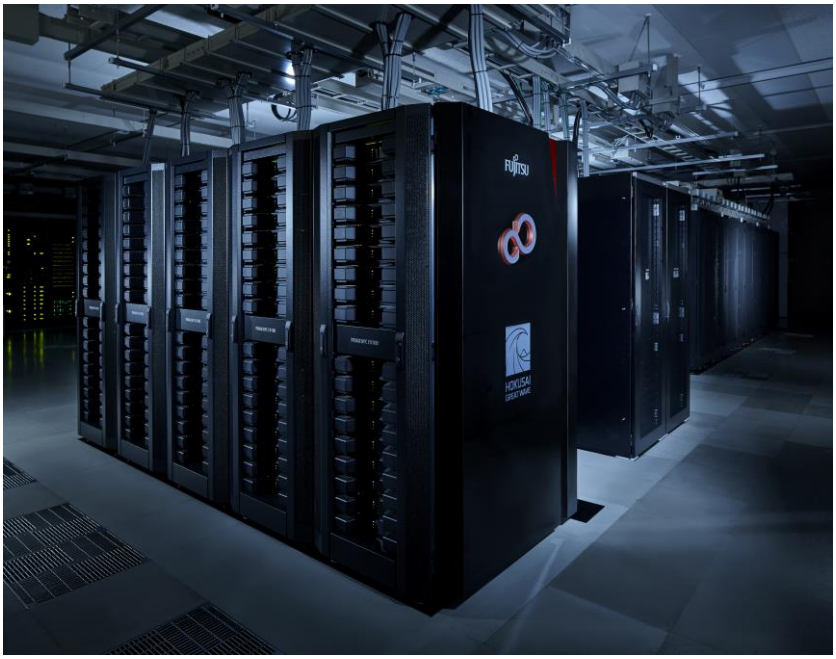
HOKUSAI GreatWave User's Guide



Version 1.6

April 1, 2016

Advanced Center for Computing and Communication,

RIKEN

HOKUSAI
GREAT WAVE

Revision History

Ver.	Issued on	Chapter	Reason for revision
1.0	Apr. 01, 2015	-	First edition
1.1	Apr. 03, 2015	0 2.2.2.1	Append Client Certificate Installation(on Ubuntu) Modify key generation (on Windows)
1.2	Apr. 28, 2015	2.2.1 2.5 5.4.5 5.4.7 5.7.5 5.7.6 7	Append the notice about installing the client certificate into JAVA environment Append the procedure of the SSH Agent Forwarding on the Windows environment Append the notice if the login shell is not bash Append the description of the mpirun option Append the option that shows the utilization of COREs Append the option that shows the priority order of projects Modify section title, and add the import the client certificate into the JAVA environment
1.3	Jun. 1, 2015	4.2 5.2.1.1 5.2.2 5.3.2 5.5.2.7	Append about how to use CUDA Append resource unit limitation of each project Append resource group for using GPU Append resource options for using GPU Append example script for GPU job
1.4	Aug. 05, 2015	5.2.2.3 5.5.2.8 5.7.7 5.10.2 5.10.3	Modify configuration of GPU resource group Append example script for multi node GPU job Append how to check resource limit of jobs (pjstat -x) Append available environment variables in jobs Append the notandum of processing mode of releasing memory on the Massively Parallel Computer (XOS_MMM_L_ARENA_FREE setting)
1.5	Sep. 14, 2015	1.5	Modify "Supercomputer System Usage Policy" URL
1.6	Apr. 01, 2016	1.5.1 1.6 2.2.5 3.1.3 5.2.2 5.4.2	Modify result of listcpu command Append description of priority control Append how to uninstall client certificate Append application form for using storage area Modify resource group settings (Small size(<=12cores) long term job(72hours) is available on ACSG) Append step job submission with multiple scripts

Table of Contents

INTRODUCTION

This User's Guide explains how to use the supercomputer system HOKUSAI GreatWave introduced by RIKEN. All users who use this system are strongly recommended to read this document, as this is helpful to gain better understanding of the system.

The content of this document is subject to change as required. The latest version of this document is available from the following User Portal:

https://hokusai.riken.jp/

The shell scripts and other contents described in this document are stored in the front end servers. For those contents, refer to the sub-directories under the following directory:

/usr/local/example

The User Portal and mailing lists are used for public announcement of the system's operation status.

In addition, several seminars on the use of this system are held several times per year. The users can find the related information in the website of Advanced Center for Computing and Communication.

http://accc.riken.jp/en/

If you have any questions about how to use the HOKUSAI GreatWave system or need for further assistance, you can send messages in an email to:

hpc@riken.jp

1. HOKUSAI GreatWave System

1.1 System Overview

The HOKUSAI GreatWave system consists of the following key components:

- Massively Parallel Computer
- Application Computing Server including ACS with Large Memory and ACS with GPU
- Front end servers that provide the users with the application interface for the system
- Two types of storages with different purposes, one of which is the Online Storage and the other of which is the Hierarchical Storage.

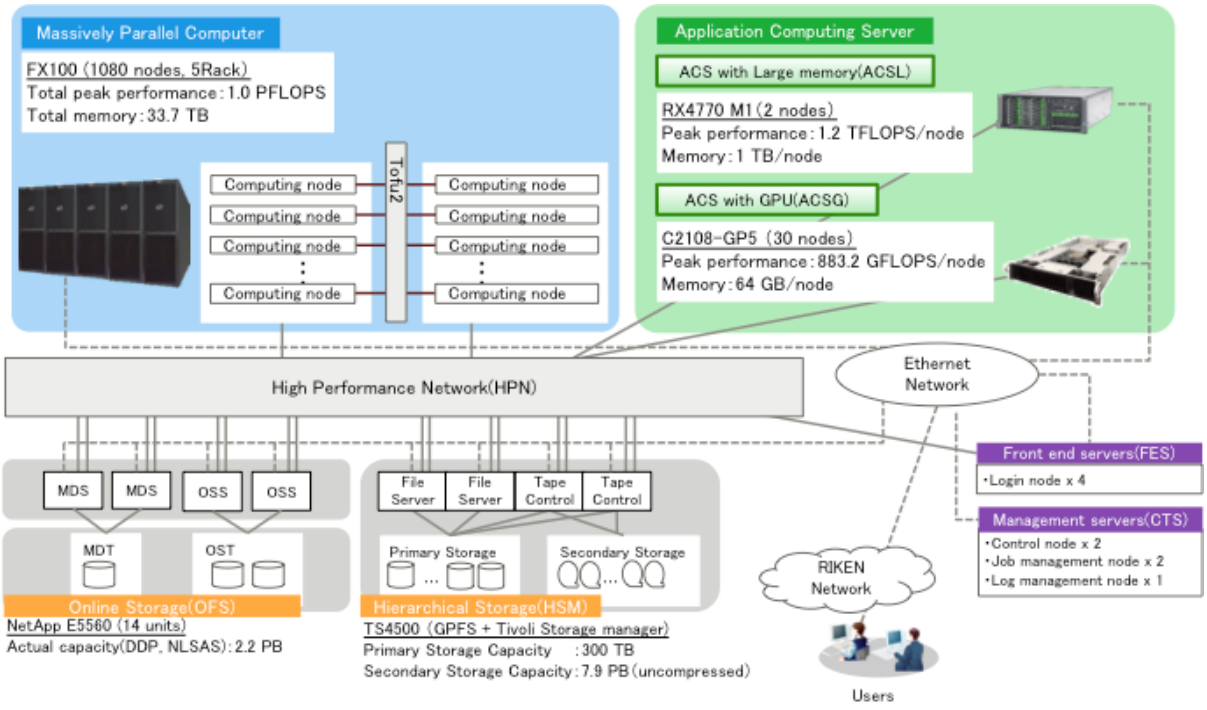


Figure 1-1 System diagram

The Massively Parallel Computer (MPC) comprises FUJITSU Supercomputer PRIMEHPC FX100. FX100, with high performance processors SPARC64 XIfx and high performance memory systems, provides a theoretical peak performance of 1 TFLOPS (double precision) and the high memory bandwidth of 480 GB/s per one node with 32 cores/CPUs. The Massively Parallel Computer of 1,080 nodes provides the a theoretical peak performance of 1 PFLOPS and a total memory capacity of 33.7 TB

and uses the 6D mesh/torus network (Torus Fusion Interconnect 2*[1]) to tightly connect each node with 12.5 GB/s high-speed link.

The ACS with Large memory (ACSL) comprises two nodes of PRIMERGY RX4770 M1. Each node provides a theoretical peak performance of 1.2 TFLOPS and a memory capacity of 1 TB. The ACS with GPU (ACSG) consists of thirty nodes of SGI C2108-GP5. Each node provides a theoretical peak performance of 883.2 GFLOPS and a memory capacity of 64 GB. Four NVIDIA Tesla K20X accelerators will be installed on each node of ACS with GPU. The InfiniBand FDR of 6.8 GB/s is used to connect each node to enable high performance communication and file sharing.

The storage environment consists of the Online Storage (OFS) and the Hierarchical Storage (HSM).

The Online Storage (OFS) is a high bandwidth online file system used for the users' home directories, the shared directories for projects and so on, and can be accessed from the Massively Parallel Computer, the Application Computing Server, and the front end servers. The total capacity is 2.2 PB.

The Hierarchical Storage (HSM) consists of the primary storage (cache disks) of 300 TB and the secondary storage (tape library devices) of 7.9 PB (uncompressed) and is the file system used to store large volumes of data files that should be retained for a long term. The users can read or write data to the tapes without manipulating the tape library devices.

You can access the HOKUSAI GreatWave system using ssh/scp for login/file transfer, or using HTTPS for the User Portal and FUJITSU Software Development Tools (Development Tools). On the front end servers, you can mainly do the following:

- create and edit programs
- compile and link programs
- manage batch jobs and launch interactive jobs
- tune and debug programs

*1 Tofu Fusion Interconnect 2 is Fujitsu's proprietary high speed interconnect.

1.2 Hardware Overview

1.2.1 Massively Parallel Computer (MPC)

- Computing performance
  CPU: SPARC64™XIfx (1.975 GHz) 1,080 units (1,080 CPUs, 34,560 cores)
  Theoretical peak performance: 1.092PFLOPS (1.975 GHz x 16 floating-point operations x 32 cores x 1,080 CPUs)
- Memory
  Memory capacity: 33.7 TB (32 GB x 1,080 units)
  Memory bandwidth: 480 GB/s/CPU
  Memory bandwidth/FLOP: 0.47 Byte/FLOP
- Interconnect (Tofu Interconnect 2)
  6D mesh/torus
  Theoretical link throughput: 12.5 GB/s x 2 (bidirectional)

1.2.2 Application Computing Server (ACS)

The Application Computing Server (ACS) consists of the ACS with Large memory (ACSL) and the ACS with GPU (ACSG).

1.2.2.1 ACS with Large Memory (ACSL)

- Computing performance
  CPU: Intel Xeon E7-4880v2 (2.50 GHz) 2units (8 CPUs, 120 cores)
  Theoretical peak performance: 2.4 TFLOPS (2.5 GHz x 8 floating-point operations x 15 cores x 8 CPUs)
- Memory
  Memory capacity: 2 TB (1TB x 2 units)
  Memory bandwidth: 85.3 GB/s/CPU
  Memory bandwidth/FLOP: 0.28 Byte/FLOP
- Local disk
  Disk capacity: 3.6 TB ((300 GB x 2 + 1.2 TB) x 2 units)
- Interconnect
  FDR InfiniBand
  Theoretical link throughput: 6.8 GB/s x 2 paths x 2 (bidirectional)

1.2.2.2 **ACS with GPU (ACSG)**

- Computing performance
  CPU: Intel Xeon E5-2670 v3 (2.30GHz) 30 units (60 CPUs, 720 cores)
  Theoretical peak performance: 26.4 TFLOPS (2.3 GHz x 16 floating-point operations x 12 cores x 60 CPUs)
- Memory
  Memory capacity: 1.8 TB (64 GB x 30 units)
  Memory bandwidth: 68.2 GB/s/CPU
  Memory bandwidth/FLOP: 0.15 Byte/FLOP
- Local disk
  Disk capacity: 18 TB ((300 GB x 2) x 30 units)
- Interconnect
  FDR InfiniBand
  Theoretical link throughput: 6.8 GB/s x 2 (bidirectional)
- Accelerator
  NVIDIA Tesla K20X x 4 devices/node

1.3 **Software Overview**

The latest information about the applications, libraries and so on, available for the HOKUSAI GreatWave system, will be published in the following User Portal:

https://hokusai.riken.jp/

The software available on the HOKUSAI GreatWave system are listed as follows:

Table 1-1 Software overview

Category	Massively Parallel Computer (MPC)	Application Computing Server (ACS)	Front End Servers
OS	XTCOS (OS for FX100) (Linux kernel version 2.6)	Red Hat Enterprise Linux 6 (Linux kernel version 2.6)	Red Hat Enterprise Linux 6 (Linux kernel version 2.6)
Compiler	Technical Computing Language (Fujitsu)	Intel Parallel Studio XE Composer Edition	Technical Computing Language (Fujitsu) Intel Parallel Studio XE Composer Edition
Library	Technical Computing Language (Fujitsu) - BLAS, LAPACK, ScaLAPACK, MPI, SSLII, C-SSLII, SSLII/MPI, Fast Basic Operations Library for Quadruple Precision	Intel MKL - BLAS, LAPACK, ScaLAPACK, Intel MPI	Technical Computing Language (Fujitsu) Intel MKL Intel MPI IMSL Fortran Numerical Library
Application	Gaussian	Gaussian, Amber, ADF, ANSYS (multiphysics) GOLD/Hermes, MATLAB, Q-Chem	GaussView, ANSYS (preppost)

You can develop programs on the front end servers for both the Massively Parallel Computer (SPARC) and the Application Computing Server (Intel) although these two systems have different architectures.

1.4 **Service Time Period**

The services of HOKUSAI GreatWave system are regularly available for 24 hours except for the time periods of the periodical maintenance, the emergency maintenance, and the equipment maintenance such as the air conditioner maintenance and the power-supply facility maintenance. The availability of HOKUSAI GreatWave system is announced via the User Portal or the mailing list.

1.5 Usage Category

We set up the following types of user categories. Please make an application appropriate for your research thesis.

- General Use
- Quick Use

For more detailed information, visit the following URL to see the "Supercomputer System Usage Policy"

http://accc.riken.jp/en/supercom/application/usage-policy/

1.5.1 Allocated Computational Resources (core time)

Allocated computational core time depends on usage category. You can check the project accounting summary such as project ID/project name, allocated computational core time, and used computational core time, expiration date of allocated computational core time by the *listcpu* command.

No new jobs can be submitted when remained computational core time runs out.

```
[username@greatwave1 ~]$ listcpu
[Q15000] Study of parallel programs
        Limit       Used       Used(%)  Expiry date --------------------
---------------------------------------
gwmpc   100,000.0   10,000.0     10.0%  2016-03-31
gwacsg   50,000.0   10,000.0     20.0%  2016-03-31
gwacsl   10,000.0   10,000.0    100.0%  2016-03-31

userA            -    9,000.0         -  -
userB            -    1,000.0         -  -
  :
```

Table 1-2 listcpu information

Item	Description
Limit	Allocated computational core time (unit: core・hour)
Used	Used computational core time (unit: core・hour)
Used (%)	Used computational core time / Allocated computational core time
Expiry date	Expiration date of the allocated computational core time

1.5.2 Special Use

For large-scale jobs or exclusive use, priority use of the computing resources (48 hours elapsed time), will be given to researchers of approved projects. The special use is available within the allocated computational time for each project.
The special use is announced by the mailing list. Research proposals need to have sufficient necessity for the special use of the system.

1.6 Job exection order

In this system, the job execution order is decided by the priority of all jobs. The priority is evaluated by the following items.

Table 1-3 listcpu information

Evaluation order	Evaluatino item	Overview
1	Fairshare value of project	Value to determine the priority of project. Calculate for each project based on the recovery rate and job execution history.
2	Fairshare value of user within a project	Value to determine the priority of users in same project.
3	Job priority	Priority of the user own job.
4	Job submittion time	Execute by the submission order.

Because the evaluation result with smaller "Evaluation order" take priority, the job which belongs to the project with larger "Fairshare value of project" gets preference over the jobs which are submitted earlier. About "fairshare value" is described in the next section.

1.6.1 Fairshare function

In this system, job execution order is decided by the fairshare value of each project and each user within a project. Fairshare value is changed continuously by starting job or recovering with time. Jobs are preferentially executed in the order of fairshare value of project.

The following figure indicates the behavior of fairshare value.

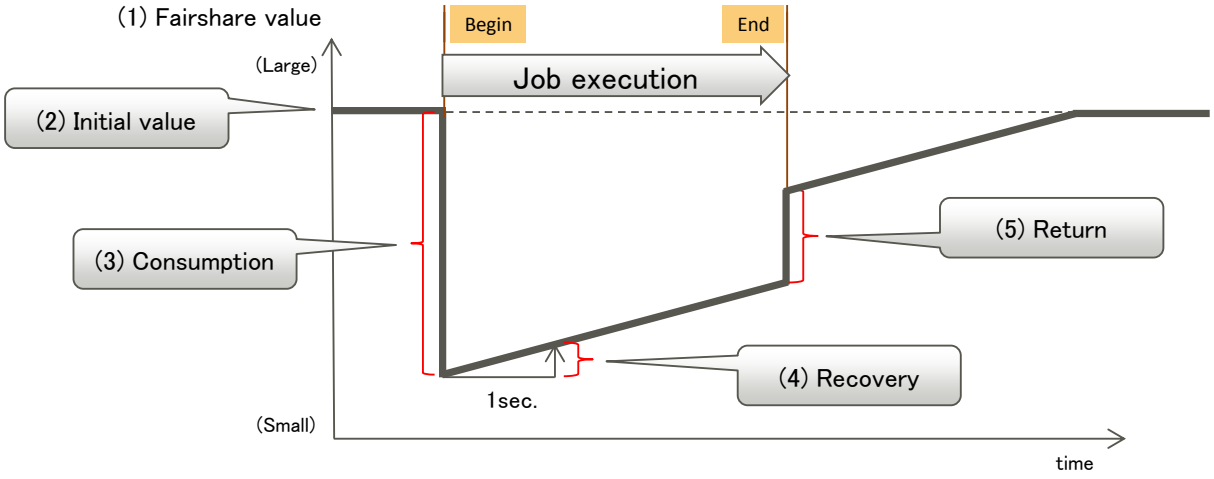


Figure 1-2 Behavior of fairshare value

Table 1-4 Term used in fairshare value

	Term	Meaning	Value in this system
(1)	Fairshare value	The priority of project	
(2)	Initial value	Initial and maximum value of fairshare value	1trillion
(3)	Consumption	Decrease from fairshare value at starting job	(Required number of cores) * (Elapse limit) [s]
(4)	Recovery	Add to fairshare value per second	Depending on the approved computation time of project
(5)	Return	Add value when the job is finished before reached to elapse limit	(Required number of cores) * (Elapse limit - Elapse time) [s]

The priority rank of project can be checked by *pjstat -p* command. Fairshare value is managed inside of system, so users cannot check and change them.

1.6.2 Backfill function

In this system, the backfill function is available to effectively use computing resource. Some idle computing resource will arise during the resource allocation process by previously described fairshare function. The job which can fill such idle resource will be run at an early time in spite of existence of other higher priority jobs. Backfilled jobs can be checked by *pjstat* command (its "START_DATE" is marked by "<").

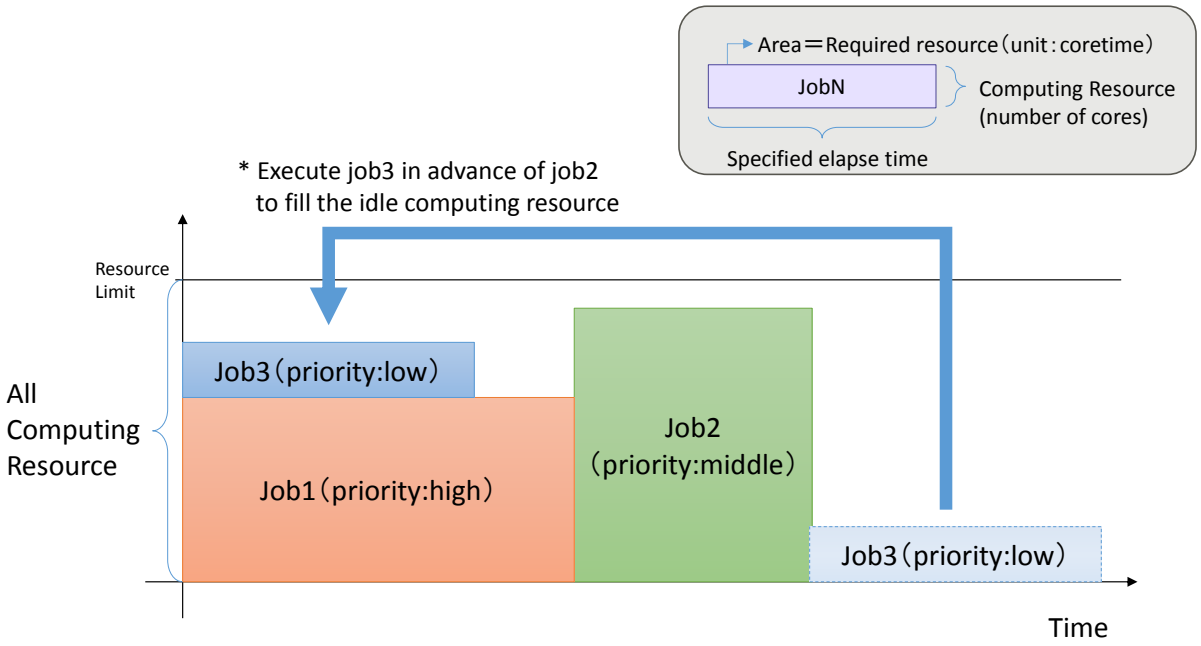


Figure 1-3 Behavior of backfill function

2. Login to HOKUSAI GreatWave System

2.1 Login Flow

The login flow for the HOKUSAI GreatWave system from account application to login as follows:

When the account is issued, the e-mail with the client certificate attachment is sent. After installing the client certificate on your PC, access the User Portal. You can login to the front end servers via SSH by registering your ssh public key on the User Portal.

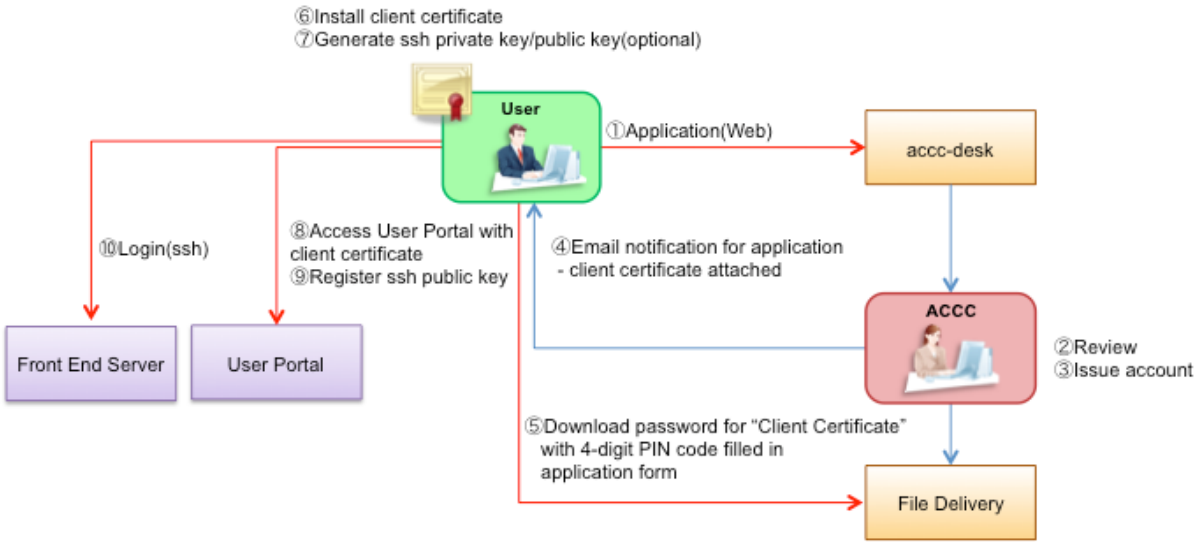


Figure 2-1 Login flow

2.2 Initial Settings

When accessing the system for the first time, login to the User Portal and make sure to do the following initial settings:

- ✓ 2.2.1 Install Client Certificate
- ✓ 2.2.2 Key Generation
- ✓ 2.2.3 Register ssh public key

2.2.1 Install Client Certificate

This section explains how to install the client certificate on the Windows, MAC or Ubuntu. You might need to install it into the browser depending on your browser. When you upload or download files using the User Portal, you need to install the client certificate into the JAVA environment. For installing the client certificate into the JAVA environment, refer to "7.1 Import Client Certificate into JAVA Environment".

2.2.1.1 Install Client Certificate (Windows Environment)

Install the client certificate ACCC sent you by e-mail.

1. Double-click the client certificate provided by ACCC.
2. The Certificate Import Wizard starts. Click "Next" button.

Figure 2-2 First screen of "Certificate Import Wizard"

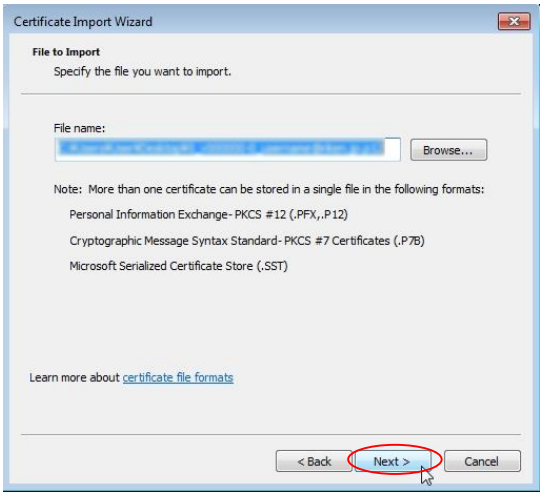
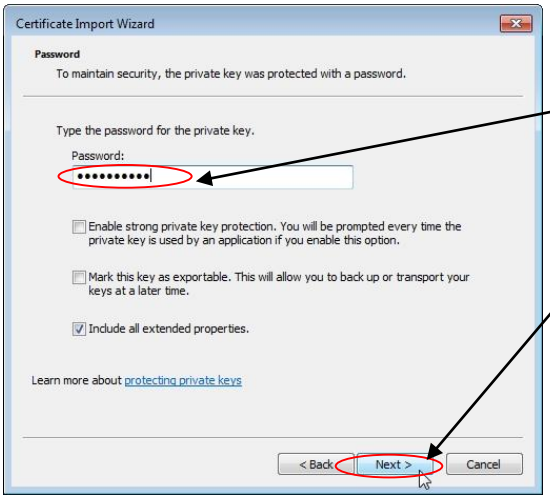


Figure 2-3 Second screen of "Certificate Import Wizard"



1. Enter the password for "Client Certificate" issued by ACCC.
2. Click "Next" button.

Figure 2-4 Third screen of "Certificate Import Wizard"

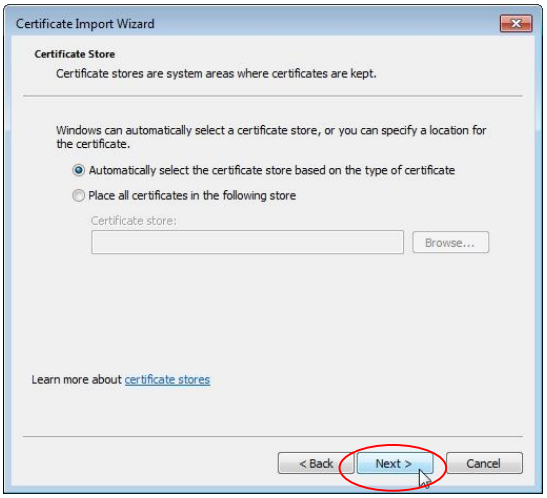


Figure 2-5 Fourth screen of "Certificate Import Wizard"

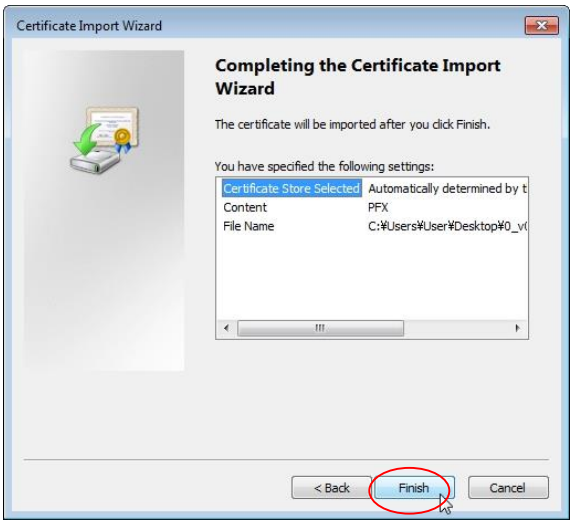


Figure 2-6 Fifth screen of "Certificate Import Wizard"

Figure 2-7 Sixth screen of "Certificate Import Wizard"

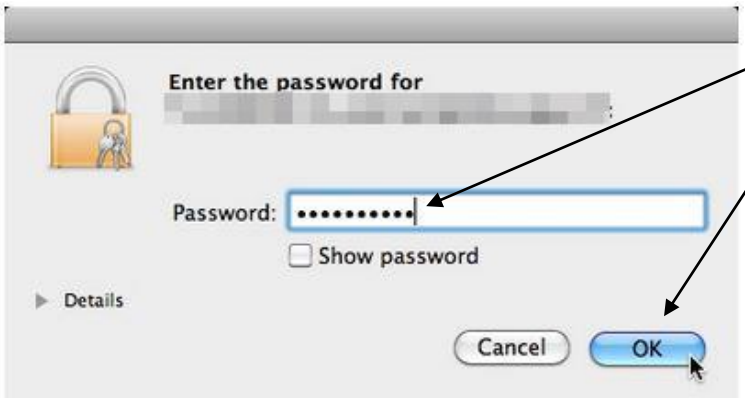
When you use the Firefox as the standard browser, refer to “2.2.1.3 **Install Client Certificate (Ubuntu Environment)**”

2.2.1.2 **Install Client Certificate (Mac Environment)**

Install the client certificate ACCC sent you by e-mail.

1. Double click the client certificate provided by ACCC.

Figure 2-8 Client certificate icon



1. Enter the password for "Client Certificate" issued by ACCC.
2. Click "OK" button.

Figure 2-9 Install the client certificate

Open "Keychain Access". (Finder > Application > Utility > Keychain Access)

Click "My Certificates" category to see available client certificates. Control-Click your client certificate for the HOKUSAI GreatWave system and select "New Identity Preference..." from the contextual menu.

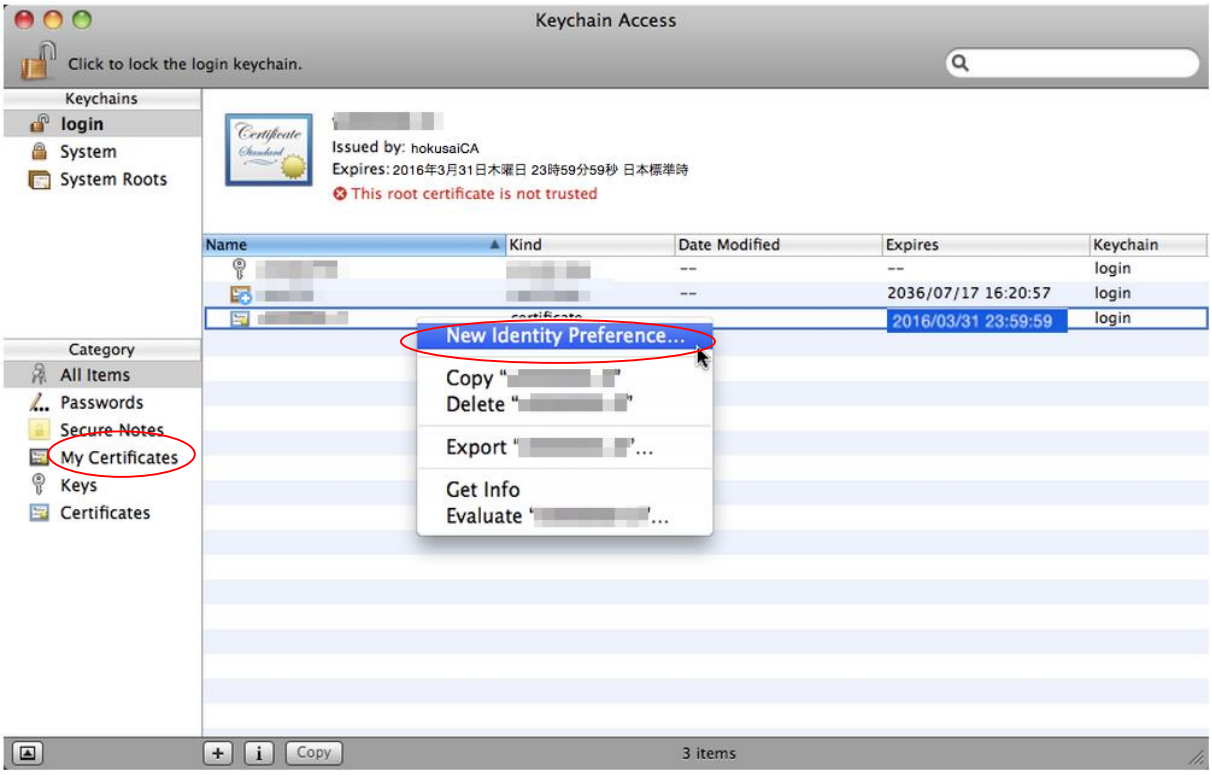


Figure 2-10 Keychain Access

Enter https://hokusai.riken.jp/ in [Location or Email Address:] and click [Add] button.

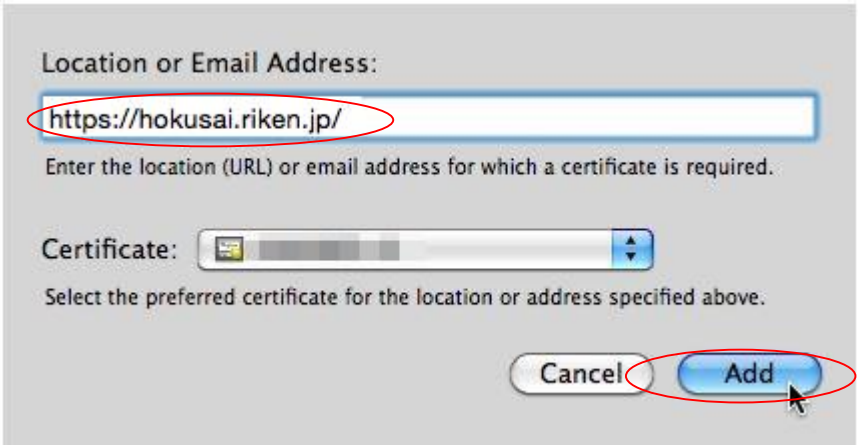


Figure 2-11 New Identity Preference

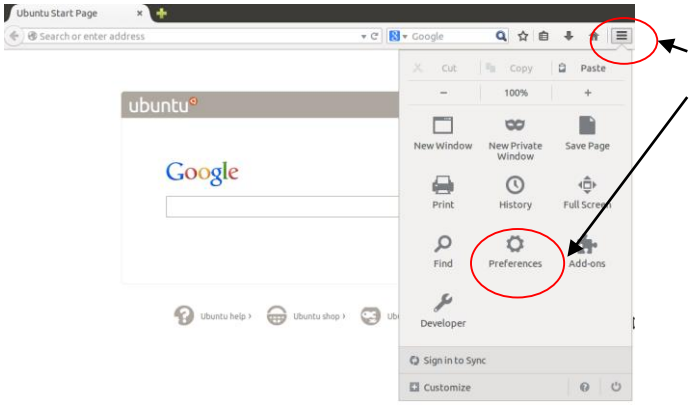
When you use the Firefox as the standard browser, refer to “2.2.1.3 **Install Client Certificate (Ubuntu Environment)**”

2.2.1.3 **Install Client Certificate (Ubuntu Environment)**

Import the client certificate ACCC sent you by e-mail.

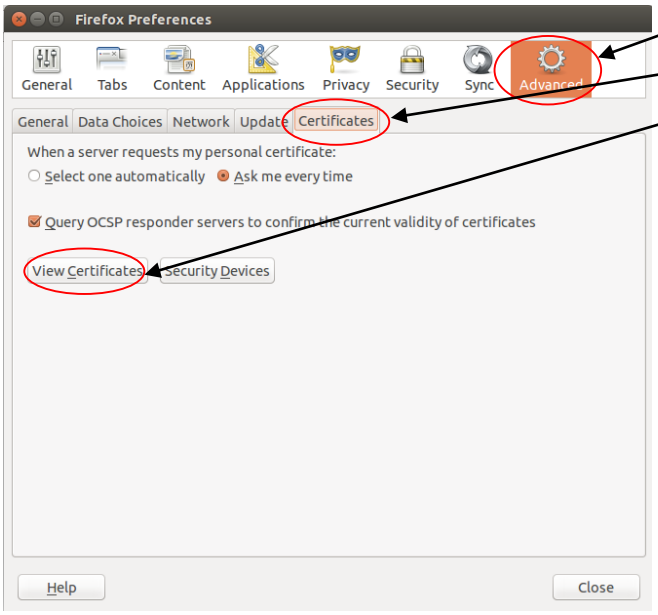
1. Launch Firefox.

Figure 2-12 Launch Firefox



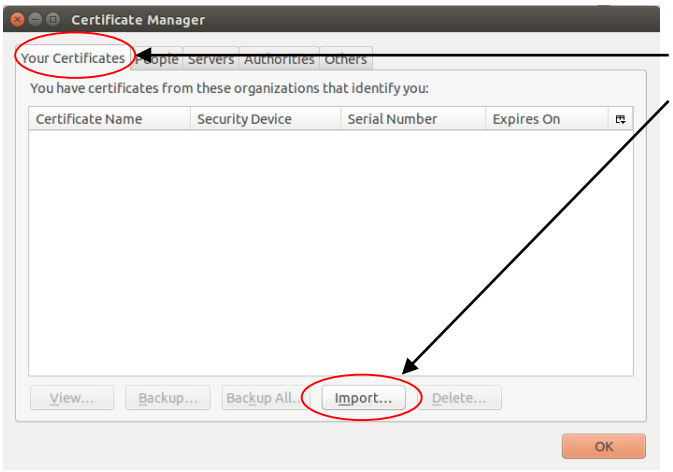
1. Click [Open menu].
2. Click [Preferences].

Figure 2-13 Firefox menu



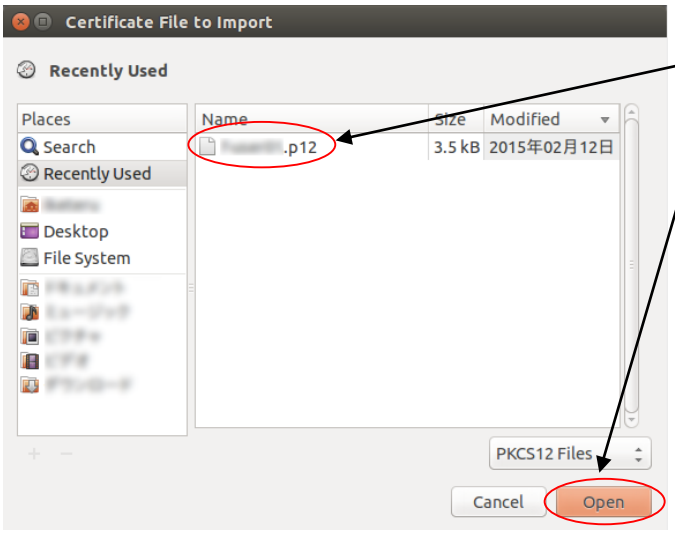
3. Click [Advanced] panel.
4. Click [Certificates] tab.
5. Click [View Certificates] button.

Figure 2-14 Firefox Preferences



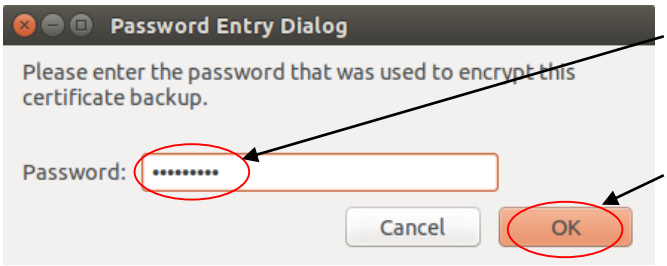
6. Click [Your Certificates]
7. Click [Import] button.

Figure 2-15 Firefox Certificate Manager



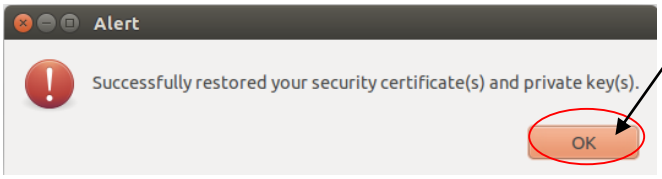
8. Select "Client Certificate" sent by ACCC.
9. Click [Open] button.

Figure 2-16 Firefox Certificate Manager Import



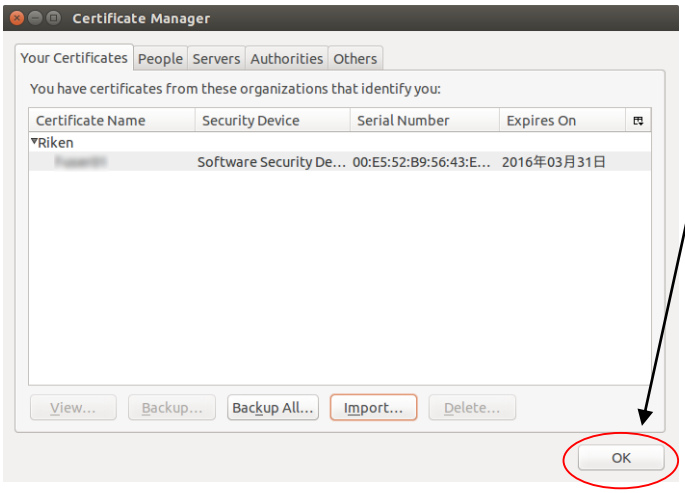
10. Enter the password for "Client Certificate" issued by ACCC.
11. Click [OK] button.

Figure 2-17 Firefox Password Entry Dialog



12. Click [OK] button.

Figure 2-18 Firefox Alert



13. Click [OK] button.

Figure 2-19 Firefox Certificate Manager

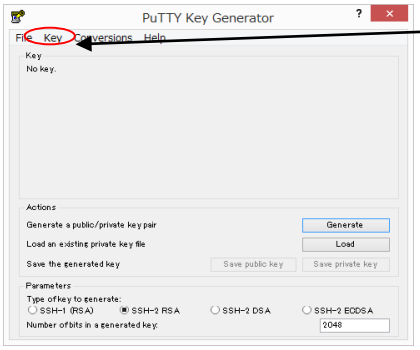
2.2.2 Key Generation

2.2.2.1 Key Generation (Windows Environment)

Generate the private key/public key pair with SSH-2 RSA method or SSH-2 DSA method on your PC. To generate the keys, use PuTTYgen utility provided with PUTTY package. If the keys are already generated, skip this step.

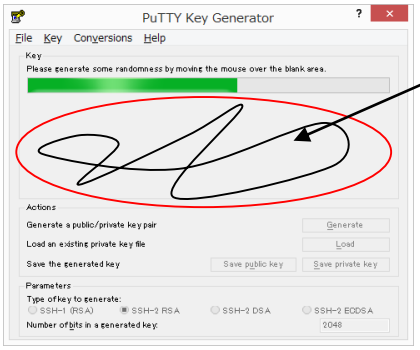
PuTTY can be downloaded from the following site:

PuTTY: http://www.chiark.greenend.org.uk/~sgtatham/putty/



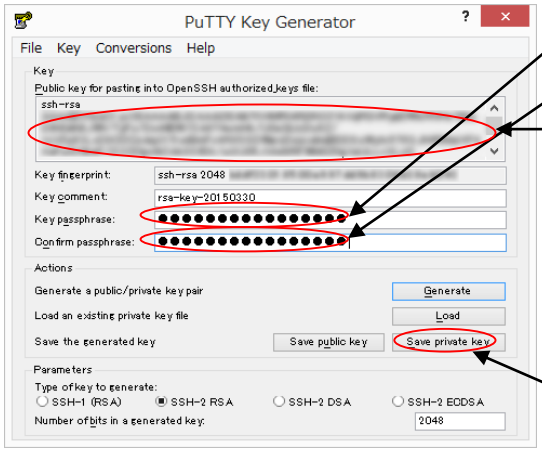
1. Click [Key] menu.
2. Click [Generate key pair].

Figure 2-20 First screen for private/public key generation with PuTTYgen



3. Move around the mouse to generate random numbers for key generation.

Figure 2-21 Second screen for private/public key generation with PuTTYgen



4. Enter the passphrase
5. Re-enter the passphrase.
6. Copy the contents of public key and register them according to 2.2.3 Register ssh public key,
   or
   save to the file(name:id_rsa.pub) using the text editor.
7. Click [Save private key] button to save.

Figure 2-22 Third screen for private/public key generation with PuTTYgen

2.2.2.2 **Key Generation (UNIX/Mac Environment)**

Generate the private key/public key pair with SSH-2 RSA method or SSH-2 DSA method on your PC. To generate the keys, use the *ssh-keygen* command. If the keys are already generated, skip this step.

- UNIX/Linux
  Launch the terminal emulator, and run the *ssh-keygen* command.
- Mac
  Launch the Terminal, and run the *ssh-keygen* command.

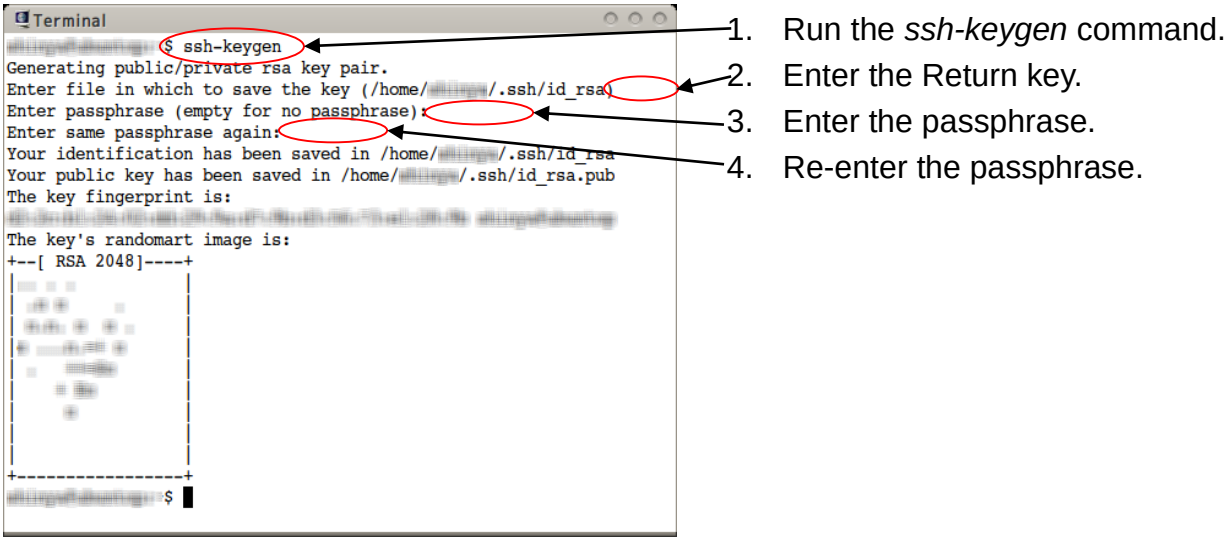


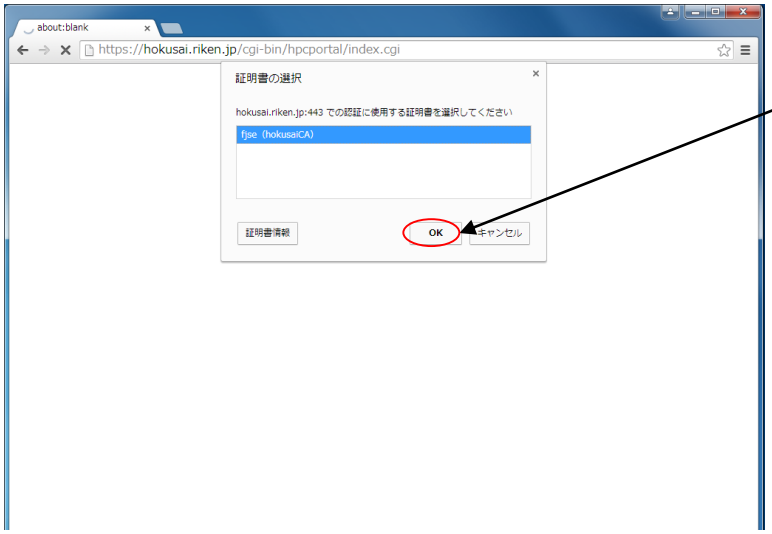
Figure 2-23 Generate key pair with the ssh-keygen command

2.2.3 Register ssh public key

(1) Access the following URL: (Use HTTPS because HTTP is not supported.)

https://hokusai.riken.jp/

(2) Login to the User Portal with the client certificate.



Select the client certificate for HOKUSAI GreatWave system, and click [OK] button.

Figure 2-24 Screen of certificate selection

(3) Click "Setting" menu.

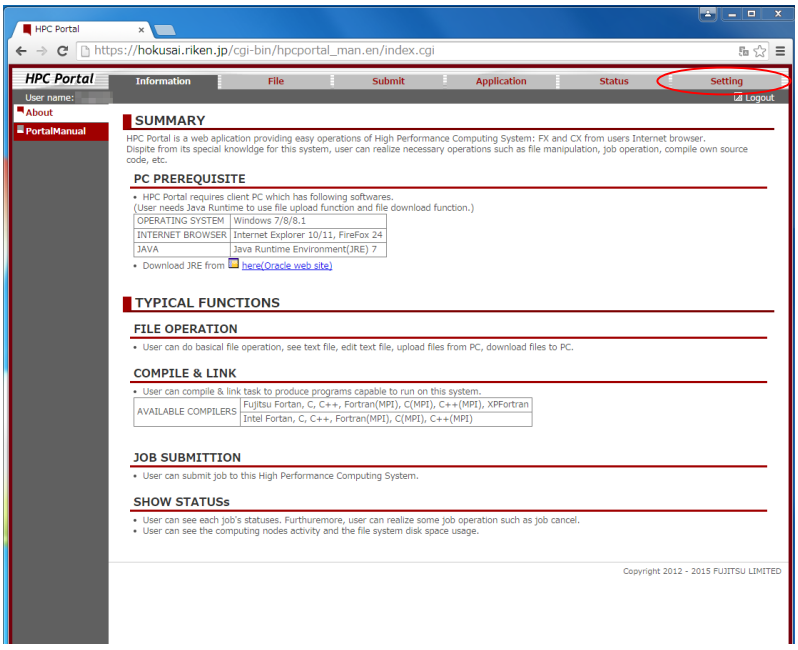


Figure 2-25 Screen of clicking "Setting" menu

(4) Click "SSHKey Set".

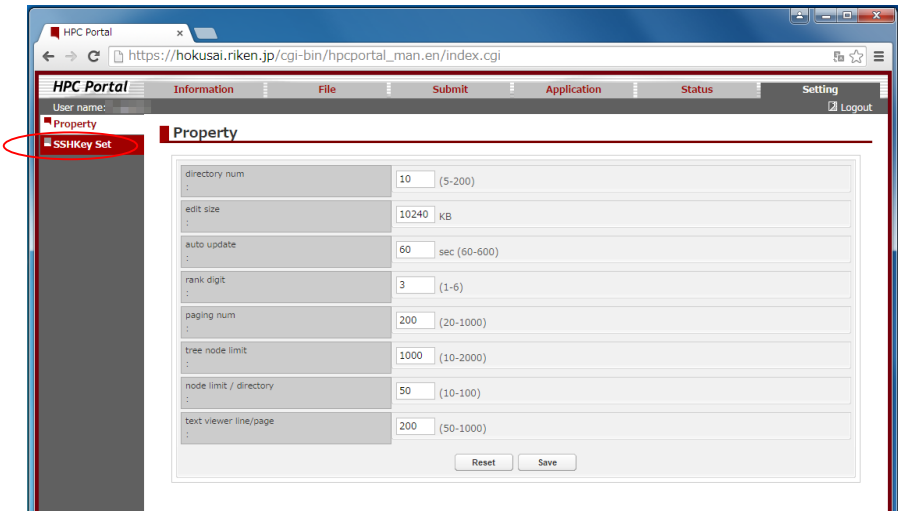


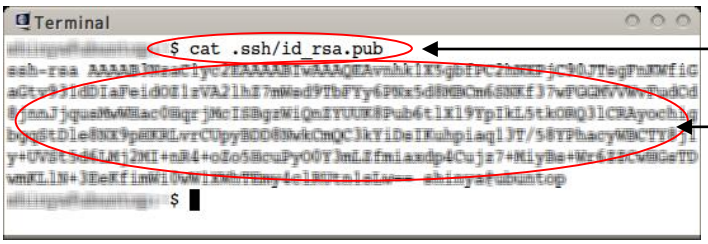
Figure 2-26 Screen of clicking "SSHKey Set"

(5) Register SSH public key

Refer to "2.2.2.1 Key Generation (Windows Environment)" for Windows and "2.2.2.2 Key Generation (UNIX/Mac Environment)" for UNIX/Mac about the key generation.

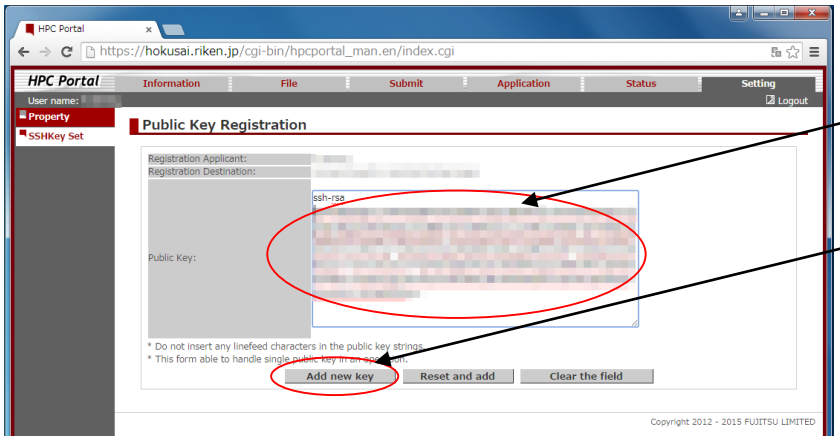
- Windows: Display the contents of the public key file (id_rsa.pub) with any text editor
- Mac: Launch Terminal, and display the contents of the public key with the *cat* command.
- UNIX/Linux: Launch virtual terminal, and display the contents of the public key with the *cat* command.

<u>The default path of public key file is ~/.ssh/id_rsa.pub or ~/.ssh/id_dsa.pub.</u>



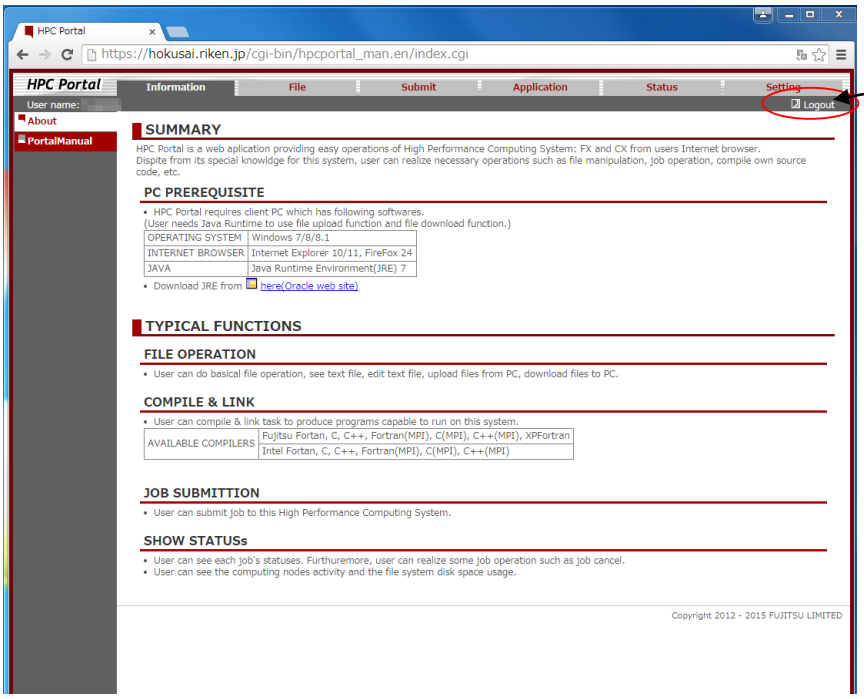
1. Display the public key.
2. Copy the contents.

Figure 2-27 Copy the contents of the public key



1. Paste the contents of public key.
2. Click [Add new key] button.

Figure 2-28 Register the public key



1. Click [Logout]

Figure 2-29 Logout from User Portal

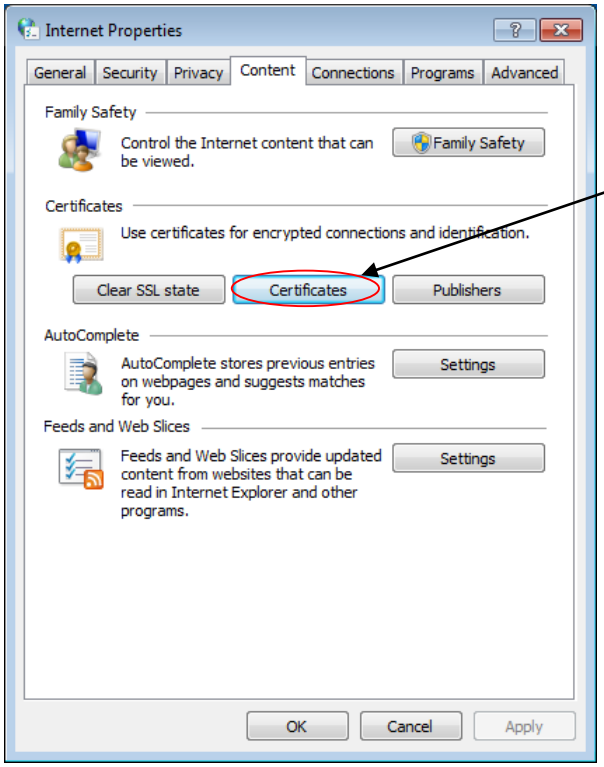
2.2.4 Update Client Certificate

When you need to update the expired client certificate, please refer the sections: "2.2.1 Install Client Certificate" to install new certificate and "2.2.5 Uninstall Client Certificate" to remove the old certificate.

2.2.5 Uninstall Client Certificate

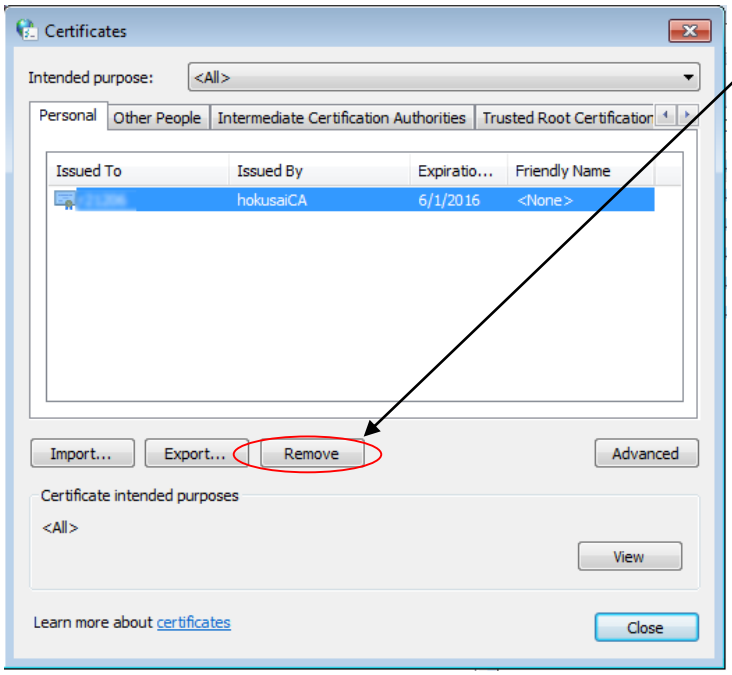
When you need to update client certificate or you do not need to use client certificate anymore, please remove the current certificate from your PC according to the following procedure.

2.2.5.1 Uninstall Client Certificate (Windows Environment)



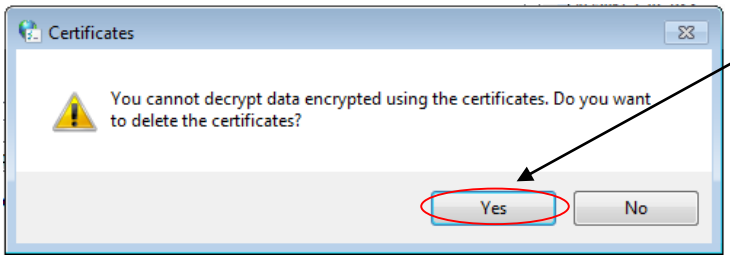
1. Click [Start Menu]-> [Control Panel]-> [Internet Options].
2. Click “Certificate” button.

Figure 2-30 Screen of "Internet Properties"



1. Select the Certificate, Click “Remove” button.

Figure 2-31 First screen of "Certificates"



1. Click “Yes” button.

Figure 2-32 Second screen of "Internet Properties"

2.2.5.2 **Uninstall Client Certificate (Mac Environment)**

Open "Keychain Access". (Finder > Application > Utility > Keychain Access)

Click "My Certificates" category to see available client certificates. Control-Click your client certificate for the HOKUSAI GreatWave system and select "New Identity Preference..." from the contextual menu.

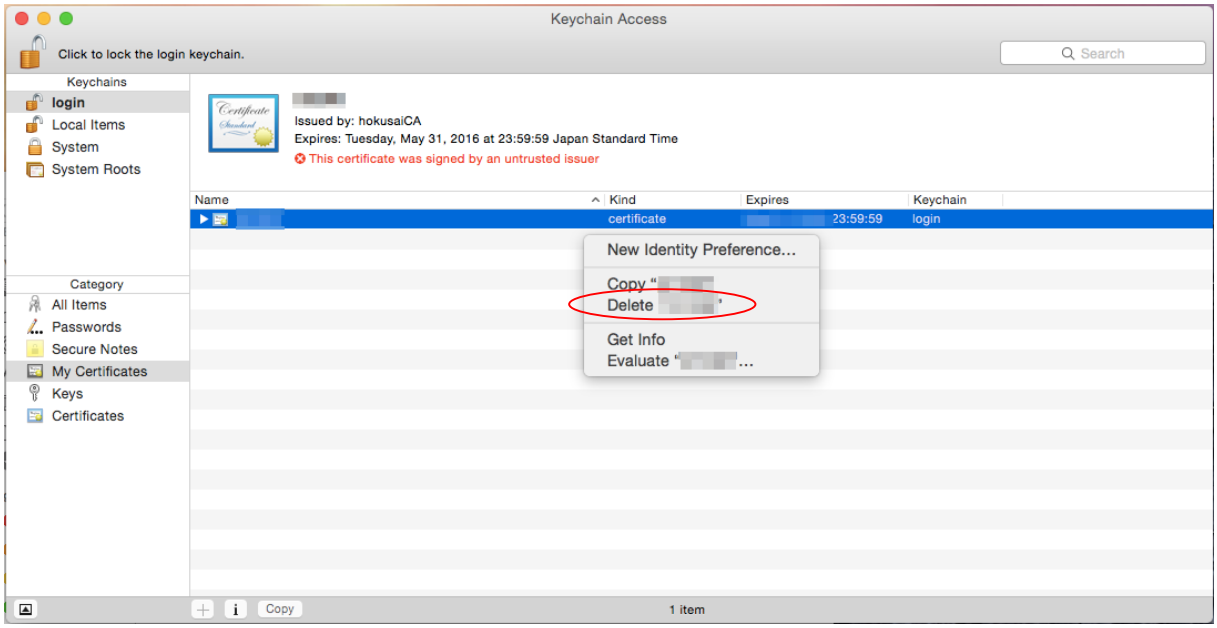


Figure 2-33 Keychain Access

Click “Delete” button.

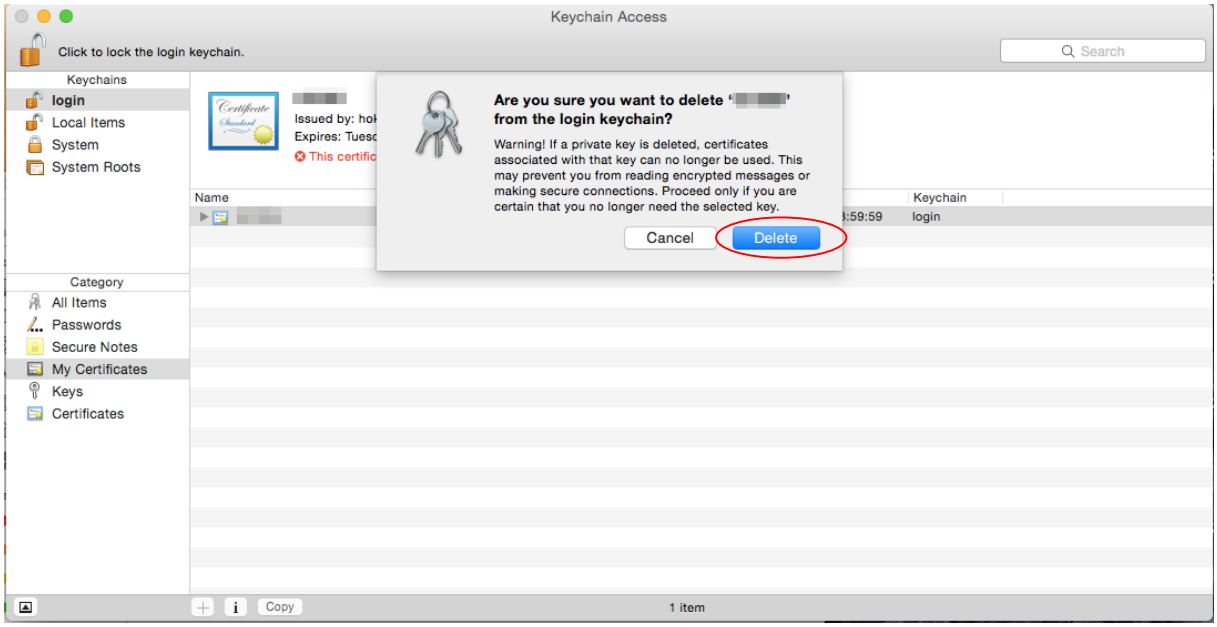


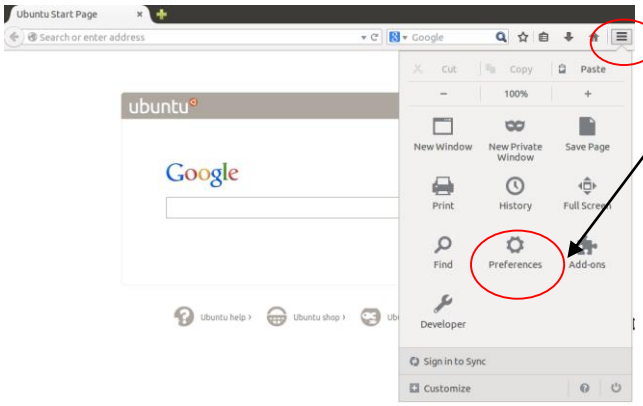
Figure 2-34 Keychain Access Delete

2.2.5.3 **Uninstall Client Certificate(Ubuntu Environment)**

Import the client certificate ACCC sent you by e-mail.

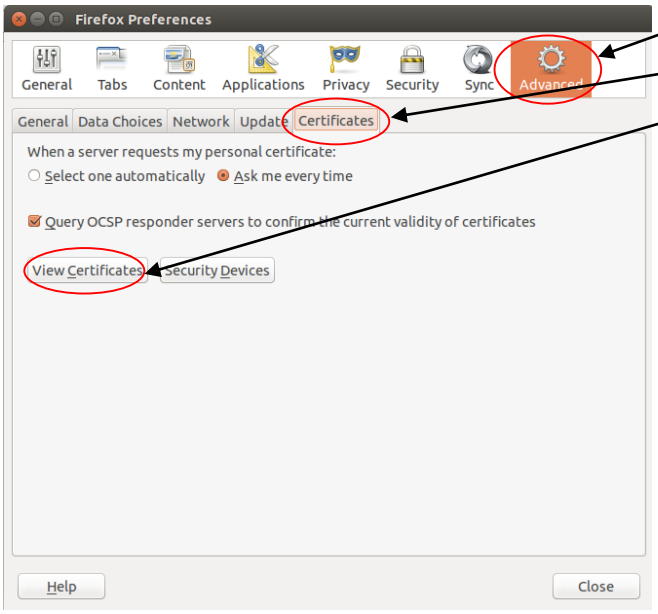
2. Launch Firefox.

Figure 2-35 Launch Firefox



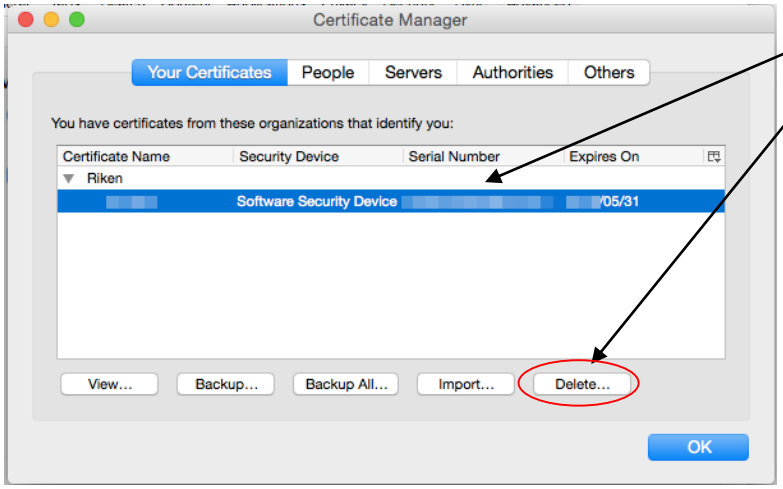
3. Click [Open menu].
4. Click [Preferences].

Figure 2-36 Firefox menu



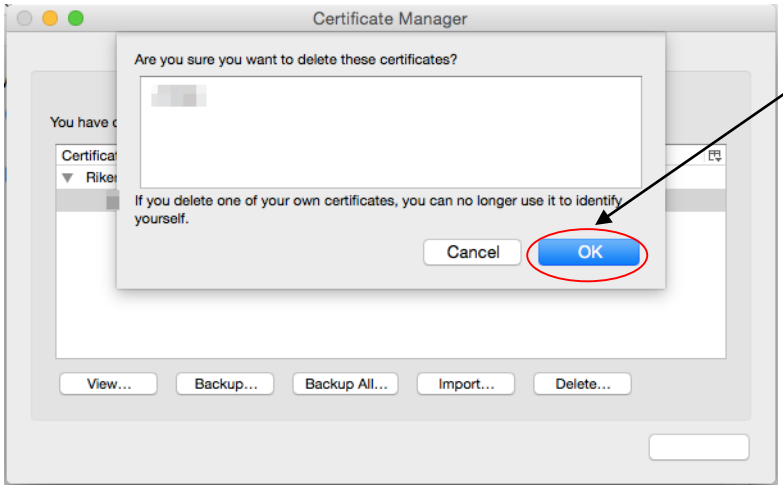
6. Click [Advanced] panel.
7. Click [Certificates] tab.
8. Click [View Certificates] button.

Figure 2-37 Firefox Preferences



9. Click Client Certificate.
10. Click [Delete] button.

Figure 2-38 Firefox Preferences



11. Click [OK] button.

Figure 2-39 Firefox Certificate Manager Delete

2.3 Network Access

The servers within the HOKUSAI GreatWave system in which you can access via SSH/HTTPS are the front end servers. The front end server consists of 4 servers.

For SSH access, only public key authentication with SSH protocol version 2 is enabled.

The User Portal enables you to register the SSH public key, operate files, manage jobs and view manuals via HTTPS.

Destination hosts are as follows:

Table 2-1 Destination hosts

Host name (FQDN)	Service	Purpose to access
greatwave.riken.jp	SSH	• Virtual terminal • File transfer
hokusai.riken.jp	HTTPS	• Register the SSH public key • Operate files • Manage jobs • View manuals • Use Development Tools (FX100)

2.4 SSH Login

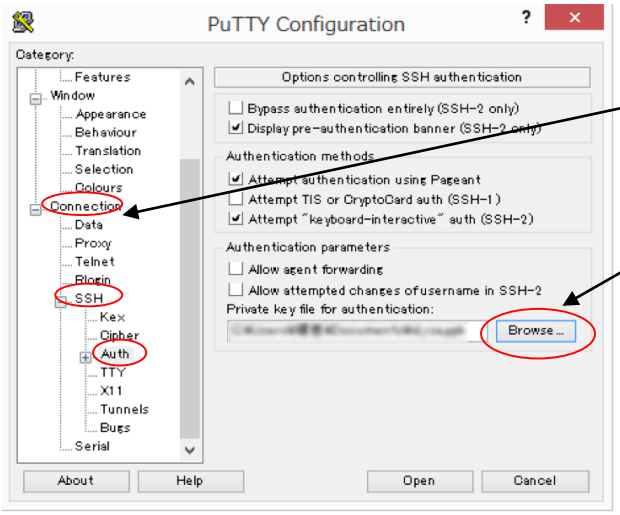
2.4.1 SSH Login (Windows Environment)

This section describes how to use PUTTY for virtual terminal while various terminal software tools are available on Windows. For users who use Cygwin, refer to 2.4.2 SSH Login (UNIX/Mac Environment).

PuTTY can be downloaded from the following site:

PuTTY: http://www.chiark.greenend.org.uk/~sgtatham/putty/

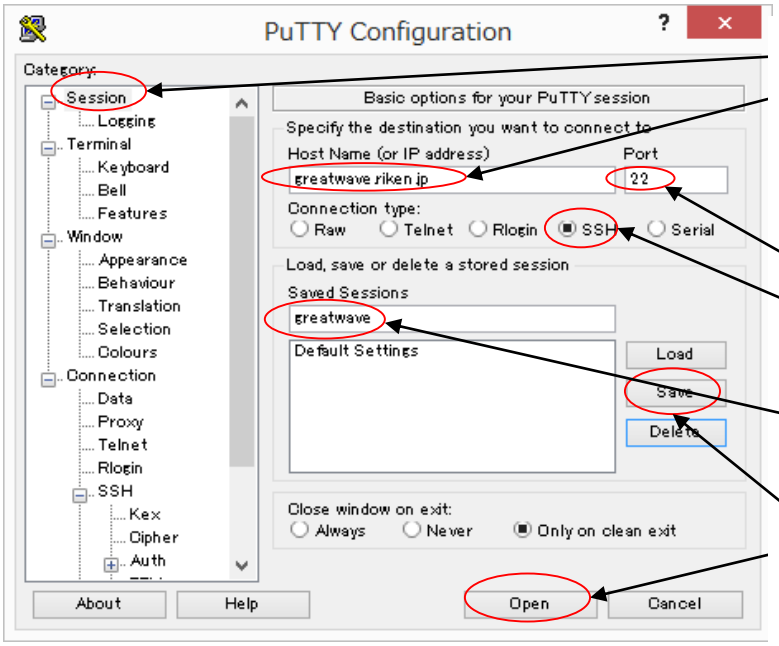
(1) Launch PuTTY



1. Click [Connection] - [SSH] - [Auth].
2. Click [Browse] button, and select the private key file.

Figure 2-40 Screen of selecting private key with PuTTY

(2) Open session to the HOKUSAI GreatWave system with PuTTY



1. Click [session]
2. Enter the following information
   [Host Name] greatwave.riken.jp
   [Port] 22
   [Connection type] SSH
3. [Saved Sessions] name(ex: greatwave)
4. Click [Save] button
5. Click [Open] button

Figure 2-41 PuTTY session screen

(3) Key generation

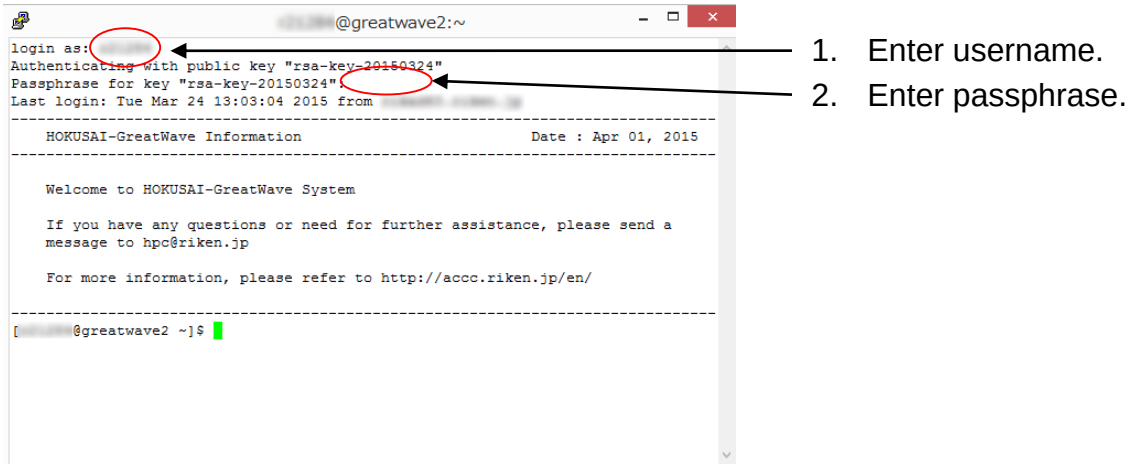


Figure 2-42 PuTTY login screen

2.4.2 SSH Login (UNIX/Mac Environment)

To login to the HOKUSAI GreatWave system from your PC via SSH, use the *ssh* command.

```
$ ssh -l username greatwave.riken.jp
The authenticity of host '
Enter passphrase for key '/home/username/.ssh/id_rsa': ++++++++   ←
Enter passphrase
[username@greatwave1 ~]$
```

2.5 SSH Agent Forwarding

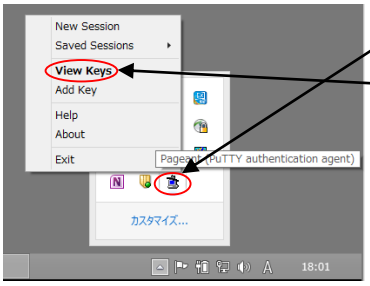
When you access external systems from the HOKUSAI GreatWave system, login to the front end servers enabling SSH Agent forwarding.

To prevent the front end servers from being used as a step to access external system, it is prohibited to store the private key on the HOKUSAI GreatWave system.

The protocols permitted to access from HOKUSAI GreatWave system to external system are only SSH, HTTP and HTTPS.

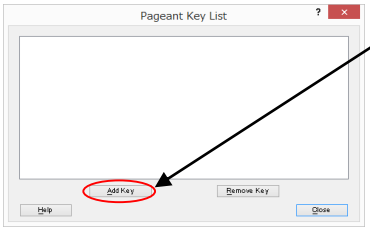
2.5.1 SSH Agent Forwarding (Windows Environment)

(1) Launch Pageant utility provided with PUTTY package

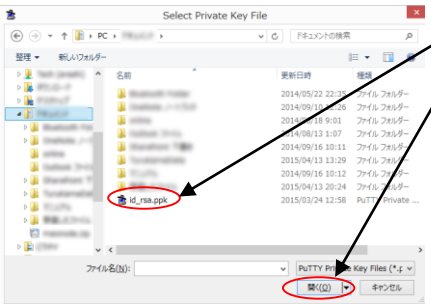


1. Right-click Pageant.
2. Click [View Keys].

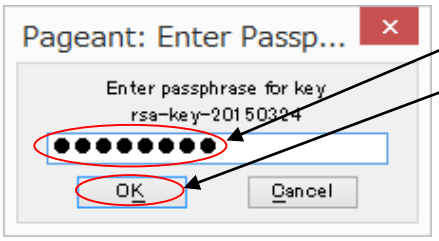
(2) Register the authentication key.



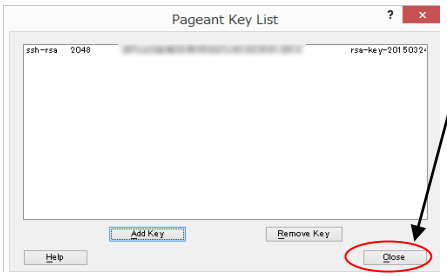
3. Click [Add Key] button.



4. Select the private key.
5. Click [Open] button.

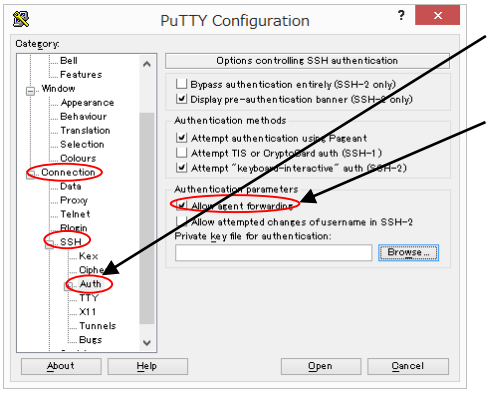


6. Enter the passphrase.
7. Click [OK] button.



8. Click "Close" button.

(3) Launch PuTTY, enable SSH Agent Forwarding and access to HOKUSAI GreatWave system.



9. Click [Connection] – [SSH] – [Auth]
10. Check [Allow agent forwarding]

2.5.2 SSH Agent Forwarding (Mac Environment/Ubuntu Environment)

The SSH Agent Forwarding is automatically launched on the Mac OS X(Yosemite) environment and the Ubuntu(14.10) environment. The SSH Agent Forwarding will be enabled if you specify the -A option when accessing to the HOKUSAI GreatWave system.

```
[username@Your-PC ~]$ ssh -A -l username greatwave.riken.jp
```

2.6 File Transfer

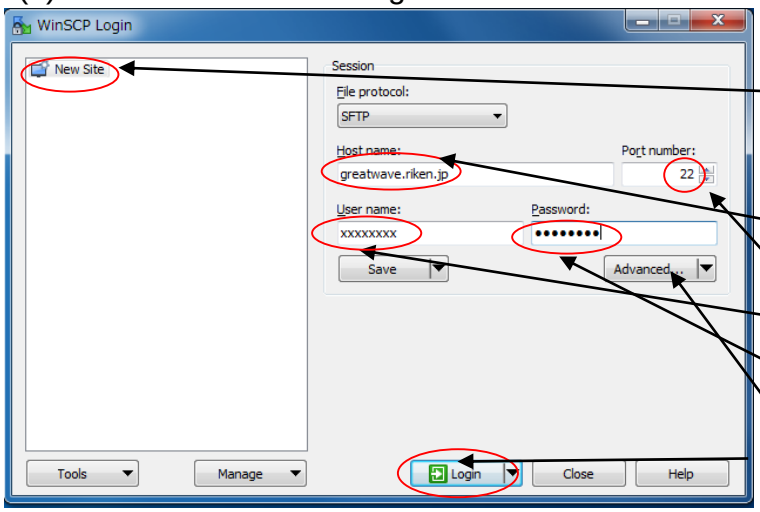
2.6.1 File Transfer (Windows Environment)

This section describes how to use WinSCP for file transfer between the your PC and HOKUSAI GreatWave system. WinSCP can be downloaded from the following site:

WinSCP: http://winscp.net/eng/index.php

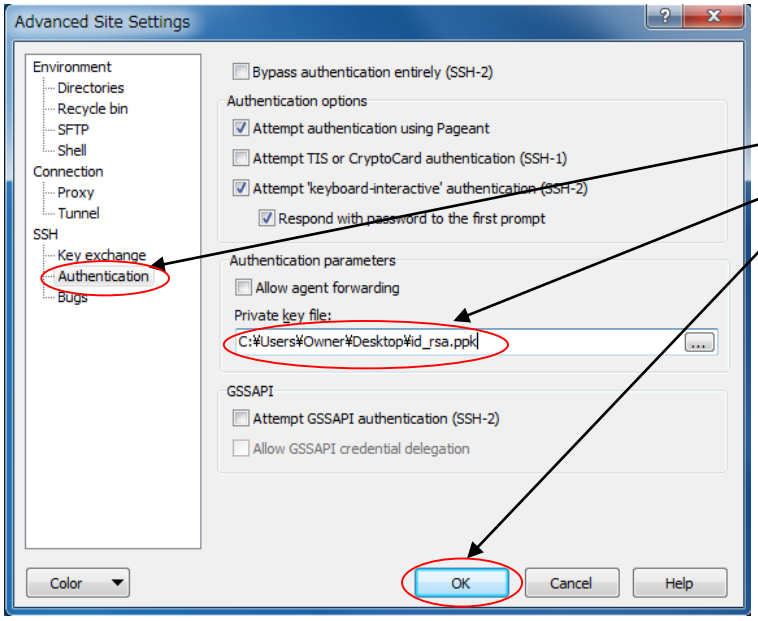
WinSCP can be used to transfer files by drag-and-drop operation after logging into HOKUSAI GreatWave system.

(4) Launch WinSCP for login



1. Click [Session]
2. Enter the following information:
   [Host name] greatwave.riken.jp
   [Port number] 22
   [User name] username
   [Password] passphrase
3. Click [Advanced] button

Figure 2-43 WinSCP Login



4. Click [Authentication]
5. Select private key file
6. Click [OK] button

Figure 2-44 WinSCP Settings

(5) Files can be downloaded or uploaded by drag-and-drop operation.

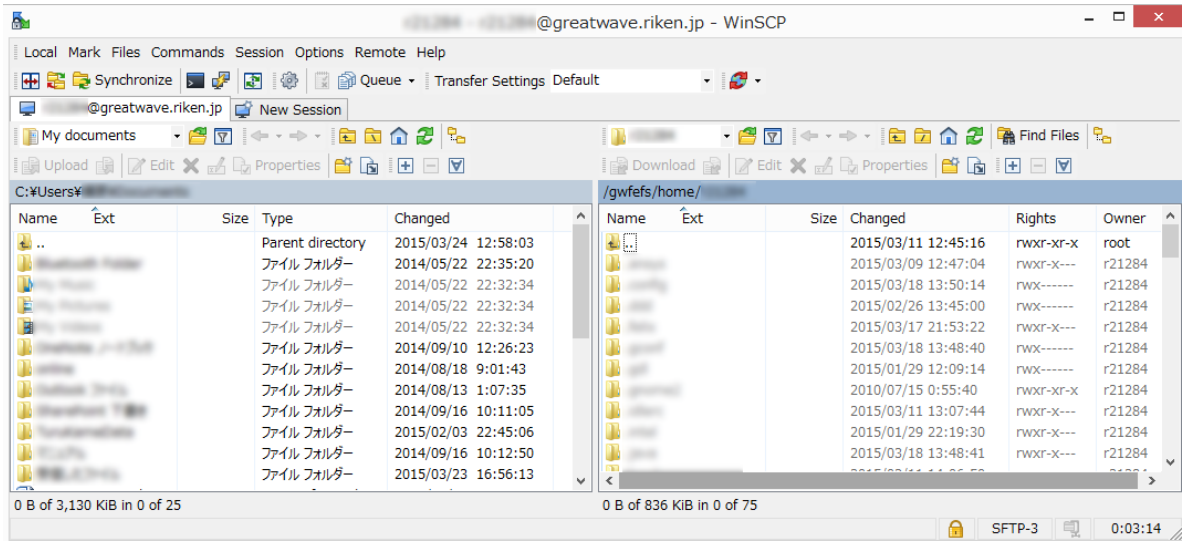


Figure 2-45 Screen after login with WinSCP

2.6.2 File Transfer (UNIX/Mac Environment)

To transfer files between the HOKUSAI GreatWave system and PC, use the *scp* (*sftp*) command.

```
$ scp local-file username@greatwave.riken.jp:remote-dir
Enter passphrase for key  : ++++++++     ← Enter passphrase
local-file     100% |**********************|  file-size  transfer-
```

2.7 Login Shell

The default login shell is set to bash. If you would like to change login shell, please contact hpc@riken.jp. The configuration files for login shell to use the HOKUSAI GreatWave system are located in each user's home directory. The original (Skelton) files are located on /etc/skel.

When you customize environment variables such as PATH, you need to add the new path to the last. Otherwise, you cannot use the system correctly.

3. File Systems Overview

3.1 Storage Area

The following storage areas are available in the HOKUSAI GreatWave system.

Table 3-1 Storage areas

Storage Type	Area	Size	Purpose
Online Storage	/home	2.2 PB	Home area
	/data		Data area (application is required per project)
	/tmp_work		Temporary area
Hierarchical Storage	/arc	7.9 PB	Archive area (application is required per project)

The access to each storage from each server is as follows:

Table 3-2 Access to storage area

Storage Type	Front end servers	MPC	ACSG	ACSL
Online Storage	○	○	○	○
Hierarchical Storage	○	×	×	×

○ : Available × : No available

3.1.1 Online Storage

The Online Storage provides home area for users and data area for the projects. The quota of home area is 4 TB per user. To use data area, application is required.

Table 3-3 Quota of the Online Storage

Directory	Block quota	Inode quota	Note
/home	4 TB	10 millions	10 millions
/data	4 TB - 52 TB	One million per 1 TB	Application is required

The Online Storage can be accessed from the front end servers, the Massively Parallel Computer and the Application Computing Server.

3.1.2 Hierarchical Storage

The Hierarchical Storage can be accessed from the front end servers. To use this area, application is required.

Table 3-4 Quota of the Hierarchical Storage

User Directory	Quota	Inode quota
/arc	2 tape volumes/8TB - 13 tape volumes/52TB	40,000 per tape volume

The main purposes of use are as follows:

- Store a large data for a long period
- Backup

Since the Hierarchical Storage stores data in tapes, it is not suitable to store smaller files such as less than 100 MB. If you store many small files, create single file archive and then store.

3.1.3 Application of using storage area

To use /data or /arc storage area, fill the following format and send an e-mail to hpc@riken.jp.

Select subject from below:
New request: /data
Additional request: /data
New request: HSM
Additional request: HSM

Message:
(1) Project ID:
(2) Name of management representative(in case project representative delegate someone):
(3) Size of request: /data( )TB, HSM ( )TB
(4) Current permitted size and usage rate (It is needed for additional request only):
(5) Reason of estimation of increase of data:

3.2 **Disk usage**

You can use the *listquota* command to display your disk usage and quota.

```
[username@greatwave1 ~]$ listquota
Disk quota for user username
          [Size]  Used(GB)  Limit(GB) Use(%) [Files]   Used(K)   Limit(K) Use(%)
------------------------------------------------------------------------------
/home/username         289       4,000   6.7%               579      10,000   5.8%

Disk quota for project(s).
          [Size]  Used(GB)  Limit(GB) Use(%) [Files]   Used(K)   Limit(K) Use(%)
------------------------------------------------------------------------------
Q99999
 +- /data/Q99999         -       4,000   0.0%                 -       1,000   0.0%
```

Table 3-5 listquota information

Item	Description
[Size]	Block usage (GB), quota (GB) and the ratio
[Files]	Inode usage (K), quota (K) and the ratio
/home/username /data/projectID /arc/projectID	Area /home : Online Storage (home area) /data: Online Storage (data area) *1 /arc: Hierarchical Storage (archive area) *1

*1 : The data area and archive area appear only when application is approved.

3.3 Temporary area

/tmp_work is available to store temporary files.

The data stored under /tmp_work is automatically removed when its modified date becomes older than one week. Use this storage area only for storing temporary files exclusively.

The users can use the *mktmp_work* command to pass data between each user. The *mktmp_work* command creates a temporary directory under /tmp_work, then the user can copy the file and change the permission to be passed on this directory, and notify the other party of this directory path.

```
[username@greatwave1 ~]$ mktmp_work
mktmp_work: INFO: /tmp_work/username.1G2XFQ30/KXrWerIviNTzZnh0 is created.
[username@greatwave1 ~]$ cp input.dat ¥
    /tmp_work/username.1G2XFQ30/KXrWerIviNTzZnh0/
[username@greatwave1 ~]$ chmod o+r ¥
    /tmp_work/username.1G2XFQ30/KXrWerIviNTzZnh0/input
```

Users other than the one who run the *mktmp_work* command are allowed only to view the files.

3.4 **Data storing method on the Online Storage**

On the Online Storage, data is distributed per file to more than one disk device (OST) via the I/O server (OSS) in the round-robin fashion. The round-robin processing enables high performance file access through more than one I/O server (OSS) even when accessing a large number of files.

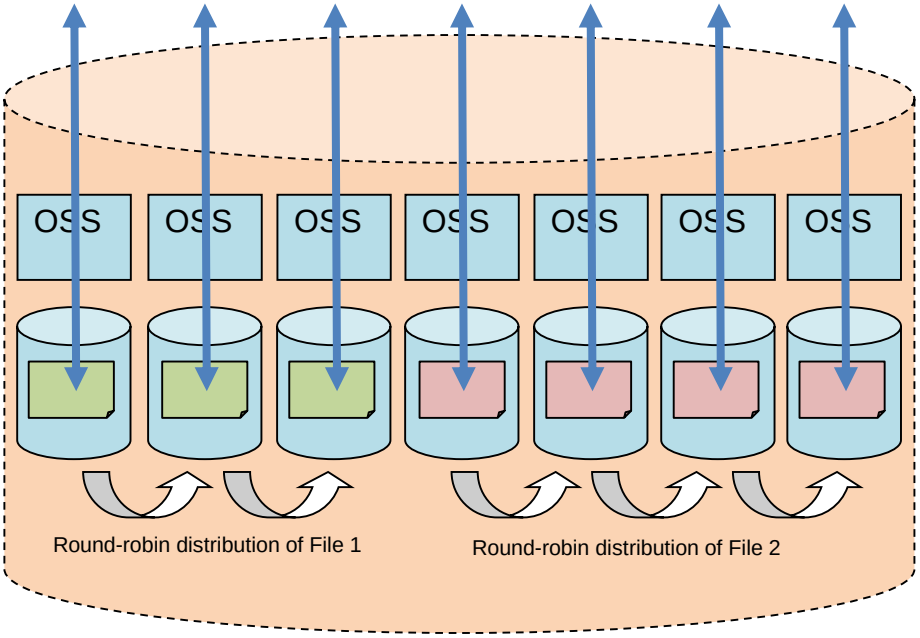


Figure 3-1 Round-Robin

4. Compile and Link

4.1 Set environment settings

The *module* command enables you to set environment variables to use compilers, libraries, applications, tools and so on.

```
$ module <subcommand> <subcommand-args>
```

The sub-commands of the *module* command are the following:

Table 4-1 Sub-commands of the *module* command

Sub command	Description
avail	List all available settings
list	List loaded settings
load module...	Load setting(s) into the shell environment
unload module...	Remove setting(s) from the shell environment
purge	Unload all loaded settings
switch module1 module2	Switch loaded module1 with modules

Example) List all available settings.

```
[username@greatwave1 ~]$ module avail

------- /gwfefs/opt/modulefiles/greatwave/apps --------
ansys/15.0(default)       vmd/1.9.2(default)
gaussview/5.0.9(default)

----- /gwfefs/opt/modulefiles/greatwave/compilers -----
gcc/4.8.4(default)
cuda/6.5
cuda/7.0(default)
intel/composer_xe_2013.1.117
intel/composer_xe_2015.1.133(default)
sparc/2.0.0-02(default)
```

* The version listed with “(default)” is the recommended version on HOKUSAI GreatWave system.

Example) Load compiler's setting for the Massively Parallel Computer.

```
[username@greatwave1 ~]$ module load sparc
```

Example) List loaded settings.

```
[username@greatwave1 ~]$ module list
Currently Loaded Modulefiles:
1) sparc/2.0.0-02
```

You cannot load the settings which conflict with the loaded settings at once. If you try to load setting, the following error messages are displayed and it failed to set.

```
[username@greatwave1 ~]$ module load intel
intel/composer_xe_2015.1.133(61):ERROR:150: Module
'intel/composer_xe_2015.1.133' conflicts with the currently loaded
module(s) 'sparc/2.0.0-02'
intel/composer_xe_2015.1.133(61):ERROR:102: Tcl command execution
failed: conflict sparc
```

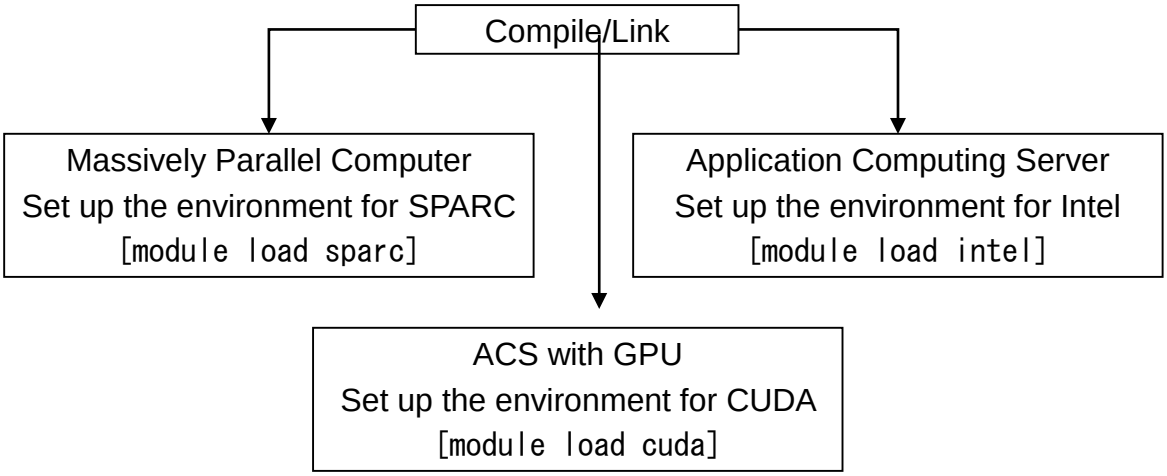
To switch the compiler or switch the version of loaded library, use the "switch" sub-command.

Example) Switch the compiler to for the Application Computing Server.

```
[username@greatwave1 ~]$ module switch sparc intel
```

4.2 **Compiler**

On the front end servers of the HOKUSAI GreatWave system, the cross compilers to create load modules which run on the Massively Parallel Computer, and the compilers to create load modules which run on the Application Computing Server are available. (The CPU architectures of the front end servers and the Application Computing Server is Intel Architecture.)



When compiling and linking for the Massively Parallel Computer on the front end servers, specify "sparc" in the argument of the *module load* command. When compiling and linking for the Application Computing Server on the front end servers, specify "intel" in the argument of the *module load* command.

Example 1) Compile and link for the Massively Parallel Computer.

```
[username@greatwave1 ~]$ module load sparc
[username@greatwave1 ~]$ module list
Currently Loaded Modulefiles:
  1) sparc/2.0.0-01
```

Example 2) Compile and link for the Application Computing Server.

```
[username@greatwave1 ~]$ module load intel
[username@greatwave1 ~]$ module list
Currently Loaded Modulefiles:
  1) intelmpi/5.0.2.044     2) intel/composer_xe_2015.1.133
```

Example 3) Compile and link for CUDA program

```
[username@greatwave1 ~]$ module load cuda
[username@greatwave1 ~]$ module list
  Currently Loaded Modulefiles:
  1) cuda/7.0
```

4.3 **Endian**

The term "endian" refers to the convention for storing multi-byte values in memory. Suppose there is a 2 byte quantity, written as 1234 which is hexadecimal. In big endian, 12 (the most significant byte) is stored in the smallest address. On the other hand, in little endian, 34 (the least significant byte) is stored in the smallest address.

In the HOKUSAI GreatWave system, the Massively Parallel Computer (SPARC architecture) are big endian, in contrast the Application Computing Server and the front end servers (Intel64 architecture) are little endian. (In the RICC system, all nodes (Intel64 architecture) are little endian.)

When porting programs or data between systems with different endian, since it is necessary to be aware of the byte order of binary file in order to avoid inconsistency of data, development and execution of programs must be performed based on the consideration of the endian.

When you run a Fortran program, you may specify the runtime option -Wl,-T to read and write unformatted data as little endian data on the Massively Parallel Computer.

Example 1) Specify the environment variable. (FORT90L)

```
export FORT90L=-Wl,-T
./a.out
```

Example 2) Specify the arguments for the load module.

```
./a.out -Wl,-T
```

If the unit number "uno" is specified in the command, such as "-Wl,-Tuno", the endian conversion is enabled only for the specified unit number.

4.4 How to Compile and Link

The commands for compilation and linkage are as follows:

Table 4-2 Compile and link commands for the Massively Parallel Computer

Type	Programming language	Command	Automatic parallelization*1	OpenMP*1
Sequential (no MPI)	Fortran	frtpx	-Kparallel	-Kopenmp
	C	fccpx		
	C++	FCCpx		
MPI parallel	Fortran	mpifrtpx		
	C	mpifccpx		
	C++	mpiFCCpx		

*1: Automatic parallelization and OpenMP options are not set by default.

Table 4-3 Compile and link commands for the Application Computing Server

Type	Programming language	Command	Automatic parallelization*1	OpenMP*1
Sequential (no MPI)	Fortran	ifort	-parallel	-openmp
	C	icc		
	C++	icpc		
MPI parallel	Fortran	mpiifort		
	C	mpiicc		
	C++	mpiicpc		

*1: Automatic parallelization and OpenMP options are not set by default.

Table 4-4 Compile and link commands for the ACS with GPU

Type	Programming language	Command	Automatic parallelization	OpenMP
GPU	CUDA	nvcc	-	-

4.4.1 Compile and Link for Massively Parallel Computer

4.4.1.1 Compile and link sequential programs

To compile and link sequential programs for the Massively Parallel Computer on the front end servers, use the *frtpx*/*fccpx*/*FCCpx* command.

```
frtpx/fccpx/FCCpx [option] file [...]
```

Example 1) Compile and link a sequential Fortran program for the Massively Parallel Computer.

```
[username@greatwave1 ~]$ frtpx seq.f
```

Example 2) Compile and link a sequential C program for the Massively Parallel Computer.

```
[username@greatwave1 ~]$ fccpx seq.c
```

4.4.1.2 Compile and link thread parallelization programs

To compile and link multi-threaded programs for the Massively Parallel Computer on the front end servers, use the *frtpx*/*fccpx*/*FCCpx* command.

```
frtpx/fccpx/FCCpx thread-option [option] file [...]
```

Example 1) Compile and link a Fortran program with the automatic parallelization for the Massively Parallel Computer.

```
[username@greatwave1 ~]$ frtpx -Kparallel para.f
```

Example 2) Compile and link a C program with the automatic parallelization for the Massively Parallel Computer.

```
[username@greatwave1 ~]$ fccpx -Kparallel para.c
```

Example 3) Compile and link an OpenMP Fortran program for the Massively Parallel Computer.

```
[username@greatwave1 ~]$ frtpx -Kopenmp omp.f
```

Example 4) Compile and link an OpenMP C program for the Massively Parallel Computer.

```
[username@greatwave1 ~]$ fccpx -Kopenmp omp.c
```

Example 5) Compile and link an OpenMP Fortran program with the automatic parallelization for the Massively Parallel Computer.

```
[username@greatwave1 ~]$ frtpx -Kparallel,openmp omp_para.f
```

Example 6) Compile and link an OpenMP C program with the automatic parallelization for the Massively Parallel Computer.

```
[username@greatwave1 ~]$ fccpx -Kparallel,openmp omp_para.c
```

4.4.1.3 **Compile and link MPI programs**

To compile and link MPI programs for the Massively Parallel Computer on the front end servers, use the *mpifrtpx*/*mpifccpx*/*mpiFCCpx* command.

```
mpifrtpx/mpifccpx/mpiFCCpx [option] file [...]
```

Example 1) Compile and link a MPI Fortran program for the Massively Parallel Computer

```
[username@greatwave1 ~]$ mpifrtpx mpi.f
```

Example 2) Compile and link a MPI C program for the Massively Parallel Computer

```
[username@greatwave1 ~]$ mpifccpx mpi.c
```

Example 3) Compile and link a Hybrid (MPI + OpenMP) Fortran program for the Massively Parallel Computer

```
[username@greatwave1 ~]$ mpifrtpx -Kopenmp mpi_omp.f
```

Example 4) Compile and link a Hybrid (MPI + OpenMP) C program for the Massively Parallel Computer

```
[username@greatwave1 ~]$ mpifccpx -Kopenmp mpi_omp.c
```

4.4.1.4 **Fortran Compiler option**

The main Fortran compiler options are the following. See the man page for more information.

Table 4-5 Fortran compiler options

Compile option	Description
-c	Produces object files only and do not link to create an executable.
-o *exe_file*	The executable is named exe_file instead of a.out. If the -c option is also specified, the object file is named exe_file.
-I *directory*	Names a directory to search for INCLUDE and module information files.
-Fixed	The program is in fixed source form. (By default, the file format is judged by the filename extension)
-Free	The program is in free source form. (By default, the file format is judged by the filename extension)
-X6	Compiles Fortran source program as FORTRAN66 source.
-X7	Compiles Fortran source program as FORTRAN77 source.
-X9	Compiles Fortran source program as Fortran 95 source.
-X03	Compiles Fortran source program as Fortran 2003 source.
-fw	Outputs only warning and serious error messages (no informative messages).
-fs	Outputs only serious error messages.
-f *msg_num*	Suppresses particular error message specified by msg_num.
-Nmaxserious=*maxnum*	Stops the compilation if s-level(serious) error detected more maxnum.
-Haefosux	Checks for argument mismatches (a), shape conformance (e), simultaneous OPEN and I/O recursive call (f), overlapping dummy arguments and extended undefined (o), subscript and substring values out of bounds (s), undefined variables (u), or module, common, or pointer undefined checks (x). These may be combined: -Haesu or -Hsu.
-NRtrap	Outputs the runtime messages if the intrinsic instructions errors and floating point exceptions are trapped (default: -NRnotrap).
-Qt	Outputs the details of the optimization information and the statistics.
-V	Displays the version and release information.

4.4.1.5 **Optimize Fortran Programs with Compiler Options**

The default optimization option of the Fortran compilers is -O2. You need to specify an appropriate optimization option to run a program faster. Recommended selections of Fortran optimization options are as follow.

Example 1) Try the following optimization option first.

```
$ frtpx -Kfastsample.f90
```

Example 2) Optimization option for a program with short innermost loops (in the example below, the number of loops is 6)

```
$ frtpx -Kfast,shortloop=6 sample.f90
```

Example 3) Other optimization options

```
$ frtpx -Kfast,noswp,striping sample.f90
```

Because effects of optimization depend on the programs' characteristics, you need to run programs to verify optimization results. In addition, some optimization options could induce related optimization options.

Table 4-6 Fortran lists main optimization options.

Optimization may affect computation results. If you apply optimization, verify execution of the execution result.

See "Fortran User's Guide 9 Optimization functions" for details.

Table 4-6 Fortran compiler optimization options

Compiler option	Description
-O [0|1|2|3]	Specify the optimization level. The default is the -O2.
-Kdalign	Generate instructions in objects aligned on 8-byte boundaries.
-Kns	Initialize non-standard floating-point mode of operation in the FPU (default: -Knons).
-Kmfunc	Apply optimization with multi-operation functions (default: -Knomfunc).
-Keval	Apply optimization by changing the method of operator evaluation (default: -Knoeval).
-Kprefetch_conditional	Generate prefetch objects for array elements in if statements or case statements.
-Kilfunc	Apply inlining of some single precision and double precision intrinsic functions (default: -Knoilfunc).
-Kfp_contract	Specify whether optimization with Floating-Point Multiply-Add/Subtract operation instructions applies or not.
-Kfp_relaxed	Specify whether reciprocal and approximate calculation applies to a floating point division or SQRT function (default: -Knofp_relaxed).
-Kfast	Option for generating an object program that runs faster on the target machine. Equivalent to -O3 -Kdalign,eval,fp_contract,fp_relaxed,ilfunc,mfunc,ns, omitfp,prefetch_conditional
-Kregion_extension	Extend a parallel region. This option can be specified when -Kparallel option is enabled.
-Kparallel	Specify automatic parallelism (default: -Knoparallel) When -Kparallel is enabled, -O2,-Kregion_extension are induced.
-Kvisimpact	Equivalent to -Kfast,parallel option
-Kocl	Enable optimization control lines (default: -Knoocl).
-Kpreex	Apply pre-examination of invariant expressions (default: -Knopreex).
-Kswp	Apply software pipelining optimization. (default: -Knoswp)
-Kshortloop=N	Apply optimization for short loops (N is between 2 and 10).
-Kstriping[=N]	Apply loop striping optimization. (Default: -Knostriping)
-Karray_private	To facilitate automatic parallelization, privatizing arrays in loops that can be privatized. Effective when -Kparallel option is specified (default: -Knoarray_private).
-Kauto	Treat local variables, except variables having the SAVE attribute and variables with initial values, as automatic variables and place them on the stack.
-Ksimd[=1|2|auto]	Generate objects optimized with SIMD extensions (default: -Ksimd=auto). -Ksimd=1: Generate objects optimized with SIMD extensions -Ksimd=2: In addition to -Ksimd=1, objects with if loops optimized with SIMD extensions. -Ksimd=auto: Determine whether to use SIMD extension for the loop automatically.
-Kopenmp	Activate OpenMP Fortran directives. (default: -Knoopenmp)
-Koptmsg[=1|2]	Set a level of optimization messages output (default: -Koptmsg=1) -Koptmsg=1: Display messages indicating optimization which may have side effects to the execution result are performed. -Koptmsg=2: In addition to -Koptmsg=1, display messages indicating optimization technique such as automatic parallelization, SIMD optimization or loop unrolling are applied.

-KXFILL[=N]	Generate an instruction (XFILL) for securing a cache line on the cache used for writing array data that is written only within loops without loading data from the data memory.

You can control optimization functions by arranging compiler options in different orders. Some examples are described below.

(1) If precision of the execution results varies

In frtpx, the -Kfast option induces -Keval, which modifies the evaluation method of operations. This -Keval option could affect precision sensitive calculations. As for this case, the -Knoeval option can be selected to prevent -Kfast to induce -Keval. Since options on command line are prioritized from the last option, the noeval option should be written after the -Kfast option as shown below.

```
[username@greatwave1 ~]$ frtpx -Kfast,noeval  sample.f90
```

(2) If compilation takes too long

Lower the level of the optimization option.

```
[username@greatwave1 ~]$ frtpx -Kfast,parallel -O2 sample.f90
```

4.4.1.6 **Fortran Environment Variable**

The environment variables that the Fortran compiler commands (*frtpx, mpifrtpx*) recognize are as follows.

(1) FORT90CPX environment variable

You can specify compiler options in FORT90CPX environment variable. The defined options in FORT90CPX are automatically passed onto the compiler. The priority of compilation options is as follows, highest first:

1. Compiler directive lines (the -Koptions is specified)
2. Compilation command operand
3. Environment variable FORT90CPX
4. Profile file
5. Default value

Example) Specify recommended options in FORT90CPX on the front end server.

```
[username@greatwave1 ~]$ export FORT90CPX=-Kfast
```

When you load the settings for libraries using the *module* command, the header search path option (-I) and the library search path option (-L) is appended to FORT90CPX environment variable.

You can check the effective options by the -Q option.

(2) TMPDIR environment variable

You can change the temporary directory used by the *frtpx* and *mpifrtpx* commands.

4.4.1.7 **C/C++ Compiler Option**

The main C/C++ compiler options are the following. See the man page for more information.

Table 4-7 C/C++ Compiler option

Compiler option	Description
-c	Create an object file.
-o *exe_file*	Change the name of an executable file or object file to exe_file. If the name of an executable file is not specified, a.out is created.
-I *directory*	Specify a directory to search for INCLUDE files.
-V	Generate compiler version information to the standard output.
-Xg	Compile codes based on the GNU C compiler specifications. If the -Xg option is specified, the C89 standard is applied for compilation. When specifying both the GNU extensions and the C99 standard, the -noansi option is required. This option is set to the environment variable for compile option as default.
-Xa	Disable -Xg option to compile codes based on the GNU C compiler specifications.
-NRtrap	Specify whether runtime trapping is detected or not (default: -NRnotrap).
-Nsrc	Generate a source list.
-Nsta	Generate statistic information.

4.4.1.8 **Optimize C/C++ Programs with Compiler Options**

The default optimization option of the C/C++ compilers is -O0. You need to specify an appropriate optimization option to run a program faster. Recommended selections of C/C++ optimization options are as follow.

Example 1) Try the following optimization option first.

```
$ fccpx -Kfastsample.c
```

Example 2) Optimization option for a program with short innermost loops. (In the example below, the number of loops is 6)

```
$ fccpx -Kfast,shortloop=6 sample.c
```

Example 3) Other optimization options.

```
$ fccpx -Kfast,noswp,striping sample.c
```

Because effects of optimization depend on the programs' characteristics, you need to run programs to verify optimization results. In addition, some optimization options could induce related optimization options. Table 4-8 C/C++ lists main optimization options.

Optimization may affect computation results. If you apply optimization, verify execution of the execution result.

See "C User's Guide 3 Serial Optimization functions" or "C++ User's Guide 3 Serial optimization functions" for details.

Table 4-8 C/C++ compiler optimization options

Compiler option	Description
-O[0|1|2|3]	Specify the optimization level. The default is the –O0.
-Kdalign	Generate instructions in objects aligned on 8-byte boundaries.
-Kns	Initialize non-standard floating-point mode of operation in the FPU (default: -Knons).
-Kmfunc	Apply optimization with multi-operation functions (default: -Knomfunc).
-Klib	Specify whether optimization is applied or not by checking operation of the standard library function (default: -Knolib).
-Keval	Apply optimization by changing the method of operator evaluation (default: -Knoeval).
-Krdconv	Specify the optimizer to assume that 4-byte signed integer loop variable do not cause overflows.
-Kprefetch_conditional	Generate prefetch objects for array elements in if statements or case statements.
-Kilfunc	Apply inlining of some single precision and double precision intrinsic functions.
-Kfp_contract	Specify whether optimization with Floating-Point Multiply-Add/Subtract operation instructions applies or not.
-Kfp_relaxed	Specify whether reciprocal and approximate calculation applies to a floating point division or SQRT function.
-x	Apply inlining of all functions defined in the source program.
-Kfast	Option for generating an object program that runs faster on the target machine. Equivalent to -O3 -Kdalign,eval,fast_matmul,fp_contract, fp_relaxed,ilfunc,lib,mfunc,ns,omitfp,prefetch_conditional,rdconv -x
-Kregion_extension	Extend a parallel region. This option can be specified when -Kparallel option is enabled.
-Kparallel	Specify automatic parallelism (default: -Knoparallel) When -Kfast is enabled, -O2 -Kregion_extension -mt option are induced.
-Kvisimpact	Equivalent to -Kfast, parallel option
-Kocl	Enable optimization control lines (default: -Knoocl).
-Kpreex	Apply pre-examination of invariant expressions (default: -Knopreex).
-Karray_private	To facilitate automatic parallelization, privatizing arrays in loops that can be privatized.
-Kopenmp	Activate OpenMP C directives (default: -Knoopenmp)
-Ksimd[=1|2|auto]	Generate objects optimized with SIMD extensions (default: -Ksimd=1). -Ksimd=1: Generate objects optimized with SIMD extensions -Ksimd=2: In addition to -Ksimd=1, objects with if loops optimized with SIMD extensions. -Ksimd=auto: Determine whether to use SIMD extension for the loop automatically.
-Koptmsg[=1|2]	Set a level of optimization messages output (default: -Koptmsg=1) -Koptmsg=1: Display messages indicating optimization which may have side effects to the execution result are performed. -Koptmsg=2: In addition to -Koptmsg=1, display messages indicating optimization technique such as automatic parallelization, SIMD optimization or loop unrolling are applied.
-Kswp	Apply software pipelining optimization. (default: -Knoswp)
-Kshortloop=N	Apply optimization for short loops (N is between 2 and 10).
-Kstriping[=N]	Apply loop striping optimization. (Default: -Knostriping)

-KXFILL[=N]	Generate an instruction (XFILL) for securing a cache line on the cache used for writing array data that is written only within loops without loading data from the data memory (default: -KNOXFILL). This option can be specified when -O2 or higher is specified.

4.4.1.9 **C Environment Variable**

The environment variables that the C compiler commands (*fccpx*, *mpifccpx*) recognize are as follows.

(1) fccpx_ENV environment variable

You can specify compiler options in fccpx_ENV. The defined compiler options in fccpx_ENV are automatically passed onto the compiler. The priority of compilation options is as follows, highest first:

1. Compilation command operand
2. Environment variable fccpx_ENV
3. Profile file
4. Default value

Example) Specify recommended options in fccpx_ENV on the front end server.

```
[username@greatwave1 ~]$ export fccpx_ENV="$fccpx_ENV -Kfast"
```

When you load the settings for libraries using the *module* command, the header search path option (-I) and the library search path option (-L) is appended to fccpx_ENV environment variable.

You can check the effective options by the -Nsta option.

(2) TMPDIR environment variable

You can change the temporary directory used by the *fccpx* and *mpifccpx* commands.

4.4.1.10 **C++ Environment Variable**

The environment variables that the C++ compiler commands (*FCCpx*, *mpiFCCpx*) recognize are as follows.

(1) FCCpx_ENV environment variable

You can specify compiler options in FCCpx_ENV. The defined compiler options in FCCpx_ENV are automatically passed onto the compiler. The priority of compilation options is as follows, highest first:

1. Compilation command operand
2. Environment variable FCCpx_ENV
3. Profile file
4. Default value

Example) Specify recommended options in FCCpx_ENV on the front end server.

```
[username@greatwave1 ~]$ export FCCpx_ENV="$FCCpx_ENV -Kfast"
```

When you load the settings for libraries using the *module* command, the header search path option (-I) and the library search path option (-L) is appended to FCCpx_ENV environment variable.

You can check the effective options by the -Nsta option.

(2) TMPDIR environment variable

You can change the temporary directory used by the *FCCpx* and *mpiFCCpx* commands.

4.4.1.11 **Native Compiler for Massively Parallel Computer**

The native compilers for the Massively Parallel Computer are available in the batch job or interactive job. Please note that compiler commands and environment variables for native compilers are different from for cross compilers.

Table 4-9 Native compiler commands for the Massively Parallel Computer

	Programing language	**Cross compiler (Front end servers)**	**Native compiler (Massively Parallel Computer)**
Sequential (Non MPI)	Fortran	frtpx	frt
	C	fccpx	fcc
	C++	FCCpx	FCC
MPI parallel	Fortran	mpifrtpx	mpifrt
	C	mpifccpx	mpifcc
	C++	mpiFCCpx	mpiFCC

Table 4-10 Environment variables of native compiler for the Massively Parallel Computer

Programing language	**Cross compiler (Front end servers)**	**Native compiler (Massively Parallel Computer)**
Fortran	FORT90CPX	FORT90C
C	fccpx_ENV	fcc_ENV
C++	FCCpx_ENV	FCC_ENV

4.4.2 Compile and Link for Application Computing Server

4.4.2.1 Compile and link sequential programs

To compile and link sequential programs for the Application Computing Server on the front end servers, use the *ifort*/*icc*/*icpc* command.

```
ifort/icc/icpc[option] file [...]
```

Example 1) Compile and link a sequential Fortran program for the Application Computing Server.

```
[username@greatwave1 ~]$ ifort seq.f
```

Example 2) Compile and link a sequential C program for the Application Computing Server.

```
[username@greatwave1 ~]$ icc seq.c
```

4.4.2.2 Compile and link thread parallelization programs

To compile and link multi-threaded programs for the Application Computing Server on the front end servers, use the *ifort*/*icc*/*icpc* command.

```
ifort/icc/icpc thread-option [option] file [...]
```

Example 1) Compile and link a Fortran program with automatic parallelization for the Application Computing Server.

```
[username@greatwave1 ~]$ ifort -parallel para.f
```

Example 2) Compile and link a C program with automatic parallelization for the Application Computing Server.

```
[username@greatwave1 ~]$ icc -parallel para.c
```

Example 3) Compile and link an OpenMP Fortran program for the Application Computing Server.

```
[username@greatwave1 ~]$ ifort -openmp omp.f
```

Example 4) Compile and link an OpenMP C program for the Application Computing Server.

```
[username@greatwave1 ~]$ icc -openmp omp.c
```

Example 5) Compile and link an OpenMP Fortran program with automatic parallelization for the Application Computing Server.

```
[username@greatwave1 ~]$ ifort -parallel -openmp omp_para.f
```

Example 6) Compile and link an OpenMP C program with automatic parallelization for the Application Computing Server.

```
[username@greatwave1 ~]$ icc -parallel -openmp omp_para.c
```

4.4.2.3 **Compile and link MPI programs**

To compile and link MPI programs for the Application Computing Server on the front end servers, use the *mpiifort*/*mpiicc*/*mpiicpc* command.

```
mpiifort/mpiicc/mpiicpc [option] file [...]
```

Example 1) Compile and link a MPI Fortran program for the Application Computing Server.

```
[username@greatwave1 ~]$ mpiifort mpi.f
```

Example 2) Compile and link a MPI C program for the Application Computing Server.

```
[username@greatwave1 ~]$ mpiicc mpi.c
```

Example 3) Compile and link a Hybrid (MPI + OpenMP) Fortran program for the Application Computing Server.

```
[username@greatwave1 ~]$ mpiifort -openmp mpi_omp.f
```

Example 4) Compile and link a Hybrid (MPI + OpenMP) C program for the Application Computing Server.

```
[username@greatwave1 ~]$ mpiicc -openmp mpi_omp.c
```

4.4.2.4 **Optimize for the Application Computing Server**

Optimization may affect computation results. If you apply optimization, verify execution of the execution result.

Figure 4-11 General optimization option

Compile Options	Description
-O0	Disables all optimizations.
-O1	Enables optimizations for speed and disables some optimizations that increase code size and affect speed. To limit code size, this option.
-O2	Enables optimizations for speed. This is the generally recommended optimization level. Vectorization is enabled at -O2 and higher levels.
-O3	Performs -O2 optimizations and enables more aggressive loop transformations such as Fusion, Block-Unroll-and-Jam, and collapsing IF statements.
-fast	Maximizes speed across the entire program. It sets the following options: -ipo, -O3, -no-prec-div, -static, -fp-model fast=2, -xHost * The -static option is not available when linking MPI programs.
-qopt-report[=*n*]	Tells the compiler to generate an optimization report. You can specify values 0 through 5 as *n*. The default is level 2.
-qopt-report-phase[=*list*]	Specifies one or more optimizer phases for which optimization reports are generated. For more detail, refer to man manual.
-qopt-report-help	Displays the optimizer phases available for report generation and a short description of what is reported at each level. No compilation is performed.
-qopt-report-routine=string	Tells the compiler to generate an optimization report for each of the routines whose names contain the specified substring. When optimization reporting is enabled, the default is -qopt-report-phase=all.

Table 4-12 Parallel peformance options

Compile Options	Description
-openmp	Enables the parallelizer to generate multi-threaded code based on OpenMP directives.
-parallel	Tells the auto-parallelizer to generate multi-threaded code for loops that can be safely executed in parallel.
-par-threshold[*n*]	Sets a threshold for the auto-parallelization of loops. (from n=0 to n=100. Default: n=100)。 0 – Loops get auto-parallelized always, regardless of computation work volume. 100 – Loops get auto-parallelized when performance gains are predicted based on the compiler analysis data. Loops get auto-parallelized only if profitable parallel execution is almost certain. To use this option, you must also specify option -parallel.
-guide[=*n*]	Lets you set a level of guidance for auto-vectorization, auto parallelism, and data transformation. When this option is specified, the compiler does not produce any objects or executables. You must also specify the -parallel option to receive auto parallelism guidance. The values available are 1 through 4. Value 1 indicates a standard level of guidance. Value 4 indicates the most advanced level of guidance. If n is omitted, the default is 4.
-qopt-matmul	Enables or disables a compiler-generated Matrix Multiply (matmul). The -qopt-matmul options tell the compiler to identify matrix multiplication loop nests (if any) and replace them with a matmul library call for improved performance. The resulting executable may get additional performance gain. This option is enabled by default if options -O3 and -parallel are specified. To disable this optimization, specify -qno-opt-matmul. This option has no effect unless option O2 or higher is set.
-coarray=shared	Enables the coarray feature of the Fortran 2008 standard.

Table 4-13 Processor-specific optimization options

Compile Options	Description
-x*target*	Tells the compiler which processor features it may target, including which instruction sets and optimizations it may generate. When you build only for ACS with GPU, specify option -xCORE-AVX2 for the Haswell microarchitecture.
-xhost	Tells the compiler to generate instructions for the highest instruction set available on the compilation host processor.

Table 4-14 Interprocedural Optimization (IPO) options and Profile-guided Optimization (PGO) options

Compile Options	Description
-ip	Determines whether additional interprocedural optimizations for single-file compilation are enabled.
-ipo[=*n*]	Enables interprocedural optimization between files. If n is 0, the compiler decides whether to create one or more object files based on an estimate of the size of the application. It generates one object file for small applications, and two or more object files for large applications. If you do not specify n, the default is 0.
-ipo-jobs[*n*]	Specifies the number of commands (jobs) to be executed simultaneously during the link phase of Interprocedural Optimization (IPO). The default is -ipo-jobs1.
-finline-functions -finline-level=2	Enables function inlining for single file compilation. Interprocedural optimizations occur. if you specify -O0, the default is OFF.
-finline-factor=*n*	Specifies the percentage multiplier that should be applied to all inlining options that define upper limits. The default value is 100 (a factor of 1).
-prof-gen	Produces an instrumented object file that can be used in profile-guided optimization.
-prof-use	Enables the use of profiling information during optimization.
-profile-functions	Inserts instrumentation calls at a function's entry and exit points.
-profile-loops	Inserts instrumentation calls at a function's entry and exit points, and before and after instrumentable loops.

Figure 4-15 Floating-point operation optimization options

Compile Options	Description
-fp-model *name*	Controls the semantics of floating-point calculations.
-ftz[-]	Flushes denormal results to zero when the application is in the gradual underflow mode. It may improve performance if the denormal values are not critical to your application's behavior.
-fimf-precision:*name*	Lets you specify a level of accuracy (precision) that the compiler should use when determining which math library functions to use. The *name* is high, medium or low. This option can be used to improve run-time performance if reduced accuracy is sufficient for the application, or it can be used to increase the accuracy of math library functions selected by the compiler. In general, using a lower precision can improve run-time performance and using a higher precision may reduce run-time performance.
-fimf-arch-consistency=*true*	Ensures that the math library functions produce consistent results across different microarchitectural implementations of the same architecture. The -fimf-arch-consistency option may decrease run-time performance. Deafult is "false”.
-prec-div	Improves precision of floating-point divides. The result is more accurate, with some loss of performance.
-prec-sqrt	Improves precision of square root implementations. The result is fully precise square root implementations, with some loss of performance.

Figure 4-16 Detailed tuning options

Compile Options	Description
-unroll[*n*]	Tells the compiler the maximum number of times to unroll loops. To disable loop enrolling, specify 0. The default is -unroll, and the compiler uses default heuristics when unrolling loops.
-qopt-prefetch[=*n*]	Enables or disables prefetch insertion optimization. The n (0:Disable-4) is the level of software prefetching optimization desired. The option -qopt-prefetch=3 is enabled by default if option -O2 or higher is set.
-qopt-block-factor=*n*	Lets you specify a loop blocking factor.
-qopt-streaming-stores *mode*	This option enables generation of streaming stores for optimization. The *mode* is as follows: "always": Enables generation of streaming stores for optimization. The compiler optimizes unde the assumption that the application is memory bound. "never": Disables generation of streaming stores for optimization. "auto": Lets the compiler decide which instructions to use.
-fno-alias	Determines whether aliasing should be assumed in the program. Default is -fno-alias.
-fno-fnalias	Specifies that aliasing should be assumed within functions. Default is -ffnalias.
-fexceptions	Enables exception handling table generation. This option enables C++ exception handling table generation, preventing Fortran routines in mixed-language applications from interfering with exception handling between C++ routines. The -fno-exceptions option disables C++ exception handling table generation, resulting in smaller code. When this option is used, any use of C++ exception handling constructs (such as try blocks and throw statements) when a Fortran routine is in the call chain will produce an error.
-vec-threshod *n*	Sets a threshold for the vectorization of loops based on the probability of profitable execution of the vectorized loop in parallel. (from n=0 to n=100. Default: n =100) 0 – loops get vectorized always, regardless of computation work volume. 100 – loops get vectorized when performance gain are predicted based on the compiler analysis data.
-vec-report[=*n*]	Controls the diagnostic information reported by the vectorizer. The n is a value denoting which diagnostic messages to report. Default is 0.

4.4.3 Compile and Link CUDA programs for ACS with GPU

To compile and link CUDA programs for the ACS with GPU on the front end servers, use the *nvcc* command.

```
nvcc [option] file [...]
```

4.5 Numerical Libraries

As for numerical libraries, BLAS/LAPACK/ScaLAPACK are available. These libraries are tuned for the Massively Parallel Computer and the Application Computing Server.

When using numerical libraries with C/C++ compilers for the Massively Parallel Computer, refer to the product manuals of each numerical library.

4.5.1 BLAS/LAPACK/ScaLAPACK for Massively Parallel Computer

When you use BLAS/LAPACK/ScaLAPACK libraries for the Massively Parallel Computer, the following options are available:

Table 4-17 BLAS/LAPACK/ScaLAPACK options

Library	Parallelism	Option	Remark
BLAS	Sequential	-SSL2	
	Thread parallel	-SSL2BLAMP	Use with -Kopenmp.
LAPACK	Sequential	-SSL2	
	Thread parallel	-SSL2BLAMP	Use with -Kopenmp.
ScaLAPACK	MPI parallel	-SCALAPACK	Depending on BLAS/LAPACK to link, specify either -SSL2 or -SSL2BLAMP.

Example 1) Use the sequential version BLAS/LAPACK.

```
$ frtpx -SSL2 blas.f
```

Example 2) Use the thread parallel version BLAS/LAPACK.

```
$ frtpx -Kopenmp -SSL2BLAMP blas.f
```

Example 3) Use ScaLAPACK (linking the sequential version of BLAS/LAPACK).

```
$ mpifrtpx -SCALAPACK -SSL2 scalapack.f
```

Example 4) Use ScaLAPACK (linking the thread parallel version of BLAS/LAPACK).

```
$ mpifrtpx -Kopenmp -SCALAPACK -SSL2BLAMP scalapack.f
```

4.5.2 SSL II (Scientific Subroutine Library II) Numerical Library

SSL II library can be used as a numerical library for the Massive Parallel Computer. You can also use C-SSL II library for the C/C++ compilers.

Table 4-18 SSL II numerical libraries overview

Library	Description
SSL II	* Thread-safe numerical library for sequential computing. * Subroutines for 10 fields (linear computation, eigenvalue and eigenvector, non-linear computation, extremal problem, interpolation and approximation, conversion, numerical differentiation and integration, differential equation, special functions, pseudo-random numbers) etc.
SSL II thread parallel functions	* Parallel numerical algorithm with an interface conforming with SMP parallel processing for important functions suitable for parallelism. * Linear computation (direct and iterative methods for linear equations, inverse matrix, eigenvalue problem, etc.), Fourier transform, pseudo-random numbers, etc.
C-SSL II	* The sequential functions subset of Fortran version of SSL II can be used with a C language interface. * Thread-safe serial functions
C-SSL II thread parallel functions	* The thread parallel functions subset of Fortran version of SSL II can be used with a C language interface.
SSL II/MPI	* Fujitsu's original, MPI parallel 3-dimensional Fourier transform routine
High-speed quad-precision basic arithmetic library	* Library for high-speed processing of quad-precision values in double-double format

The lists of compiler options are the following. Although SSL II (C-SSL II) library has serial and thread parallel functions, the users can use both functions together because sub-routine names are different.

Table 4-19 SSL II options

Library	Parallelism	Option	Remark
SSL II C-SSL II	Sequential	-SSL2	
	Thread parallel		Use with -Kopenmp.
SSL II/MPI	MPI parallel	-SSL2MPI	Use with -Kopenmp. Depending on BLAS/LAPACK used, specify either -SSL2 or -SSL2BLAMP.

Example 1) Use the sequential version SSL II.

```
$ frtpx -SSL2 ssl2seq.f
```

Example 2) Use the thread parallel version SSL II.

```
$ frtpx -Kopenmp -SSL2 ssl2para.f
```

Example 3) Use the sequential version C-SSL II.

```
$ fccpx -SSL2 cssl2.c
```

Example 4) Use SSL II/MPI.

```
$ mpifrtpx -Kopenmp -SSL2MPI -SSL2 ssl2mpi.f
```

4.5.3 BLAS/LAPACK/ScaLAPACK for Application Computing Server

When you use BLAS/LAPACK/ScaLAPACK libraries for the Application Computing Server, the following options are available:

Table 4-20 BLAS/LAPACK/ScaLAPACK options

Library	Parallelism	Option	Remark
BLAS	Sequential	-mkl=sequential	
	Thread parallel	-mkl=parallel	
LAPACK	Sequential	-mkl=sequential	
	Thread parallel	-mkl=parallel	
ScaLAPACK	MPI parallel	-mkl=cluster	

Example 1) Use the sequential version BLAS/LAPACK.

```
$ ifort -mkl=sequential blas.f
```

Example 2) Use the thread parallel version BLAS/LAPACK.

```
$ ifort -mkl=parallel blas.f
```

Example 3) Use ScaLAPACK (linking the sequential version of BLAS/LAPACK).

```
$ mpiifort -mkl=cluster scalapack.f
```

About the combinations than the above, refer to the following URL:

https://software.intel.com/en-us/articles/intel-mkl-link-line-advisor

5. Batch Job and Interactive Job

5.1 Job Management System

The job management system manages all batch jobs and interactive jobs over the HOKUSAI GreatWave system. Users request job requirements such as resource unit, resource group, number of nodes, number of cores, and elapsed time to the job management system for the job to be executed.

On the Massively Parallel Computer, by applying non-contiguous mode for node assignment, free computing resources (cores, memory) are aggressively allocated to the jobs without considering node shape (Torus/Mesh).

There are two types of jobs users can submit to the HOKUSAI GreatWave system.

Table 5-1 Job types

Job type	Usage
Batch job	Execute jobs in batch mode. When a node failure occurs, your job is re-executed if the --restart option is given.
Interactive job	Execute jobs in interactive mode by entering data on user terminals. Mainly used for debugging. Jobs are not re-executed when an error such as a node failure occurs.

Batch jobs can be classified into three types depending on how they are submitted.

Table 5-2 Batch job types

Job classification	Purpose	How to submit
Normal job	Execute jobs based on a job script.	Refer to "5.4.1 Normal "
Step job	Handle multiple jobs as a group having the execution order or dependency relationship.	Refer to "0 Step Job"
Bulk job	Consist of multiple instances of the same normal job submitted at the same time for execution.	Refer to "5.4.3 Bulk Job"

Users can use the following commands to run jobs.

Table 5-3 Job commands

Function	Command
Submit a job	pjsub
See a job status	pjstat
Delete a job	pjdel
Display a job script	pjcat

5.2 Job Execution Resource

When submitting a job, specify the “resource unit” that means the hardware where a job runs and the “resource group” that means the software.

5.2.1 Resource Unit

The following three types of resource units that specify the hardware a job runs are prepared:

Table 5-4 Resource units

Resource Unit	Where the job is executed
gwmpc	Massively Parallel Computer (MPC)
gwacsl	ACS with GPU (ACSG)
gwacsg	ACS with Large memory (ACSL)

Although specification of the resource unit is mandatory, the following file can be used to contain the settings by which the resource unit to be used is fixed. The settings in this file are ignored when the resource unit is explicitly specified in the job script or on the command line.

Table 5-5 Fixing the resource unit to be used

Setting file name	Setting value
/home/*username*/.cltkrc	GW_DEF_RSCUNIT="resource unit name"

Example)

```
[username@greatwave1 ~] cat $HOME/.cltkrc
GW_DEF_RSCUNIT=gwmpc
```

5.2.1.1 **Resource Unit settings for Project**

Each project can use the following resources at a time.

Table 5-6 Concurrent resource usage limit for Project

Resource Unit	Number of running cores	Number of running nodes	Number of submitted jobs	Number of submitted bulk subjobs
gwmpc	General: 8,192 Quick: 1,024	General: 256 Quick: 32	500	5,000
gwacsl	General: 120 Quick: 60	2	100	100
gwacsg	General: 720 Quick: 96	30	100	100

5.2.2 Resource Group

The resource groups that specify the software to be executed are prepared on each resource unit. If you run an ISV application, specify an appropriate resource group when submitting a job. With some resource group, the user of general subject is allowed to use more resources than the user of simple subject.

When no resource group is specified, the following resource group is automatically selected by default.

Table 5-7 Default resource group

Job type	Default resource group
Batch job	batch
Interactive job	interact

The following section describes the list of the available resource group.

5.2.2.1 **Resource Group for Massively Parallel Computer (MPC)**

Table 5-8 Resource group for Massively Parallel Computer

Resource Group	Specific use	Job Type	Maximum elapsed time[*2]	Maximum number of cores	Maximum number of nodes
batch	Gerenal job	Batch	72hrs	512	16
			24hrs	General:8,192 Quick:1,024	General:256 Quick: 32
gaussian	Gaussian	Batch	72hrs	32	1
interact[*2]	Interactive use	Interactive	2hrs	128	4
special[*1]	Large scale parallel	Batch	48hrs	34,560	1,080

*1 Application is required. (See the section 5.2.2.4)

*2 A user can submit and run only 1 interactive job at the same time.

*3 Default elapsed time for batch jobs is 12hrs.

5.2.2.2 **Resource Group for ACS with Large memory (ACSL)**

Table 5-9 Resource Group for ACS with Large memory (ACSL)

Resource Group	Specific use	Job Type	Maximum elapsed time *2	Maximum number of cores	Maximum number of nodes
batch	Gerenal job	Batch	24hrs	General:120 Quick: 60	2
ansys	ANSYS	Batch	12hrs	1	1
gaussian*1	Gaussian	Batch	24hrs	60	1
vm*1	Gaussian	Batch	720hrs	16	1
interact*2	Interactive use	Interactive	2hrs	General:120 Quick: 60	2
special*1	Large scale parallel	Batch	48hrs	120	2

*1 Application is required. (See the section 5.2.2.4)

*2 A user can submit and run only 1 interactive job at the same time.

*3 Default elapsed time for batch jobs is 12hrs.

5.2.2.3 **Resource Group for ACS with GPU (ACSG)**

Table 5-10 Resource Group for ACS with GPU (ACSG)

Resource Group	Specific use	Job Type	Maximum elapsed time*2	Maximum number of cores	Maximum number of nodes
batch	Gerenal job	Batch	72hrs	12	1
gpu	GPU job				
amber	Amber				
amber_gpu	Amber(GPU)		24hrs	General:720 Quick: 96	30
adf	ADF				
qchem	Q-Chem				
matlab*1	MATLAB	Batch	12hrs	24	1
gaussian	Gaussian	Batch	24hrs	24	1
vm*1	Gauusian	Batch	720hrs	16	1
interact*2	Interactive use	Interactive	2hrs	General:720 Quick: 96	30
interact_gpu *2	Interactive use GPU job	Interactive	2hrs	General:720 Quick: 96	30
special*1	Large scale parallel	Batch	48hrs	720	30

*1 Application is required. (See the section 5.2.2.4)

*2 A user can submit and run only 1 interactive job at the same time in the whole ACS with GPU (ACSG) resource unit.

*3 Default elapsed time for batch jobs is 12hrs.

5.2.2.4 **Resource Group for Which Submitting an Application Required**

To use some resource groups, it is required to submit an application.

Table 5-11 Resource groups for which submitting an application is required

Resource Group	Description
special	Large scale parallel jobs that are not allowed to run during the regular operation are allowed to be executed during the specific period specified by ACCC.
matlab	To run a MATLAB, the user must have the following license. Sequential job: MATLAB Parallel job: Parallel Computing Toolbox
vm	To run a Gaussian for long time on the VM (Virtual Machine) . The limitations the resources are as follows: ● Maximum number of nodes: 1 ● Maximum number of cores: 16 ● Maximum amount of memory: 64GB
ansys	ANSYS (multiphysics) can be executed for only one job simultaneously in the HOKUSAI GreatWave system.

The user who wants to use above resource groups should prepare the following information and contact hpc@riken.jp

- User name, Project ID, Period
- Reason

5.3 Job Submission Options

When submitting jobs, specify the following three options as necessary.

- Basic Options
- Resource Options
- MPI Options (only for Massively Parallel Computer)

5.3.1 Basic Options

The basic options you can specify to your job are the following.

Table 5-12 Basic options for submitting a job

Option	Description
-g *projectID*	Specify a project ID that consumes core time to execute a job
-j	Direct output of standard error of the job to standard output
--mail-list	Set an email address
-m	Set email notification
b	Send email notification on starting a job
e	Send email notification on finishing a job
r	Send email notification on re-executing a job
-N *name*	Specify a job name
-o *filename*	Write standard out put to a specified file
-p *priority*	Job priority (Only available within user own jobs)
--restart	Specify a job not to be re-executed when a failure occurs (default: --norestart)
--interact	Submit a job as an interactive job
--step	Submit a job as a step job (This option is not available when "vm" is specified as resource group)
jid=*jobid*	Set a job ID to associate with
sn=*subjobid*	Set an own sub-job number
sd=*form*	Set a dependency statement
--bulk --sparam start-end	Submit a job as a bulk job
-X	Inherit environment variables for used for job submission to the job execution environment
-s	Output job statistic information when finishing a job
-S	Output detail information of each node after items specified by the -s option

5.3.2 Resource Options

You can specify the resources to be allocated to a job by using the -L option.

Table 5-13 Resource options (common)

Option		Description
-L		Specify upper limits of resources needed to execute jobs
	rscunit=*name*	Specify resource unit (required option)
	rscgrp=*name*	Specify resource group
	elapse=*elapselimit*	Specify elapsed time ([[hour:]minute:]second)
	proc-core=*size*	Specify a maximum core file size limit for a process (default: 0, maximum: unlimited)
	proc-data=*size*	Specify a maximum data segment size limit for a process (default: unlimited)
	proc-stack=*size*	Specify a maximum stack segment size limit for a process (default: 32Gi (Massively Parallel Computer), unlimited (Application Computing Server))

The resource options only for the Massive Parallel Computer are as follows:

Table 5-14 Resource options (only Massively Parallel Computer)

Option		Description
-L		Specify upper limits of resources needed to execute jobs
	node=*num*	For multinode(33 cores or more) job Specify the number of nodes (Node-exclusive)
	vnode=([core=*num*][;core-mem=*size*])	For single node(32 cores or less) jobs Specify the number of cores (maximum: 32) and the amount of memory per core(maximum: 30Gi) (Node-sharing)

When you request the number of nodes using the -L node option, node is exclusively allocated to the job, and all cores and job memory are available.

The resource options only for the Application Computing Server are as follows:

Table 5-15 Resource options (only Application Computing Server)

Option		Description
-L		Specify upper limits of resources needed to execute jobs
	vnode=*num*	Specify the number of nodes
	vnode-core=*num*	Specify the number of cores per node. - Maximum number of ACSL: 60 - Maximum number for ACSG: 24
	core-mem=*size*	Specify the amount of memory per core - Maximum amount of ACSL: 960Gi - Maximum number of ACSG: 60,000Mi

You can use GPU on GPU application server with the following options.

Table 5-16 Resource options for GPU job

オプション		説明
-L		Specify resource usage
	rscunit=gwacsg	Specify "gwacsg"
	rscgrp=*name*	Specify resource group name ("gpu" or resource gropu for GPU application)
	vnode=*num*	Specify number of node (currently supported 1 node job only)
-x		Specify environment variables
	gpu_per_vnode=*num*	Specify number of GPU to use (max: 4)

When you set the amount of memory, the units can be set as following string:

Table 5-17 Available unit for the amount of memory

Unit	Description
Ki	kibibyte (2^10)
Mi	mebibyte (2^20)
Gi	gibibyte (2^30)

The default amount of memory per core is as follows:

Table 5-18 Default amount of memory per core

System	Default amount of memory per core
Massively Parallel Computer (MPC)	960Mi
ACS with Large memory (ACSL)	16Gi
ACS with GPU (ACSG)	2,500Mi

When you require the memory more than default amount of memory per core, more cores could be allocated based on required memory. Be aware that the core time is calculated based on the number of allocated cores and elapsed time.

Example) Request 3840Mi as amount of memory per core for the Massively Parallel Computer

```
[username@greatwave1 ~]$ pjsub --interact -L rscunit=gwmpc -L
"vnode=(core=1;mem=3840Mi)" -g Q99999
pjsub: WARNING: submitted job uses more cpu-core than specified due to
the size of memory. (1 -> 4)
[INFO] PJM 0000 pjsub Job 29774 submitted.
[INFO] PJM 0081 .connected.
[INFO] PJM 0082 pjsub Interactive job 29774 started.
[username@greatwave1 ~]$ numactl --show
policy: default
preferred node: current
physcpubind: 0 1 2 3        ← 4cores are allocated
cpubind: 0
nodebind: 0
membind: 0 1
```

When you don't specify the number of processes/threads with the MPI options/OMP_NUM_THREADS environment variable, the program may run with the unintended number of processed/threads and the performance may degrade.

5.3.3 MPI options (only for Massively Parallel Computer)

In the Massively Parallel Computer, the options of an MPI job are the following. You can specify the --mpi option followed by the execution parameter.

Table 5-19 MPI options

Option		Description
--mpi		Set various parameters for MPI jobs
	proc=*num*	Set the maximum number of statically invoked processes (this is required for multiple nodes and multiple processes per node)
	rank-map-bynode[=*rankmap*]	Once one process is generated for a node, move to the next node and allocate a rank with round-robin allocation (exclusive option to the rank-map-bychip option)
	rank-map-bychip[:*rankmap*]	Once [the number of nodes equivalent to proc divided by shape] is generated, move to the next node and allocate a rank (exclusive option to the rank-map-bynode option)(default)
	rank-map-hostfile=*filename*	Allocate a rank for process to be generated according to filename

5.4 Submit Batch Jobs

To execute a batch job, the user creates a "job script" in addition to a program and submits the job script to the job management system as a batch job. The description of a command line includes options such as a resource unit, a resource group, elapsed time and the number of nodes as well as commands to be executed. The user uses the *pjsub* command to submit a job script. The submitted jobs are automatically executed by the job management system based on the status of free computing resources and the priority among projects.

5.4.1 Normal Job

To submit a normal job, use the *pjsub* command with the job script which is executed as a batch job.

```
pjsub [option] [job-script]
```

- If a job script is not specified, a script is read from standard input.
- Job submission options can be set by defining directives in a job script or in standard input.
- If a job is successfully submitted, an identification number (job ID) is assigned to the job.

Example) Submit a normal job.

```
[username@greatwave1 ~]$ pjsub  run.sh
[INFO]PJM 0000 pjsub Job 12345 submitted.
```

5.4.2 Step Job

A step job is a job model that aggregates multiple batch jobs and defines a job chain having an execution order and dependency of the batch jobs. A step job consists of multiple sub-jobs, which are not executed concurrently. The figure below outlines the process sequence of a step job.

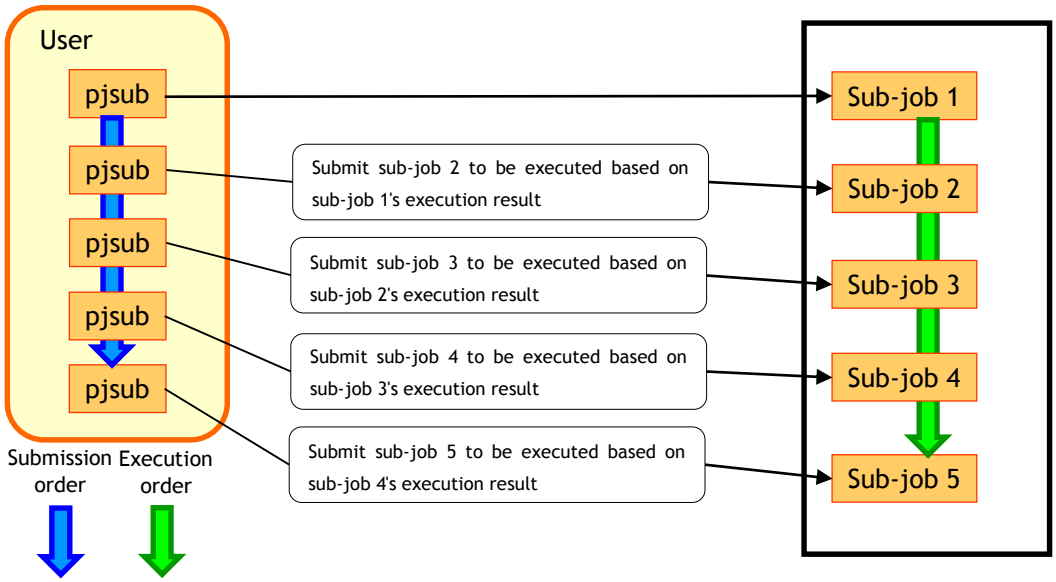


Figure 5-1 General flow of a step job

The format of submitting a step job is as follows:

```
pjsub    --step    [--sparam    "sn=stepno[,dependency]"]    jobscript
```

Table 5-20 Step job dependency statements

Condition	Description
NONE	Indicate no dependency
Termination status == value[,value,value..] Termination status != value[,value,value..] Termination status > value Termination status >= value Termination status < value Termination status <= value	Value can be any number For "==" and "!=", multiple values separated with a comma can be specified. Example: ec==1,3,5 → True if the termination status is 1, 3 or 5 ec!=1,3,5 → True if the termination status is not 1, 3 or 5

Table 5-21 Cancellation types available for step job dependency statements

Cancellation type	Description
one	Cancel only the current job
after	Cancel the current job and recursively cancel jobs dependent on the current job
all	Cancel the current job and all subsequent jobs

Example 1) Submit a step job containing three sub-jobs

```
[username@greatwave1 ~]$ pjsub --step step1.sh
[INFO]PJM 0000 pjsub Job 12345_0 submitted.
[username@greatwave1 ~]$ pjsub --step --sparam jid=12345 step2.sh
[INFO]PJM 0000 pjsub Job 12345_1 submitted.
[username@greatwave1 ~]$ pjsub --step --sparam jid=12345 step3.sh
[INFO]PJM 0000 pjsub Job 12345_2 submitted.
```

Example 2-1) Submit a step job containing three sub-jobs at a time (When a failure occurred, the affected job is failed and the following jobs will be continued.)

```
[username@greatwave1 ~]$ pjsub --step step1.sh step2.sh step3.sh
[INFO]PJM 0000 pjsub Job 12345_0 submitted.
[INFO]PJM 0000 pjsub Job 12345_1 submitted.
[INFO]PJM 0000 pjsub Job 12345_2 submitted.
```

Example 2-2) Submit a step job containing three sub-jobs at a time (When a failure occurred, the affected job is run again.)

```
[username@greatwave1 ~]$ pjsub --step --restart step1.sh step2.sh ¥
step3.sh
[INFO]PJM 0000 pjsub Job 12345_0 submitted.
[INFO]PJM 0000 pjsub Job 12345_1 submitted.
[INFO]PJM 0000 pjsub Job 12345_2 submitted.
```

Example 2-3) Submit a step job containing three sub-jobs at a time (When a failure occurred, the affected job and the following jobs are canceled.)

```
[username@greatwave1 ~]$ pjsub --step --sparam "sd=pc!=0:all" ¥
step1.sh step2.sh step3.sh
[INFO]PJM 0000 pjsub Job 12345_0 submitted.
[INFO]PJM 0000 pjsub Job 12345_1 submitted.
[INFO]PJM 0000 pjsub Job 12345_2 submitted.
```

Example 3) Submit a step job containing three sub-jobs with step number and dependency statement options

```
[username@greatwave1 ~]$ pjsub --step --sparam "sn=1" step1.sh
[INFO]PJM 0000 pjsub Job 12345_1 submitted.
[username@greatwave1 ~]$ pjsub --step --sparam ¥
       "jid=12345, sn=2, sd=ec!=0:after:1" step2.sh
[INFO]PJM 0000 pjsub Job 12345_2 submitted.
[username@greatwave1 ~]$ pjsub --step --sparam ¥
       "jid=12345, sn=3, sd=ec==0:one:1" step3.sh
[INFO]PJM 0000 pjsub Job 12345_3 submitted.
```

5.4.3 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 format of submitting a bulk job is as follows:

```
pjsub --bulk --sparam start-end jobscript
```

A job script for a bulk job is designed such that input/output 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 PJM_BULKNUM environment variable in the sub job.

5.4.4 Job Output

A batch job's standard output file and standard error output file are written under the job submission directory or to files specified at job submission.

Standard output generated during the job execution is written to a standard output file and error messages generated during the job execution are written to a standard error output file. If no standard output and standard error output files are specified at job submission, the following files are generated for output.

Jobname.oXXXXX --- Standard output file
Jobname.eXXXXX --- Standard error output file
(XXXXX is a job ID assigned at job submission)

5.4.5 Job Script

To submit a batch job, create a job script using the *vi* command or the *emacs* command.

(1) At the top of a job script, put "#!" followed by a path of shell.
[Sample]

```
#!/bin/sh
```

If your login shell is not bash and you execute the module commands in the job script written in sh, you need to specify "#!/bin/sh -l".

(2) From the second line onward, specify submission options using directives starting with "#PJM".
[Sample]

```
#PJM -L node=1              Specify a number of nodes
#PJM -L elapse=1:00:00      Specify elapsed time limit
#PJM -j                     Merge the standard error
```

(3) After job submission options, set runtime environment variables and specify program execution.
[Sample]

```
export OMP_NUM_THREADS=16   Set environment variable
./a.out                     Run a program
```

5.4.6 NUMA Architecture

The each node of Massively Parallel Computer and the Application Computing Server uses the NUMA (Non-Uniform Memory Access) architecture. It is expected that the assigning the processes and threads in consideration of the memory access decreases the execution time. For example, we recommend to specify the following the number of threads when you execute a multi-threaded program.

Table 5-22 Recommended number of threads

System	Recommended number of threads
Massively Parallel Computer (MPC)	4, 8, 16
ACS with Large memory (ACSL)	15 or less
ACS with GPU (ACSG)	12 or less

5.4.7 Execute MPI Program

5.4.7.1 Mpirun options for Massively Parallel Computer

Table 5-23 mpirun option for the Massively Parallel Computer

Option	Description
-np *n*	Specifies the number of parallel processes for the MPI program.
-stdin *file*	Loads from the file with the filename specified at *file*, the standard input for all parallel processes that were generated by executing the MPI program.
-stdout *file*	Saves the parallel process standard output to the file with the filename specified at *file*.
-stdout-proc *file*	Saves the parallel process standard output, separately for each process, to the file with the filename "*file.rank-number*". The character string for the "rank-number" is the actual rank number under MPI_COMM_WORLD, expressed as a numeric character string of the same number of digits.
-stderr *file*	Saves the parallel process standard error output to the file with the name specified at *file*.
-stderr-proc *file*	Saves the parallel process standard error output to the file with the filename "*file.rank-number*". The character string for the "*rank-number*" is the actual rank number under MPI_COMM_WORLD, expressed as a numeric character string of the same number of digits.
-std *file*	The parallel process standard output and standard error output are saved in the file with the name specified at *file*.
-std-proc *file*	Saves the parallel process standard output and standard error output, separately for each process, to the file with the filename "*file.rank-number*". The character string for the "r*ank-number*" is the actual rank number under MPI_COMM_WORLD, expressed as a numeric character string of the same number of digits.
-stdprefix rank	Prepends rank to output.
-nompi	Specifis that the user executes the shell script which does not execute an MPI program executable file or the executable file other than an MPI program.

[Sample] Specify stdin.txt as the standard input and stdout.txt as the standard output.

```
mpirun -np 4 -stdin stdin.txt -stdout stdout.txt ./a.out
```

5.4.7.2 Mpirun options for Application Computing Server

Table 5-24 mpirun options for the Application Computing Server

Option	Description
-np *n*	Specifies the number of parallel processes for the MPI program.
-ppn *n*	Places consecutive *n* processes on each host
-rr	Involves "round robin" startup scheme. Equivalent to -ppn 1.
-s *spec*	Redirects stdin to all or 1,2 or 2-4,6 MPI processes (0 by default).
-prepend-rank	Prepends rank to output.

5.5 Example script for batch job

5.5.1 Job Script for Massively Parallel Computer (MPC)

5.5.1.1 Sequential Job Script on Single Node for Massively Parallel Computer (Node-sharing job)

The following is a sample script for executing the job below.

- Resource Unit : gwmpc
- Resource Group : batch
- Number of nodes : 1 node
- Number of processes (threads) : 1 process (1 thread)
- Elapsed time : 10 minutes
- ProjectID : Q99999
- Merging standard error output with standard output : Yes

```
[username@greatwave1 ~]$ vi mpc-seq.sh
 #!/bin/sh
 #------ pjsub option --------#
 #PJM -L rscunit=gwmpc
 #PJM -L rscgrp=batch
 #PJM -L vnode=(core=1)
 #PJM -L elapse=10:00
 #PJM -g Q99999
 #PJM -j
 #------- Program execution -------#
 ./a.out
```

5.5.1.2 Multi-threaded Job Script on Single Node (32 cores or less, and Node share) for Massively Parallel Computer (Node-sharing job)

The following is a sample script for executing the job below.

- Resource Unit : gwmpc
- Resource Group : batch
- Number of nodes : 1 node
- Number of processes (threads) : 1 process (16 threads)
- Elapsed time : 10 minutes
- ProjectID : Q99999
- Merging standard error output with standard output : Yes

```
[username@greatwave1 ~]$ vi mpc-thread.sh
 #!/bin/sh
 #------ pjsub option --------#
 #PJM -L rscunit=gwmpc
 #PJM -L rscgrp=batch
 #PJM -L vnode=(core=16)
 #PJM -L elapse=10:00
 #PJM -g Q99999
 #PJM -j
 #------- Program execution -------#
 export OMP_NUM_THREADS=16
 ./a.out
```

When you run a thread parallelized program on the Massively Parallel Computer, OMP_NUM_THREADS environment variable must be specified. According to specifying amount of memory, a number of allocated cores changes. When you don't specify the number of processes/threads, the program may run with the unintended number of processes/threads and the performance may degrade.

OMP_NUM_THREADS ：Number of threads (-L vnode core option)

5.5.1.3 MPI Parallel Job Script on Single Node (32 cores or less) for Massively Parallel Computer (Node-sharing job)

The following is a sample script for executing the job below.

- Resource Unit : gwmpc
- Resource Group : batch
- Number of nodes : 1 node
- Number of processes (threads) : 16 processes (1 thread)
- Elapsed time : 10 minutes
- ProjectID : Q99999
- Merging standard error output with standard output : Yes

```
[username@greatwave1 ~]$ vi mpc-single-mpi.sh
 #!/bin/sh
 #------ pjsub option --------#
 #PJM -L rscunit=gwmpc
 #PJM -L rscgrp=batch
 #PJM -L vnode=(core=16)
 #PJM --mpi proc=16
 #PJM -L elapse=10:00
 #PJM -g Q99999
 #PJM -j
 #------- Program execution -------#
 mpirun ./a.out
```

When you run a MPI program on the Massively Parallel Computer, the --mpi option of the *pjsub* command must be specified. According to specifying amount of memory, a number of allocated cores changes. When you don't specify the number of processes, the program may run with the unintended number of processes and the performance may degrade.

--mpi proc : Number of total processes (-L vnode core)

5.5.1.4 Hybrid (Multi-thread + MPI) Parallel Job Script on Single Node (32 cores or less) for Massively Parallel Computer (Node-sharing job)

The following is a sample script for executing the job below.

- Resource Unit : gwmpc
- Resource Group : batch
- Number of nodes : 1 node
- Number of processes (threads) : 2 processes (16 threads)
- Elapsed time : 10 minutes
- ProjectID : Q99999
- Merging standard error output with standard output : Yes

```
[username@greatwave1 ~]$ vi mpc-single-hybrid.sh
 #!/bin/sh
 #------ pjsub option --------#
 #PJM -L rscunit=gwmpc
 #PJM -L rscgrp=batch
 #PJM -L vnode=(core=32)
 #PJM --mpi proc=2
 #PJM -L elapse=10:00
 #PJM -g Q99999
 #PJM -j
 #------- Program execution -------#
 export OMP_NUM_THREADS=16
 mpirun ./a.out
```

When you run a Hybrid program on the Massively Parallel Computer, the --mpi option of the *pjsub* command and OMP_NUM_THREADS environment variable must be specified. According to specifying amount of memory, a number of allocated cores changes. When you don't specify the number of processes/threads, the program may run with the unintended number of processes/threads and the performance may degrade.

OMP_NUM_THREADS ：Number of threads (-L vnode core / --mpi proc)
--mpi proc ：Number of total processes

5.5.1.5 MPI Parallel Job Script on Multinode (33 cores or more) for Massively Parallel Computer (Node-exclusive job)

The following is a sample script for executing the job below.

- Resource Unit : gwmpc
- Resource Group : batch
- Number of nodes : 4 node
- Number of processes (threads) : 128 processes (1 thread)
- Number of processes per node : 32 processes
- Elapsed time : 10 minutes
- ProjectID : Q99999
- Merging standard error output with standard output : Yes

```
[username@greatwave1 ~]$ vi mpc-multi-mpi.sh
 #!/bin/sh
 #------ pjsub option --------#
 #PJM -L rscunit=gwmpc
 #PJM -L rscgrp=batch
 #PJM -L node=4
 #PJM --mpi proc=128
 #PJM -L elapse=10:00
 #PJM -g Q99999
 #PJM -j
 #------- Program execution -------#
 mpirun ./a.out
```

When you run a MPI program on the Massively Parallel Computer, the --mpi option of the *pjsub* command must be specified. According to specifying amount of memory, a number of allocated cores changes. When you don't specify the number of processes, the program may run with the unintended number of processes and the performance may degrade.

--mpi proc : Number of total processes (-L node * 32 cores)

5.5.1.6 Hybrid (Multi-thread + MPI) Parallel Job Script on Multinode (33 cores or more) for Massively Parallel Computer (Node-exclusive job)

The following is a sample script for executing the job below.

- Resource Unit : gwmpc
- Resource Group : batch
- Number of nodes : 4 node
- Number of processes (threads) : 8 processes (16 threads)
- Number of processes per node : 2 processes
- Elapsed time : 10 minutes
- ProjectID : Q99999
- Merging standard error output with standard output : Yes

```
[username@greatwave1 ~]$ vi mpc-multi-hybrid.sh
 #!/bin/sh
 #------ pjsub option --------#
 #PJM -L rscunit=gwmpc
 #PJM -L rscgrp=batch
 #PJM -L node=4
 #PJM --mpi proc=8
 #PJM -L elapse=10:00
 #PJM -g Q99999
 #PJM -j
 #------- Program execution -------#
 export OMP_NUM_THREADS=16
 mpirun ./a.out
```

When you run a Hybrid program on the Massively Parallel Computer, the --mpi option of the *pjsub* command and OMP_NUM_THREADS environment variable must be specified. According to specifying amount of memory, a number of allocated cores changes. When you don't specify the number of processes/threads, the program may run with the unintended number of processes/threads and the performance may degrade.

OMP_NUM_THREADS : Number of threads (-L vnode core / --mpi proc)
--mpi proc : Number of total processes

When you request the number of nodes using the -L node option, all computing resources in each node (32 cores and 30Gi memory) are assigned to a job.

5.5.2 Job Script for Application Computing Server

5.5.2.1 Sequential Job Script on Single Node for Application Computing Server

The following is a sample script for executing the job below.

- Resource Unit : gwacsg
- Resource Group : batch
- Number of nodes : 1 node
- Number of processes (threads) : 1 process (1 thread)
- Elapsed time : 10 minutes
- ProjectID : Q99999
- Merging standard error output with standard output : Yes

```
[username@greatwave1 ~]$ vi acs-seq.sh
 #!/bin/sh
 #------ pjsub option --------#
 #PJM -L rscunit=gwacsg
 #PJM -L rscgrp=batch
 #PJM -L vnode=1
 #PJM -L vnode-core=1
 #PJM -L elapse=10:00
 #PJM -g Q99999
 #PJM -j
 #------- Program execution -------#
 ./a.out
```

5.5.2.2 **Multi-threaded Job Script on Single Node for Application Computing Server**

The following is a sample script for executing the job below.

- Resource Unit : gwacsg
- Resource Group : batch
- Number of nodes : 1 node
- Number of processes (threads) : 1 process (12 threads)
- Elapsed time : 10 minutes
- ProjectID : Q99999
- Merging standard error output with standard output : Yes

```
[username@greatwave1 ~]$ vi acs-thread.sh
 #!/bin/sh
 #------ pjsub option --------#
 #PJM -L rscunit=gwacsg
 #PJM -L rscgrp=batch
 #PJM -L vnode=1
 #PJM -L vnode-core=12
 #PJM -L elapse=10:00
 #PJM -g Q99999
 #PJM -j
 #------- Program execution -------#
 export OMP_NUM_THREADS=12
 ./a.out
```

When you run a thread parallelized program on the Application Computing Server, OMP_NUM_THREADS environment variable must be specified. According to specifying amount of memory, a number of allocated cores changes. When you don't specify the number of threads, the program may run with the unintended number of threads and the performance may degrade.

OMP_NUM_THREADS ：Number of threads (-L vnode-core option)

5.5.2.3 **MPI Parallel Job Script on Single Node for Application Computing Server**

The following is a sample script for executing the job below.

- Resource Unit : gwacsg
- Resource Group : batch
- Number of nodes : 1 node
- Number of processes (threads) : 12 processes (1 thread)
- Elapsed time : 10 minutes
- ProjectID : Q99999
- Merging standard error output with standard output : Yes

```
[username@greatwave1 ~]$ vi acs-single-mpi.sh
 #!/bin/sh
 #------ pjsub option --------#
 #PJM -L rscunit=gwacsg
 #PJM -L rscgrp=batch
 #PJM -L vnode=1
 #PJM -L vnode-core=12
 #PJM -L elapse=10:00
 #PJM -g Q99999
 #PJM -j
 #------- Program execution -------#
 mpirun -np 12 ./a.out
```

When you run a MPI program on the Application Computing Server, the -np option of the *mpirun* command must be specified. According to specifying amount of memory, a number of allocated cores changes. When you don't specify the number of processes, the program may run with the unintended number of processes and the performance may degrade.

-np : Number of total processes (-L vnode-core)

5.5.2.4 Hybrid (Multi-thread + MPI) Parallel Job Script on Single Node for Application Computing Server

The following is a sample script for executing the job below.

- Resource Unit : gwacsg
- Resource Group : batch
- Number of nodes : 1 node
- Number of processes (threads) : 2 processes (6 threads)
- Number of cores : 12 cores (2 x 6)
- Elapsed time : 10 minutes
- ProjectID : Q99999
- Merging standard error output with standard output : Yes

```
[username@greatwave1 ~]$ vi acs-single-hybrid.sh
 #!/bin/sh
 #------ pjsub option --------#
 #PJM -L rscunit=gwacsg
 #PJM -L rscgrp=batch
 #PJM -L vnode=1
 #PJM -L vnode-core=12
 #PJM -L elapse=10:00
 #PJM -g Q99999
 #PJM -j
 #------- Program execution -------#
 export OMP_NUM_THREADS=6
 mpirun -np 2 ./a.out
```

When you run a Hybrid program on the Application Computing Server, the -np option of the *mpirun* command and OMP_NUM_THREADS environment variable must be specified. According to specifying amount of memory, a number of allocated cores changes. When you don't specify the number of processes/threads, the program may run with the unintended number of processes/threads and the performance may degrade.

OMP_NUM_THREADS : Number of threads (-L vnode-core / number of total processes)
-np : Number of total processes

5.5.2.5 **MPI Parallel Job Script on Multinode for Application Computing Server**

The following is a sample script for executing the job below.

- Resource Unit : gwacsg
- Resource Group : batch
- Number of nodes : 2 node
- Number of processes (threads) : 48 processes (1 threads)
- Number of processes per node : 24 processes
- Elapsed time : 10 minutes
- ProjectID : Q99999
- Merging standard error output with standard output : Yes

```
[username@greatwave1 ~]$ vi acs-multi-mpi.sh
 #!/bin/sh
 #------ pjsub option --------#
 #PJM -L rscunit=gwacsg
 #PJM -L rscgrp=batch
 #PJM -L vnode=2
 #PJM -L vnode-core=24
 #PJM -L elapse=10:00
 #PJM -g Q99999
 #PJM -j
 #------- Program execution -------#
 mpirun -np 48 -ppn 24 ./a.out
```

When you run a MPI program on the Application Computing Server, the -np option and the --ppn option of the *mpirun* command must be specified. According to specifying amount of memory, a number of allocated cores changes. When you don't specify the number of processes, the program may run with the unintended number of processes and the performance may degrade.

-np : Number of total processes
-ppn : Number of processes per node

5.5.2.6 Hybrid (Multi-thread + MPI) Parallel Job Script on Single Node for Application Computing Server

The following is a sample script for executing the job below.

- Resource Unit : gwacsg
- Resource Group : batch
- Number of nodes : 2 node
- Number of processes (threads) : 4 processes (12 threads)
- Number of processes per node : 2 processes
- Elapsed time : 10 minutes
- ProjectID : Q99999
- Merging standard error output with standard output : Yes

```
[username@greatwave1 ~]$ vi acs-multi-hybrid.sh
 #!/bin/sh
 #------ pjsub option --------#
 #PJM -L rscunit=gwacsg
 #PJM -L rscgrp=batch
 #PJM -L vnode=2
 #PJM -L vnode-core=24
 #PJM -L elapse=10:00
 #PJM -g Q99999
 #PJM -j
 #------- Program execution -------#
 export OMP_NUM_THREADS=12
 mpirun -np 4 -ppn 2 ./a.out
```

When you run a Hybrid program on the Application Computing Server, the -np option and the --ppn option of the *mpirun* command, and OMP_NUM_THREADS environment variable must be specified. According to specifying amount of memory, a number of allocated cores changes. When you don't specify the number of processes/threads, the program may run with the unintended number of processes/threads and the performance may degrade.

OMP_NUM_THREADS : Number of threads (-L vnode-core / number of processes per node)
-np : Number of total processes
-ppn : Number of processes per node

5.5.2.7 **GPU job script for single node job on Application Computing Server**

The following is a sample script for executing the job below.

- Resource Unit : gwacsg
- Resource Group : gpu
- Number of nodes : 1 node
- Number of processes (threads) : 2 processes
- Elapsed time : 10 minutes
- Number of GPU : 2
- ProjectID : Q99999
- Merging standard error output with standard output : Yes

```
[username@greatwave1 ~]$ vi acsg-gpu.sh
 #!/bin/sh
 #------ pjsub option --------#
 #PJM -L rscunit=gwacsg
 #PJM -L rscgrp=gpu
 #PJM -L vnode=1
 #PJM -L vnode-core=2
 #PJM -L elapse=10:00
 #PJM -x gpu_per_vnode=2
 #PJM -g Q99999
 #PJM -j
 #------- Program execution -------#
 mpirun -np 2 ./a.out
```

Specify number of required GPU(up to 4) by "-x gpu_per_vnode" option. (NOT "-L", but "-x" option. Please be careful.) You can refer the value as gpu_per_vnode environment variable in the job script.

GPU jobs are run by consuming computational core time as 6 times of elapse time per GPU. In the case of above script, the following message is displayed and the job consumes computational core time of "12 * job elapse time".

```
[username@greatwave1 ~]$ pjsub acsg-gpu.sh
pjsub: WARNING: submitted job consumes more cpu-core than specified on
each node due to GPU option. (vnode-core: 2 -> 12)
[INFO] PJM 0000 pjsub Job 12345 submitted.
```

5.5.2.8 **GPU job script for multinode job on Application Computing Server**

The following is a sample script for executing the job below.

- Resource Unit : gwacsg
- Resource Group : gpu
- Number of nodes : 2 node
- Number of processes (threads) : 4 processes(1 thread)
- Elapsed time : 10 minutes
- Number of GPU : 2 boards per node
- ProjectID : Q99999
- Merging standard error output with standard output : Yes

```
[username@greatwave1 ~]$ vi acsg-multinode-gpu.sh
 #!/bin/sh
 #------ pjsub option --------#
 #PJM -L rscunit=gwacsg
 #PJM -L rscgrp=gpu
 #PJM -L vnode=2
 #PJM -L vnode-core=2
 #PJM -L elapse=10:00
 #PJM -x gpu_per_vnode=2
 #PJM -g Q99999
 #PJM -j
 #------- Program execution -------#
 mpirun -np 4 -ppn 2 gpurun ./a.out
```

Specify number of required GPU(up to 4) by "-x gpu_per_vnode" option. (NOT "-L", but "-x" option. Please be careful.)

Put "gpurun" command just before the executable file to aquire GPU resources on the every job execution nodes.

GPU jobs are run by consuming computational core time as 6 times of elapse time per GPU. In the case of above script, the following message is displayed and the job consumes computational core time of "12 * job elapse time".

```
[username@greatwave1 ~]$ pjsub acsg-multinode-gpu.sh
pjsub: WARNING: submitted job consumes more cpu-core than specified on
each node due to GPU option. (vnode-core:2 -> 12)
[INFO] PJM 0000 pjsub Job 12345 submitted.
```

5.6 **Execute Interactive Jobs**

To execute an interactive job, specify the "--interact" option on the *pjsub* command line. The job management system allocates interactive jobs to execute in interactive mode.

When submitting an interactive job, job submission options are specified as arguments on the command line.

```
pjsub --interact [--sparam wait-time=sec] [option...]
```

By specifying the wait time, the interactive job will wait for the specified time and resource assignment if the computing resource is insufficient. (The interactive job does not wait without specifying wait-time.)

When no command is executed for 10 minutes in the interactive job, the interactive job ends

Example 1) Execute an interactive job (sequential) for the Massively Parallel Computer

```
[username@greatwave1 ~]$ pjsub --interact -L rscunit=gwmpc -g Q99999
[INFO] PJM 0000 pjsub Job 12345 submitted.
[INFO] PJM 0081 .connected.
[INFO] PJM 0082 pjsub Interactive job 12345 started.
[username@gwmpc0001 ~]$ frt hello_world.f95
[username@gwmpc0001 ~]$ ./a.out
Hello world
[username@gwmpc0001 ~]$ exit
exit
[INFO] PJM 0083 pjsub Interactive job 12345 completed.
```

Example 2) Execute an interactive job (16 MPI parallel) with single node for the Massively Parallel Computer

```
[username@greatwave1 ~]$ pjsub --interact -L rscunit=gwmpc ¥
   -L "vnode=(core=16)" --mpi proc=16 -g Q99999
[INFO] PJM 0000 pjsub Job 12346 submitted.
[INFO] PJM 0081 .connected.
[INFO] PJM 0082 pjsub Interactive job 12346 started.
[username@gwmpc0001 ~]$ mpifrt hello_world_mpi.f95
[username@gwmpc0001 ~]$ mpirun ./a.out
Hello world from rank 0 process
(snip)
Hello world from rank 15 process
[username@gwmpc0001 ~]$ exit
exit
[INFO] PJM 0083 pjsub Interactive job 12346 completed.
```

When specifying the -L vnode options, enclose with single quotation or double quotation to avoid special handling by the shell.

Example 3) Execute an interactive job (64 MPI parallel) with multiple nodes for the Massively Parallel Computer

```
[username@greatwave1 ~]$ pjsub --interact  -L rscunit=gwmpc ¥
    -L node=2 --mpi proc=64 -g Q99999
[INFO] PJM 0000 pjsub Job 12347 submitted.
[INFO] PJM 0081 .connected.
[INFO] PJM 0082 pjsub Interactive job 12347 started.
[username@gwmpc0001 ~]$ mpifrt hello_world_mpi.f95
[username@gwmpc0001 ~]$ mpirun ./a.out
Hello world from rank 0 process
(snip)
Hello world from rank 63 process
[username@gwmpc0001 ~]$ exit
exit
[INFO] PJM 0083 pjsub Interactive job 12347 completed.
```

Example 4) Execute an interactive job (sequential) for the ACS with GPU.

```
[username@greatwave1 ~]$ pjsub --interact -L rscunit=gwacsg ¥
    -g Q99999
[INFO] PJM 0000 pjsub Job 12345 submitted.
[INFO] PJM 0081 .connected.
[INFO] PJM 0082 pjsub Interactive job 12345 started.
[username@gwacsg01 ~]$ ifort hello_world.f95
[username@gwacsg01 ~]$ ./a.out
Hello world
[username@gwacsg01 ~]$ exit
exit
[INFO] PJM 0083 pjsub Interactive job 12345 completed.
```

Example 5) Execute an interactive job (12 MPI parallel) with single node for the ACS with GPU.

```
[username@greatwave1 ~]$ pjsub --interact -L rscunit=gwacsg ¥
    -L vnode=1 -L vnode-core=12 -g Q99999
[INFO] PJM 0000 pjsub Job 12346 submitted.
[INFO] PJM 0081 .connected.
[INFO] PJM 0082 pjsub Interactive job 12346 started.
[username@gwacsg01 ~]$ mpiifort hello_world_mpi.f95
[username@gwacsg01 ~]$ mpirun -np 12 ./a.out
Hello world from rank 0 process
(snip)
Hello world from rank 11 process
[username@gwacsg01 ~]$ exit
exit
[INFO] PJM 0083 pjsub Interactive job 12346 completed.
```

Example 6) Execute an interactive job (48 MPI parallel) with multiple nodes for the ACS with GPU.

```
[username@greatwave1 ~]$ pjsub --interact -L rscunit=gwacsg ¥
    -L vnode=2 -L vnode-core=24 -g Q99999
[INFO] PJM 0000 pjsub Job 12347 submitted.
[INFO] PJM 0081 .connected.
[INFO] PJM 0082 pjsub Interactive job 12347 started.
[username@gwacsg01 ~]$ mpiifort hello_world_mpi.f95
[username@gwacsg01 ~]$ mpirun -np 48 -ppn 24 ./a.out
Hello world from rank 0 process
(snip)
Hello world from rank 47 process
[username@gwacsg01 ~]$ exit
exit
[INFO] PJM 0083 pjsub Interactive job 12347 completed.
```

5.7 **Job Status**

Use the *pjstat* command to check the status of submitted jobs and resource information.

```
pjstat [option] [JOBID[JOBID…]]
```

Table 5-25 pjstat option

Option	Description
None	Display information of queuing jobs and running jobs.
-A	Display information of jobs of all users in the same project.
-g projectID	Display information of jobs which belong specified project.
-H [day=xx]	Display information of completed jobs (Last 3 days by default). If you set day=7, pjstat outputs the information of last 7 days(maximum: Last 30 days).
-E	Display step job and bulk job information.
-v	Display additional job information that is not included in the standard format.
-s	In addition to information displayed with the -v option, detailed information such as resources usage status and resource limitations is displayed.
-S	In addition to information displayed with the -s option, information about each node allocated to the job is displayed.
--rsc	Display resource group information.
--un	Display node status
--uc	Display core status
-p	Display priority order of projects

5.7.1 Job status

The *pjstat* command displays status of jobs that are currently running or are in the queue.

Because a projected time on the START_DATE field the indication, a projected time fluctuates based on system congestion and priority amoung projects.

```
[username@greatwave1 ~]$ pjstat

  ACCEPT QUEUED  STGIN  READY RUNING RUNOUT STGOUT   HOLD  ERROR   TOTAL
       0      1      0      0      2      0      0      0      0       3
s      0      1      0      0      2      0      0      0      0       3
JOB_ID  JOB_NAME MD ST  USER     START_DATE      ELAPSE_LIM NODE_REQUIRE  VNODE  CORE V_MEM
1234    job.sh   NM RUN username 01/01 00:00:00  0012:00:00 16            -      -    -
1235    job.sh   NM RUN username 01/01 01:00:00  0012:00:00 -             2      12   1024 MiB
1236    job.sh   NM QUE username (01/02 00:00)   0012:00:00 -             2      12   1024 MiB
```

Table 5-26 Job status

Field	Description
JOB_ID	Job ID For sub-jobs, Subjob ID
JOB_NAME	Job name
MD	Job model (NM: Normal job, ST: Step job, BU: Bulk job)
ST	Job state (See Table 5-27 Job state)
USER	User name who executed the job
START_DATE	Projected start time or time started "(MM/DD hh:mm)" After the execution is started "MM/DD hh:mm:ss" As for jobs to which backfill is applied, "<" is added after the time. "(MM/DD hh:mm)<" or "MM/DD hh:mm:ss<"
ELAPSE_LIM	Elapsed time limit "hhhh:mm:ss"
NODE_REQUIRE	For Massively Parallel Computer: • multinode job: Number of nodes • single node job: 1 For Application Computing Server: "-" is output.
VNODE	Number of nodes (only for Application Computing Server)
CORE	Number of cores per node (only for Application Computing Server)
V_MEM	Amount of memory per node (only for Application Computing Server)

Table 5-27 Job state

Status	Description
ACC	Accepted job submission
QUE	Waiting for job execution
RNA	Acquiring resources required job execution
RUN	Executing job
RNO	Waiting for completion of job termination processing
EXT	Exited job end execution
CCL	Exited job execution by interruption
ERR	In fixed state due to an error
RJT	Rejected job submission

5.7.2 Detailed Job Status (-v option)

The -v option displays detailed job information.

```
[username@greatwave1 ~]$ pjstat -v

  ACCEPT QUEUED  STGIN  READY RUNING RUNOUT STGOUT   HOLD  ERROR   TOTAL
       0      0      0      0      0      0      0      0      0       0
s      0      0      0      0      0      0      0      0      0       0

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
2171       STDIN      NM RUN username projectID -              0000:00:00 0002:00:00 1
-      -    -            -     -     -         RNA 0   140 0  127 02/10 12:38:39 gwmpc    -
```

Table 5-28 Job detailed information (Additional field in -v option)

Field	Description
GROUP	ProjectID
ELAPSE_TIM	Elapsed time limit
V_POL	Arrangement policy of virtual node (only for Application Computing Server)
E_POL	Execution mode policy (only for Application Computing Server)
RANK	The allocation rule of the rank (only for Application Computing Server)
LST	Last processing state of the job
EC	Job script exit code
PC	PJM code
SN	Signal number
PRI	Job priority (0: low <-> 255: high)
ACCEPT	Job submission date
RSC_UNIT	Resource unit
REASON	Error message

5.7.3 Ended Job Status (-H option)

The -H option displays ended job information in addition to submitted jobs.

```
[username@greatwave1 ~]$ pjstat -H

  ACCEPT QUEUED  STGIN  READY RUNING RUNOUT STGOUT   HOLD  ERROR   TOTAL
       0      0      0      0      0      0      0      0      0       0
s      0      0      0      0      0      0      0      0      0       0

  REJECT   EXIT CANCEL   TOTAL
       0      0      0       0
s      0    180      0     180

JOB_ID     JOB_NAME   MD ST  USER     START_DATE      ELAPSE_LIM NODE_REQUIRE    VNODE  CORE V_MEM
2135       run.sh     NM EXT username 02/08 09:58:02  0012:00:00 -               1      60   16384
MiB
```

Table 5-29 Ended job status

Field	Description
JOB_ID	Job ID For sub-jobs, Subjob ID
JOB_NAME	Job name
MD	Job model (NM: Normal job, ST: Step job, BU: Bulk job)
ST	Job state (See Table 5-27 Job state)
USER	User name who executed the job
START_DATE	Start time
ELAPSE_LIM	Elapsed time limit "hhhh:mm:ss"
NODE_REQUIRE	For Massively Parallel Computer: • multinode job: Number of nodes • single node job: 1 For Application Computing Server: "-" is output.
VNODE	Number of nodes (only for Application Computing Server)
CORE	Number of cores per node (only for Application Computing Server)
V_MEM	Amount of memory per node (only for Application Computing Server)

5.7.4 Resource Unit and Resource Group Status (--rsc option)

The --rsc option displays resource groups available for the user.

```
[username@greatwave1 ~]$ pjstat  --rsc
RSCUNIT                 RSCUNIT_SIZE  RSCGRP                      RSCGRP_SIZE
-------------------------------------------------------------------------------
gwmpc  [ENABLE,START]   6x5x3         batch    [ENABLE,START]   1068
                                      gaussian [ENABLE,START]   1068
                                      interact [ENABLE,START]   12
-------------------------------------------------------------------------------
gwacsg [ENABLE,START]   30            adf      [ENABLE,START]   30
                                      amber    [ENABLE,START]   30
                                      batch    [ENABLE,START]   30
                                      gaussian [ENABLE,START]   30
                                      interact [ENABLE,START]   30
-------------------------------------------------------------------------------
gwacsl [ENABLE,START]   2             ansys    [ENABLE,START]   2
                                      batch    [ENABLE,START]   2
                                      gaussian [ENABLE,START]   2
                                      interact [ENABLE,START]   2
-------------------------------------------------------------------------------
* [ENABLE/DISABLE]: New jobs can be submitted or not.
  [START/STOP]    : QUE jobs can be started or not.
```

Table 5-30 Resource unit and resource group information

Field	Description
RSCUNIT	Resource unit name and its status. Displayed statuses are the following. ENABLE : Jobs can be submitted DISABLE : Jobs cannot be submitted START : Jobs can be executed STOP : Jobs cannot be executed
RSCUNIT_SIZE	Size of resource unit. [Massively Parallel Computer] Number of Tofu is expressed in X,Y,Z coordinates. Format: XxYxZ [Application Computing Server] The number of nodes N which makes up the resource unit is displayed.
RSCGRP	Resource group name and its status
RSCGRP_SIZE	Size of resource group [Massively Parallel Computer] Number of nodes is expressed in X,Y,Z coordinates. Format: XxYxZ (Cuboid) or N (number of nodes) [Application Computing Server] The number of nodes N that makes up the resource unit is displayed.

5.7.5 Status of Node and Core Usage (-un option and uc option)

The -un option displays the status of node usage HOKUSAI GreatWave system.

```
[username@greatwave1 ~]$ pjstat -un
The status of node usage                                          Ratio Used/Total
--------------------------------------------------------------------------------
gwmpc     ************************************************---     95.9%(1036/1080)
gwacsg    **************************************************      100.0%(  30/  30)
gwacsl    **************************************************      100.0%(   2/   2)
```

Table 5-31 Status of node usage

Field	Description
Ratio	Used ratio
Used	Used number of nodes
Total	Total number of nodes

The -uc option displays the status of core usage HOKUSAI GreatWave system.

```
[username@greatwave1 ~]$ pjstat -uc
The status of core usage                                        Ratio  Used/Total
--------------------------------------------------------------------------------
gwmpc     ************************************************---  95.8%(33116/34560)
gwacsg    ************************************************--  96.1%(  692/  720)
gwacsl    ************************************************** 100.0%(  120/  120)
```

Table 5-32 Status of core usage

Field	Description
Ratio	Used ratio
Used	Used number of cores
Total	Total number of cores

The jobs are scheduled by priority order of projects. When the jobs wait whose priority order is higher than your priority order, your jobs are not executed if the unused nodes or cores exist.

5.7.6 Priority Order of Projects (-p option)

The -p option displays the priority order of projects per resource unit.

```
[username@greatwave1 ~]$ pjstat -p
Project priority in fair-share function
[Q99999]
 +- gwmpc :   10th
 +- gwacsg:   1st
 +- gwacsl:   3rd
```

5.7.7 Resource limit of job submission (-x option)

The -x option displays the upper limit of number of cores, nodes and elapse time of each resource group.

The following example indicates that you can submit up to 72 hours job with less than or equal 512 cores (16 nodes) and up to 24 hours job with less than or equal 8,192 cores (256 nodes) to the batch resource group of gwmpc.

```
[username@greatwave1 ~]$ pjstat -x
Limits on resources

PROJECT  RSCUNIT  RSCGRP        MAX_CORE (NODE)  MAX_ELAPSE
=========================================================
Q99999   gwmpc    batch            512 (  16)    72:00:00
                                   8192 ( 256)    24:00:00
                  gaussian          32 (   1)    72:00:00
                  interact         128 (   4)     2:00:00
 :
```

5.8 Cancel jobs

Use the *pjdel* command to cancel submitted jobs.

```
pjdel JOBID [JOBID…]
```

Specify job IDs to cancel to the argument on the *pjdel* command line.

```
[username@greatwave1 ~]$ pjdel 12345
[INFO] PJM 0100 pjdel Job 12345 canceled.
```

5.9 Display a job script

Use the *pjcat* command to display the job script.

```
pjcat -s JOBID
```

Specify a job ID to display to the argument on the *pjcat* command line.

```
[username@greatwave1 ~]$ pjcat -s 12345
#!/bin/sh

#PJM -L rscunit=gwmpc
#PJM -L rscgrp=batch
#PJM -L vnode=(core=1)

./a.out
```

5.10 Environment Variable

5.10.1 Environment Variables for Thread Parallelization and MPI Execution

This section explains main environment variables specified to execute thread parallelized programs or MPI programs.

Table 5-33 Runtime environment variables (common)

Environment Variable	Description
OMP_NUM_THREADS	When executing a multi-threaded program by OpenMP or auto parallelization, set the number of threads to the OMP_NUM_THREADS environment variable. If this variable is not specified, the number of cores available for the job is set.

Table 5-34 Runtime environment variables (Massively Parallel Computer)

Environment Variable	Description
FLIB_FASTOMP	Determine whether the high-speed runtime library is adopted instead of the usual runtime library at parallelization by OpenMP. Default is FALSE. The high-speed runtime library achieves a fast execution time by partly limiting the OpenMP specification. - Nested parallelism is inhibited - The number of threads specified with the num_threads clause or the omp_set_num_threads function must be the same as defined by OMP_NUM_THREADS environment variable.
PARALLEL	Specify the number of auto parallelized threads if you execute a program parallelized by both OpenMP and auto parallelization and the number of OMP threads is different from the number of auto parallelized threads.
FLIB_FASTOMP	The diagnostic message "jwe1050i-w" (the on-chip hardware barrier cannot be used) error can be controlled. Default is TRUE. The Notes are follows: • When the on-chip hardware barrier is used, CPU Binding is needed. • When the number of threads is 1, the on-chip hardware barrier cannot be used. • When the number of CPUs to a process is three or less, a on-chip hardware barrier may not be able to use.
THREAD_STACK_SIZE	Set the stack size of each thread in K bytes. If this variable is not specified, the value set to ulimit -s (unlimited) is used. If the OMP_STACKSIZE is set, either the value of OMP_STACKSIZE or the value of this variable, whichever is greater, becomes the stack size.

Table 5-35 Runtime environment variables (Application Computing Server)

Environment Variable	Description
KMP_AFFINITY	Control to bind threads to physical processors. Default of HOKUSAI GreatWave is "compact"
I_MPI_PIN_DOMAIN	Control to bind MPI processes to physical processors. Default of HOKUSAI GreatWave is "omp".

5.10.2 Environment Variables for Jobs

The job management system configures the following environment variables for jobs.

Table 5-36 Available environment variables in jobs

Envirnment variable	Description
PJM_ENVIRONMENT	"BATCH" for a batch job; ”INTERACT” for an interactive job
PJM_JOBID	Job ID
PJM_JOBNAME	Job name
PJM_O_WORKDIR	The current directory at the *pjsub* command execution
PJM_COMMENT	The string set to the --comment option on the *pjsub* command line
PJM_MAILSENDTO	The mail destination user set to the --mail-list option on the *pjsub* command line
PJM_STEPNUM	Step number (set only for step jobs)
PJM_BULKNUM	Bulk number(set only for bulk jobs)
PJM_SUBJOBID	Sub-job ID (set only for step jobs)

In addition, some resource options specified at job submitting are set as the following environment variables.

Table 5-37 Available environment variables in jobs

Envirnment variable	Description
PJM_RSCUNIT	Name of resource unit (-L rscunit)
PJM_RSCGRP	Name of resource group (-L rscgrp)
PJM_NODE	Number of nodes Massively Parallel Computer Node-exclusive job: value of "-L node" Node-sharing job: 1 Application Computing Server value of "-L vnode"
PJM_NODE_CORE	Cores per node Massively Parallel Computer Node-exclusive job: value of "-L core" (default 32) Node-sharing job: value of "-L vnode=(core=N)" Application Computing Server value of "-L vnode-core"
PJM_TOTAL_CORE	Total number of cores (PJM_NODE * PJM_NODE_CORE)
PJM_NODE_MEM	Amount of memory per node Massively Parallel Computer Node-exclusive job: value of "-L node-mem" Node-sharing job: value of "-L vnode=(mem=N)" Application Computing Server value of "-L vnode-mem"
PJM_NODE_MEM_BYTE	PJM_NODE_MEM in the byte unit
PJM_CORE_MEM	Amount of memory per core Massively Parallel Computer Node-exclusive job: 960Mi Node-sharing job: value of "-L vnode=(core-mem=N)" Application Computing Server value of "-L core-mem"
PJM_CORE_MEM_BYTE	PJM_CORE_MEM in the byte unit
PJM_ELAPSE	Elapse limit (value of "-L elapse")
PJM_ELAPSE_SEC	PJM_ELAPSE in the second unit

You should use the -X option on the *pjsub* command line to pass environment variables set before job submission on a login node to a job. Otherwise, no environment variables are passed.

5.10.3 Environment Vairables for handling released heap memory

In the Massively Parallel Computer, XOS_MMM_L_ARENA_FREE=2 is the default setting as the environment variables of processing mode at releasing memory. For more detail of the variable, please refer the result of "manpx lpgparm" command on the login node.

Table 5-38 XOS_MMM_L_ARENA_FREE configuration

Option		Description
XOS_MMM_L_ARENA_FREE		
	1	Give priority to efficiency of memory use
	2	Give priority to performance of memory allocation

Due to this configuration, some jobs which use much memory will fail by exceeding memory limit, or memory allocation throughput will be slow at the multi thread programs. If you encounter such problems, please set XOS_MMM_L_ARENA_FREE=1 in the job script and avoid those problems.

This environment variable is only available on the Massively Parallel Computer.

Example of setting "XOS_MMM_L_ARENA_FREE" in job script

```
[username@greatwave1 ~]$ cat go.sh
#!/bin/sh

#PJM -L rscunit=gwmpc
#PJM -L rscgrp=batch
#PJM -L vnode=(core=1)

export XOS_MMM_L_ARENA_FREE=1
./a.out
```

Check the specifications of memory allocation

```
[username@greatwave1 ~]$ manpx lpgparm
```

6. Development Tools

6.1 Massively Parallel Computer

6.1.1 FUJITSU Software Development Tools (Development Tools)

For the Massively Parallel Computer, FUJITSU Software Development Tools (Development Tools) are available. The development tools support various phases of application development process in a GUI environment. Development Tools implement multiple advanced development tools: the file explorer, the editor, the debugger, and the profiler.

For more detail, refer to " Programming Workbench Users Guide", " Debugger Users Guide" and "Profiler Users Guide".

6.1.2 Install Development Tools

(1) Access the User Portal.

https://hokusai.riken.jp/

(2) Select "Tools".

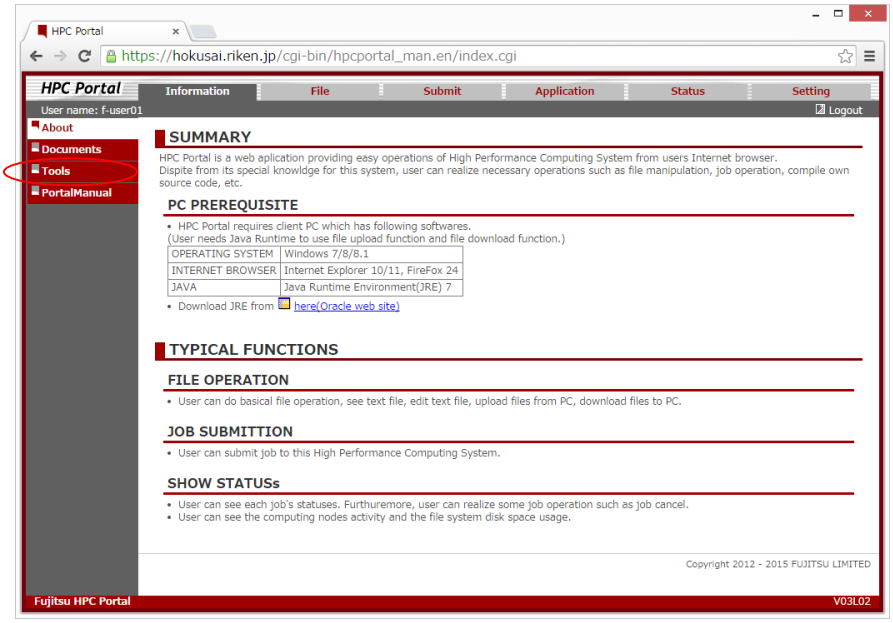


Figure 6-1 Screen of selecting "Tools"

(3) Download "Programing Workbench" for your PC (Linux is not supported).

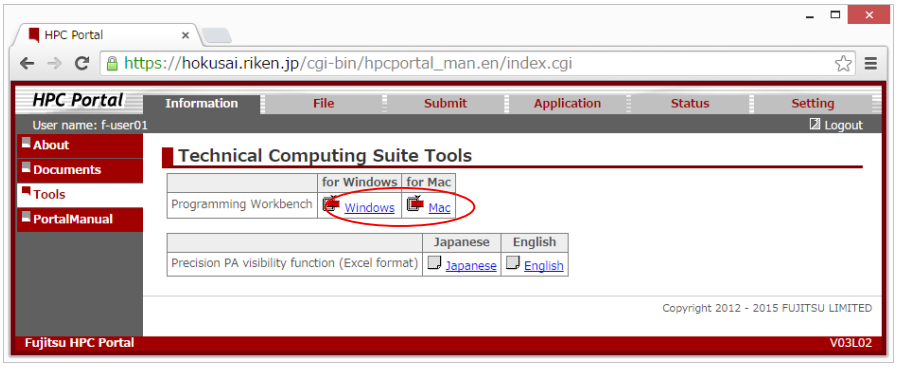
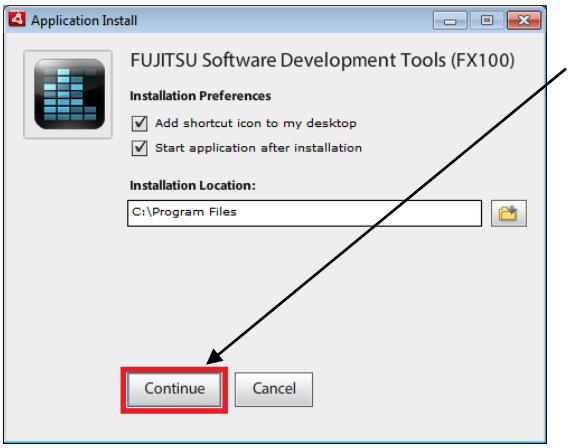


Figure 6-2 Programming Workbench Download screen

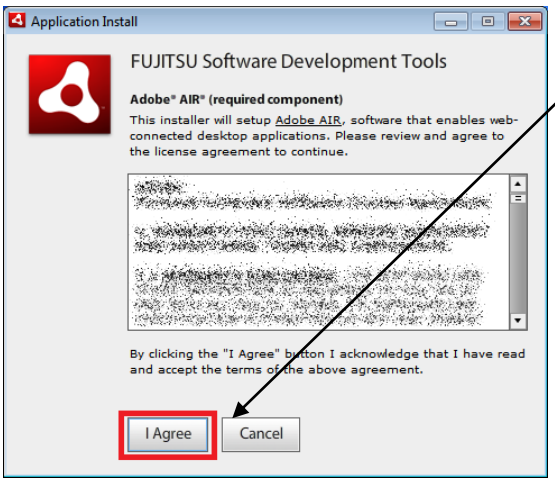
(4) Execute downloaded file.



Click the "Continue" button

Figure 6-3 Programming Workbench Install screen

(5) If Adobe AIR is not installed on your PC, Adobe AIR License Agreement dialog appears.

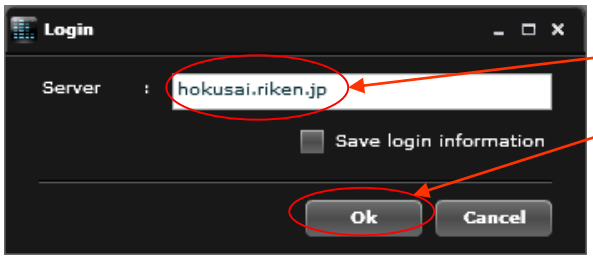


Click [I Agree] button to accept the license agreement.

Figure 6-4 Adobe AIR Install screen

6.1.3 How to Use Development Tools

(1) Launch FUJITSU Software Development Tools (FX100). Enter "**hokusai.riken.jp"** as server.



1. Enter "hokusai.riken.jp"
2. Click "OK" button

Figure 6-5 Development Tools login screen

(2) When the confirmation of certificate appears, select the client certificate for HOKUSAI GreatWave system and Click [OK] button.

(3) The menu of Development Tools appears.

Click icons to start each tool.
(1)File explorer
(2)Editor
(3)Builder
(4)Executor
(5)Debugger
(6)Profiler

Figure 6-6 Development Tools menu screen

6.2 Application Computing Server

The following tools for the Application Computing Server are available.

- Intel VTune Amplifier XE (Performance profiler)
- Intel Inspector XE (Memory and Thread Debugger)
- Intel Advisor XE (Thread design and prototype)
- Intel Trace Analyzer & Collector (MPI Communications Profiling and Analysis)

7. User Portal

To use the User Portal, launch the browser and access to the following URL.

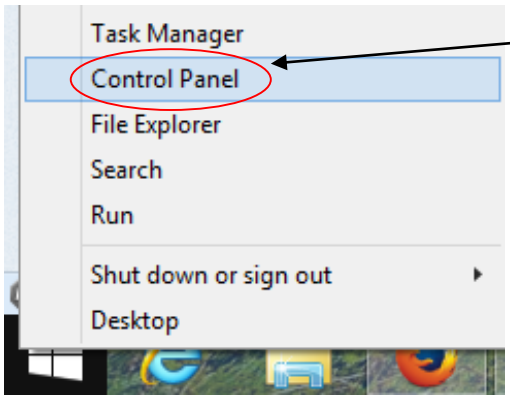
https://hokusai.riken.jp

7.1 Import Client Certificate into JAVA Environment

The upload and download functions of the User Portal use JAVA. To use these functions, import the client certificate info JAVA environment.

7.1.1 Launch Java Control Panel

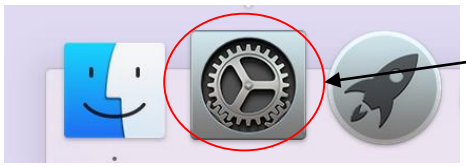
7.1.1.1 Launch Java Control Panel (Windows Environment)



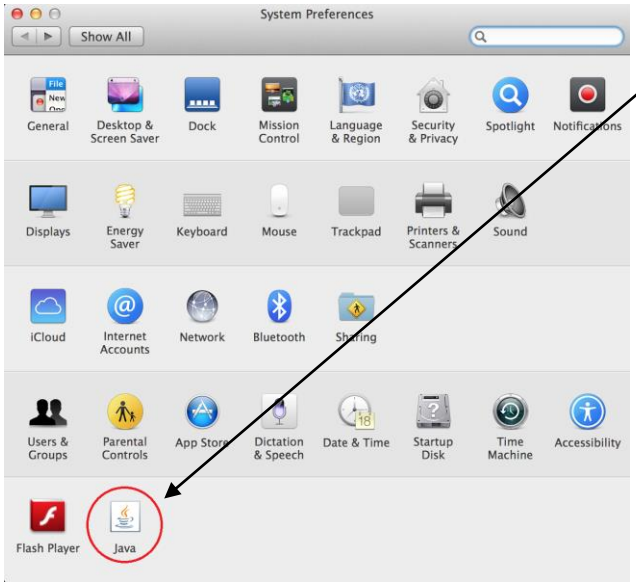
1. Press Windows + X on your keyborad, and click [Control Panel].

2. Double-click [Java].

7.1.1.2 Launch Java Control Panel (Mac Environment)



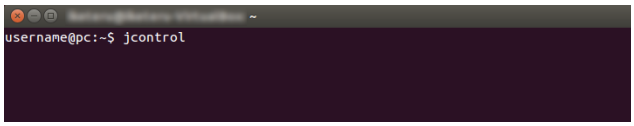
1. Click [System Preferences].



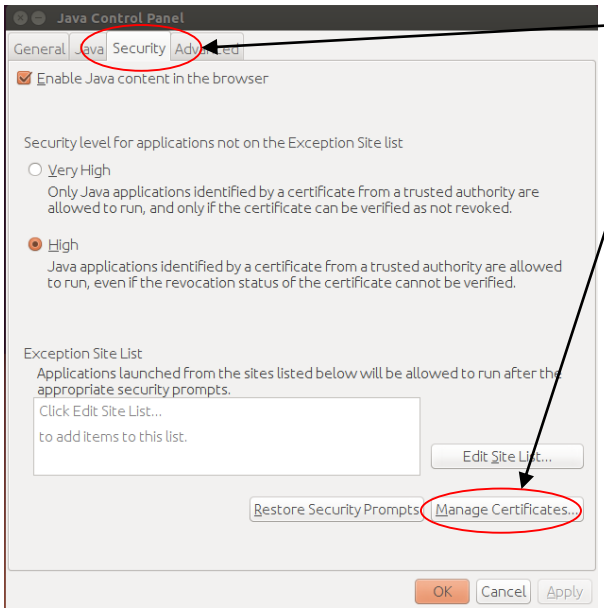
2. Double-click [Java].

7.1.1.3 **Launch Java Control Panel (Ubuntu Environment)**

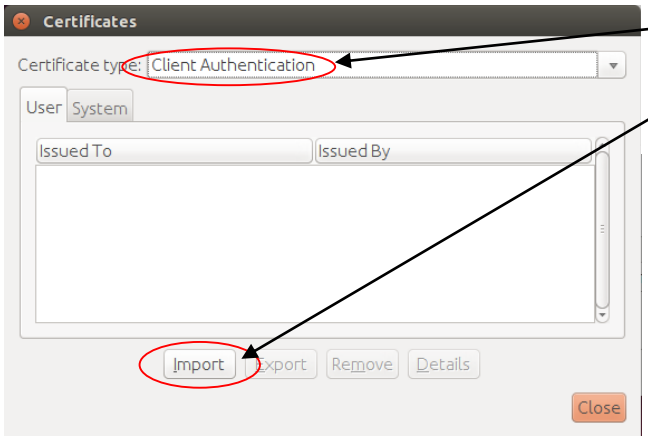
Open the Terminal and execute the *jcontrol* command.



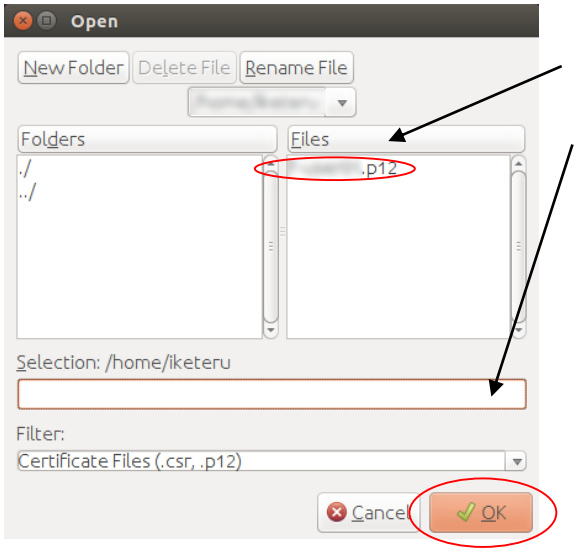
7.1.2 **Import Client Certificate (Common)**



1. Click [Security] tab.
2. Click [Manage Certificates...] button.



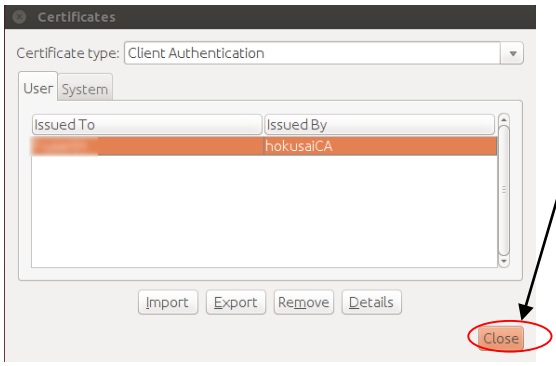
3. Select [Client Authentication] as [Certificate type].
4. Click [Import] button.



5. Select the client certificate provided by ACCC.
6. Click [OK] button.

7. Enter the password for "Client Certificate" issued by ACCC.
8. Click [OK] button.

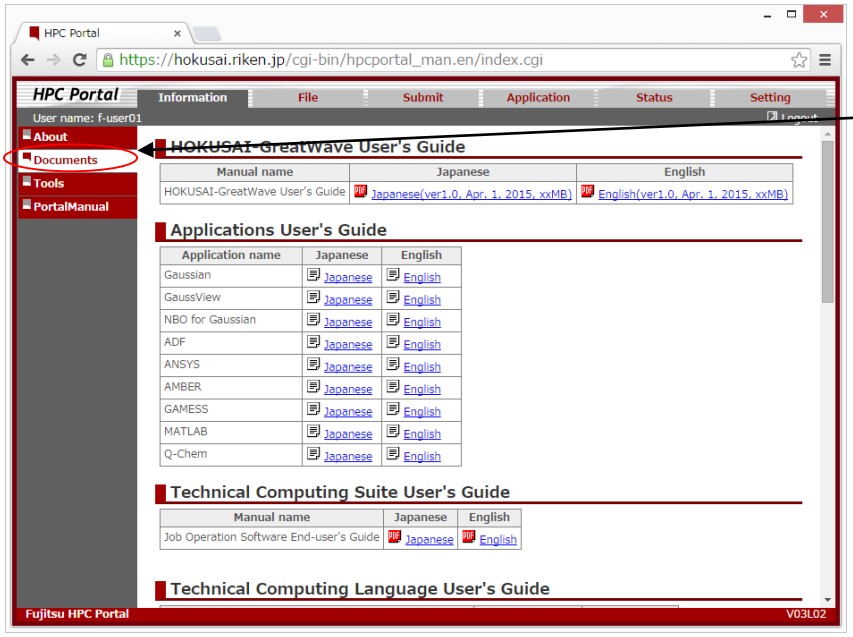
9. Enter the password for "Client Certificate" issued by ACCC.
10. Click [OK] button.



12. Click [Close] button.

7.2 **Manuals**

All reference manuals can be downloaded from the User Portal.



Click [Documents]

Figure 7-1 Screen of Documents