for each process per job |
| proc-cpu=*limit* | Maximum CPU time for each process per job<br>The minimum value is 1 second. |
| proc-crproc=*limit* | Maximum number of processes that can be generated with the real user ID on the node executing a process |

| Format | Description |
|---|---|
| proc-data=*limit* | Maximum data segment size for each process per job |
| proc-lockm=*limit* | Maximum lock memory size for each process per job |
| proc-msgq=*limit* | Maximum POSIX message queue size that can be reserved with the real user ID on the node executing a process |
| proc-openfd=*limit* | Maximum number of file descriptors for each process per job |
| proc-psig=*limit* | Maximum number of pending signals with the real user ID on the node executing a process |
| proc-filesz=*limit* | Maximum file size for each process per job |
| proc-stack=*limit* | Maximum stack size for each process per job |
| proc-vmem=*limit* | Maximum virtual memory size for each process per job |

Specify *limit* in the expression format for time shown as follows.

Table 2.8 Expression formats for amounts of resources

| Amount | Expression format |
|---|---|
| Time | You can specify a time in seconds or in the MM:SS, HH:MM:SS, or HHhMMmSSs format. You can also use the uppercase letters of h, m, and s.<br>Unless otherwise noted, the specifiable times are in a range of 1 to 2147483647 seconds. For no limit, specify unlimited.<br><br>Example:<br>elapse=100 (100 seconds)<br>elapse=1:40 (1 minute 40 seconds)<br>elapse=6:10:30 (6 hours 10 minutes 30 seconds)<br>elapse=10h30m50s (10 hours 30 minutes 50 seconds)<br>elapse=300m (300 minutes)<br>elapse=unlimited (No limit) |
| Amount of memory (bytes) | Express the value as a power of two (unit).<br>The units Ki (kibi = $2^{10}$), Mi (mebi = $2^{20}$), Gi (gibi = $2^{30}$), Ti (tebi = $2^{40}$), and Pi (pebi =$2^{50}$) are powers of two. If omitted, Mi is assumed specified.<br>Do not enter any space between the numerical value and the unit representing a power of two.<br>Unless otherwise noted, the specifiable values are in a range of 0 bytes to 2147483647 Mi bytes. For no limit, specify unlimited.<br><br>Example:<br>node-mem=256Mi (256Mi bytes = 256*$2^{20}$ bytes)<br>node-mem=256 (Same as above)<br>node-mem=10Gi (10Gi bytes = 10*$2^{30}$ bytes)<br>node-mem=unlimited (No limit) |
| Disk capacity<br>File size (bytes) | Express the value as a power of 10 (unit).<br>The units K (kilo = $10^{3}$), M (mega = $10^{6}$), G (giga = $10^{9}$), T (tera = $10^{12}$), and P (peta =$10^{15}$) are powers of 10. If omitted, M is assumed specified. Do not enter any space between the numerical value and the unit representing a power of 10.<br>Unless otherwise noted, the specifiable values are in a range of 0 bytes to 2147483647 M bytes. For no limit, specify unlimited.<br><br>Example:<br>node-quota=10M (10M bytes = 10*$10^{6}$ bytes)<br>node-quota=10 (Same as above)<br>node-quota=10G (10G bytes = 10*$10^{9}$ bytes)<br>node-quota=unlimited (No limit) |

| Amount | Expression format |
|---|---|
| Numerical value | For the amount of any resource other than the above, specify only a numerical value.<br>Unless otherwise noted, the specifiable values are in a range of 0 to 2147483647. For no limit, specify unlimited. |

These values cannot exceed the upper limit values defined by the job ACL function. For any option not specified, the corresponding default value defined by the job ACL function is applied. To check the values defined by the job ACL function, see "2.2.2 Checking restriction information."

For details on how a job behaves when it uses more resources than the specified upper limit, see "2.3.2.3 Job operation when a job exceeds an upper limit on amount of resources."

The following examples show the specification of job resources for the FX10 compute nodes.

- Example of allocation of 64 nodes to a job

```
$ pjsub -L "node=64" job.sh
```

  The job is executed on the specified number of nodes. The job cannot use more than the specified number of nodes. For example, suppose an MPI program requires multiple nodes. If the MPI program requires more nodes than the allocated number of nodes, the program ends with an error.

  For PRIMERGY compute nodes, the number of the virtual nodes (-L "vnode=") is necessary. For details, see "2.3.2.2 Specifying a node resource."

- Example of setting 86400 seconds as the executable time of a job

```
$ pjsub -L "elapse=86400" job.sh
```

  If the running job exceeds the specified job executable time, the job is forcibly terminated.

- Example of setting a 256 Mi byte limit on the memory used for each node of a job.

```
$ pjsub -L "node-mem=256Mi" job.sh
```

  The job is executed within this limit on the memory used for each node. Moreover, not even with a request can the limit on the memory used per node be exceeded.

The following example shows a combination of the above-described options.

This example submits the MPI program prog with the following limit values: number of allocated nodes (2), executable time (86400 seconds), and upper limit on the memory used per node (256 Mi bytes). Also, the job script job.sh, not the arguments of the pjsub command, specify the limit values.

```
$ cat job.sh
#!/bin/sh
#PJM -L "node=2"                                                    (1)
#PJM -L "elapse=86400"                                              (2)
#PJM -L "node-mem=256Mi"                                            (3)
export PATH=/opt/FJSVfxlang/<version>/bin:$PATH                     (4)
export LD_LIBRARY_PATH=/opt/FJSVfxlang/<version>/lib64:$LD_LIBRARY_PATH  (5)
mpiexec ./prog                                                      (6)
$ pjsub job.sh
```

(1) Number of allocated nodes: 2
(2) Executable time: 86400 seconds
(3) Upper limit on the memory used per node: 256Mi bytes
(4) Environment setting for MPI job execution (for FX10)
(5) Environment setting for MPI job execution (for FX10)
(6) mpiexec command executing the MPI program prog

### Note

The mpiexec command and environment variables in the above example are descriptions required for executing the MPI program. The method is different in the FX100 or FX10 compute node and the PRIMERGY compute node. For details, see the MPI user's manual, which is a Technical Computing Language manual.

### Information

Before executing a POSIX thread program, set the maximum virtual memory size (proc-vmem) to unlimited or to a sufficiently larger value than the maximum stack size (proc-stack).
Normally, the thread stack for POSIX threads is allocated to the area acquired by mmap(2). If not set to unlimited, the maximum stack size (proc-stack) is the default thread stack size. Therefore, if the maximum virtual memory size is not sufficiently larger than the maximum stack size, the thread stack area cannot be secured and the program ends abnormally.
For details on the thread stack, check the POSIX thread specifications.

## 2.3.2.2 Specifying a node resource

The node allocation method to the job is different in the FX100 or FX10 compute node and the PRIMERGY compute node.

[FX100/FX10: Allocating nodes (node-exclusive job)]

Jobs are executed on FX100 or FX10 compute nodes located in a virtual space. When submitting a job, the user can select a one-dimensional, two-dimensional, or three-dimensional space as the space for placing the job.The user specifies the node shape in this logical space. For details on node shapes, see "1.7 Node Resource Allocation."

Use the node parameter of the -L or --rsc-list option of the pjsub command to specify the node shape, which must be able to store all existing processes during job execution.

```
{-L | --rsc-list} node=shape[:strict][:torus|:mesh|:noncont]
```

Table 2.9 Node shape specification formats [FX100/FX10]

| Format | Description |
|---|---|
| *shape*[:strict] | Specify the size along each axis in *shape*, according to the dimensions of the node shape, which is *X*, *X*x*Y*, or *X*x*Y*x*Z*.<br>If the node shape is three-dimensional, you can specify strict, such as in XxYxZ:strict.<br>Normally, the allocated node shape as specified is automatically rotated to fit in the available node space. However, if strict is specified, the allocated node shape is not rotated. To express one-dimensional and two-dimensional shapes in the same way, use the strict specification in a three-dimensional shape expression. |
| :torus<br>or<br>:mesh<br>or<br>:torus | If the job is a node-exclusive job, you can specify a node allocation method (torus mode, mesh mode, or non-contiguous mode).<br>The string "torus" means torus mode, in which computer resources are allocated to jobs in Tofu (12 nodes) units.<br>The string "mesh" means mesh mode, in which computer resources are allocated to jobs in units of nodes.<br>The string "noncont" means non-contiguous mode, in which computer resources are allocated to jobs in units of nodes.<br>For details on each method of allocating nodes, see "1.7.2.1 Node allocation in units of nodes or Tofus."<br>When omitted, the default value defined with the job ACL function is followed. |

```
[Example] Executing a job with a node shape of 4 x 4 x 2 in torus mode
$ pjsub -L node=4x4x2:torus job.sh
```

## Note

- The specified node shape cannot exceed the maximum dimensions of the system. If it exceeds the maximum dimensions, the job submission will failed. For details on how to find out the maximum dimensions, see "2.2.1 Checking resource units and resource groups."
  The node shape may not fit within the maximum dimensions as is but may fit when rotated. If so, it is automatically rotated. The user does not need to specify the rotated shape.
- For a node-exclusive job that uses the staging function, neither mesh mode nor non-contiguous mode can be specified. If mesh mode or non-contiguous mode is specified for such a job, job submission will encounter an error.
- Each of node allocation methods ":mesh," ":torus," and ":noncont" can be specified only when the corresponding function item in the job ACL (execute pjsub(torus), execute pjsub(mesh), and execute pjsub(noncont), respectively) is set to "enable" and execute pjsub(mesh-torus) is also set to "enable."
  If specification of a node allocation method is impossible or omitted, the node allocation method to use is determined by the settings of the job ACL items default mesh and default allocation-mode (see the table below).

| | | Value set for default allocation-mode | | |
|---|---|---|---|---|
| | | torus | mesh | noncont |
| Value set for default mesh | off | torus mode | mesh mode | non-contiguous mode |
| | on | mesh mode | mesh mode | non-contiguous mode |

- If the node=*shape* parameter is specified together with the vnode=*num* parameter (described below), which allocates virtual nodes to a node-sharing job, job submission encounters an error.

The following example shows the specification of a three-dimensional node shape for a job executing the MPI program prog_A. (The node dimensions are 4, 3, and 2 on the X-axis, Y-axis, and Z-axis, respectively.)

```
$ cat job.sh                                                              (1)
#!/bin/sh
export PATH=/opt/FJSVfxlang/<version>/bin:$PATH                           (2)
export LD_LIBRARY_PATH=/opt/FJSVfxlang/<version>/lib64:$LD_LIBRARY_PATH   (3)
mpiexec ./prog_A
$ pjsub -L "node=4x3x2" job.sh                                            (4)
```

(1) Job script job.sh that executes the MPI program prog_A
(2) Environment setting for executing the MPI job
(3) Environment setting for executing the MPI job
(4) Submission of the job with a 4x3x2 node shape specified

[FX100/FX10: Allocating virtual nodes (node-sharing job)]

To allocate virtual nodes to a node-sharing job, specify the number of CPU cores and the amount of memory for the virtual nodes. Use the vnode parameter for the -L or --rsc-list option of the pjsub command to specify the number of CPU cores and amount of memory per virtual node. The specified values must be within the scope of resources per node.

```
{-L | --rsc-list} vnode=[num][,vnode-core=num][,{core-mem=size|vnode-mem=size}]
or
{-L | --rsc-list} vnode=[num][([core=num][;core-mem=size|mem=size])]
```

Table 2.10 Parameters used for virtual node specification [FX100/FX10]

| Parameter | Description |
|---|---|
| [*num*] | Number of virtual nodes to allocate. (*) At present, only 1 can be specified.<br>The notation vnode= with *num* omitted has the same meaning as vnode=1. |
| vnode-core=*num*<br>core=*num* | Number of CPU cores per virtual node |
| vnode-mem=*size*<br>mem=*size* | Amount of memory per virtual node.<br>However, the specified value is rounded up in the following units. |

| Parameter | Description |
|---|---|
| | FX100 compute node: 32MiByte<br>FX10 compute node: 256MiByte<br>This parameter is used exclusively with the core-mem parameter. |
| core-mem=*size* | Amount of memory per CPU core.<br>The amount of memory for one virtual node is the value for the number of size * CPU cores rounded up in the following units.<br>FX100 compute node: 32MiByte<br>FX10 compute node: 256MiByte<br>This parameter is used exclusively with the mem and vnode-mem parameter. |

For the memory amount expression format, see "Table 2.8 Expression formats for amounts of resources."

```
[Example] The number of virtual nodes is 1, the number of CPU cores in a virtual node is 5, and the
memory amount per virtual node is 30Mi bytes.

$ pjsub -L "vnode=1,vnode-core=5,vnode-mem=30Mi" job.sh
or
$ pjsub -L "vnode=1(core=5;mem=30Mi)" job.sh
```

### Note

- The specification of unlimited for the core-mem, vnode-mem or mem parameter means that the job management function places no memory limitation on the job. The job can use as much memory as permitted by the OS. However, if another job begins to run on the same node, the job may run short of memory.
- Job submission encounters an error in the following cases.
  - The node or node-mem parameter (the parameter is for node-exclusive jobs) is specified together with the vnode parameter.
  - The value specified for the vnode parameter is other than 1.
  - The value specified for the vnode-core or core parameter is equal to or less than 0 or greater than the number of CPU cores within the node.
  - The core-mem, and vnode-mem or mem parameters are specified at the same time.
  - The memory size specified for the core-mem, vnode-mem or mem parameter exceeds the upper limit defined for the job ACL function.
  - An attempt is made with the vnode parameter to allocate a virtual node to a job that uses the staging function.

[PRIMERGY]

Node resources are specified with a number of nodes and a number of virtual nodes. Also, for a virtual node, specify the number of CPU cores and the amount of memory.

```
{-L | --rsc-list} "vnode=num[,vnode-core=size][,{core-mem=size|vnode-mem=size}][,node=num]"
or
{-L | --rsc-list} "vnode=num[([core=num][;core-mem=size|mem=size])][,node=num]"
```

Table 2.11 Node resource specification formats [PG]

| Format | Description |
|---|---|
| vnode=*num* | Specify the number of virtual nodes. |
| vnode-core=*num*<br>core=*num* | The value is the number of CPU cores per virtual node. |
| vnode-mem=*size*<br>mem=*size* | The value is the amount of memory per virtual node.<br>This parameter and core-mem are mutually exclusive. |

| Format | Description |
|---|---|
| core-mem=*size* | The value is the amount of memory per CPU core.<br>This parameter, vnode-mem, and mem are mutually exclusive. |
| node=*num* | Specify the number of nodes to allocate to the job. If the specified number of nodes is greater than the number of virtual nodes, the number of allocated nodes is only the number of virtual nodes.<br>This option is available when the virtual node placement policy is UNPACK (-P "vn-policy=unpack"). |

For the expression format for memory amount, see "Table 2.8 Expression formats for amounts of resources."

```
[Example] The number of virtual nodes is 1, the number of CPU cores in a virtual node is 5, and the
memory amount per virtual node is 30Mi bytes.

$ pjsub -L "vnode=1,vnode-core=5,vnode-mem=30Mi" job.sh
or
$ pjsub -L "vnode=1(core=5;mem=30Mi)" job.sh
```

**Note**

- For a job submitted to a PRIMERGY compute node, the parameter "vnode=*num*" must be specified.
- The parameter "vnode-mem=unlimited" or "mem=unlimited" means that the memory limitation to the job by the job management function does not work. The job can use the memory as much as OS permits. However, the memory for the job might be insufficient when other jobs run in the node.
  To avoid the lack of memory, select the job execution mode policy SIMPLEX (-P exec-policy=simplex) so that the job can occupy the node.
- vnode-mem and core-mem cannot be specified at the same time.

You can specify a node allocation concept (node selection policy) on PRIMERGY compute nodes. For details, see "2.3.4 Specifying a node selection policy [PG]."

### 2.3.2.3 Job operation when a job exceeds an upper limit on amount of resources

This section describes job operation in cases where the amount of resources used by a job exceeds a set upper limit during job execution.

Table 2.12 Operation of a job exceeding an upper limit on amount of resources

| Resource | Operation |
|---|---|
| node-cpu<br>elapse | The SIGXCPU signal is sent to all processes in the job.<br>Then, after 10 seconds elapse, the SIGKILL signal is sent to any of these processes still existing in the job in order to forcibly terminate them. The purpose of this 10-second delay is to allow for the execution of postprocessing of every process.<br>Therefore, implement processing as required by capturing the SIGXCPU signal from the job script and each process executed from the job script. |
| node-mem [FX100/FX10]<br>vnode-mem<br>mem<br>core-mem | - FX100 or FX10 compute nodes<br>The process memory acquisition request fails. (Example: malloc() returns NULL.)<br>- PRIMERGY compute nodes<br>The OS forcibly terminates the processes that request memory acquisition.<br>The subsequent job operation depends on the program and job script setup.<br>Note that the administrator may have configured the system to forcibly terminate jobs that use an amount of resources exceeding a set upper limit. For details on the settings of the system that you use, contact the administrator. |
| node-quota [FX10] | A file output error occurs in a job.<br>Post-error job operation depends on the program and job script setup. |

| Resource | Operation |
|---|---|
| proc-* | Upper limit values such as proc-core are upper limit values established by the OS (system call setrlimit()) for individual processes in a job.<br>Each of them corresponds to the following resources, which can be specified in the system call setrlimit():<br><br>proc-core: RLIMIT_CORE<br>proc-cpu: RLIMIT_CPU<br>proc-crproc: RLIMIT_NPROC<br>proc-data: RLIMIT_DATA<br>proc-lockm: RLIMIT_MEMLOCK<br>proc-msgq: RLIMIT_MSGQUEUE<br>proc-openfd: RLIMIT_NOFILE<br>proc-psig: RLIMIT_SIGPENDING<br>proc-filesz: RLIMIT_FSIZE<br>proc-stack: RLIMIT_STACK<br>proc-vmem: RLIMIT_AS<br><br>The specified upper limit values are set as software and hardware limits on these resources. Operation when a process exceeds an upper limit value follows OS specifications. For details, see setrlimit(2) in the man manual of Linux. Note that FX100 and FX10 compute node operation conforms to Linux specifications. |

The job resource limit values also apply to the prologue and epilogue processes set by the administrator (see "1.11 Prologue and Epilogue Function").
The prologue and epilogue processes are executed as a part of a job. Therefore, even when a user-created job script or program does not exceed a job resource limit value, its prologue or epilogue process may exceed the limit value.
For details on the amount of resources required for the prologue and epilogue processes, contact the administrator.

### 2.3.2.4 Specifying job statistical information output

You can output job statistical information as job execution results by using the -s or -S option of the pjsub command.

```
{ -s | -S } [ --spath output destination ]
```

The -S option outputs more detailed statistical information than the -s option. For details on their differences, see "A.2 Job Statistical Information."

The job statistical information is output to the job statistical information file "job name+.i+job ID" file in the current directory at the job submission time. To specify an output destination, specify a file name in the --spath option.

The following expression can be used in the file name.

Table 2.13 Notation for specifying the file name of the job statistical information file

| Notation | Meaning |
|---|---|
| %j | Job ID |
| %J | Sub job ID |
| %b | Bulk number |
| %s | Step number |
| %n | Job name |

If you want to receive job statistical information by e-mail instead of in a file, use the -m option.

```
-m {s | S}
```

In this case, the arguments s and S have the same meanings as the -s and -S options, respectively.

### 2.3.2.5 Specifying staging [FX10]

In a system that uses staging, designate the files required for job execution and the file output by the job as staging targets in the options of the pjsub command.

The files required for job execution are a job script, programs called from within the job script, and the input data required by the programs.

The following table lists the major pjsub command options for specifying the staging target files.

Table 2.14 How to specify the stage-in target files

| Specification format | Description |
|---|---|
| {-I \| --stgin } "*srcfile dstfile*"<br>or<br>{-I \| --stgin } '*srcfile dstfile*' | Stages in the *srcfile* file on the login node as the *dstfile* file on the compute node.<br>The file *dstfile* is the relative path to the directory where the job starts. |
| {-I \| --stgin } "*srcfile dstdir*/"<br>or<br>{-I \| --stgin } '*srcfile dstdir*/' | Stages in the *srcfile* file on the login node to the *dstdir* directory on the compute node.<br>The directory *dstdir* is the relative path to the directory where the job starts.<br>Note: This specification format requires a slash (/), indicating a directory, after the *dstdir* directory. |
| {-D \| --stgin-dir } "*srcdir dstdir*"<br>or<br>{-D \| --stgin-dir } '*srcdir dstdir*' | Stages in files in the *srcdir* directory on the login node to the *dstdir* directory on the compute node.<br>The directory *dstdir* is the relative path to the directory where the job starts. |
| --stgin-basedir *basedir* | Specifies the base directory, representing the relative path of the stage-in target file *scrfile* and directory *srcdir*. Use it in combination with the --stgin or --stgin-dir option.<br>Stated differently, the *basedir*/*srcfile* file and the *basedir*/*srcdir* directory are the stage-in targets.<br>If this option is omitted, the current directory when the job was submitted is the base directory. |

Table 2.15 How to specify the stage-out target files

| Specification format | Description |
|---|---|
| {-O \| --stgout } "*srcfile dstfile*"<br>or<br>{-O \| --stgout } '*srcfile dstfile*' | Stages out the *srcfile* file on the compute node as the *dstfile* file on the login node. |
| {-O \| --stgout } "*srcfile dstdir*/"<br>or<br>{-O \| --stgout } '*srcfile dstdir*/' | Stages out the *srcfile* file on the compute node to the *dstdir* directory on the login node.<br>Note: This specification format requires a slash (/), indicating a directory, after the *dstdir* directory. |
| {-E \| --stgout-dir } "*srcdir dstdir*"<br>or<br>{-E \| --stgout-dir } '*srcdir dstdir*' | Stages out the *srcdir* directory on the compute node as the *dstdir* directory on the login node. |
| --stgout-basedir *basedir* | Specifies the base directory of the stage-out destination. Use it in combination with the --stgout or --stgout-dir option.<br>Stated differently, the *basedir*/*srcfile* file and the *basedir*/*srcdir* directory are the stage-out destinations.<br>If this option is omitted, the current directory when the job was submitted is the base directory. |

Note

You can specify multiple staging target files *srcfile* or directories *srcdir* by delimiting them with a space.
Note that the staging target (*srcfile* or *srcdir*) and the staging destination (*dstfile* or *dstdir*/) in the options are enclosed in single or double quotation marks so that they are specified as a single character string.

```
[Correct example]
--stgin "srcfile1 srcfile2 srcfile3 dstdir/"

[Incorrect example]
--stgin srcfile1 srcfile2 srcfile3 dstdir/
```

When writing files in a job script, you can write each file on a single line.

```
#PJM --stgin "srcfile1 dstdir/"
#PJM --stgin "srcfile2 dstdir/"
#PJM --stgin "srcfile3 dstdir/"
```

The descriptions in the following example show job submission options used with staging.

```
[Staging conditions]
Staging target directories: Locations in /G
Stage-in target files: a.out, in.dat, and param.txt in the /G/UserA directory
Stage-in destination: Current directory (.) at the job execution start time
Stage-out target file: out.dat in the job execution directory
Stage-out destination: Locations in the /G/UserA directory

[Example of staging coding]
#PJM --stgin-basedir /G/UserA --stgin "./a.out ./in.dat ./param.txt ."
#PJM --stgout "./out.dat /G/UserA"
```

For MPI programs, staging specifications must consider ranks. For details on how to specify it, see "2.3.5.8 Using a rank number directory [FX10]."

When specifying the target files for staging, you can specify the staging target files in detail by using the notation shown below.

Table 2.16 Notation for specifying staging targets

| Purpose | Notation |
|---|---|
| Job ID | %j |
| Sub job ID | %J |
| Bulk number | %b |
| Step number | %s |
| Job name | %n |
| Path name expansion | *: Matches any string containing a space character.<br>? : Matches any single character.<br>[*abc*]: Matches any character in parentheses.<br>Note: This conforms to the bash shell specifications. |

## Note

- The notation %j, %J, %b and %s cannot be used for the --stgout-basedir option.
- The notation *, ?, [, ] and the space character cannot be used for the file name *dstfile* or the directory name *dstdir* as stage-out destination. To use these characters, escape them by the backslash (\).

The following example uses the above notation.

```
[Example] Staging out the job output file out.data as [<job name>.out.data]
#PJM --stgout "./out.data %n.out.data"

[Example] Staging out the output file outdata.ID with a job ID attached
#PJM --stgout "./outdata.%j ./"
```

In the system that uses the staging function, the job cannot usually be accessed to a global file system by the compute node.

The option --use-global of the pjsub command is necessary for the job that accesses a global file system.

```
$ pjsub --use-global job.sh
```

The path names of the global file systems are set for each resource unit on which the job is executed. Confirm the path name to the administrator. The administrator has not occasionally permitted --use-global option by job ACL function. Confirm "pjsub(--use-global)" line of item "execute" in the output of the pjacl command.

When the staging function is not used, the global file system can be accessed on the compute nodes. Therefore, the --use-global option is not necessary at job submission.

### Note

- In the system in which the staging function is available, the job is staged by default. To submit a job which is not staged, specify the option --no-stging to the pjsub command.
  However, the system might be configured to stage the job always, or the administrator might configure the system not to be able to specify the option --no-stging.
  For details on the configurations and settings of the system that you use, contact the administrator.
- If you submit any of the following jobs in an environment where the staging function is available, an error occurs:
  - Node-exclusive job in mesh mode or con-contiguous mode
  - Node-sharing job for which virtual node allocation (vnode=) is specified

  To submit such a job, specify the --no-stging option.

## 2.3.2.6 Specifying automatically re-execution for a job

When submitting a job, you can specify whether to automatically re-execute any job aborted during execution, such as because of a system failure, by using the --restart or --norestart option of the pjsub command.

Table 2.17 Options specifying whether a job can be executed again

| Option | Description |
|---|---|
| --restart | Automatically re-executes the job if it ended abnormally. |
| --norestart | Does not automatically re-execute the job, even if it ended abnormally. |

### Note

- If neither of these options is specified, operations in this case may change with the system settings. Ask the administrator about these operations on the system used by users.
- Interactive jobs cannot be automatically re-executed. If these options are specified for these jobs, they are ignored.

## 2.3.2.7 Specifying an execution start time

Normally, the jobs submitted by users are executed as soon as possible, with the execution order determined according to resource availability and priority. However, users can specify execution start times.

To specify an execution start time, use the --at option of the pjsub command in the following format.

| Format | Description |
|---|---|
| --at *YYYYMMDD*[*hhmm*] | *YYYY* is the year, *MM* is the month, and *DD* is the day.<br>*hh* is the hour, and *mm* is the minute. If *hhmm* is omitted, 00:00 is assumed specified.<br>The specification in seconds is not available. |

The following example shows the submission of a job with the execution start time specified.

```
$ pjsub --at 201008011511 job.sh   (*)Execution of job.sh starts at 15:11 on August 1, 2010.
```

### Note

- Execution of a job with a specified execution start time will not start before the specified time even if resources are available. Also, depending on the availability of resources, the job may be executed later than the specified time.
- An interactive job cannot have a specified execution start time. If one is specified, it is ignored.

### Information

The job operation software adjusts the start of the staging process of a job so that the staging completes by the job execution start time. The job might be executed after the execution start time, even if the staging process is not completed at the time.

## 2.3.2.8 Specifying a job priority

Users can set execution priority for only the jobs that they submitted. Job priority is specified in the -p option of the pjsub command. The specified priority is an integer in a range of 0 to 255.

```
$ pjsub -p priority job.sh
```

The lowest priority is 0, and the highest is 255.

### See

Users can check the priorities of their own jobs by using the -v option of the pjstat command. For details, see "2.4.1 Displaying a job list."

### Information

- Any jobs that have the same priority are executed in order of submission.
- If the -p option is not specified, the priority is determined according to the administrator's settings.

## 2.3.2.9 Specifying the standard output and standard error output files of a batch job

The standard output and standard error output of a batch job are output to files.
The file names are automatically determined from the job name and job ID. (See "2.6.1 Referencing job execution results.")

The -o option and -e option of the pjsub command are used by users who want to specify the file names. To direct the standard error output to the standard output, specify the -j option.

Table 2.18 Specifying the standard output and standard error output files of a batch job

| Format | Description |
|---|---|
| -o *filename* | Outputs the standard output of the job to *filename* file. |
| -e *filename* | Outputs the standard error output of the job to *filename* file. |
| -j | Directs the standard error output of the job to the standard output of the job. |

You can use the following notation to specify the output file names.

| Notation | Meaning |
|---|---|
| %j | Job ID |

| Notation | Meaning |
|---|---|
| %J | Sub job ID |
| %b | Bulk number |
| %s | Step number |
| %n | Job name |

The following example shows the specification of output file names using the above notation.

```
$ pjsub -o '%j[%b].stdout' -e '%j[%b].stderr'
```

In this example, a sub job with bulk number 10 in a bulk job with job ID 100 has the following file names. The standard output file is 100[10].stdout, and the standard error output file is 100[10].stderr.

### Note

If individual sub jobs in a bulk job or step job have the same file specified for their output destinations, the file will contain a mixture of output from each sub job. Consequently, the output from each sub job may not be readable in the output results in the file.

### Information

The output destination files of the standard output and standard error output for an interactive job cannot be specified in the -o and -e options of the pjsub command.

## 2.3.3 How to submit each type of job

Jobs other than normal jobs require specific specifications according to the type of job. This section describes differences between specification methods by type of job.

### 2.3.3.1 How to submit a bulk job

Specify the --bulk option of the pjsub command to declare the job as a bulk job.

Also, indicate the number of sub jobs you want to submit as a bulk job by using the --sparam option with start and end values for the bulk number.

```
$ pjsub --bulk --sparam "StartingBulkNumber-EndingBulkNumber" [jobscript]
```

You can specify a value from 0 to 999999 for a bulk number. The ending bulk number must be greater than the starting bulk number. The bulk number is incremented by one.

### Note

The upper limit on the number of sub jobs handled in one bulk job is 65535.

The following example shows the submission of a bulk job.

```
Program executed with each of ten data files, in-0.dat to in-9.dat, as input
$ cat bulkjob.sh
#!/bin/sh
...
INFILE=in-${PJM_BULKNUM}.dat                    (1)
OUTFILE=out-${PJM_BULKNUM}.dat                  (2)
./program ${INFILE} ${OUTFILE}                  (3)
...
```

```
$ pjsub --bulk --sparam "0-9" bulkjob.sh       (4)
```

(1) Determines the input data file name from the bulk number.
(2) Determines the output data file name from the bulk number.
(3) Specifies the input/output data files in the arguments of the program.
(4) Submits the sub job bulkjob.sh with the bulk number from 0 to 9.

The above example shows only the part related to the bulk job.

### 2.3.3.2 How to submit a step job

Specify the --step option of the pjsub command to declare the job as a step job.
For the step job, specify the job ID or the job name. It indicates which the sub job is associated with the step job.

#### [Method of specifying the job ID]

Submit a sub job with the job ID of the step job. In this case, the options of the first sub job differ from the options of the second and subsequent sub jobs.

1. Submission of the first sub job

```
$ pjsub --step stepjob1.sh
[INFO] PJM 0000 pjsub Job 100_0 submitted.
```

2. Submission of the second and subsequent sub jobs

   Specify a job ID with the --sparam "jid=" option to indicate that the job is associated with the first sub job.

```
$ pjsub --step --sparam "jid=100" stepjob2.sh
[INFO] PJM 0000 pjsub Job 100_1 submitted.
```

   If the first sub job has already ended, the sub job submission will fail because the corresponding job does not exist.

#### [Method of specifying the job name]

Make the sub job name same and submit sub job with the job name specified. In this method, the first job and the jobs after that can be submitted by same way basically.

1. Submission of the first sub job

   Set the job name with the --sparam "jnam=" option when submitting a sub job.

```
$ pjsub --step --sparam "jnam=mystepjob" stepjob1.sh
[INFO] PJM 0000 pjsub Job 200_0 submitted.
```

2. Submission of the second and subsequent sub jobs

   The --sparam "jnam=" option is for setting the job name, and means that the sub job is associated with an existing step job of same name.

```
$ pjsub --step --sparam "jnam=mystepjob" stepjob2.sh
[INFO] PJM 0000 pjsub Job 200_1 submitted.
```

   If the step job which has the corresponding job name does not exist, new step job will be created.

Note the following when you use the --sparam "jnam=" option.

- If --sparam "jnam=" option is specified, the job name specified with the option is set to the sub job regardless of the -N or --name option.
- If the --sparam "jnam=" option is specified to the pjsub command, --sparam "jnam=" option in the job script is ignored.

- If you submit a sub job with the job name for the step job which has been submitted without --sparam "jnam=" option, specify the job name of the first sub job (step number 0).
- If you specify the job name with -N or --name option, not with the --sparam "jnam=" option, new step job will be created. Therefore, two or more step jobs of the same job name might exist. If you submit a sub job specified job name with --sparam "jnam=" option, the sub job is assumed to be associate with the latest step job.

For the submission order, the step number of a submitted job is incremented by one from 0, unless otherwise specified. The user can specify a step number in the --sparam "sn=" option.

```
$ pjsub --step --sparam "sn=10" stepjob1.sh   # The step number is set to 10.
```

When the user specifies a step number, the number must be greater than the largest step number at that point. You can check the step number by using the pjstat command. (See "2.4 Checking the Job Status.")

You can submit multiple sub jobs for one step job specified in arguments of pjsub command by delimiting with commas or blanks.

```
$ pjsub --step stepjob1.sh,stepjob2.sh      (or pjsub --step stepjob1.sh stepjob2.sh)
[INFO] PJM 0000 pjsub Job 300_0 submitted.
[INFO] PJM 0000 pjsub Job 300_1 submitted.
```

In the above example, the step number is set as 0, 1,... in the order of specification of the job scripts. If the --sparam "sn=" option is specified, the setting of step numbers begins with the specified value.
You can also submit multiple sub jobs for an already existing step job.

```
$ pjsub --step stepjob1.sh
[INFO] PJM 0000 pjsub Job 400_0 submitted.
$ pjsub --step --sparam "jid=400" stepjob2.sh,stepjob3.sh
 (or pjsub --step --sparam "jid=400" stepjob2.sh stepjob3.sh)
[INFO] PJM 0000 pjsub Job 400_1 submitted.
[INFO] PJM 0000 pjsub Job 400_2 submitted.
```

## Note

Note the following about submitting multiple sub jobs.

- The option specified by the argument of the pjsub command is applied to all sub jobs that are submitted at the same time. Specify the option in the job scripts, if you want to specify different option for each job.
  Note the following when you specify the job name, the job ID and the step number.
  - If you want to set different job name for each sub job, specify the -N or --name option in each job script. When the --sparam "jnam=" option is specified in the job script, the --sparam "jnam=" option needs to be specified in all other scripts which are submitted at same time. Also their job name need to be same.
  - The option in the first job script is effective, if you specified the --sparam "jid=" options in the plural job scripts which are submitted at same time.
  - The --sparam "sn=" option specified in an argument of the pjsub command represents the step number for the first sub job. The step number of following sub jobs is incremented by one.
- If you want to specify the standard output file, the standard error output file or the job statistical information file for the sub job, these files need to be different file names for each sub job. For example, the step number is used for format of their file names. If these files are same name, the output of the latest sub job overwrites them.

## Information

When the comma (,) is in the job script name, you need to specify "none" with the --script-delimiter option not to be interpreted as a separator.

```
$ pjsub --script-delimiter none --step step,job1.sh
```

In the submission of the second and subsequent sub jobs, use the --sparam "sd=" option to specify an operation based on the results of the preceding sub job. It is called the "dependency expression" of a step job.

The dependency expression is specified in the following format.

```
--sparam "sd=form[:[deletetype][:stepno[:stepno[...]]]]"
```

*form* represents a condition used to determine whether to execute a sub job that is submitted. *deletetype* represents the detailed operation performed if that sub job is not executed. The *form* condition applies to the execution results of a sub job, and *stepno* represents the step number of this sub job. If *stepno* is omitted, the results of the last sub job become the target.

The following tables list values that can be specified in *form* and *deletetype*.

| *form* | Descriptions |
|---|---|
| NONE | Indicates that there is no dependent sub job. It is the same as not specifying "sd=".<br>Each sub job that is submitted is executed when the preceding sub job ends.<br>Note: Any subsequent sub job deleted by a preceding sub job will not be executed. |
| *param*==*value*<br>*param*!=*value*<br>*param*>*value*<br>*param*<*value*<br>*param*>=*value*<br>*param*<=*value* | Indicates conditions for deleting (not executing) a sub job according to the *deletetype* specification.<br>*param* is ec or pc. The meaning is as follows.<br>ec: End status of the job script of a dependent sub job<br>pc: Job end code (PJM code) of a dependent sub job<br>You can specify multiple values for the conditions == and != by delimiting them with a comma (,). |

## Information

**Difference between the job script end status (ec) and the job end code (PC)**

Job end code (pc) shows whether the job operation management function had processed the job normally. Even if the exit status of the job script (ec) is not 0, the job end code is 0 when the job is processed normally.
The job end code might not be 0 though the exit status of the job script is 0 when the job could not be processed normally (ex. exceeding elapse time limit).
For this reason, users need to use these values according to their respective needs. For the meanings of the values of the job end code (pc), see the description of the "PC" item in "Table A.2 Output items of the pjstat command (items added at specification of -v option)" in "A.1 Output of the pjstat and pjstat -v."

| *deletetype* | Description |
|---|---|
| one | Deletes only this sub job.<br>The subsequent sub jobs that are dependent on the results of this sub job are not deleted.<br>If *deletetype* is omitted, one is assumed specified. |
| after | Deletes this sub job and only the subsequent dependent sub jobs. |
| all | Deletes this sub job and all subsequent sub jobs. |

The following example shows the submission of a sub job for a step job with job ID 500.
In this example, if the end code of the job script of the sub job with step number 0 is not 0, the subsequent sub jobs are not executed.

```
$ pjsub --step --sparam "jid=500,sd=ec!=0:all:0" stepjob2.sh
```

## Note

Enclose the arguments of the --sparam option in single or double quotation marks so that they are recognized as a single character string.

You may want a sub job to anticipate the completion of stage-out of the previous sub job. An example is a sub job that uses the output results of the previous sub job as input.
In such cases, use the --sparam "send=" option. If omitted, "send=yes" is assumed specified.

| | |
|---|---|
| send=yes | The sub job that is submitted is not staged in until stage-out of the previous sub job has completed.<br>Note: yes is interchangeable with YES, Y, and y. |
| send=no | Stage-in of the sub job that is submitted starts without waiting for stage-out of the previous sub job to complete.<br>Note: no is interchangeable with NO, N, and n. |

## Note

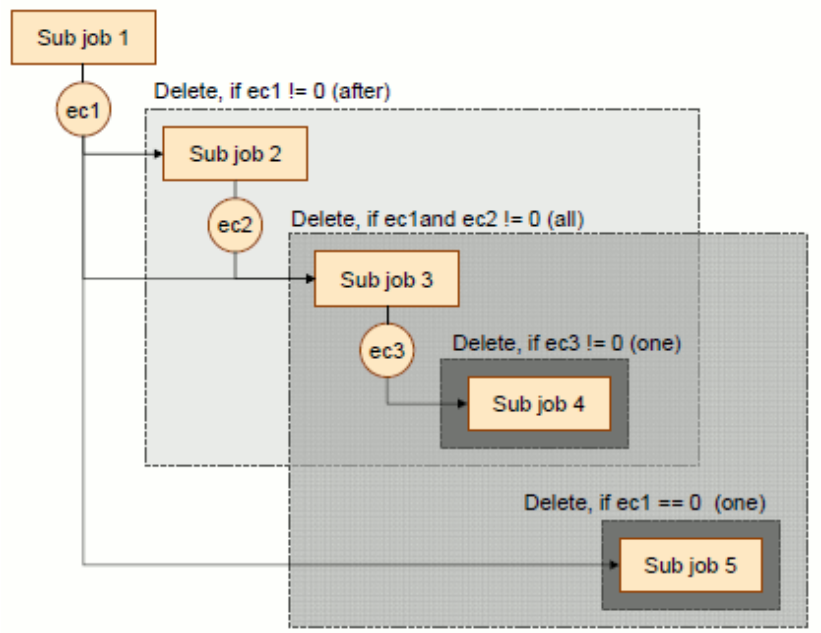
If a sub job of a step job treats the output results of the preceding sub job as the stage-in target file, the target file does not exist when the sub job is submitted.
If the specified stage-in target file does not exist, the pjsub command returns an error when submitting the sub job.
Therefore, when submitting such a sub job, specify confirm=no as shown below to skip the check for the stage-in target file.

```
$ pjsub --step ... --stgin "srcfile dstdir/ confirm=no" ...
```

For details on the staging-related options, see the man page of the pjsub command.

The following is a specific example of a step job.

Figure 2.5 Example of a step job




- Sub job 1

  The sub job is executed first.

- Sub job 2

  If the end code of sub job 1 is not 0, sub job 2 and all other jobs dependent on sub job 2 are deleted. Sub jobs 3 and 4 are affected by the deletion of sub job 2, and they are also deleted.

- Sub job 3

  If the end codes of sub jobs 1 or 2 are not 0, sub job 3 and all subsequent jobs are deleted. Sub jobs 4 and 5 are also deleted.

- Sub job 4

  If the end code of sub job 3 is not 0, only sub job 4 is deleted. Sub job 5 is not deleted.

- Sub job 5

  If the end code of sub job 1 is 0, only sub job 5 is deleted.

The following shows this submitted step job.

```
$ pjsub --step --sparam "sn=1" job1.sh                                  (1)
[INFO] PJM 0000 pjsub Job 500_1 submitted.                              (2)
$ pjsub --step --sparam "jid=500,sn=2,sd=ec!=0:after:1" job2.sh         (3)
$ pjsub --step --sparam "jid=500,sn=3,sd=ec!=0:all:1:2" job3.sh
$ pjsub --step --sparam "jid=500,sn=4,sd=ec!=0:one:3" job4.sh
$ pjsub --step --sparam "jid=500,sn=5,sd=ec==0:one:1" job5.sh
```

(1) Submitted as step number 1
(2) Job ID 500 in this example
(3) Submitting step number 2 and subsequent steps below

You can submit the sub jobs of a single step job to different resource units or resource groups. When doing this, you need to specify the names of the resource units or resource groups to which the sub jobs are to be submitted. If you omit this specification, the sub jobs will be submitted to the default resource unit or resource group that is set in the job ACL function.
The following example submits the sub jobs with step numbers 1 and 3 to the resource unit rscunitPG and the sub job with step number 2 to the resource unit rscunitFX:

```
$ pjsub --step -L "rscunit=rscunitPG" --sparam "sn=1" job-PG1.sh
[INFO] PJM 0000 pjsub Job 600_1 submitted.
$ pjsub --step -L "rscunit=rscunitFX" --sparam "jid=600,sn=2,sd=ec!=0:after:1" job-FX.sh
[INFO] PJM 0000 pjsub Job 600_2 submitted.
$ pjsub --step -L "rscunit=rscunitPG" --sparam "jid=600,sn=3,sd=ec!=0:all:1:2" job-PG2.sh
[INFO] PJM 0000 pjsub Job 600_3 submitted.
```

### 2.3.3.3 How to submit a workflow job

Workflow job is a method of controlling the submission of multiple jobs by a user. User creates the shell script that submits multiple jobs, and executes it as shown in "1.2.1.4 Workflow job."

### 2.3.3.4 How to submit an interactive job

Specify the --interact option of the pjsub command to declare the job as an interactive job.

**Note**

The interactive job is able to submit only on the login node.

With an interactive job, the user can interactively enter the job contents from a terminal. Alternatively, the user can specify the job contents in the job script in the same way as for a batch job.

If no job script is specified in the arguments of the pjsub command, interactive input of the job contents is assumed, and the shell is started on pseudo terminal and waits for input.

```
$ pjsub --interact
[INFO] PJM 0000 pjsub Job 405916 submitted.             (1)
[INFO] PJM 0081 .connected.                             (2)
[INFO] PJM 0082 pjsub Interactive job 405916 started.   (3)
$                                                       (4)
...
$ exit                                                  (5)
[INFO] PJM 0083 pjsub Interactive job 405916 completed. (6)
```

(1) Message indicating interactive job submission
(2) Message indicating that the interactive job is being prepared
(3) Message indicating the start of the interactive job
(4) Shell prompt in the interactive job
(5) Exit from the shell
(6) Message indicating the end of the interactive job

The prompt in this example is that displayed by the shell on the compute node that executes the interactive job. The path of the current directory on the compute node is the same as that of the current directory on the login node when the pjsub command was executed.

The interactive job ends upon exit from the shell or when the job is deleted by the pjdel command. (See "2.5.1 Deleting a job.")

The input from a pseudo terminal is passed to the shell in the interactive job. To direct the special operation to the interactive job, use a tilde sign '~' as an escape character. The following operation can be directed by the tilde sign and the input following it. The tilde sign as the escape character should be the first character in the new line.

Table 2.19 Escape characters in the interactive job

| Input | Action |
|---|---|
| ~~ | Sends the tilde sign to the shell |
| ~. | Terminates the interactive job. The shell that was executed in the interactive job is terminated, and the pseudo terminal is closed. |
| ~<CTRL-z> | Suspend the pjsub command that executes interactive job, and back to the pseudo terminal in which the pjsub command is executed. |
| ~? | Display the list of the escape characters. |

If the job script is specified in the arguments of the pjsub command, the job is executed according to the job script contents in the same way as a batch job.

```
$ pjsub --interact interactjob.sh
[INFO] PJM 0000 pjsub Job 405918 submitted.
[INFO] PJM 0081 .connected.
[INFO] PJM 0082 pjsub Interactive job 405918 started.
...                                          <- Job script output contents
[INFO] PJM 0083 pjsub Interactive job 405918 completed.
```

Immediately after the interactive job is submitted, computer resources are allocated, and the job is executed. If the job cannot be executed immediately because of insufficient resources, it waits for allocation of resources. This may affect the execution of subsequent jobs.

To avoid this issue, you can specify a maximum wait time (seconds) for resource allocation for an interactive job by using the --sparam "wait-time=" option. If the resource allocation wait time exceeds the specified time, the job is canceled.

```
$ pjsub --interact --sparam "wait-time=600"                                    (1)
[INFO] PJM 0000 pjsub Job 291 submitted.
[INFO] PJM 0081 .
[INFO] PJM 0080 pjsub Interactive job 291 is canceled due to the resource allocation timeout.
       The timeout period "t" can be specified by "--sparam wait-time=t".      (2)
```

(1) The maximum wait for resource allocation is 600 seconds (10 minutes).
(2) The job was canceled because the wait time exceeded 600 seconds.

The following table lists the pjsub command options that are ignored in an interactive job.

Table 2.20 pjsub command options ignored in an interactive job

| Option | Description |
|---|---|
| --at | Job execution start time |
| -e | Standard error output file |
| -j | Directing the standard error output to the standard output file |
| --bulk, --step | Job model |
| -m e | E-mail notification of execution end |
| -m r | E-mail notification of execution restart |
| --restart | Enable automatically re-execution of a job |
| -o | Standard output file |
| -p | Job priority |

| Option | Description |
|---|---|
| -w | Wait for the pjsub command to return |
| Options related to staging | See "2.3.2.5 Specifying staging [FX10]." |

### 2.3.3.5 How to submit an emergency job [FX100]

To submit a job as an emergency job for FX100 compute nodes, use the pjsub command with the --enforce option specified.

```
$ pjsub --enforce emerg.sh
```

Only batch-type normal jobs can be submitted as emergency jobs.

Table 2.21 Jobs that can be submitted as emergency jobs

| | |
|---|---|
| Jobs that can be submitted as emergency jobs | Job type: Batch jobs<br>and<br>Job model: Normal jobs<br><br>You can submit only a node-exclusive job as an emergency job in torus mode or mesh mode. |
| Jobs that cannot be submitted as emergency jobs | Jobs other than the above.<br>That is,<br>Job type: Interactive jobs<br>or<br>Job model: Step jobs, bulk jobs |

Note

- Emergency jobs can be submitted only by users with privileges of job operation administrator or higher, by users who have been granted submission privileges by an administrator via the job ACL function.
- The --enforce and --at options of the pjsub command are mutually exclusive.
  This means that you cannot specify an execution time for an emergency job.
- Even an emergency job is not executed immediately in the following cases:
  - There are insufficient physical resources due to a node failure etc.
  - A previously submitted emergency job uses compute nodes, and the emergency job submitted subsequently cannot secure the compute nodes.
  - Stop processing is being performed for a running job.
  - The assignment time scheduled for an emergency job overlaps the deadline schedule period(*).
    (*) It is a period where jobs are not to be executed because of maintenance etc. This period is set by the administrator.

  In the above cases, the submitted emergency job is placed in the QUEUED state and waits for the resources to become available. If the user does not have the emergency job submission privilege, the pjsub command returns an error upon job submission.
- If you specify "unlimited" for the memory usage limit value (node-mem) for an emergency job, you should pay careful attention to the amount of available memory.
  Setting the memory usage limit value to "unlimited" for a job means that the memory available on a node at the time of execution is used. This also applies to an emergency job.
  If, however, there are running jobs for which the memory usage limit value is set to other than "unlimited," that secured amount of memory cannot be used for the emergency job.
  If you want to secure an amount of memory that can reliably be used by an emergency job, specify a specific value other than "unlimited."

Information

The following limits are not applied to an emergency job:

- Concurrent job acceptance limit
- Concurrent job execution limit
- Concurrent node usage limit

## 2.3.4 Specifying a node selection policy [PG]

On PRIMERGY compute nodes, you can specify a node selection policy in addition to node resources. (See "1.7.3 Virtual node allocation on PRIMERGY compute nodes.")

### 2.3.4.1 Virtual node placement policy

The virtual node placement policy is specified by the -P "vn-policy=" option of the pjsub command.

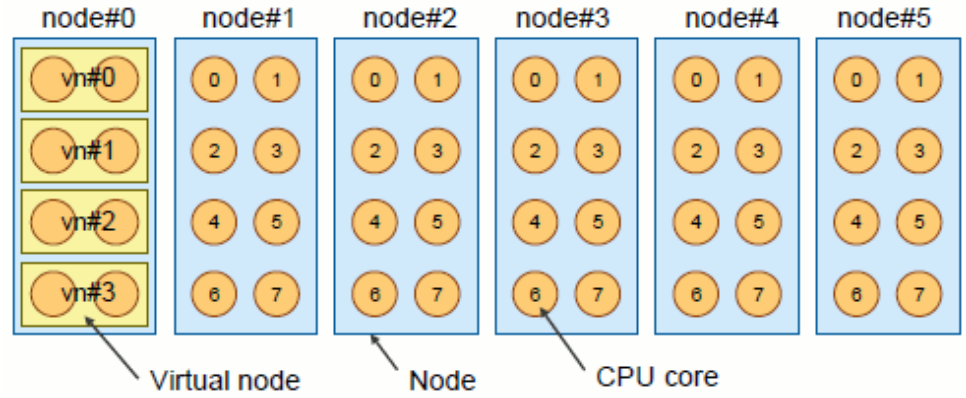
Absolutely PACK

To always place all virtual nodes on one node, specify the -P "vn-policy=abs-pack" option.

The following example shows placement with Absolutely PACK of four virtual nodes having two CPU cores.

```
$ pjsub -L "vnode=4,vnode-core=2" -P "vn-policy=abs-pack" job.sh
or
$ pjsub -L "vnode=4(core=2)" -P "vn-policy=abs-pack" job.sh
```

Figure 2.6 Virtual node placement with Absolutely PACK



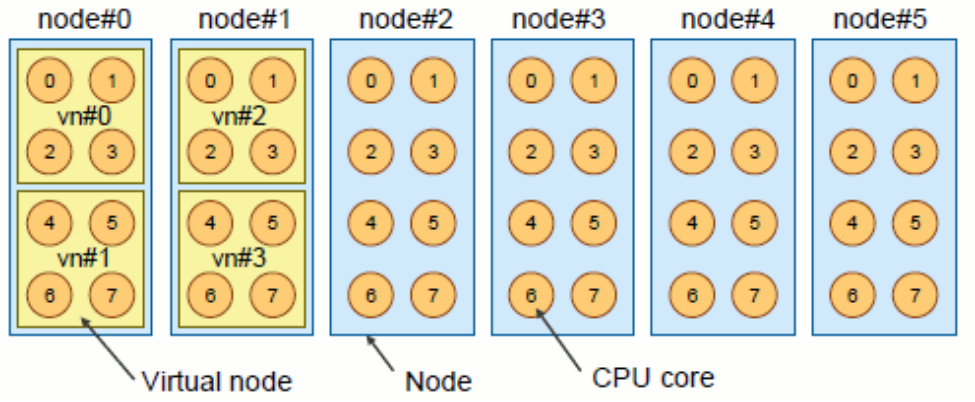

PACK

To place virtual nodes on only one node if at all possible, specify the -P "vn-policy=pack" option.

The following example shows placement with PACK of four virtual nodes having four CPU cores.

```
$ pjsub -L "vnode=4,vnode-core=4" -P "vn-policy=pack" job.sh
or
$ pjsub -L "vnode=4(core=4)" -P "vn-policy=pack" job.sh
```

Figure 2.7 Virtual node placement with PACK




In the above example, two nodes are used because only two virtual nodes can be placed on one node.

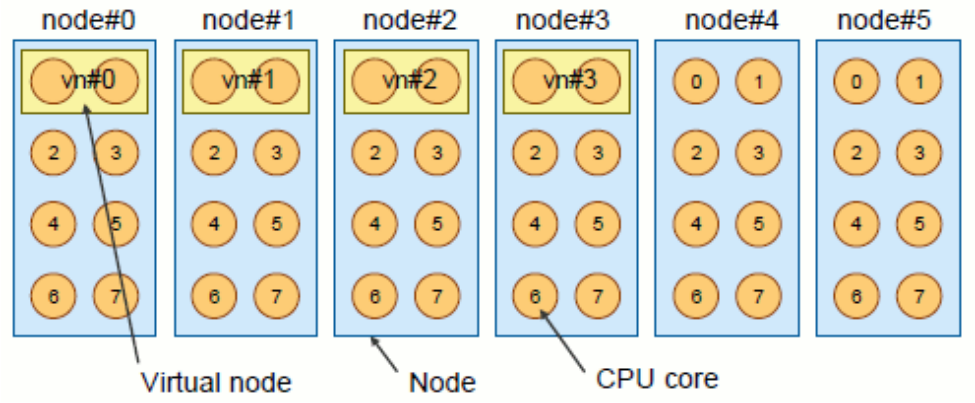
Absolutely UNPACK

To always place each virtual node on different nodes, specify the -P "vn-policy=abs-unpack" option.

The following example shows placement with Absolutely UNPACK of four virtual nodes having two CPU cores.

```
$ pjsub -L "vnode=4,vnode-core=2" -P "vn-policy=abs-unpack" job.sh
or
$ pjsub -L "vnode=4(core=2)" -P "vn-policy=abs-unpack" job.sh
```

Figure 2.8 Virtual node placement with Absolutely UNPACK

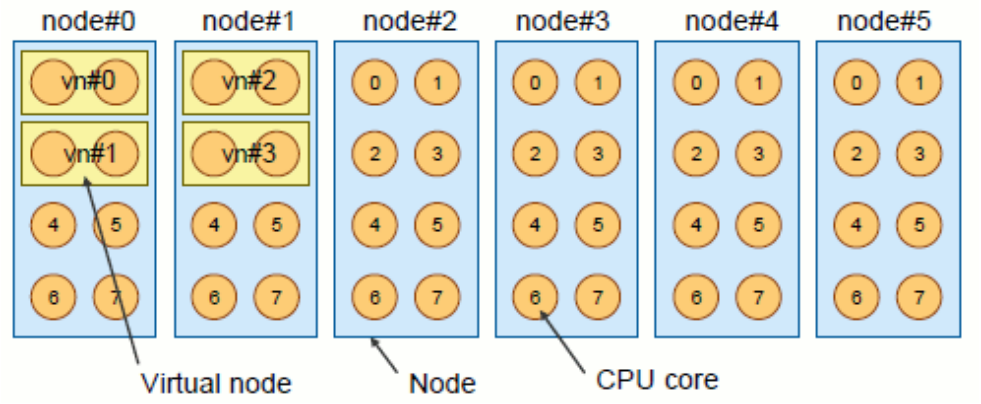



With Absolutely UNPACK, you can also define a specification where *N* (number) virtual nodes is always placed on different nodes. In this case, specify the -P "vn-policy=abs-unpack=*N*" option.

The following example shows placement with Absolutely UNPACK of four virtual nodes having two CPU cores, with two virtual nodes per node.

```
$ pjsub -L "vnode=4,vnode-core=2" -P "vn-policy=abs-unpack=2" job.sh
or
$ pjsub -L "vnode=4(core=2)" -P "vn-policy=abs-unpack=2" job.sh
```

Figure 2.9 Placement of two virtual nodes per node with Absolutely UNPACK




Note

The pjsub command with "vnode=*M*" (the number of virtual nodes) and "vn-policy=abs-unpack=*N*" specified returns an error if *M* is not a multiple of *N*.

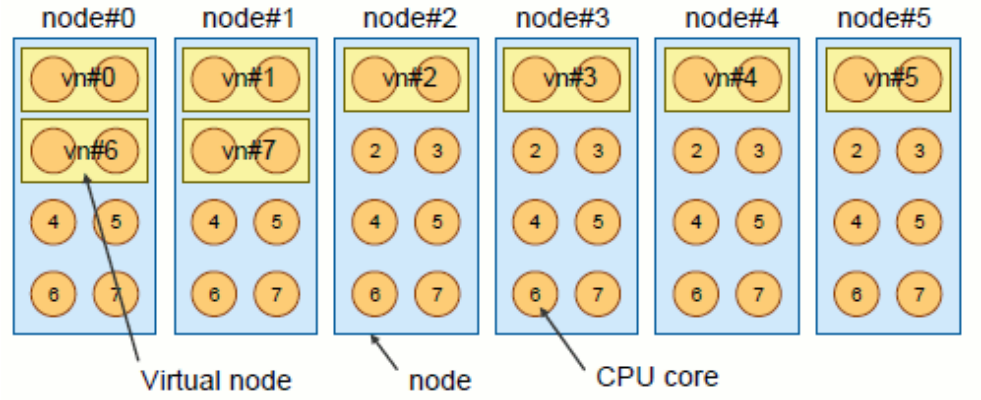
UNPACK

To place each virtual node on only different nodes if at all possible, specify the -P "vn-policy=unpack" option.

The following example shows placement with UNPACK of eight virtual nodes having two CPU cores.

```
$ pjsub -L "vnode=8,vnode-core=2" -P "vn-policy=unpack" job.sh
or
$ pjsub -L "vnode=8(core=2)" -P "vn-policy=unpack" job.sh
```

Figure 2.10 Virtual node placement with UNPACK

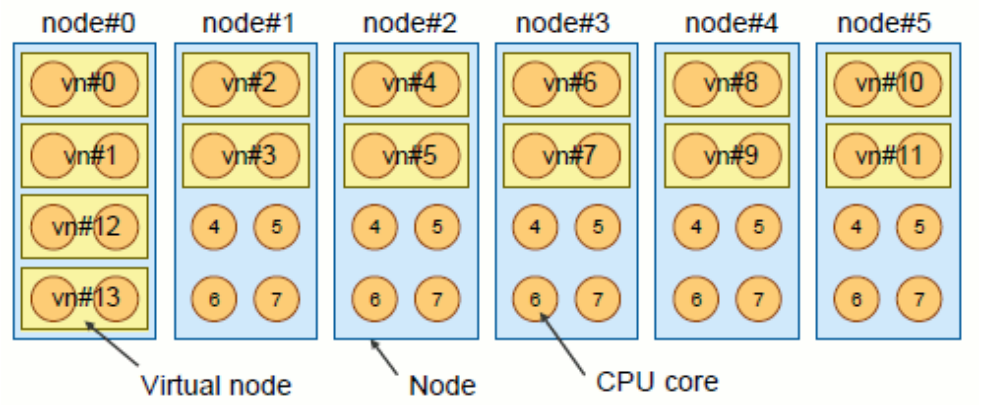



With UNPACK, you can also define a specification where N (number) virtual nodes are placed on only different nodes if at all possible. In this case, specify the option -P "vn-policy=unpack=*N*".

The following example shows placement with UNPACK of 14 virtual nodes having two CPU cores, with only two virtual nodes per node if at all possible.

```
$ pjsub -L "vnode=14,vnode-core=2" -P "vn-policy=unpack=2" job.sh
or
$ pjsub -L "vnode=14(core=2)" -P "vn-policy=unpack=2" job.sh
```

Figure 2.11 Placement of two virtual nodes per node with UNPACK



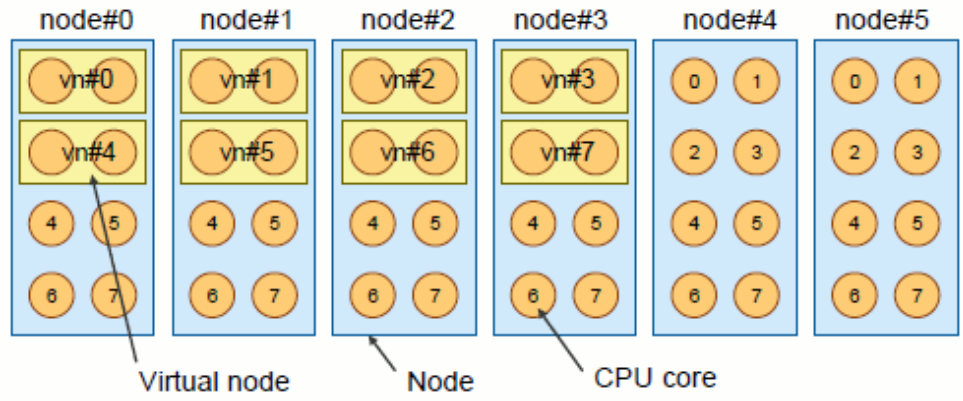

## Note

The pjsub command with "vnode=*M*" (the number of virtual nodes) and "vn-policy=unpack=*N*" specified returns an error if *M* is not a multiple of *N*.

Suppose you specify -P "vn-policy=unpack" with not only a number of virtual nodes but also a number of nodes (-L vnode=M,node=*N*, where *M* > *N*). Then, *M*/*N* virtual nodes will be placed on each node.

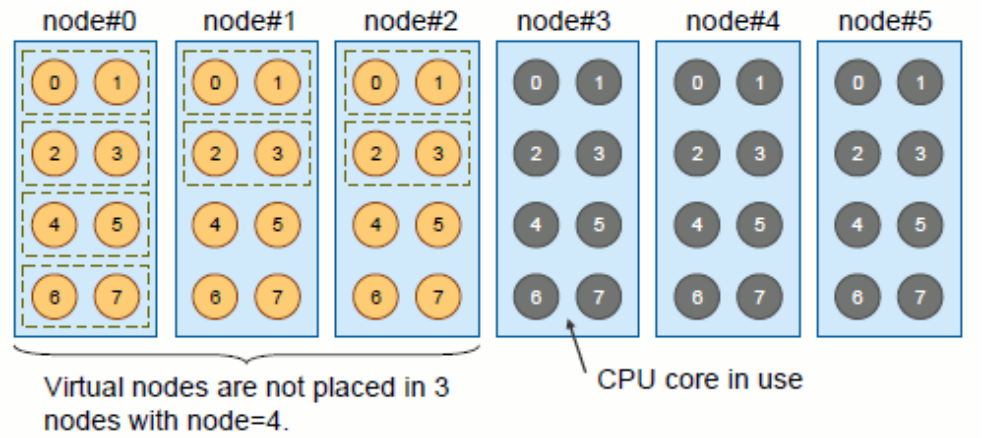
```
$ pjsub -L "vnode=8,vnode-core=2,node=4" -P "vn-policy=unpack" job.sh
or
$ pjsub -L "vnode=8(core=2),node=4" -P "vn-policy=unpack" job.sh
```

Figure 2.12 UNPACK and specification of the number of nodes (1)




Also, if the number of nodes is specified, the specified number of nodes is always necessary. Therefore, if there are insufficient nodes as shown in following figure, virtual nodes cannot be placed even when -P "vn-policy=unpack" is specified.

Figure 2.13 UNPACK and specification of the number of nodes (2)




In the above example, UNPACK placement does not use three nodes only (nodes 0, 1, 2) since the specified number of nodes is four.

### 2.3.4.2 Rank map

For details on rank maps, see "2.3.5 Submitting an MPI job"

### 2.3.4.3 Node selection method

There are two methods of selecting the nodes for placing virtual nodes: distributed method and concentrated method.

The administrator sets which method is used, and users cannot specify it.

The following shows the difference in allocation between the distributed method and concentrated method. In this case, five virtual nodes having two CPU cores are submitted in a way (UNPACK) that they are placed on only different nodes if at all possible.

```
$ pjsub -L "vnode=5,vnode-core=2" -P "vn-policy=unpack" job.sh
or
$ pjsub -L "vnode=5(core=2)" -P "vn-policy=unpack" job.sh
```

Figure 2.14 Node selection with the distributed method

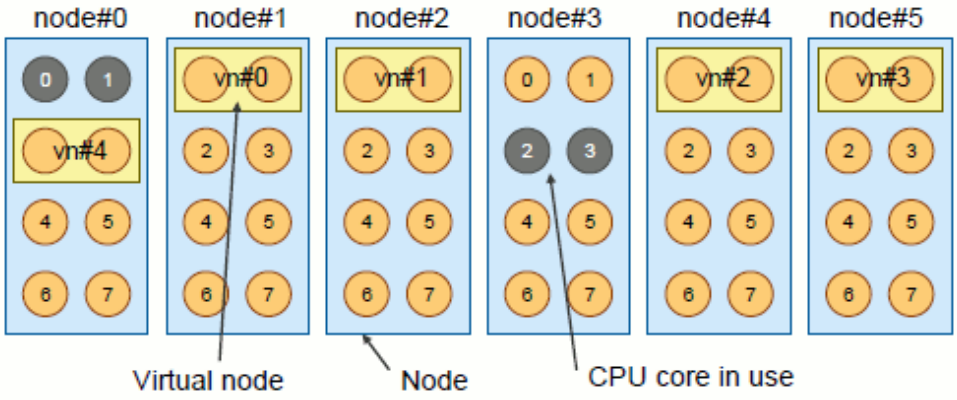
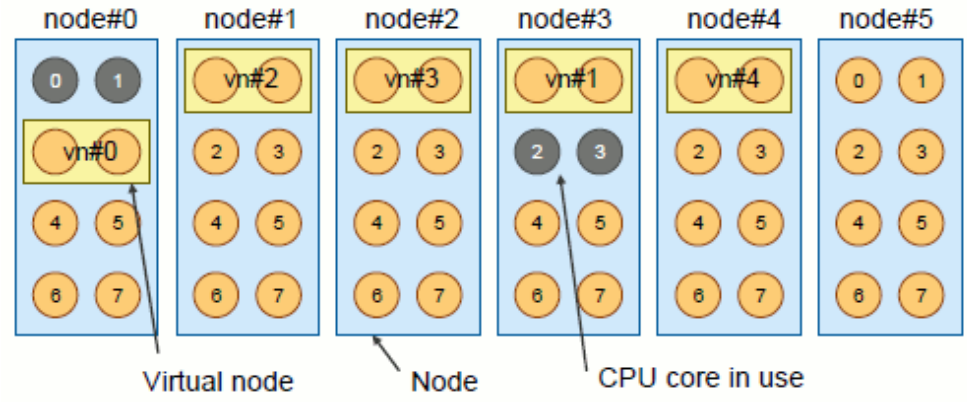


Figure 2.15 Node selection with the concentrated method




## 2.3.4.4 Priority control of allocated nodes

Priority control of allocated nodes is a method of selecting nodes according to the priority set for the nodes.

The administrator sets whether to use this method and the priority of the nodes. Therefore, users do not have to be aware of either when submitting jobs.

## 2.3.4.5 Execution mode policy

An execution mode policy specifies whether to allow submitted jobs to share nodes with other jobs.

For this policy, SIMPLEX (occupy) and SHARE (share) can be specified in the -P "exec-policy=" option of the pjsub command.

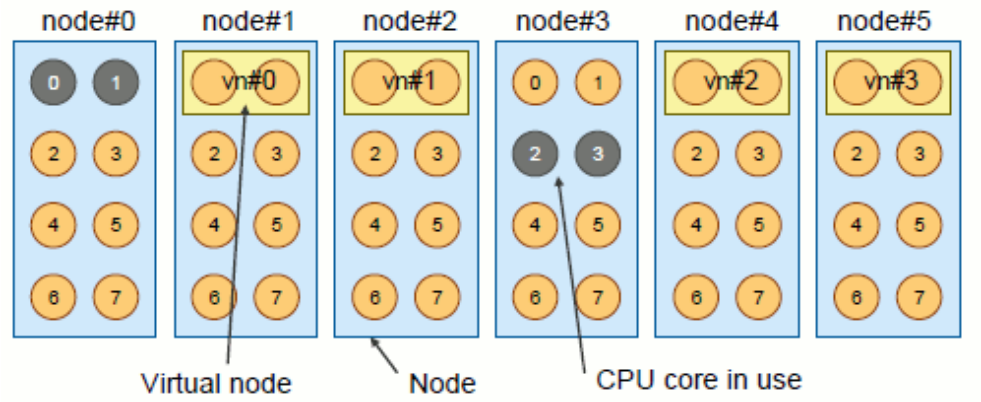
The following examples specify SIMPLEX and SHARE respectively.

SIMPLEX

In this example, four virtual nodes having two CPU cores are always placed on different nodes, and the job occupies these nodes.

```
$ pjsub -L "vnode=4,vnode-core=2" -P "vn-policy=abs-unpack,exec-policy=simplex" job.sh
or
$ pjsub -L "vnode=4(core=2)" -P "vn-policy=abs-unpack,exec-policy=simplex" job.sh
```

Figure 2.16 Node occupation with SIMPLEX specification



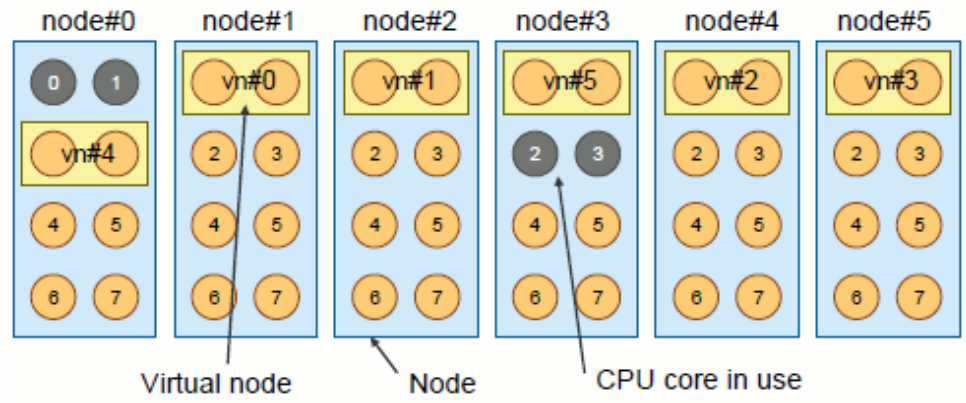

In the above example, only four nodes (node#1, node#2, node#4, and node#5) can be occupied. Therefore, five or more virtual nodes cannot be allocated with SIMPLEX.

SHARE

In this example, six virtual nodes having two CPU cores are always placed on different nodes, and the nodes are allowed to be shared with other jobs.

```
$ pjsub -L "vnode=6,vnode-core=2" -P "vn-policy=abs-unpack,exec-policy=share" job.sh
or
$ pjsub -L "vnode=6(core=2)" -P "vn-policy=abs-unpack,exec-policy=share" job.sh
```

Figure 2.17 Node sharing with SHARE specification




The fifth and sixth virtual nodes (vn4, vn5) in the above example share the nodes with other jobs.

## 2.3.5 Submitting an MPI job

When submitting an MPI job, you can specify the following parameters in the --mpi option of the pjsub command:

- Shape of the process [FX100/FX10]
- Number of processes to create
- Rules on assigning ranks for created processes
- Use of the rank number directory [FX10]

**Note**

To execute an MPI program on a single node (node = 1) or a single virtual node (vnode=1), be sure to specify the --mpi option.

The following example describes specification of the --mpi option.

### 2.3.5.1 Specifying the shape of the process [FX100/FX10]

You can specify the shape of the initial processes (node shape) by using the shape parameter of the --mpi option of the pjsub command for MPI job as a node-exclusive job on the FX100 or FX10 compute nodes.

The types of process shape are one-dimensional, two-dimensional, and three-dimensional. The specified number of dimensions must be the same as the node shape specified in the node parameter of the -L option (or --rsc-list). If the shape parameter is omitted from the --mpi option, the specification in the node parameter of the -L option is used.

```
[one-dimensional shape]   --mpi "shape=X"
[two-dimensional shape]   --mpi "shape=XxY"
[three-dimensional shape] --mpi "shape=XxYxZ"
```

The process created at MPI program initiation and the process dynamically created are executed respectively in the node shape specified by the shape parameter, and they do not share the node shape.
Therefore, node shape (-L node=) allocated to the MPI job should be an enough size where all the MPI processes including the process dynamically created can be executed at the same time in the job.
For instance, dynamic process creation fails because it is allocated only by one node when '-L node=1' is specified at submitting the MPI job.
When the process dynamically created ends, the node that it used can be used to create the process dynamically newly.

The following example shows specification of a three-dimensional process shape (four nodes on the X-axis, three on the Y-axis, and two on the Z-axis) for the MPI program prog_A.

```
$ cat job.sh                                                                    (1)
#!/bin/sh
export PATH=/opt/FJSVfxlang/<version>/bin:$PATH                                 (2)
export LD_LIBRARY_PATH=/opt/FJSVfxlang/<version>/lib64:$LD_LIBRARY_PATH         (3)
mpiexec ./prog_A
$ pjsub -L "node=4x3x2" --mpi "shape=4x3x2" job.sh                              (4)
```

(1) job.sh job script that executes the MPI program prog_A
(2) Environment setting for executing the MPI job
(3) Environment setting for executing the MPI job
(4) Submits a job with a specified node shape of 4 x 3 x 2 and process shape 4 x 3 x 2.

### 2.3.5.2 Specifying the number of processes to create

The number of processes created at starting the MPI program or can be created dynamically are specified by the proc parameter of --mpi option of the pjsub command.

```
--mpi "proc=procnum"
```

The value of proc parameter is as follows.

- For a node-exclusive job to be executed on the FX100 or FX10 compute nodes

  The maximum value that can be specified for the proc parameter is "*<shape parameter>* * *N*". Here, *N* is the number of CPU cores per computer node, which is 32 for the FX100 compute node, and 16 for the FX10 compute node.
  Specifying the maximum value makes the job a flat parallel job, which executes processes on all the cores of each node so that the processes are fixed to different cores without duplication.
  In other words, this maximum value is as follows: the number of processes generable in a node is "*<proc parameter>* / *<The number of nodes specified with "shape" parameter>*" (It rounds up below the decimal point). This value is also applied to the processes which are created dynamically.
  When the proc parameter is omitted, the number of nodes specified with the shape parameter is used. When the shape parameter is also omitted, the number of nodes specified with the node parameter of -L option is used

- For a node-sharing job to be executed on the FX100 or FX10 compute nodes

  The maximum value that can be specified for the proc parameter is equal to the number of CPU cores per virtual node (the value assigned to the core parameter of the -L option).
  When the proc parameter is omitted, 1 is assumed specified.

- For a job to be executed on the PRIMERGY compute nodes

  The maximum value that can be specified for the proc parameter is the number of virtual nodes (vnode parameter of -L option).
  In other words, the number of processes generable in a virtual node is 1.
  When the proc parameter is omitted, the number of virtual nodes is used.

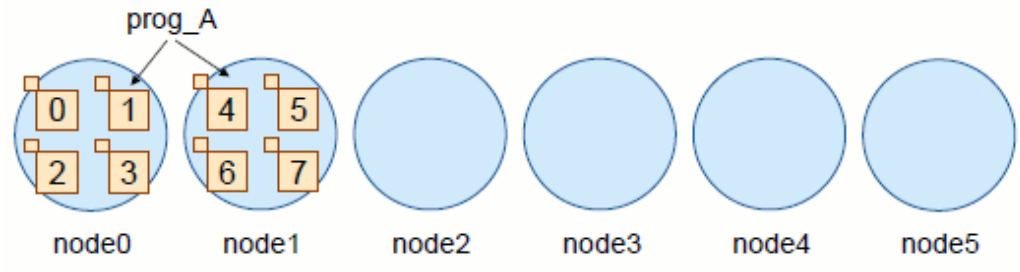
When the value specified by the proc parameter is larger than the maximum value, the pjsub command refuses the acceptance of the job.

The following is an example of MPI program that creates processes dynamically on FX10 compute nodes, and it explains the relation among node, shape and proc parameter.

```
$ cat job.sh
#!/bin/sh
export PATH=/opt/FJSVfxlang/<version>/bin:$PATH
export LD_LIBRARY_PATH=/opt/FJSVfxlang/<version>/lib64:$LD_LIBRARY_PATH
mpiexec ./prog_A
$ pjsub -L "node=6" --mpi "shape=2,proc=8" job.sh
```

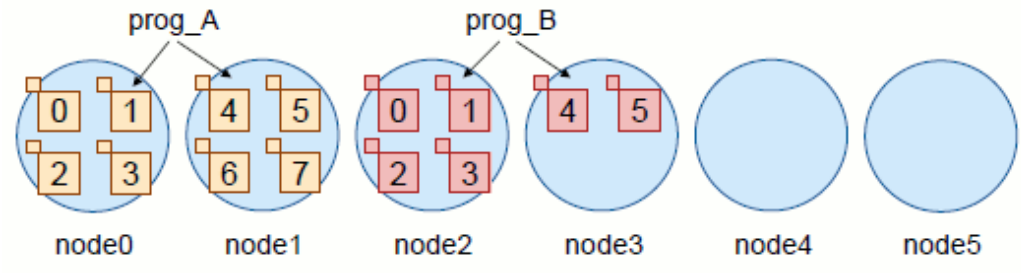
1. In this example, 6 nodes are allocated to a node-exclusive job (-L "node=6")

2. When the MPI program prog_A is executed, 8 processes are created on two nodes, node0 and node1 ("shape=2,proc=8"). Four processes are created in a node (proc/shape=4).



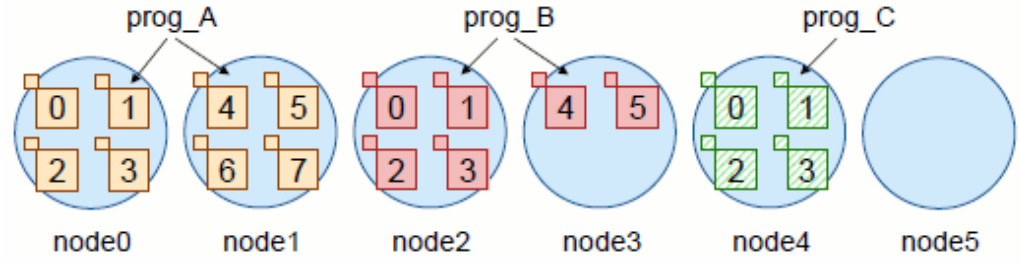

3. When the MPI program prog_A creates 6 processes of the MPI program prog_B using function MPI_Comm_spawn, the processes are created on node2 and node3 ("shape=2"). The maximum number of processes per a node is 4 as well as prog_A.

```
MPI_Comm_spawn("prog_B", NULL, 6, ...);
```



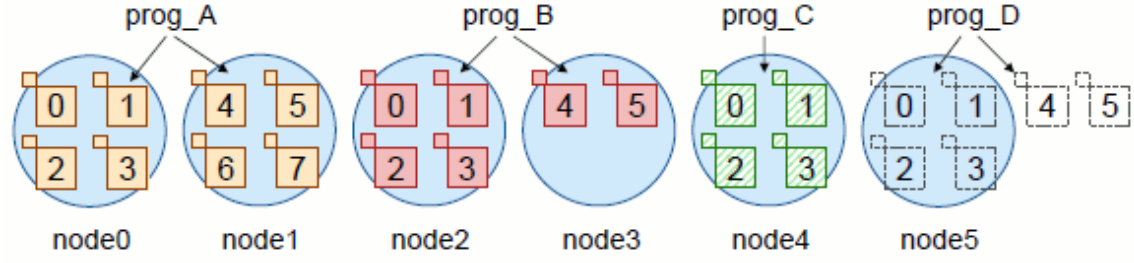

4. When the MPI program prog_A creates 4 processes of program prog_C, the processes are created on node4 as well as prog_A. At this time, the processes are created on node 4 so that prog_C should not share nodes with prog_B though there is free CPUs in node 3.

```
MPI_Comm_spawn("prog_C", NULL, 4, ...);
```




5. The generation of process fails, when the MPI program prog_A try to create 6 processes of program prog_D further. Because free node is only node5 though the program prog_D needs 2 nodes.

```
MPI_Comm_spawn("prog_D", NULL, 6, ...);
```




The upper limit for the number of MPI processes generated per node can be specified for jobs executed with FX100 or FX10 compute clusters.

Table 2.22 Upper limit for number of MPI processes generated per node [FX100/FX10]

| Option | Meaning |
|---|---|
| --mpi max-proc-per-node=*n* | Upper limit for number of MPI processes generated per node.<br>This is valid only for FX100 and FX10 node-exclusive jobs. |

When this is specified with the --mpi proc=*n* option, the operation is as follows.

Table 2.23 Combinations of options specifying the number of MPI processes generated and their operation [FX100/FX10]

| --mpi proc=*n1* | --mpi max-proc-per-node=*n2* | Operation |
|---|---|---|
| With specification | Without specification | [At start of program]<br>In accordance with the rank allocation rule in the next section, *n1*/*shape* (or *n1*/*node* process) MPI processes are generated per node, up to the total number of *n1* processes.<br>*shape*: Number of nodes specified with the -L shape parameter<br>*node*: Number of nodes specified with the -L node parameter (when not specifying -L shape parameter)<br><br>[At generation of dynamic process]<br>Up to *n1*/*shape* MPI processes in one virtual node can be generated. |
| Without specification | With specification | [At start of program]<br>*n2* MPI processes for each node are generated.<br><br>[At generation of virtual processes]<br>Up to *n2* MPI processes can be generated for each node. |
| With specification | With specification | [At start of program]<br>In accordance with the rank allocation rule in the next section, up to *n2* MPI processes are generated for one node, up to the total number of *n1* processes.<br><br>[At generation of dynamic process]<br>Up to *n2* MPI processes can be generated for each node. |

## Note

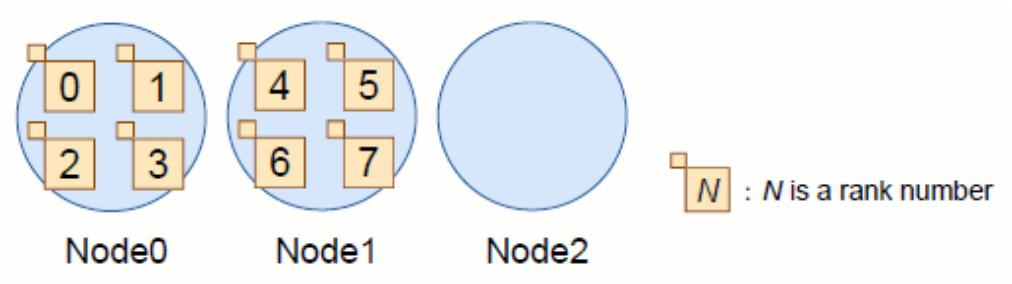
The above operations are the default operations that result when an option is not specified for the mpiexec command that is specified within the job script.

Examples are given below.

[Example 1]

```
[Job Script]
#PJM -L node=3
#PJM --mpi "shape=2,proc=8"
mpiexec a.out
```

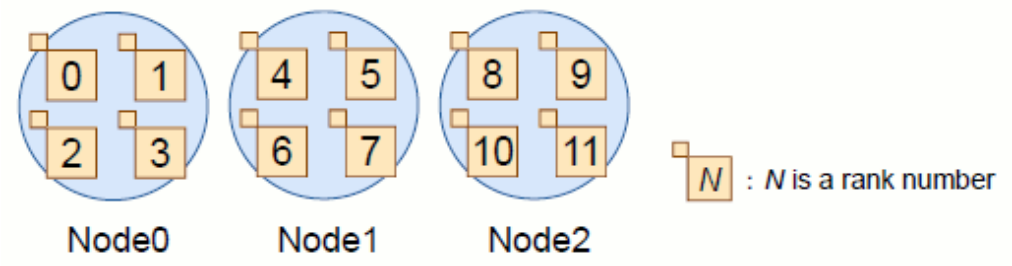
In this case, the MPI process placement is as follows.




[Example 2]

```
[Job Script]
#PJM -L node=3
#PJM --mpi max-proc-per-node=4
mpiexec a.out
```

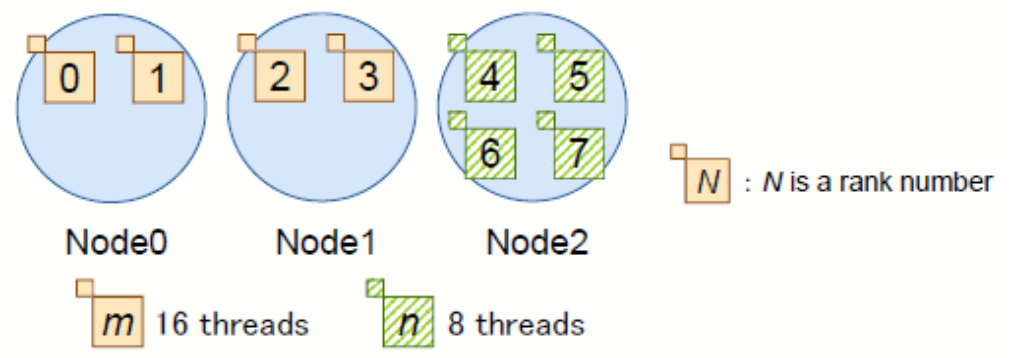
In this case, the MPI process placement is as follows.




[Example 3]

```
[Job Script]
#PJM -L node=3
#PJM --mpi max-proc-per-node=4
mpiexec --vcoordfile file a.out       (*) For detail of --vcoordfile option,
                                          See the manual "MPI User's Guide" of
                                          Technical Computing Language

[Contents of the file]
(0) core=16
(0) core=16
(1) core=16
(1) core=16
(2) core=8
(2) core=8
(2) core=8
(2) core=8
```

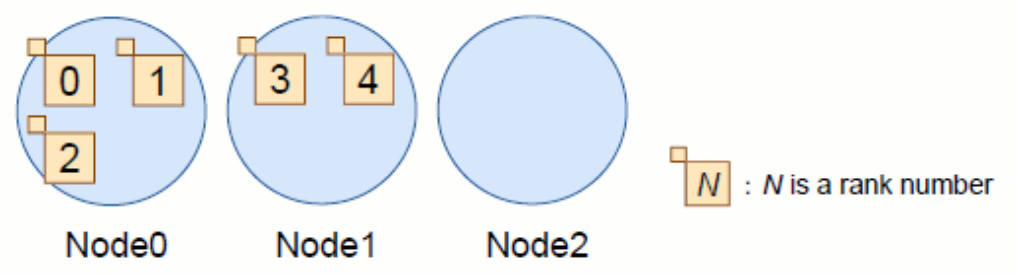
In this case, the MPI process placement is as follows.



[Example 4]

```
[Job Script]
#PJM -L node=3
#PJM --mpi proc=5
#PJM --mpi max-proc-per-node=3
mpiexec a.out
```

In this case, the MPI process placement is as follows.




## 2.3.5.3 The rules on assigning nodes for the ranks

MPI assigns the number that is called rank to each process for the identification of the one like process ID. The job operation management function assigns nodes for each process in order of the rank. You can specify a rule on selecting the nodes to be assigned for ranks in the --mpi option of the pjsub command.

- To assign nodes in sequence from the same node for consecutive ranks, specify the rank-map-bychip parameter. For details, see "2.3.5.4 rank-map-bychip parameter."
  To assign different nodes in sequence for consecutive ranks, specify the rank-map-bynode parameter. For details, see "2.3.5.5 rank-map-bynode parameter."
  These parameters are mutually exclusive.
  If neither is specified, nodes are assigned as follows.
  - For FX100 or FX10 compute nodes (only for node-exclusive jobs)
    It operates assuming that the rank-map-bychip parameter is specified.
  - For PRIMERGY compute nodes
    Virtual node IDs are set in the order of virtual node placement based on the virtual node placement policy. On PRIMERGY compute nodes, virtual node IDs correspond to the rank order (rank numbers).

**Information**

To generate one rank (process) for one node, the node assignment method is the same regardless of the specified parameter.

- For FX100 and FX10 compute nodes, you can specify which nodes are selected and their sequence in the rank-map-hostfile parameter when assigning different nodes for ranks. For details, see "2.3.5.7 Node specification by rank-map-hostfile parameter [FX100/FX10]."
  You can specify the rank-map-hostfile parameter together with the rank-map-bychip or rank-map-bynode parameter.
  If the rank-map-hostfile parameter specification is omitted, the rank-map-bynode or rank-map-bychip parameter is assumed to be *rankmap*. For details, see "2.3.5.6 The order of assigning node specified for rankmap [FX100/FX10]."

For the FX100 or FX10 compute nodes, the efficiency of the communication can be improved by arranging it so that the communication distance is shorten between the processes.

**Note**

For FX100 and FX10 compute nodes, you can specify the rank-map-bychip, rank-map-bynode, and rank-map-hostfile parameters only for node-exclusive jobs.

It explains the assigning rule that can be specified --mpi option at the following.

### 2.3.5.4 rank-map-bychip parameter

This section describes the procedure for allocating nodes with the rank-map-bychip parameter.

#### For FX100 or FX10 compute nodes (only for node-exclusive jobs)

The followings are the formats of rank-map-bychip parameter.

```
--mpi "rank-map-bychip[:rankmap]"
--mpi "rank-map-bychip,rank-map-hostfile=filename"
```

Nodes are allocated as follows:

1. Allocate the specified number of processes for one node.
   For this number of ranks, specify the value of (proc parameter)/(number of nodes in shape parameter (rounded up to a whole number)). If the shape parameter is not specified, the specified value in the node parameter is used.
2. Select the next node.
   You can specify the order of nodes in the *rankmap* specification or rank-map-hostfile parameter. (See "2.3.5.7 Node specification by rank-map-hostfile parameter [FX100/FX10].") If *rankmap* and the rank-map-hostfile parameter are specified at the same time, the rank-map-hostfile parameter has priority. For details on values specified in *rankmap*, see "2.3.5.6 The order of assigning node specified for rankmap [FX100/FX10]."
3. Repeat steps 1 and 2 to assign nodes for all ranks.

#### For PRIMERGY compute nodes

The following is the format of rank-map-bychip parameter.

```
--mpi "rank-map-bychip=n"
```

Nodes or virtual nodes are allocated as follows:

1. Place *n* virtual nodes, which is the number specified in --mpi "rank-map-bychip=*n*", on one node.
   Specify the rank-map-bychip=*n* parameter together with unpack=*m* or abs-unpack=*m*, where *m* must be a multiple of *n*.
   The order of the virtual nodes to allocate is ascending order of node ID and virtual node ID. Also, the rank number of the process to allocate corresponds to the virtual node ID.
2. Select the next node.
3. Repeat steps 1 and 2 until all the virtual nodes are allocated.

### 2.3.5.5 rank-map-bynode parameter

This section describes the procedure for allocating nodes with the rank-map-bynode parameter.

#### For FX100 or FX10 compute nodes (only for node-exclusive jobs)

The followings are the formats of rank-map-bynode parameter.

```
--mpi "rank-map-bynode[=rankmap]"
--mpi "rank-map-bynode,rank-map-hostfile=filename"
```

Nodes are allocated as follows:

1. Once a node is assigned for one rank, assign another node for the next rank.
   The order of the assigned node can be specified by *rankmap* or the rank-map-hostfile parameter (See "2.3.5.7 Node specification by rank-map-hostfile parameter [FX100/FX10]"). It gives priority to the rank-map-hostfile parameter when *rankmap* and the rank-map-hostfile parameter are specified at the same time. See "2.3.5.6 The order of assigning node specified for rankmap [FX100/FX10]" for the specification for *rankmap*.
2. After finishing node assignment for all ranks, return to the first assigned node.
3. Repeat steps 1 and 2 to assign nodes for all ranks.

#### For PRIMERGY compute nodes

The followings are the formats of rank-map-bynode parameter.

```
--mpi "rank-map-bynode"
```

Nodes or virtual nodes are allocated as follows:

1. Once a virtual node in a node is assigned for one rank, assign a virtual node in another node for the next rank.
   The order of the virtual nodes to allocate is ascending order of node ID and virtual node ID. Also, the rank number of the process to allocate corresponds to the virtual node ID.
2. After finishing allocation for the virtual nodes in all the allocated nodes, return to the node allocated first, and allocate the another unused virtual node to ranks.
3. Repeat steps 1 and 2 to assign virtual nodes for all ranks.

### 2.3.5.6 The order of assigning node specified for rankmap [FX100/FX10]

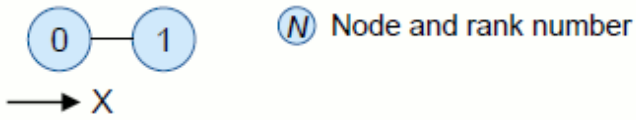
The *rankmap* specified with rank-map-bynode or the rank-map-bychip parameter for the node-exclusive job on FX100 or FX10 compute nodes is the one that which axially whether the node is assigned to the process and specified the node shape by the shape parameter. Specification is expressed by combining character X, Y, and Z that shows the axis. For example, you appoint it with "XY" in the case of the two-dimensional shape and appoint it like "XYZ" in the case of the three-dimensional shape. When the coordinate origin is assumed to be rank 0, the rank is arranged axially of the specification with *rankmap*, and it reaches the edge of shape, it moves to the following axis.

The *rankmap* cannot be specified for one dimension shape. In that case, the node shape is considered to be a torus X axially. The assigning node moves from the node of rank 0 to the adjoining node.

The image of the node assigned when one dimension shape is specified is shown as follows.

```
[one dimension shape] --mpi "rank-map-bynode"
[one dimension shape] --mpi "rank-map-bychip"
```

Figure 2.18 The order of assigning node in one dimension




The assigning node is done for the job that specifies the node shape by one dimension according to one dimension coordinates.

When two is specified for the shape parameter of --mpi option and four is specified for the proc parameter, each process assigning of the rank-map-bynode parameter and the rank-map-bychip parameter becomes it as follows.

Table 2.24 When you specify "--mpi shape=2 proc=4" by assigning one dimension the node

| The order of assigning node | Assigned rank | |
|---|---|---|
| | For rank-map-bynode | For rank-map-bychip |
| 0th | rank 0 and 2 | rank 0 and 1 |
| 1st | rank 1 and 3 | rank 2 and 3 |

Figure 2.19 For rank-map-bynode (one dimension)

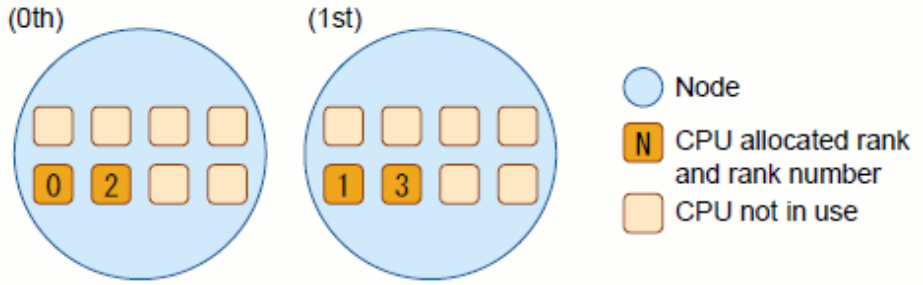
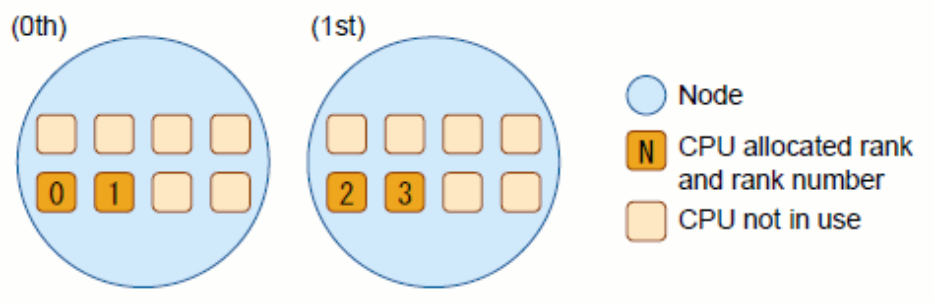


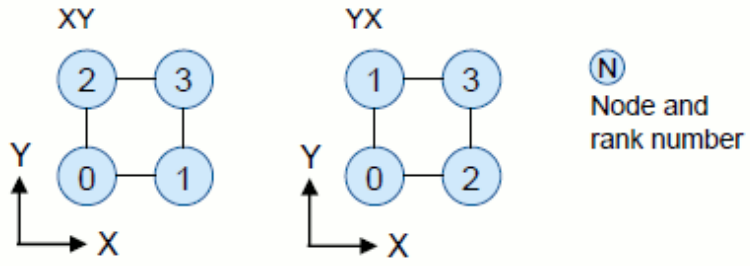
Figure 2.20 For rank-map-bychip (one dimension)




Next, the image of the node assigned when two dimensions shape is specified is shown. It is considered that XY is specified when *rankmap* is not specified.

```
[two dimensions shape] --mpi "rank-map-bynode[={XY|YX}]"
[two dimensions shape] --mpi "rank-map-bychip[:{XY|YX}]"
```

Figure 2.21 The order of assigning node in two dimensions




The assigning node is done for the job that specifies the node shape by two dimensions in order shown in figure above.

When 2x2 is specified for the shape parameter of --mpi option and eight is specified for the proc parameter, each process assigning of the rank-map-bynode parameter and the rank-map-bychip parameter becomes it as follows.

Table 2.25 When you specify "--mpi shape=2x2 proc=8" by assigning two dimensions the node

| The order of assigning node | Assigned rank | |
|---|---|---|
| | For rank-map-bynode | For rank-map-bychip |
| 0th | rank 0 and 4 | rank 0 and 1 |
| 1st | rank 1 and 5 | rank 2 and 3 |
| 2nd | rank 2 and 6 | rank 4 and 5 |
| 3rd | rank 3 and 7 | rank 6 and 7 |

Figure 2.22 For rank-map-bynode (two dimensions)

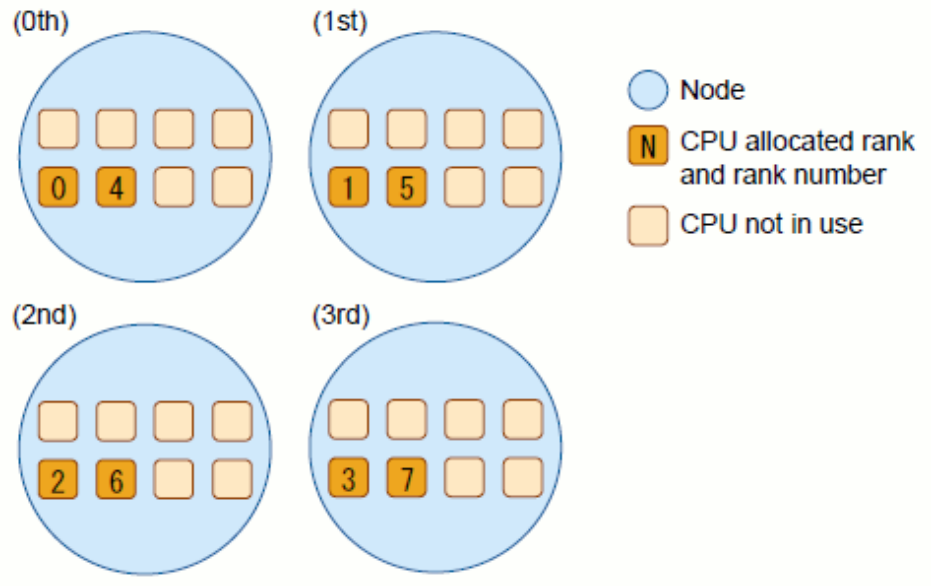
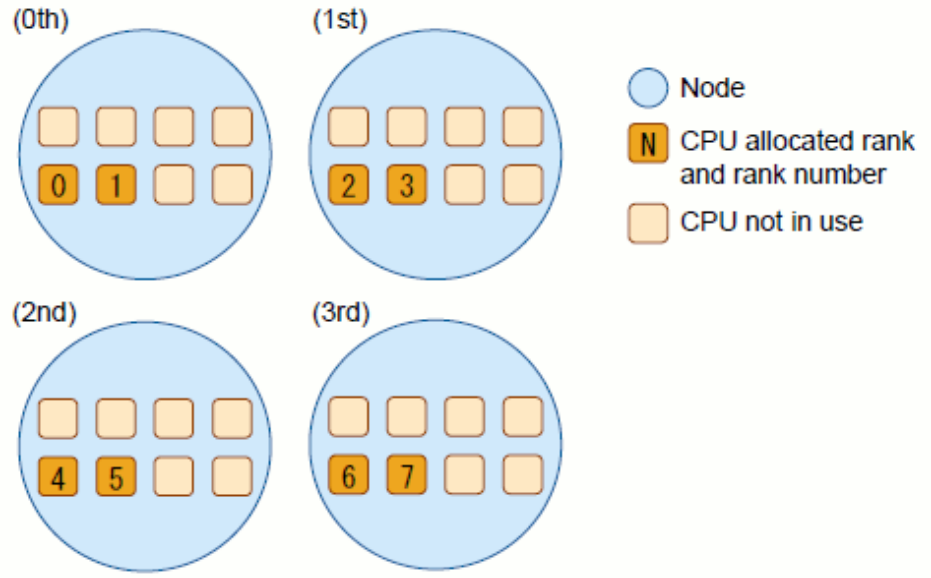


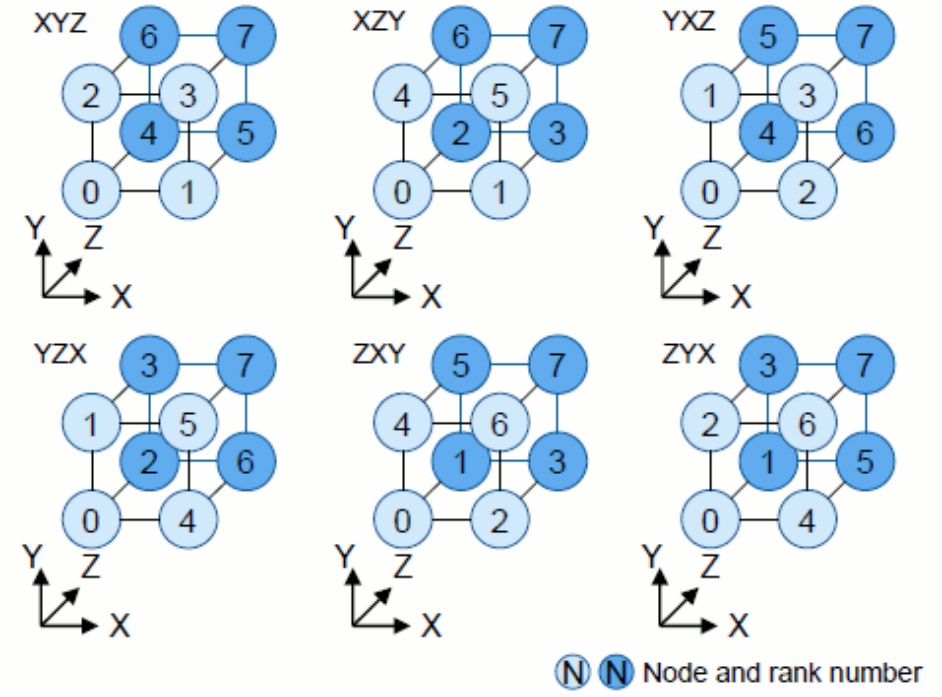

Figure 2.23 For rank-map-bychip (two dimensions)




Next, the image of the node assigned when three dimensions shape is specified is shown. It is considered that XYZ is specified when *rankmap* is not specified.

```
[three dimensions shape] --mpi "rank-map-bynode[={XYZ|XZY|YXZ|YZX|ZXY|ZYX}]"
[three dimensions shape] --mpi "rank-map-bychip[:{XYZ|XZY|YXZ|YZX|ZXY|ZYX}]"
```

Figure 2.24 The order of assigning node in three dimensions




The assigning node is done for the job that specifies the node shape by three dimensions in order shown in figure above.

When 2x2x2 is specified for the shape parameter of --mpi option and 16 is specified for the proc parameter, each process assigning of the rank-map-bynode parameter and the rank-map-bychip parameter becomes it as follows.

Table 2.26 When you specify "--mpi shape=2x2x2 proc=16" by assigning three dimensions the node

| The order of assigning node | Assigned rank | |
|---|---|---|
| | For rank-map-bynode | For rank-map-bychip |
| 0th | rank 0 and 8 | rank 0 and 1 |
| 1st | rank 1 and 9 | rank 2 and 3 |
| 2nd | rank 2 and 10 | rank 4 and 5 |
| 3rd | rank 3 and 11 | rank 6 and 7 |
| 4th | rank 4 and 12 | rank 8 and 9 |
| 5th | rank 5 and 13 | rank 10 and 11 |
| 6th | rank 6 and 14 | rank 12 and 13 |
| 7th | rank 7 and 15 | rank 14 and 15 |

Figure 2.25 For rank-map-bynode (three dimensions)

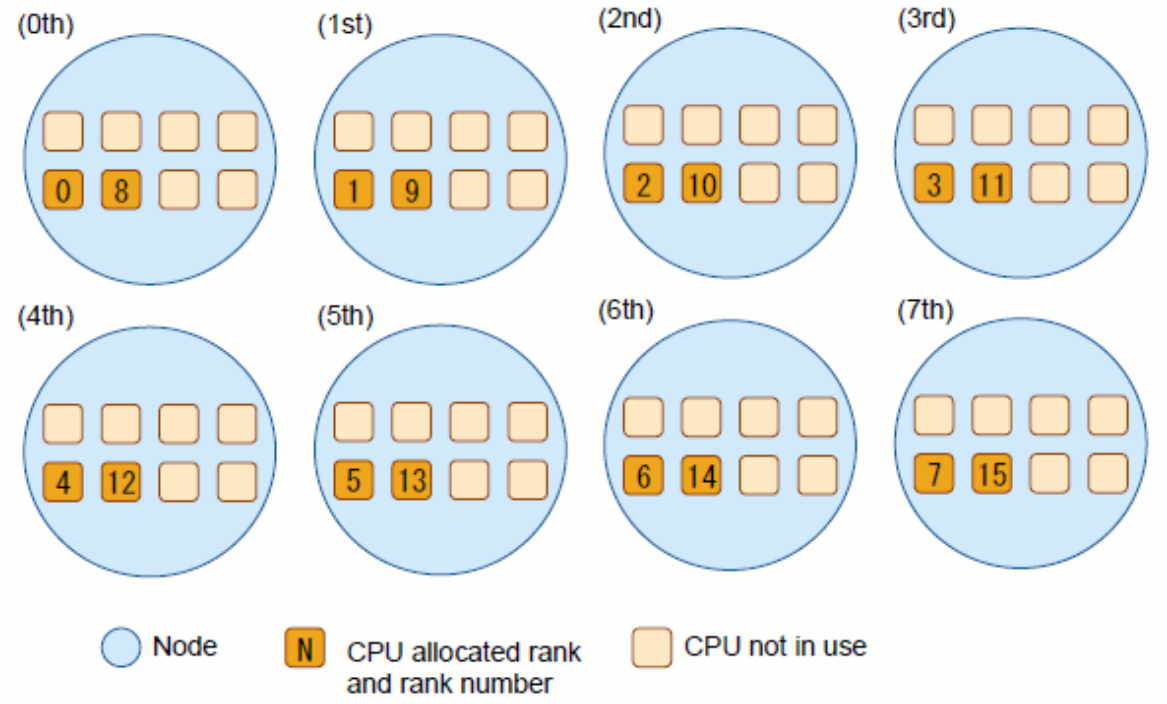
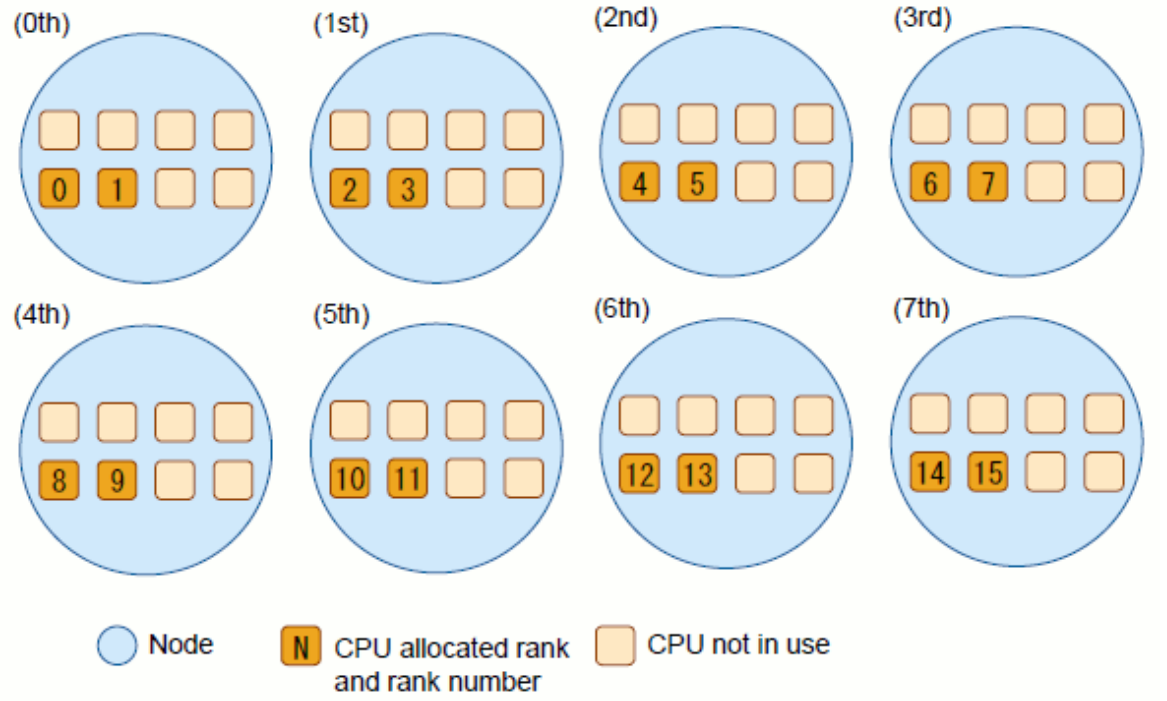



Figure 2.26 For rank-map-bychip (three dimensions)




## 2.3.5.7 Node specification by rank-map-hostfile parameter [FX100/FX10]

The method of assigning node by the rank-map-hostfile parameter for the node-exclusive job is an assigning the nodes for the ranks according to the specified a file *filename*.

The followings are the formats of rank-map-hostfile parameter.

```
--mpi "rank-map-hostfile=filename"
--mpi "rank-map-bychip,rank-map-hostfile=filename"
--mpi "rank-map-bynode,rank-map-hostfile=filename"
```

MPI assigns a node to the rank according to the specified a file *filename*.

The node is specified together with one-dimensional, two-dimensional, or three-dimensional coordinates, depending on the node shape.

Write one set of coordinates in parentheses per line in a file *filename*.

Table 2.27 How to write a rank-map-hostfile parameters

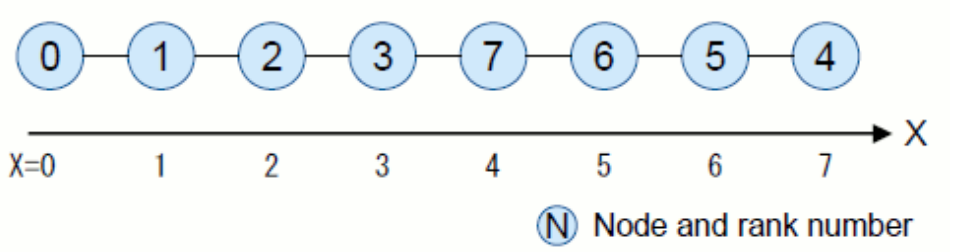
| To specify a coordinate | Content describe |
|---|---|
| one-dimensional coordinate | (X) |
| two-dimensional coordinate | (X, Y) |
| three-dimensional coordinate | (X, Y, Z) |

The reading permission to the user who submits the job is necessary for a file *filename*.

The following example shows specification of one-dimensional rank assignment using the rank-map-hostfile parameter.

```
$ cat rankmapfile-1
(0)
(1)
(2)
(3)
(7)
(6)
(5)
(4)
$ pjsub -L "node=8" --mpi "rank-map-hostfile=rankmapfile-1" job.sh
```

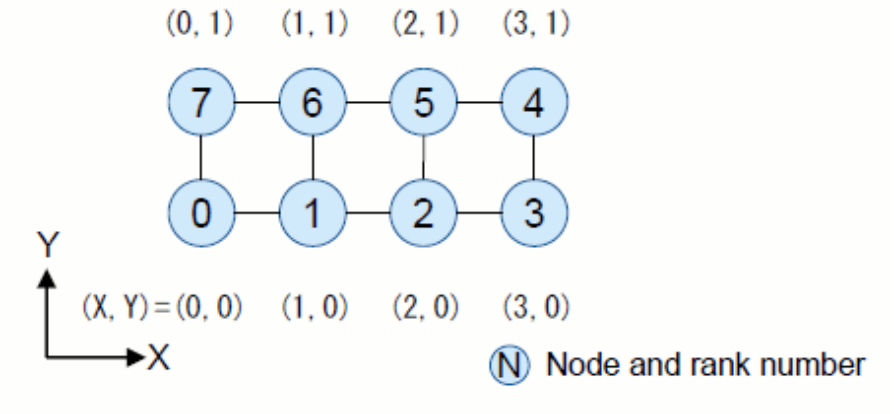
Figure 2.27 Assigning node figure of one dimension using rank-map-hostfile parameter




The following example shows specification of two-dimensional rank assignment using the rank-map-hostfile parameter.

```
$ cat rankmapfile-2
(0,0)
(1,0)
(2,0)
(3,0)
(3,1)
(2,1)
(1,1)
(0,1)
$ pjsub -L "node=4x2" --mpi "rank-map-hostfile=rankmapfile-2" job.sh
```

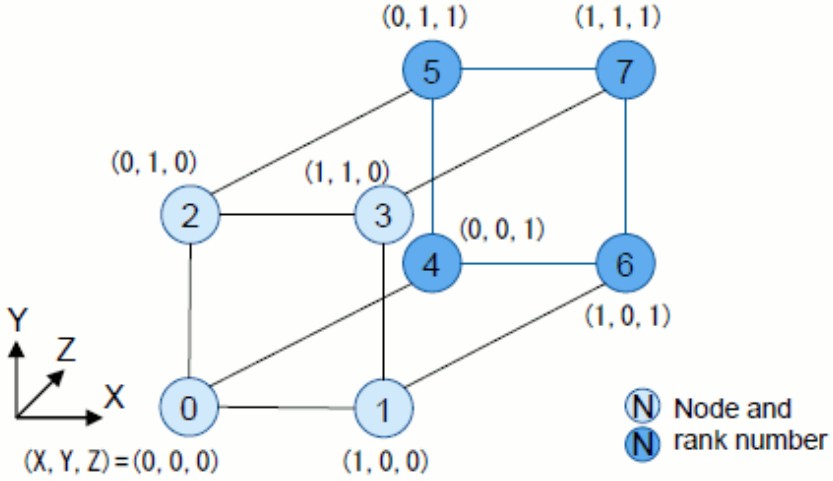
Figure 2.28 Assigning node figure of two dimensions using rank-map-hostfile parameter




The following example shows specification of three-dimensional rank assignment using the rank-map-hostfile parameter.

```
$ cat rankmapfile-3
(0,0,0)
(1,0,0)
(0,1,0)
(1,1,0)
(0,0,1)
(0,1,1)
(1,0,1)
(1,1,1)
$ pjsub -L "node=2x2x2" --mpi "rank-map-hostfile=rankmapfile-3" job.sh
```

Figure 2.29 Assigning node figure of three dimensions using rank-map-hostfile parameter




## Note

- The blank line under the file *filename* is disregarded.
- The coordinates written in the file *filename* must be values within the range specified by the shape parameter representing the node shape. As an example, for shape=2x3, the coordinates (0,0), (0,1), (0,2), (1,0), (1,1), and (1,2) can be written.
- To specify the rank-map-hostfile parameter together with the rank-map-bychip parameter, the coordinates written in the file *filename* must conform to the following:
  - The number of coordinates in the file *filename* must be equal to the number of nodes in the specified shape represented by the shape parameter.
    As an example, for shape=3x2, the number of nodes is 6, so write six coordinates in the file *filename*.
    If the number of written coordinates is less than the number of nodes in the specified shape represented by the shape parameter,

the pjsub command rejects the acceptance of the job. If the number of written coordinates is greater than the number of nodes in the specified shape represented by the shape parameter, the extra coordinates are disregarded.

  - Multiple identical coordinates cannot be written in the file *filename*.

- To specify the rank-map-hostfile parameter together with the rank-map-bynode parameter, the coordinates written in the file *filename* must conform to the following:

  - Suppose that the number of coordinates written in the file *filename* is less than the number of processes specified by the proc parameter. In this case, node assignment proceeds up to the node at the last coordinates and then returns to the node at the first coordinates.
    If the number of coordinates is greater than the specified number of processes in the proc parameter, the extra coordinates are disregarded.

  - The file *filename* may contain identical written coordinates if the number or coordinates is less than or equal to the number of CPU cores per compute node.

### 2.3.5.8 Using a rank number directory [FX10]

If each parallel process of an MPI program uses the same directory, there may be input/output conflicts, possibly affecting performance. Distributed input/output loads can be expected from the use of a different directory for each parallel process (rank) in the node-exclusive job. This directory is called a "rank number directory."

To use rank number directories in an MPI program, specify the --mpi use-rankdir option.

Table 2.28 How to specify use of a rank number directory

| Specification format | Description |
|---|---|
| --mpi use-rankdir | Executes each rank of the MPI program in the respective rank number directory.<br>If nothing is specified, no rank number directory is created, and each rank uses the same directory. |

#### Information

In the job execution using staging function, the load-balancing of the file input-output can be expected by option --mpi use-rankdir. However, it might be ineffectual by the program executed in the job.

If rank number directories are used, staging is required for each rank number directory.

Therefore, specification of the staging target files (e.g., --stgin) can be done as shown below with consideration of rank numbers.

```
[Stage in]
--stgin "rank=range src %r:dst"
--stgin-dir "rank=range srcdir %r:dstdir"

[Stage out]
--stgout "rank=range %r:src dst"
--stgout-dir "rank=range %r:srcdir dstdir"
```

#### Note

When the rank number directories are used, note the followings.

- The path of MPI program specified to the mpiexec command is assumed as a relative path to the rank number directory regardless of the current directory.
- The files required by process which is created dynamically must be stage-in into the top directory of the job (shared with ranks).

The following table shows the notation.

Table 2.29 Staging target file notation that considers rank numbers

| Purpose | Notation |
|---|---|
| Single rank number | %r |
| | %0<n>r<br>Column <n> expresses the rank number. The string is left-filled with zeros. (Example: %03r) |
| Rank number range specification | rank=$N_1$-$N_2$<br>$N_1$ omitted: 0 (Example: rank=-9 means *rank number* 0 to *rank number* 9)<br>$N_2$ omitted: Number of MPI processes - 1 (Example: rank=0- means *rank number* 0 to *number of MPI processes* - 1) |
| | rank=* (all rank numbers)<br>Note: If you specify this on the command line, escape the asterisk so that the shell does not interpret it. |
| Path name expansion | *: Matches any string containing a space character.<br>? : Matches any single character.<br>[*abc*]: Matches any character in parentheses.<br>Note: These are the same as the bash shell specifications. |
| | %r:*path*<br>Use the string to specify the stage-in destination or stage-out source. *path* points to a location in the rank number directory. The rank number %r is in the target range (rank=*range*) of the specified rank number. |

## Note

The notation %r cannot be used for --stgout-basedir option.

The following examples use the above notation to specify staging with regard to rank number directories.

- For all rank numbers, the ./a.out, ./in.dat, and ./param.txt files are staged in to "./" in the rank number directory.

```
#PJM --mpi use-rankdir
#PJM --stgin "rank=* ./a.out %r:./"
#PJM --stgin "rank=* ./in.dat %r:./"
#PJM --stgin "rank=* ./param.txt %r:./"
```

  You can write the above specifications on one line as shown below.

```
#PJM --mpi use-rankdir --stgin "rank=* ./a.out ./in.dat ./param.txt  %r:./"
```

  Even if the rank number specification "rank=*" is omitted, the meaning is the same.

```
#PJM --mpi use-rankdir --stgin "./a.out ./in.dat ./param.txt  %r:./"
```

- Files with names that include a rank number are staged in to their respective rank number directories.

```
#PJM --mpi use-rankdir
#PJM --stgin "rank=* ./programA  %r:./"
#PJM --stgin "rank=* ./in.%03r   %r:./"
#PJM --stgin "rank=* ./sc.%03r   %r:./"
```

  In the above example, suppose the rank numbers are 0 to 15. Then, the files in.000 to in.015 and the files sc.000 to sc.015 are staged in to their respective rank number directories.

### Note

If a specified file does not exist, stage-in fails, and the job is rejected. Consequently, the job enters the REJECT state.
If a specified file does not exist at the stage-out time, processing continues.

- Files are staged in to specific rank number directories.

```
#PJM --mpi use-rankdir
#PJM --stgin "rank=* ./bmt.exe ./param.in %r:./"
#PJM --stgin "rank=* ./data.%02r %r:./"
#PJM --stgin "rank=0 ./seq0100.pdb.0[0-7] %r:./"
#PJM --stgin "rank=0 ./seq0200.pdb.0[0-9] %r:./"
#PJM --stgin "rank=0 ./seq0200.pdb.1[0-5] %r:./"
```

The above example stages in the bmt.exe and param.in files to all the rank number directories. It stages in each data.?? ("??" is a two-digit rank number) file to the rank number directory corresponding to the rank number.

Also, the files seq0100.pdb.00 to seq0100.pdb.07 and the files seq0200.pdb.00 to seq0200.pdb.15 are staged in to the rank number directory of rank number 0.

- Files are staged out from specific rank number directories.

```
#PJM --mpi use-rankdir
#PJM --stgout "rank=0 ./out.000 ./"
#PJM --stgout "rank=0 ./out.txt ./"
```

The above example stages out the files out.000 and out.txt in the rank number directory of rank number 0 to the current directory at the job submission time.

- A file name is changed to stage out the file.

```
#PJM --mpi use-rankdir
#PJM --stgout "rank=0 ./output ./%n.output"
#PJM --stgout "rank=* ./out.%03r results_111/"
```

For rank number 0, the above example changes the file name of the output file to *job name*.output and stages out the file. Also, for all rank numbers, each out.??? (??? is a three-digit rank number) file is staged out to the results_111 directory.

The following example shows more complicated specifications.

```
#PJM --mpi use-rankdir
#PJM --stgout "rank=0     ./output ./"
#PJM --stgout "rank=0-31  ./output_1.%02r results_1/"
#PJM --stgout "rank=32-63 ./output_2.%02r results_2/%n.output_2.%02r"
```

In this case, for rank number 0, the output file is staged out. For rank numbers 0 to 31, each output_1.?? (?? is a two-digit rank number) file is staged out to the results_1 directory. For rank numbers 32 to 63, each output_2.?? file is staged out as the results_2/*job name*.output_2.?? file.

## 2.3.6 Examples of specifying MPI job execution [FX100/FX10]

The following examples are MPI job (node-exclusive job) for FX100 or FX10 that attention is necessary to the node shape.

### See

For details on how to use the mpiexec command to execute an MPI program, see the "MPI User's Guide," which is the Technical Computing Language manual.

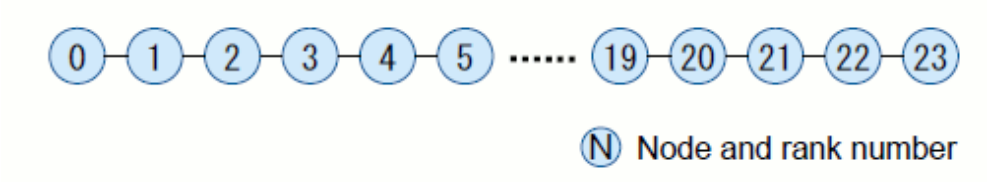
### 2.3.6.1 Executing a job in a one-dimensional node shape

The following example shows execution of the MPI program prog_A by the mpiexec command. 24 processes are continuously mapped in one-dimensional nodes.

```
$ cat job.sh
#!/bin/sh
#PJM -L "node=24"                                                    (1)
#PJM --mpi "shape=24"                                                (2)
export PATH=/opt/FJSVfxlang/<version>/bin:$PATH                      (3)
export LD_LIBRARY_PATH=/opt/FJSVfxlang/<version>/lib64:$LD_LIBRARY_PATH  (4)
mpiexec -n 24 ./prog_A                                               (5)
$ pjsub job.sh
```

(1) Node shape: one-dimensional with 24 nodes
(2) Process shape: one-dimensional with 24 nodes
(3) Environment setting for executing the MPI job
(4) Environment setting for executing the MPI job
(5) Executes prog_A with 24 parallel processes.

Figure 2.30 Executing a job in a one-dimensional shape




With the following specifications, 24 processes can also be mapped in one-dimensional nodes.

```
$ cat job.sh
#!/bin/sh
#PJM -L "node=24"                                                    (1)
export PATH=/opt/FJSVfxlang/<version>/bin:$PATH                      (2)
export LD_LIBRARY_PATH=/opt/FJSVfxlang/<version>/lib64:$LD_LIBRARY_PATH  (3)
mpiexec -n 24 ./prog_A                                               (4)
$ pjsub job.sh
```

(1) Node shape: one-dimensional with 24 nodes
(2) Environment setting for executing the MPI job
(3) Environment setting for executing the MPI job
(4) Executes prog_A with 24 parallel processes. The process shape is the same as the node shape with -L node=24.

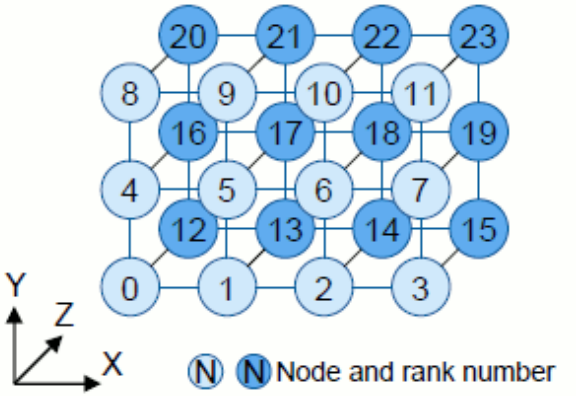
### 2.3.6.2 Executing a job in a three-dimensional node shape

The following example shows execution of the MPI program prog_A by the mpiexec command. 24 processes are continuously mapped in three-dimensional nodes.

```
$ cat job.sh
#!/bin/sh
#PJM -L "node=4x3x2"                                                 (1)
export PATH=/opt/FJSVfxlang/<version>/bin:$PATH                      (2)
export LD_LIBRARY_PATH=/opt/FJSVfxlang/<version>/lib64:$LD_LIBRARY_PATH  (3)
mpiexec ./prog_A
$ pjsub job.sh
```

(1) Node shape: three-dimensional with 24 nodes
(2) Environment setting for executing the MPI job
(3) Environment setting for executing the MPI job

Figure 2.31 Executing a job in a three-dimensional shape




## 2.3.6.3 Executing a program several times in one job

When using a single job to execute an MPI program several times, be careful with the required number of nodes.

For the node shape specified in the pjsub command (the specified value of --mpi shape parameter, or the specified value of -L node if shape parameter is omitted), specify the maximum number of processes.

Using the mpiexec command option (-n, --n, -np or --np) for specifying the number of processes, specify an arbitrary value that is less than the maximum number of processes.

```
$ cat job.sh
#!/bin/sh
#PJM -L "node=4x3x2"                                                    (1)
export PATH=/opt/FJSVfxlang/<version>/bin:$PATH                         (2)
export LD_LIBRARY_PATH=/opt/FJSVfxlang/<version>/lib64:$LD_LIBRARY_PATH (3)
mpiexec -n 12 ./prog_A                                                  (4)
mpiexec ./prog_B                                                        (5)
mpiexec -n 16 ./prog_C                                                  (6)
$ pjsub job.sh
```

(1) Three-dimensional with 24 nodes
(2) Environment setting for executing the MPI job
(3) Environment setting for executing the MPI job
(4) 12 parallel processes
(5) Number of parallel processes: Maximum number of created processes (24)
(6) 16 parallel processes

First, determine the rank assignment, and use it until the specified process.

Figure 2.32 Determining the rank assignment

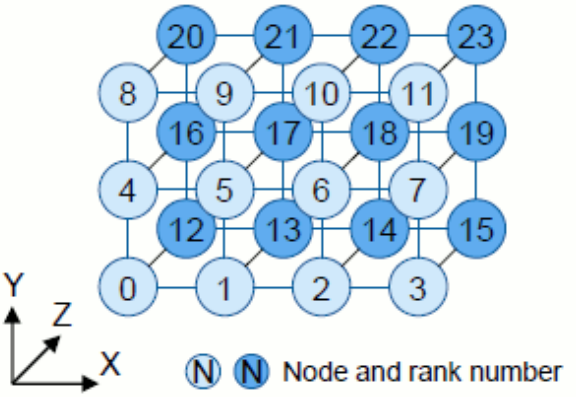
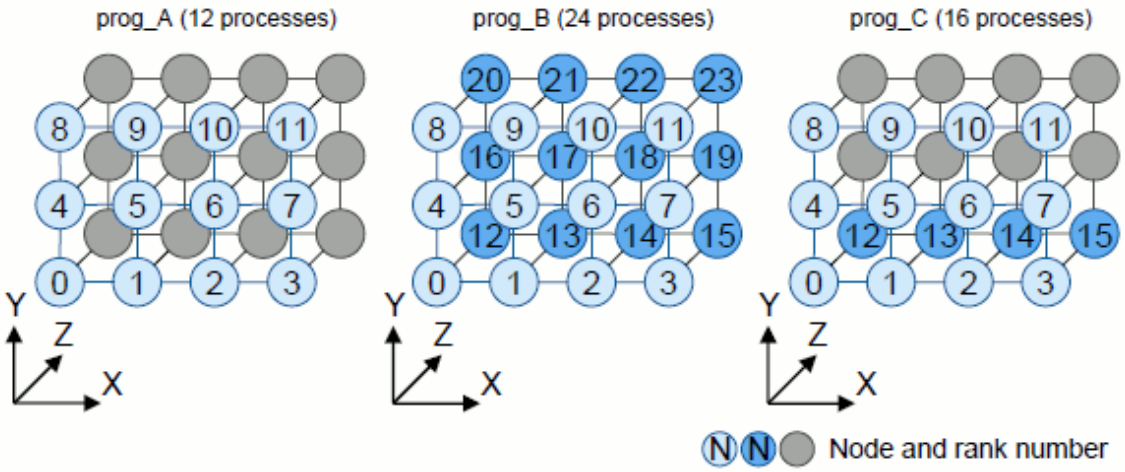


Figure 2.33 Process execution example




### 2.3.6.4 Example of executing multi-processes in one node job that specifies rank-map-bynode parameter and rank-map-hostfile parameter

The example with the rank-map-bynode and the rank-map-hostfile parameters, and the job creates 2 processes on one node is executed is shown below.

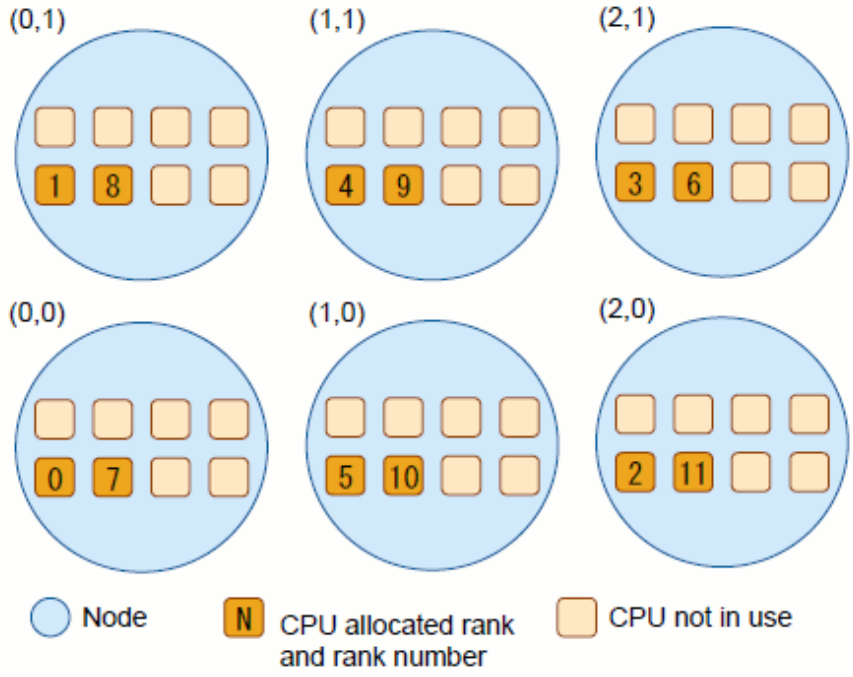
In the following examples, the node shown by coordinates of the first line of *hostfile* is assumed to be rank 0, and the node is assigned every one line.

```
$ cat hostfile
(0, 0)
(0, 1)
(2, 0)
(2, 1)
(1, 1)
(1, 0)
(2, 1)
(0, 0)
(0, 1)
(1, 1)
(1, 0)
(2, 0)
$ cat job.sh
#!/bin/sh
#PJM -L "node=3x2"                                                    (1)
#PJM --mpi "rank-map-hostfile=hostfile"
#PJM --mpi "proc=12"
#PJM --mpi "rank-map-bynode"
export PATH=/opt/FJSVfxlang/<version>/bin:$PATH                       (2)
export LD_LIBRARY_PATH=/opt/FJSVfxlang/<version>/lib64:$LD_LIBRARY_PATH   (3)
mpiexec ./prog_A
```

(1) Two-dimensional with 6 nodes.
(2) Environment setting for executing the MPI job
(3) Environment setting for executing the MPI job

The number of processes created to each one node becomes 12/ (3x2) =2.

Figure 2.34 Assigning node order of using a rank-map-bynode parameter and rank-map-hostfile parameter




### 2.3.6.5 Example of executing multi-processes in one node job that specifies rank-map-bychip parameter and rank-map-hostfile parameter

The example with the rank-map-bychip and the rank-map-hostfile parameters, and the job creates 2 processes on one node is executed is shown below.

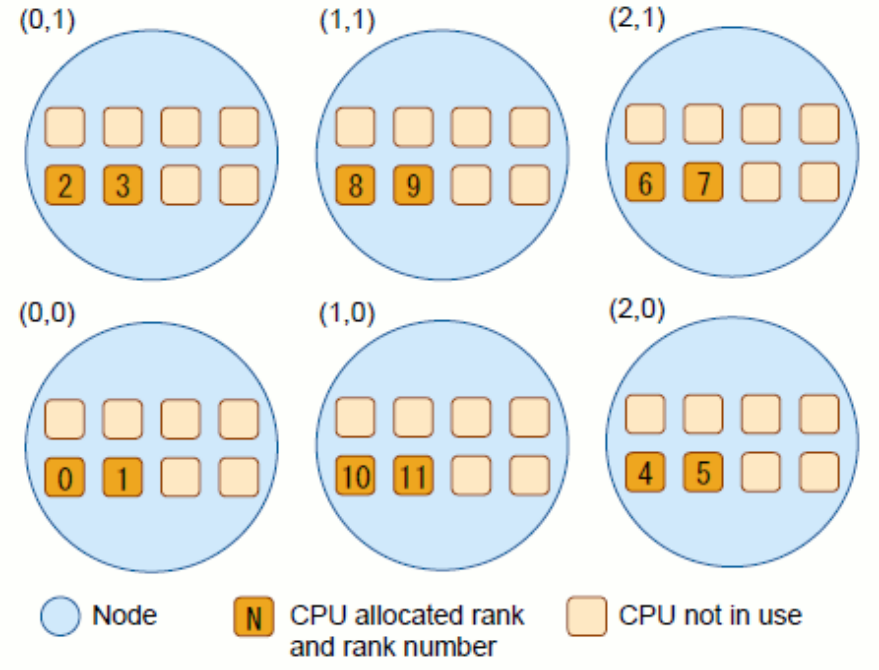
In the following examples, the node shown by coordinates of the first line of the *hostfile* file is assumed to be rank 0, and the number specified by the rank-map-bychip parameter is assigned. The remaining line in the *hostfile* file is disregarded.

```
$ cat hostfile
(0, 0)
(0, 1)
(2, 0)
(2, 1)
(1, 1)
(1, 0)
(2, 1)
(0, 0)
(0, 1)
(1, 1)
(1, 0)
(2, 0)
$ cat job.sh
#!/bin/sh
#PJM -L "node=3x2"                                                    (1)
#PJM --mpi "rank-map-hostfile=hostfile"
#PJM --mpi "proc=12"
#PJM --mpi "rank-map-bychip=2"
export PATH=/opt/FJSVfxlang/<version>/bin:$PATH                       (2)
export LD_LIBRARY_PATH=/opt/FJSVfxlang/<version>/lib64:$LD_LIBRARY_PATH (3)
mpiexec ./prog_A
$ pjsub job.sh
```

(1) Two-dimensional with 6 nodes.
(2) Environment setting for executing the MPI job
(3) Environment setting for executing the MPI job

The number of processes created to one node becomes 12/ (3x2) =2.

Figure 2.35 Assigning node order of using a rank-map-bychip parameter and rank-map-hostfile parameter




## 2.3.6.6 How to execute an MPI program in the MPMD model

The following example shows specification of execution of an MPI program consisting of several different programs. It is an MPI program in the MPMD (Multiple Program Multiple Data) model.

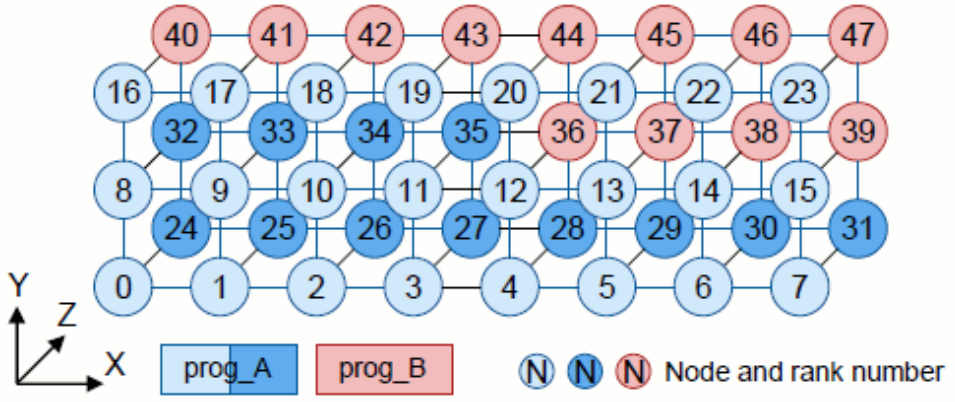
For the number of processes specified in the pjsub command (the specified value of --mpi shape, or the specified value of -L node if shape is omitted), specify the total parallelization value (number of processes) of each program. Using a colon as a delimiter in a combination of the number of processes to be generated and the MPI program name, assign processes from the beginning of the *filename* file.

Using the option (-n) for specifying the number of mpiexec processes, specify the parallelization of each program (number of processes). If the -n option is not specified, the value specified in the -L node option is passed. Consequently, the specified number would exceed the total number of nodes, and the job would end in an error.

```
$ cat job.sh
#!/bin/sh
#PJM -L "node=8x3x2"                                                    (1)
export PATH=/opt/FJSVfxlang/<version>/bin:$PATH                         (2)
export LD_LIBRARY_PATH=/opt/FJSVfxlang/<version>/lib64:$LD_LIBRARY_PATH (3)
mpiexec -n 36 ./prog_A : -n 12 ./prog_B                                 (4)
$ pjsub job.sh
```

(1) Three-dimensional with 48 nodes. Total number of nodes required for prog_A and prog_B
(2) Environment setting for executing the MPI job
(3) Environment setting for executing the MPI job
(4) The option for specifying parallel processes cannot be omitted for MPMD.

Figure 2.36 Specifying sequential execution of a job




## 2.3.6.7 Specifying a rank for an MPI program in the MPMD model

You can specify rank assignment for an MPI program in the MPMD model by using the --mpi rank-map-hostfile parameter of the pjsub command.

Using a colon as a delimiter, specify a combination of the number of processes to be generated and the MPI program name in the mpiexec command.

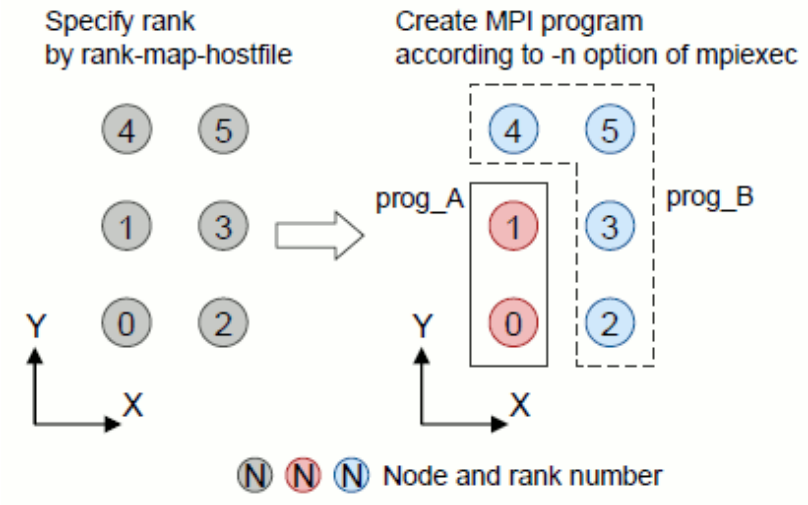
The job will assign a process from the beginning of the *filename* file.

The following is a specification example.

```
$ cat filename                                                              (1)
(0,0)
(0,1)
(1,0)
(1,1)
(0,2)
(1,2)
$ cat job.sh
#!/bin/sh
#PJM -L "node=2x3"
#PJM --mpi "rank-map-hostfile=filename"                                     (2)
export PATH=/opt/FJSVfxlang/<version>/bin:$PATH                             (3)
export LD_LIBRARY_PATH=/opt/FJSVfxlang/<version>/lib64:$LD_LIBRARY_PATH     (4)
mpiexec -n 2 ./prog_A : -n 4 ./prog_B                                       (5)
$ pjsub job.sh
```

(1) File specifying the rank map
(2) Specifies the rank map.
(3) Environment setting for executing the MPI job
(4) Environment setting for executing the MPI job
(5) Executes the MPI programs prog_A and prog_B in the MPMD model.

Figure 2.37 Assigning an MPMD program shape




## 2.3.6.8 How to execute a hybrid parallel program

The following example shows execution of a hybrid parallel program that uses both process parallelism and thread parallelism.

If the -n option of the mpiexec command is omitted, the value specified in the -L option of the pjsub command is automatically passed.

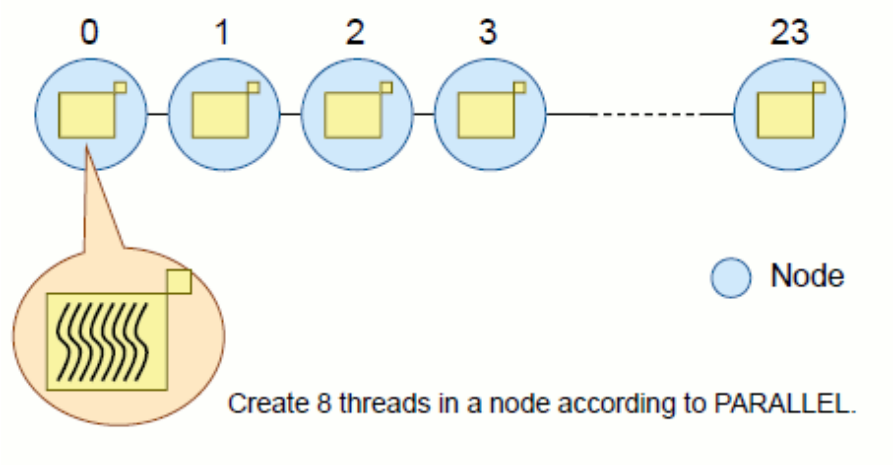
```
$ cat job.sh
#!/bin/sh
#PJM -L "node=24"

PARALLEL=8                                                                    (1)
export PARALLEL
export PATH=/opt/FJSVfxlang/<version>/bin:$PATH                               (2)
export LD_LIBRARY_PATH=/opt/FJSVfxlang/<version>/lib64:$LD_LIBRARY_PATH       (3)
mpiexec ./prog_A                                                              (4)
$ pjsub job.sh
```

(1) Specifies the number of threads.
(2) Environment setting for executing the MPI job
(3) Environment setting for executing the MPI job
(4) 24 parallel processes

### Note

In the above example, the environment variable PARALLEL specifies the number of threads in automatic parallelization. For details, see the Technical Computing Language manual.

Figure 2.38 Hybrid parallel program generated




# 2.4 Checking the Job Status

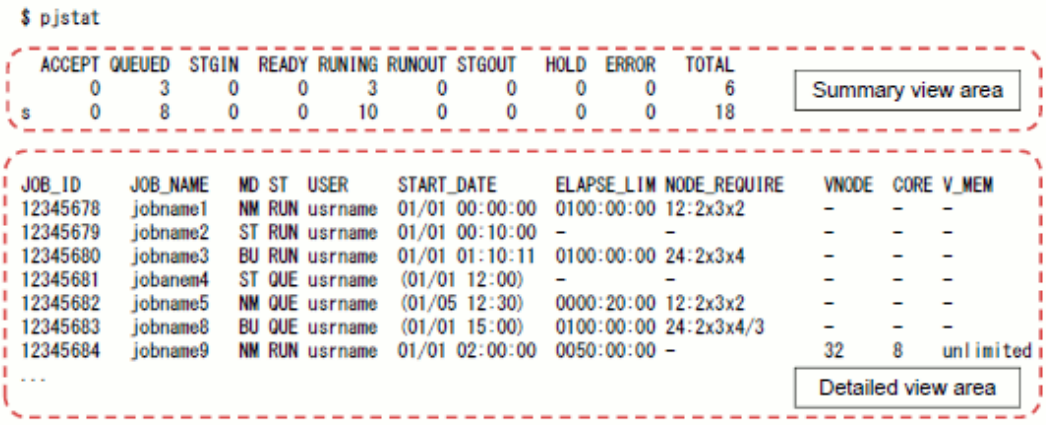
This section describes how to check the execution status of a job after submitting the job.

## 2.4.1 Displaying a job list

Users can use the pjstat command to check the states of their own submitted jobs and other related information.

The display area of the pjstat command is split into a summary view area and a detailed view area.

Figure 2.39 pjstat command display format




### Information

For the meanings of display items, see "A.1 Output of the pjstat and pjstat -v."

Summary view area

The summary view area shows counts of jobs in each state.

The displayed counts of jobs are divided into two rows. The top row shows the counts of jobs excluding sub jobs. The bottom row (with "s" at the beginning) shows the counts of all jobs, including sub jobs, but they do not include the jobs to which the sub jobs belong.

Detailed view area

The detailed view area shows detailed information on each job.

## Note

- By default, the pjstat command displays only the jobs accessible to the user who executed the command. If the display targets in the -A or other options of the pjstat command include a job that the user is not allowed to reference, some information is not displayed. That being said, the administrator can reference all jobs.
- The states of step and bulk jobs are the integrated states of multiple sub jobs. As a result, their job states are not necessarily going to match the state of each of their sub jobs.
  In particular, bulk jobs generate a variety of states. For example, even a bulk job in the RUNOUT state may have some sub jobs still in the QUEUED and RUNNING states.
  To find out the true states of step and bulk jobs, use the -E option of the pjstat command to display sub job states as well. For details, see "e. Displaying sub jobs."
- The pjstat command display can be displayed with the old Technical Computing Suite V1.0L20 format, by using the --compat option. This can be used to maintain display format compatibility. For details, see "A.3 Display for compatibility with previous versions."

The following examples show the display of job information with the pjstat command for FX10 compute node.

a. Displaying information on only a specific job

With a job ID or sub job ID specified, the pjstat command displays information on only the specified job. Multiple job IDs can be specified. In this case, the upper summary view area also shows the counts of the specified job. The lower summary view area is for all jobs of the user who executes the pjstat command. When the administrator executes the pjstat command, the lower summary view area shows the job of all users.

```
$ pjstat 1006900

  ACCEPT QUEUED  STGIN  READY RUNING RUNOUT STGOUT   HOLD  ERROR   TOTAL
       0      0      0      0      1      0      0      0      0       1
s      0      0      0      0      1      0      0      0      0       1

JOB_ID    JOB_NAME  MD ST  USER    START_DATE      ELAPSE_LIM NODE_REQUIRE   VNODE  CORE V_MEM
1006900   test.sh   NM RUN pjm     10/07 16:04:19  0024:00:00 12:2x3x2       -      -    -
```

You can specify multiple job IDs and sub job IDs by listing them or specifying a range.

- Listing

```
$ pjstat 1006900 1006902                                              (1)
$ pjstat '12345680[2]' '12345680[3]' '12345680[4]' '12345680[5]'      (2)
$ pjstat 12345679_2 12345679_3 12345679_4 12345679_5                  (3)
```

(1) Multiple job IDs
(2) Multiple sub job IDs (bulk job)
(3) Multiple sub job IDs (step job)

- Specifying a range

The job IDs, bulk numbers, or step numbers in a range specification have a hyphen between them.

```
$ pjstat 100690-100692         (1)
$ pjstat '12345680[2-5]'       (2)
$ pjstat 12345679_2-5          (3)
```

(1) job ID 100690, 100691, 100692
(2) sub job ID 12345680[2], 12345680[3], 12345680[4], 12345680[5]
(3) sub job ID 12345679_2, 12345679_3, 12345679_4, 12345679_5

b. Displaying more detailed job information

You can display more detailed information on each job in the detailed view area by specifying the -v option of the pjstat command.

```
$ pjstat -v
```

```
  ACCEPT QUEUED  STGIN  READY RUNING RUNOUT STGOUT   HOLD  ERROR   TOTAL
       0      2      0      0      1      0      0      0      0       3
s      0      2      0      0      1      0      0      0      0       3

JOB_ID     JOB_NAME   MD ST  USER     START_DATE      ELAPSE_TIM ELAPSE_LIM NODE_REQUIRE
(*)VNODE  CORE V_MEM     V_POL E_POL RANK  LST EC  PC  SN PRI ACCEPT         RSC_UNIT RSC_GRP
(*)REASON
1006900    test.sh    NM RUN pjm      10/07 16:04:19  0000:00:34 0024:00:00 12:2x3x2
(*)-      -    -         -     -     -     RNA -   -   -  127 10/07 16:04:58 rsc_unit rsc_grp
(*)-
1006901    test.sh    NM QUE pjm      (10/08 16:30)   -          0024:00:00 12:2x3x2
(*)-      -    -         -     -     -     ACC -   -   -  127 10/07 16:04:59 rsc_unit rsc_grp
(*)-
1006902    test.sh    NM QUE pjm      -               -          0024:00:00 373248:72x72x72
(*)-      -    -         -     -     -     ACC -   -   -  127 10/07 16:05:29 rsc_unit rsc_grp
(*)INSUFF NODE
```

(*) This line continues from the previous line and is actually displayed on the previous line.

c. Customizing the displayed items in the detailed view area.

You can customize the displayed items in the detailed view area by using the --choose option of the pjstat command. The command displays information according to the order of the item names listed in the arguments of the --choose option. For details of the item names that can be specified, see the man page of the pjstat command.

The following example shows the display of the job ID, job name, job model, and job execution start time.

```
$ pjstat --choose jid,jnam,jmdl,sdt

  ACCEPT QUEUED  STGIN  READY RUNING RUNOUT STGOUT   HOLD  ERROR   TOTAL
       0      1      0      0      1      0      0      0      0       2
s      0      5      0      0      5      0      0      0      0      10

JOB_ID     JOB_NAME   MD START_DATE
12345678   jobname1   NM 01/01 00:00:00
12345679   jobname5   BU (01/05 12:30)
```

d. Displaying only jobs in a specific status.

Using the --filter option, you can display only jobs that satisfy the specified conditions. You can specify the following conditions in the --filter option. For details of the item names that can be specified, see the man page of the pjstat command.

| Condition | Format | Description |
|---|---|---|
| Value specification | --filter "*item*=*value*" | Any item that matches the specified value becomes a display target.<br>Example: --filter "jmdl=NM"<br>Display the items where the job model (jmdl) is the normal job (NM).<br>Example: --filter "st=RNA+RUN"<br>Display the jobs in the RUNNING-A or RUNNING state.<br>Note: The "+" symbol indicates the OR condition. To use the "+" symbol as an ordinary character, add a backslash ("\+"). |
| Range specification | --filter "*item*=*range*" | Any item included in the specified range is a display target.<br>Example: --filter "jid=1-10"<br>Display the jobs whose job IDs are each in a range of 1 to 10.<br>Example: --filter "jid=10-"<br>Display the job whose job IDs are each 10 or higher.<br>Example: --filter "jid=-20"<br>Display the jobs whose job IDs are each 20 or lower.<br>Example: --filter "jid=1-10+21-30"<br>Display the jobs whose job IDs are each in a range of 1 to 10 or 21 to 30. |

| Condition | Format | Description |
|---|---|---|
| Regular expression specification | --filter "*item=regularexpression*" | Any item consisting of a character string that matches the specified regular expression is a display target.<br>You can use the following two types of regular expressions.<br>. (period): Any single character except a line break<br>* (asterisk): Any character string with 0 or more characters except a line break<br>Example: --filter "jnam=jobA*"<br>Display the jobs whose names begin with "jobA".<br>Example: --filter "jnam=job."<br>Display the jobs whose names each consist of "job" followed by one character.<br>Example: --filter "jnam=jobA*+jobB*"<br>Display the jobs whose names begin with "jobA" or "jobB". |

### Note

- If both the --filter option and the --choose option are specified, only the items specified in the --choose option are the targets of the --filter option. Any item specified in the --filter option but not specified in the --choose option is ignored.
- If the -A option or --all option is specified, other users' jobs becomes display targets, but hidden information such as user names is ignored even if specified in the --filter option.

e. Displaying sub jobs

By default, the detailed view area does not show the sub jobs of any bulk job or step job. To display sub jobs, specify the -E or --expand option.

```
$ pjstat

  ACCEPT QUEUED  STGIN  READY RUNING RUNOUT STGOUT   HOLD  ERROR   TOTAL
       0      3      0      0      3      0      0      0      0       6
s      0      7      0      0     11      0      0      0      0      18

JOB_ID     JOB_NAME   MD ST  USER     START_DATE      ELAPSE_LIM NODE_REQUIRE  VNODE  CORE V_MEM
12345678   jobname1   NM RUN usrname  01/01 00:00:00  0100:00:00 12:2x3x2      -      -    -
12345679   jobname2   ST RUN usrname  01/01 00:10:00  -          -             -      -    -
12345680   jobname3   BU RUN usrname  01/01 01:10:11  0100:00:00 24:2x3x4      -      -    -
...

$ pjstat -E

  ACCEPT QUEUED  STGIN  READY RUNING RUNOUT STGOUT   HOLD  ERROR   TOTAL
       0      8      0      0     12      0      0      0      0      20
s      0      7      0      0     11      0      0      0      0      18

JOB_ID     JOB_NAME   MD ST  USER     START_DATE      ELAPSE_LIM NODE_REQUIRE  VNODE  CORE V_MEM

12345678   jobname1   NM RUN  usrname  01/01 00:00:00  0100:00:00 12:2x3x2      -      -    -
12345679   jobname2   ST RUN  usrname  01/04 12:30:00  -          -             -      -    -
12345679_2 jobname2   ST RUN  usrname  01/04 12:30:00  0100:00:00 12:2x3x2      -      -    -
12345679_3 jobname2   ST QUE  usrname  -               0100:00:00 12:2x3x2      -      -    -
12345679_4 jobname2   ST QUE  usrname  -               0100:00:00 12:2x3x2      -      -    -
12345679_5 jobname2   ST QUE  usrname  -               0100:00:00 12:2x3x2      -      -    -
12345680   jobname3   BU RUN  usrname  01/01 01:10:11  0100:00:00 12:2x3x2      -      -    -
12345680[0] jobname3   BU RUN  usrname  01/01 20:15:00  0100:00:00 12:2x3x2      -      -    -
12345680[1] jobname3   BU RUN  usrname  01/01 20:15:00  0100:00:00 12:2x3x2      -      -    -
12345680[2] jobname3   BU RUN  usrname  01/01 20:15:00  0100:00:00 12:2x3x2      -      -    -
12345680[3] jobname3   BU RUN  usrname  01/01 20:15:00  0100:00:00 12:2x3x2      -      -    -
...
```

In the above example, the step job (job ID 12345679) has four sub jobs (sub job IDs 12345679_2, 12345679_3, 12345679_4, and 12345679_5). The example shows that the sub job of sub job ID 12345679_2 is being executed.

Also, the bulk job (job ID 12345680) has four sub jobs (sub job IDs 12345680[0], 12345680[1], 12345680[2], and 12345680[3]). The example shows that all the sub jobs are being executed.

To display the information on only a specific sub job, specify the sub job ID in an argument of the pjstat command. The following example shows the display for sub job 12345680[0] of the bulk job.

```
$ pjstat '12345680[0]'

  ACCEPT QUEUED  STGIN  READY RUNING RUNOUT STGOUT   HOLD  ERROR   TOTAL
       0      0      0      0      1      0      0      0      0       1
s      0      0      0      0      1      0      0      0      0       1

JOB_ID       JOB_NAME  MD ST  USER     START_DATE      ELAPSE_LIM NODE_REQUIRE  VNODE  CORE V_MEM
12345680[0] jobname3  BU RUN usrname  06/01 20:15:00  0100:00:00 12:2x3x2      -      -    -
```

### Note

If you specify a sub job ID of a bulk job in an argument of the pjstat command, be sure to escape the brackets ("[ ]") with single quotation marks, etc. so that the shell does not process the brackets.

The pjstat command has more options than described above. For details, see the man page of the pjstat command.

## 2.4.2 Displaying job statistical information

You can use the -s or -S option of the pjstat command to display job statistical information on a running job. In the same manner as with the pjsub command, the -S option can display more detailed job statistical information than the -s option.

The following example shows the display of job statistical information.

### Note

- If no job ID is specified, the command displays job statistical information on all the jobs of the user. Job statistical information is displayed with many details, so be aware of that fact when you have numerous jobs.
- Information on the prologue process executed when a job starts is added to the job statistical information for the running job. However, whether the elapsed time of job includes the prologue process depends on the system settings. For details, contact the administrator.
- The FX100 compute node has, besides the CPU core allocated to jobs, a CPU core called the assistant core. The assistant core handles OS interrupt processing and daemon processing that deteriorate job execution performance.
  In jobs, MPI asynchronous communication processing is executed on this assistant core. Therefore, the job statistical information of jobs being executed with the FX100 compute node includes, besides the resources allocated to jobs, the assistant core use information related to MPI asynchronous communication processing.

```
$ pjstat -s

  ACCEPT QUEUED  STGIN  READY RUNING RUNOUT STGOUT   HOLD  ERROR   TOTAL
       0      0      0      0      4      0      0      0      0       4
s      0      7      0      0      8      0      0      0      0      15

JOB ID                  : 9417
JOB NAME                : script.sh
JOB TYPE                : BATCH
JOB MODEL               : NM
RETRY NUM               : 0
SUB JOB NUM             : -
USER                    : user
```

```
GROUP                   : group
RESOURCE UNIT           : rsc_unit
RESOURCE GROUP          : rsc_grp
APRIORITY               : 127
PRIORITY                : 127
SHELL                   : /bin/sh
COMMENT                 :
...
```

See

For details on the displayed job statistical information, see "A.2 Job Statistical Information" or the man manual of the pjstat command.

## 2.4.3 Displaying how a node is used by jobs

Use the pjshowrsc command with the -v 1 option to check the jobs running on a node.

```
$ pjshowrsc -c cluster -v 1
[ CLST: cluster ]
[ NODE: 0x01010010 ]
      RSC     TOTAL    FREE   ALLOC
      cpu        32       0      32
      mem      57Gi       0    57Gi
      lfs         -       -       -

 RUNNING_JOBS: 1022                          <- Job ID of job running on node

[ NODE: 0x0101011 ]
      RSC     TOTAL    FREE   ALLOC
      cpu        32      16      16
      mem      57Gi    30Gi    27Gi
      lfs         -       -       -

 RUNNING_JOBS: 2551                          <- Job ID of job running on node

[ NODE: 0x0101012 ]
      RSC     TOTAL    FREE   ALLOC
      cpu        32      16      16
      mem      57Gi    30Gi    27Gi
      lfs         -       -       -

 RUNNING_JOBS: 2551                          <- Job ID of job running on node
```

By using the pjshowrsc command with the -v 3 option, you can check not only the jobs running on a node, but also the jobs that use the node as a communication path.

```
$ pjshowrsc -c cluster -v 3
[ CLST: cluster ]
[ NODE: 0x01010010 ]
      RSC     TOTAL    FREE   ALLOC
      cpu        32       0      32
      mem      57Gi       0    57Gi
      lfs         -       -       -

 RUNNING_JOBS: 1022
 JOBS_USING_ROUTE: 1030,1031,1032,1033,1034   <- Job IDs of jobs that use node as communication path

[ NODE: 0x0101011 ]
      RSC     TOTAL    FREE   ALLOC
      cpu        32      16      16
      mem      57Gi    30Gi    27Gi
```

```
      lfs         -       -       -

 RUNNING_JOBS: 2551
 JOBS_USING_ROUTE: 1030,1031,1032,1033,1034   <- Job IDs of jobs that use node as communication path
[ NODE: 0x0101012 ]
      RSC     TOTAL    FREE   ALLOC
      cpu        32      16      16
      mem      57Gi    30Gi    27Gi
      lfs         -       -       -

 RUNNING_JOBS: 2551
 JOBS_USING_ROUTE: 1030,1031,1032,1033,1034   <- Job IDs of jobs that use node as communication path
```

Information

A job that runs on an FX100 or FX10 compute node may include a job process that performs inter-node communication that goes through a node that is not allocated to the job.
Even if a node in the communication path fails due to a software error, etc., job execution is not affected as long as the interconnect used for communication and the ICC (Interconnect Controller) that controls it are normal. However, if a node becomes incapable of communication due to an ICC failure, etc., all jobs that use the node as a communication path are affected and aborted.

## 2.4.4 Confirming job end

Use the following methods to find out whether a job has ended.

- Checking the job state by using the pjstat command

  Once a job ends (EXIT, REJECT, or CANCEL state), the pjstat command normally does not display it any longer. However, you can still display the job by specifying the -H or --history option.

```
$ pjstat -H

  ACCEPT QUEUED  STGIN  READY RUNING RUNOUT STGOUT   HOLD  ERROR   TOTAL
       0      0      0      0      2      0      0      0      0       2
s      0      0      0      0      2      0      0      0      0       2

  REJECT     EXIT    CANCEL    TOTAL                                          (1)
       1        1         1        3
s      1        1         1        3

JOB_ID     JOB_NAME   MD ST  USER     START_DATE      ELAPSE_LIM NODE_REQUIRE    VNODE  CORE V_MEM
1234567    jobname1   NM EXT usrname  01/01 00:00:00  0100:00:00 12:2x3x2        -      -    - (2)
1234570    jobname1   NM RJT usrname  01/01 00:00:00  0100:00:00 12:2x3x2        -      -    - (3)
1234590    jobname1   NM CCL usrname  01/01 00:00:00  0100:00:00 12:2x3x2        -      -    - (4)
1234600    jobname1   NM RUN usrname  01/01 00:00:00  0100:00:00 12:2x3x2
1234601    jobname1   NM RUN usrname  01/01 00:00:00  0100:00:00 12:2x3x2
```

  (1) Summary with the counts of ended jobs
  (2) Job in the EXIT state
  (3) Job in the REJECT state
  (4) Job in the CANCEL state

Note

The states of ended jobs are stored only for the length of time set by the administrator. Therefore, even the above operation cannot display the states of the jobs whose storage period has expired.

- E-mail notification

  If e-mail notification is specified in the -m option of the pjsub command at the job submission time, an e-mail is sent to the user when the job ends.

  The user will always be notified by e-mail, even if the job ends abnormally. For details, see "2.6.1 Referencing job execution results."

- Waiting for job end with the pjwait command

  Users can use the pjwait command to wait for the end of a specific job.

  The pjwait command does not return until the specified job ends.

  The following example shows the submission of three jobs, with the pjwait command waiting for these jobs to end.

```
$ pjsub job1.sh
[INFO] PJM 0000 pjsub Job 5300 submitted.
$ pjsub job2.sh
[INFO] PJM 0000 pjsub Job 5301 submitted.
$ pjsub job3.sh
[INFO] PJM 0000 pjsub Job 5302 submitted.
$ pjwait 5300 5301 5302                          (1)
5300 0 0 -                                       (2)
5301 0 1 -
5302 0 0 -
```

  (1) Waiting until all the specified jobs end
  (2) Job ID, the job end code, the end status of the job script and the signal number in each ended job.

# 2.5 Job Operations

This section describes the operations for jobs.

See

The operations described below may not be possible depending on the job state. For details, see "Appendix E Operations on Jobs."

## 2.5.1 Deleting a job

The operation to cancel a submitted job is called "job deletion." To delete a job, specify the job ID in the pjdel command. Deleting a running job stops the job.

```
$ pjdel jobid [ jobid...]
```

The following message appears when the job is deleted normally.

```
[INFO] PJM 0100 pjdel Job jobid canceled.
```

If the specified job does not exist, the following message appears.

```
[ERR.] PJM 0112 pjdel Job non-existing job ID does not exist.
```

Note

- If you specify a job although you have no privileges for deleting it, the command operation will be the same as when a non-existing job is specified.
- When a job in the ACCEPT state is deleted, it switches to the REJECT state. When a job in a state other than the ACCEPT state is deleted, it switches to the CANCEL state. However, if staging is in use, the job enters the STGOUT state to stage out files, and then enters the CANCEL state.

- To perform job deletion that involves the aborting of a running prologue script (RUNNING-P state) or a running epilogue script (RUNNING-E state), specify the --enforce option. Note also that if you delete a job with the --enforce option before epilogue script execution, the epilogue script will be left unexecuted.
  In some cases, the administrator configures the settings so that users cannot specify the --enforce option. Check the pjacl command output item, execute pjdel(--enforce).

## 2.5.2 Sending a signal to a job

Users can send signals to running jobs (RUNNING state) by using the pjsig command.

```
$ pjsig -s signal job ID
```

Users can specify a signal with a signal name (e.g., SIGHUP) or a signal number (1 to 64).

The following example sends the signal SIGKILL (signal number 9) to a job.

```
$ pjsig -s 9 1                                                  (1)
[INFO] PJM 0700 pjsig Accepted job 1 is sent signal 9.

$ pjsig -s SIGKILL 2                                            (2)
[INFO] PJM 0700 pjsig Accepted job 2 is sent signal SIGKILL.
```

(1) Specifies a signal by a signal number.
(2) Specifies a signal by a signal name.

## 2.5.3 Holding a job and canceling the hold

When the pjdel command deletes a running job, the job ends. To execute the job again, the user has to submit the job again.

If the pjhold command is used, the job is aborted but stays in the submitted state. This is called "job hold," and this state is called the HOLD state.

To cancel the HOLD state, use the pjrls command. This reschedules the job to execute it again.

If the system stops during job execution, hold the job once, and then cancel the hold after operation resumes. This saves the effort of submitting the job again.

### Note

- The job can be held only at the status of QUEUED, STGIN, READY, RUNNING-A or RUNNING.
  The job that automatically re-execution is disabled by --norestart option or the setting of job operation can be held only at the status of QUEUED, STGIN and READY.
- To perform job holding that involves the aborting of a running prologue script (RUNNING-P state) or a running epilogue script (RUNNING-E state), specify the --enforce option. Note also that if you hold a job with the --enforce option before epilogue script execution, the epilogue script will be left unexecuted.
  In some cases, the administrator configures the settings so that users cannot specify the --enforce option. Check the pjacl command output item, execute pjhold(--enforce).
- If the hold state for an emergency job is canceled, the job is executed again as an emergency job.

The following example holds a job and cancels the hold.

```
$ pjhold 1 2
[INFO] PJM 0300 pjhold Accepted job 1.
[INFO] PJM 0300 pjhold Accepted job 2.

$ pjstat -v 1-2

  ACCEPT QUEUED  STGIN  READY RUNING RUNOUT STGOUT  HOLD ERROR TOTAL
       0      0      0      0      0      0      0     2     0     2
s      0      0      0      0      0      0      0     2     0     2
```

```
JOB_ID     JOB_NAME   MD ST  USER   ...     REASON
1          jobname1   NM HLD user1  ...     user1
2          jobname2   NM HLD user1  ...     user1

$ pjrls 1 2
[INFO] PJM 0400 pjrls Job 1 released.
[INFO] PJM 0400 pjrls Job 2 released.

$ pjstat -v 1-2

  ACCEPT QUEUED  STGIN  READY RUNING RUNOUT STGOUT   HOLD ERROR TOTAL
       0      2      0      0      0      0      0      0     0     2
s      0      2      0      0      0      0      0      0     0     2


JOB_ID     JOB_NAME   MD ST  USER   ...  LST ...  REASON
1          jobname1   NM QUE user1  ...  HLD ...  -
2          jobname2   NM QUE user1  ...  HLD ...  -
```

If the administrator submits a high-priority job called an "emergency job," users' jobs enter the hold state. This aborts the running jobs.

## 2.5.4 Changing job parameters

After a user submits a job, the user can use the pjalter command to change the following job parameters.

- Limit value on executable time of a job (elapse)
- Resource unit for executing a job (rscunit)
- Resource group for executing a job (rscgrp)
- Priorities among jobs of the same user (-p)

```
$ pjalter [-L | --rsc-list] rscname=value jobid       (1)
$ pjalter -p priority jobid                           (2)
```

(1) changing the upper limit
(2) changing the priority

### Note

- The user might not been permitted to execute as for the pjalter command.
- The pjalter command can be executed for emergency jobs, but their priority changes would be ignored.
- Job parameters can be changed when the job status is QUEUED, HOLD, or ERROR.

## 2.5.5 Referencing the output results of a running job

The job script of a job being executed can be checked with the -s option of the pjcat command.

```
$ pjcat -s jobid
```

While the staging function is in use for a job on the FX10 compute nodes, the output file of the job exists on the local file system of the compute node until it is staged out.

You can use the pjlist, pjcat, and pjget commands to reference the output file of the job on the compute node.

Table 2.30 Job output result reference when using the staging function [FX10]

| Command and argument | Description |
|---|---|
| pjlist [*option*] *jobID* [*rank*] | Displays the file list of the job execution directory.<br>The rank number *rank* is specified, the files under the rank number directory are listed. The rank number *rank* is omitted, the files under the top directory of the job are listed.<br>You can use -a, -l, and -R for *option*. They have the same meaning as the ls command options. For details, see the man page of the ls command.<br>-a : Displays the files which the name of file starts with dot ".".<br>-l : Displays the information such as file permission and file size.<br>-R : Displays the subdirectories recursively. |
| pjcat [-e \| -o] *job ID*<br>pjcat [-e \| -o] -f *job ID* | Displays the standard output of the job, and the standard error output of the job.<br>-e: Displays the standard error output.<br>-o: Displays the standard output.<br>-f: Continues displaying the contents until the job ends. |
| pjget [*option*] *job ID* [*rank*:]*src dst*<br>pjget [*option*] *jobID* [*rank*:]*src* [[*rank*:]*src* ...] *dst* | Copies the file (or directory) *src* under the job execution directory to the path *dst*. The rank number *rank* is specified, the file *src* is assumed to be the file under the rank number directory. The rank number *rank* is omitted, the file *src* is assumed to be the file under the top directory of the job.<br>If the path *dst* is a directory, the plural files *src* can be specified by delimiting them with a space.<br>You can use -f, -p, and -r for *option*. They have the same meaning as the cp command options. For details, see the man page of the cp command.<br>-f: deletes existing destination file, if necessary.<br>-p: set the owner and authority etc. of the source file to the destination file.<br>-r: copy the directory recursively. |

# 2.6 Checking Job Results

This section describes how to check job execution results.

## 2.6.1 Referencing job execution results

The standard output and standard error output of a job are created as separate files in the current directory at the job submission time. However, if the job is a bulk job or step job, the standard output and standard error output are output as separate files for each sub job.

The following table lists the default file names for the output destination. You can change the file name when submitting a job. (See "2.3.2.9 Specifying the standard output and standard error output files of a batch job.")

| Job output | Output destination file name |
|---|---|
| Standard output | *job name*.o.*job ID*<br>However, for a bulk job or step job, the job ID part is replaced by a sub job ID.<br>Example 1: The job name is job.sh, and the job ID is 123456.<br>job.sh.o.123456<br>Example 2: The job name is bulkjob.sh, and the sub job IDs of the bulk job are 123456[0], 123456[1], and so on.<br>bulkjob.sh.o.123456[0], bulkjob.sh.o.123456[1],...<br>Example 3: The job name is stepjob.sh, and the sub job IDs of the step job are 123456_0, 123456_1, and so on.<br>stepjob.sh.o.123456_0, stepjob.sh.o.123456_1,... |
| Standard error output | *job name*.e.*job ID*<br>However, for a bulk job or step job, the job ID part is replaced by a sub job ID.<br>Example 1: The job name is job.sh, and the job ID is 123456.<br>job.sh.e.123456 |

| Job output | Output destination file name |
|---|---|
| | Example 2: The job name is bulkjob.sh, and the sub job IDs of the bulk job are 123456[0], 123456[1], and so on.<br>bulkjob.sh.e.123456[0], bulkjob.sh.e.123456[1],...<br>Example 3: The job name is stepjob.sh, and the sub job IDs of the step job are 123456_0, 123456_1, and so on.<br>stepjob.sh.e.123456_0, stepjob.sh.e.123456_1,... |

## Note

- If the job script name begins with a single-byte numeric character, "J" is added to the beginning of the output file name.
- The job script name part of an output file name (including the letter "J" added to the beginning) consists of up to 63 characters.
- For a job submitted from the standard input and not by a job script, the job script name part is "STDIN".
- If the standard output and standard error output for a bulk job or step job have the same file name specified, the file will contain a mixture of output from each sub job.

The following message may be output to the job output. This message is output by the function of the parallel execution environment that generates processes on the compute node.

```
[????] PLE nnnn message text
or
[????] PLE nnnn plexec message text
Here, [????] means [ERR.] or [INFO], and nnnn is a four-digit integer.
"plexec" is the internal command of the parallel execution environment.
```

For example, the following message is output to the beginning of the standard output file of a job when the job is re-executed.

```
[INFO] PLE 1700 The job restarts.
```

## See

For details of the messages that the plexec command outputs, see "F.7.1 plexec command."

If an error occurs in a job, the user who executed the pjsub command for the job is notified of the error by e-mail. If job acceptance is rejected, such notification is not sent.

The following example shows e-mail notification.

```
Subject: PJM job: 6814. error.                    (1)

Job name:       pjmtest.sh                        (2)
Job owner:      user1                             (3)
Mail sent at:   Fri Jun 11 21:33:55 JST 2010      (4)


Reason: Node down.                                (5)
```

(1) Indication of abnormal end of the job of job ID 6814
(2) Job name
(3) Name of the user who submitted the job
(4) E-mail send time
(5) Cause of the abnormal end

For details on the messages in the e-mail notification, see "Appendix B Notification Message Related to Job Execution."

### Note

The mail is send to the account on the job management node, if --mail-list option of the pjsub command is not specified. The mail delivery to the user depends on the system design. For details, ask the administrator.

### Information

If an error occurs in a job, the reason is output to job statistical information.

## 2.6.2 Outputting job statistical information

If you specify the -s or -S option of the pjsub command when submitting a job, job statistical information will be output to either of the following files. One is a file (file name: "job name"+".i"+"job ID") in the current directory at the job submission time, and the other is the file specified in the --spath option. For details, see "A.2 Job Statistical Information."

Otherwise, you can reference the job statistical information on ended jobs by specifying the -H option and the -s or -S option of the pjstat command.

```
$ pjstat -H                                                                    (1)
...
JOB_ID     JOB_NAME   MD ST  USER     START_DATE      ELAPSE_LIM NODE_REQUIRE
1234567    jobname1   NM EXT usrname  01/01 00:00:00  0100:00:00 12:2x3x2
...

$ pjstat -H  -s 1234567                                                        (2)
...
```

(1) Checks for the job IDs of ended jobs.
(2) Specifies a job ID to display job statistical information. (The -S option displays details.)

For details on the displayed items and their meanings, see the man manual of the pjstat command.

### Note

- The pjstat command does not display the states of any jobs whose storage period has expired, so users cannot reference these jobs. The administrator sets the storage period of a job state.
- The statistical information output in job execution on FX100 or FX10 compute nodes differs slightly from that of PRIMERGY compute nodes.
- Information the prologue and epilogue processes set by the administrator is added to statistical information of the job.
  However, whether the elapsed time of job includes the prologue and epilogue processes depends on the system settings. For details, contact the administrator.
- The job statistical information of jobs executed with the FX100 compute node includes, besides the resources allocated to jobs, the assistant core use information related to the MPI asynchronous communication processing.

### Information

A job that is aborted because of a system problem, such as a node failure, may be automatically re-executed. (See "2.3.2.6 Specifying automatically re-execution for a job.") In such cases, the re-execution results are the job completion results.

# 2.7 Effects of Node Failures on Jobs

This section describes the effect of a node failure during job execution on the job.

## 2.7.1 Effect of a management node failure

In Technical Computing Suite, the failure of a management node (such as a control node or job management node) does not affect running jobs.
However, the response to the failure varies as follows depending on the system configuration used:

- If the management node is in a redundant configuration
  The failed active node automatically switches to the standby node so that job operation continues.

- If the management node is not in a redundant configuration
  Even if the control node fails, the job operation is continued.
  If the failed node is a job management node or job management sub node, new jobs can neither be submitted nor executed in the cluster or node group which is managed failed node.
  If the failed node is a storage cluster management node, MDS node, OSS node or IO node, submission of new job or input/output of the job may not be processed normally.

## 2.7.2 Effect of a compute node failure

Any failure of a compute node during job execution has the following effect on the job.

Table 2.31 Effects on interactive jobs or jobs that do not use the staging function

| Target | Re-execution allowed (*) | Re-execution not allowed (*) or interactive job |
|---|---|---|
| Job | The job is canceled and transitions to the QUEUED state. After computer resources are assigned, the job is re-executed. | The job is canceled and terminated. |
| Job statistical information file (*.i file) | Job statistical information from the job cancellation time is not output. The job re-execution results are output. | Job statistical information from the job cancellation time is output. |
| Standard output (*.o file) and standard error output (*.e file) of the job | The results until the job was canceled and the job re-execution results are output. | The results until the job was canceled are output. |
| File output by the job | The results until the job was canceled are output.<br>If an existing file at the re-execution time has the same name, the handling depends on the job file operation specifications. | The results until the job was canceled are output. |

Table 2.32 Effects on jobs that use the staging function

| Target | Re-execution allowed (*) | Re-execution not allowed (*) |
|---|---|---|
| Job | The job is canceled and transitions to the QUEUED state. After computer resources are assigned, the job is re-executed. | The job is canceled and terminated. |
| Job statistical information file (*.i file) | Job statistical information from the job cancellation time is not output. The job re-execution results are output. | Job statistical information from the job cancellation time is output. |
| Standard output (*.o file) and standard error output (*.e file) of the job | The output results up to the job cancellation time are not staged out and are lost. | The results until the job was canceled are output, and their files are staged out. |
| Stage-out file (pjsub --stgout *file*) | The output files up to the job cancellation time are not staged out and are lost. | The existing file at the job cancellation time is staged out. |
| Stage-out directory (pjsub --stgout-dir *dir*) | The output files under the directory up to the job cancellation time are not staged out and are lost. | The existing directory at the job cancellation time is staged out. |

(*) The --restart or --norestart option specified at the job submission time or the job ACL function settings specify whether job re-execution is permitted.

# Appendix A Job Information

This appendix describes the information of jobs output with the pjstat command and the pjsub command.

Note

Bulk jobs and step jobs have information corresponding to the job ID and information corresponding to the subjob ID.
In this appendix, the former in particular is called summary information of the bulk or step job.

## A.1 Output of the pjstat and pjstat -v

The information output by the pjstat command and its meaning is described below.

```
$ pjstat

  ACCEPT QUEUED  STGIN  READY RUNING RUNOUT STGOUT   HOLD  ERROR   SWAP   TOTAL
  nnnnnn nnnnnn nnnnnn nnnnnn nnnnnn nnnnnn nnnnnn nnnnnn nnnnnn nnnnnn nnnnnnn
s nnnnnn nnnnnn nnnnnn nnnnnn nnnnnn nnnnnn nnnnnn nnnnnn nnnnnn nnnnnn nnnnnnn

JOB_ID     JOB_NAME   MD ST  USER     START_DATE      ELAPSE_LIM NODE_REQUIRE     VNODE  CORE
(*)V_MEM
XXXXXXXXXX XXXXXXXXXX XX XXX XXXXXXXX MM/DD hh:mm:ss  hhhh:mm:ss nnnnnn:XXxYYxZZ  nnnnnn nnn
(*)nnnnnnnnnnMiB
```

(*) This line continues from the previous line and is actually displayed on the previous line.

Table A.1 Output items of the pjstat command (no -v option specification)

| Name | Description |
|---|---|
| ACCEPT | Number of jobs waiting for job acceptance |
| QUEUED | Number of jobs waiting for job execution |
| STGIN<br>[FX10] | Number of jobs during stage-in |
| READY | Number of jobs waiting for job execution |
| RUNING | Number of jobs being executed |
| RUNOUT | Number of jobs waiting job end |
| STGOUT<br>[FX10] | Number of jobs during stage-out |
| HOLD | Number of jobs in fixed state due to a user |
| ERROR | Number of jobs in fixed state due to an error |
| SWAP<br>[FX100] | Number of jobs during swap-out period (Status SWAPOUT, SWAPPED, SWAPIN)<br>The SWAP column is displayed only when the administrator enables display. |
| TOTAL | Number of total jobs |
| JOB_ID | Job ID (10-digit decimal number)<br>For the sub job, the sub job ID is output. (*jobID*[*BulkNumber*] *or jobID_StepNumber*) |
| JOB_NAME | Job name (Only first 10 characters) |
| MD | Job model<br>NM: Normal job<br>ST: Step job<br>BU: Bulk job |

| Name | Description |
|---|---|
| ST | Current processing state of the job<br>ACC: Accepted job submission<br>CCL: Exit by job cancelation<br>ERR: In fixed state due to an error<br>EXT: Completed of the job termination processing<br>HLD: Holding status by user operation<br>QUE: Waiting for job execution<br>RDY: Ready to execute the job [FX10]<br>RJT: Rejected job submission<br>RNA: Acquiring resources required for job execution<br>RNE: In the processing of the epilogue<br>RNO: Waiting for completion of job termination processing<br>RNP: In the processing of the prologue<br>RUN: Executing job<br>SIN: In the processing of the stage-in [FX10]<br>SOT: In the processing of the stage-out [FX10]<br>SWD: Swap-out completed [FX100]<br>SWI: In the processing of the swap-in [FX100]<br>SWO: In the processing of the swap-out [FX100] |
| USER | Name of user name who executes job (Only first 8 characters) |
| START_DATE | Planned execution start time or execution start time<br>- (*MM*/*DD hh*:*mm*)<br>Planned execution start time. The planned execution start time is enclosed in parentheses.<br>- (*YYYY*/*MM*/*DD*)<br>Planned execution start time (after one year or later)<br>- (*MM*/*DD hh*:*mm*)#<br>If the planned execution start time is after the scheduling period, "#" is added after the time. In this case, the planned execution start time is after displayed time.<br>- *MM*/*DD hh*:*mm*:*ss*<br>Actual start time of the job<br>- (*MM*/*DD hh*:*mm*)< or *MM*/*DD hh*:*mm*:*ss*<<br>As for jobs to which backfill is applied, "<" is added after the time.<br>- (*MM*/*DD hh*:*mm*)@ or *MM*/*DD hh*:*mm*:*ss*@<br>As for jobs to which the execution start time is specified, "@" is added after the time.<br>For the summary information of the step job, the information of the running sub job is output.<br>If there is no running sub job, the information of the sub job to be executed next is output. |
| ELAPSE_LIM | Elapsed time limit<br>"*hhhh*:*mm*:*ss*"<br>When column overflow occurs, output is executed with *ss* omitted.<br>For the summary information of the step job, "-" is output. |
| NODE_ REQUIRE | - FX100 or FX10 compute node<br>- For a node-exclusive job<br>Shape and number of nodes when job is submitted: "*nnnnnn*:*XX*x*YY*x*ZZ*"<br>When format above is too short to hold information, only node shape is output.<br>- For a node-sharing job<br>Number of virtual nodes specified when the job is submitted. At present, "1" is output.<br>- PRIMERGY compute node<br>Number of nodes when the job is submitted: "*nnnnnn*"<br>If nothing is specified, "-" is output. |

| Name | Description |
|---|---|
| | For the summary information of the step job, "-" is output. |
| VNODE | Number of virtual nodes "*nnnnnn*"<br>For a job executed on an FX100 or FX10 compute node, "-" is output. |
| CORE | Number of CPU cores per virtual node "*nnn*"<br>For a job executed on an FX100 or FX10 compute node, "-" is output. |
| V_MEM | Amount of memory per virtual node " *nnnnnnnnn*MiB"<br>This is the value of the parameter vnode-mem or mem that is specified when the job is submitted.<br>For a job with core-mem specified, the value is multiplied by the number of CPU cores per virtual node. core-mem is the memory amount per CPU core.<br>For a job executed on an FX100 or FX10 compute node, "-" is output. |

You can output additional information by adding the additional information option -v.

```
$ pjstat -v

  ACCEPT QUEUED  STGIN  READY RUNING RUNOUT STGOUT   HOLD  ERROR   SWAP   TOTAL
  nnnnnn nnnnnn nnnnnn nnnnnn nnnnnn nnnnnn nnnnnn nnnnnn nnnnnn nnnnnn nnnnnnn
s nnnnnn nnnnnn nnnnnn nnnnnn nnnnnn nnnnnn nnnnnn nnnnnn nnnnnn nnnnnn nnnnnnn

JOB_ID     JOB_NAME   MD ST  USER     GROUP   START_DATE      ELAPSE_TIM ELAPSE_LIM (*)
NODE_REQUIRE    VNODE  CORE V_MEM        V_POL E_POL RANK       LST EC  PC  SN PRI (*)
ACCEPT          RSC_UNIT REASON
XXXXXXXXXX XXXXXXXXXX XX XXX XXXXXXXX XXXXXXX MM/DD hh:mm:ss< hhhh:mm:ss hhhh:mm:ss (*)
nnnnnn:XXxYYxZZ nnnnnn nnn nnnnnnnnnnMiB XXXXX XXXXX XXXXXXX    XXX XXX XXX XX XXX (*)
MM/DD hh:mm:ss  XXXXXXXX XXXXXXXXXXXXXXXX
```

(*) Actually, this line continues to the next line.

Table A.2 Output items of the pjstat command (items added at specification of -v option)

| Name | Description |
|---|---|
| GROUP | Group name of user executing job<br>(Only first 8 characters)<br>For the summary information of the step job, the information of the running sub job is output.<br>If there is no running sub job, the information of the sub job to be executed next is output. |
| ELAPSE_TIM | Elapsed time limit "*hhhh:mm:ss*"<br>When column overflow occurs, output is executed with *ss* omitted.<br>The job elapsed time does not include the swap period.<br>For the summary information of the bulk job or step job, "-" is output. |
| V_POL<br>[PG] | Virtual node placement policy<br>A_PCK : Absolutely PACK<br>PACK : PACK<br>A_UPK : Absolutely UNPACK<br>UPCK : UNPACK<br>For a job executed on an FX100 or FX10 compute node, or the summary information of the step job, "-" is output. |
| E_POL<br>[PG] | Execution mode policy<br>SHARE : SHARE<br>SMPLX : SIMPLEX<br>For a job executed on an FX100 or FX10 compute node, or the summary information of the step job,"-" is output. |
| RANK<br>[PG] | Allocation methods with a rank map<br>bynode : rank-map-bynode<br>bychip : rank-map-bychip |

| Name | Description |
|---|---|
| | For a job executed on an FX100 or FX10 compute node, or the summary information of the step job, "-" is output. |
| LST | Last processing state of the job (prior state from which the "current processing state of the job" is shifted)<br>For the meaning of the value, see the item ST in "Table A.1 Output items of the pjstat command (no -v option specification))." |
| EC | Exit code of a job script<br>For the summary information of the bulk job or step job, "-" is output. |
| PC | Job end code (PJM code)<br>Code that indicates the job manager processing result of the job execution<br>For the summary information of the bulk job or step job, "-" is output.<br>The meanings of the codes are as follows.<br>0: Normal end of job<br>1: Cancellation with pjdel command<br>2: Reject by the judgment of the job accept ion<br>3: Reject the execution due to the ticket control<br>4: Holding with pjhold command<br>6: Cancellation due to evaluation of step job dependency expression<br>7: Cancellation due to deadline forcibly<br>8: Cancellation due to the ticket control<br>9: Specification of norestart job, so job is EXIT when rebuild Job information.<br>10: Job execution timeout due to CPU time limit violation<br>11: Job execution timeout due to elapsed time limit violation<br>12: Forced termination due to use of excessive amount of memory<br>13: Forced termination due to use of excessive amount of disk space<br>14: Waiting for changing the status of the bulk job<br>16: Job termination due to an current directory or standard input/standard output/standard error output file inaccessible.<br>20: Node failure<br>21: Failure in job script execution<br>22: ICC error<br>23: Termination due to OOM Killer<br>24: Failure of collecting the enhanced CPU statistical information (The job statistical information EXEC INST NUM and EXEC SIMD NUM) [FX10]<br>25: Failure in HA<br>26: Error in the prologue or epilogue process<br>27: Error in the resource management exit process<br>30: Job swap abort due to swap-out processing or swap-in processing failure [FX100]<br>31: Job swap abort due to job that cannot be swapped out [FX100]<br>50: Error in the stage-in process (shift to the ERROR state) [FX10]<br>51: Error in the stage-out process (shift to the ERROR state) [FX10]<br>52: Job termination due to an error of stage-in processing [FX10]<br>53: An error of stage-in processing (the processing continues) [FX10]<br>54: An error of stage-out processing (the processing continues) [FX10]<br>100: Internal error in the job manager<br>120: Internal error in the job scheduler<br>140: Internal error in job resource management<br>160: Internal error in the Tofu library [FX100/FX10] |
| SN | Signal number<br>For the summary information of the bulk job or step job, "-" is output. |
| PRI | Job priority<br>The priority of an emergency job is 256. The priority of a non-emergency job is 0 to 255.<br>For the summary information of the step job, "-" is output. |

| Name | Description |
|---|---|
| ACCEPT | Job submission date and time "*MM*/*DD hh*:*mm*:*ss*"<br>For the summary information of the step job, "-" is output. |
| RSC_UNIT | Resource unit when job is submitted<br>For the summary information of the step job, "-" is output. |
| REASON | Error message<br>Message corresponding to result code for some processing of a job regardless of whether job is executed<br>For the summary information of the bulk job or step job, "-" is output.<br>The meanings of output messages are as follows.<br>-: No error<br>*MM*/*DD hh*:*mm* DELAY : Job execution is scheduled after the specified start time.<br>CPUTIME LIMIT EXCEEDED : Exceeded limit of the CPU time<br>ELAPSE LIMIT EXCEEDED: Exceeded limit of the elapse time<br>FILE IO ERROR : Current directory at which the job is submitted is not accessible<br>GATE CHECK: Canceled by ticket control<br>IMPOSSIBLE SCHED: Impossible scheduling<br>INSUFF CPU: Insufficient number of physical CPUs<br>INSUFF DISK: Insufficient physical capacity of disks [FX10]<br>INSUFF MEMORY: Insufficient physical memory<br>INSUFF NODE: Insufficient number of physical nodes<br>INTERNAL ERROR: Internal error<br>INVALID HOSTFILE : Invalid host file specified in the rank-map-hostfile parameter of the pjsub command [FX100/FX10]<br>LIMIT OVER DISK: Exceeded limit of disk usage in job execution [FX10]<br>LIMIT OVER MEMORY: Exceeded limit of memory in job execution<br>LOST COMM: All-to-all communication among parallel processes not guaranteed<br>NO CURRENT DIR : Unable to access current directory or standard input/standard output/standard error output file at user job submission<br>NOT PREFERRED: Scheduling is suppressed during the selection of a job to be stopped [FX100]<br>RSCGRP NOT EXIST: Resource group does not exist<br>RSCGRP STOP : Resource group stopped<br>RSCUNIT NOT EXIST: Resource unit does not exist<br>RSCUNIT STOP: Resource unit stopped<br>RUNLIMIT EXCEED: Exceeded limit number for simultaneous execution<br>STGIN ERROR: Error in stage-in [FX10]<br>STGOUT ERROR: Error in stage-out [FX10]<br>SWAPOUT WAITING: Waiting for the stop processing of the running job to complete [FX100]<br>USELIMIT EXCEED : Waiting for execution due to limitation of simultaneous node or CPU core usage<br>USER NOT EXIST : Job execution user does not exist in the system<br>Other character strings : *reasonmessage* specified with the --reason option to the pjdel, pjhold command<br>On the job holding by the pjhold command, the string is output as "*username*: *reasonmessage*". If the --reason option is omitted, the string is output as "*username:*"<br>(*) "*username*" is the name of user who executed the pjhold command. |

# A.2 Job Statistical Information

Job statistical information is output when the -s or -S option of the pjstat command and pjsub command is specified.

- pjstat -s output example

```
$ pjstat -s

  ACCEPT QUEUED  STGIN  READY RUNING RUNOUT STGOUT   HOLD  ERROR   TOTAL
       0      0      0      0      1      0      0      0      0       1
```

```
s      0      0      0      0      1      0      0      0      0      1

 JOB ID                      : 100
 JOB NAME                    : job.sh
 JOB TYPE                    : BATCH
 JOB MODEL                   : NM
 ...
 Name                        : Value
 ...
 START BULKNO                : -
 END BULKNO                  : -
```

- pjstat -S output example

```
$ pjstat -S

  ACCEPT QUEUED  STGIN  READY RUNING RUNOUT STGOUT   HOLD  ERROR   TOTAL
       0      0      0      0      1      0      0      0      0       1
s      0      0      0      0      1      0      0      0      0       1

 JOB ID                      : 100
 JOB NAME                    : job.sh
 JOB TYPE                    : BATCH
 JOB MODEL                   : NM
 ...
 START BULKNO                : -
 END BULKNO                  : -

 VNODE ID                    : -            (*)
 NODE ID                     : 0x00014DBF
 ...
 CPU BITMAP (USE)            : 0xF
 RANK NO                     : 0

(*) Information for each node or virtual node
```

- pjsub -s output example (job statistical information file)

```
Job Information

 JOB ID                    : 100
 JOB NAME                  : job.sh
 JOB TYPE                  : BATCH
 ...
 Name                      : Value
 ...
 START BULKNO              : -
 END BULKNO                : -
```

- pjsub -S output example (job statistical information file)

```
Job Information

 JOB ID                    : 100
 JOB NAME                  : job.sh
 JOB TYPE                  : BATCH
 ...
 Name                      : Value
 ...
 START BULKNO              : -
 END BULKNO                : -

--------------------------------------------------------------------------------
Node Information                              (*)
```

```
 NODE ID                   : 0x00014DBE
 TOFU COORDINATE           : (23,17,16)
 ...
 CPU BITMAP (USE)          : 0xF
 RANK NO                   : 0

--------------------------------------------------------------------------------
Node Information                                (*)

 NODE ID                   : 0x00014DBE
 ...
 CPU BITMAP (USE)          : 0xF
 RANK NO                   : 0

(*)Statistic information for each node or virtual node
   For jobs on the PRIMERGY compute node, "VNode Information" applies.
```

The statistical information output with the pjstat command and pjsub command is as shown in "Table A.3 Job statistical information items (items output by -s option or -S option)" and "Table A.4 Job statistical information items per node or virtual node (output only with -S option specified)."
The symbols appearing in the pjstat column and pjsub column have the following meanings:
O: The item is output. However, note that the value may be a blank or hyphen.
-: No items are output.
FH, FX: The item is output only for jobs of FX100 (FH) or FX10 (FX) compute nodes.
PG: The item is output only for jobs of PRIMERGY compute node jobs.
*: The item may be set by the administrator so that it is not output.

Table A.3 Job statistical information items (items output by -s option or -S option)

| Item name | Description | pjstat -s/-S | pjsub -s/-S |
|---|---|---|---|
| JOB ID | Job ID<br>For the sub job, the sub job ID is output. | O | O |
| JOB NAME | Job name<br>Job script file name is output as job name.<br>If job is submitted from standard input, "STDIN" is output.<br>For the summary information of the step job, this item is the job name of the running sub job (or next sub job if there is no running sub job) in the output of the pjstat command or a blank in the output of the pjsub command. | O | O |
| JOB TYPE | Job type<br>BATCH : Batch job<br>INTERACT : Interactive job | O | O |
| JOB MODEL | Job model<br>NM : Normal job<br>ST : Step job<br>BU : Bulk job | O | O |
| RETRY NUM | Number of retries<br>Number of job retries<br>For the summary information of the bulk job or step job, the total for sub jobs is output. | O | O |
| SUB JOB NUM | Number of sub jobs in the bulk job or the step job<br>The output format is "*number of completed sub jobs*/*total number of sub jobs*".<br>For the normal job and the sub job, "-" is output. | O | O |

| Item name | Description | pjstat -s/-S | pjsub -s/-S |
|---|---|---|---|
| USER | User name<br>Name of user who executes job | O | O |
| GROUP | Group name<br>Name of group that executes job | O | O |
| RESOURCE UNIT | Resource unit name for which job was executed<br>For the summary information of the step job, this item is the job name of the running sub job (or next sub job if there is no running sub job) in the output of the pjstat command or a blank in the output of the pjsub command. | O | O |
| RESOURCE GROUP | Resource group name for which job was executed<br>For the summary information of the step job, this item is "-" in the output of the pjstat command or a blank in the output of the pjsub command. | O | O |
| APRIORITY | Priority in the resource unit<br>0: Low priority, 255: High priority<br>For the summary information of the step job, this item is "-" in the output of the pjstat command or 0 in the output of the pjsub command. | O | O |
| PRIORITY | Priority<br>0: Low priority, 255: High priority<br>For the summary information of the step job, this item is "-" in the output of the pjstat command or 0 in the output of the pjsub command.<br>The priority of an emergency job is 256.<br>The priority of a non-emergency job is 0 to 255. | O | O |
| SHELL | Job execution shell<br>Absolute path name of shell to be used for execution of job script<br>For the summary information of the step job, this item is "-" in the output of the pjstat command or a blank in the output of the pjsub command. | O | O |
| COMMENT | Comment<br>For the summary information of the step job, "-" is output. | O | O |
| LAST STATE | Last state of job<br>Transition from prior state to ended job state<br>ACC: Accepted job submission<br>ERR: In fixed state due to an error<br>HLD: Holding status by user operation<br>QUE: Waiting for job execution<br>RDY: Ready to execute the job [FX10]<br>RNA: Acquiring resources required for job execution<br>RNE: In the processing of the epilogue<br>RNO: Waiting for completion of job termination processing<br>RNP: In the processing of the prologue<br>RUN: Executing job<br>SIN: In the processing of the stage-in [FX10]<br>SOT: In the processing of the stage-out [FX10]<br>For the summary information of the step job, this item is "-" in the output of the pjsub command. | O | O |
| STATE | State of job<br>ACC: Accepted job submission<br>CCL: Ended due to cancellation of job execution<br>ERR: In fixed state due to an error<br>EXT: Completed job termination processing<br>HLD: Holding status by user operation<br>QUE: Waiting for job execution | O | O |

| Item name | Description | pjstat -s/-S | pjsub -s/-S |
|---|---|---|---|
| | RDY: Ready to execute the job [FX10]<br>RNA: Acquiring resources required for job execution<br>RNE: In the processing of the epilogue<br>RNO: Waiting for completion of job termination processing<br>RNP: In the processing of the prologue<br>RUN: Executing job<br>SIN: In the processing of the stage-in [FX10]<br>SOT: In the processing of the stage-out [FX10]<br>SWD: Swap-out completed [FX100]<br>SWI: In the processing of the swap-in [FX100]<br>SWO: In the processing of the swap-out [FX100] | | |
| PRM DATE | PRM collection time<br>Time when data was collected from compute nodes<br>For the summary information of the bulk job or step job, "-" is output. | O | O |
| EXIT CODE | Exit code<br>Exit code of job script<br>For the summary information of the bulk job or step job, "-" is output. | O | O |
| SIGNAL NO | Signal number<br>Signal number when job script is ended due to receipt of signal<br>For the summary information of the bulk job or step job, "-" is output. | O | O |
| PJM CODE | Job end code (PJM code)<br>Code that indicates job manager processing result of job execution<br>For the summary information of the bulk job or step job, "-" is output.<br>The meanings of the codes are as follows.<br>0: Normal end of job<br>1: Cancellation with pjdel command<br>2: Reject by the judgment of the job acception<br>3: Reject the execution due to the ticket control<br>4: Holding with pjhold command<br>6: Cancellation due to evaluation of step job dependency expression<br>7: Cancellation due to deadline forcibly<br>8: Cancellation due to the ticket control<br>9: Specification of norestart job, so job is EXIT when rebuild Job information.<br>10: Job execution timeout due to CPU time limit violation<br>11: Job execution timeout due to elapsed time limit violation<br>12: Forced termination due to use of excessive amount of memory<br>13: Forced termination due to use of excessive amount of disk space<br>14: Waiting for changing the status of the bulk job<br>16: Job termination due to an current directory or standard input/ standard output/standard error output file inaccessible.<br>20: Node failure<br>21: Failure in job script execution<br>22: ICC error<br>23: Termination due to OOM Killer<br>24: Failure of collecting the enhanced CPU statistical information (The job statistical information EXEC INST NUM and EXEC SIMD NUM) [FX10]<br>25: Failure in HA<br>26: Error in the prologue or epilogue process<br>27: Error in the resource management exit process<br>30: Aborted because swap-out or swap-in processing failed during job swap [FX100]<br>31: Aborted because the job cannot be swapped out [FX100] | O | O |

| Item name | Description | pjstat -s/-S | pjsub -s/-S |
|---|---|---|---|
| | 50: Error in the stage-in process (shift to the ERROR state) [FX10]<br>51: Error in the stage-out process (shift to the ERROR state) [FX10]<br>52: Job termination due to an error of stage-in processing [FX10]<br>53: An error of stage-in processing (the processing continues) [FX10]<br>54: An error of stage-out processing (the processing continues) [FX10]<br>100: Internal error in the job manager<br>120: Internal error in the job scheduler<br>140: Internal error in job resource management<br>160: Internal error in the Tofu library [FX100/FX10] | | |
| REASON | Error message<br>Message corresponding to result code for some processing of a job regardless of whether job is executed<br>For the summary information of the bulk job or step job, "-" is output.<br>The meanings of output messages are as follows.<br>-: No error<br>*MM/DD hh:mm* DELAY : Job execution is scheduled after the specified start time.<br>CPUTIME LIMIT EXCEEDED : Exceeded limit of the CPU time<br>ELAPSE LIMIT EXCEEDED: Exceeded limit of the elapse time<br>FILE IO ERROR : Current directory at which the job is submitted is not accessible<br>GATE CHECK: Canceled by ticket control<br>IMPOSSIBLE SCHED: Impossible scheduling<br>INSUFF CPU: Insufficient number of physical CPUs<br>INSUFF DISK: Insufficient physical capacity of disks [FX10]<br>INSUFF MEMORY: Insufficient physical memory<br>INSUFF NODE: Insufficient number of physical nodes<br>INTERNAL ERROR: Internal error<br>INVALID HOSTFILE : Invalid host file specified in the rank-map-hostfile parameter of the pjsub command [FX100/FX10]<br>LIMIT OVER DISK: Exceeded limit of disk usage in job execution [FX10]<br>LIMIT OVER MEMORY: Exceeded limit of memory in job execution<br>LOST COMM: All-to-all communication among parallel processes not guaranteed<br>NO CURRENT DIR : Unable to access current directory or standard input/standard output/standard error output file at user job submission.<br>NOT PREFERRED: Scheduling is suppressed during the selection of a job to be stopped [FX100]<br>RSCGRP NOT EXIST: Resource group does not exist<br>RSCGRP STOP : Resource group stopped<br>RSCUNIT NOT EXIST: Resource unit does not exist<br>RSCUNIT STOP: Resource unit stopped<br>RUNLIMIT EXCEED: Exceeded limit number for simultaneous execution<br>STGIN ERROR: Error in stage-in [FX10]<br>STGOUT ERROR: Error in stage-out [FX10]<br>SWAPOUT WAITING: Waiting for the stop processing of the running job to complete [FX100]<br>USELIMIT EXCEED : Waiting for execution due to limitation of simultaneous node or CPU core usage<br>USER NOT EXIST : Job execution user does not exist in the system<br>Other character strings : *reasonmessage* specified with the --reason option to the pjdel, pjhold command<br>On the job holding by the pjhold command, the string is output as | O | O |

| Item name | Description | pjstat -s/-S | pjsub -s/-S |
|---|---|---|---|
| | "*username*: *reasonmessage*". If the --reason option is omitted, the string is output as "*username*:"<br>(*)"*username*" is the name of user who executed the pjhold command. | | |
| MAIL SEND FLAG | Whether mail is sent<br>The flag is shown by following characters.<br>b: At the start of the processing<br>e: At the end of the processing<br>r: Error occured<br>s: Statistical information / S Statistical information with node information:<br>-: do not send the mail<br>"bers" or "berS" shows that the mail will be send for all status.<br>"b-r-" shows that the mail will be send at the begging of the processing and error. | O | O |
| MAIL ADDRESS | Mail address<br>For the summary information of the step job, this item is "-" in the output of the pjstat command or a blank in the output of the pjsub command. | O | O |
| STEP DEPENDENCY EXP | Step job dependency<br>For the job other than sub job of step job, "-" is output. | O | O |
| STEP EXITING WAIT MODE | Attribute of wait for transfer of step job result<br>If --step --sparam "send=yes" is specified to the pjsub command, it is 1.<br>If --step --sparam "send=no" is specified to the pjsub command, it is 2.<br>For the job other than sub job of step job, "-" is output. | O | O |
| FILE MASK | File mask<br>umask value of user who submits job<br>For the summary information of the step job, this item is "-" in the output of the pjstat command or "0000" in the output of the pjsub command. | O | O |
| STANDARD OUT FILE | Standard output file name<br>Path name of standard output file of job<br>For the interactive job and the summary information of the bulk job or step job, "-" is output. | O | O |
| STANDARD ERR FILE | Standard error output file name<br>Path name of standard error output file of job<br>For the interactive job and the summary information of the bulk job or step job, "-" is output. | O | O |
| INFORMATION FILE | Statistical information file name<br>Path name of file in which statistical information of job currently displayed is stored<br>For the summary information of the step job, this item is "-" in the output of the pjstat command or a blank in the output of the pjsub command. | O | O |
| PJSUB DIRECTORY | Path name of the directory on which the job was submitted.<br>For the summary information of the step job, this item is a blank. | O | O |
| FILE SYSTEM NAME | The name of external file system that is specified to the pjsub command.<br>For the summary information of the step job, this item is a blank. | O | O |
| APPLICATION NAME | The application name that was specified to the pjsub command.<br>For the summary information of the step job, this item is a blank. | O | O |
| ACCEPT DATE | Job submission time "*YYYY*/*MM*/*DD HH*:*MM*:*SS*"<br>For the summary information of the step job, this item is "-" in the output of the pjstat command or the acceptance time of the first submitted sub job in the output of the pjsub command | O | O |

| Item name | Description | pjstat -s/-S | pjsub -s/-S |
|---|---|---|---|
| QUEUED DATE | Last queuing time "*YYYY*/*MM*/*DD HH*:*MM*:*SS*"<br>For the summary information of the bulk job or step job, "-" is output. | O | O |
| EXIT DATE | EXIT and CANCEL state transition time<br>"*YYYY*/*MM*/*DD HH*:*MM*:*SS*"<br>For the summary information of the bulk job or step job, "-" is output. | O | O |
| LAST HOLD USER | User name who hold the job in last time<br>For the summary information of the bulk job or step job, "-" is output. | O | O |
| HOLD NUM | Times of holding the job<br>For the summary information of the bulk job or step job, "-" is output. | O | O |
| HOLD TIME | Accumulation time that the job is in the HOLD status<br>For the summary information of the bulk job or step job, the total for sub jobs is output. | O | O |
| JOB START DATE | Job execution start time "*YYYY*/*MM*/*DD HH*:*MM*:*SS*"<br>For the summary information of the step job, this item is the start time of the running sub job in the output of the pjstat command or "-" in the output of the pjsub command. | O | O |
| JOB END DATE | Job execution end time "*YYYY*/*MM*/*DD HH*:*MM*:*SS*"<br>For the summary information of the bulk job or step job, "-" is output. | O | O |
| JOB DELETE DATE (REQUIRE) | Deletion request time "*YYYY*/*MM*/*DD HH*:*MM*:*SS*"<br>For the summary information of the bulk job or step job, "-" is output. | O | O |
| JOB DELETE DATE | Deletion time "*YYYY*/*MM*/*DD HH*:*MM*:*SS*"<br>For the summary information of the bulk job or step job, "-" is output. | O | O |
| STAGE IN START DATE<br>[FX10] | Stage-in start time "*YYYY*/*MM*/*DD HH*:*MM*:*SS*"<br>For the summary information of the job not using staging, the bulk job, and the step job, and for jobs on FX100 and PRIMERGY compute nodes, "-" is output. | O | FH/FX |
| STAGE IN END DATE<br>[FX10] | Stage-in end time "*YYYY*/*MM*/*DD HH*:*MM*:*SS*"<br>For the summary information of the job not using staging, the bulk job, and the step job, and for jobs on FX100 and PRIMERGY compute nodes, "-" is output. | O | FH/FX |
| STAGE IN SIZE<br>[FX10] | Amount of file transfer that occurs in the stage-in process<br>Display format is *x* TB (*y*), *x* GB (*y*), or *x* MB (*y*).<br>The value *y* represents the value *x* in megabytes, kilobytes, or bytes according to whether the unit of the value *x* is TB (terabytes), GB (gigabytes), or MB (megabytes).<br>For the summary information of the bulk job or step job, the total for sub jobs is output.<br>For the job on the FX100 compute node, "0.0MB (0)" is output.<br>For the job on the PRIMERGY compute node, "-" is output. | O | FH/FX |
| STAGE OUT START DATE<br>[FX10] | Stage-out start time "*YYYY*/*MM*/*DD HH*:*MM*:*SS*"<br>For the summary information of the job not using staging, the bulk job, and the step job, and for jobs on FX100 and PRIMERGY compute nodes, "-" is output. | O | FH/FX |
| STAGE OUT END DATE<br>[FX10] | Stage-out end time "*YYYY*/*MM*/*DD HH*:*MM*:*SS*"<br>For the summary information of the job not using staging, the bulk job, and the step job, and for jobs on FX100 and PRIMERGY compute nodes, "-" is output. | O | FH/FX |
| STAGE OUT SIZE<br>[FX10] | Amount of file transfer that occurs in the stage-out process<br>Display format is x TB (y), x GB (y), or x MB (y). | O | FH/FX |

| Item name | Description | pjstat -s/-S | pjsub -s/-S |
|---|---|---|---|
| | The value y represents the value x in megabytes, kilobytes, or bytes according to whether the unit of the value x is TB (terabytes), GB (gigabytes), or MB (megabytes).<br>For the summary information of the bulk job or step job, the total for sub jobs is output.<br>For the job on the FX100 compute node, "0.0MB (0)" is output.<br>For the job on the PRIMERGY compute node, "-" is output. | | |
| NODE NUM (REQUIRE) | 1. FX100 or FX10 compute node: For a node-exclusive job<br>Number of required nodes and shape<br>- Three-dimensional shape *N*: *X*x*Y*x*Z*[:mesh\|:noncont]<br>- Two-dimensional shape *N*: *X*x*Y*[:mesh\|:noncont]<br>- One-dimensional shape *N*[:mesh\|:noncont]<br>(*) *N* is number of nodes and *X, Y,* and *Z* are coordinate values.<br>":mesh" represents mesh mode and ":noncont" represents non-contiguous mode.<br>If neither ":mesh" nor ":noncont" is displayed, it means that torus mode is being used.<br>2. FX100 or FX10 compute node: For a node-sharing job<br>*N*: Number of required virtual nodes<br>3. PRIMERGY<br>Number of required nodes<br>"-" is output when the number of nodes is not specified in job submitting.<br>For the summary information of the step job, this item is "-" in the output of the pjstat command, or 0 (FX100/FX10) or "-" (PG) in the output of the pjsub command.<br>For the summary information of the step job containing both a sub job that runs on an FX100 or FX10 compute node and a sub job that runs on a PRIMERGY compute node, this item is "-" in the output of the pjsub command. | O | O |
| VNODE NUM (ALLOC)<br>[PG] | The number of allocated virtual nodes<br>For the summary information of the bulk job, the total for sub jobs is output.<br>For the summary information of the step job or the job on FX100 and FX10 compute node, "-" is output. | O | PG |
| CPU NUM (REQUIRE) | Number of required CPU cores<br>For the summary information of the step job, this item is "-" in the output of the pjstat command, or 0 (FX100/FX10) or "-" (PG) in the output of the pjsub command.<br>For the summary information of the step job containing both a sub job that runs on an FX100 or FX10 compute node and a sub job that runs on a PRIMERGY compute node, this item is "-" in the output of the pjsub command. | O | O |
| ELAPSE TIME (LIMIT) | Elapsed time limit value "*DD HH*:*MM*:*SS* (*s*)"<br>Elapsed time limit value specified when job is submitted<br>Display format is "*DD HH*:*MM*:*SS* (*s*)".<br>If time is no more than 24 hours, *DD* (number of days) is omitted.<br>For the summary information of the step job, "00:00:00 (0)" is output. | O | O |
| MEMORY SIZE (LIMIT) | Physical memory amount limit value for each node<br>For unlimited, "unlimited" is displayed.<br>If it is not specified in command, <DEFAULT> is displayed at end of | O | O |

| Item name | Description | pjstat -s/-S | pjsub -s/-S |
|---|---|---|---|
| | display.<br>For the summary information of the bulk job or step job, "-" is output. | | |
| DISK SIZE (LIMIT)<br>[FX10] | Disk space limit value for each node<br>For the summary information of the bulk job or step job, "-" is output.<br>For jobs on the FX100 and PRIMERGY compute node, this has no meaning. | O | O |
| CPU TIME (LIMIT) | CPU time limit value<br>For the summary information of the bulk job or step job, "-" is output. | O | O |
| PROC CPU LIMIT | CPU use time for each process<br>For the summary information of the bulk job or step job, "-" is output. | O | O |
| PROC COREFILE LIMIT | Core file size limit value for each process<br>For the summary information of the bulk job or step job, "-" is output. | O | O |
| PROC CREATE PROCESS LIMIT | Limit value of number of processes<br>For the summary information of the bulk job or step job, "-" is output. | O | O |
| PROC DATA LIMIT | Data segment limit value for each process<br>For the summary information of the bulk job or step job, "-" is output. | O | O |
| PROC LOCKED MEMORY LIMIT | Lock memory size limit value for each process<br>For the summary information of the bulk job or step job, "-" is output. | O | O |
| PROC MESSAGE QUEUE LIMIT | POSIX message queue size limit value for each process<br>For the summary information of the bulk job or step job, "-" is output. | O | O |
| PROC OPEN FILES LIMIT | File descriptor limit value for each process<br>For the summary information of the step job, "-" is output. | O | O |
| PROC PENDING SIGNAL LIMIT | Limit value of number of signals for each process<br>For the summary information of the bulk job or step job, "-" is output. | O | O |
| PROC PERMFILE LIMIT | Limit value of number of process file sizes<br>For the summary information of the bulk job or step job, "-" is output. | O | O |
| PROC STACK LIMIT | Stack segment limit value for each process<br>For the summary information of the bulk job or step job, "-" is output. | O | O |
| PROC VIRTUAL MEMORY LIMIT | Virtual memory size limit value for each process<br>For the summary information of the bulk job or step job, "-" is output. | O | O |
| NODE NUM (ALLOC)<br>[FX100/FX10] | 1. For a node-exclusive job<br>Shape and number of assigned nodes<br>*N*: *X*x*Y*x*Z* (torus mode)<br>*N*: *X*x*Y*x*Z*:mesh (mesh mode)<br>*N*:noncont (non-contiguous mode)<br>*N* is number of nodes, and *X*, *Y*, and *Z* are coordinate values (Unusable nodes are included).<br>2. For a node-sharing job<br>Shape and number of nodes allocated<br>*N*: *X*x*Y*x*Z*<br>For the summary information of the step job or job on PRIMERGY compute nodes, "-" is output.<br>For the summary information of the bulk job, the total for sub jobs is output. | O | FH/FX |
| MEMORY SIZE (ALLOC) | Assigned physical memory amount for each node<br>Unit is in MiB ($2^{20}$ bytes).<br>Displayed down to first decimal place<br>Value is rounded up. | O | O |

| Item name | Description | pjstat -s/-S | pjsub -s/-S |
|---|---|---|---|
| | Value in parentheses is in units of bytes.<br>For unlimited, "unlimited" is displayed.<br>For the summary information of the bulk job or step job, "-" is output. | | |
| CPU NUM (ALLOC) | Total number of assigned CPU cores<br>Number of CPU cores that have actually been assigned<br>For the summary information of the bulk job or step job, the total for sub jobs is output. | O | O |
| ELAPSE TIME (USE) | Execution elapsed time "*DD HH*:*MM*:*SS* (*s*)"<br>Actual period of time from job start to job end.<br>If time is no more than 24 hours, *DD* (number of days) is omitted.<br>The time is rounded up to an integral number of seconds.<br>For the summary information of the bulk job or step job, the total for sub jobs is output.<br>The job elapsed time does not include the swap period. | O | O |
| NODE NUM (UNUSED) | 1. FX100 or FX10 compute node: For a node-exclusive job<br>Number of nodes that are allocated to the job but cannot be used<br>2. FX100 or FX10 compute node: For a node-sharing job<br>Number of virtual nodes that are allocated to the job but cannot be used<br>3. PRIMERGY compute node<br>Of the allocated nodes, the number of nodes that cannot be used<br>For the summary information of the bulk job or step job, the total for sub jobs is output. | O | FH/FX |
| NODE NUM (USE) | 1. FX100 or FX10 compute node: For a node-exclusive job<br>Number of nodes allocated to the job and actually used by the job<br>2. FX100 or FX10 compute node: For a node-sharing job<br>Number of virtual nodes allocated to the job and actually used by the job<br>3. PRIMERGY compute node<br>Of the allocated nodes, the actual number of nodes used by jobs<br>For the summary information of the bulk job or step job, the total for sub jobs is output.<br>For the summary information of the step job containing both a sub job that runs on an FX100 or FX10 compute node and a sub job that runs on a PRIMERGY compute node, this item is "-". | O | FH/FX |
| NODE ID (USE) | Node ID list of nodes that have been used<br>Displayed in hexadecimal notation (e.g.) 0xaaaaaaaa<br>All nodes that have been used are displayed with single-byte space character used as delimiter.<br>For the summary information of the bulk job or step job, "-" is output. | O | O |
| TOFU COORDINATE (USE) [FX100/FX10] | Tofu coordinate list of nodes that have been used<br>Displayed in format of (*X, Y, Z*)<br>All nodes that have been used are displayed with single-byte space character used as delimiter.<br>*X, Y,* and *Z* are coordinate values of Tofu coordinates.<br>Since Tofu includes multiple nodes, same coordinates are displayed more than once. | O | FH/FX |

| Item name | Description | pjstat -s/-S | pjsub -s/-S |
|---|---|---|---|
| | For the summary information of the bulk job or step job, and job on the PRIMERGY compute nodes, "-" is output. | | |
| MAX MEMORY SIZE (USE) | Maximum amount of physical memory used<br>Unit is in MiB ($2^{20}$ bytes).<br>Displayed down to first decimal place<br>Value is rounded up.<br>Value in parentheses is in units of bytes.<br>For the summary information of the bulk job or step job, the maximum amount in sub jobs. | O | O |
| CPU NUM (USE) | Total number of CPU cores that have been used<br>Of assigned CPU cores, number of CPU cores that have actually been used by job.<br>For the summary information of the bulk job or step job, the total for sub jobs is output. | O | O |
| VIRTUAL MEMORY SIZE (USE)<br>[PG] | Maximum amount of virtual memory that have been used<br>Unit is in MiB ($2^{20}$ bytes).<br>Displayed down to first decimal place<br>Value is rounded up.<br>Value in parentheses is in units of bytes.<br>For the summary information of the bulk job or step job, the total for sub jobs is output.<br>For the job on FX100 and FX10 compute nodes, "-" is output. | O | PG |
| USER CPU TIME (USE) | Total of user CPU time (unit: ms)<br>User CPU time spent for job execution<br>For the summary information of the bulk job or step job, the total for sub jobs is output. | O | O |
| SYSTEM CPU TIME (USE) | Total of system CPU time (unit: ms)<br>System CPU time spent for job execution<br>For the summary information of the bulk job or step job, the total for sub jobs is output. | O | O |
| CPU TIME (TOTAL) | Total of user CPU time and system CPU time<br>For the summary information of the bulk job or step job, the total for sub jobs is output. | O | O |
| DISK SIZE (USE)<br>[FX10] | Disk space use amount<br>For the summary information of the job not using staging, the bulk job, and the step job, and for jobs on FX100 and PRIMERGY compute nodes, "-" is output. | O | O |
| I/O SIZE | Number of IO transfer bytes<br>All IO for network and files<br>Unit is in MB ($10^6$ bytes).<br>Displayed down to first decimal place<br>Value is rounded up.<br>Value in parentheses is in units of bytes.<br>For the summary information of the bulk job or step job, the total for sub jobs is output. | O | O |
| FILE I/O SIZE | Number of IO transfer bytes of files<br>IO for files<br>Unit is in MB ($10^6$ bytes).<br>Displayed down to first decimal place<br>Value is rounded up.<br>Value in parentheses is in units of bytes. | O | O |

| Item name | Description | pjstat -s/-S | pjsub -s/-S |
|---|---|---|---|
| | For the summary information of the bulk job or step job, the total for sub jobs is output. | | |
| EXEC INST NUM<br>[FX10] | Number of executed instructions (See "Note" below)<br>For the summary information of the bulk job or step job, the total for sub jobs is output.<br>If the value exceeds upper limit (0xfffffffffffffffe), "MAX" is output.<br>For the job on the FX100 and PRIMERGY compute nodes, and for the interactive job, 0 is output. | O | O |
| EXEC SIMD NUM<br>[FX10] | Number of executed SIMD instructions (See "Note" below)<br>For the summary information of the bulk job or step job, the total for sub jobs is output.<br>If the value exceeds upper limit (0xfffffffffffffffe), "MAX" is output.<br>For the job on the FX100 and PRIMERGY compute nodes, and for the interactive job, 0 is output. | O | O |
| PRO START DATE | Prologue execution start time "*YYYY/MM/DD HH:MM:SS*"<br>For the summary information of the bulk job or step job, "-" is output. | O | O |
| PRO END DATE | Prologue execution end time "*YYYY/MM/DD HH:MM:SS*"<br>For the summary information of the bulk job or step job, "-" is output. | O | O |
| PRO EXIT CODE | Exit code of the prologue script<br>0: Normal end<br>1: Instruction to shift the job to the ERROR state<br>2: Instruction to execute the job again<br>3: Instruction to shift the job to the HOLD state<br>4: Instruction to delete the job<br>101: Forced termination of the prologue process due to cancellation or holding of the job<br>102: Execution failure of the prologue script<br>103: Forced termination of the prologue process due to OOM killer<br>104: Forced termination of the prologue process due to hardware failure<br>105: Forced termination of the prologue process due to node down<br>106: Forced termination of the prologue process due to excessive amount of memory<br>107: Forced termination of the prologue process due to CPU time limit violation<br>108: Forced termination of the prologue process due to elapsed time limit violation<br>109: Forced termination of the prologue process due to an current directory inaccessible.<br>255: Other errors<br>For the summary information of the bulk job or step job, "-" is output. | O | O |
| EPI START DATE | Epilogue execution start time "*YYYY/MM/DD HH:MM:SS*"<br>For the summary information of the bulk job or step job, "-" is output. | O | O |
| EPI END DATE | Epilogue execution end time "*YYYY/MM/DD HH:MM:SS*"<br>For the summary information of the bulk job or step job, "-" is output. | O | O |
| EPI EXIT CODE | Exit code of the epilogue script<br>0: Normal end<br>1: Instruction to shift the job to the ERROR state<br>2: Instruction to execute the job again<br>3: Instruction to shift the job to the HOLD<br>4: Instruction to delete the job<br>101: Forced termination of the epilogue process due to cancellation or holding of the job | O | O |

| Item name | Description | pjstat -s/-S | pjsub -s/-S |
|---|---|---|---|
| | 102: Execution failure of the epilogue script<br>103: Forced termination of the epilogue process due to OOM killer<br>104: Forced termination of the epilogue process due to hardware failure<br>105: Forced termination of the epilogue process due to node down<br>106: Forced termination of the epilogue process due to excessive amount of memory<br>107: Forced termination of the epilogue process due to CPU time limit violation<br>108: Forced termination of the epilogue process due to elapsed time limit violation<br>109: Forced termination of the epilogue process due to an current directory inaccessible.<br>255: Other errors<br>For the summary information of the bulk job or step job, "-" is output. | | |
| SWAPOUT DATE<br>[FX100] | Swap-out start time *YYYY*/*MM*/*DD hh*:*mm*:*ss*<br>*YYYY*: Year, *MM*: Month, *DD*: Day, *hh*: Hours, *mm*: Minutes, *ss*: Seconds<br>For a job that was not swapped, and a job on the FX10 or PRIMERGY compute nodes, "-" is output.<br>For the summary information of the bulk job or step job, "-" is output. | O* | O* |
| SWAPIN DATE<br>[FX100] | Swap-in start time *YYYY*/*MM*/*DD hh*:*mm*:*ss*<br>*YYYY*: Year, *MM*: Month, *DD*: Day, *hh*: Hours, *mm*: Minutes, *ss*: Seconds<br>The swap-in start scheduled time is shown in parentheses.<br>"-" is displayed in any of the following cases:<br>* No swap-in occurred.<br>* The swap-in start scheduled time for the job cannot be determined.<br>* The job ended abnormally after the swap-out and before swap-in because of a node failure etc.<br>* The job was on the FX10 or PRIMERGY compute nodes.<br>* It is the summary information of the bulk job or step job.<br>For a second or subsequent job swap, the previous value may be output until the SWAPIN DATE value is determined. | O* | O* |
| SWAP TIME<br>[FX100] | Cumulative job swap time *DD HH*:*MM*:*SS* (*s*)<br>*DD*: Days, *HH*: Hours, *MM*: Minutes, *SS*: Seconds, *s*: Total seconds of the accumulated time<br>*DD* is omitted if less than 24 hours.<br>For a job that was not swapped and a job on the FX10 or PRIMERGY compute nodes, "00:00:00 (0)" is output.<br>For the summary information of the bulk job or step job, "-" is output. | O* | O* |
| SWAPOUT NUM<br>[FX100] | Number of times that the job was swapped out<br>For a job that was not swapped and a job on the FX10 or PRIMERGY compute nodes, 0 is output.<br>For the summary information of the bulk job or step job, "-" is output. | O* | O* |
| HOST NAME | Host name on which the pjsub command executed.<br>For the summary information of the bulk job or step job, "-" is output. | - | O |
| NODE ID (RANK) | Node IDs of the allocated nodes, and the rank numbers corresponding to the nodes.<br>Display format is "*node_id*(*rank_number*)"<br>For the summary information of the bulk job or step job, "-" is output. | - | O |
| MPI STATIC PROC (REQUIRE) | Number of processes (static processes) generated at the time of MPI program startup | O | O |

| Item name | Description | pjstat -s/-S | pjsub -s/-S |
|---|---|---|---|
| | For the summary information of the bulk job or step job, the total for sub jobs is output. | | |
| MPI PROC (REQUIRE) | Upper limit of the number of processes generated by the MPI program<br>This value is the sum of the upper limits of the following: the number of processes (static processes) generated at the time of MPI program startup and the number of processes (dynamic processes) generated by the MPI program during execution.<br>For the summary information of the bulk job or step job, the total for sub jobs is output. | O | O |
| MPI STATIC PROC (ALLOC) | Number of processes (static processes) generated at the time of MPI program startup<br>For the summary information of the bulk job or step job, the total for sub jobs is output. | O | O |
| MPI PROC (ALLOC) | Upper limit of the number of processes generated by the MPI program<br>This value is the sum of the upper limits of the following: the number of processes (static processes) generated at the time of MPI program startup and the number of processes (dynamic processes) generated by the MPI program during execution.<br>For the summary information of the bulk job or step job, the total for sub jobs is output. | O | O |
| TOTAL JOB PROC<br>[FX100][PG] | Accumulation number of the job process<br>For running job, this item is 0 in the output of the pjstat command.<br>For the summary information of the bulk job or step job, the total for sub jobs is output.<br>For the job on the FX10 compute node, 0 is output. | O | O |
| MAX JOB PROC<br>[FX100][PG] | Maximum number of processes<br>For running job, this item is 0 in the output of the pjstat command.<br>For the summary information of the bulk job or step job, the total for sub jobs is output.<br>For the job on the FX10 compute node, 0 is output. | O | O |
| ASSISTANT CORE (USE)<br>[FX100] | Number of assistant cores used for the job<br>For the summary information of the bulk job or step job, the total for sub jobs is output.<br>For the job on the FX10 compute node, 0 is output.<br>For the job on the PRIMERGY compute nodes, "-" is output. | O | O |
| ASSISTANT CORE USER CPU TIME (USE)<br>[FX100] | Total user CPU time spent for assistant cores (msec)<br>For the summary information of the bulk job or step job, the total for sub jobs is output.<br>For the job on the FX10 compute node, 0 is output.<br>For the job on the PRIMERGY compute nodes, "-" is output. | O | O |
| ASSISTANT CORE SYSTEM CPU TIME (USE)<br>[FX100] | Total system CPU time spent for assistant cores (msec)<br>For the summary information of the bulk job or step job, the total for sub jobs is output.<br>For the job on the FX10 compute node, 0 is output.<br>For the job on the PRIMERGY compute nodes, "-" is output. | O | O |
| ASSISTANT CORE MAX USER CPU TIME (USE)<br>[FX100] | User CPU use time of a node where the sum of the user CPU use time and system CPU use time becomes the maximum (msec)<br>For the summary information of the bulk job or step job, the maximum amount in sub jobs.<br>For the job on the FX10 compute node, 0 is output.<br>For the job on the PRIMERGY compute nodes, "-" is output. | O | O |

| Item name | Description | pjstat -s/-S | pjsub -s/-S |
|---|---|---|---|
| ASSISTANT CORE MAX SYSTEM CPU TIME (USE)<br>[FX100] | System CPU use time of a node where the sum of user CPU use time and system CPU use time becomes the maximum (msec)<br>For the summary information of the bulk job or step job, the maximum amount in sub jobs.<br>For the job on the FX10 compute node, 0 is output.<br>For the job on the PRIMERGY compute nodes, "-" is output. | O | O |
| AFFECTED NODE ID | If a communication path ICC error (FX100/FX10) or node-down event occurred and affected the job result, the node ID of the relevant node is output.<br>If there is no node that affected the job result or in the case of the summary information of the bulk job or step job, "-" is output. | O | O |
| BACKFILL | Whether a backfill concerning job scheduling occurred<br>0: No backfill occurred.<br>1: A backfill occurred.<br>For the summary information of the bulk job or step job, "-" is output. | O | O |
| NUMA POLICY<br>[FX100][PG] | NUMA polocy<br>0: PACK<br>1: UNPACK<br>For the summary information of the step job, "-" is output.<br>For the job on the FX10 compute node, this has no meaning. | O | O |
| SPECIFIED JOB START DATE | Job start time specified with -at option of the pjsub command *YYYY/MM/DD HH:MM:SS*<br>For the job on the FX10 compute node, "-" is output.<br>When the job start time is not specified with the --at option of the pjsub command , "-" is output. | O | O |
| START BULKNO | Start bulk number<br>For the sub job of the bulk job, the bulk number is output.<br>For the normal job and the step job, "-" is output. | O | O |
| END BULKNO | End bulk number<br>For the sub job of the bulk job, the bulk number is output.<br>For the normal job and the step job, "-" is output. | O | O |

Table A.4 Job statistical information items per node or virtual node (output only with -S option specified)

| Item name | Description | pjstat -S | pjsub -S |
|---|---|---|---|
| VNODE ID<br>[PG] | Virtual node ID (Decimal notation) | O | PG |
| NODE ID | Node ID (Hexadecimal notation) | O | O |
| TOFU COORDINATE<br>[FX100/FX10] | Tofu coordinate list of nodes that have been used (*X, Y, Z*)<br>*X, Y* and *Z* are coordinate values of the Tofu coordinates. | O | FH/FX |
| NODE AVAILABLE | Node status<br>available: Node is usable.<br>unavailable: Node is not usable during execution<br>unable to allocate: Node is not usable when it is assigned. | O | FH/FX |
| ARCHI INFORMATION | Machine type<br>FH: FX100<br>FX: FX10<br>PG: PRIMERGY | O | FH/FX |
| MEMORY SIZE (LIMIT) | Memory amount limit value<br>Upper limit value of memory amount in node<br>Unit is in MiB ($2^{20}$ bytes). | O | O |

| Item name | Description | pjstat -S | pjsub -S |
|---|---|---|---|
| | Displayed down to first decimal place<br>Value is rounded up.<br>Value in parentheses is in units of bytes.<br>For unlimited, "unlimited" is displayed.<br>If it is not specified in command, <DEFAULT> is displayed at end of display. | | |
| CPU TIME (LIMIT) | CPU time limit value (second)<br>For unlimited, "unlimited" is displayed.<br>If it is not specified in command, <DEFAULT> is displayed at end of display. | O | O |
| DISK SIZE (LIMIT)<br>[FX10] | Disk space limit value<br>Unit is in MB.<br>Displayed down to first decimal place<br>Value is rounded up.<br>Value in parentheses is in units of bytes.<br>For unlimited, "unlimited" is displayed.<br>If it is not specified in command, <DEFAULT> is displayed at end of display.<br>For jobs on the FX100 and PRIMERGY compute node, this has no meaning. | O | O |
| MEMORY SIZE (ALLOC) | Assigned memory amount<br>Memory amount assigned to the job in node<br>Unit is in MiB ($2^{20}$ bytes)<br>Displayed down to first decimal place<br>Value is rounded up.<br>Value in parentheses is in units of bytes.<br>For unlimited, "unlimited" is displayed. | O | O |
| CPU NUM (ALLOC) | Number of assigned CPU cores<br>Number of CPU cores assigned to job in node | O | O |
| MAX MEMORY SIZE (USE) | Maximum amount of used memory<br>Maximum amount of memory that has been used by job in node<br>Unit is in MiB ($2^{20}$ bytes).<br>Displayed down to first decimal place<br>Value is rounded up.<br>Value in parentheses is in units of bytes. | O | O |
| VIRTUAL MEMORY SIZE (USE)<br>[PG] | Maximum amount of used virtual memory<br>Maximum amount of virtual memory that have been used by job in node<br>Unit is in MiB ($2^{20}$ bytes).<br>Displayed down to first decimal place<br>Value is rounded up.<br>Value in parentheses is in units of bytes.<br>For a job on FX100 and FX10 compute nodes, "-" is output. | O | PG |
| CPU NUM (USE) | Number of CPU cores that have been used<br>Of assigned CPU cores, number of CPU cores that have actually been used by job | O | O |
| USER CPU TIME (USE) | Total of used user CPU time (unit: ms)<br>User CPU time spent for job execution in node | O | O |
| SYSTEM CPU TIME (USE) | Total of used system CPU time (unit: ms)<br>System CPU time spent for job execution in node | O | O |
| CPU TIME (TOTAL) | Total CPU use time in node (unit: ms)<br>Total of used user CPU time and used system CPU time | O | O |

| Item name | Description | pjstat -S | pjsub -S |
|---|---|---|---|
| DISK SIZE (USE) | Disk space use amount<br>Unit is in MB.<br>Displayed down to first decimal place<br>Value is rounded up.<br>Value in parentheses is in units of bytes. | O | O |
| I/O SIZE | Number of transferred bytes<br>All IO for network and files<br>Unit is in MB ($10^6$ bytes).<br>Displayed down to first decimal place<br>Value is rounded up.<br>Value in parentheses is in units of bytes. | O | O |
| FILE I/O SIZE | Number of IO transfer bytes of files<br>IO for files<br>Unit is in MB ($10^6$ bytes).<br>Displayed down to first decimal place<br>Value is rounded up.<br>Value in parentheses is in units of bytes. | O | O |
| EXEC INST NUM<br>[FX10] | Number of executed instructions (See "Note" below)<br>For a job on the FX100 and PRIMERGY compute nodes, and for the interactive job, 0 is output. | O | O |
| EXEC SIMD NUM<br>[FX10] | Number of executed SIMD instructions (See "Note" below)<br>For a job on the FX100 and PRIMERGY compute nodes, and for the interactive job, 0 is output. | O | O |
| PROC NUM | Number of processes in progress | O | - |
| PERF COUNT 1 (*1)<br>[FX100/FX10] | FX100: Number of 1FLOPS instructions<br>FX10: Number of SIMD load/store instructions<br>For the job on the PRIMERGY compute node, "-" is output.<br>For the interactive job, 0 is output. | O | O |
| PERF COUNT 2 (*1)<br>[FX100/FX10] | FX100: Number of 2FLOPS instructions<br>FX10: Number of SIMD floating-point operation instructions<br>For the job on the PRIMERGY compute node, "-" is output.<br>For the interactive job, 0 is output. | O | O |
| PERF COUNT 3 (*1)<br>[FX100/FX10] | FX100: Number of 4FLOPS instructions<br>FX10: Number of SIMD FMA instructions (*2)<br>For the job on the PRIMERGY compute node, "-" is output.<br>For the interactive job, 0 is output. | O | O |
| PERF COUNT 4 (*1)<br>[FX100/FX10] | FX100: Number of 8FLOPS instructions<br>FX10: Number of zero cycles<br>For the job on the PRIMERGY compute node, "-" is output.<br>For the interactive job, 0 is output. | O | O |
| PERF COUNT 5 (*1)<br>[FX100/FX10] | Number of execution cycles<br>For the job on the PRIMERGY compute node, "-" is output.<br>For the interactive job, 0 is output. | O | O |
| PERF COUNT 6 (*1)<br>[FX100/FX10] | Number of execution instructions<br>For the job on the PRIMERGY compute node, "-" is output.<br>For the interactive job, 0 is output. | O | O |
| PERF COUNT 7 (*1)<br>[FX100/FX10] | FX100: Number of 16FLOPS instructions<br>FX10: Number of floating-point operation instructions<br>For the job on the PRIMERGY compute node, "-" is output.<br>For the interactive job, 0 is output. | O | O |

| Item name | Description | pjstat -S | pjsub -S |
|---|---|---|---|
| PERF COUNT 8 (*1)<br>[FX10] | FX10: Number of FMA instructions (*2)<br>For the job on the FX100 compute node and the interactive job, 0 is output.<br>For the job on the PRIMERGY compute node, "-" is output. | O | O |
| PERF COUNT 9 (*1)<br>[FX100/FX10] | Maximum number of execution cycles<br>For the job on the PRIMERGY compute node, "-" is output.<br>For the interactive job, 0 is output. | O | O |
| MPI STATIC PROC (ALLOC) | Number of processes (static processes) generated on the node concerned or virtual node concerned at the time of MPI program startup | O | O |
| MPI PROC (ALLOC) | Upper limit of the number of processes generated by the MPI program on the node concerned or virtual node concerned<br>This value is the sum of the upper limits of the following: the number of processes (static processes) generated at the time of MPI program startup and the number of processes (dynamic processes) generated by the MPI program during execution. | O | O |
| TOTAL JOB PROC<br>[FX100][PG] | Accumulation number of the job process on the node concerned or the virtual node concerned<br>For running job, this item is 0 in the output of the pjstat command.<br>For the summary information of the bulk job, the total for sub jobs is output.<br>For the job on the FX10 compute node, 0 is output. | O | O |
| MAX JOB PROC<br>[FX100][PG] | Maximum numbers of processes generated by the job on the node concerned or the virtual node concerned<br>For running job, this item is 0 in the output of the pjstat command.<br>For the job on the FX10 compute node, 0 is output. | O | O |
| ASSISTANT CORE (USE)<br>[FX100] | Number of assistant cores used for the job on the node concerned or the virtual node concerned<br>For the job on the FX10 compute node, 0 is output.<br>For the job on the PRIMERGY compute node, "-" is output. | O | O |
| ASSISTANT CORE USER CPU TIME (USE)<br>[FX100] | Total user CPU time spent for assistant cores on the node concerned or the virtual node concerned (msec)<br>For the job on the FX10 compute node, 0 is output.<br>For the job on the PRIMERGY compute node, "-" is output. | O | O |
| ASSISTANT CORE SYSTEM CPU TIME (USE)<br>[FX100] | Total system CPU time spent for assistant cores on the node concerned or the virtual node concerned (msec)<br>For the job on the FX10 compute node, 0 is output.<br>For the job on the PRIMERGY compute node, "-" is output. | O | O |
| ASSISTANT CORE MAX USER CPU TIME (USE)<br>[FX100] | Maximum user CPU time spent for an assistant core on the node concerned or the virtual node concerned (msec)<br>For the job on the FX10 compute node, 0 is output.<br>For the job on the PRIMERGY compute node, "-" is output. | O | O |
| ASSISTANT CORE MAX SYSTEM CPU TIME (USE)<br>[FX100] | Maximum system CPU time spent for an assistant core on the node concerned or the virtual node concerned (msec)<br>For the job on the FX10 compute node, 0 is output.<br>For the job on the PRIMERGY compute node, "-" is output. | O | O |
| CPU BITMAP (ALLOC) | Bitmap (hexadecimal) representation of the CPU ID of the CPU allocated<br>The bit corresponding to CPU ID allocated is set from the subordinate position. | O | O |

| Item name | Description | pjstat -S | pjsub -S |
|---|---|---|---|
| CPU BITMAP (USE) | Bitmap (hexadecimal) representation of the CPU ID of the CPU actually used<br>The bit corresponding to CPU ID used is set from the subordinate position. | O | O |
| RANK NO | Rank number of the job executed on the relevant node or virtual node<br>If multiple rank numbers exist, they are separated by a space. | O | O |

(*1) You can calculate job performance information from the items PERF COUNT *n*. For details, see "A.4 Calculating Job Performance Information."
(*2) FMA: Floating-point Multiply and Add

### Note

- In following cases, the value of the item "EXEC INST NUM", "EXEC SIMD NUM" and "PERF COUNT *n*" may be 0:
  - The profiler, runtime information output function or debugger of Technical Computing Language is used in a job.
  - The xospastop command is executed in a job (For the xospastop command, see "2.1.1.7 Creating a job that uses PAPI library or strace command [FX100/FX10].")

  The job statistical information EXEC INST NUM, EXEC SIMD NUM and PERF COUNT *n* are not summed up for the MPI processing system other than Technical Computing Language.
- The job statistical information of jobs executed with the FX100 compute node include not only the calculation of the processing with CPU cores allocated to jobs, but also the calculation of the MPI asynchronous communication processing processed by the assistant core.
  Therefore, with FX100 compute node jobs, item CPU NUM(ALLOC) and CPU NUM(USE) may increase to a size larger than item CPU NUM(REQUIRE). Also, item USER CPU TIME (USE), SYSTEM CPU TIME (USE), and CPU TIME(TOTAL) include the time of CPUs executed by the assistant core.

## A.3 Display for compatibility with previous versions

For the pjstat command display to maintain compatibility, the display format of Technical Computing Suite V1.0L20 is supported. For display in the format of the previous version, V1.0L20, by the pjstat command executed in V1.0L30, use the --compat option.

```
$ pjstat [--compat [ph | pg]] ...
```

You can specify the ph or pg argument for the --compat option. They have the following meaning.

Table A.5 --compat option and display format

| Option | Meaning |
|---|---|
| --compat | Displays output in the V1.0L20 format according to the model of the compute node executing the job.<br>However, if FX100 or FX10 and PRIMERGY compute nodes are mixed in the cluster, an error occurs. |
| --compat ph | Displays output in the V1.0L20 format as intended for FX10 compute nodes, so jobs on the FX100 and FX10 compute nodes are the display targets.<br>If specified for a cluster consisting of only PRIMERGY compute nodes, this option displays only the header because no job is a display target. |
| --compat pg | Displays output in the V1.0L20 format as intended for PRIMERGY compute nodes, so jobs on the PRIMERGY compute nodes are the display targets.<br>If specified for a cluster consisting of only FX100 or FX10 compute nodes, this option displays only the header because no job is a display target. |

The targets of display in the V1.0L20 format by the --compat option are the following content displayed by the pjstat command: summary, job statistical information from the -s or -S option, and data from the --data option.

With the --compat option specified, the command displays some items only for jobs running on FX100 or FX10 compute nodes, or on PRIMERGY compute nodes, as shown below.

Table A.6 Items whose output target changes when the --compat option is specified

| Item name | For FX100 or FX10 compute nodes | For PRIMERGY compute nodes |
|---|---|---|
| STGIN [FX10] | o | - |
| READY | o | - |
| STGOUT [FX10] | o | - |
| NODE_REQUIRE | o | - |
| NODE | - | o |
| VNODE | - | o |
| CORE | - | o |
| V_MEM | - | o |
| V_POL | - | o |
| E_POL | - | o |
| RANK | - | o |

o: Displayed, -: Not displayed

The following display examples show the command with the --compat option specified.

- Example of display in the V1.0L20 format for jobs on FX100 or FX10 compute nodes

```
$ pjstat --compat ph

  ACCEPT QUEUED  STGIN  READY RUNING RUNOUT STGOUT   HOLD  ERROR   TOTAL
  nnnnnn nnnnnn nnnnnn nnnnnn nnnnnn nnnnnn nnnnnn nnnnnn nnnnnn nnnnnnn
s nnnnnn nnnnnn nnnnnn nnnnnn nnnnnn nnnnnn nnnnnn nnnnnn nnnnnn nnnnnnn


JOB_ID     JOB_NAME   MD ST  USER     START_DATE      ELAPSE_LIM NODE_REQUIRE
XXXXXXXXXX XXXXXXXXXX XX XXX XXXXXXXX MM/DD hh:mm:ss  hhhh:mm:ss nnnnnn:XXxYYxZZ

$ pjstat -v --compat ph

  ACCEPT QUEUED  STGIN  READY RUNING RUNOUT STGOUT   HOLD  ERROR   TOTAL
  nnnnnn nnnnnn nnnnnn nnnnnn nnnnnn nnnnnn nnnnnn nnnnnn nnnnnn nnnnnnn
s nnnnnn nnnnnn nnnnnn nnnnnn nnnnnn nnnnnn nnnnnn nnnnnn nnnnnn nnnnnnn

JOB_ID     JOB_NAME   MD ST  USER     GROUP   START_DATE      ELAPSE_TIM ELAPSE_LIM (*)
NODE_REQUIRE    LST EC  PC  SN PRI ACCEPT         RSC_UNIT REASON
XXXXXXXXXX XXXXXXXXXX XX XXX XXXXXXXX XXXXXXX MM/DD hh:mm:ss< hhhh:mm:ss hhhh:mm:ss (*)
nnnnnn:XXxYYxZZ XXX XXX XXX XX XXX MM/DD hh:mm:ss XXXXXXXX XXXXXXXXXXXXXXXX
```

Remarks) In the above example, (*) indicates where a line feed was inserted because of space limitations. Actually, the command output appears on one line.

- Example of display in the V1.0L20 format for jobs on PRIMERGY compute nodes

```
$ pjstat --compat pg

  ACCEPT QUEUED   RUNING   RUNOUT     HOLD    ERROR    TOTAL
  nnnnnn nnnnnn nnnnnnnn nnnnnnnn nnnnnnnn nnnnnnnn nnnnnnnn
s nnnnnn nnnnnn nnnnnnnn nnnnnnnn nnnnnnnn nnnnnnnn nnnnnnnn


JOB_ID     JOB_NAME   MD ST  USER     START_DATE      ELAPSE_LIM NODE   (*)
VNODE  CORE V_MEM
```

```
XXXXXXXXXX XXXXXXXXXX XX XXX XXXXXXXX MM/DD hh:mm:ss  hhhh:mm:ss nnnnnn (*)
nnnnnn nnn  nnnnnnnnnnMiB

$ pjstat -v --compat pg

  ACCEPT QUEUED   RUNING   RUNOUT     HOLD    ERROR    TOTAL
  nnnnnn nnnnnn nnnnnnnn nnnnnnnn nnnnnnnn nnnnnnnn nnnnnnnn
s nnnnnn nnnnnn nnnnnnnn nnnnnnnn nnnnnnnn nnnnnnnn nnnnnnnn

JOB_ID     JOB_NAME   MD ST  USER     GROUP    START_DATE      ELAPSE_TIM ELAPSE_LIM NODE (*)
VNODE  CORE V_MEM         V_POL E_POL RANK    LST EC  PC  SN PRI ACCEPT (*)
RSC_UNIT REASON
XXXXXXXXXX XXXXXXXXXX XX XXX XXXXXXXX XXXXXXXX MM/DD hh:mm:ss< hhhh:mm:ss hhhh:mm:ss nnnnnn (*)
nnnnnn nnn  nnnnnnnnnnMiB XXXXX XXXXX XXXXXXX XXX XXX XXX XX XXX MM/DD hh:mm:ss (*)
XXXXXXXX XXXXXXXXXXXXXXXX
```

Remarks) In the above example, (*) indicates where a line feed was inserted because of space limitations. Actually, the command output appears on one line.

# A.4 Calculating Job Performance Information

You can calculate job-related performance information from the items PERF COUNT *n* in job statistical information as follows.
The meanings of the symbols in the table below are as follows:

```
SUM(n): Sum of the values of items output to job statistical information (Table A.4) for each node or
virtual node
MAX(n): Maximum value of the values of items output to job statistical information (Table A.4) for
each node or virtual node
freq: CPU operating frequency (Hz)
```

Table A.7 How to calculate job performance information

| item | FX100 | FX10 |
|---|---|---|
| Number of execution instructions | SUM(PERF COUNT 6) | SUM(PERF COUNT 6) |
| Instruction execution time | Calculation not possible | (SUM(PERF COUNT 5)-SUM(PERF COUNT 4)/ freq |
| Number of executed SIMD instructions | Calculation not possible | SUM(PERF COUNT 1)<br>+SUM(PERF COUNT 2)<br>+SUM(PERF COUNT 3) |
| Floating-point operation | SUM(PERF COUNT 1)<br>+2*SUM(PERF COUNT 2)<br>+4*SUM(PERF COUNT 3)<br>+8*SUM(PERF COUNT 4)<br>+16*SUM(PERF COUNT 7) | SUM(PERF COUNT 7)<br>+2*SUM(PERF COUNT 8)<br>+2*SUM(PERF COUNT 2)<br>+4*SUM(PERF COUNT 3) |
| MIPS | Number of execution instructions<br>/MAX(PERF COUNT 9)<br>*freq/1000000 | Number of execution instructions<br>/MAX(PERF COUNT 9)<br>*freq/1000000 |
| MFLOPS | Floating-point operation/MAX(PERF COUNT 9)*freq/1000000 | Floating-point operation/MAX(PERF COUNT 9)*freq/1000000 |
| MIPS peak performance ratio (%) | MIPS<br>/(number of available CPU cores *4*freq/1000000)<br>*100 | MIPS<br>/(number of available CPU cores*4*freq/1000000)<br>*100 |
| MFLOPS peak performance ratio (%) | MFLOPS/(number of available CPU cores *16*freq/1000000)<br>*100 | MFLOPS/(number of available CPU cores *8*freq/1000000)<br>*100 |

| item | FX100 | FX10 |
|---|---|---|
| SIMD instruction ratio (%) | Calculation not possible | Number of executed SIMD instructions<br>/Number of execution instructions<br>*100 |

Note

For the CPU operating frequency, ask the administrator.

# Appendix B Notification Message Related to Job Execution

If job execution does not end normally and if the user gave an instruction to be notified of job start and job end, the user receives a detailed notification by e-mail.

The following table lists the contents of the Reason line in e-mail notification. For details of the notification by e-mail, see "2.6.1 Referencing job execution results."

Table B.1 E-mail notification contents

| Message | Meaning | State of job | Action |
|---|---|---|---|
| Reason: Internal error: *code*. | Internal error occurred An internal code is output at *code*. | Transition to ERROR state | Contact the administrator. The administrator shall collect investigation data according to the "Administrator's Guide for Maintenance," and then contact a Fujitsu systems engineer (SE) with the collected data together with the output message. |
| Reason: Node down. | A node failure occurred while being executing the job or requesting the job. | Transition to ERROR state | Contact the administrator. The administrator shall collect investigation data according to the "Administrator's Guide for Maintenance," and then contact a Fujitsu systems engineer (SE) with the collected data together with the output message. |
| Reason: Unable to analyze pjm *code*. | An invalid code was received from the job resource management function or job scheduler function. | Transition to ERROR state | Contact the administrator. The administrator shall collect investigation data according to the "Administrator's Guide for Maintenance," and then contact a Fujitsu systems engineer (SE) with the collected data together with the output message. |
| Reason: Fail to write the jobinfo file.<br>*Details*<br>path:*pathname* | It failed in the output of statistical information.<br>A cause is output at *Details*.<br>The path could not write *pathname* is displayed. | Termination of job | Contact the administrator, if the message "internal error" is output at *Details*. The administrator shall collect investigation data according to the "Administrator's Guide for Maintenance," and then contact a Fujitsu systems engineer (SE) with the collected data together with the output message.<br>In other than the above, check the write permission of *pathname*.<br>Also, show the job ID to check with the administrator whether statistical information on the job can be obtained from the information recorded in the system.<br>If the pmdumpjobinfo command is able to obtain statistical information on the job, the administrator should consider providing information to the user. |
| Reason: Fail to write the jobinfo record.<br>*Details*<br>path:*pathname* | | | |
| Reason: Gate check. | The job was canceled because of a gate check set by the administrator. | Delete | No action is necessary. |

| Message | Meaning | State of job | Action |
|---|---|---|---|
| Fail to get user name or group name. | Information on the group or user was not able to be acquired. | Transition to ERROR state | The existence of the group or user is confirmed. |
| Host file is not correct. | The rank-map-hostfile parameter of the pjsub command does not specify the correct host file. | Transition to ERROR state | Confirm the REASON with execute the pjstat command -v option.<br>If output in the REASON is the INVALID HOSTFILE, review the host file.<br>In other case, contact the administrator. The administrator shall collect investigation data according to the "Administrator's Guide for Maintenance," and then contact a Fujitsu systems engineer (SE) with the collected data together with the output message. |
| Fail to access current directory. | When the job submitted, the current directory was inaccessible. | Transition to ERROR state | The existence of the current directory is confirmed. |
| The current directory does not exist or have permission. path: *pathname*. | At job submission, the current directory or the standard input/ standard output/standard error output file was inaccessible.<br>The *pathname* is a path name of the current directory. | Termination of job | Confirm the existence of the current directory at job submission and, the path name of the standard input/standard output/ standard error output file.<br>Or, confirm whether the user submitting the job has access privileges for them. |
| Job *jobid* was canceled due to the swap processing failure of the system. | Swap-out processing or swap-in processing failed and the job was aborted. | Transition to QUEUED or termination | Since the aborted job is automatically re-executed, the user does not need to take any action. However, for jobs that automatically re-execution is disabled, the job must be resubmitted. |
| Job *jobid* canceled by pjdel command from *username*. | The pjdel command executed by the user *username* deleted the job. | Delete | No action is necessary. |
| Job *jobid* canceled, because of norestart job. | The job is not re-executed because automatically re-execution is disabled. | Delete | To execute the job, resubmit it. |
| The job was canceled because of the deadline period. | The job was interrupted according to the deadline schedule. | Transition to QUEUED state | If there are free resources, the job is re-executed. If not, the job will be re-executed after the deadline schedule ends. |
| CPU time limit exceeded. | The CPU utilization time exceeded the limit value. | Termination of job | Check the resource limit value for the job. |
| Elapse time limit exceeded. | The elapsed time exceeded the limit value. | Termination of job | Check the resource limit value for the job. |
| Memory size limit exceeded. | The memory usage exceeded the limit value. | Termination of job | Check the resource limit value for the job. |
| Killed by OOM killer. | OOM Killer in the OS forcibly terminated the job. | Termination of job | Contact the administrator. The administrator shall collect investigation data according to the "Administrator's Guide for Maintenance," and then contact a Fujitsu systems engineer (SE) with the collected data together with the output message. |

| Message | Meaning | State of job | Action |
|---|---|---|---|
| Job *jobid* was deleted by system factor. | The job was deleted due to a system factor. | Delete | Contact the administrator. The administrator shall collect investigation data according to the "Administrator's Guide for Maintenance," and then contact a Fujitsu systems engineer (SE) with the collected data together with the output message. |
| Abnormal end. | An abnormality occurred in job execution. | Termination of job | Contact the administrator. The administrator shall collect investigation data according to the "Administrator's Guide for Maintenance," and then contact a Fujitsu systems engineer (SE) with the collected data together with the output message. |
| Staging errors occurred. Refer to the staging information file. *Details* | An error occurred in staging processing. | - | Check the contents of the staging information file indicated in Details. |
| Job *jobid* is started. | The job started. | - | No action is necessary.<br>This notification message is output only if the -m b option is specified in the pjsub command. |
| Job *jobid* is completed. | The job ended normally. | - | No action is necessary.<br>This notification message is output only if the -m e option is specified in the pjsub command. |
| Job *jobid* restarted. | The job was re-executed. | - | No action is necessary.<br>This notification message is output only if the -m r option is specified in the pjsub command. |

# Appendix C Executing programs of MPI processing system other than Technical Computing Language

This appendix describes examples of executing MPI programs of the MPI processing system other than Technical Computing Language on the job operation software and the notes on executing them. These MPI programs include Intel MPI, LAM/MPI, MPICH, or PlatformMPI.

### [NOTE]

- To execute an MPI process on a node or virtual node, use the pjrsh command which is supplied with the job operation software instead of rsh and ssh command.
  However, the pjrsh command does not transfer the standard input to the remote process. Therefore, the functions (for example, the standard input transfer function in Intel MPI (Hydra)) that use the standard input for the pjrsh command do not operate normally.
  When it is necessary to provide an input to MPI processes, which are remote processes, instead of using the standard input of the mpiexec process, ensure that the files are read from each MPI process.

- To allocate the MPI process for each virtual node, specify respectively the -L "vnode" option and the -P "vn-policy" option of the pjsub command when submitting a job, as shown below. [PG]

  Table C.1 Options of the pjsub command for MPI processing system other than Technical Computing Language [PG]

| Options of the pjsub command | Meaning |
|---|---|
| -L "vnode=*N*" | Specifies the number of execution nodes *N* to vnode, which is the number of virtual nodes allocated to the node. |
| -P "vn-policy=abs-unpack" | Specifies abs-unpack as vn-policy, which is a virtual node allocation policy for a job. |

- The job statistical information items EXEC INST NUM and EXEC SIMD NUM acquired with FX10 nodes do not include execution results for MPI processing system other than Technical Computing Language

- To execute an MPI program by specifying the host name, operate as follows.

  a. Executing on FX100 or FX10 compute nodes
     Use the pjshowip command.
     The pjshowip command is executed in a job script and outputs the allocated node list (IP addresses of each rank number) to the standard output.

  b. Executing on PRIMERGY compute nodes
     Use the environment variable PJM_O_NODEINF.
     An environment variable PJM_O_NODEINF is set by the job operation software. It indicates the path of the file storing the node list (a list of IP addresses) which is a list of the nodes allocated to the job.

  For details, refer to the following example.

- To delete an MPI job, send a signal of the MPI processing system that executes the cleanup process (such as resource releases), and then delete the job. If a job is deleted without sending the signal for executing the cleanup process, it may have some impact on operations. For example, the subsequent job may not be executed.

  For signals that execute the cleanup process, check the specifications of MPI processing systems. Generally, the signal SIGINT or SIGTERM executes the cleanup process.

### [Examples of execution on each MPI processing system]

The examples of execution on each MPI processing system are shown below.
For details of the way of executing the MPI programs and the version dependency of MPI, confirm the specifications of each MPI processing system.

- Intel MPI 4.1.3 [PG]

Submitting the job script that executes the following a series of operations enables execution of MPI programs of Intel MPI on the virtual node of the job.

**[Using the Scalable Process Management System (Hydra)]**

1. Setting environment variables

   Set Hydra environment variables as follows.

   a. I_MPI_HYDRA_BOOTSTRAP=rsh

   b. I_MPI_HYDRA_BOOTSTRAP_EXEC=/bin/pjrsh

   c. I_MPI_HYDRA_HOST_FILE="${PJM_O_NODEINF}"

   Henceforth, the pjrsh command is used instead of rsh and ssh commands.

2. MPI program execution

   Execute the MPI program that was created with Intel MPI. To execute the program, use the mpiexec.hydra command.

[Example] Job script that executes a process parallel MPI program 'a.out' (when using Hydra)

```
...
export I_MPI_PERHOST=1
export I_MPI_HYDRA_BOOTSTRAP=rsh                        1.a.
export I_MPI_HYDRA_BOOTSTRAP_EXEC=/bin/pjrsh            1.b.
export I_MPI_HYDRA_HOST_FILE="${PJM_O_NODEINF}"         1.c.
mpiexec.hydra -n 4 a.out                                2.
...
```

Note

MPI programs of Intel MPI cannot be executed on the virtual node of the job using Multipurpose Daemon (MPD).

- LAM/MPI 7.1.4 [PG]

Submitting the job script that executes the following series of operations enables execution of MPI programs of LAM/MPI on the virtual node of the job.

1. Starting LAM runtime environment

   a. Set a unique value in the node to the environment valuable LAM_MPI_SESSION_SUFFIX of LAM/MPI.
   As a unique value, a job ID (environment variable PJM_JOBID) is available.
   By using this, a user can start more than one LAM runtime environment in a node.

   b. Set /bin/pjrsh to the environment variable LAMRSH of LAM/MPI. By setting it, pjrsh command is used instead of rsh and ssh command.

   c. As a host name list, specify the node list file ${PJM_O_NODEINF} to start the LAM runtime environment (lamboot).

2. Executing an MPI program
   Execute an MPI program created on LAM/MPI. There is no special operation required.

3. Stopping LAM runtime environment
   Stop the LAM runtime environment that has been started earlier (lamhalt). There is no special operation required.

[Example] Job script that executes a process parallel MPI program (a.out)

```
...
export LAM_MPI_SESSION_SUFFIX="${PJM_JOBID}"        1.a.
export LAMRSH=/bin/pjrsh                            1.b.
lamboot "${PJM_O_NODEINF}"                          1.c.
mpiexec -n 4 a.out                                  2.
lamhalt                                             3.
...
```

- MPICH2 1.5 [PG]

Submitting the job script that executes the following a series of operations enables execution of MPI programs of MPICH2 on the virtual node of the job.

1. Add the following options to start MPD daemon (mpdboot).

```
-f "${PJM_O_NODEINF}" -r /bin/pjrsh
```

   The -f option specifies the node list file ${PJM_O_NODEINF} that was allocated to the job.
   The -r option specifies to use pjrsh command instead of rsh and ssh command.

2. Executing an MPI program
   Execute an MPI program created on MPICH2. There is no special operation required.

3. Ending MPD daemon
   End the MPD daemon that has started earlier (mpdallexit). There is no special operation required.

[Example] Job script that executes a process parallel MPI program (a.out)

```
...
mpdboot -n 4 -f "${PJM_O_NODEINF}" -r /bin/pjrsh      1.
mpiexec.mpd -n 4 a.out                                2.
mpdallexit                                            3.
...
```

**[Using the Scalable Process Management System (Hydra)]**

1. Setting environment variables
   Set Hydra environment variables as follows.

   a. HYDRA_BOOTSTRAP=rsh

   b. HYDRA_BOOTSTRAP_EXEC=/bin/pjrsh

   c. HYDRA_HOST_FILE="${PJM_O_NODEINF}"

   Henceforth, the pjrsh command is used instead of rsh and ssh commands.

2. MPI program execution
   Execute the MPI program that was created with MPICH2. To execute the program, use the mpiexec.hydra command.

[Example] Job script that executes a process parallel MPI program (a.out) (when using Hydra)

```
...
export HYDRA_BOOTSTRAP=rsh                              1.a.
export HYDRA_BOOTSTRAP_EXEC=/bin/pjrsh                  1.b.
export HYDRA_HOST_FILE="${PJM_O_NODEINF}"               1.c.
mpiexec.hydra -n 4 a.out                                2.
...
```

- OpenMPI 1.8

As shown below, MPI program of OpenMPI can be executed on a node or virtual node of the job.

1. Add the following to the installation destination of OpenMPI, etc/openmpi-mca-params.conf file.

```
orte_rsh_agent=/bin/pjrsh
```

2. Executing an MPI program

   a. Executing on FX100 or FX10 compute nodes
      Specify the node list file which was output by the pjshowip command with the -machinefile option of mpiexec in the job script.
      [Example] Job script that executes a process parallel MPI program (a.out)

```
xospastop
pjshowip > ./iplist_"${PJM_JOBID}"
```

```
mpiexec -n 4 -machinefile ./iplist_"${PJM_JOBID}" a.out
rm -f ./iplist_"${PJM_JOBID}"
```

b. Executing on PRIMERGY compute node
Specify the node list file ${PJM_O_NODEINF} with the -machinefile option of mpiexec in the job script.
[Example] Job script that executes a process parallel MPI program (a.out)

```
mpiexec -n 4 -machinefile "${PJM_O_NODEINF}" a.out
```

- Platform MPI 9.1.2 [PG]

Submitting the job script that executes the following series of operations enables execution of MPI programs of Platform MPI on the virtual node of the job.

1. Set /bin/pjrsh to the environment variable MPI_REMSH of Platform MPI. By setting it, pjrsh command is used instead of rsh and ssh command.
2. Executing an MPI program

   Execute an MPI program created on Platform MPI. Then, specify the node list file ${PJM_O_NODEINF} with the -hostfile option.

[Example] Job script that executes a process parallel MPI program (a.out)

```
...
export MPI_REMSH=/bin/pjrsh                          1.
mpirun -np 4 -hostfile "${PJM_O_NODEINF}" a.out      2.
...
```

# Appendix D Notes on Systems That Use Emergency Jobs [FX100]

This appendix describes the effects on user programs when emergency jobs are submitted to the FX100 compute node and notes to observe when creating a user program.

## D.1 User program to be executed as an emergency job

It is not necessary to build special processing into a user program in order to execute that program as an emergency job.

## D.2 User program that is stopped to allow emergency job execution

### D.2.1 Re-creating a user program

A user program that is stopped to allow the execution of an emergency job may need to be re-created (re-linked) to enable job swap. If this operation is not taken, the user program cannot use job swap and the job in which the program is executing may be aborted.

The following user programs need to be re-created:

a. A non-MPI program written in C or C++ languages,

b. which is created without specifying the -Kopenmp and -Kparallel compile-time options, and

c. in which the environment variable LD_PRELOAD is set by overwriting before the execution of the user program.

If a job that is being executed is not swapped but instead aborted upon the execution of an emergency job, check whether the above conditions apply to the user program executed in that job. If they apply, re-create the user program by specifying the following link option for both the cross compiler and the own compiler.

```
-lfjpnckpt -L/opt/FJSVXosCkpt/lib
```

**Information**

You can set the above link option for the compile-time profile or to an environment variable specific to each compile-time compiler (fcc_ENV, FCC_ENV for the own compiler; fccpx_ENV, FCCpx_ENV for the cross-compiler).
For details, see the manuals of Technical Computing Language below.

- "C Language User's Guide (for PRIMEHPC FX100)"
- "C++ Language User's Guide (for PRIMEHPC FX100)"

### D.2.2 Impacts of job swaps on user programs

When a job being executed is stopped to execute an emergency job, the method of doing this varies depending on the job type.

Table D.1 Type of running job and stop method

| Type of running job | Stop method |
|---|---|
| Normal job | Job swap or abort |
| Step job | Job swap or abort |
| Bulk job | Abort |
| Interactive job | Abort |
| Emergency job | Not stopped<br>(Wait until the running emergency job is completed) |

If the node allocation method for jobs is non-contiguous mode, the running job is not stopped to execute an emergency job, as follows.

Table D.2 Node allocation method and stop method

| Node allocation method | Stop method |
|---|---|
| torus mode | Job swap or abort |
| mesh mode | Job swap or abort |
| non-contiguous mode | Not stopped<br>(Wait until the running the job is completed) |

If the method used to stop a running job is "job swap or abort" in the above table, the method to use is determined from the job state, as follows.

Table D.3 State of a running job and corresponding stop method

| Job state | Stop method |
|---|---|
| RUNNING-A | Abort |
| RUNNING-P | Abort |
| RUNNING | Job swap |
| RUNNING-E | Abort |
| RUNOUT | Job swap |

## Note

- A job to be stopped is automatically selected according to the policy set by the administrator. There is also the case where the administrator stops a specific job.
- If a job swap failed and the job cannot be restored to its original state, the job is aborted. Even when a job can be restored to its original state, the job is aborted if it is selected again to be stopped.
  Note that, if a job swap specified by the administrator using the pmswapout or pmswapin command failed, the job is aborted only when it cannot be restored to its original state.
- Job swap operation may involve some IO load so as to save/load the contents of the memory to/from the global file system. When that operation for a job couldn't complete within 60 minutes, for example, because of the system being under heavy IO load, the job is aborted.
- Suppose the submission of an emergency job causes a job to be swapped out. Then, if another emergency job is submitted, the swapped-out job may be aborted so that resources can be allocated to this emergency job.
- You can submit only a node-exclusive job as an emergency job. Therefore, if multiple node-sharing jobs exist on a node where an emergency job is placed, all of these node-sharing jobs are to be stopped.

Also, when job swap is generated, the impacts on the user program are as follows.

- Impact on user programs using IPC (Inter Process Communication)
  If a job using the following system calls is swapped for emergency job execution, the system calls may return with errno=EINTR.
  - System V Message Queue : msgrcv(2), msgsnd(2)
  - System V Semaphore : semop(2)

If these system calls returned with errno=EINTR, take measures to make the system calls again if necessary.

- Impact on user programs that handle the time difference
  If a program that calculates the elapsed execution time from the time difference is stopped by job swap, note that the resultant elapsed time is not that period in which the user program actually ran. Depending on the user program implementation, the program may not produce the expected results.
- Impact on user programs using timers
  The Linux timer that is set by the alarm(2) function etc. measures time even during the swap period. For this reason, the timer may

have already expired when swap-in of the job is completed.
Also, the sleep(3) function returns by the occurrence of a job swap even before the specified time elapses.

- Impact on processing that links with processes outside the job
  The results of processing that links with processes outside the job are not guaranteed.

- Impact on user programs that use the vfork(2) system call
  If a child process is created by the system call vfork(2) in a job and the job becomes a target to be swapped before the child process invokes an exec system call (execve(2) or _exit(2)), the job may be aborted.
  Therefore, invoke an exec system call immediately after the generation of a child process by the vfork(2) system call.

- Impact on programs that use a file lock
  If a program running in a job uses a file lock at the system call fcntl(2), the job may be aborted when all of the following conditions are satisfied.

  - The target of the file lock is a file on FEFS.
  - The system call fcntl(2) is blocked for 250 seconds or longer to wait for acquisition and release of the file lock.
  - The job is targeted for a job swap.

- Impacts on user programs using the Technical Computing Language
  If a user program created using the Technical Computing Language is swapped to execute an emergency job, the job has the following impact.

  - Impact on MPI programs

    - The MPI statistical information measured by specifying the mpi_print_stats MCA parameter is not correct.
    - When the mpi_deadlock_timeout MCA parameter is specified to detect a deadlock state, and if a job swap occurs during the wait for communication, the wait time measured up to that point is reset at the swap-in time and measurement restarts.
    - If the MPI program calls the following MPI functions even once, it cannot be swapped.

      | MPI_Open_port function |
      |---|
      | MPI_Comm_accept function |
      | MPI_Comm_connect function |
      | MPI_Comm_disconnect function |
      | MPI_Lookup_name function |
      | MPI_Publish_name function |
      | MPI_Unpublish_name function |

      If multiple mpiexec processes are running in parallel and the MPI programs started by those processes are linked with each other using the above group of functions, the programs cannot be stopped safely by job swap processing.

  - Impact on the runtime information output function
    The elapsed time obtained by the runtime information output function may produce the values not intended by the user.

  - Impact on the profiler
    The elapsed time and hardware monitor information obtained by the profiler may not be the values intended by the user.

  - Impact related to the debugger
    A job using an interactive debugger cannot be swapped out. Therefore, such a job may be aborted by a submission of an emergency job.

### Note

To see whether a user program was stopped by job swap, check the SWAP TIME and SWAPOUT NUM job statistical information items, which are output by the pjsub or pjstat command.
If these values are other than 0, the job was stopped by job swap at least once. This means that user programs executed in the job may

have been swapped out during execution.
Note that the output of these items may be disabled by a setting made by the administrator. In such a case, contact the administrator.

### See

- For details on the MCA parameters, see the "MPI User's Guide (PRIMEHPC FX100)," which is a Technical Computing Language manual.
- For details on how to use the MPI functions, see the "MPI User's Guide (PRIMEHPC FX100)," which is a Technical Computing Language manuals, as well as MPI specifications.
- For details on how to use the runtime information output function, see the "Runtime Information Output Function User's Guide (PRIMEHPC FX100)," which is a Technical Computing Language manual.
- For details on how to use the profiler, see the "Profiler User's Guide (PRIMEHPC FX100)," which is a Technical Computing Language manual.
- For details on how to use an interactive debugger, see the "Debugger User's Guide (PRIMEHPC FX100)," which is a Technical Computing Language manual.

# Appendix E Operations on Jobs

This appendix shows whether operations can be performed on jobs depending on the job state and job type.

The meanings of the symbols in the tables below are as follows.

```
O: Operation on the job is permitted.
X: Operation on the job is not permitted.
-: The job never enters that status.
```

The meanings of the cells that are split in two under the Bulk job or Step job column are as follows.

```
Top: Operation on the bulk job or step job (job ID specified as the operation target)
Bottom: Operation on a sub job of the bulk job or step job (sub job ID specified as the operation
target)
```

The cells that are not split in two refer commonly to both to the bulk jobs and step jobs and their sub jobs.

Table E.1 Deleting a job (pjdel)

| Job status | Normal job | Bulk job | Step job | Interactive job |
|---|---|---|---|---|
| ACCEPT | O | O | O | O |
| REJECT | X | X | X | X |
| QUEUED | O | O | O | O |
| STGIN | O | -<br>O | O | - |
| READY | O | -<br>O | O | - |
| RUNNING-A | O | -<br>O | -<br>O | O |
| RUNNING-P | X(*1) | -<br>X(*1) | X(*1) | X(*1) |
| RUNNING | O | O | O | O |
| RUNNING-E | X(*1) | -<br>X(*1) | X(*1) | X(*1) |
| RUNOUT | X(*2) | O<br>X(*2) | -<br>X(*2) | O |
| CANCEL | X | X | X | X |
| ERROR | O | O | O | - |
| STGOUT | X(*2) | -<br>X(*2) | X(*2) | - |
| EXIT | X | X | X | X |
| HOLD | O | O | O | - |
| SWAPOUT | O | - | O | - |
| SWAPPED | O | - | O | - |
| SWAPIN | O | - | O | - |

(*1) If you specify the --enforce option, the operation is possible. However, you need to have the privileges to specify the --enforce option.
(*2) With the --no-stgout option specified, the operation is possible.

Table E.2 Holding a job (pjhold)

| Job status | Normal job | Bulk job | Step job | Interactive job |
|---|---|---|---|---|
| ACCEPT | X | X | X | X |
| REJECT | X | X | X | X |
| QUEUED | O | O | O | X |
| STGIN | O | -<br>O | O | - |
| READY | O | -<br>O | O | - |
| RUNNING-A | O(*1) | -<br>O(*1) | -<br>O(*1) | X |
| RUNNING-P | X(*2) | -<br>X(*2) | X(*2) | X |
| RUNNING | O(*1) | O(*3)<br>O(*1) | O(*1) | X |
| RUNNING-E | X(*2) | -<br>X(*2) | X(*2) | X |
| RUNOUT | X | X | -<br>X | X |
| CANCEL | X | X | X | X |
| ERROR | X | X | X | - |
| STGOUT | X | -<br>X | X | - |
| EXIT | X | X | X | X |
| HOLD | X | X | X | - |
| SWAPOUT | X | - | X | - |
| SWAPPED | X | - | X | - |
| SWAPIN | X | - | X | - |

(*1) If you specify the --norestart option in the pjsub command when submitting the job or sub job, the operation is not possible.
(*2) If you specify the --enforce option, the operation is possible. However, you need to have the privileges to specify the --enforce option.
(*3) The target is sub jobs that can be held in bulk jobs. Sub jobs that cannot be held are ignored.

Table E.3 Release a held job (pjrls)

| Job status | Normal job | Bulk job | Step job | Interactive job |
|---|---|---|---|---|
| ACCEPT | X | X | X | X |
| REJECT | X | X | X | X |
| QUEUED | X | X | X | X |
| STGIN | X | -<br>X | X | - |
| READY | X | -<br>X | X | - |
| RUNNING-A | X | - | - | X |

| Job status | Normal job | Bulk job | Step job | Interactive job |
|---|---|---|---|---|
| | | X | X | |
| RUNNING-P | X | -<br>X | X | X |
| RUNNING | X | O(*)<br>X | X | X |
| RUNNING-E | X | -<br>X | X | X |
| RUNOUT | X | O(*)<br>X | X | X |
| CANCEL | X | X | X | X |
| ERROR | X | X | X | - |
| STGOUT | X | -<br>X | -<br>X | - |
| EXIT | X | X | X | X |
| HOLD | O | O | O | - |
| SWAPOUT | X | - | X | - |
| SWAPPED | X | - | X | - |
| SWAPIN | X | - | X | - |

(*)The target is sub jobs that can be released the holding status in bulk jobs. Sub jobs that cannot be released the holding status are ignored.

Table E.4 Waiting a job (pjwait)

| Job status | Normal job | Bulk job | Step job | Interactive job |
|---|---|---|---|---|
| ACCEPT | O | O | O | O(*) |
| REJECT | O | O | O | O(*) |
| QUEUED | O | O | O | O(*) |
| STGIN | O | -<br>O | O | - |
| READY | O | -<br>O | O | - |
| RUNNING-A | O | -<br>O | -<br>O | O(*) |
| RUNNING-P | O | -<br>O | O | O(*) |
| RUNNING | O | O | O | O(*) |
| RUNNING-E | O | -<br>O | O | O(*) |
| RUNOUT | O | O | -<br>O | O(*) |
| CANCEL | O | O | O | O(*) |
| ERROR | O | O | O | - |

| Job status | Normal job | Bulk job | Step job | Interactive job |
|---|---|---|---|---|
| STGOUT | O | -<br>O | O | - |
| EXIT | O | O | O | O(*) |
| HOLD | O | O | O | - |
| SWAPOUT | O | - | O | - |
| SWAPPED | O | - | O | - |
| SWAPIN | O | - | O | - |

(*) The pjwait command ends normally without waiting for an interactive job to end or displaying any information.

Table E.5 Sending a signal to a job (pjsig)

| Job status | Normal job | Bulk job | Step job | Interactive job |
|---|---|---|---|---|
| ACCEPT | X | X | X | X |
| REJECT | X | X | X | X |
| QUEUED | X | X | X | X |
| STGIN | X | -<br>X | X | - |
| READY | X | -<br>X | X | - |
| RUNNING-A | X | -<br>X | -<br>X | X |
| RUNNING-P | X | -<br>X | X | X |
| RUNNING | O | O | O | O |
| RUNNING-E | X | -<br>X | X | X |
| RUNOUT | X | X | -<br>X | X |
| CANCEL | X | X | X | X |
| ERROR | X | X | X | - |
| STGOUT | X | -<br>X | X | - |
| EXIT | X | X | X | X |
| HOLD | X | X | X | - |
| SWAPOUT | X | - | X | - |
| SWAPPED | X | - | X | - |
| SWAPIN | X | - | X | - |

Table E.6 Changing job parameter (pjalter)

| Job status | Normal job | Bulk job | Step job | Interactive job |
|---|---|---|---|---|
| ACCEPT | X | X | X | X |

| Job status | Normal job | Bulk job | Step job | Interactive job |
|---|---|---|---|---|
| REJECT | X | X | X | X |
| QUEUED | O | O | O(*) | X |
| | | X | | |
| STGIN | X | - | X | - |
| | | X | | |
| READY | X | - | X | - |
| | | X | | |
| RUNNING-A | X | - | - | X |
| | | X | X | |
| RUNNING-P | X | - | X | X |
| | | X | | |
| RUNNING | X | X | X | X |
| RUNNING-E | X | - | X | X |
| | | X | | |
| RUNOUT | X | X | - | X |
| | | | X | |
| CANCEL | X | X | X | X |
| ERROR | O | O | O(*) | - |
| | | X | | |
| STGOUT | X | - | X | - |
| | | X | | |
| EXIT | X | X | X | X |
| HOLD | O | O | O(*) | - |
| | | X | | |
| SWAPOUT | X | - | X | - |
| SWAPPED | X | - | X | - |
| SWAPIN | X | - | X | - |

(*) The resource unit name cannot be changed for any operation on a sub job.

Table E.7 Coping the files under the job execution directory (pjget)

| Job status | Normal job | Bulk job | Step job | Interactive job |
|---|---|---|---|---|
| ACCEPT | X | X | X | X |
| REJECT | X | X | X | X |
| QUEUED | X | X | X | X |
| STGIN | X | - | X | - |
| | | X | | |
| READY | X | - | X | - |
| | | X | | |
| RUNNING-A | X | - | - | X |
| | | X | X | |

| Job status | Normal job | Bulk job | Step job | Interactive job |
|---|---|---|---|---|
| RUNNING-P | X | -<br>X | X | X |
| RUNNING | O(*) | X<br>O(*) | X<br>O(*) | X |
| RUNNING-E | X | -<br>X | X | X |
| RUNOUT | X | X | -<br>X | X |
| CANCEL | X | X | X | X |
| ERROR | X | X | X | - |
| STGOUT | X | -<br>X | X | - |
| EXIT | X | X | X | X |
| HOLD | X | X | X | - |
| SWAPOUT | X | - | X | - |
| SWAPPED | X | - | X | - |
| SWAPIN | X | - | X | - |

(*) Operation is possible only if the staging function is used.

Table E.8 Displaying the file list of the job execution directory (pjlist)

| Job status | Normal job | Bulk job | Step job | Interactive job |
|---|---|---|---|---|
| ACCEPT | X | X | X | X |
| REJECT | X | X | X | X |
| QUEUED | X | X | X | X |
| STGIN | X | -<br>X | X | - |
| READY | X | -<br>X | X | - |
| RUNNING-A | X | -<br>X | -<br>X | X |
| RUNNING-P | X | -<br>X | X | X |
| RUNNING | O(*) | X<br>O(*) | X<br>O(*) | X |
| RUNNING-E | X | -<br>X | X | X |
| RUNOUT | X | X | -<br>X | X |
| CANCEL | X | X | X | X |
| ERROR | X | X | X | - |

| Job status | Normal job | Bulk job | Step job | Interactive job |
|---|---|---|---|---|
| STGOUT | X | -<br>X | X | - |
| EXIT | X | X | X | X |
| HOLD | X | X | X | - |
| SWAPOUT | X | - | X | - |
| SWAPPED | X | - | X | - |
| SWAPIN | X | - | X | - |

(*)Operation is possible only if the staging function is used.

Table E.9 Displaying a job script (pjcat -s)

| Job status | Normal job | Bulk job | Step job | Interactive job |
|---|---|---|---|---|
| ACCEPT | O | O | X<br>O | O(*) |
| REJECT | X | X | X | X |
| QUEUED | O | O | X<br>O | O(*) |
| STGIN | O | -<br>O | X<br>O | - |
| READY | O | -<br>O | X<br>O | - |
| RUNNING-A | O | -<br>O | -<br>O | O(*) |
| RUNNING-P | O | -<br>O | X<br>O | O(*) |
| RUNNING | O | O | X<br>O | O(*) |
| RUNNING-E | O | -<br>O | X<br>O | O(*) |
| RUNOUT | O | O | -<br>O | O(*) |
| CANCEL | X | X | X<br>O | X |
| ERROR | O | O | X<br>O | - |
| STGOUT | O | -<br>O | X<br>O | - |
| EXIT | X | X | X | X |
| HOLD | O | O | X<br>O | - |
| SWAPOUT | O | - | X | - |

| Job status | Normal job | Bulk job | Step job | Interactive job |
|---|---|---|---|---|
| | | | O | |
| SWAPPED | O | - | X | - |
| | | | O | |
| SWAPIN | O | - | X | - |
| | | | O | |

(*)If the input is not in a job script like for a terminal or here-document, you cannot specify the -s option.

Table E.10 Displaying the output of the job (pjcat -o, pjcat -e)

| Job status | Normal job | Bulk job | Step job | Interactive job |
|---|---|---|---|---|
| ACCEPT | X | X | X | X |
| REJECT | X | X | X | X |
| QUEUED | X | X | X | X |
| STGIN | X | - | X | - |
| | | X | | |
| READY | X | - | X | - |
| | | X | | |
| RUNNING-A | X | - | - | X |
| | | X | X | |
| RUNNING-P | X | - | X | X |
| | | X | | |
| RUNNING | O(*) | X | X | X |
| | | O(*) | O(*) | |
| RUNNING-E | X | - | X | X |
| | | X | | |
| RUNOUT | X | X | - | X |
| | | | X | |
| CANCEL | X | X | X | X |
| ERROR | X | X | X | - |
| STGOUT | X | - | X | - |
| | | X | | |
| EXIT | X | X | X | X |
| HOLD | X | X | X | - |
| SWAPOUT | X | - | X | - |
| SWAPPED | X | - | X | - |
| SWAPIN | X | - | X | - |

(*)Operation is possible only if the staging function is used.

# Appendix F References

This appendix describes how to refer the man pages of the commands, notification to specify the command option, and the command messages.

## F.1 About man pages

Refer the man pages of the command in the job operation software on the login node by man command.

```
$ man command
```

## F.2 About command options

Keep the following in mind when specifying command options.

1. Specify each option one by one.

   [EXAMPLE] The command *command* supporting -a, -b and -c for its option

   ```
   $ command -a -b -c
   ```

   A command such as the one in the following example is determined to have a format error and is abnormally terminated:

   $ command -abc

2. The method of specifying more than one argument differs depending on the argument of the command itself and the argument of the option.

   [COMMAND ARGUMENTS]

   If multiple arguments are specified, separate them with a space.

   ```
   $ command value1 value2 value3 ...
   ```

   As follows, if the option is not specified correctly ,determined to have a format error and is abnormally terminated:

   $ command value1,value2,value3 ...

   [OPTION ARGUMENTS]

   When specifying more than one value, using a comma (,) to concatenate them.

   ```
   $ command -option value1,value2,value3,...
   ```

   As follows, if the option or value is not specified correctly determined to have a format error and is abnormally terminated:

   [EXAMPLE 1] Option -A is specified twice.

   $ command -A 1000 -A 3000

   [EXAMPLE 2] Two X arguments are specified for option -A.

   $ command -A X=1000,X=3000

## F.3 About command messages

Command messages are output in the following format.

```
[PRIORITY] component-name message-ID command-name message-text
```

- priority: Shows the level of importance of the message.
- component-name: Name indicating the function of job operation software
- message-ID: ID that is a message identifier

- command-name: Name of the command that displayed the message
- message-text: Displays information or error details.

The detail of the command messages priority is following.

Table F.1 Command message priority

| Priority | Meaning |
|---|---|
| INFO | Information message:<br>The message displays process-related information. |
| NOTE | Notification message:<br>The message reports the start or stop of a process. |
| WARN | Warning message:<br>A process encountered a problem but has continued. |
| ERR. | Error message:<br>A process encountered a problem and did not continue.<br>Note: The notation of priority has "ERR" followed by a dot (.). |

### Information

As shown in the following examples, character string "Parallelnavi" may be contained in the command messages. The string means that the message is an output of Job Operation Software.

```
$ pjshowrsc --help
pjshowrsc (Parallelnavi) ...
...
```

The flowing describes the meanings and actions for the messages.

# F.4 Messages of the command for referring to job information

## F.4.1 pjstat command

When a problem occurs during the execution of the pjstat command, the following messages are output to the standard error output:

#### Error message

**[ERR.] PJM 0201 pjstat Unknown option *opt*.**
**Try `pjstat --help` for more information.**

Meaning

The specified *opt* is an unrecognizable option.

Action

Specify the correct option.

**[ERR.] PJM 0202 pjstat Combination of option is illegal (*opt*).**
**Try `pjstat --help` for more information.**

Meaning

The combination of specified options *opt* is invalid.

Action

Specify the correct combination of options.

### [ERR.] PJM 0203 pjstat Unknown option argument *arg*. Try `pjstat --help` for more information.

#### Meaning

The specified *arg* is an unrecognizable argument of the option, or no argument is specified to the option.

#### Action

Specify the correct argument.

### [ERR.] PJM 0204 pjstat Argument format error *arg*.

#### Meaning

The setting value of the argument *arg* in the option is incorrect.

#### Action

Specify the correct setting value.

### [ERR.] PJM 0211 pjstat Invalid jobid syntax *jobid*.

#### Meaning

The specification of the job ID *jobid* s incorrect.

#### Action

Specify the correct job ID.

### [ERR.] PJM 0270 pjstat No execute.

#### Meaning

The user is not permitted to execute the command.

#### Action

Display the job ACL settings with the pjacl command, and check whether the pjstat command can be executed (execute pjstat). If necessary, ask the administrator to add permission.

### [ERR.] PJM 0290 pjstat PJM daemon is not present.

#### Meaning

The job manager function is not working, or communication with the job manager function is disabled.

#### Action

The system might not have been able to accept the request temporarily. Retry the command execution. Contact the administrator when the situation does not change in about ten minutes. The administrator is requested to check the operating state of the job manager function and the system state.

### [ERR.] PJM 0291 pjstat Internal error: *details*.

#### Meaning

An internal error occurred.

*details*: Detailed information on maintenance

#### Action

The system might not have been able to accept the request temporarily. Retry the command execution. Contact the administrator when the situation does not change in about ten minutes. The administrator shall collect investigation data according to the "Administrator's Guide for Maintenance," and then contact a Fujitsu systems engineer (SE) with the collected data together with the output message.

### [ERR.] PJM 0293 pjstat This command can be executed on SMM/L/JCM.

Meaning

The command was executed on a node other than a control, login, or job management node.

Action

Execute the command on a control, login, or job management node.

**[ERR.] PJM 0295 pjstat No cluster specified on SMM.**

Meaning

No cluster name is specified. Execution on a control node requires a cluster name.

Action

Specify a cluster name in the -c option of the pjstat command, or set a cluster name in the environment variable PXMYCLST.

**[ERR.] PJM 0297 The resource unit specified (Or, default) does not exist: *rscuname*.**

Meaning

The specified resource unit *rscuname* or the default resource unit *rscuname* defined by the job ACL function does not exist.

Action

Specify the resource unit again correctly. Alternatively, check whether the default resource unit exists.

### Information Message

**[INFO] PJM 0281 pjstat Data loading...**

Meaning

Display preparation is in progress. Please wait.

Action

This message appears when display does not start within three seconds. One dot (.) appears every three seconds.

## F.4.2 pjacl command

If an error occurs during the execution of the pjacl command, one of the following messages is output to the standard error output.

### Error message

**[ERR.] PJM 3201 pjacl Unknown option *opt*.**
**Try `pjacl --help` for more information.**

Meaning

The specified option *opt* is unknown.

Action

Specify the correct option.

**[ERR.] PJM 3203 pjacl Unknown option argument *arg*.**
**Try `pjacl --help` for more information.**

Meaning

The specified *arg* is an unrecognizable argument of the option, or no argument is specified to the option.

Action

Specify the correct argument.

### [ERR.] PJM 3240 pjacl User(*uname*) does not exist.

#### Meaning

The specified user name *uname* does not exist.

#### Action

Confirm that the user name *uname* is registered in the system. Alternatively, check whether it can be retrieved from a name service.

### [ERR.] PJM 3241 pjacl Group(*gname*) does not exist.

#### Meaning

The specified group name *gname* does not exist.

#### Action

Confirm that the group name *gname* is registered in the system. Alternatively, check whether it can be retrieved from a name service.

### [ERR.] PJM 3270 pjacl No execute.

#### Meaning

The user is not permitted to execute the command.

#### Action

Confirm the permission to execute the pjacl command by contacting the administrator.

### [ERR.] PJM 3271 pjacl No permit: *target*.

#### Meaning

The user is not permitted to display the specified user or group.

*target*: Indicates the user or group for which there is no permission as follows.

user(*uname*): The user does not have permission for the specified user *uname*.

group(*gname*): The user does not have permission for the specified group *gname*.

#### Action

Display the job ACL function settings with the pjacl command. Then, confirm the permission (permit pjacl) for the displayed objects of the pjacl command for the specified user or group in the range of the target resource unit or resource group. If necessary, ask the administrator to add permission.

### [ERR.] PJM 3291 pjacl Internal error: *details*.

#### Meaning

An internal error occurred.

*details*: Detailed information for maintenance

#### Action

The system might not have been able to accept the request temporarily. Retry the command execution. Contact the administrator when the situation does not change in about ten minutes. The administrator is requested to collect investigation data according to the "Administrator's Guide for Maintenance," and then contact a Fujitsu systems engineer (SE) with the collected data together with the output message.

### [ERR.] PJM 3293 pjacl This command can be executed on L/JCM.

#### Meaning

The command was executed on a node other than a login or job management node.

### Action

Execute the command on a login or job management node.

#### [ERR.] PJM 3294 pjacl Communication failed to system management function.

### Meaning

The command cannot get information from the system management function.

### Action

Contact the administrator. The administrator should check the settings of the system management function for the node on which the command executed or the job management node. If the problem cannot be resolved, the administrator is requested to collect investigation data according to the "Administrator's Guide for Maintenance," and then contact a Fujitsu systems engineer (SE) with the collected data together with the output message.

#### [ERR.] PJM 3295 pjacl In the standby node, pjacl cannot be used.

### Meaning

The pjacl command cannot be used on the standby job management node.

### Action

Execute the pjacl command on the login node or active job management node.

## F.4.3 pjshowrsc command

When a problem occurs during the execution of the pjshowrsc command, the following messages are output to the standard error output:

### Error message

#### [ERR.] PRM 3201 pjshowrsc Duplicated option is specified: *opt*

### Meaning

The option opt is specified more than once.

*opt*: Option name

### Action

Specify the option correctly, and execute the command again.

#### [ERR.] PRM 3202 pjshowrsc Conflicting option is specified

### Meaning

An incompatible option is specified.

### Action

Specify the option correctly, and execute the command again.

#### [ERR.] PRM 3203 pjshowrsc Cluster name is not specified

### Meaning

No cluster name is specified.

### Action

Specify a cluster name, and execute the command again.

#### [ERR.] PRM 3204 pjshowrsc Invalid option: *opt*

### Meaning

The specified option is invalid.

*opt*: Option name

### Action

Review the option, and execute the command again.

---

## [ERR.] PRM 3205 pjshowrsc -v [level]: The level must be 0,1,3

### Meaning

The specified level is incorrect.

### Action

Specify the detail level correctly, and execute the command again.

---

## [ERR.] PRM 3206 pjshowrsc Specification of cluster name is wrong: *clstname*

### Meaning

The specified cluster name *clstname* is incorrect.

### Action

Specify the correct cluster name, and execute the command again.

---

## [ERR.] PRM 3207 pjshowrsc Specification of resource unit name is wrong: *rscuname*

### Meaning

The specified resource unit name *rscuname* is incorrect.

### Action

Specify the correct resource unit name, and execute the command again.

---

## [ERR.] PRM 3208 pjshowrsc The cluster other than compute cluster is specified: *clstname*

### Meaning

The specified cluster name *clstname* is not for a compute cluster.

### Action

Specify the cluster name of a compute cluster, and execute the command again.

---

## [ERR.] PRM 3209 pjshowrsc Specification ID (node group ID, boot group ID, or node ID) is wrong *id*

### Meaning

*id*, which is the specified node group ID, boot group ID, or node ID, is incorrect.

### Action

Specify the correct node group ID, boot group ID, or node ID, and execute the command again.

---

## [ERR.] PRM 3210 pjshowrsc Specification method or specified range is wrong: *range*

### Meaning

The method of specifying the range or the specified range *range* is incorrect.

### Action

Specify the correct range, and execute the command again.

### [ERR.] PRM 3211 pjshowrsc The same ID (node group ID, boot group ID, or node ID) is specified: *id*

#### Meaning

*id*, which is the specified node group ID, boot group ID, or node ID, is the same as that already specified.

#### Action

Specify the correct node group ID, boot group ID, or node ID, and execute the command again.

### [ERR.] PRM 3212 pjshowrsc Specified node group ID, boot group ID, or node ID doesn't exist: *id*

#### Meaning

*id*, which is the specified node group ID, boot group ID, or node ID, does not exist.

#### Action

Specify the correct node group ID, boot group ID, or node ID, and execute the command again.

### [ERR.] PRM 3213 pjshowrsc There is no resource in the resource unit: *rscuname*

#### Meaning

The specified resource unit name *rscuname* does not exist.

#### Action

Specify the correct resource unit name, and execute the command again.

### [ERR.] PRM 3214 pjshowrsc In this node, pjshowrsc cannot be used.

#### Meaning

The pjshowrsc command cannot be used on this node.

#### Action

Execute it on the active control node, job management node, or login node.

### [ERR.] PRM 3215 pjshowrsc In the standby node, pjshowrsc cannot be used.

#### Meaning

The pjshowrsc command cannot be used on a standby node.

#### Action

Execute it on the active control node, job management node, or login node.

### [ERR.] PRM 3216 pjshowrsc Permission denied ID: *id*

#### Meaning

The user does not have the privileges to display the state of the specified node (node ID: *id*).

#### Action

Confirm that the specified node ID is correct. If the specified node ID is correct, the user does not have the display privileges for that node ID. If necessary, contact the administrator to confirm the display privileges for the relevant node.

### [ERR.] PRM 3217 pjshowrsc Connection error: *clstname*

#### Meaning

Resource information display failed because communication was not possible with the active job management node in the cluster *clstname*.

### Action

Ask the cluster administrator to check the system state. The cluster administrator is requested to check the state of the job management node in the cluster *clstname* by using the pashowclst command. If the error occurs even though the job management node is in the normal state, collect investigation data according to the "Administrator's Guide for Maintenance." Then, contact a Fujitsu systems engineer (SE) with the collected data together with the output message.

## [ERR.] PRM 3218 pjshowrsc Job resource management function is not starting: *ret*

### Meaning

The job resource management function has not been started.

*ret*: Maintenance information

### Action

Wait a moment, and execute the command again.

If the same error recurs, contact the administrator. The administrator is requested to collect investigation data according to the "Administrator's Guide for Maintenance," and then contact a Fujitsu systems engineer (SE) with the collected data together with the output message.

## [ERR.] PRM 3219 pjshowrsc The same resource unit name is specified: *rscuname*

### Meaning

The same resource unit name is already specified.

### Action

Specify the correct resource unit name, and execute the command again.

## [ERR.] PRM 3220 pjshowrsc No memory (*details*)

### Meaning

Memory acquisition failed.

*details*: Detailed information for maintenance

### Action

Contact the administrator. The administrator is requested to collect investigation data according to the "Administrator's Guide for Maintenance," and then contact a Fujitsu systems engineer (SE) with the collected data together with the output message.

## [ERR.] PRM 3222 pjshowrsc Permission denied rscunitname: *rscuname*

### Meaning

The user does not have the privileges to display the state of the specified resource unit.

*rscuname*: Resource unit name

### Action

Confirm that the specified resource unit name is correct. If the specified resource unit name is correct, the user does not have the display privileges for that resource unit. If necessary, contact the administrator to confirm the display privileges for the relevant resource unit.

## [ERR.] PRM 3226 pjshowrsc There is no information that can be displayed with specified node group ID, boot group ID, or node ID : *id*

### Meaning

There is no node information that can be displayed for the specified node group ID, boot group ID, or node ID *id*.

### Action

The compute resources for the specified node group ID, boot group ID, or node ID *id* may be inactive. Ask the administrator to check the node status. The administrator should check the node status for the specified node group ID, boot group ID, or node ID *id* by using

the pashowclst command.
If the error occurs even though the node status is normal, collect investigation data according to the "Administrator's Guide for Maintenance." Then, contact a Fujitsu systems engineer (SE) with the collected data together with the output message.

### [ERR.] PRM 3227 pjshowrsc There is no information that can be displayed with specified resource unit : *rscuname*

#### Meaning

There is no node information that can be displayed for the specified resource unit name *rscuname*.

#### Action

The compute resources for the specified resource unit *rscuname* may be inactive. Ask the administrator to check the node status. The administrator should check the node status of the specified resource unit *rscuname* by using the pashowclst command.
If the error occurs even though the node status is normal, collect investigation data according to the "Administrator's Guide for Maintenance." Then, contact a Fujitsu systems engineer (SE) with the collected data together with the output message.

### [ERR.] PRM 3228 pjshowrsc A required option is not specified

#### Meaning

An indispensable option is not specified.

#### Action

Confirm the correct option, and execute the command again.

### [ERR.] PRM 3299 pjshowrsc Internal error (*details*)

#### Meaning

An internal error occurred.

*details*: Detailed information for maintenance

#### Action

Contact the administrator. The administrator is requested to collect investigation data according to the "Administrator's Guide for Maintenance," and then contact a Fujitsu systems engineer (SE) with the collected data together with the output message.

## Warning message

### [WARN] PRM 3101 pjshowrsc Connection error: *clstname*

#### Meaning

Resource information display failed because communication was not possible with the active job management node in the cluster *clstname*.

#### Action

Ask the cluster administrator to check the system state. The cluster administrator is requested to check the state of the job management node in the cluster *clstname* by using the pashowclst command. If the error occurs even though the job management node is in the normal state, collect investigation data according to the "Administrator's Guide for Maintenance." Then, contact a Fujitsu systems engineer (SE) with the collected data together with the output message.

## Information message

### [INFO] PRM 3001 pjshowrsc There are no resources can be displayed

#### Meaning

No resource can be displayed.

#### Action

No action is necessary.

## F.4.4 pjstgchk command [FX10]

When a problem occurs during the execution of the pjstgchk command, the following messages are output to the standard error output:

### Error message

#### [ERR.] PJM 4020 pjstgchk Cannot open the file (*path*) : *code*.

Meaning

The specified file *path* cannot be opened.

*code*: Internal code for maintenance

Action

Confirm that the file name and path are correct.

#### [ERR.] PJM 4021 pjstgchk File pathname too long: *type*.

Meaning

The file name is too long.

Action

Specify a job script file name within 4096 characters.

#### [ERR.] PJM 4023 pjstgchk Multiple script files specified. Try `pjstgchk --help` for more information.

Meaning

Multiple job script files were specified.

Action

Do not specify more than one job script file.

#### [ERR.] PJM 4025 pjstgchk Line length exceeds max characters in script file.

Meaning

The maximum number of characters allowed per line in a job script file has been exceeded.

Action

Use up to 4096 characters, including the newline character, per line.

#### [ERR.] PJM 4026 pjstgchk Unable to determine current working directory.

Meaning

Information on the current directory could not be retrieved.

Action

Confirm that the current directory exists, and execute the command again.

#### [ERR.] PJM 4027 pjstgchk Current working directory path contains a newline character.

Meaning

The current directory name is incorrect.

Action

Review the current directory name, and execute the command again. Do not use the linefeed code, backslash + 'n', in a directory name.

#### [ERR.] PJM 4091 pjstgchk Internal error: *details*.

Meaning

An internal error occurred.

*details*: Detailed information for maintenance

Action

The system might not have been able to accept the request temporarily. Retry the command execution. Contact the administrator when the situation does not change in about ten minutes. The administrator is requested to collect investigation data according to the "Administrator's Guide for Maintenance," and then contact a Fujitsu systems engineer (SE) with the collected data together with the output message.

## F.4.5 pjget command [FX10]

When a problem occurs during the execution of the pjget command, the following messages are output to the standard error output:

### Error message

#### [ERR.] PJM 4201 pjget Unknown option *opt*.<br>Try `pjget --help` for more information.

Meaning

The specified option *opt* is unknown.

Action

Specify the correct option.

#### [ERR.] PJM 4210 pjget No jobid(s) specified.<br>Try `pjget --help` for more information.

Meaning

No job ID is specified.

Action

Specify a job ID.

#### [ERR.] PJM 4211 pjget Invalid jobid syntax *jobid*.

Meaning

The specified job ID *jobid* is incorrect.

Action

Specify the correct job ID.

#### [ERR.] PJM 4212 pjget Job *jobid* does not exist.

Meaning

The specified job (job ID: *jobid*) does not exist. Alternatively, the user is not the owner of the specified job or does not have the privileges to retrieve the specified file.

Action

Confirm that the job exists. Alternatively, verify the owner of the job and the privileges to able to retrieve the file.

#### [ERR.] PJM 4213 pjget Job *jobid* is not staging job.

### Meaning

The command was executed for the job which is the batch job with the --no-stging option at job submission or the interactive job.

### Action

The pjget command cannot be used for the non-staging job.

---

#### [ERR.] PJM 4214 pjget Job *jobid* is not running.

### Meaning

The command was executed for the job not in the state of RUNNING.

### Action

Execute the command again after waiting for the state of the job to become RUNNING.

---

#### [ERR.] PJM 4220 pjget Cannot open the file (*path*) : *code*.

### Meaning

The specified file *path* cannot be opened.

*code*: Internal code for maintenance

### Action

Confirm that the file name and path are correct.

---

#### [ERR.] PJM 4290 pjget PJM daemon is not present.

### Meaning

The job manager function is not working.

### Action

The system might not have been able to accept the request temporarily. Retry the command execution. Contact the administrator when the situation does not change in about ten minutes. The administrator is requested to check the operating state of the job manager function.

---

#### [ERR.] PJM 4291 pjget Internal error: *details*.

### Meaning

An internal error occurred.

*details*: Detailed information for maintenance

### Action

The system might not have been able to accept the request temporarily. Retry the command execution. Contact the administrator when the situation does not change in about ten minutes. The administrator is requested to collect investigation data according to the "Administrator's Guide for Maintenance," and then contact a Fujitsu systems engineer (SE) with the collected data together with the output message.

## F.4.6 pjlist command [FX10]

When a problem occurs during the execution of the pjlist command, the following messages are output to the standard error output:

### Error message

---

#### [ERR.] PJM 4101 pjlist Unknown option *opt*.<br>Try `pjlist --help` for more information.

### Meaning

The specified option *opt* is unknown.

### Action

Specify the correct option.

---

### [ERR.] PJM 4110 pjlist No jobid(s) specified. Try `pjlist --help` for more information.

### Meaning

No job ID is specified.

### Action

Specify a job ID.

---

### [ERR.] PJM 4111 pjlist Invalid jobid syntax *jobid*.

### Meaning

The specified job ID *jobid* is incorrect.

### Action

Specify the correct job ID.

---

### [ERR.] PJM 4112 pjlist Job *jobid* does not exist.

### Meaning

The specified job (job ID: *jobid*) does not exist. Alternatively, the user is not the owner of the specified job or does not have the privileges to display the file list created by this job.

### Action

Confirm that the job exists. Alternatively, verify the owner of the job and the privileges to be able to display the file list.

---

### [ERR.] PJM 4113 pjlist Job *jobid* is not staging job.

### Meaning

The command was executed for the job which is the batch job with the --no-stging option at job submission or the interactive job.

### Action

The pjlist command cannot be used for the non-staging job.

---

### [ERR.] PJM 4114 pjlist Job *jobid* is not running.

### Meaning

The command was executed for the job not in the state of RUNNING.

### Action

Execute the command again after waiting for the state of the job to become RUNNING.

---

### [ERR.] PJM 4190 pjlist PJM daemon is not present.

### Meaning

The job manager function is not working.

### Action

The system might not have been able to accept the request temporarily. Retry the command execution. Contact the administrator when the situation does not change in about ten minutes. The administrator is requested to check the operating state of the job manager function.

---

### [ERR.] PJM 4191 pjlist Internal error: *details*.

### Meaning

An internal error occurred.

*details*: Detailed information for maintenance

### Action

The system might not have been able to accept the request temporarily. Retry the command execution. Contact the administrator when the situation does not change in about ten minutes. The administrator is requested to collect investigation data according to the "Administrator's Guide for Maintenance," and then contact a Fujitsu systems engineer (SE) with the collected data together with the output message.

## F.4.7 pjcat command

When a problem occurs during the execution of the pjcat command, the following messages are output to the standard error output:

### Error message

#### [ERR.] PJM 4301 pjcat Unknown option *opt*.<br>Try `pjcat --help` for more information.

### Meaning

The specified option *opt* is unknown.

### Action

Specify the correct option.

#### [ERR.] PJM 4310 pjcat No jobid(s) specified.<br>Try `pjcat --help` for more information.

### Meaning

No job ID is specified.

### Action

Specify a job ID.

#### [ERR.] PJM 4311 pjcat Invalid jobid syntax *jobid*.

### Meaning

The specified job ID *jobid* is incorrect.

### Action

Specify the correct job ID.

#### [ERR.] PJM 4312 pjcat Job *jobid* does not exist.

### Meaning

The specified job (job ID: *jobid*) does not exist. Alternatively, the user is not the owner of the specified job or does not have the privileges to display the standard output, standard error output, or job script file contents of the specified job.

### Action

Confirm that the job exists. Alternatively, verify the owner of the job and the privileges to be able to display the file contents.

#### [ERR.] PJM 4313 pjcat Job *jobid* is not staging job.

### Meaning

The command was executed for the job which is the batch job with the --no-stging option at job submission or the interactive job.

Action

The pjcat command cannot be used for the non-staging job.

**[ERR.] PJM 4314 pjcat Job *jobid* is not running.**

Meaning

The command was executed for the job not in the state of RUNNING.

Action

Execute the command again after waiting for the state of the job to become RUNNING.

**[ERR.] PJM 4390 pjcat PJM daemon is not present.**

Meaning

The job manager function is not working.

Action

The system might not have been able to accept the request temporarily. Retry the command execution. Contact the administrator when the situation does not change in about ten minutes. The administrator is requested to check the operating state of the job manager function.

**[ERR.] PJM 4391 pjcat Internal error: *details*.**

Meaning

An internal error occurred.

*details*: Detailed information for maintenance

Action

The system might not have been able to accept the request temporarily. Retry the command execution. Contact the administrator when the situation does not change in about ten minutes. The administrator is requested to collect investigation data according to the "Administrator's Guide for Maintenance," and then contact a Fujitsu systems engineer (SE) with the collected data together with the output message.

### Warning message

**[WARN] PJM 4321 pjcat STDERR of job *jobid* is merged with STDOUT.**

Meaning

The standard error of the job (*jobid*) cannot be shown with the -e option of the pjcat command because the standard error is redirected to the standard output by the -j option of the pjsub command.

Action

Use the -o option of the pjcat command.

# F.5 Messages of the command for job operation

## F.5.1 pjsub command

When a problem occurs during the execution of the pjsub command, the following messages are output to the standard error output:

### Error message

**[ERR.] PJM 0001 pjsub Unknown option *opt*.**
**Script file *filename* line *lineno*.**
**Try `pjsub --help` for more information.**

### Meaning

The specified option opt is unknown. The location of the detected problem is line *lineno* of the job script *filename*.

### Action

Specify the correct option, or specify an argument for the option.

---

## [ERR.] PJM 0002 pjsub Combination of option is illegal. Try `pjsub --help` for more information.

### Meaning

The combination of specified options is invalid.

### Action

Specify the correct combination of options.

---

## [ERR.] PJM 0003 pjsub Unknown option argument *arg*. Script file *filename* line *lineno*. Try `pjsub --help` for more information.

### Meaning

For the option, the specified argument arg is unknown, or no argument is specified. The location of the detected problem is line lineno of the job script *filename*.

### Action

Specify the correct argument.

---

## [ERR.] PJM 0004 pjsub Argument format error *arg*.

### Meaning

The setting value of the argument *arg* in the option is incorrect.

### Action

Specify the correct setting value.

---

## [ERR.] PJM 0005 pjsub Resource value is out of range: *rscname*.

### Meaning

1. The resource limit value of *rscname*=*xxx* specified in the -L|--rsc-list option is outside the specified range.
2. The amount of resources for the specified resource unit and resource group exceeds the limit.

### Action

1. Check the specifiable range with the pjacl command.
2. Check the amount of resources for the resource unit and resource group, with the --rsc option of the pjstat command.

---

## [ERR.] PJM 0006 pjsub The parameter of MPI option is illegal: *kind*

### Meaning

The --mpi option is incorrect.

*kind*: Incorrect parameter type

### Action

Review the specified value, and execute the command again.

### [ERR.] PJM 0007 pjsub Staging option error (*errnum*). Refer to the staging information file. (*filename*)

#### Meaning

The staging command is incorrect

*errnum*: Number of error locations
*filename* : The file name in which the error information of the staging is stored.

#### Action

Check the contents of the error in the file *filename*. After resolving the problem, submit the job again.

### [ERR.] PJM 0008 pjsub Option arguments must be enclosed in quotation mark.

#### Meaning

The double quote or single quote is not a pair at the description of the argument of the pjsub command in the job script.

#### Action

Describe the double quotes or the single quotes by the pair.

### [ERR.] PJM 0009 pjsub Illegal option argument: *arg*

#### Meaning

The specified argument *arg* is incorrect.

#### Action

Specify the correct argument.

### [ERR.] PJM 0012 pjsub Job *jobid* does not exist.

#### Meaning

The specified job (job ID: *jobid*) does not exist. Or the user is not the job owner of the specified job.

#### Action

Confirm that the job exists. Or verify the owner of the job.

### [ERR.] PJM 0018 pjsub Job name is mismatch. (*filename*)

#### Meaning

The job name which is specified with the --sparam "jnam=" option in the job script (*file name*) is different from the job name of other job scripts.

#### Action

Specify same name in all job scripts when you specify the job name with --sparam "jnam=" option in the job scripts.

### [ERR.] PJM 0020 pjsub Cannot open the file (*path*) : *code*.

#### Meaning

The specified file path cannot be opened.

*code*: Internal code for maintenance

#### Action

Confirm that the file name and path are correct.

### [ERR.] PJM 0021 pjsub File pathname too long: *type*.

### Meaning

The file name is too long.

| type | Meaning |
|---|---|
| Scriptfile | Job script file |
| Hostfile | Host file specified with rank-map-hostfile parameter |
| Infofile | Job statistical information output file specified with --spath option |

### Action

Specify a job script file name within 4096 characters.

---

## [ERR.] PJM 0022 pjsub File format error: *filename*. File line *lineno*.

### Meaning

Line *lineno* of the file *filename* contains a format error.

### Action

Write the file in the correct format.

---

## [ERR.] PJM 0023 pjsub Multiple script files specified. Try `pjsub --help` for more information.

### Meaning

Multiple job script files were specified.

### Action

Do not specify more than one job script file.

---

## [ERR.] PJM 0025 pjsub Line length exceeds max characters in script file.

### Meaning

The maximum number of characters allowed per line in a job script file has been exceeded.

### Action

Use up to 4096 characters, including the newline character, per line.

---

## [ERR.] PJM 0026 pjsub Unable to determine current working directory.

### Meaning

Information on the current directory could not be retrieved.

### Action

Confirm that the current directory exists on the login node, and execute the command again.

---

## [ERR.] PJM 0027 pjsub Current working directory path contains a newline character.

### Meaning

The current directory name is incorrect. The directory name contains a linefeed code.

### Action

Review the current directory name, and execute the command again. Do not use the linefeed code, backslash +'n', in a directory name.

---

## [ERR.] PJM 0028 pjsub Unable to make the file: *path*.

### Meaning

The specified file *path* cannot be created.

### Action

Confirm that the specified file *path* can be created.

---

## [ERR.] PJM 0040 pjsub User does not exist. Not found *uid* in passwd file.

### Meaning

The user name could not be obtained based on the user ID *uid*.

### Action

Confirm that the user corresponding to the ID is registered in the system.

---

## [ERR.] PJM 0041 pjsub Group does not exist. Not found *gid* in passwd file.

### Meaning

The group name could not be obtained based on the group ID *gid*.

### Action

Confirm that the group corresponding to the ID is registered in the system.

---

## [ERR.] PJM 0056 pjsub The specified execution time is earlier than the current time: *time*

### Meaning

The time *time* specified for the --at option is earlier than the current time.

### Action

Specify a future time for the --at option.

---

## [ERR.] PJM 0060 pjsub Subjob number overflowed.

### Meaning

The sub job number is outside the prescribed range.

### Action

Specify a sub job number from 0 to 65535 for a step job, and a sub job number from 0 to 999999 for a bulk job.

---

## [ERR.] PJM 0062 pjsub The value "n" is too large: "--mpi rank-map-bychip=n".

### Meaning

The value *n* specified for the option --mpi rank-map-bychip=*n* exceeds the number of CPU cores installed on the compute node.

### Action

Submit the job again specifying a value smaller than the number of CPU cores installed on the compute node.

---

## [ERR.] PJM 0063 pjsub The value "m" must be a multiple of the value "n": "-P vn-policy=unpack/abs-unpack=m", "--mpi rank-map-bychip=n".

### Meaning

The value *m*, which is specified in the option -P "vn-policy=unpack/abs-unpack=*m*" must be a multiple of the value *n*, which is specified in the option --mpi "rank-map-bychip=*n*."

### Action

The value specified in the option -P "vn-policy=unpack/abs-unpack=*m*" must be a multiple of the value *n* specified in the option --mpi "rank-map-bychip=*n*."

---

## [ERR.] PJM 0064 pjsub The number of virtual nodes "m" must be a multiple of the value "n": "-L vnode=m", "-P vn-policy=unpack/abs-unpack=n".

### Meaning

The number of virtual nodes *m*, which is specified in the option -L "vnode=*m*" must be a multiple of the value *n*, which is specified in the option -P "vn-policy=unpack/abs-unpack=*n*."

### Action

The value specified in the option -L "vnode=*n*" must be a multiple of the value specified in the option -P "vn-policy=unpack/abs-unpack=*m*."

---

## [ERR.] PJM 0065 pjsub The option "-P vn-policy=pack/abs-pack" and "--mpi rank-map-bychip=n" are mutually exclusive.

### Meaning

The option -P vn-policy=pack/abs-pack and --mpi rank-map-bychip=*n* are mutually exclusive.

### Action

Do not specify the options -P vn-policy=pack/abs-pack and --mpi rank-map-bychip=*n* at the same time.

---

## [ERR.] PJM 0066 pjsub The number of processes "n" must be smaller than or equal to the number of virtual nodes "m": "-L vnode=m", "--mpi proc=n".

### Meaning

The number of processes *n*, which is specified in the option --mpi "proc=*n*" must be equal to or lower than the number of virtual nodes *m*, which is specified in the option -L "vnode=*m*."

### Action

For the number of processes *n*, which is to be specified in the option --mpi "proc=*n*," specify a value equal to or lower than the number of virtual nodes *m*, which is to be specified in the option -L "vnode=*m*."

---

## [ERR.] PJM 0067 pjsub The option "-L vnode=n" must be specified.

### Meaning

The number of virtual nodes (the option -L "vnode=*n*") is necessary to submit a job to PRIMERGY compute nodes.

### Action

Submit a job again specifying the number of virtual nodes with the option -L "vnode=*n*".

---

## [ERR.] PJM 0068 pjsub The node type of job and resource unit do not match.

### Meaning

Job acceptance failed because the type of compute node belonging to the resource unit was changed during job acceptance processing.

### Action

Check the resource unit configuration on the system, and then submit the job again.

---

## [ERR.] PJM 0070 pjsub No execute: *opt*.

### Meaning

Command execution or the specified option is not permitted.

*opt* : If execution of the pjsub command is not permitted, "pjsub" is displayed. If the specified option is not permitted, "pjsub-xxxx" is displayed.

### Action

Display the job ACL settings with the pjacl command. Then, check whether the pjsub command can be executed (execute pjsub or execute pjsub(--*xxxx*), where *xxxx* is the option name) for the target resource unit and resource group. If necessary, ask the administrator to add permission.

---

## [ERR.] PJM 0071 pjsub No permit: *target*.

### Meaning

Command execution is not permitted for the permition '*target*.'

target is displayed in following form.

group (*group*) : The group name on the operating system.

### Action

Display the job ACL settings with the pjacl command. Then, confirm the permission for the permission of the pjsub command (permit pjsub) for the target resource unit and resource group to which the job is submitted. If necessary, ask the administrator to add permission.

---

## [ERR.] PJM 0072 pjsub Submit limit over: *target*.

### Meaning

The upper limit on the number of jobs accepted simultaneously has been reached.
"ru-accept" at *target* indicates a batch job. "ru-interact-accept" at *target* indicates an interactive job.

### Action

With the --rscunit option of the pjacl command, check the number of jobs accepted simultaneously for the resource unit to be changed. The number of jobs accepted simultaneously is the "acceptable job" or "acceptable job (interact)" value of "limit in rscunit (each users)," "limit in rscunit (total users in same group)," or "limit in rscunit (total all users)." If necessary, ask the administrator to increase the number of jobs accepted simultaneously.

---

## [ERR.] PJM 0073 pjsub Option value is out of range: *opt*.

### Meaning

The value specified by the option *opt* exceeds the range defined by the job ACL function.

### Action

With the pjacl command, check the upper limit on the limit values that can be specified for the option *opt*. If necessary, ask the administrator to increase the number of accepted commands.

---

## [ERR.] PJM 0079 pjsub Gate check error.

### Meaning

An error occurred in gate check.

### Action

A balance necessary for the job submission is insufficient. The user is requested to contact the administrator.

---

## [ERR.] PJM 0084 pjsub Interactive job can be executed on L.

### Meaning

An interactive job is executed on a node other than login node.

### Action

Execute interactive jobs on a login node.

### [ERR.] PJM 0085 pjsub Interactive job *jobid* failed.

#### Meaning

Execution of an interactive job (job ID: *jobid*) failed.

#### Action

Execute the command again.

### [ERR.] PJM 0089 pjsub Not supported: *message*.

#### Meaning

The specified function or combination of functions is not supported by the current version of Technical Computing Suite.

#### Action

By referring to the information displayed in *message*, correct the option specification to avoid the use of an unsupported function or combination of functions, and then submit the job again.

### [ERR.] PJM 0090 pjsub PJM daemon is not present.

#### Meaning

The job manager function is not working, or communication with the job manager function is disabled.

#### Action

The system might not have been able to accept the request temporarily. Retry the command execution. Contact the administrator when the situation does not change in about ten minutes. The administrator is requested to check the operating state of the job manager function and the system state.

### [ERR.] PJM 0091 pjsub Internal error: *details*.

#### Meaning

An internal error occurred.

*details*: Detailed information for maintenance

#### Action

The system might not have been able to accept the request temporarily. Retry the command execution. Contact the administrator when the situation does not change in about ten minutes. The administrator is requested to collect investigation data according to the "Administrator's Guide for Maintenance," and then contact a Fujitsu systems engineer (SE) with the collected data together with the output message.

### [ERR.] PJM 0093 pjsub This command can be executed on L/JCM.

#### Meaning

The command was executed on a node other than a login or job management node.

#### Action

Execute the command on a login or job management node.

### [ERR.] PJM 0095 pjsub The "rank-map-bychip=*n*" is too large.

#### Meaning

The specified value *n* is too large.

#### Action

Specify a value that is smaller than the value *m*, which is specified in the option -P "vn-policy=unpack/abs-unpack=*m*."

## Warning message

### [WARN] PJM 0024 pjsub Too many arguments in script file. Script file *filename* line *lineno*. Extra arguments ignored: *arg*.

#### Meaning

The maximum number of options (64) allowed per line in a job script file has been exceeded. The location of the detected problem is line *lineno* of the job script file *filename*.

#### Action

The subsequent options that exceed the upper limit are ignored, and processing continues.

## Information message

### [INFO] PJM 0000 pjsub Job *jobid* submitted.

#### Meaning

The job was accepted normally. The job ID is *jobid*.

#### Action

No action is necessary.

### [INFO] PJM 0080 pjsub Interactive job jobid is canceled due to the resource allocation timeout. The timeout period "t" can be specified by "--sparam wait-time=t".

#### Meaning

The interactive job (job ID: jobid) was canceled because the computer resources to be allocated to the job could not be determined within a certain period. That was due to insufficient free computer resources or the job ACL limit value.

#### Action

Check the availability of resources and the job ACL setting value, and submit the job again. You can specify the wait time for the completion of computer resource allocation in the --sparam "wait-time=" option of the pjsub command. The default value is 0 seconds.

### [INFO] PJM 0081 .....connected.

#### Meaning

An interactive job is being prepared. One dot (.) appears every three seconds. 'connected' appears when the preparation is complete.

#### Action

No action is necessary.

### [INFO] PJM 0082 pjsub Interactive job *jobid* started.

#### Meaning

Execution of an interactive job (job ID: *jobid*) has begun.

#### Action

No action is necessary.

### [INFO] PJM 0083 pjsub Interactive job *jobid* completed.

#### Meaning

An interactive job (job ID: *jobid*) has been completed.

#### Action

No action is necessary.

**[INFO] PJM 0086 pjsub Interactive job *jobid* canceled.**

Meaning

An interactive job (job ID: *jobid*) was canceled.

Action

No action is necessary.

## F.5.2 pjdel command

When a problem occurs during the execution of the pjdel command, the following messages are output to the standard error output:

### Error message

**[ERR.] PJM 0101 pjdel Unknown option *opt*.**
**Try `pjdel --help` for more information.**

Meaning

The specified *option* is an unrecognizable option.

Action

Specify the correct option.

**[ERR.] PJM 0102 pjdel Combination of option is illegal.**
**Try `pjdel --help` for more information.**

Meaning

The combination of specified options is invalid.

Action

Specify the correct combination of options.

**[ERR.] PJM 0103 pjdel Unknown option argument *arg*.**
**Try `pjdel --help` for more information.**

Meaning

For the option, the specified argument *arg* is unknown. Or no argument is specified for the option.

Action

Specify the correct argument.

**[ERR.] PJM 0104 pjdel Argument format error *arg*.**

Meaning

The setting value of the argument *arg* in the option is incorrect.

Action

Specify the correct setting value.

**[ERR.] PJM 0110 pjdel No jobid(s) specified.**
**Try `pjdel --help` for more information.**

Meaning

The job ID is not specified.

### Action

Specify the job ID.

---

## [ERR.] PJM 0111 pjdel Invalid jobid syntax *jobid*.

### Meaning

The specification of the job ID *jobid* s incorrect.

### Action

Specify the correct job ID.

---

## [ERR.] PJM 0112 pjdel Job *jobid* does not exist.

### Meaning

The specified job (job ID: *jobid*) does not exist. Alternatively, the user is not the owner of the specified job or does not have the privileges to delete this job.

### Action

Confirm that the job exists. Alternatively, verify the owner of the job and the privileges to be able to delete the job.

---

## [ERR.] PJM 0113 pjdel Job *jobid* state error.

### Meaning

The command cannot be executed in the state of the specified job (job ID: *jobid*).

### Action

Execute it when the state of the job is QUEUED, STGIN, READY, RUNNING-A, RUNNING, RUNOUT, HOLD, or ERROR.

---

## [ERR.] PJM 0170 pjdel No execute.
## [ERR.] PJM 0170 pjdel No execute: *opt*, *jobid*.

### Meaning

There are two types of messages.
If the message does not display a job ID, the pjdel command execution is not permitted.
If *opt* is such as "pjdel-*xxxx*", the user is not permitted to specify the option *xxxx* for the target job *jobid*.

### Action

Display the job ACL settings with the pjacl command, and check the permission to execute the pjdel command (execute pjdel) or .to specify the option (execute pjdel(--*xxxx*)). If necessary, ask the administrator to add permission.

---

## [ERR.] PJM 0171 pjdel No permit: *jobid*.

### Meaning

The user is not permitted to operate the target job (job ID: *jobid*).
The operation is not permitted for the user or group executing the target job.

### Action

Display the job ACL settings with the pjacl command. Then, confirm the permission (permit pjdel) for the operated objects of the pjdel command for the resource unit and resource group to which the target job is submitted. If necessary, ask the administrator to add permission.

---

## [ERR.] PJM 0190 pjdel PJM daemon is not present.

### Meaning

The job manager function is not operating, or the communication to the job manager function is unavailable.

### Action

The system might not have been able to accept the request temporarily. Retry the command execution. Contact the administrator when the situation does not change in about ten minutes. The administrator must confirm the job manager function is executing.

---

**[ERR.] PJM 0191 pjdel Internal error: *details*.**

### Meaning

An internal error occurred.

*details*: Detailed information on maintenance

### Action

The system might not have been able to accept the request temporarily. Retry the command execution. Contact the administrator when the situation does not change in about ten minutes. The administrator shall collect investigation data according to the "Administrator's Guide for Maintenance," and then contact a Fujitsu systems engineer (SE) with the collected data together with the output message.

---

**[ERR.] PJM 0193 pjdel This command can be executed on L/JCM.**

### Meaning

The command was executed on a node other than a login or job management node.

### Action

Execute the command on the login node or the job management node.

#### Information message

---

**[INFO] PJM 0100 pjdel Job *jobid* canceled.**

### Meaning

The deletion of the job *jobid* was accepted.

### Action

No action is necessary.

## F.5.3 pjhold command

If an error occurs during the execution of the pjhold command, one of the following messages is output to the standard error output. Also, the information message that reports process completion, etc. is output to standard output.

#### Error message

---

**[ERR.] PJM 0301 pjhold Unknown option *opt*.**
**Try `pjhold --help` for more information.**

### Meaning

The specified option opt is unrecognizable.

### Action

Specify the correct option.

---

**[ERR.] PJM 0302 pjhold Combination of option is illegal (*opt*).**
**Try `pjhold --help` for more information.**

### Meaning

The combination of specified options *opt* is invalid.

### Action

Specify the correct combination of the options.

---

## [ERR.] PJM 0303 pjhold Unknown option argument *arg*. Try `pjhold --help` for more information.

### Meaning

The specified *arg* is an unrecognizable argument of the option, or no argument is specified to the option.

### Action

Specify the correct argument.

---

## [ERR.] PJM 0304 pjhold Argument format error *arg*.

### Meaning

The setting value of the argument *arg* in the option is incorrect.

### Action

Specify the correct setting value.

---

## [ERR.] PJM 0310 pjhold No jobid(s) specified. Try `pjhold --help` for more information.

### Meaning

No job ID is specified.

### Action

Specify a job ID.

---

## [ERR.] PJM 0311 pjhold Invalid jobid syntax *jobid*.

### Meaning

The specified job ID *jobid* is incorrect.

### Action

Specify the correct job ID.

---

## [ERR.] PJM 0312 pjhold Job *jobid* does not exist.

### Meaning

The specified job (job ID: *jobid*) does not exist.

### Action

Confirm that the job exists. Alternatively, verify the owner of the job and the privileges to be able to place the job in the hold state.

---

## [ERR.] PJM 0313 pjhold Job *jobid* state error.

### Meaning

The command cannot be executed because of the state of the specified job (job ID: jobid).

### Action

Check that the job is in the state QUEUED, STGIN, READY, RUNNING-A or RUNNING that the job can be held.
If the automatically re-execution is disabled for the job by --norestart option or job operation setting, the job can be held only at the status of QUEUED, STGIN and READY.

---

## [ERR.] PJM 0314 pjhold Job *jobid* type error.

### Meaning

The command cannot be executed with the type of the specified job (job ID: *jobid*).

### Action

Check the type of the specified job.

---

## [ERR.] PJM 0370 pjhold No execute.
## [ERR.] PJM 0370 pjhold No execute: *opt*, *jobid*.

### Meaning

There are two types of messages.
If the message does not display a job ID, the pjhold command execution is not permitted.
If *opt* is such as "pjhold-*xxxx*", the user is not permitted to specify the option *xxxx* for the target job *jobid*.

### Action

Display the job ACL settings with the pjacl command, and and check the permission to execute the pjhold command (execute pjhold) or .to specify the option (execute pjhold(--*xxxx*)). If necessary, ask the administrator to add permission.

---

## [ERR.] PJM 0371 pjhold No permit: *jobid*.

### Meaning

The user is not permitted to operate the target job (job ID: *jobid*).

The operation is not permitted for the user or group executing the target job.

### Action

Display the job ACL settings with the pjacl command. Then, confirm the permission (permit pjhold) for the operated objects of the pjhold command for the resource unit and resource group to which the target job is submitted. If necessary, ask the administrator to add permission.

---

## [ERR.] PJM 0390 pjhold PJM daemon is not present.

### Meaning

The job manager function is not working, or communication with the job manager function is disabled.

### Action

The system might not have been able to accept the request temporarily. Retry the command execution. Contact the administrator when the situation does not change in about ten minutes. The administrator is requested to check the operating state of the job manager function or the system state.

---

## [ERR.] PJM 0391 pjhold Internal error: *details*.

### Meaning

An internal error occurred.

*details*: Detailed information for maintenance

### Action

The system might not have been able to accept the request temporarily. Retry the command execution. Contact the administrator when the situation does not change in about ten minutes. The administrator is requested to collect investigation data according to the "Administrator's Guide for Maintenance," and then contact a Fujitsu systems engineer (SE) with the collected data together with the output message.

---

## [ERR.] PJM 0393 pjhold This command can be executed on L/JCM.

### Meaning

The command was executed on a node other than a login or job management node.

Action

Execute the command on a login or job management node.

### Information message

**[INFO] PJM 0300 pjhold Accepted job *jobid*.**

Meaning

The request to place a job (job ID: *jobid*) in the hold state was accepted.

Action

No action is necessary.

## F.5.4 pjrls command

If an error occurs during the execution of the pjrls command, one of the following messages is output to the standard error output. Also, the information message that reports process completion, etc. is output to standard output.

### Error message

**[ERR.] PJM 0401 pjrls Unknown option *opt*.**
**Try `pjrls --help` for more information.**

Meaning

The specified option *opt* is unrecognizable.

Action

Specify the correct option.

**[ERR.] PJM 0410 pjrls No jobid(s) specified.**
**Try `pjrls --help` for more information.**

Meaning

No job ID is specified.

Action

Specify a job ID.

**[ERR.] PJM 0411 pjrls Invalid jobid syntax *jobid*.**

Meaning

The specified job ID *jobid* is incorrect.

Action

Specify the correct job ID.

**[ERR.] PJM 0412 pjrls Job *jobid* does not exist.**

Meaning

The specified job (job ID: *jobid*) does not exist.

Action

Confirm that the job exists. Alternatively, verify the owner of the job and the privileges to be able to cancel the hold state of the job.

**[ERR.] PJM 0413 pjrls Job *jobid* state error.**

### Meaning

The command cannot be executed because of the state of the specified job (job ID: *jobid*).

### Action

Check the job state.

---

## [ERR.] PJM 0470 pjrls No execute.

### Meaning

The user is not permitted to execute the command.

### Action

Display the job ACL settings with the pjacl command, and check whether the pjrls command can be executed (execute pjrls). If necessary, ask the administrator to add permission.

---

## [ERR.] PJM 0471 pjrls No permit: *jobid*.

### Meaning

The user is not permitted to operate the target job (job ID: *jobid*).

Cancellation of the hold state is not permitted for the user or group executing the target job.

### Action

Display the job ACL settings with the pjacl command. Then, confirm the permission (permit pjrls) for the operated objects of the pjrls command for the resource unit and resource group to which the target job is submitted. If necessary, ask the administrator to add permission.

---

## [ERR.] PJM 0490 pjrls PJM daemon is not present.

### Meaning

The job manager function is not working, or communication with the job manager function is disabled.

### Action

The system might not have been able to accept the request temporarily. Retry the command execution. Contact the administrator when the situation does not change in about ten minutes. The administrator is requested to check the operating state of the job manager function or the system state.

---

## [ERR.] PJM 0491 pjrls Internal error: *details*.

### Meaning

An internal error occurred.

*details*: Detailed information for maintenance

### Action

The system might not have been able to accept the request temporarily. Retry the command execution. Contact the administrator when the situation does not change in about ten minutes. The administrator is requested to collect investigation data according to the "Administrator's Guide for Maintenance," and then contact a Fujitsu systems engineer (SE) with the collected data together with the output message.

---

## [ERR.] PJM 0493 pjrls This command can be executed on L/JCM.

### Meaning

The command was executed on a node other than a login or job management node.

### Action

Execute the command on a login or job management node.

### Information message

**[INFO] PJM 0400 pjrls Job *jobid* released.**

Meaning

The hold state of the job (job ID: jobid) was canceled.

Action

No action is necessary.

## F.5.5 pjwait command

When a problem occurs during the execution of the pjwait command, the following messages are output to the standard error output:

### Error message

**[ERR.] PJM 0601 pjwait Unknown option *opt*.**
**Try `pjwait --help` for more information.**

Meaning

The specified option *opt* is unrecognizable.

Action

Specify the correct option.

**[ERR.] PJM 0604 pjwait Argument format error *arg*.**

Meaning

The setting value of the argument *arg* in the option is incorrect.

Action

Specify the correct setting value.

**[ERR.] PJM 0610 pjwait No jobid(s) specified.**
**Try `pjwait --help` for more information.**

Meaning

No job ID is specified.

Action

Specify a job ID.

**[ERR.] PJM 0611 pjwait Invalid jobid syntax *jobid*.**

Meaning

The specified job ID *jobid* is incorrect.

Action

Specify the correct job ID.

**[ERR.] PJM 0670 pjwait No execute.**

Meaning

The user is not permitted to execute the command.

### Action

Display the job ACL settings with the pjacl command, and check whether the pjwait command can be executed (execute pjwait). If necessary, ask the administrator to add permission.

---

**[ERR.] PJM 0671 pjwait No permit: *jobid*.**

### Meaning

The user does not have sufficient permission to operate the job (job ID: *jobid*).

There is no operation permission for the user or group executing the target job.

### Action

Display the job ACL settings with the pjacl command. Then, confirm the permission (permit pjwait) for the operated objects of the pjwait command for the resource unit and resource group to which the target job is submitted. If necessary, ask the administrator to add permission.

---

**[ERR.] PJM 0690 pjwait PJM daemon is not present.**

### Meaning

The job manager function is not working, or communication with the job manager function is disabled.

### Action

The system might not have been able to accept the request temporarily. Retry the command execution. Contact the administrator when the situation does not change in about ten minutes. The administrator is requested to check the operating state of the job manager function or the system state.

---

**[ERR.] PJM 0691 pjwait Internal error: *details*.**

### Meaning

An internal error occurred.

*details*: Detailed information for maintenance

### Action

The system might not have been able to accept the request temporarily. Retry the command execution. Contact the administrator when the situation does not change in about ten minutes. The administrator is requested to collect investigation data according to the "Administrator's Guide for Maintenance," and then contact a Fujitsu systems engineer (SE) with the collected data together with the output message.

---

**[ERR.] PJM 0693 pjwait This command can be executed on L/JCM.**

### Meaning

The command was executed on a node other than a login or job management node.

### Action

Execute the command on a login or job management node.

## F.5.6 pjsig command

If an error occurs during the execution of the pjsig command, one of the following messages is output to the standard error output. Also, the information message that reports process completion, etc. is output to standard output.

### Error message

---

**[ERR.] PJM 0701 pjsig Unknown option *opt*.**
**Try `pjsig --help` for more information.**

### Meaning

The specified option *opt* is unrecognizable.

### Action

Specify the correct option.

---

## [ERR.] PJM 0702 pjsig Combination of option is illegal (*opt*). Try `pjsig --help` for more information.

### Meaning

The combination of specified options *opt* is invalid.

### Action

Specify the correct combination of the options.

---

## [ERR.] PJM 0703 pjsig Unknown option argument *arg*. Try `pjsig --help` for more information.

### Meaning

The specified *arg* is an unrecognizable argument of the option, or no argument is specified to the option.

### Action

Specify the correct argument.

---

## [ERR.] PJM 0704 pjsig Argument format error *arg*.

### Meaning

The setting value of the argument *arg* in the option is incorrect.

### Action

Specify the correct setting value.

---

## [ERR.] PJM 0710 pjsig No jobid(s) specified. Try `pjsig --help` for more information.

### Meaning

No job ID is specified.

### Action

Specify a job ID.

---

## [ERR.] PJM 0711 pjsig Invalid jobid syntax *jobid*.

### Meaning

The specified job ID *jobid* is incorrect.

### Action

Specify the correct job ID.

---

## [ERR.] PJM 0712 pjsig Job *jobid* does not exist.

### Meaning

The specified job (job ID: *jobid*) does not exist.

### Action

Confirm that the job exists.

### [ERR.] PJM 0713 pjsig Job *jobid* state error.

#### Meaning

The command cannot be executed because of the state of the specified job (job ID: *jobid*).

#### Action

Check the state of the specified job.

### [ERR.] PJM 0714 pjsig Job *jobid* type error.

#### Meaning

The command cannot be executed because of the type of the specified job (job ID: *jobid*).

#### Action

Check the type of the specified job.

### [ERR.] PJM 0770 pjsig No execute.

#### Meaning

The user is not permitted to execute the command.

#### Action

Display the job ACL settings with the pjacl command, and check whether the pjsig command can be executed (execute pjsig). If necessary, ask the administrator to add permission.

### [ERR.] PJM 0771 pjsig No permit: *jobid*.

#### Meaning

The user does not have operation permission for the job (job ID: *jobid*).

There is no operation permission for the user or group executing the target job.

#### Action

Display the job ACL settings with the pjacl command. Then, confirm the permission (permit pjsig) for the operated objects of the pjsig command for the resource unit and resource group to which the target job is submitted. If necessary, ask the administrator to add permission.

### [ERR.] PJM 0790 pjsig PJM daemon is not present.

#### Meaning

The job manager function is not working, or communication with the job manager function is disabled.

#### Action

The system might not have been able to accept the request temporarily. Retry the command execution. Contact the administrator when the situation does not change in about ten minutes. The administrator is requested to check the operating state of the job manager function or the system state.

### [ERR.] PJM 0791 pjsig Internal error: *details*.

#### Meaning

An internal error occurred.

*details*: Detailed information for maintenance

#### Action

The system might not have been able to accept the request temporarily. Retry the command execution. Contact the administrator when the situation does not change in about ten minutes. The administrator is requested to collect investigation data according to the

"Administrator's Guide for Maintenance," and then contact a Fujitsu systems engineer (SE) with the collected data together with the output message.

#### [ERR.] PJM 0793 pjsig This command can be executed on L/JCM.

Meaning

The command was executed on a node other than a login or job management node.

Action

Execute the command on a login or job management node.

### Information message

#### [INFO] PJM 0700 pjsig Accepted job *jobid* is sent signal *sig*.

Meaning

A signal *sig* was sent for the job (job ID: *jobid*).

*sig*: Displays the specified signal.

Action

No action is necessary.

## F.5.7 pjalter command

When a problem occurs during the execution of the pjalter command, the following messages are output to the standard error output. Also, the information message that reports process completion, etc. is output to standard output.

### Error message

#### [ERR.] PJM 0501 pjalter Unknown option *opt*.<br>Try `pjalter --help` for more information.

Meaning

The specified option opt is unrecognizable.

Action

Specify the correct option.

#### [ERR.] PJM 0502 pjalter Combination of option is illegal.<br>Try `pjalter --help` for more information.

Meaning

The combination of specified options *opt* is invalid.

Action

Specify the correct combination of the options.

#### [ERR.] PJM 0504 pjalter Argument format error *arg*.

Meaning

The setting value of the argument arg in the option is incorrect.

Action

Specify the correct setting value.

#### [ERR.] PJM 0505 pjalter Resource value is out of range: *rscname*, *jobid*.

### Meaning

The resource information specified with the -L|--rsc-list option for the job specified by the job ID *jobid* is invalid.

- If *rscname* is rscunit:
  - The specified resource unit does not exist.
  - An attempt was made to perform either of following resource unit changes, which is not allowed:
    Change from a resource unit with FX100 or FX10 compute nodes to a resource unit with PRIMERGY compute nodes
    Change from a resource unit with PRIMERGY compute nodes to a resource unit with FX100 or FX10 compute nodes
- If *rscname* is rscgrp:
  - The specified resource group does not exist.
  - An attempt was made to change the resource unit although no resource group is specified.
    You also need to specify a resource group because no default resource group is set for the change-to resource unit.
- If *rscname* is any of the following:
  node, node-cpu, elapse, node-mem, node-quota, proc-core, proc-cpu, proc-crproc, proc-data, proc-lockm, proc-msgq, proc-openfd, proc-psig, proc-filesz, proc-stack, proc-vmem, vnode-core, vnode-mem, subjobnum, core, mem, core*core-mem, -p (priority)

  Each of these values represents a job ACL definition item and means that the specified limit exceeds the upper limit defined by the indicated definition item.
  Note, however, that the meanings of core, mem, core*core-mem, and -p (priority) are different from those indicated above, as follows:
  core: Indicates that the upper limit defined by upper vnode-core is exceeded. [FX100/FX10]
  mem, core*core-mem: Indicates that the upper limit defined by upper vnode-mem is exceeded. [FX100/FX10]
  -p (priority): upper priv-pri

### Action

Specify the resource unit name or resource group name correctly. Alternatively, make sure that no resource limit values exceed the upper limit defined in the job ACL.

---

## [ERR.] PJM 0510 pjalter No jobid(s) specified. Try `pjalter --help` for more information.

### Meaning

No job ID is specified.

### Action

Specify a job ID.

---

## [ERR.] PJM 0511 pjalter Invalid jobid syntax *jobid*.

### Meaning

The job ID *jobid* is incorrect.

### Action

Specify the correct job ID.

---

## [ERR.] PJM 0512 pjalter Job *jobid* does not exist.

### Meaning

The job specified by the job ID *jobid* does not exist.

### Action

Confirm that the job ID is correct.

---

## [ERR.] PJM 0513 pjalter Job *jobid* state error.

### Meaning

The command cannot be executed in the state of the job specified by the job ID *jobid*.

### Action

Execute the pjalter command when the state of the job is QUEUED, HOLD, or ERROR.

---

## [ERR.] PJM 0514 pjalter Job *jobid* type error.

### Meaning

The command cannot be executed for the type of the job specified by the job ID *jobid*.

### Action

Check the type of the specified job.

---

## [ERR.] PJM 0515 pjalter Job *jobid* model error.

### Meaning

There is a specification that is inappropriate for the model of the job specified with the job ID *jobid*.
You cannot specify a sub job ID for a step job when changing the resource unit name.
You cannot specify a sub job ID for a bulk job.

### Action

Check the model of the specified job and review the specified job ID.

---

## [ERR.] PJM 0570 pjalter No execute.
## [ERR.] PJM 0570 pjalter No execute: *jobid*.

### Meaning

There are two types of messages.
If the message does not display a job ID, pjalter command execution is not permitted.
If it displays the job ID *jobid*, the parameter change failed because the user does not have the privileges to submit job *jobid* on the specified resource unit or resource group.

### Action

Display the job ACL settings with the pjacl command. Confirm the pjalter command execution privileges (execute pjalter) and the privileges to submit the job to the specified resource unit or resource group (execute pjsub-*).
If necessary, ask the administrator to add the pjalter command execution privileges and job submission privileges.

---

## [ERR.] PJM 0571 pjalter No permit: *jobid*.

### Meaning

The user does not have sufficient permission to operate the job.
Changing of the parameters is not permitted for the user or group executing the target job.

### Action

Display the job ACL settings with the pjacl command. Then, confirm the permission (permit pjalter) for the operated objects of the pjalter command for the resource unit and resource group to which the target job is submitted. If necessary, ask the administrator to add permission.

---

## [ERR.] PJM 0572 pjalter Submit limit over: *target*, *jobid*.

### Meaning

The following problem was encountered in an attempt made to change the resource unit or group for executing the job specified with the job ID *jobid*: The maximum number of jobs that can be concurrently accepted for the change-to resource unit or group is reached.

The value of *target* indicates the following:

- ru-accept: Number of batch jobs that can be concurrently accepted for a resource unit
- ru-interact-accept: Number of interactive jobs that can be concurrently accepted for a resource unit
- rg-accept: Number of batch jobs that can be concurrently accepted for a resource group
- rg-interact-accept: Number of interactive jobs that can be concurrently accepted for a resource group

### Action

Using the pjacl command with the --rscunit or --rscgrp option, confirm the maximum number of jobs that can be concurrently accepted for the change-to resource unit or group.
The maximum number of jobs that can be concurrently accepted for a resource unit is the "acceptablejob" or "acceptable job(interact)" value of the item "limit in rscunit (each users)," "limit in rscunit (total users in same group)," or "limit in rscunit (total all users)."
The maximum number of jobs that can be concurrently accepted for a resource group is the "acceptablejob" or "acceptable job(interact)" value of the item "limit in rscgroup (each users)," "limit in rscgroup (total users in same group)," or "limit in rscgroup (total all users)."
If necessary, ask the administrator to increase the limit of the number of concurrently acceptable jobs.

---

## [ERR.] PJM 0573 pjalter Option value is out of range: *target*, *jobid*.

### Meaning

The value of the option *target* specified for job *jobid* exceeds the valid range defined in the job ACL.

### Action

Display the job ACL settings with the pjacl command, and confirm the definition value of the specified option. If necessary, ask the administrator to change the limit value.

---

## [ERR.] PJM 0589 pjalter Not supported: *message*

### Meaning

The current version of Technical Computing Suite does not support the parameter change of the specified job.

### Action

If you want to change the specified parameter, delete the target job and submit it again specifying with suitable parameter.

---

## [ERR.] PJM 0590 pjalter PJM daemon is not present.

### Meaning

The job manager function is not working, or communication with the job manager function is disabled.

### Action

Contact the administrator. The administrator is requested to check the operating state of the job manager function or the system state.

---

## [ERR.] PJM 0591 pjalter Internal error: *details*.

### Meaning

An internal error occurred.

*details*: Detailed information for maintenance

### Action

Contact the administrator. The administrator is requested to collect investigation data according to the "Administrator's Guide for Maintenance," and then contact a Fujitsu systems engineer (SE) with the collected data together with the output message.

---

## [ERR.] PJM 0593 pjalter This command can be executed on L/JCM.

### Meaning

The command was executed on a node other than a login or job management node.

Action

Execute the command on a login or job management node.

### Information message

**[INFO] PJM 0500 pjalter Job *jobid* is accepted *opt*.**

Meaning

The change to the job (jobID: *jobid*) parameter specified by the option *opt* was accepted.

Action

No action is necessary.

# F.6 Message of the command concerning process generation

## F.6.1 pjexe command [PG]

When a problem occurs during the execution of the pjexe command, the following messages are output to the standard error output:

### Error message

**[ERR.] PLE 9100 pjexe The --vnode must be specified.**

Meaning

The --vnode option is not specified in the pjexe command.

Action

Specify the --vnode option in the pjexe command, and execute the command again.

**[ERR.] PLE 9101 pjexe command must be specified.**

Meaning

No program is specified in the pjexe command.

Action

Specify a program in the pjexe command, and execute the command again.

**[ERR.] PLE 9102 pjexe *opt* option is not supported.**

Meaning

The pjexe command does not support the specified option *opt*.

Action

Specify the correct option in the pjexe command, and execute the command again.

**[ERR.] PLE 9103 pjexe *opt* option is invalid.**

Meaning

The arguments of the option *opt* specified in the pjexe command contain an error.

Action

Confirm the arguments of the option specified in the pjexe command.

**[ERR.] PLE 9104 pjexe Duplicated a option.**

Meaning

An option specified in the pjexe command is repeated.

Action

Confirm the options specified in the pjexe command.

#### [ERR.] PLE 9105 pjexe The specified vnode(*vnodearg*) is invalid.

Meaning

The argument *vnodearg* of the --vnode option of the pjexe command contains an error.

Action

Confirm the argument value in the --vnode option of the pjexe command.

#### [ERR.] PLE 9106 pjexe The plexec cannot be executed.(CODE=*X*,*Y*,*Z*)

Meaning

The plexec command cannot be executed.

*X*, *Y*, *Z*: Internal codes

Action

Contact the administrator. The administrator is requested to collect investigation data according to the "Administrator's Guide for Maintenance," and then contact a Fujitsu systems engineer (SE) with the collected data together with the output message.

#### [ERR.] PLE 9107 pjexe cannot get memory.(CODE=*X*,*Y*,*Z*)

Meaning

There is insufficient memory.

*X*, *Y*, *Z*: Internal codes

Action

Increase the amount of memory to be allocated to the job, and execute the job again. If the same error recurs, contact the administrator. The administrator is requested to estimate the system memory again.

### Information message

#### [INFO] PLE 9149 pjexe Try 'pjexe --help' for more information.

Meaning

The usage of the pjexe command is incorrect.

Action

Confirm whether the use of the pjexe command is correct referring to the message displayed immediately before.

## F.6.2 pjrsh command

When a problem occurs during the execution of the pjrsh command, the following messages are output to the standard error output:

### Error message

#### [ERR.] PLE 9150 pjrsh IP address must be specified.

Meaning

No IP address is specified in the pjrsh command.

### Action

Specify an IP address in the pjrsh command, and execute the command again.

---

## [ERR.] PLE 9151 pjrsh command must be specified.

### Meaning

No program is specified in the pjrsh command.

### Action

Specify a program in the pjrsh command, and execute the command again.

---

## [ERR.] PLE 9152 pjrsh *opt* option is not supported.

### Meaning

The pjrsh command does not support the specified option *opt*.

### Action

Specify the correct option in the pjrsh command, and execute the command again.

---

## [ERR.] PLE 9153 pjrsh failed to initialize PSM.

### Meaning

Initialization of the system management function failed.

### Action

Contact the administrator. The administrator is requested to collect investigation data according to the "Administrator's Guide for Maintenance," and then contact a Fujitsu systems engineer (SE) with the collected data together with the output message.

---

## [ERR.] PLE 9154 pjrsh IP address (*ipaddress*) is invalid.

### Meaning

The IP address *ipaddress* specified in the pjrsh command contains an error.

### Action

Confirm the IP address specified in the pjrsh command.

---

## [ERR.] PLE 9155 pjrsh failed to finalize PSM.

### Meaning

The end processing of the system management function failed.

### Action

Contact the administrator. The administrator is requested to collect investigation data according to the "Administrator's Guide for Maintenance," and then contact a Fujitsu systems engineer (SE) with the collected data together with the output message.

---

## [ERR.] PLE 9156 pjrsh The plexec cannot be executed.(CODE=*X*,*Y*,*Z*)

### Meaning

The plexec command cannot be executed.

### Action

Contact the administrator. The administrator is requested to collect investigation data according to the "Administrator's Guide for Maintenance," and then contact a Fujitsu systems engineer (SE) with the collected data together with the output message.

---

## [ERR.] PLE 9157 pjrsh cannot get memory.(CODE=*X*,*Y*,*Z*)

#### Meaning

There is insufficient memory.

*X*, *Y*, *Z*: Internal codes

#### Action

Increase the amount of memory to be allocated to the job, and execute the job again. If the same error recurs, contact the administrator. The administrator is requested to estimate the system memory again.

### Information message

#### [INFO] PLE 9199 pjrsh Try 'pjrsh --help' for more information.

#### Meaning

The usage of the pjrsh command is incorrect.

#### Action

Confirm whether the use of the pjrsh command is correct referring to the message displayed immediately before.

## F.6.3 pjshowip command [FX100/FX10]

When a problem occurs during the execution of the pjshowip command, the following messages are output to the standard error output. The messages of the pjshowip command may seem as the message of the plestat command which is internal command of the pjshowip command.

### Error message

#### [ERR.] PLE 9250 pjshowip *opt* option is not supported.

#### Meaning

The pjshowip command does not support the specified option *opt*.

#### Action

The option cannot be specified for the pjshowip command. Execute the job again without the option of pjshowip command.

#### [ERR.] PLE 9006 plestat PLE service error occurred.(CODE=*code*)

#### Meaning

The error occurred in the parallel execution environment daemon.

#### Action

Contact the administrator. The administrator is requested to collect investigation data according to the "Administrator's Guide for Maintenance," and then contact a Fujitsu systems engineer (SE) with the collected data together with the output message.

#### [ERR.] PLE 9007 plestat Cannot get memory.(CODE=*code*)

#### Meaning

The node lacks memory resources.

#### Action

Execute the job again. If the problem persists, contact the administrator. The administrator shall check total memory of the system again.

#### [ERR.] PLE 9010 plestat PSM function error.(CODE=*code*)

#### Meaning

Calling the system management function failed.

### Action

Contact the administrator. The administrator is requested to collect investigation data according to the "Administrator's Guide for Maintenance," and then contact a Fujitsu systems engineer (SE) with the collected data together with the output message.

---

#### [ERR.] PLE 9011 plestat Cannot get job id.

### Meaning

Job ID retrieval failed.

### Action

Execute the pjshowip command in the job sctipt.

---

#### [ERR.] PLE 9012 plestat Cannot execute as sequential job.

### Meaning

The pjshowip command cannot be executed in the sequential job.

### Action

Execute the pjshowip command in the process parallel job. That is, the pjshowip command should be executed in the job which is allocated the two or more nodes, or the job which is specified the option "--mpi proc=*num*" of the pjsub command.

# F.7 Messages in job outputs

## F.7.1 plexec command

When the mpiexec command is executed in a parallel execution environment, the plexec command, which is an internal command of the parallel execution environment, is called. Therefore, if a problem occurs, messages of the plexec command may be output.

The plexec command outputs the following messages to the standard error output:

**Error message**

---

#### [ERR.] PLE 0001 plexec '--np' must be specified.

### Meaning

The --np option is not specified for the internal command 'plexec'.

### Action

Review the number of processes specified with the -c|-np|--np|-n|--n option of the mpiexec command.

---

#### [ERR.] PLE 0002 plexec PLE service error occurred.(nid=*nodeid*)(CODE=*code*)

### Meaning

Connection with the parallel execution environment daemon failed.

### Action

Contact the administrator. The administrator shall collect investigation data according to the "Administrator's Guide for Maintenance," and then contact a Fujitsu systems engineer (SE) with the collected data together with the output message.

---

#### [ERR.] PLE 0003 plexec cannot create file(*filename*).(rank=*rank*)(nid=*nodeid*)(CODE=*code*)

### Meaning

The file specified with the --of option of the mpiexec command cannot be created.

### Action

Check whether files can be created in the specified directory.

### [ERR.] PLE 0005 plexec The specified number of processes is too many.(nid=*nodeid*)(CODE=*code*)

#### Meaning

The number of processes specified with the --c|-np|--np|-n|--n option of the mpiexec command exceeded the maximum specified with the pjsub command.

#### Action

Review the number of processes specified with the -c|-np|--np|-n|--n option of the mpiexec command.

### [ERR.] PLE 0006 plexec cannot get memory.(nid=*nodeid*)(CODE=*code*)

#### Meaning

The node lacks memory resources.

#### Action

Execute the command again. If the problem persists, contact the administrator. The administrator shall check total memory of the system again.

### [ERR.] PLE 0007 plexec A system error occurred.(rank=*rank*)(nid=*nodeid*)(CODE=*code*)

#### Meaning

A system error occurred.

#### Action

Contact the administrator. The administrator shall collect investigation data according to the "Administrator's Guide for Maintenance," and then contact a Fujitsu systems engineer (SE) with the collected data together with the output message.

### [ERR.] PLE 0008 plexec must be started sequentially.(nid=*nodeid*)(CODE=*code*)

#### Meaning

The mpiexec command was started more than once.

#### Action

Do not issue the mpiexec command repeatedly.

### [ERR.] PLE 0009 plexec cannot write file(*filename*).(rank=*rank*)(nid=*nodeid*)(CODE=*code*)

#### Meaning

An error occurred in a write operation to the file specified by a --of type option of the mpiexec command.

#### Action

Check the disk free space. If there is not sufficient free space in the specified output disk, change the output disk to a disk that has sufficient free space. Then execute this command again.

If the problem persists even if there is sufficient free space in the specified output disk, contact the administrator. The administrator shall collect investigation data according to the "Administrator's Guide for Maintenance," and then contact a Fujitsu systems engineer (SE) with the collected data together with the output message.

### [ERR.] PLE 0010 plexec cannot open file(*filename*).(nid=*nodeid*)(CODE=*code*)

#### Meaning

The file specified with the --stdin option of the mpiexec command cannot be opened.

#### Action

Review the file name specified with the --stdin option of the mpiexec command.

### [ERR.] PLE 0011 plexec The program is not specified.

#### Meaning

No program is specified.

#### Action

Specify a program for the mpiexec command.

### [ERR.] PLE 0012 plexec The executing user is illegal on the node.(nid=*nodeid*)(CODE=*code*)

#### Meaning

There is an inconsistency in the user information in the node.

#### Action

Contact the administrator. The administrator shall check whether the user information is consistent between the nodes.

### [ERR.] PLE 0013 plexec The error occurred by setting the directory.(dir=*dirname*)(nid=*nodeid*) (CODE=*code*)

#### Meaning

A parallel process failed to move to the directory *dirname*.

#### Action

Contact the administrator. The administrator shall check whether the user has execute permission of the directory dirname on each node.

### [ERR.] PLE 0014 plexec The process terminated abnormally.(rank=*rank*)(nid=*nodeid*) (exitstatus=*exitstatus*)(CODE=*code1*,*code2*,*code3*)
### [ERR.] PLE The program that the user specified may be illegal or inaccessible on the node.(nid=*nodeid*)

#### Meaning

The process terminated abnormally. The program that the user specified may be illegal or inaccessible on the node.

#### Action

See job end code of the job statistical information file.
If the job ended normally, check whether the cause of the abnormal termination resided in the program specified with the mpiexec command, or whether the program can be referred on the node (node ID *nodeid*).
If the job was submitted with the option of the memory limitation (proc-data, proc-lockm, proc-stak, proc-vmem), the program might have been ended by exceeding the memory limitation.
If memory limitation was not specified, the administrator would set its default values by job ACL function. Check them by using pjacl command, and ask the administrator to change these values if necessary.
If the job ended abnormally, take an appropriate action according to job end code.
If the job statistical information file was not output, submit the job with -s or -S option of pjsub command again and investigate the cause.

### [ERR.] PLE 0015 plexec cannot execute as sequential job.

#### Meaning

The command cannot be executed as a sequential job.

#### Action

To execute an MPI program on a single node (node =1) or a single virtual node (vnode=1), be sure to specify the --mpi option.

### [ERR.] PLE 0016 plexec The program arguments is too long

#### Meaning

An argument of the specified program is too long.

### Action

Check the arguments of the program specified with the mpiexec command.

---

## [ERR.] PLE 0017 plexec The process terminated with the signal. (rank=*rank*)(nid=*nodeid*)(sig=*signal*)

### Meaning

The specified program ended with a signal.

### Action

See job end code of the job statistical information file.
If the job ended normally, check whether the cause of the signal *signal* resided in the program specified with the mpiexec command.
If the value of *signal* was 9 (SIGKILL), the program might have been terminated by memory limitation of operating system on the node. This indicates that memory resource allocated to the job was insufficient for creating processes of the program.
If the job was submitted with a specified upper limit (node-mem) on memory usage by a node or a specified memory amount (vnode-mem or mem) per virtual node, increase that amount.
If the job was submitted with no memory setting option, the administrator would set its default values by job ACL function. Check them by using pjacl command, and ask the administrator to change these values if necessary.
Note that if the value of vnode-mem or mem is 'unlimited', then the program also might have been terminated by resource competition with other job on the same node. To avoid the lack of memory, select the job execution mode policy SIMPLEX so that the job can occupy the node.
If the job ended abnormally, take an appropriate action according to job end code.
If the job statistical information file was not output, submit the job with -s or -S option of pjsub command again and investigate the cause.

---

## [ERR.] PLE 0018 plexec The program(*program filename*) cannot be executed.(nid=*nodeid*)(error=*error message*)(CODE=*code1*,*code2*,*code3*)

### Meaning

The specified program could not be executed.

### Action

Check whether the program specified with the mpiexec command can be executed.

---

## [ERR.] PLE 0019 plexec One of MPI processes was aborted. (rank=*rank*)(nid=*nodeid*)(CODE=*code1*,*code2*,*code3*)

### Meaning

One of MPI processes was aborted.

### Action

Check whether the program specified with the mpiexec command has any problems.

---

## [ERR.] PLE 0020 plexec The MPI internal error occurred.[The directory (*dirname*) specified with --tmpdir is illegal.](nid=*nodeid*)(CODE=*code1*,*code2*,*code3*)

### Meaning

mpiexec command ended by internal error.

### Action

Contact the administrator. The administrator shall collect investigation data according to the "Administrator's Guide for Maintenance," and then contact a Fujitsu systems engineer (SE) with the collected data together with the output message.

---

## [ERR.] PLE 0021 plexec The interactive job has aborted with the signal.(sig=*signum*)

### Meaning

The interactive job was forcibly terminated by the signal *signum*.

### Action

If necessary, execute the interactive job again.

---

## [ERR.] PLE 0022 plexec has timed out.

### Meaning

The internal command plexec terminated because of a timeout. This message is output if interactive job execution cannot start within a certain period after the computer resources to be allocated to the job are determined. The execution start wait time is 15 + (specified value of --sparam "wait-time=" option of pjsub command) seconds. The default value of this option is 0 seconds, so if the option is not specified, the execution start wait time is 15 seconds.

### Action

Notify the administrator that job execution failed to start. The administrator should collect investigation data according to the "Administrator's Guide for Maintenance," and then contact a Fujitsu systems engineer (SE) with the collected data together with the output message. If necessary, execute the interactive job again.

---

## [ERR.] PLE 0024 plexec There is no authority to execute the demanded operation.

### Meaning

The user does not have the privileges to perform the requested operation.

### Action

Confirm the execution privileges for the operation performed by the executed command.

---

## [ERR.] PLE 0030 plexec Duplicated a nodelist.

### Meaning

The --vcoordfile and --vnodefile options of the mpiexec command are specified at the same time.

### Action

The --vcoordfile and --vnodefile options of the mpiexec command cannot be specified at the same time. Review the options specified for the mpiexec command.

---

## [ERR.] PLE 0037 plexec failed to initialize PSM.

### Meaning

Initialization of the system management function failed.

### Action

Contact the administrator. The administrator is requested to collect investigation data according to the "Administrator's Guide for Maintenance," and then contact a Fujitsu systems engineer (SE) with the collected data together with the output message.

---

## [ERR.] PLE 0041 plexec The specified prefix(*prefix*) is invalid.

### Meaning

There is an error in the specification of the --ofprefix option of the mpiexec command.

### Action

Review the value specified with the --ofprefix option of the mpiexec command.

---

## [ERR.] PLE 0042 plexec A virtual coordinate "*vcoord*" is already used by other plexec.

### Meaning

The logical coordinate *vcoord* node is being used by a process executed by another mpiexec command.

### Action

A process cannot be created for the same node from multiple mpiexec commands. Review the contents of the file specified with the --vcoordfile option of the mpiexec command.

## [ERR.] PLE 0043 plexec The specified virtual coordinate "*vcoord*" is invalid.

### Meaning

The specified logical coordinate *vcoord* is invalid.
The specified logical coordinate *vcoord* may be out of the range of the nodes allocated to the job or belong to a dimension incompatible with the node shape.

### Action

Review the contents of the file specified with the --vcoordfile option of the mpiexec command.

## [ERR.] PLE 0044 plexec The file "*filename*" specified with --vcoordfile does not exist.

### Meaning

The file *filename* specified with the --vcoordfile option of the mpiexec command does not exist.

### Action

Check the file name specified with the --vcoordfile option of the mpiexec command.

## [ERR.] PLE 0045 plexec A virtual node "*vnode*" is already used by other plexec.

### Meaning

The virtual node *vnode* is being used by a process executed by another mpiexec command.

### Action

A process cannot be created for the same virtual node from multiple mpiexec commands. Review the contents of the file specified with the --vnodefile option of the mpiexec command.

## [ERR.] PLE 0046 plexec The specified virtual node "*vnode*" is invalid.

### Meaning

The specified virtual node *vnode* is invalid.
The specified virtual node *vnode* may be out of the range of virtual nodes allocated to the job.

### Action

Review the contents of the file specified with the --vnodefile option of the mpiexec command.

## [ERR.] PLE 0047 plexec The file "*filename*" specified with --vnodefile does not exist.

### Meaning

The file *filename* specified with the --vnodefile option of the mpiexec command does not exist.

### Action

Review the file name specified with the --vnodefile option of the mpiexec command.

## [ERR.] PLE 0048 plexec The number of processes exceed the number of virtual coordinates specified by --vcoordfile.

### Meaning

The number of processes specified with the -c|-np|--np|-n|--n option of the mpiexec command exceeds the number of lines of the file specified with the --vcoordfile option.

### Action

Make the number of processes specified with the -c|-np|--np|-n|--n option of the mpiexec command equal to or less than the number of lines of the file specified with the --vcoordfile option.

## [ERR.] PLE 0049 plexec The number of processes exceed the number of virtual nodes specified by --vnodefile.

### Meaning

The number of processes specified with the -c|-np|--np|-n|--n option of the mpiexec command exceeds the number of lines of the file specified with the --vnodefile option.

### Action

Make the number of processes specified with the -c|-np|--np|-n|--n option of the mpiexec command equal to or less than the number of lines of the file specified with the --vnodefile option.

## [ERR.] PLE 0050 plexec cannot be executed any further.

### Meaning

The number of times the mpiexec command was executed more than once exceeded the maximum value 128.

### Action

Review the job contents, and reduce the number of times the mpiexec command is executed more than once to a value that is equal to or less than the maximum value allowed by the job operation software.

## [ERR.] PLE 0051 plexec The usernames and uids on all nodes must be defined identically.

### Meaning

The user name and the user information regarding the user ID are inconsistent.

### Action

The system administrator should confirm that the user information is consistent on each node.

## [ERR.] PLE 0053 plexec Incompatible version of MPI library.

### Meaning

The combination of the executed job and version level of the specified MPI library cannot be executed.

### Action

Use the latest version of the MPI library.

## [ERR.] PLE 0054 plexec The number of processes exceed the limit on virtual coordinate *vcoord*.

### Meaning

The number of processes to be generated on the node at the logical coordinate *vcoord* exceeds the limit on the number of processes that can be generated per the logical coordinate.
The limit on the number of processes that can be generated per the logical coordinate is the value specified in the --mpi "proc=" option or --mpi "max-proc-per-node=" option.

### Action

Make sure that the number of processes to be generated on the specified logical coordinate *vcoord* does not exceed the limit on the number of processes that can be generated per the logical coordinate.

## [ERR.] PLE 0055 plexec The number of processes exceed the limit on virtual node *vnode*.

### Meaning

Multiple processes cannot be generated on the virtual node *vnode* at the same time.

### Action

Only one process can be generated on the virtual node. Review the number of processes to be generated on the virtual node.

## [ERR.] PLE 0056 plexec The number of unused virtual coordinates are not sufficient.

### Meaning

The number of unused nodes is too small to generate processes.

### Action

Confirm each of the following specifications is correct: the number of nodes to be allocated to the job, and the limit on the number of processes that can be generated per node.
If only the number of CPU cores assigned to each process is specified in the --vcoordfile option of the mpiexec command, the specification order may be the cause of this error.
The job operation software determines the nodes where CPU cores can be reserved depending on the sequence of specified numbers of CPU cores. Some specified sequences may have a node shortage due to an attempt to reserve an unnecessarily large number of nodes.
For example, suppose a node group has 16 CPU cores per node. Then, a sequence of 15, 1, 15, and 1 CPU core for 4 processes requires 2 nodes in total to be allocated. In contrast, a sequence of 15, 15, 1, and 1 CPU core requires 3 nodes in total to be allocated.
To avoid this problem, explicitly specify logical node coordinates too so that the processes are placed appropriately.

## [ERR.] PLE 0058 plexec The number of CPU cores on a virtual coordinate *vcoord* are not sufficient.

### Meaning

The node at the logical coordinate *vcoord* does not have enough CPU cores to allocate.

### Action

Make the total number of CPU cores assigned to processes generated on a node equal to or less than the number of CPU cores installed on the node.

## [ERR.] PLE 0060 plexec Format error in --vcoordfile.

### Meaning

The file specified in the --vcoordfile option of the mpiexec command contains a format error.

### Action

Review the contents of the file specified in the --vcoordfile option of the mpiexec command.
For the location and details of the error, see the PLE 0093 information message output at the same time.

## [ERR.] PLE 0061 plexec Format error in --vnodefile.

### Meaning

The file specified in the --vnodefile option of the mpiexec command contains a format error.

### Action

Review the contents of the file specified in the --vnodefile option of the mpiexec command.
For the location and details of the error, see the PLE 0093 information message output at the same time.

## [ERR.] PLE 0062 plexec Cannot set numanode_assign_policy.

**Meaning**

Core allocation to the process cannot be implemented under the policy for allocating cores to NUMA nodes.

**Action**

Review the policy for allocating cores to NUMA nodes, that is specified in the process.
For details on the core allocation policy, see the Technical Computing Language manual "MPI User's Guide."

## [ERR.] PLE 0905 plexec cannot get the job information. (rank=*rank*)(nid=*nodeid*)(CODE=*code*)

### Meaning

Job information retrieval failed.

### Action

Execute program in the job script. If the program is executed in the job script, contact the administrator. The administrator shall collect investigation data according to the "Administrator's Guide for Maintenance," and then contact a Fujitsu systems engineer (SE) with the collected data together with the output message.

## Warning message

### [WARN] PLE 0601 plexec Cannot set "memory_allocation_policy".(policy=*policy*)(rank=*rank*)(nid=*nid*) (errorno=*error_number*)

### Meaning

The NUMA memory allocation policy cannot be set for the process. The process will execute with the default NUMA memory allocation policy.

### Action

Review the NUMA memory allocation policy that is specified in the process.
For details on the NUMA memory allocation policy, see the Technical Computing Language manual "MPI User's Guide."

### [WARN] PLE 0602 plexec "numanode_assign_policy" is simplex, but the job is not simplex.(rank=*rank*) (nid=*nid*)

### Meaning

For the process, simplex (NUMA node exclusive) was specified as the policy for allocating cores to the NUMA node, but the job is not a node-exclusive job, and it may not be possible to make the NUMA node exclusive.

### Action

Review the job submission method.
For details on the policy for allocating cores to the NUMA node, see the Technical Computing Language manual "MPI User's Guide."

### [WARN] PLE 0603 plexec Invalid "numanode_assign_policy".(policy=*policy*)(rank=*rank*)(nid=*nid*) (error=*error message*)

### Meaning

An invalid value was specified for the environment variable OMPI_MCA_plm_ple_numanode_assign_policy, under the policy for allocating cores to processes. The process will execute with the default core allocation policy.

### Action

Review the value of the environment variable OMPI_MCA_plm_ple_numanode_assign_policy.
For details on the core allocation policy and the environment variable OMPI_MCA_plm_ple_numanode_assign_policy, see the Technical Computing Language manual "MPI User's Guide."

## Information message

### [INFO] PLE 0090 plexec The process terminated by '~.'

### Meaning

The running interactive job ended.

### Action

No action is necessary.

### [INFO] PLE 0091 plexec ~^Z[suspend tty]

### Meaning

A running job has been suspended.

### Action

No action is necessary.

---

**[INFO] PLE 0092 plexec Supported escape characters:**
**~. : terminate processes**
**~^Z : suspend tty**
**~? : display the usage of escape characters**
**~~ : send '~'**
**Note that escape characters are only effective at the beginning of line.**

### Meaning

The following escape characters are supported for interactive jobs:

~. : Exits the job shell in execution and ends the interactive job.

~^Z : Suspend the interactive job.

~? : Lists escape characters.

~~ : Transfers the tilde symbol for input for the interactive job.

These characters are valid only when input at the beginning of a line.

### Action

No action is necessary.

---

**[INFO] PLE 0093 plexec Format error: *filename*:*line*:*message***

### Meaning

If the file *filename* contains a format error, this message is output together with an error message to show the location (line) and details (*message*) of the error.
If the file *filename* contains multiple errors, this message is output for each error.

### Action

See the detailed message *message* and the error message that was output at the same time, and correct line line in the file *filename*.

# F.8 Command List

The following tables are lists of commands described in this manual. All end-users and the administrators can use these commands. However, the administrator restricts the execution of users by setting the job ACL function.

Table F.2 For referencing to the job information

| Command name | Scene | Available node |
|---|---|---|
| pjstat | To confirm the state of submitted jobs | Login node<br>Job management node<br>Control node |
| pjacl | To confirm the contents of the job ACL database | Login node<br>Job management node |
| pjshowrsc | To confirm the entire amount or the usage status of activated computer resources | Login node<br>Job management node<br>Control node |
| pjstgchk<br>[FX10] | To confirm the format of the file staging command line | Login node<br>Job management node |

| Command name | Scene | Available node |
|---|---|---|
| pjget<br>[FX10] | To retrieve the output files of running jobs | Login node<br>Job management node |
| pjlist<br>[FX10] | To confirm the output file lists of running jobs | Login node<br>Job management node |
| pjcat | To confirm the contents of the standard output, standard error output, or job scripts of running jobs | Login node<br>Job management node |

Table F.3 For operation of the job

| Command name | Scene | Available node |
|---|---|---|
| pjsub | To submit jobs | Login node<br>Job management node |
| pjdel | To delete jobs | Login node<br>Job management node |
| pjhold | To hold submitted jobs | Login node<br>Job management node |
| pjrls | To cancel the job hold | Login node<br>Job management node |
| pjwait | To wait for a job to complete | Login node<br>Job management node |
| pjsig | To send signals to running jobs | Login node<br>Job management node |
| pjalter | To change the parameter of the job | Login node<br>Job management node |

Table F.4 For creating of the process in the job

| Command name | Scene | Available node |
|---|---|---|
| pjexe [PG] | To execute the sequential program all together on the plural nodes | Compute node |
| pjrsh | To execute MPI programs using the pjrsh command instead of rsh/ssh command when the programs are created by not using Technical Computing Language. | Compute node |
| pjshowip<br>[FX100/FX10] | To output the list of the node allocated to the job. The pjshowip is used to execute the MPI program specifying the host name. | Compute node |

# Glossary

This glossary describes the key terms that appear in this manual.

### Batch job

A batch job is a job which is executed un-interactively.

### Bulk job

A bulk job consists of multiple instances of the same job submitted at the same time for execution. A bulk job is used to execute individual jobs with different input parameters. Each job is called a sub job.

### Cluster

A cluster is the unit of a group of nodes sharing job operations and files. The job operation software has a compute cluster for executing jobs and a storage cluster for sharing files.

### Computer resource

Computer resources are the required resources for executing applications in jobs. These resources include CPUs, memory, disk resources, networks, CPU time, etc. Sometimes, they are simply called resources.

### Emergency job [FX100]

An emergency job is a job that must be executed with the highest priority during operation. To execute an emergency job, resources may be released by aborting a job already being executed or by a job swap.

### Execution mode policy [PG]

An execution mode policy covers the method of occupation of nodes by jobs. The policy can specify whether the nodes are occupied by one job or shared with other jobs.

### Fair share function

The fair share function is a function for determining the priorities of users and groups according to job execution results so that users and groups each have fair use of the system. After a user or group executes jobs frequently or a large-scale job, the user's or group's subsequent jobs have a lower execution priority.

### Global file system

The global file system is a large-scale file system shared between the nodes in the system. The related software FEFS configures the global file system.

### Interactive job

An interactive job is a job in which the interactive operation is possible for the program.

### Job

A job is the unit of execution of a program created by a user. The job that execution is requested (submitted) to the Job Operation Software is allocated the computer resources, and execution is controlled by the job scheduler function and the job resource management function.

### Job ACL function

The job ACL function sets the standard and maximum values of the resource limit values for each group and user. Jobs submitted without specified values are set with the standard values of this job ACL.

### Job parameter

A job parameter is information specified at the job submission time. Examples include a limit value, priority, resource unit for execution, and resource group for execution. Some attributes can be changed before job execution.

### Job hold

Job hold is an operation that keeps the job state from changing. If the job is in the execution wait state, the job is not executed until the hold is canceled. If the job is running, the job is aborted and executed again when the hold is canceled.

### Job model

A job model is a classification by job structure. Examples include normal jobs, bulk jobs, step jobs, master worker jobs, and workflow jobs.

### Job script

A job script is a shell script with the written program execution procedure. The job operation software executes a job based on this job script.

### Job selection policy

The job selection policy is evaluation criteria for determining the job execution priority. It is set by the administrator.

### Job statistical management function

The job statistical management function outputs statistical information such the start and end times, CPU time, memory utilization, etc. You can use this statistical information to bill users and monitor the use status.

### Job submission

A job submission is a job execution request that is issued to the job operation management function.

### Job swap function [FX100]

The job swap function stops a job being executed while maintaining its status, temporarily releases CPU resources and memory resources and, when it restarts the job, resumes the job execution with the job status in effect when the job was stopped.

### Job type

A job type is a classification by job execution format. Examples include batch jobs and interactive jobs.

### Local file system

The local file system is a file system accessed only by an individual compute node, whereas the global file system is a shared file system.

### Mesh mode [FX100/FX10]

Mesh mode is one method of allocating nodes for node-exclusive jobs. The smallest unit of node allocation for jobs is one node. Allocated nodes are mutually adjacent in the Tofu coordinate space.

### MPMD (Multiple Program Multiple Data)

MPMD is a programming model for parallelization. It uses multiple different programs for process sharing.

### Multinode job

A multinode job is a job that requires multiple nodes for execution.

### Node-exclusive job [FX100/FX10]

A node-exclusive job is a job executed by monopolizing a node allocated to it.

### Node selection method [PG]

A node selection method is the method of selecting the nodes to allocate to jobs. The method specifies which of the following has priority: unused nodes or nodes already in use.

### Node-sharing job [FX100/FX10]

A node-sharing job is a job executed with a virtual node. This involves the division of physical nodes among multiple jobs.

### Non-contiguous mode [FX100/FX10]

Non-contiguous mode is one method of allocating nodes for node-exclusive jobs. The smallest unit of node allocation for jobs is one node. Nodes are allocated so that they are mutually adjacent in the Tofu coordinate space as much as possible.

### NUMA node

In NUMA (Non-Uniform Memory Access) architecture, in which the cost of accessing memory shared by multiple CPU cores is not uniform, this is the CPU core group in which the memory access cost is the same.

### NUMA allocation policy

NUMA allocation policy consists of rules on the allocation of jobs to NUMA nodes. There is a method for allocating a job so that it fits in a NUMA node, and a method for allocating a job so that it spans NUMA nodes.

### Parallel job

A parallel job is a job for executing a parallel processing program using multiple threads and/or multiple processes.

### Priority control of allocated nodes [PG]

Priority control of allocated nodes is a method of setting priority on each node and allocating nodes to jobs according to that priority.

### Rank

A rank is a concept equivalent to a process ID in an MPI program.

### Rank number

A rank number is a number that is set in the order of rank generation.

### Rank number directory [FX10]

A rank number directory is an execution directory assigned to each rank. It is used to separate directories in the execution of each rank in order to prevent IO conflicts between ranks.

### Sequential job

A sequential job is a job that executes a single-thread or single-process program.

### Single node job

A single node job is a job that requires only one node for execution.

### SPMD (Single Program Multiple Data)

SPMD is a programming model for parallelism. In contrast to the MPMD model, SPMD uses multiple identical programs for process sharing.

### Stage in [FX10]

In staging, stage in is the transfer of the required files (program and data) from the global file system to the local file system on a compute node when a job is executed.

### Stage out [FX10]

In staging, stage out is the file transfer of the results output on the local file system to the global file system when a job ends.

### Staging [FX10]

Staging is the automatic transfer of files between the global file system and the local file system before and after job execution. This mechanism prevents IO conflicts on the global file system during job execution by executing the job on the local file system of a compute node.

### Step job

A step job is a group of jobs that have an execution order or dependency relationship. Each job is called a sub job.

### Sub job

Sub job is the processing unit of job in the bulk job and step job.

### Swap in [FX100]

Swap in involves allocating CPU resources and memory resources to a swapped out job and restarting the job.

### Swap out [FX100]

In the swap-out function, a swap out involves stopping a job being executed while maintaining its status, and temporarily releasing the CPU resources and memory resources.

### Swap period [FX100]

The swap period is the time period from the start of a swap out to the completion of swap in.

### Torus mode [FX100/FX10]

A method of allocating nodes to a node-exclusive job. The smallest node allocation unit for jobs is the Tofu unit (12 nodes). Allocated nodes are mutually adjacent in the Tofu coordinate space.

### Virtual node

A virtual node is the unit of resource allocation. It is the concept of grouping CPU cores and memory together.

### Virtual node placement policy [PG]

A virtual node placement policy is the concept of how virtual nodes are placed as opposed to physical nodes.

### Workflow job

A workflow job is a group of jobs with a defined execution flow (conditional branching and repetition). Specifically, it indicates not a job but a job control method.