

FUJITSU Software

Technical Computing Suite V2.0

A horizontal band with a red-to-dark-red gradient, featuring abstract, glowing white and red lines that swirl and intersect, creating a sense of motion and energy.

Profiler User's Guide

(PRIMEHPC FX100)

J2UL-1891-02ENZ0(00)
November 2015

Preface

Purpose of This Manual

This guide describes the features and usage of the Profiler statistical information (referred to as the "Instant Profiler" in this guide), the section information (referred to as the "Advanced Profiler" in this guide), and the time series information (referred to as the "Tracer" in this guide).

Intended Readers

This guide is intended for those who use the Profiler to tune up applications. It is assumed that readers of this guide have the following knowledge:

- Knowledge of developing programs and basic Linux commands
- Knowledge of Microsoft(R) Excel(R)

Organization of This Manual

[Chapter 1 Overview of the Profiler](#)

Provides a functional overview of the Profiler

[Chapter 2 Instant Profiler](#)

Describes the output information of the Instant Profiler, including the types of information and how to use it

[Chapter 3 Advanced Profiler](#)

Describes the output information of the Advanced Profiler, including the types of information and how to use it

[Chapter 4 Tracer](#)

Explains how to use the Tracer

[Chapter 5 Tofu PA](#)

Explains how to use Tofu PA

[Chapter 6 Open Source Profiler](#)

Explains how to use mpiP as open source profiler.

[Chapter 7 Glossary](#)

Describes the terminology used in this guide

[Appendix A Considerations for Using the Profiler](#)

Describes the key points to consider when using the Profiler

[Appendix B Troubleshooting](#)

Explains how to troubleshoot the Profiler

[Appendix C Notes on Migration from FX10 system to FX100 system](#)

Explains notes when migrating from FX10 to FX100

[Appendix D Compatibility Information \(FX10 system\)](#)

Provides compatibility information as notes on migrating

Syntax Description Symbols

A syntax description symbol is a symbol that has specific meaning in syntax. The following symbols are used in this guide.

Symbol name	Symbol	Description
Selection symbols	{ }	Indicates that only one of the enclosed items can be selected

Symbol name	Symbol	Description
		Indicates that it is used as a delimiter in a list of items
Optional symbol	[]	Indicates that the enclosed item can be omitted The { } (braces selection symbols) and [] (brackets optional symbol) have the same meaning.
Default symbol	<u>–</u> (underline)	Indicates the default value when all items enclosed in [] (brackets optional symbol) are omitted
Repeat symbol	...	Indicates that the item just before this can be specified repeatedly

Abbreviations

The following abbreviations are used in this manual:

Full Name	Abbreviation
Microsoft(R) Office Excel(R) 2007 Microsoft(R) Excel(R) 2010 Microsoft(R) Excel(R) 2013 Microsoft(R) Excel(R) 2011 for Mac	Excel

Export Controls

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

Trademarks

- Linux is a trademark or registered trademark of Linus Torvalds in the United States and other countries.
- OpenMP is a trademark of OpenMP Architecture Review Board.
- Microsoft, and Excel are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.
- Mac is a registered trademark of Apple Inc.
- All other trademarks or registered trademarks appearing in this manual are trademarks or registered trademarks of their respective owners.

Date of Publication and Version

Version	Manual code
November 2015, 2nd Version	J2UL-1891-02ENZ0(00)
February 2015, Version 1.1	J2UL-1891-01ENZ0(01)
October 2014, 1st Version	J2UL-1891-01ENZ0(00)

Copyright

Copyright FUJITSU LIMITED 2014-2015

Update History

Changes	Location	Version
The article COARRAY feature is added.	2.2.2 A.1 A.2	2nd Version

Changes	Location	Version
	A.3 A.5	
The article Link Time Optimization is added.	2.2.2 A.1	
Table 2.1 fipp command options is modified.	2.2.3	
The article of Instant Profiler information (GUI format) is modified.	2.3.3.3.2	
Table 3.4 fapp command options is modified.	3.2.5	
The article of Advanced Profiler information (GUI format) is modified.	3.3.3.3.1	
The article "MPI 3.0 function" is added.	3.4.4	
The article of Largepage information is modified.	3.4.6	
The article of Precision PA visibility function (Excel format) is modified.	3.5.4	
The article of Sampling intervals is modified.	A.1	
The article of SIGVTALRM is modified.	A.1	
The article of Output of loop/line information is modified.	2.2.2 2.3.3.3.1 A.1	
The article of Cost information is modified.	A.1	
The article of DT_RPATH is modified.	A.1 A.2	
The article of -H event_number option is added.	A.2	
The article of vtunify-mpi application is added.	A.3	
Fixed the error in writing.	-	
The article on "-T t_no" and "-p p_no" in Table 2.2 fipp command options is corrected.	2.2.4	Version 1.1
The article of Advanced Profiler routine is corrected.	3.2.1	
The article of Advanced Profiler routine (precision PA) is corrected.	3.2.2	
The article on "-p p_no" in Table 3.5 fapp command options is corrected.	3.2.6	
Viewing the Excel sheets is changed.	3.5.4	
The function to insert MPI_Barrier before or after collective communication functions is added.	6.2.3 6.3 A.5	
The article of DT_RPATH is corrected.	A.1 A.2	
The article of cost information is corrected.	A.1	
MPI profiling interface is added.	A.1 A.2	

All rights reserved.
The information in this manual is subject to change without notice.

Contents

Chapter 1 Overview of the Profiler.....	1
1.1 Tuning and Profiler.....	1
1.2 Functional overview.....	1
Chapter 2 Instant Profiler.....	3
2.1 Overview of the Instant Profiler.....	3
2.2 Using the Instant Profiler.....	3
2.2.1 Environment variables.....	4
2.2.2 Compilation.....	5
2.2.3 fipp command.....	5
2.2.4 fippxx command.....	8
2.2.5 Section specification feature of the Instant Profiler.....	10
2.3 Instant Profiler information (GUI format).....	11
2.3.1 Overview of the Profiler feature.....	12
2.3.2 Starting the Profiler.....	12
2.3.3 Profiler information window.....	12
2.3.3.1 Profiling data selection window.....	13
2.3.3.2 Overview of the Instant Profiler information window.....	13
2.3.3.3 Instant Profiler information.....	16
2.3.3.3.1 Summary information.....	16
2.3.3.3.2 Topology/Panel information.....	26
2.3.3.3.3 Bar Chart information.....	29
2.3.3.3.4 Data Compare information.....	29
2.3.3.4 Source code information.....	30
2.3.3.5 Call Graph information.....	32
2.4 Instant Profiler information (text/CSV formats).....	34
2.4.1 Overview of the Profiler feature.....	34
2.4.2 Environment information for instant profiling data collection.....	35
2.4.3 Time statistical information.....	36
2.4.4 Hardware monitor information.....	36
2.4.4.1 Measured information of the Hardware monitor information.....	37
2.4.4.2 Formulas of the Hardware monitor information.....	38
2.4.4.3 Using the Hardware monitor information.....	41
2.4.4.4 Output format of the Hardware monitor information.....	41
2.4.5 Cost information.....	43
2.4.6 Call Graph information.....	47
2.4.7 Source code information.....	47
Chapter 3 Advanced Profiler.....	49
3.1 Overview of the Advanced Profiler.....	49
3.2 Using the Advanced Profiler.....	50
3.2.1 Advanced Profiler routine.....	50
3.2.2 Advanced Profiler routine (precision PA).....	52
3.2.3 Environment variables.....	53
3.2.4 Compilation.....	53
3.2.5 fapp command.....	53
3.2.6 fappxx command.....	56
3.3 Advanced Profiler information (GUI format).....	58
3.3.1 Overview of the Profiler feature.....	58
3.3.2 Starting the Profiler.....	59
3.3.3 Profiler information window.....	59
3.3.3.1 Profiling data selection window.....	59
3.3.3.2 Overview of the Advanced Profiler information window.....	60
3.3.3.3 Advanced Profiler information.....	62
3.3.3.3.1 Topology/Panel information.....	63

3.3.3.3.2 Bar Chart information.....	85
3.3.3.3.3 Data Compare information.....	85
3.4 Advanced Profiler information (text/CSV formats).....	85
3.4.1 Overview of the Advanced Profiler feature.....	85
3.4.2 Environment information for advanced profiling data collection.....	86
3.4.3 Basic information.....	87
3.4.4 MPI information.....	87
3.4.4.1 Formulas of the message length.....	88
3.4.4.2 Output format of the MPI information.....	90
3.4.5 Hardware monitor information.....	91
3.4.5.1 Events list.....	91
3.4.5.2 Formulas of the Hardware monitor information.....	94
3.4.5.3 Using the Hardware monitor information.....	99
3.4.5.4 Output format of the Hardware monitor information.....	100
3.4.6 Largepage information.....	114
3.4.6.1 Measurement information on Largepage memory use information.....	114
3.4.6.2 Output format of largepage memory use information.....	115
3.4.6.3 Measurement information on Largepage statistical information.....	115
3.4.6.4 Output format of largepage statistical information.....	117
3.5 Precision PA visibility function (Excel format).....	119
3.5.1 Overview.....	119
3.5.2 Collecting data (execution).....	120
3.5.2.1 Specifying the measurement range.....	120
3.5.2.2 Compiling/linking.....	120
3.5.2.3 Collecting data.....	120
3.5.3 Analyzing data.....	121
3.5.3.1 Converting data.....	121
3.5.3.2 Excel operations.....	121
3.5.3.2.1 Resolving security warnings.....	121
3.5.3.2.2 Specifying a process number.....	121
3.5.3.2.3 Specifying the segment name (measurement range).....	122
3.5.3.2.4 Generating Excel sheets.....	122
3.5.4 Viewing the Excel sheets.....	123
3.5.4.1 Performance information.....	124
3.5.4.2 Memory Cache information.....	125
3.5.4.3 SIMD information.....	126
3.5.4.4 Cache information.....	127
3.5.4.5 Instruction information.....	128
3.5.4.6 Balance information.....	129
3.5.4.7 XFILL flag.....	129
3.5.4.8 Time information.....	130
Chapter 4 Tracer.....	133
4.1 Overview of the Tracer.....	133
4.1.1 Overview of features.....	133
4.1.1.1 Information collection feature.....	133
4.1.1.2 Local trace data files integration feature.....	133
4.1.2 Preparation for using the Tracer.....	133
4.1.2.1 Compilation/Integration environment.....	133
4.1.2.2 Compilation/Execution/Integration environment.....	134
4.1.3 Flow for using the Tracer.....	134
4.1.3.1 Compilation.....	135
4.1.3.2 Information collection.....	135
4.1.3.3 Local trace data files integration.....	136
4.2 Using the Tracer.....	137
4.2.1 Compilation.....	137
4.2.1.1 Format.....	137

4.2.1.2 Options.....	138
4.2.1.3 Operand.....	139
4.2.1.4 Environment variables for compilation.....	139
4.2.1.5 Example of compilation.....	140
4.2.2 Information collection.....	141
4.2.2.1 Environment variables for execution.....	141
4.2.3 Local trace data file integration feature.....	143
4.2.3.1 Format.....	145
4.2.3.2 Operand.....	145
4.2.3.3 Options.....	146
4.2.3.4 Example of execution.....	146
4.2.3.5 Trace data files.....	147
4.3 Trace information.....	148
4.3.1 MPI trace.....	148
4.3.1.1 Compilation.....	148
4.3.1.2 Execution.....	148
4.3.1.3 MPI functions collected by the Tracer.....	148
4.3.2 User function trace.....	150
4.3.2.1 Compilation.....	150
4.3.2.2 Execution.....	151
4.3.3 VampirTrace API trace.....	151
4.3.3.1 Usage.....	151
4.3.3.2 Compilation.....	152
4.3.3.3 Execution.....	152
4.3.4 I/O trace.....	152
4.3.4.1 Compilation.....	152
4.3.4.2 Execution.....	153
4.3.4.3 I/O functions collected by the Tracer.....	153
4.3.5 Memory trace.....	153
4.3.5.1 Compilation.....	153
4.3.5.2 Execution.....	154
4.3.5.3 Memory functions collected by the Tracer.....	154
Chapter 5 Tofu PA.....	155
5.1 Overview of Tofu PA.....	155
5.1.1 Tuning and Tofu PA information acquisition feature.....	155
5.1.2 Overview of the feature.....	155
5.2 Using the Tofu PA information acquisition feature.....	155
5.2.1 Overview of the Tofu PA information acquisition feature.....	155
5.2.2 Specifying the measurement section.....	157
5.2.3 Compilation.....	160
5.2.4 Execution.....	160
5.2.5 Output file name.....	162
5.2.6 File formats.....	162
5.2.7 Visibility.....	164
Chapter 6 Open Source Profiler.....	166
6.1 Overview of mpiP.....	166
6.2 Using mpiP.....	166
6.2.1 Compilation.....	166
6.2.2 Linking.....	166
6.2.3 Execution.....	167
6.2.4 mpiP Output.....	168
6.3 Functional Detail.....	169
6.3.1 mpiP Report Information.....	169
6.3.1.1 Header Information.....	169
6.3.1.2 MPI Time Information.....	170
6.3.1.3 Callsite Information.....	170

6.3.1.4 Aggregate Time Information.....	171
6.3.1.5 Aggregate Sent Message Size.....	171
6.3.1.6 Callsite Time Statics.....	172
6.3.1.7 Callsite Message Sent Statistics.....	172
6.3.2 Control of Profiling Range for mpiP.....	173
6.3.3 MPI functions collected by mpiP.....	173
Chapter 7 Glossary.....	176
Appendix A Considerations for Using the Profiler.....	180
A.1 Instant Profiler.....	180
A.2 Advanced Profiler.....	184
A.3 Tracer.....	186
A.4 Tofu PA.....	187
A.5 Open Source Profiler.....	188
Appendix B Troubleshooting.....	189
B.1 Instant Profiler.....	189
B.2 Advanced Profiler.....	189
B.3 Tracer.....	189
B.4 Tofu PA.....	190
Appendix C Notes on Migration from FX10 system to FX100 system.....	191
C.1 Measured information of Hardware monitor information in Instant Profiler is changed.....	191
C.2 Measured information of Hardware monitor information in Advanced Profiler is changed.....	191
C.3 Frequency of the collection data and the analyzing data for the precision PA visibility function (Excel format) in Advanced Profiler is changed.....	192
C.4 Presentation item of the precision PA visibility function (Excel format) in Advanced Profiler is changed.....	193
Appendix D Compatibility Information (FX10 system).....	196
D.1 Migration to V1.0L30(Generation Number:09).....	196
D.1.1 The -c option of the vtunifypx command and vtunify-mpi application of the tracer is abolished.....	196
D.1.2 VT_MAX_MPI_COMMS/VT_MAX_MPI_WINS of the environment variable for execution is abolished.....	196
D.1.3 The record in the trace data of the MPI_Address function is abolished.....	197

Chapter 1 Overview of the Profiler

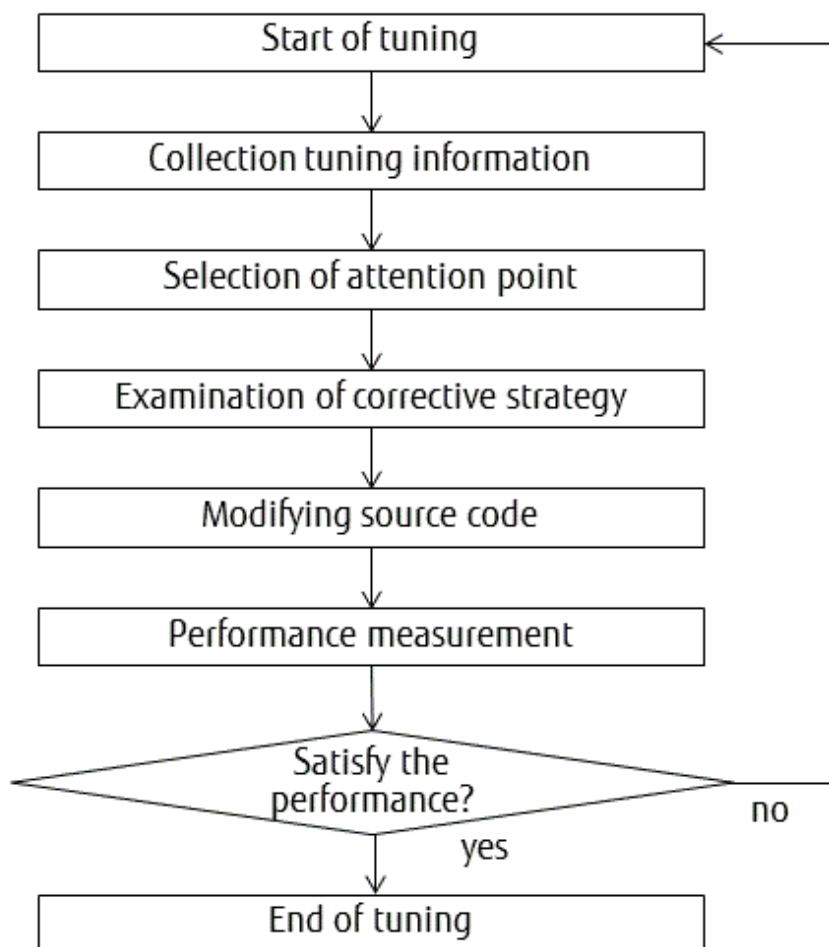
This chapter describes the features and usage of the Profiler.

1.1 Tuning and Profiler

Tuning means the improvement of an application so that the execution of the application takes considerably less time. To tune an application, collect the tuning information, consider a way to improve the application, modify it, and measure the performance, as shown in "Figure 1.1 Tuning operation". Usually, significant tuning can be achieved by finding the part in the application with the most executions and speeding it up. The tuning information (for example, the distribution of execution time) can be obtained using the Profiler. When tuning starts, it is recommended you analyze the application using the Profiler.

The Profiler can collect the tuning information for an application created by the compiler of this product.

Figure 1.1 Tuning operation



1.2 Functional overview

The Profiler consists of the Instant Profiler and the Advanced Profiler. The former collects the tuning information by the sampling system, and the latter collects the tuning information of a specific section. Additionally, the Tracer collects the tuning information on the time series. The basic features of the Instant Profiler, the Advanced Profiler, and the Tracer are listed below.

The Instant Profiler collects and outputs the execution performance information for an application by using commands. Refer to "[Chapter 2 Instant Profiler](#)" for information on the commands.

The Instant Profiler outputs the following information. Refer to "[Chapter 2 Instant Profiler](#)" for details.

- Time statistical information
Outputs the breakdown of the elapsed time, the user CPU time, and the system CPU time
- Cost information
Performs sampling while executing an application and outputs the count by the unit of the procedure, loop, or line as a cost
- Hardware monitor information
Outputs the processor state on executing an application
- Call Graph information
Outputs the call route of a procedure with the cost
- Source code information
Outputs the source code with the cost information added to each line

The Advanced Profiler collects and outputs execution performance information for an application by using commands. Refer to "[Chapter 3 Advanced Profiler](#)" for information on the commands.

The Advanced Profiler outputs the following information. Refer to "[Chapter 3 Advanced Profiler](#)" for details.

- Basic information
Outputs the call count and time information of the measurement section
- MPI information
Outputs the average, maximum, and minimum values of the count, message length, elapsed time, and wait time of MPI functions
- Hardware monitor information
Outputs the processor state on executing an application
- Largepage performance information
Largepage performance information in the measurement section is output.

The Tracer collects the execution information for the time series of an application. Refer to "[Chapter 4 Tracer](#)" for details.

- Execution information
Collects the execution information for the time series of the MPI library and user functions

Chapter 2 Instant Profiler

This chapter describes the features and usage of the Instant Profiler.

2.1 Overview of the Instant Profiler

The Instant Profiler collects the statistical tuning information for an entire application.

The basic functions of the Instant Profiler are described below.

The Instant Profiler collects and outputs execution performance information for an application using the `fipp` and `fipppx` commands.

Refer to "[2.2.3 fipp command](#)" for information on the `fipp` command.

The Instant Profiler outputs the following information:

Time statistical information

Outputs the breakdown of the elapsed time, the user CPU time, and the system CPU time

Cost information

Performs sampling while executing an application and outputs the count by the unit of procedure, loop, or line as a cost

Hardware monitor information

Outputs the processor state on executing the application

Call Graph information

Outputs the call route of a procedure with the cost

Source code information

Outputs the source code with the Cost information added to each line

Moreover, it is also possible to output the tuning information for a specific section by using the following feature:

Section specification feature of the Instant Profiler

The Time statistical information, the Cost information, the Hardware monitor information, the Call Graph information, and the Source code information for the specified time base range are output.

2.2 Using the Instant Profiler

The following commands can be used with the Instant Profiler.

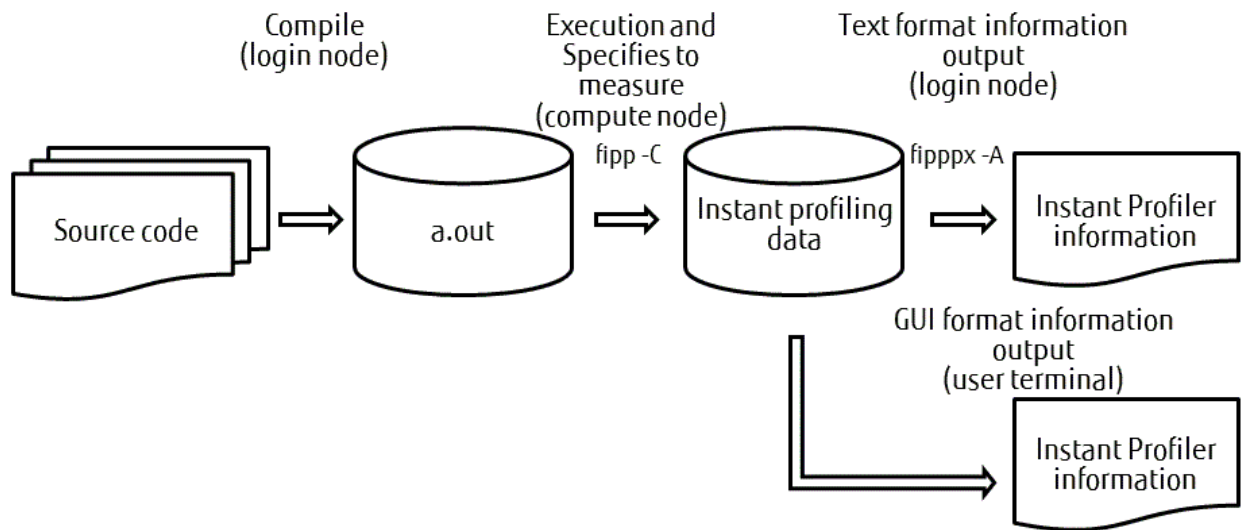
`fipp`

Collects the instant profiling data for an application in the compute node of the FX100 system

`fipppx`

Outputs the contents of the instant profiling data on the login node at the front end

Figure 2.1 Collection of the instant profiling data and output of the Instant Profiler information



fipp -C : Collects the instant profiling data
 fipp -A : Outputs the Instant Profiler information

Collection of the instant profiling data

The fipp command collects the instant profiling data.

Output of the Instant Profiler information

The Instant Profiler information is output in the following formats:

GUI format

The Instant Profiler information can be output in the GUI format.

Refer to "2.3 Instant Profiler information (GUI format)" for information on how to output the GUI format.

Text format/CSV format

The fipp command outputs the Instant Profiler information in the text or CSV format. It uses the saved instant profiling data to output the Instant Profiler information in the text or CSV format.

Refer to "2.4 Instant Profiler information (text/CSV formats)" for details.

The usage of the Instant Profiler is described below.

2.2.1 Environment variables

It is necessary to correctly set the following environment variables to use the Instant Profiler.

Environment variable	Value
PATH	/opt/FJSVmxlang/bin
LD_LIBRARY_PATH	/opt/FJSVmxlang/lib64

To use the batch queuing system and the MPI processing system, setting additional values besides those mentioned above may be necessary. Refer to the "Job Operation Software First Step Guide" for information on the batch queuing system. Refer to the "MPI User's Guide" for information on the MPI system.

Usage example

Usage example of the Instant Profiler is shown below:

- Example 1 : An MPI application, a.out, of two parallels is executed, and the tuning information is collected.

```
$ fipp -C -d FIPP_Example mpiexec -n 2 ./a.out
```

- Example2 : The Instant Profiler information is output in the text format using the output instant profiling data, "FIPP_Example".

```
$ fipp -A FIPP_Example
```

2.2.2 Compilation

To use the Instant Profiler, it is necessary to create an application that is linked with the tool library.

The behavior of the tool can be manipulated by using options of the `frtpx/fccpx/FCCpx/mpifrtpx/mpifccpx/mpiFCCpx` commands.

Refer to the "Fortran User's Guide", the "C User's Guide", the "C++ User's Guide", or the "MPI User's Guide" for information on how to specify these options.

The format and description of the option for the tool are described below.

[`-Ntl_trt` | `-Ntl_notrt`]

This option specifies whether to create an application that is linked with the tool libraries. It is necessary to specify this option when linking. The default is `"-Ntl_trt"`.

`-Ntl_trt`

An application that is linked with the tool libraries is created. In this case, when the application is executed, the Debugger and the Profiler features can be used.

`-Ntl_notrt`

An application is created without linking it with the tool libraries. In this case, when the application is executed, the Debugger and the Profiler features cannot be used.

These following points must be taken into consideration when using the Instant Profiler.

Notes

Fortran

Refer to "[-P userfunc option](#)" in "[Appendix A Considerations for Using the Profiler](#)" if `"-Nnoline"` (compiler option) is enabled.

Refer to "[COARRAY feature](#)" in "[Appendix A Considerations for Using the Profiler](#)" if `"-Ncoarray"` (compiler option) is enabled.

Refer to "[Link Time Optimization](#)" in "[Appendix A Considerations for Using the Profiler](#)" if `"-Klto"` (compiler option) is enabled.

C/C++ language

Refer to "[Output of loop information](#)" in "[Appendix A Considerations for Using the Profiler](#)" if `"-O0"` (optimization option) is used.

Refer to "[-P userfunc option](#)" in "[Appendix A Considerations for Using the Profiler](#)" if `"-Nnoline"` (compiler option) is used.

Refer to "[Link Time Optimization](#)" in "[Appendix A Considerations for Using the Profiler](#)" if `"-Klto"` (compiler option) is enabled.

XPFortran

An XPFortran program is treated as an MPI program.

It is translated to an MPI program, and the XPFortran program is translated and executed in the translation and the execution environment of the MPI program. Specify the option for the tool when you translate the MPI program. If the `"-Nxpflne"` option of the XPFortran translator command is used, the line number and source information on the XPFortran program can be output.

Refer to the "XPFortran User's Guide" for details.

2.2.3 fipp command

The `fipp` command collects the instant profiling data of an application.

Format

```
fipp -C -d profiling_data [ -I item ] [ -l limit ] [ -H [hardmon] ] [ -P cost_typ ]
[ -S section ] [ -i interval ] [ -m memsize ] [ -L cost_line ] exec-file [ exec_option ... ]
```

Options

The table below describes the options that can be specified for the fipp command.

Table 2.1 fipp command options

Option	Description/Specified value (unit)
-C	Specifies the collection processing of the instant profiling data. This option is mandatory.
-I <i>item</i>	Specifies the collection items of the Instant Profiler information. Delimit using a comma if two or more items are specified for <i>item</i> . <i>item</i> : { { call nocall } { hwm nohwm } } The default is "-Inocall,nohwm".
call nocall	Specifies whether to collect the Call Graph information. - call : Collects the Call Graph information - nocall : Does not collect the Call Graph information The default is "nocall".
hwm nohwm	Specifies whether to collect the Hardware monitor information. - hwm : Collects the Hardware monitor information - nohwm : Does not collect the Hardware monitor information The default is "nohwm".
-l <i>limit</i>	Specifies the output number of the procedure information output to the instant profiling data. - <i>limit</i> : Integers from 0 through 2,147,483,647 can be specified to define the range (output number) Everything is output in case of 0. Procedure information more than the output number is output as "__other__". The default is 0.
-H [<i>hardmon</i>]	Specifies measurement of the Hardware monitor information. - <i>hardmon</i> : Measures the Hardware monitor information <i>hardmon</i> can be omitted (only "-H" can be specified). Delimit using commas if two or more items are specified for <i>hardmon</i> . The default is as follows: - If the "-lhwm" option is used : -Hevent=Statistics,mode=sys - If the "-lnohwm" option is used : Hardware monitor information is not collected It is considered that "-lhwm" is specified when this option is specified. <i>hardmon</i> : { event= <i>event</i> mode= <i>mode</i> }
event= <i>event</i> : { Instruction_SIMD MEM_access Statistics }	Specifies the measurement event of the Hardware monitor information. Any of the following can be specified for <i>event</i> : - Instructions_SIMD : Execution instruction detail (SIMD)

Option	Description/Specified value (unit)
	<ul style="list-style-type: none"> - MEM_access : Memory access situation - Statistics : CPU core operation situation <p>The default is "event=Statistics".</p>
<p>mode=<i>mode</i> :</p> <p>{ sys usr }</p>	<p>Specifies the measurement mode of the Hardware monitor information.</p> <p>One of the following can be specified for <i>mode</i>.</p> <ul style="list-style-type: none"> - sys : Collects information on the kernel mode and the user mode - usr : Collects information on the user mode <p>The default is "mode=sys".</p>
<p>-L <i>cost_line</i></p>	<p>Specifies how to collect the detail information of shared library.</p> <p>The default is "-Lnoshared".</p>
<p>shared noshared</p>	<p>Specifies whether to collect the Cost information for the shared library in the unit of the loop and in each line.</p> <ul style="list-style-type: none"> - shared : For the shared library, start line number and end line number of procedure cost distribution, loop cost distribution information and line cost distribution information is collected. - noshared : For the shared library, start line number and end line number of procedure cost distribution, loop cost distribution information and line cost distribution information is not collected.
<p>-P <i>cost_typ</i></p>	<p>Specifies how to collect the library Cost information.</p> <p>The default is "-Pnouserfunc".</p>
<p>userfunc nouserfunc</p>	<p>Specifies how to collect the costs of the library called from a user procedure.</p> <ul style="list-style-type: none"> - userfunc : The library cost called from the procedure is included in the cost of the call former user procedure. When userfunc is specified, it is necessary to specify the "-Icall" option. - nouserfunc : The library cost called from the procedure is not included in the cost of the user procedure. Costs of the library are individually collected.
<p>-S <i>section</i></p>	<p>Specifies the measurement range of the Instant Profiler information</p> <p>The default is "-Stotal".</p>
<p>total range</p>	<ul style="list-style-type: none"> - total : Collects the Instant Profiler information for an entire application - range : Collects the Instant Profiler information within the range specified by the Instant Profiler routine
<p>-d <i>profiling_data</i></p>	<p>Specifies the name of instant profiling data (directory that contains the instant profiling data file) to <i>profiling_data</i> by using the relative path or the absolute path.</p> <p>If the directory does not exist, it is created. If the directory exists, it must be empty.</p> <p>This option is mandatory.</p>
<p>-i <i>interval</i></p>	<p>Specifies the time interval between successive measurements of the Instant Profiler information.</p> <ul style="list-style-type: none"> - <i>interval</i> : Integers from 10 through 3,600,000 can be specified to define the range (millisecond) <p>The default is 100.</p> <p>Incidentally, the time interval of measurements might be changed by the measure against noise of the OS (*1).</p> <p>Refer to "Sampling interval" in "Appendix A Considerations for Using the Profiler" for details.</p> <p>*1 : It is the measure to reduce the impact of system daemons on the job.</p>

Option	Description/Specified value (unit)
<i>-m memsize</i>	Specifies the working memory size required for collecting the instant profiling data. This size area of each thread is reserved. <ul style="list-style-type: none"> - <i>memsize</i> : Integers from 1 through 2,147,483 can be specified to define the range (KB) The default is 3000.
<i>exec-file</i> [<i>exec_option ...</i>]	Specifies the target execution file for the instant profiling data collection and the option. <ul style="list-style-type: none"> - <i>exec-file</i> : Specifies mpiexec when MPI application is used. Specify the absolute path or the relative path containing the current directory ("./") if specifying the execution file that starts in "-". The shell script cannot be specified. - <i>exec_option ...</i> : Specifies the option to <i>exec-file</i>. The character string following an execution file name is regarded as the option to an execution file.

2.2.4 fippox command

The fippox command outputs the Instant Profiler information in the text or CSV format.

Format

```
fippox -A [ -I item ] [ -l limit ] [ -T t_no ] [ -f func_name ] [ -o outfile ]
      [ -p p_no ] [ -t type ] -d profiling_data
```

Options

The table below describes the options that can be specified for the fippox command.

Table 2.2 fippox command options

Option	Description/Specified value (unit)
-A	Specifies the output processing of the Instant Profiler information. This option is mandatory.
-I <i>item</i>	Specifies the output items of the Instant Profiler information. Delimit using commas if two or more items are specified for <i>item</i> . <i>item</i> : { { <i>cpu</i> <i>nocpu</i> } { <i>balance</i> <i>nobalance</i> } { <i>call</i> <i>nocall</i> } { <i>hwm</i> <i>nohwm</i> } { <i>src[:path]...</i> <i>nosrc</i> } } The default is "-Icpu".
<i>cpu</i> <i>nocpu</i>	Specifies whether to output the Cost information. <ul style="list-style-type: none"> - <i>cpu</i> : Outputs the Cost information - <i>nocpu</i> : Does not output the Cost information The default is "nocpu".
<i>balance</i> <i>nobalance</i>	Specifies whether to output the parallel balance graph of the Cost information. However, the "nobalance" option becomes effective in sequential applications, and the balance information on the Cost information is not output. <ul style="list-style-type: none"> - <i>balance</i> : Outputs parallel balance graph of the Cost information - <i>nobalance</i> : Does not output parallel balance graph of the Cost information The default is "nobalance".
<i>call</i> <i>nocall</i>	Specifies whether to output the Call Graph information.

Option	Description/Specified value (unit)
	<ul style="list-style-type: none"> - call : Outputs the Call Graph information - nocall : Does not output the Call Graph information <p>The default is "nocall".</p>
hwm nohwm	<p>Specifies whether to output the Hardware monitor information.</p> <ul style="list-style-type: none"> - hwm : Outputs the Hardware monitor information - nohwm : Does not output the Hardware monitor information <p>The default is "nohwm".</p>
src[: <i>path</i>] ... nosrc	<p>Specifies whether to output the Source code information.</p> <ul style="list-style-type: none"> - src[:<i>path</i>] ... : Outputs the Source code information Specify the directory path where the source code exists as <i>path</i>. Delimit using colon(:) if two or more items are specified for <i>path</i>. If <i>path</i> is omitted, the source file path specified at application compilation is referred. - nosrc : Does not output the Source code information <p>The default is "nosrc".</p>
-l <i>limit</i>	<p>Specifies the number of procedure information output to the Cost information of the Instant Profiler information.</p> <ul style="list-style-type: none"> - <i>limit</i> : Integers from 0 through 2,147,483,647 can be specified to define the range (output number) <p>Everything is output in case of 0.</p> <p>The default is 10.</p>
-T <i>t_no</i>	<p>Specifies the target thread to output the Instant Profiler information.</p> <p>Delimit using commas if two or more target thread numbers (<i>t_no</i>) are specified. The one specified later is valid, if two or more target thread numbers (<i>t_no</i>) are specified. An error is detected, if <i>t_no</i> is omitted.</p> <p><i>t_no</i> : { all <i>N</i>[,<i>N</i>]... limit=<i>n</i> }</p> <p>The default is "-Tall".</p>
all <i>N</i> [, <i>N</i>] ... limit= <i>n</i>	<ul style="list-style-type: none"> - all : Information on all threads is output in order with a high cost. - <i>N</i>[,<i>N</i>] ... : Information on thread number <i>N</i> is output ahead of information on the thread with a high cost. When thread number <i>N</i> does not exist, specification is disregarded. - limit=<i>n</i> : Information on <i>n</i> threads is output. When 0 or the value that exceeds the number of threads is specified for <i>n</i>, information on all threads is output. The default is "limit=0".
-f <i>func_name</i>	<p>Specifies to output the Instant Profiler information of a specific procedure (<i>func_name</i>). <i>func_name</i> is the procedure name used by the application.</p> <p>Even if the cost of the procedure <i>func_name</i> is outside the range of the "-l" option, the Instant Profiler information about <i>func_name</i> is output.</p> <p>The Instant Profiler information about <i>func_name</i> is not output in the following cases.</p> <ul style="list-style-type: none"> - If the information of the procedure <i>func_name</i> is not output to the instant profiling data. - If the cost of the procedure <i>func_name</i> is 0.
-o <i>outfile</i>	<p>Specifies the output destination of the Instant Profiler information.</p>

Option	Description/Specified value (unit)
	<p>If stdout is specified as <i>outfile</i>, the Instant Profiler information is output to the standard output. Specify the absolute path or the relative path containing the current directory (".") if specifying <i>outfile</i> that starts in "-".</p> <p>The default is "-ostdout".</p>
-p <i>p_no</i>	<p>Specifies the target process to be input and output with the Instant Profiler information.</p> <p>The information of application unit of the Instant Profiler information is calculated using the instant profiling data of target process specified by this option.</p> <p>Delimit using commas if two or more target process numbers (<i>p_no</i>) are specified.</p> <p>The one specified later is valid, if two or more target process numbers (<i>p_no</i>) are specified.</p> <p>An error is detected, if <i>p_no</i> is omitted.</p> <p><i>p_no</i>: { all <i>N</i>,<i>N</i>... input=<i>n</i> limit=<i>m</i> }</p> <p>The default is "-pinput=0,limit=16".</p>
all <i>N</i> , <i>N</i> ... input= <i>n</i> limit= <i>m</i>	<ul style="list-style-type: none"> - all : Data on all processes is input and information on all processes is output in order with a high cost. - <i>N</i>,<i>N</i> ... : Data on process number <i>N</i> is input and information on process number <i>N</i> is output ahead of information on the process with a high cost. When process number <i>N</i> does not exist, specification is disregarded. - input=<i>n</i> : Data on <i>n</i> processes is input. When the value that exceeds 0 or the number of processes is specified for <i>n</i>, data on all processes is input. The default is "input=0". - limit=<i>m</i> : Information on <i>m</i> processes is output. When 0 or the value that exceeds <i>n</i> is specified for <i>m</i>, information on <i>n</i> processes is output. The default is "limit=16".
-t <i>type</i>	<p>Specifies the output format of the Instant Profiler information.</p> <p>The default is "-ttext".</p>
csv text	<ul style="list-style-type: none"> - csv : Outputs the Instant Profiler information in the CSV format If the "csv", "-Ibalance", or "-Isrc" option is specified with any other option of them, the cost balance information or the Source code information is not output. - text : Outputs the Instant Profiler information in the text format
-d <i>profiling_data</i>	<p>Specifies the name of instant profiling data (directory that contains the instant profiling data file) to <i>profiling_data</i> by using the relative path or the absolute path.</p> <p>If specifying <i>profiling_data</i> that starts in "-", specify the absolute path or the relative path containing the current directory (".").</p> <p>When this option is specified at the end of an optional list of the fippxx command, "-d" can be omitted.</p> <p>This option is mandatory.</p>

2.2.5 Section specification feature of the Instant Profiler

The Instant Profiler section specification feature allows you to specify a range to measure the Cost information. To specify a time base range on the source code, a subroutine is inserted at the measurement start position and the position where the Cost information measurement ends.

The Instant Profiler section specification feature can be used as a subroutine of the Fortran language or a function of the C/C++ language. When a function of the C or C++ language is used, the prototype of the function is declared. Otherwise, you need to include the header file of the Instant Profiler section specification feature. To collect the instant profiling data by using the Instant Profiler section specification feature, specify the object application for an operand of the fipp command and specify the "-Srange" option.

Refer to "2.2.3 fipp command" for information on the "-Srange" option.

The following table provides an overview of the section specification feature of the Instant Profiler.

Table 2.3 Overview of the section specification feature of the Instant Profiler

Language	Header file	Function name (Instant Profiler routine)	Function	Arguments
Fortran	-	fipp_start	Starts the Cost information measurement	-
		fipp_stop	Ends the Cost information measurement	-
C/C++	fj_tool/fipp.h	void fipp_start	Starts the Cost information measurement	-
		void fipp_stop	Ends the Cost information measurement	-

Two or more time base ranges of the Cost information can be specified. However, the time base range of the Cost information cannot be specified for a nest.

A usage example of the Instant Profiler section specification feature is given below.



Example

Usage example of the section specification feature of the Instant Profiler

```
#include <fj_tool/fipp.h>

#define SIZE 3000
double a[SIZE][SIZE],b[SIZE][SIZE],c[SIZE][SIZE];

main()
{
    int i,j;

    fipp_start();

    for(i=0;i<SIZE;i++){
        for(j=0;j<SIZE;j++){
            a[i][j]=(double)(i+j*0.5);
            b[i][j]=(double)(i+j*1.5);
            c[i][j]=a[i][j]+b[i][j];
        }
    }

    fipp_stop();
}
```

Execute the Instant Profiler section specification feature for a parallel-process application in the process whose Cost information is to be measured.

For instance, to measure the Cost information for all processes in a parallel-process application, execute the Instant Profiler section specification feature in the part where all the processes operate.

2.3 Instant Profiler information (GUI format)

This section describes the Instant Profiler information.

The Instant Profiler information is displayed using the Profiler GUI.

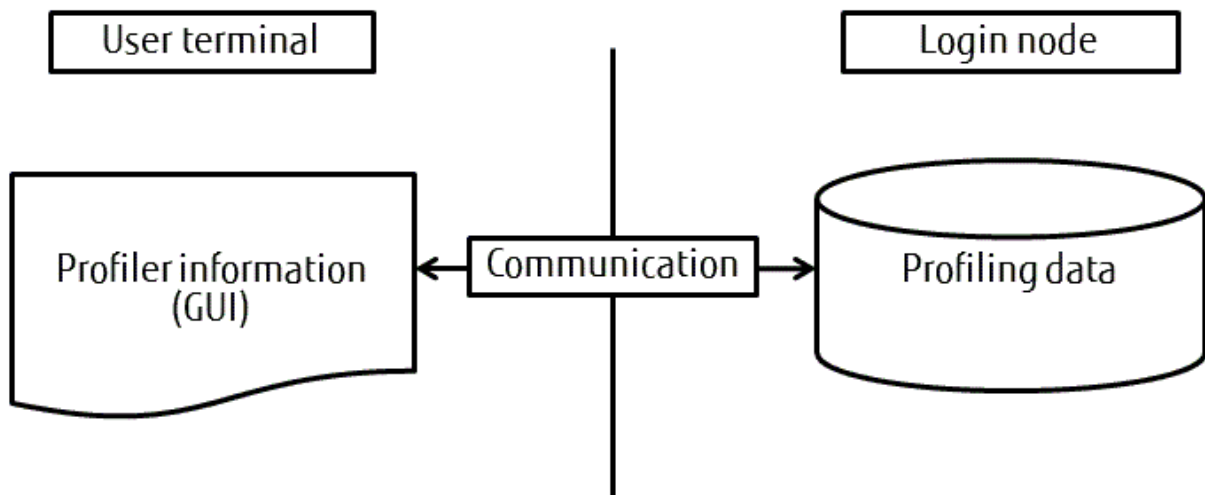
2.3.1 Overview of the Profiler feature

The profiler function analyses and gives visibility to the profiling data collected by the Instant Profiler (fipp command) and the Advanced Profiler (fapp command).

The data collected by the Instant Profiler is referred to as the instant profiling data, and the data collected by the Advanced Profiler is referred to as the advanced profiling data.

The diagram of Profiler information is shown below.

Figure 2.2 Diagram of Profiler information



Specify the profiling data on the login node to visualize it using the Profiler feature of the user terminal.

2.3.2 Starting the Profiler

Click the **Profiler** icon in the main window of FUJITSU Software Development Tools (FSDT) to start the Profiler. Refer to the "Programming Workbench User's Guide" for information on FSDT.

Figure 2.3 Profiler icon



2.3.3 Profiler information window

This section describes the Profiler information window.

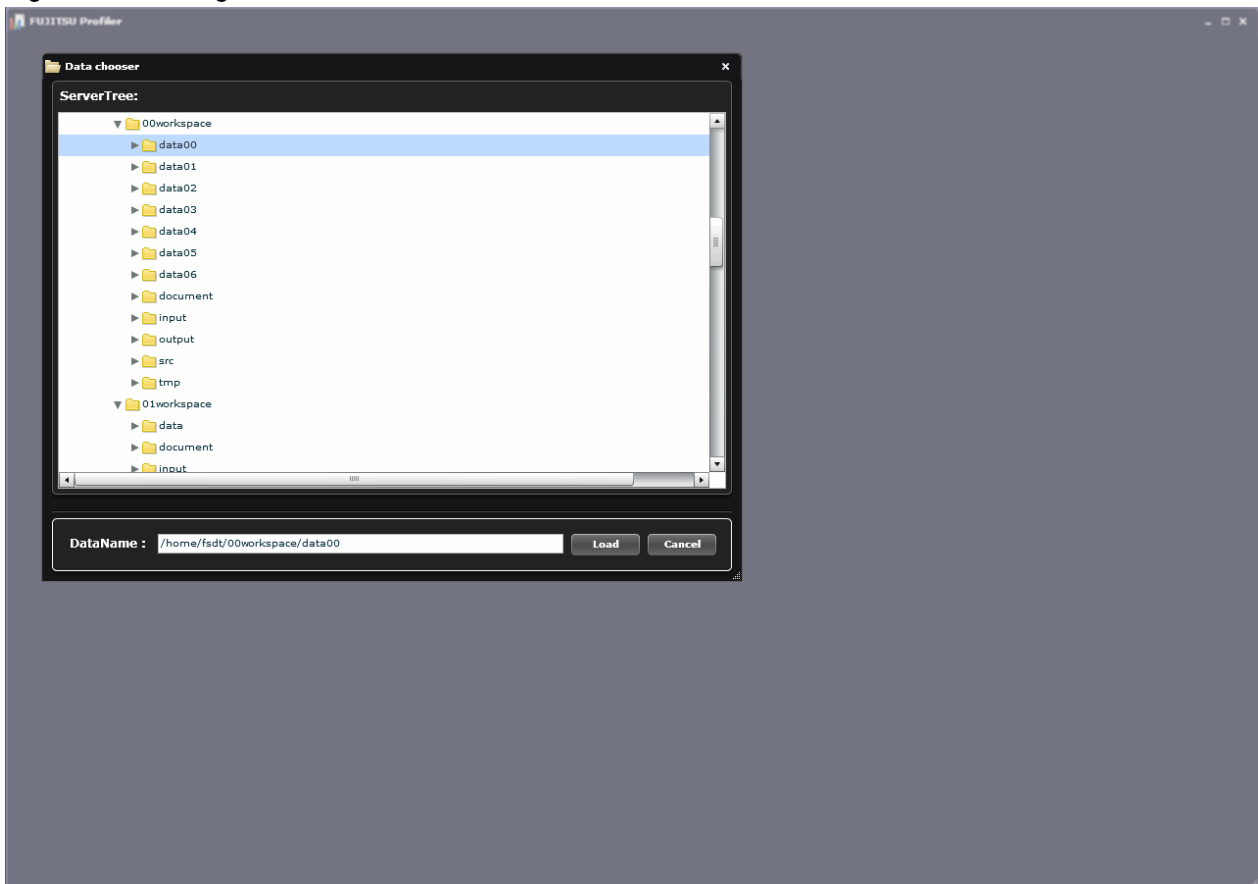
2.3.3.1 Profiling data selection window

When the Profiler is started, the Profiling data selection window is displayed. The directories on the login node are displayed in the profiling data selection window in a tree structure. Select the instant profiling data from the tree, and then click **Load** to start the data reading. When the data reading completes, the Instant Profiler information window is displayed.

Note that the instant profiling data is a directory.

Up to 9216 parallel processes can be displayed by the Profiler. Profiling data of parallel processes that exceed this count cannot be used by the Profiler.

Figure 2.4 Profiling data selection window

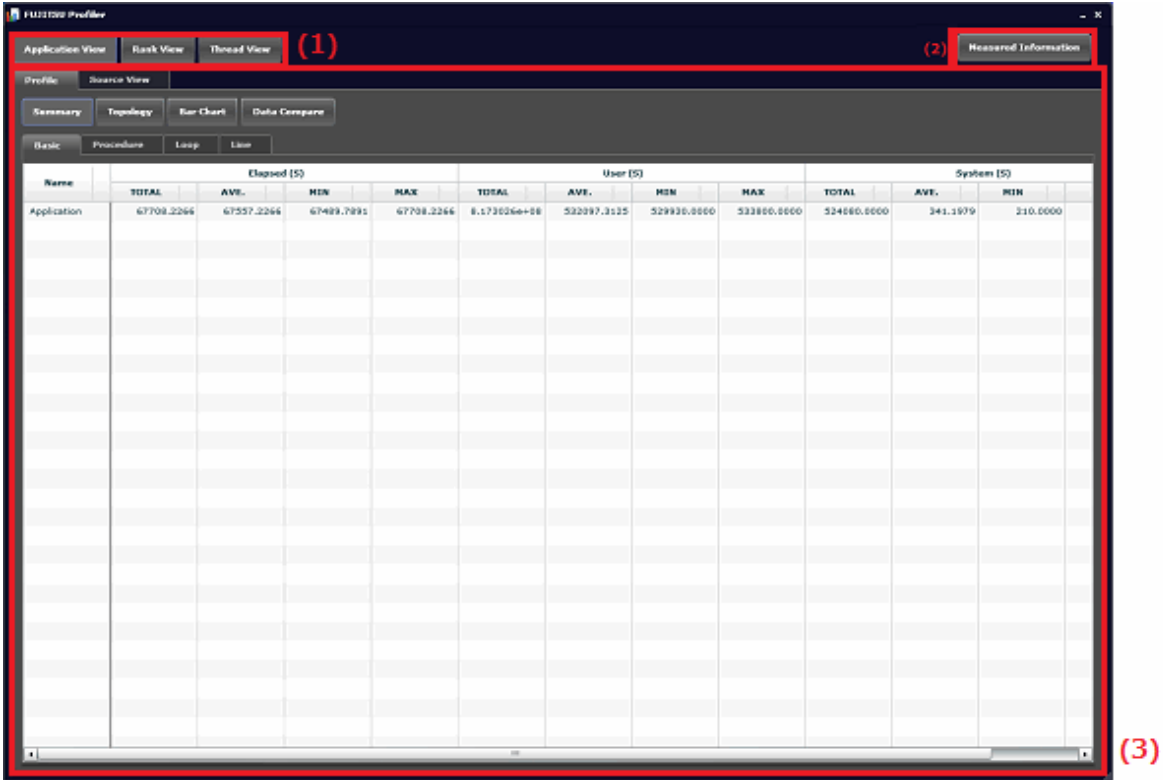


2.3.3.2 Overview of the Instant Profiler information window

When the instant profiling data is input, the Instant Profiler information window is displayed. This window comprises the following elements:

- (1) Display unit switching buttons
- (2) Measured Information button
- (3) Instant Profiler information area

Figure 2.5 Instant Profiler information window



(1) Display unit switching buttons

The unit of display shown in the Instant Profiler information window can be switched.

Table 2.4 Display unit

Button	Display unit
Application View	Total information on each application is displayed.
Rank View	Total information on a specific rank is displayed
Thread View	Information on a specific thread is displayed

If either **Rank View** or **Thread View** is selected, a box for rank selection is displayed.

If **Thread View** is selected, a box for thread selection is displayed.

(2) Measured Information button

Click the Measured Information button in the Instant Profiler information window to display the Measured Information window. The measured state of an application, such as the frequency of the machine and the measured information type, is displayed in the Measured Information window.

Figure 2.6 Measured Information window

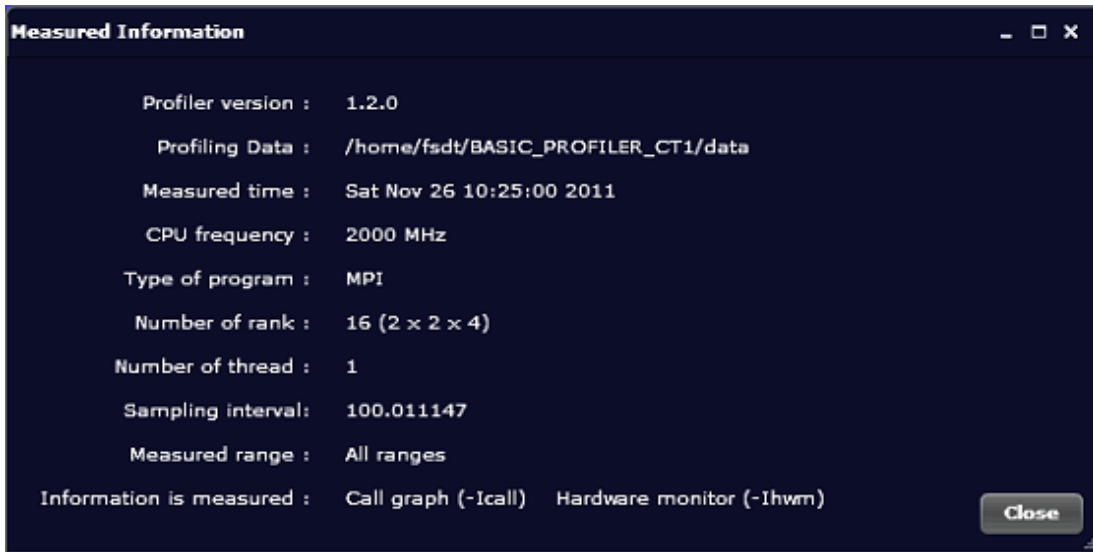


Table 2.5 Measured Information

Label	Information
Profiler version	Version of the Profiler feature
Profiling Data	Name of the displayed profiling data
Measured time	Measured date and time
CPU frequency	Frequency of the measured machine
Type of program	Execution format of the application <ul style="list-style-type: none"> - SERIAL : Sequential - Thread : Thread parallel - Thread (OpenMP) : Thread parallel by OpenMP - Thread (AUTO) : Thread parallel by automatic parallel function of the compiler - MPI : MPI
Number of rank	Number of ranks
Number of thread	Number of threads
Sampling interval	Average sampling interval
Measured range	Measured range <ul style="list-style-type: none"> - All range : Measurement of the entire application (default) - Selected range : Sampling section specification measurement ("-Srange" is used in the fipp command)
Information is measured	Collected information <ul style="list-style-type: none"> - Default : the Time statistical information, and the Cost information - Call graph (-Icall) : the Time statistical information, the Cost information, and the Call Graph information - Hardware monitor (-Ihwm) : the Time statistical information, the Cost information, and the Hardware monitor information - Call graph (-Icall) Hardware monitor (-Ihwm) : the Time statistical information, the Cost information, the Call Graph information, and the Hardware monitor information

(3) Instant Profiler information area

The following information is displayed in the Instant Profiler information area. The information display can be switched by using the available tabs.

Table 2.6 Instant Profiler information

Tab	Information
Profile	The Cost information is displayed according to the unit of display, such as procedure, loop, or line.
Source View	Information on the total cost of each source is displayed.
Call Graph	Information related to the call of a procedure is displayed. This information is only displayed in Thread View . Moreover, it is only displayed when the Call Graph information is measured (if "-Icall" is used in the fipp command).

2.3.3.3 Instant Profiler information

A description of the elements on the **Profile** tab is given below.

The display method can be switched using the display switch buttons in **Application View** and **Rank View**.

In the **Thread View**, there is no display switch and the summary information is always displayed.

Table 2.7 Display methods

Button	Information
Summary	The Cost information is displayed in the table format for the unit of display, such as procedure, loop, or line.
Topology	The distribution information to the rank on the Cost information for the unit of display, such as procedure, loop, or line, is displayed according to the topology shape of the application. This information is only displayed in Application View .
Panel	The distribution information to the thread on the Cost information for the unit of display, such as procedure, loop, or line is displayed. This information is only displayed in Rank View .
Bar Chart	The distribution information to the rank or the thread on the Cost information for the unit of display, such as procedure, loop, or line is displayed in the form of a Bar Chart. This information is only displayed in Application View and Rank View .
Data Compare	The whole graph of three times (for three items) of displayed by selecting the row in the Topology, Panel and Bar Chart is arranged vertically and displayed. This information is only available in Application View and Rank View .

2.3.3.3.1 Summary information

The total Cost information for each unit is displayed in the table format. The items of the Cost information differ according to the unit of the total.

Table 2.8 Total unit

Tab	Total unit
Basic	The cost is totaled by each display. This information becomes a display only of one line.
Procedure	The cost is totaled in the unit of the procedure. The cost of the synchronous thread barrier waiting function and the MPI function calls and is summed up to the cost of former user procedure.

Tab	Total unit
Loop	The cost is totaled in the unit of the loop. When compiling, optional optimization should be effective to total each loop. Refer to " Output of loop information " in " Appendix A Considerations for Using the Profiler " for details.
Line	The cost of each line of the user source is totaled.

The display items are described below.

In case of **Thread View**, because the hit cannot be downed more than it, it becomes the item of considerable Total respectively. Moreover, deflection is done and the female Bar Chart is not displayed.

Table 2.9 Display items of Basic

Label	Description
Name	Display unit name
Elapsed (S)	Elapsed time
Total	Elapsed time as the unit of display Maximum elapsed time between ranks in Application View Maximum elapsed time between threads in Rank View
AVE.	Average value between ranks in Application View Average value between threads in Rank View
MIN	Minimum value between ranks in Application View Minimum value between threads in Rank View
MAX	Maximum value between ranks in Application View Maximum value between threads in Rank View
User (S)	User CPU time
Total	Time of user CPU as the unit of display Total value between ranks in Application View Total value between threads in Rank View
AVE.	Average value between ranks in Application View Average value between threads in Rank View
MIN	Minimum value between ranks in Application View Minimum value between threads in Rank View
MAX	Maximum value between ranks in Application View Maximum value between threads in Rank View
System (S)	System CPU time
Total	System CPU time as the unit of display Total value between ranks in Application View Total value between threads in Rank View
AVE.	Average value between ranks in Application View Average value between threads in Rank View
MIN	Minimum value between ranks in Application View Minimum value between threads in Rank View

Label	Description
MAX	Maximum value between ranks in Application View Maximum value between threads in Rank View
Operation (S)	Operation time
Total	Operation time as the unit of display Total value between ranks in Application View Total value between threads in Rank View
AVE.	Average value between ranks in Application View Average value between threads in Rank View
MIN	Minimum value between ranks in Application View Minimum value between threads in Rank View
MAX	Maximum value between ranks in Application View Maximum value between threads in Rank View
Communication (S)	Communication time
Total	Communication time as the unit of display Total value between ranks in Application View Total value between threads in Rank View
AVE.	Average value between ranks in Application View Average value between threads in Rank View
MIN	Minimum value between ranks in Application View Minimum value between threads in Rank View
MAX	Maximum value between ranks in Application View Maximum value between threads in Rank View
MFLOPS	Floating-point arithmetic execution efficiency (average of the number of floating-point arithmetic executions per second) Number of operations is obtained by multiplying by 1 + 2 * Number of operations is obtained by multiplying by 2 + 4 * Number of operations is obtained by multiplying by 4 + 8 * Number of operations is obtained by multiplying by 8 + 16 * Number of operations is obtained by multiplying by 16) / Elapsed time / 1.0e+6 Number of operations is obtained by multiplying by 1 : 1FLOPS_instructions Number of operations is obtained by multiplying by 2 : 2FLOPS_instructions Number of operations is obtained by multiplying by 4 : 4FLOPS_instructions Number of operations is obtained by multiplying by 8 : 8FLOPS_instructions Number of operations is obtained by multiplying by 16 : 16FLOPS_instructions When the following options are all effective in the fipp command, this value is displayed. -lhwm and -Hevent=Statistics
AVE.	Average value between ranks in Application View Average value between threads in Rank View
MIN	Minimum value between ranks in Application View Minimum value between threads in Rank View
MAX	Maximum value between ranks in Application View

Label	Description
	Maximum value between threads in Rank View
MFLOPS/PEAK (%)	Rate of actual measurement values to the logical peak value of MFLOPS MFLOPS / (Number of execution core * MFLOPS peak value of each core) * 100 When the following options are all effective in the fipp command, this value is displayed. -lhwm and -Hevent=Statistics
AVE.	Average value between ranks in Application View Average value between threads in Rank View
MIN	Minimum value between ranks in Application View Minimum value between threads in Rank View
MAX	Maximum value between ranks in Application View Maximum value between threads in Rank View
MIPS	Instruction execution efficiency (average of the number of instruction executions per second) Number of instruction executions / Elapsed time / 1.0e+6 Number of instruction executions : effective_instruction_counts When the following options are all effective in the fipp command, this value is displayed. -lhwm and, -Hevent=Statistics or -Hevent=Instructions_SIMD
AVE.	Average value between ranks in Application View Average value between threads in Rank View
MIN	Minimum value between ranks in Application View Minimum value between threads in Rank View
MAX	Maximum value between ranks in Application View Maximum value between threads in Rank View
MIPS/PEAK (%)	Rate of actual measurement values to the logical peak value of MIPS MIPS / (Number of execution cores * MIPS peak value of each core) * 100 When the following options are all effective in the fipp command, this value is displayed. -lhwm and, -Hevent=Statistics or -Hevent=Instructions_SIMD
AVE.	Average value between ranks in Application View Average value between threads in Rank View
MIN	Minimum value between ranks in Application View Minimum value between threads in Rank View
MAX	Maximum value between ranks in Application View Maximum value between threads in Rank View
Floating point operations (%)	Rate of the number of floating-point arithmetic executions in operand of instruction (Number of operations is obtained by multiplying by 1 + 2 * Number of operations is obtained by multiplying by 2 + 4 * Number of operations is obtained by multiplying by 4 + 8 * Number of operations is obtained by multiplying by 8 + 16 * Number of operations is obtained by multiplying by 16) / (Number of instruction executions + Number of operations is obtained by multiplying by 2 + 3 * Number of operations is obtained by multiplying by 4 + 7 * Number of operations is obtained by multiplying by 8 + 15 * Number of operations is obtained by multiplying by 16) * 100 Number of operations is obtained by multiplying by 1 : 1FLOPS_instructions Number of operations is obtained by multiplying by 2 : 2FLOPS_instructions

Label	Description
	<p>Number of operations is obtained by multiplying by 4 : 4FLOPS_instructions</p> <p>Number of operations is obtained by multiplying by 8 : 8FLOPS_instructions</p> <p>Number of operations is obtained by multiplying by 16 : 16FLOPS_instructions</p> <p>Number of instruction executions : effective_instruction_counts</p> <p>When the following options are all effective in the fipp command, this value is displayed.</p> <p>-lhwm and -Hevent=Statistics</p>
AVE.	<p>Average value between ranks in Application View</p> <p>Average value between threads in Rank View</p>
MIN	<p>Minimum value between ranks in Application View</p> <p>Minimum value between threads in Rank View</p>
MAX	<p>Maximum value between ranks in Application View</p> <p>Maximum value between threads in Rank View</p>
Instructions	<p>Executed number of instructions (number of instruction executions)</p> <p>Number of instruction executions : effective_instruction_counts</p> <p>When the following options are all effective in the fipp command, this value is displayed.</p> <p>-lhwm and -Hevent=Instructions_SIMD</p>
AVE.	<p>Average value between ranks in Application View</p> <p>Average value between threads in Rank View</p>
MIN	<p>Minimum value between ranks in Application View</p> <p>Minimum value between threads in Rank View</p>
MAX	<p>Maximum value between ranks in Application View</p> <p>Maximum value between threads in Rank View</p>
Load/Store (SIMD) (%)	<p>Rate of SIMD and 4SIMD load/store instruction number of instruction executions</p> <p>Number of committed "SIMD and 4SIMD load/store" instructions / Number of instruction executions * 100</p> <p>Number of committed "SIMD and 4SIMD load/store" instructions : XSIMD_load_store_instructions</p> <p>Number of instruction executions : effective_instruction_counts</p> <p>When the following options are all effective in the fipp command, this value is displayed.</p> <p>-lhwm and, -Hevent=Instructions_SIMD or -Hevent=MEM_access</p>
AVE.	<p>Average value between ranks in Application View</p> <p>Average value between threads in Rank View</p>
MIN	<p>Minimum value between ranks in Application View</p> <p>Minimum value between threads in Rank View</p>
MAX	<p>Maximum value between ranks in Application View</p> <p>Maximum value between threads in Rank View</p>
Floating operand (SIMD) (%)	<p>Rate of SIMD and 4SIMD floating-point instruction number of instruction executions</p> <p>Number of committed "SIMD and 4SIMD floating-point" instruction / Number of instruction executions * 100</p> <p>Number of committed "SIMD and 4SIMD floating-point" instruction : XSIMD_floating_instructions</p> <p>Number of instruction executions : effective_instruction_counts</p>

Label	Description
	<p>When the following options are all effective in the fipp command, this value is displayed.</p> <p>-Ihwm and -Hevent=Instructions_SIMD</p>
AVE.	<p>Average value between ranks in Application View</p> <p>Average value between threads in Rank View</p>
MIN	<p>Minimum value between ranks in Application View</p> <p>Minimum value between threads in Rank View</p>
MAX	<p>Maximum value between ranks in Application View</p> <p>Maximum value between threads in Rank View</p>
FMA (SIMD) (%)	<p>Rate of floating-point FMA instructions (SIMD execution) in number of instruction executions</p> <p>Number of committed "4SIMD floating-point multiply and add/sub" and "4SIMD floating-point trigonometric" instructions / Number of instruction execution * 100</p> <p>Number of committed "4SIMD floating-point multiply and add/sub" and "4SIMD floating-point trigonometric" instructions : XSIMD_fma_instructions</p> <p>Number of instruction executions : effective_instruction_counts</p> <p>When the following options are all effective in the fipp command, this value is displayed.</p> <p>-Ihwm and -Hevent=Instructions_SIMD</p>
AVE.	<p>Average value between ranks in Application View</p> <p>Average value between threads in Rank View</p>
MIN	<p>Minimum value between ranks in Application View</p> <p>Minimum value between threads in Rank View</p>
MAX	<p>Maximum value between ranks in Application View</p> <p>Maximum value between threads in Rank View</p>
Fixed operand (SIMD) (%)	<p>Rate of SIMD and 4SIMD fixed point partitioned add/sub and integer multiply add instructions in number of instruction executions</p> <p>Number of committed "SIMD and 4SIMD fixed point partitioned add/sub" and "integer multiply add" instructions / Number of instruction execution * 100</p> <p>Number of committed "SIMD and 4SIMD fixed point partitioned add/sub" and "integer multiply add" instructions : fixed_point_instructions</p> <p>Number of instruction executions : effective_instruction_counts</p> <p>When the following options are all effective in the fipp command, this value is displayed.</p> <p>-Ihwm and -Hevent=Instructions_SIMD</p>
AVE.	<p>Average value between ranks in Application View</p> <p>Average value between threads in Rank View</p>
MIN	<p>Minimum value between ranks in Application View</p> <p>Minimum value between threads in Rank View</p>
MAX	<p>Maximum value between ranks in Application View</p> <p>Maximum value between threads in Rank View</p>
SIMD (%)	<p>Rate of SIMD instructions to the number of total instruction executions</p> <p>(Number of committed "SIMD and 4SIMD load/store" instructions + Number of committed "4SIMD floating-point" instructions + Number of committed "4SIMD floating-point multiply and add/sub" and "4SIMD floating-point trigonometric" instructions) / Number of instruction executions * 100</p>

Label	Description
	<p>Number of committed "SIMD and 4SIMD load/store" instructions : XSIMD_load_store_instructions</p> <p>Number of committed "SIMD and 4SIMD floating-point" instructions : XSIMD_floating_instructions</p> <p>Number of committed "4SIMD floating-point multiply and add/sub" and "4SIMD floating-point trigonometric" instructions : XSIMD_fma_instructions</p> <p>Number of committed "SIMD and 4SIMD fixed point partitioned add/sub" and "integer multiply add" instructions : fixed_point_instructions</p> <p>Number of instruction executions : effective_instruction_counts</p> <p>When the following options are all effective in the fipp command, this value is displayed.</p> <p>-lhwm and -Hevent=Instructions_SIMD</p>
AVE.	<p>Average value between ranks in Application View</p> <p>Average value between threads in Rank View</p>
MIN	<p>Minimum value between ranks in Application View</p> <p>Minimum value between threads in Rank View</p>
MAX	<p>Maximum value between ranks in Application View</p> <p>Maximum value between threads in Rank View</p>
SIMD Load/Store (%)	<p>Rate of SIMD and 4SIMD floating-point loading store instructions that occupies it to the number of floating-point loading store instructions</p> <p>Number of committed "SIMD and 4SIMD load/store" instructions / (Number of committed "SIMD and 4SIMD load/store" instructions + Number of committed "load/store" instructions) * 100</p> <p>Number of committed "SIMD and 4SIMD load/store" instructions : XSIMD_load_store_instructions</p> <p>Number of committed "load/store" instructions : load_store_instructions</p> <p>When the following options are all effective in the fipp command, this value is displayed.</p> <p>-lhwm and -Hevent=Instructions_SIMD</p>
AVE.	<p>Average value between ranks in Application View</p> <p>Average value between threads in Rank View</p>
MIN	<p>Minimum value between ranks in Application View</p> <p>Minimum value between threads in Rank View</p>
MAX	<p>Maximum value between ranks in Application View</p> <p>Maximum value between threads in Rank View</p>
Prefetch	<p>Number of prefetch instructions</p> <p>Number of committed "prefetch" instructions + Number of committed "NonSIMD, SIMD and 4SIMD of indirect prefetch" instructions</p> <p>Number of committed "prefetch" instructions : prefetch_instructions</p> <p>Number of committed "NonSIMD, SIMD and 4SIMD of indirect prefetch" instructions : nonSIMD_XSIMD_indirect_prefetch_instructions</p> <p>When the following options are all effective in the fipp command, this value is displayed.</p> <p>-lhwm and -Hevent=Instructions_SIMD</p>
AVE.	<p>Average value between ranks in Application View</p> <p>Average value between threads in Rank View</p>
MIN	<p>Minimum value between ranks in Application View</p> <p>Minimum value between threads in Rank View</p>

Label		Description
	MAX	Maximum value between ranks in Application View Maximum value between threads in Rank View
Indirect Prefetch		Number of committed "NonSIMD, SIMD and 4SIMD of indirect prefetch" instructions Number of committed "NonSIMD, SIMD and 4SIMD of indirect prefetch" instructions : nonSIMD_XSIMD_indirect_prefetch_instructions When the following options are all effective in the fipp command, this value is displayed. -lhwm and -Hevent=Instructions_SIMD
	AVE.	Average value between ranks in Application View Average value between threads in Rank View
	MIN	Minimum value between ranks in Application View Minimum value between threads in Rank View
	MAX	Maximum value between ranks in Application View Maximum value between threads in Rank View
Load/Store (NOSIMD) (%)		Rate of load/store instruction number of instruction executions (NOSIMD execution) in number of instruction executions Number of committed "load/store" instruction (NOSIMD execution) / Number of instruction executions * 100 Number of committed "load/store" instruction (NOSIMD execution) : load_store_instructions Number of instruction executions : effective_instruction_counts When the following options are all effective in the fipp command, this value is displayed. -lhwm and -Hevent=Instructions_SIMD
	AVE.	Average value between ranks in Application View Average value between threads in Rank View
	MIN	Minimum value between ranks in Application View Minimum value between threads in Rank View
	MAX	Maximum value between ranks in Application View Maximum value between threads in Rank View
Memory access throughput (core) (GB/S)		Average of the amount of data transfer per second between memory and CPU core Zero in Thread View Amount of data transfer between memory and CPU core * 256 / Elapsed time / 1.0e+9 Amount of data transfer between memory and CPU core : L2_miss_dm + L2_miss_pf + L2_wb_dm + L2_wb_pf When the following options are all effective in the fipp command, this value is displayed. -lhwm and -Hevent=MEM_access
	AVE.	Average value between ranks in Application View Average value between threads in Rank View
	MIN	Minimum value between ranks in Application View Minimum value between threads in Rank View
	MAX	Maximum value between ranks in Application View Maximum value between threads in Rank View

Label	Description
Write back (Ratio)	<p>Rate of writeback in memory access</p> <p>Zero in Thread View</p> <p>Memory CPU core writing data amount / Amount of data transfer between memory CPU cores * 100</p> <p>Memory CPU core writing data amount : L2_wb_dm + L2_wb_pf</p> <p>Amount of data transfer between memory CPU cores : L2_miss_dm + L2_miss_pf + L2_wb_dm + L2_wb_pf</p> <p>When the following options are all effective in the fipp command, this value is displayed.</p> <p>-lhwm and -Hevent=MEM_access</p>
AVE.	<p>Average value between ranks in Application View</p> <p>Average value between threads in Rank View</p>
MIN	<p>Minimum value between ranks in Application View</p> <p>Minimum value between threads in Rank View</p>
MAX	<p>Maximum value between ranks in Application View</p> <p>Maximum value between threads in Rank View</p>
Memory access throughput (peak)(core) (GB/S)	<p>Memory access throughput peak value of each core</p> <p>Zero in Thread View</p> <ul style="list-style-type: none"> - When the write backing rate is smaller than 50% $240 + 240 * ((\text{Memory CPU core writing data amount} / \text{Amount of data transfer between memory CPU cores}) / (1 - \text{Memory CPU core writing data amount} / \text{Amount of data transfer between memory CPU cores}))$ - When the write backing rate is 50% or more $240 + 240 * ((1 - \text{Memory CPU core writing data amount} / \text{Amount of data transfer between memory CPU cores}) / (\text{Memory CPU core writing data amount} / \text{Amount of data transfer between memory CPU cores}))$ <p>Memory CPU core writing data amount : L2_wb_dm + L2_wb_pf</p> <p>Amount of data transfer between memory CPU cores : L2_miss_dm + L2_miss_pf + L2_wb_dm + L2_wb_pf</p> <p>When the following options are all effective in the fipp command, this value is displayed.</p> <p>-lhwm and -Hevent=MEM_access</p>
AVE.	<p>Average value between ranks in Application View</p> <p>Average value between threads in Rank View</p>
MIN	<p>Minimum value between ranks in Application View</p> <p>Minimum value between threads in Rank View</p>
MAX	<p>Maximum value between ranks in Application View</p> <p>Maximum value between threads in Rank View</p>
Memory access throughput / PEAK (core) (%)	<p>Rate of memory access throughput peak value in memory access throughput</p> <p>Zero in Thread View</p> <p>Memory access throughput (core) / memory access throughput (peak)(core) * 100</p> <p>When the following options are all effective in the fipp command, this value is displayed.</p> <p>-lhwm and -Hevent=MEM_access</p>
AVE.	<p>Average value between ranks in Application View</p>

Label	Description
	Average value between threads in Rank View
MIN	Minimum value between ranks in Application View Minimum value between threads in Rank View
MAX	Maximum value between ranks in Application View Maximum value between threads in Rank View

Table 2.10 Display items of Procedure/Loop/Line

Label	Description of item	Procedure	Loop	Line
Name	Procedure name	Y	Y	Y
Start	Beginning line number within the range "-" when it is not a user procedure	Y	Y	N
End	End line number within the range "-" when it is not a user procedure	Y	Y	N
Line	Line number	N	N	Y
Operation (S) (Bar Chart)	Graph where deflection to which it is lucky at operation time is shown	Y	Y	Y
Communication (S) (Bar Chart)	Graph where deflection to which it is lucky at communication time is shown	Y	Y	Y
Cost (Bar Chart)	Graph where deflection of cost is shown	Y	Y	Y
Thread Barrier Cost (Bar Chart)	Graph where deflection of cost of synchronous thread barrier waiting is shown	Y	Y	Y
MPI Library Cost (Bar Chart)	Graph where deflection of MPI library cost is shown	Y	Y	Y
Operation (S)	Operation time	Y	Y	Y
Total	Value at operation time as unit The total of the unit broken down by 1 degree	Y	Y	Y
AVE.	The average value of unit broken down by 1 degree	Y	Y	Y
MIN	The minimum value of unit broken down by 1 degree	Y	Y	Y
MAX	The maximum value of unit broken down by 1 degree	Y	Y	Y
Communication (S)	Communication time	Y	Y	Y
Total	Value at communication time as unit The total of the unit broken down by 1 degree	Y	Y	Y
AVE.	The average value of unit broken down by 1 degree	Y	Y	Y
MIN	The minimum value of unit broken down by 1 degree	Y	Y	Y
MAX	The maximum value of unit broken down by 1 degree	Y	Y	Y
Cost	Cost	Y	Y	Y
Total	Value of cost as unit Total of unit broken down by 1 degree	Y	Y	Y

Label	Description of item	Procedure	Loop	Line
AVE.	Average value of unit broken down by 1 degree	Y	Y	Y
MIN	Minimum value of unit broken down by 1 degree	Y	Y	Y
MAX	Maximum value of unit broken down by 1 degree	Y	Y	Y
Thread Barrier Cost	Cost of synchronous thread barrier waiting	Y	Y	Y
Total	Value of synchronous thread barrier waiting as unit Total of unit broken down by 1 degree	Y	Y	Y
AVE.	Average value of unit broken down by 1 degree	Y	Y	Y
MIN	Minimum value of unit broken down by 1 degree	Y	Y	Y
MAX	Maximum value of unit broken down by 1 degree	Y	Y	Y
MPI Library Cost	Cost of MPI library	Y	Y	Y
Total	Value of cost of MPI library as unit Total of unit broken down by 1 degree	Y	Y	Y
AVE.	Average value of unit broken down by 1 degree	Y	Y	Y
MIN	Minimum value of unit broken down by 1 degree	Y	Y	Y
MAX	Maximum value of unit broken down by 1 degree	Y	Y	Y
Nest	Nest level of loop	N	Y	N
Kind	Type of loop (DO, WHILE, UNTIL, ARRAY, FOR, GOTO, OTHER)	N	Y	N
Exec	Execution form of loop (information when compiling) SERIAL : Sequential OpenMP : OpenMP AUTO : Automatic parallel	N	Y	N

Table 2.11 Colors in Bar Chart

Color	Description of color
Blue	Minimum value
Green	Average value
Red	Maximum value

2.3.3.3.2 Topology/Panel information

In the Topology information, the distribution shape to the rank of the cost displayed by summary information can be displayed by the form along the communication topology.

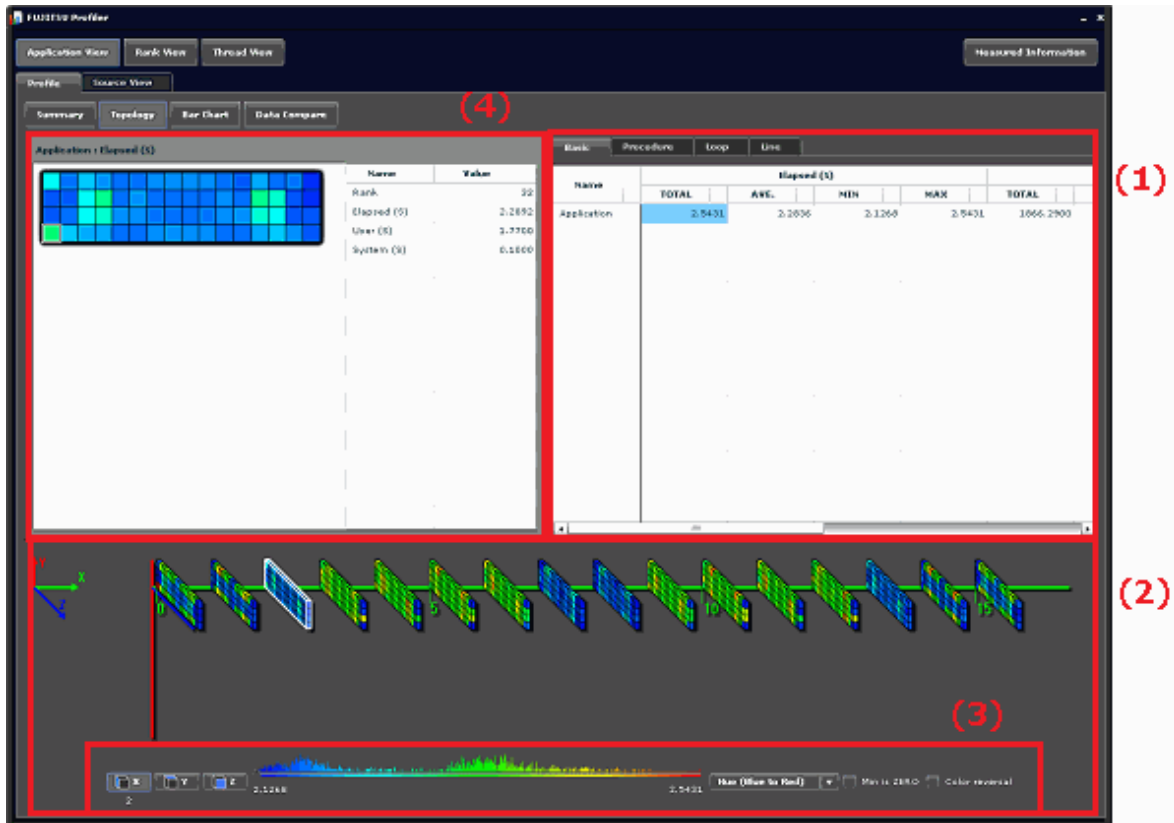
In the Panel information, the distribution shape to the thread of the cost displayed by the summary information can be displayed.

The display form of Panel information becomes one dimension shape and the same of Topology information.

The Topology/Panel information window comprises the following elements:

- (1) Instant Profiler information list
- (2) Whole graph
- (3) Color histogram
- (4) Zoom information panel

Figure 2.7 Topology information window



(1) Instant Profiler information list

Information that excludes the row of the Bar Chart form is displayed from summary information in the Instant Profiler information list by the table form. Refer to "2.3.3.3.1 Summary information" for information on the summary information.

By clicking the cell, distribution situations between parallel about item information on selection row concerning selection line are displayed in the whole graph. Moreover, the selected unit name and item name are displayed in the title of the zoom information panel.

The item of displayed distribution information becomes information as the group row of the selected cell. Even if either TOTAL, AVE., MIN or MAX is selected, it becomes a same deal. Moreover, it is an item that has distribution information that shows the performance. The row of Name, Start, End, Line, Nest, Kind, and Exec does not have distribution information. It is assumed the one having been selected for the row with the distribution information at the right of the selection row when selected. It is assumed the one having been selected for the row with the first distribution information of the line when there is no row with the distribution information right.

In the initial state, the Basic tab is displayed with the distribution information selected in the first row.

(2) Whole graph

Distribution information between parallels on the item selected by the Instant Profiler information list is displayed in the whole graph by the graph form.

A range selection frame is displayed when the cursor is moved to the whole graph, and information that corresponds to the range selection frame is displayed in the zoom information panel.

The range selection frame is a fixed size that can be displayed in the zoom information panel for parallel numbers.

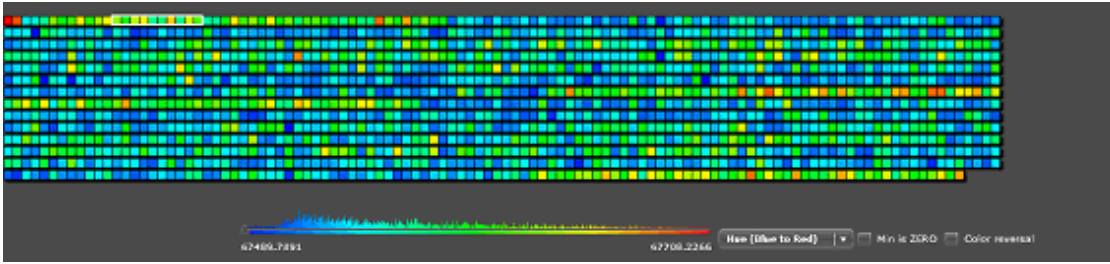
The range selection position can be fixed or released by clicking in the whole graph.

When all parallel numbers are installed on the zoom information panel, the range selection frame is not displayed.

Shape in the selection border for the color and three dimension shape used for the whole graph can be changed by the color histogram.

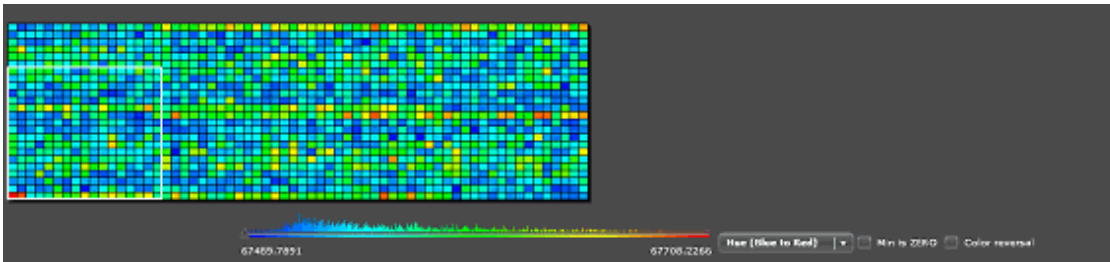
The example of a whole graph is shown as follows. A white frame in figure is a range selection frame.

Figure 2.8 Example of the whole graph with one-dimensional shape/Panel information



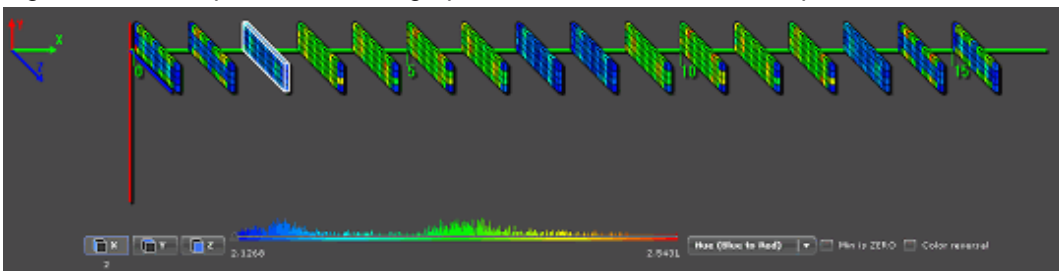
In case of a one-dimensional shape and the Panel information, a parallel unit is arranged from the left to the right. It moves to the next line if it is not possible to display it in one line.

Figure 2.9 Example of the whole graph with two-dimensional shape



In case of a two-dimensional shape, the bottom left is the starting point (0,0).

Figure 2.10 Example of the whole graph with three-dimensional shape



In case of a three-dimensional shape, the intersection of the vertical Y axis (red axis line), the diagonal Z axis (blue axis line), and the horizontal X axis (green axis line) is the starting point.

(3)Color histogram

The color histogram shows the relation between the color and the value used for the whole graph and the graph in the zoom information panel. Note that the occurrence rate of a value is shown according to the height of the color Bar Chart.

The color histogram comprises the following elements:

- a. Range selection frame buttons

The range selection frame button is displayed when displayed by three dimension shape. The range of the selection of the range selection frame is revocable because of the button that has been selected. Moreover, the coordinates location information about the dimension on the axis selected in Whole graph is displayed under the button that has been selected.

Button label	Selected range
X	X axis is fixed. Aspect on the same X axis can be selected.
Y	Y axis is fixed. Aspect on the same Y axis can be selected.
Z	Z axis is fixed. Aspect on the same Z axis can be selected.

- b. Histogram

It displays a histogram. The occurrence rate of a value and the relation between the value and the color is shown.

c. Color mode box

The color mode of the color histogram can be changed. The color modes are described below.

When the Instant Profiler is started, "Hue (Blue to Red)" is selected by default.

Table 2.12 Color mode

Label	Description
Hue (Blue to Red)	The cost is allocated from blue to red in 256 colors. By default, the color becomes red from blue as the cost rises from low-cost.
Tone (Blue to Red)	The cost is allocated blue, white, and red tones. By default, the color becomes blue, white, and red as the cost rises from low-cost.
4 colors	The cost is allocated four colors (blue, green, yellow, and red). By default, the color becomes blue, green, yellow, and red as the cost rises from low-cost. The color allocation threshold can be changed by the operation of the thumb. The default position of thumb is as follows: First thumb: Center of the average value and the minimum value Second thumb: Average value Third thumb: Center of the average value and the maximum value

d. Color histogram controlled check boxes

The minimum value of the histogram and the color order can be changed.

Label	Description
MIN is ZERO	Uses 0 as the minimum value for the color histogram
Color reversal	Reverses the order of the color histogram color scheme

(4)Zoom information panel

In the zoom information panel, information within the range selected in the whole graph is expanded and displayed. A white frame to select one unit is displayed when the cursor is moved to the zoom information panel. The Cost information on the selected unit is displayed in a table in the panel. The displayed item is an item displayed in the Instant Profiler information list. The range selection position can be fixed or released by clicking in the zoom information panel. The unit that has been selected in the Zoom information panel is told to the box where the rank of **Rank View** and **Thread View** is selected and the box and where the thread is selected.

2.3.3.3.3 Bar Chart information

In the Bar Chart information, the distribution between parallel units of the cost displayed in the Cost information is displayed as a Bar Chart. The displayed items and the operation methods are the same as the Topology/Panel information.

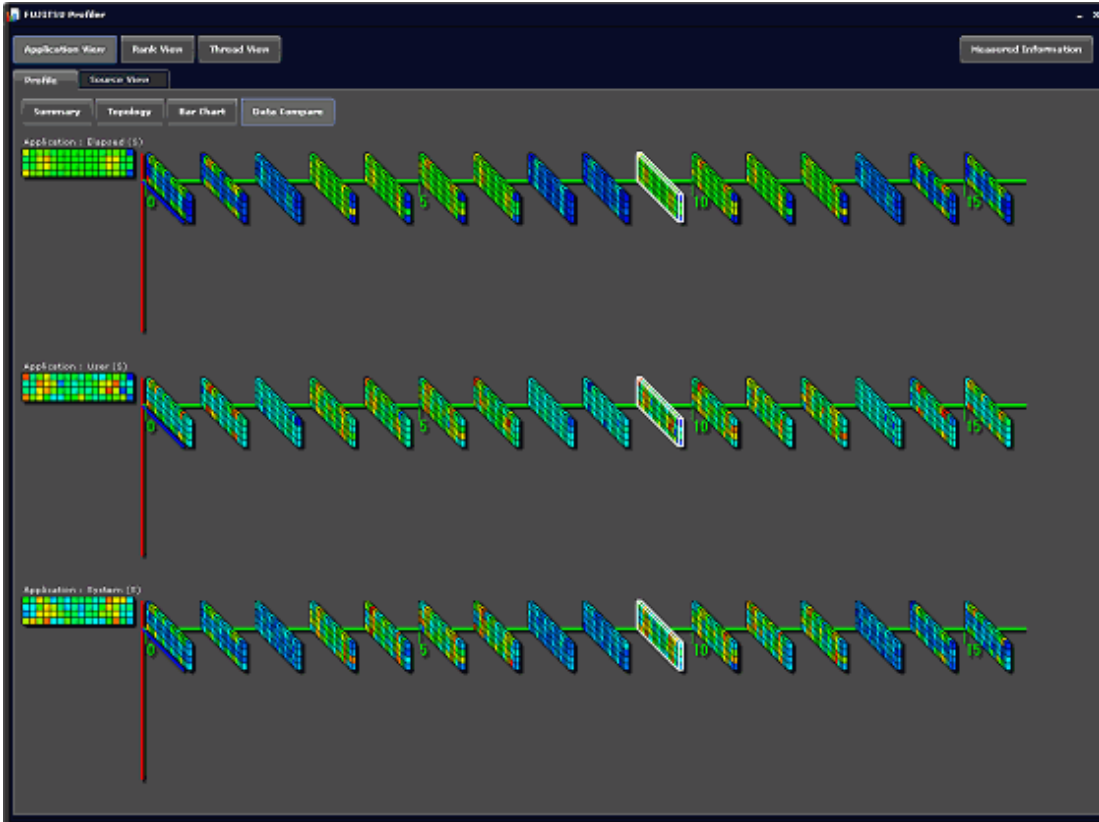
2.3.3.3.4 Data Compare information

In the Data Compare information, the whole graph of three times (for three items) of information displayed in the Topology, Panel and Bar Chart information is arranged vertically and displayed.

The graph of the same item with a different form is updated without being added.

When the cursor is moved to the whole graph, the range selection frame is displayed, and the information that corresponds to the selection frame is extracted to the left and displayed.

Figure 2.11 Data Compare information window



2.3.3.4 Source code information

The information displayed on the **Source View** tab is described below.

There are two windows for the Source code information:

(a) Source list information window

A list of source files with their respective cost is displayed in the source list information window for source files of an application.

Select a source, and then click the **Open** button located at bottom right of the window to display the source list information window, based on information on the **Path** row of the selected row.

Figure 2.12 Source list information window

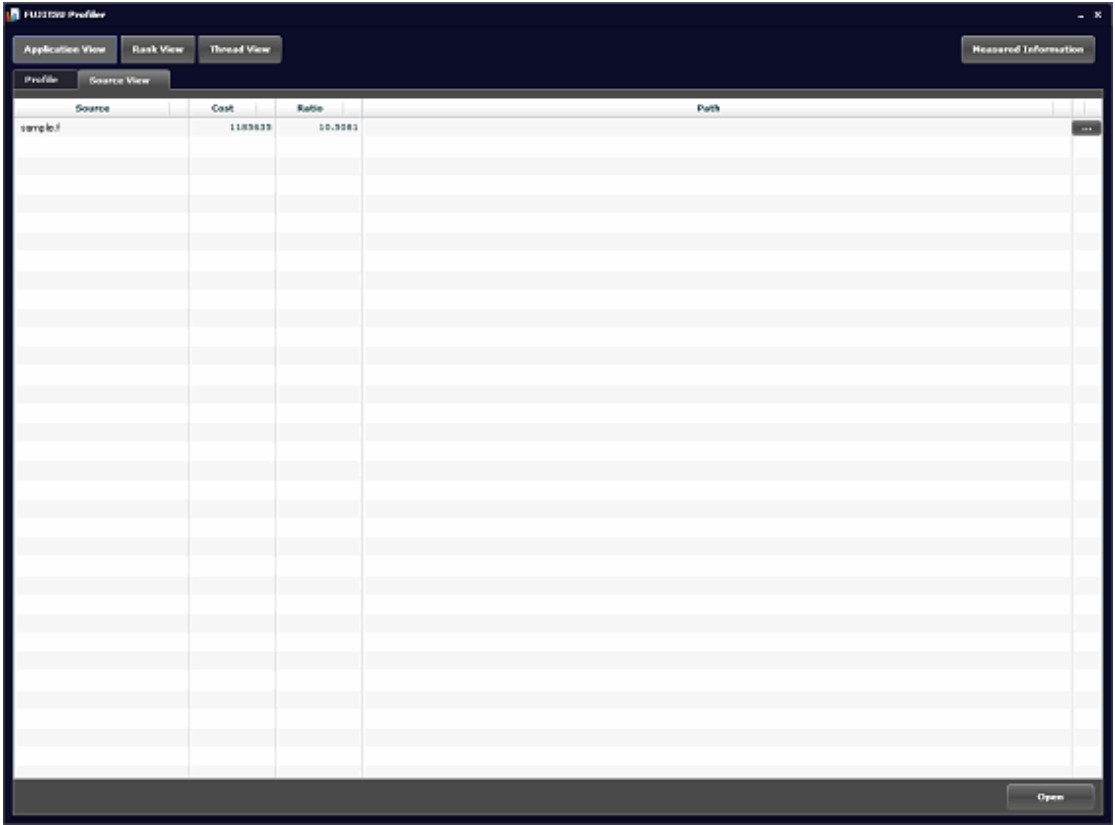


Table 2.13 Items in the source list information window

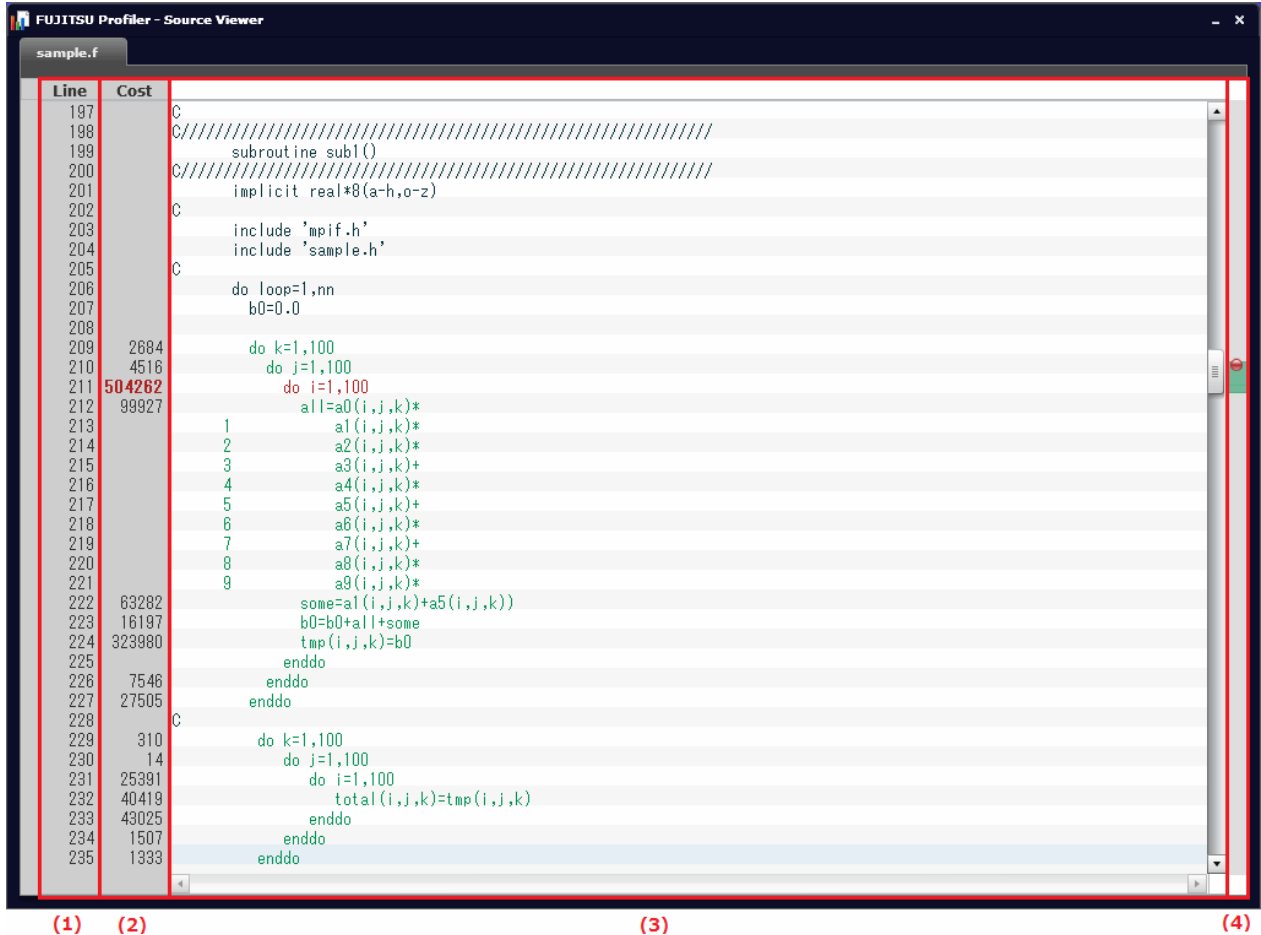
Label	Description
Source	Source file name
Cost	Cost for a source
Ratio	Ratio of the cost of the source to the cost of the entire unit of display
Path	Path of the source file The file selection dialog box appears when the button located on the right is pressed. When a file is selected from the file selection dialog box, the selected file is replaced as the value of Path .

(b)Source code information window

The Source code information window is displayed as a window besides the main window in the Profiler. The Source code information window can be displayed by switching two or more source codes by the window of the tab type in one window. The Cost information to make them correspond to the source code about the source code selected from the source list window on the Source code information window is displayed. The source code cannot be edited from the Source code information window. Each tab of the Source code information window comprises the following elements:

- (1) Line number
- (2) Cost value
- (3) Source code
- (4) Jump map

Figure 2.13 Source code information window



(1)Line number

It is a line number in the source code.

(2)Cost value

It is the cost value that corresponds to each line of the source code. The value is not displayed in the line without the cost. The high load part where the cost exceeds 50% of the entire unit of display is displayed in red.

(3)Source code

It is the source code. The line parallel made a thread when compiling is displayed in green. The high load part where the cost exceeds 50% of the entire unit of display is displayed in red.

(4)Jump map

The part colored green and red in the cost value and the source code areas is mapped and displayed. It corresponds to the scrollbar, and information on the corresponding line can be displayed by scrolling the scrollbar to the area of pigmented material of the map. Moreover, a relevant position can be displayed by clicking on the map.

2.3.3.5 Call Graph information

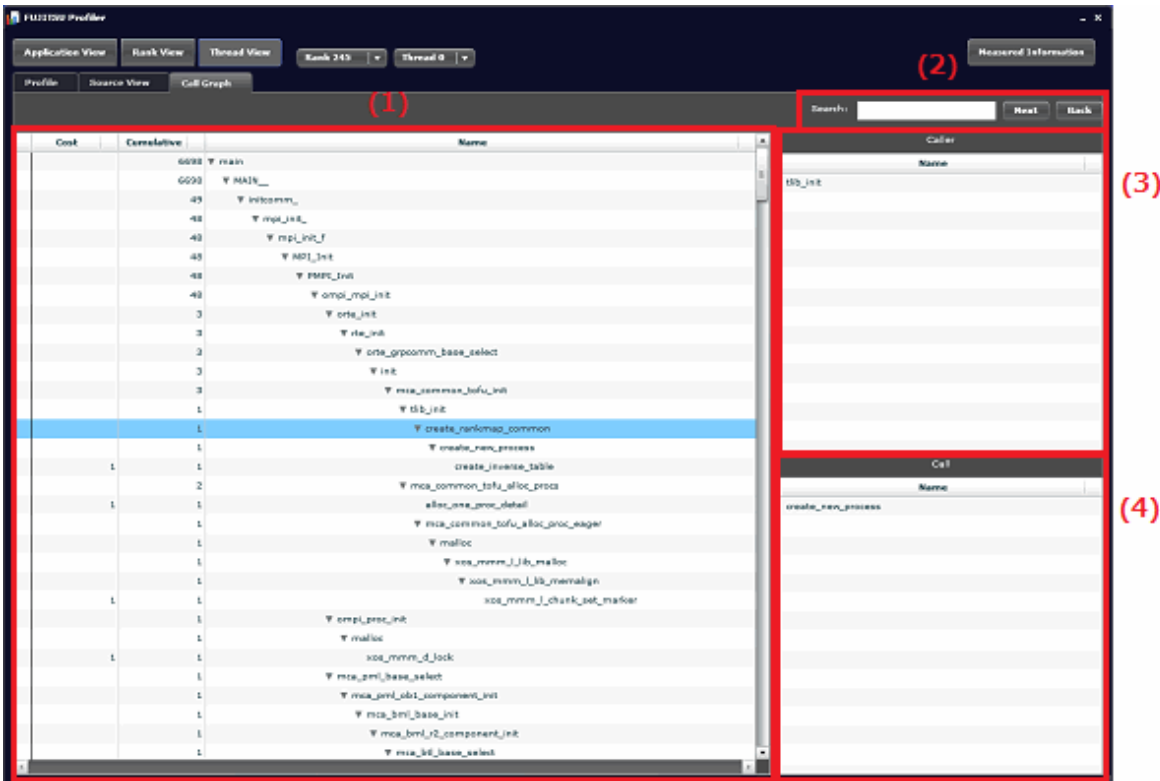
The information displayed on the **Call Graph** tab is described below.

The Call Graph information window comprises the following elements:

- (1) Call Graph
- (2) Search function
- (3) Caller list

(4) Call list

Figure 2.14 Call Graph information window



(1) Call graph

The Call Graph displays information on the call relation of a procedure in the tree format.

The **Caller** and **Call** lists are renewed by centering there when the procedure is clicked from the Call graph.

Table 2.14 Items in the Call Graph information window

Label	Description
-	A red circle is displayed around the high load part where the call cost exceeds 50 % of the entire unit of display.
Cost	Cost value that hangs to the procedure on call relation
Cumulative	Accumulation value of the cost that hangs to the procedure on call relation and route from now on
Name	Procedure name

(2) Search function

The retrieval function circulating retrieves the input function name. It goes for the retrieval toward from the selection position or the head to the under when the **Next** button is clicked. It lives in the retrieval toward from the selection position or the end on when the **Back** button is clicked. The retrieval key should be completely corresponding.

(3) Caller list

The procedure name in which the procedure that has been selected is called is having a look displayed by the Call graph. Not only parents with the node that has been selected but also the call origins of the same procedure that exists in another route are displayed. It changes into the display centering on the corresponding section in the Call graph when the procedure name displayed in the **Caller** list is selected.

(4)Call list

The procedure name that the procedure that has been selected calls is having a look displayed by the Call graph. Not only the child with the node that has been selected but also the procedure that the same procedure that existed in another route called is displayed. It changes into the display centering on the corresponding section in the Call graph when the procedure name displayed in the **Call** list is selected.

2.4 Instant Profiler information (text/CSV formats)

This chapter describes the Instant Profiler information (text/CSV formats) output by the fipppx command.

2.4.1 Overview of the Profiler feature

The Instant Profiler information comprises the following information:

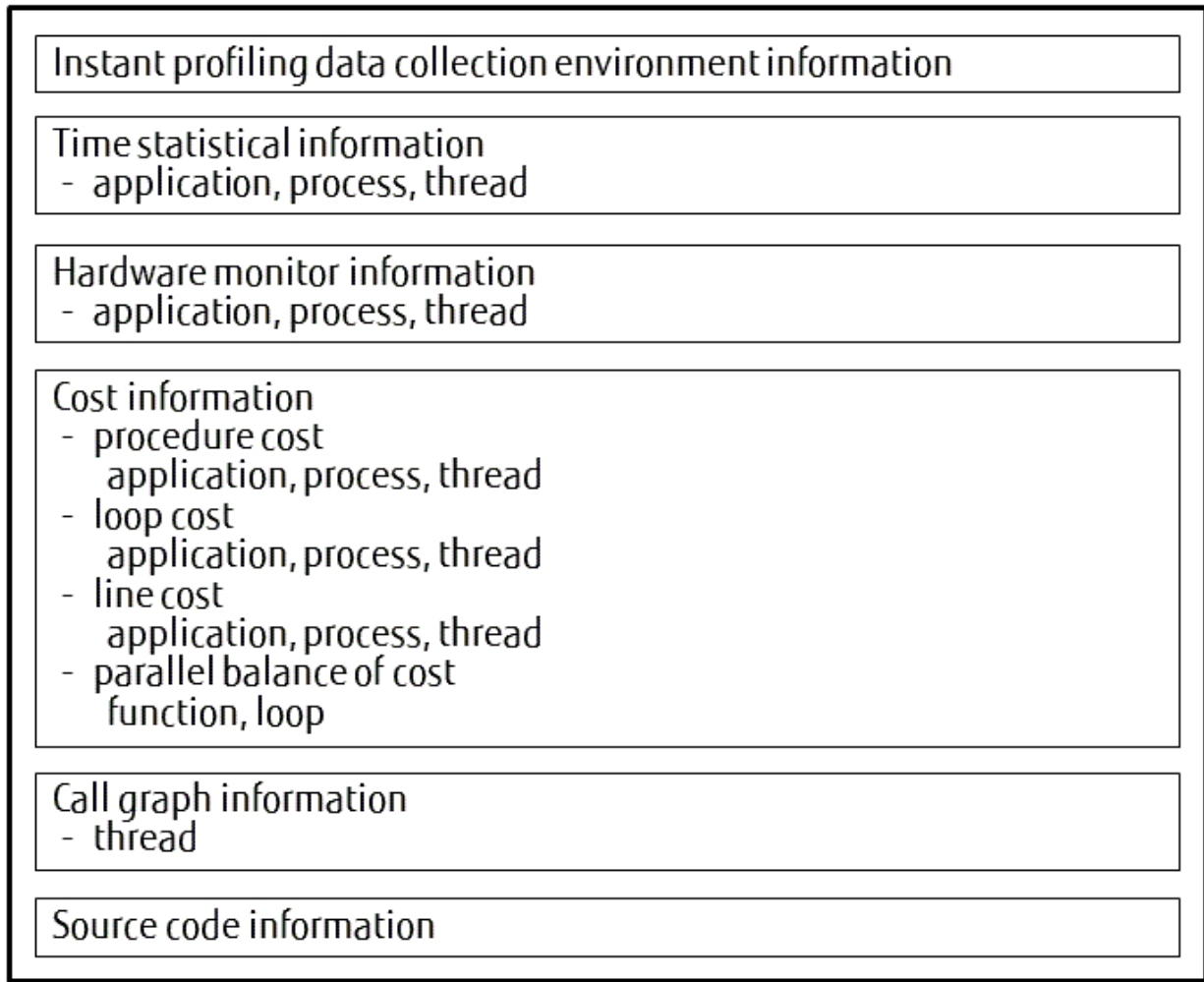
The output of each information can be controlled by using options of the fipppx command.

Refer to "[Table 2.2 fipppx command options](#)" for information on options that control information.

- Environment information for instant profiling data collection
- Time statistical information
- Hardware monitor information
- Cost information
- Call Graph information
- Source code information

The following figure shows the composition of the Instant Profiler information in the text format.

Figure 2.15 Composition of the Instant Profiler information in the text format



2.4.2 Environment information for instant profiling data collection

The environment information outputs the details of the execution environment where the instant profiling data is collected.

Figure 2.16 Output format of the environment information for instant profiling data collection

```

-----
Fujitsu Instant Performance Profiler Version @vl
Measured time           : @date
CPU frequency           : @pno @frequency (MHz)
Type of program         : @type
Average at sampling interval : @interval (ms)
Measured range          : @range
Virtual coordinate      : (@x, @y, @z)
-----

```

Table 2.15 Output items of the environment information for instant profiling data collection

Output item	Description
@vl	Version level
@date	Collection date of instant profiling data
@pno	Process number
@frequency	Frequency of execution processor

Output item	Description
@type	Execution format of the application SERIAL : Sequential Thread (AUTO) : Automatic parallel Thread (OpenMP) : OpenMP MPI : MPI MPI (AUTO) : MPI + Automatic parallel MPI (OpenMP) : MPI + OpenMP
@interval	Sampling interval The average value of the sampling interval is output.
@range	Measured range All ranges : Entire application selected range : Specified range of the section specifying function of Instant Profiler
@x, @y, @z	Logical shape of an MPI application execution (displayed in case of MPIs)

2.4.3 Time statistical information

The elapsed time, the user CPU time, and the system CPU time of an application, each process, and each thread is output in the Time statistical information.

Figure 2.17 Output format of the Time statistical information

```

Time statistics
-----
Elapsed(s)   User(s)   System(s)
-----
@elapse     @user     @system   @level
-----
@elapse     @user     @system   @level   @pno

```

Table 2.16 Output items of the Time statistical information

Output item	Description
@elapse	Elapsed time (s)
@user	User CPU time (s)
@system	System CPU time (s)
@level	Information total level (Application, process number, or thread number)
@pno	Process number or thread number

2.4.4 Hardware monitor information

The Hardware monitor information outputs the operating condition of the processor during application execution.

This information is output when the "-lhwm" option is used.

Applications written in Fortran, C, or C++ are converted to hardware instructions by the compiler and are executed by the processor.

Some functions to execute the instruction at fast are prepared in the processor.

By using each speed-up function, the execution CPU time of the application can be decreased.

The Hardware monitor information shows the measured value of the operating condition of the processor during application execution.

Each measurement of the Hardware monitor information is aggregated by the following unit and is output:

- Application
- Process
- Thread

2.4.4.1 Measured information of the Hardware monitor information

The following table describes the measured information of the Hardware monitor information.

Table 2.17 Measured information of the Hardware monitor information

Event name	Measured information	Description
Instructions_SIMD	Elapsed time (s)	Elapsed time for executing instructions of the measured range
	Instructions	Executed number of instructions
	MIPS	Instruction execution efficiency (average of number of instructions executed per second)
	MIPS/PEAK (%)	Rate of actual measurement values to the logical peak value of MIPS
	Load/Store (SIMD) (%) (*1)	Rate of floating-point loading store SIMD or SIMD instruction that occupies it to the number of instruction executions
	Floating operand (SIMD) (%) (*1)	Rate of floating-point operation instructions (SIMD executions) that occupies it to the number of instruction executions
	FMA (SIMD) (%) (*1)	Rate of floating-point FMA instructions (SIMD execution) in number of instruction executions
	Fixed operand (SIMD) (%) (*1)	Rate of fixed-point operation instructions (SIMD executions) that occupies it to the number of instruction executions
	SIMD (%) (*1)	Rate of number of SIMD instructions in number of instruction executions
	SIMD load store instruction rate (%)	Rate of SIMD instruction that occupies it to load store instruction
	Prefetch	Number of prefetch instructions
	Indirect prefetch	Number of indirect prefetch instructions
MEM_access	Elapsed time (s)	Elapsed time for executing instructions of the measured range
	Load/Store (SIMD) (%) (*1)	Rate of floating-point loading store SIMD or SIMD instructions that occupies it to the number of instruction executions
	Load/Store (NOSIMD) (%) (*1)	Rate of number of floating-point loading store instructions in number of instruction executions
	Memory access throughput (core) (GB/S)	Average data transfer between memory and CPU core per second
	Write back rate (%) (*2)	Rate of writing data amount to the data transfer between memory CPU cores
	Memory access throughput (PEAK) (*2)	Peak value of memory throughput calculated based on write backing rate

Event name	Measured information	Description
	Memory access throughput peak rate (%) (*2)	Rate of survey data to memory throughput peak value
Statistics	Elapsed time (s)	Elapsed time for executing instructions of the measured range
	MFLOPS	Floating-point arithmetic execution efficiency (average floating-point arithmetic executions per second)
	MFLOPS/PEAK (%)	Rate of actual measurement values to the logical peak value of MFLOPS
	MIPS	Instruction execution efficiency(average of number of instruction executions per second)
	MIPS/PEAK (%)	Rate of actual measurement values to the logical peak value of MIPS
	Floating operand (%)	Rate of number of floating-point arithmetic execution in operand of instruction

*1 : SIMD instruction processes two or more operands in one instruction. NOSIMD instruction processes one operand in one instruction.

*2 : "Write back rate", "Memory access throughput (PEAK)", and "Memory access throughput peak rate" output only the unit of the application, and the power output of the unit of the process and each thread becomes 0.

2.4.4.2 Formulas of the Hardware monitor information

The following table describes the calculation formulas for events in the Hardware monitor information.

Table 2.18 Formulas of the Hardware monitor information (Instructions_SIMD)

Event name	Measured information (unit)	Formula
Instructions_SIMD	Elapsed time (s)	Elapsed time for executing instructions of the measured range
	Instructions	Executed number of instructions
	MIPS	Number of instruction executions / Elapsed time / 1.0e+6
	MIPS/PEAK (%) (*1)	MIPS / (Number of execution cores * MIPS peak value of each core) * 100
	Load/Store(SIMD) (%)	Number of floating-point loading store SIMD instructions (SIMD and 4SIMD) / Number of instruction executions * 100
	Floating operand(SIMD) (%)	Number of floating-point arithmetic SIMD instructions (SIMD and 4SIMD) / Number of instruction executions * 100
	FMA(SIMD) (%)	Number of floating-point FMA SIMD instructions (SIMD and 4SIMD) / Number of instruction executions * 100
	Fixed operand(SIMD) (%)	Number of fixed-point arithmetic SIMD instructions (SIMD and 4SIMD) / Number of instruction executions * 100
	SIMD (%)	(Number of floating-point loading store SIMD instructions (SIMD and 4SIMD) + Number of floating-point arithmetic SIMD instructions (SIMD and 4SIMD) + Number of floating-point FMA SIMD instructions (SIMD and 4SIMD) + Number of fixed-point arithmetic SIMD instructions (SIMD and 4SIMD)) / Number of instruction executions * 100
	SIMD load store instruction rate (%)	Number of floating-point loading SIMD store instructions (SIMD and 4SIMD) / (Number of floating-point loading store

Event name	Measured information (unit)	Formula
		instructions + Number of floating-point loading store SIMD instructions (SIMD and 4SIMD) * 100
	Prefetch	Number of prefetch instructions (exclude "indirect prefetch" instruction) + Number of indirect prefetch instructions
	Indirect prefetch	Number of indirect prefetch instructions

*1 : The MIPS peak value of each core is calculated by the following expression:

$\text{Frequency in CPU core} * \text{Number of instructions that can be issued per cycle} * 1000(\text{MIPS})$

The values in the calculation expressions are used from the following PA events.

- | | |
|--|-----------------------------------|
| (1) effective_instruction_counts | (2) XSIMD_load_store_instructions |
| (3) XSIMD_floating_instructions | (4) XSIMD_fma_instructions |
| (5) XSIMD_fixed_point_instructions | (6) prefetch_instructions |
| (7) nonSIMD_XSIMD_indirect_prefetch_instructions | (8) load_store_instructions |

- | | |
|---|-----|
| Executed number of instructions | (1) |
| Number of floating-point loading store SIMD instructions (SIMD and 4SIMD) | (2) |
| Number of floating-point arithmetic SIMD instructions (SIMD and 4SIMD) | (3) |
| Number of floating-point FMA SIMD instructions (SIMD and 4SIMD) | (4) |
| Number of fixed-point arithmetic SIMD instructions (SIMD and 4SIMD) | (5) |
| Number of floating-point loading store instructions | (8) |
| Number of prefetch instructions (exclude "indirect prefetch" instruction) | (6) |
| Number of indirect prefetch instructions | (7) |

Table 2.19 Formulas of the Hardware monitor information (MEM_access)

Event name	Measured information (unit)	Formula
MEM_access	Elapsed time (s)	Elapsed time for executing instructions of the measured range
	Load/Store(SIMD) (%)	Number of floating-point loading store SIMD instructions (SIMD and 4SIMD) / Number of instruction executions * 100
	Load/Store(NOSIMD) (%)	Number of floating-point loading store instructions / Number of instruction executions * 100
	Memory access throughput (core) (GB/S)	Amount of data transfer between memory and CPU core * 256 / Elapsed time / 1.0e+9
	Write back rate (%)	Memory CPU core writing data amount / Amount of data transfer between memory CPU cores * 100
	Memory access throughput (PEAK)	<ul style="list-style-type: none"> - When the write backing rate is smaller than 50% $240 + 240 * ((\text{Memory CPU core writing data amount} / \text{Amount of data transfer between memory CPU cores}) / (1 - \text{Memory CPU core writing data amount} / \text{Amount of data transfer between memory CPU cores}))$ - When the write backing rate is 50% or more $240 + 240 * ((1 - \text{Memory CPU core writing data amount} / \text{Amount of data transfer between memory CPU cores}) / (\text{Memory CPU core writing data amount} / \text{Amount of data transfer between memory CPU cores}))$

Event name	Measured information (unit)	Formula
	Memory access throughput peak rate (%)	Memory access throughput / Memory access throughput (PEAK) * 100

The values in the calculation expressions are used from the following PA events.

- | | |
|----------------------------------|-----------------------------------|
| (1) effective_instruction_counts | (2) XSIMD_load_store_instructions |
| (3) load_store_instructions | (4) L2_miss_dm |
| (5) L2_miss_pf | (6) L2_wb_dm |
| (7) L2_wb_pf | |

- | | |
|---|-----------------------|
| Number of instruction executions | (1) |
| Number of floating-point loading store SIMD instructions (SIMD and 4SIMD) | (2) |
| Number of floating-point loading store instructions | (3) |
| Amount of data transfer between memory CPU cores | (4) + (5) + (6) + (7) |
| Memory CPU core writing data amount | (6) + (7) |

Table 2.20 Formulas of the Hardware monitor information (Statistics)

Event name	Measured information (unit)	Formula
Statistics	Elapsed time (s)	Elapsed time for executing instructions of the measured range
	MFLOPS	Number of floating-point arithmetic instructions / Elapsed time / 1.0e+6
	MFLOPS/PEAK (%) (*1)	MFLOPS / (Number of execution cores * MFLOPS peak value of each core) * 100
	MIPS	Number of instruction executions / Elapsed time / 1.0e+6
	MIPS/PEAK (%) (*2)	MIPS / (Number of execution cores * MIPS peak value of each core) * 100
	Floating operand (%)	Number of floating-point arithmetic instructions / Total operand * 100

*1 : The MFLOPS peak value of each core is calculated by the following expression:

Frequency in CPU core * Number of product sum operation of floating-point * Number of floating-point functional units * Number of operand processing of SIMD instructions * 1000(MFLOPS)
--

*2 : The MIPS peak value of each core is calculated by the following expression:

Frequency in CPU core * Number of instructions that can be issued per cycle * 1000(MIPS)
--

The values in the calculation expressions are used from the following PA events.

- | | |
|--|--------------------------------------|
| (1) effective_instruction_counts | (2) 1FLOPS_instructions |
| (3) 2FLOPS_instructions | (4) 4FLOPS_instructions |
| (5) 8FLOPS_instructions | (6) 16FLOPS_instructions |
| Number of instruction executions | (1) |
| Number of floating-point arithmetic instructions | (2) + 2*(3) + 4*(4) + 8*(5) + 16*(6) |
| Total operand | (1) + (3) + 3*(4) + 7*(5) + 15*(6) |

2.4.4.3 Using the Hardware monitor information

The Hardware monitor information is used to confirm the execution performance of a program.

As MIPS and MFLOPS values are close to each peak value, it is a program of the high execution and operation performances.

2.4.4.4 Output format of the Hardware monitor information

The following shows the output format of the Hardware monitor information.

Figure 2.18 Output format of the Hardware monitor information (Instructions_SIMD)

```

Performance monitor event:Instruction_SIMD
*****
@level - performance monitor
*****

-----
Elapsed(s)      Inst      MIPS  MIPS/PEAK(%)
-----
@elapsed        @inst      @mips  @mips/peak  @level
-----
@elapsed        @inst      @mips  @mips/peak  @level  @pno

Elapsed(s)  LS_SIMD-or(%)  Float_SIMD-or(%)  Fma_SIMD-or(%)  Fixed_SIMD-or(%)
-----
@elapsed    @ls_simd      @float_simd      @fma_simd      @fixed_simd  @level
-----
@elapsed    @ls_simd      @float_simd      @fma_simd      @fixed_simd  @level  @pno

Elapsed(s)      SIMD(%)  LS_SIMD_rate(%)  Prefetch  Indirect_pf
-----
@elapsed        @simd    @ls_simd_rate    @prefetch    @indirect_pf  @level
-----
@elapsed        @simd    @ls_simd_rate    @prefetch    @indirect_pf  @level  @pno

```

Table 2.21 Output items of the Hardware monitor information (Instructions_SIMD)

Output item	Description
@elapsed	Elapsed time (s)
@inst	Number of instruction executions
@mips	MIPS
@mips/peak	MIPS peak performance rate (%)
@ls_simd	Loading store instruction rate (SIMD) (%)
@float_simd	Floating-point arithmetic instruction rate (SIMD) (%)
@fma_simd	Floating-point FMA instruction rate (SIMD) (%)
@fixed_simd	Fixed-point arithmetic instruction rate (SIMD) (%)
@simd	SIMD instruction rate (%)
@ls_simd_rate	SIMD loading store instruction rate (%)
@prefetch	Number of prefetch instructions
@indirect_pf	Number of indirect prefetch instructions
@name	Group name
@no	Advance number
@pno	Process number
@thno	Thread number

Figure 2.19 Output format of the Hardware monitor information (MEM_access)

```

Performance monitor event:MEM_access

*****
@level - performance monitor
*****

Elapsed(s)      LS_SIMD-or(%)  LS-or(%)      Mem throughput
                -----
                @ls_simd    @ls           @mem @level
                -----
                @elapsed    @ls_simd    @ls           @mem @level @pno

Elapsed(s)      mem_write(%)   Mem throughput  Mem throughput
                -----
                @mem_write  @mem_tp_peak  @mem_tp/peak  @level
                -----
                @elapsed    @mem_write  @mem_tp_peak  @mem_tp/peak @level @pno
    
```

Table 2.22 Output format of the Hardware monitor information (MEM_access)

Output item	Description
@elapsed	Elapsed time (s)
@ls_simd	Loading store instruction rate (SIMD) (%)
@ls	Loading store instruction rate (NOSIMD) (%)
@mem	Memory access throughput (core) (GB/S)
@mem_write	Write back rate (%)
@mem_tp_peak	Memory access throughput (PEAK)
@mem_tp/peak	Memory access throughput / PEAK (%)
@name	Group name
@no	Advance number
@pno	Process number
@thno	Thread number

Figure 2.20 Output items of the Hardware monitor information (Statistics)

```

Performance monitor event:Statistics

*****
@level - performance monitor
*****

Elapsed(s)      MFLOPS  MFLOPS/PEAK(%)
                -----
                @mflops  @mflops/peak  @level
                -----
                @elapsed  @mflops  @mflops/peak  @level @pno

Elapsed(s)      MIPS  MIPS/PEAK(%)  Floating-op(%)
                -----
                @mips  @mips/peak  @fl-op  @level
                -----
                @elapsed  @mips  @mips/peak  @fl-op  @level @pno
    
```

Table 2.23 Output format of the Hardware monitor information (Statistics)

Output item	Description
@elapsed	Elapsed time (s)
@mflops	MFLOPS
@mflops/peak	MFLOPS peak performance rate (%)
@mips	MIPS
@mips/peak	MIPS peak performance rate (%)
@fl-op	Floating-point arithmetic rate (%)
@name	Group name
@no	Advance number
@pno	Process number
@thno	Thread number

2.4.5 Cost information

The Cost information comprises the following information:

- Procedure cost distribution information
- Loop cost distribution information
- Line cost distribution information

In case of parallel applications, the cost balance information can be output as the procedure cost distribution information and the loop cost distribution information.

Procedure cost distribution information

In the procedure cost distribution information, the procedure cost and the synchronous waiting cost between threads and the start and end line number of a procedure and the procedure name are output for each application, process, or thread.

This information is output when the "-Icpu" option is used.

However, the cost balance information in the procedure cost distribution information will be output when the "-Icpu" and "-Ibalance" options are used.

The following figure shows the output format of the procedure cost distribution information, while the following table describes the items output in the procedure cost distribution information.

Figure 2.21 Output format of the procedure cost distribution information

```

-----
Procedures profile
*****
@level - procedures
*****

      Cost      %   Operation (s)   Barrier      %   Start   End
-----
      @cost  @cost-rate      @ope  @barrier @barrier-rate      --   --   @level
-----
      @cost  @cost-rate      @ope  @barrier @barrier-rate      @start @end  @name
-----
      MPI      % Communication (s)   Start   End
-----
      @mpi  @mpi-rate      @comm      --   --   @level
-----
      @mpi  @mpi-rate      @comm      @start @end  @name

-- Parallel balance of cost --
@name @start - @end
|-----|
|-----| @balance @cost @pno (*1)
|-----|
|-----|

```

*1 : The cost balance information is output when the "-Ibalance" option is used.

Table 2.24 Output items of the procedure cost distribution information

Output item	Meaning of output item
@level	Information total level (Application, process number, or thread number)
@cost	Procedure cost
@cost-rate	Rate of the cost of @level or @name to the cost of information of the total level (%)
@ope	Operation time (s)
@barrier	Synchronous wait cost between threads (*1)
@barrier-rate	Rate of synchronous waiting cost between threads to the procedure cost (%) (*1)
@mpi	MPI cost (*2)
@mpi-rate	Rate of MPI cost to the procedure cost (%) (*2)
@comm	Communication time (s) (*2)
@start	Start line number of a procedure
@end	End line number of a procedure
@name	Procedure name
@balance	Cost balance of a procedure between processes or threads (%) (*3)
@pno	Process number or thread number

*1 : Output when a thread-parallel application is measured

*2 : Output when an MPI application is measured

*3 : Output when the "-Ibalance" option is used

For a thread-parallel application, the parallel part is output as information on the created procedure.

The identifier is added to the created procedure after the procedure name.

The following table provides the procedure names of thread-parallel applications.

Table 2.25 Procedure names of thread-parallel applications

Language	Type of created procedure	Created procedure name
Fortran	Automatic-parallel procedure	<i>Procedure._PRL_Number_</i>
	OpenMP-parallel procedure	<i>Procedure._OMP_Number_</i>
	TASK syntax	<i>Procedure._TSK_Number_</i>
C/C++	Automatic-parallel procedure	<i>Procedure._PRL_Number</i>
	OpenMP-parallel procedure	<i>Procedure._OMP_Number</i>
	TASK syntax	<i>Procedure._TSK_Number</i>

Loop cost distribution information

In the loop cost distribution information, the loop cost and the synchronous waiting cost between threads and the nest level and the loop type and the compilation type of loop and the start/end line number of loop and the procedure name that the loop belongs are output for each application, process, or thread.

This information is output when the "-Icpu" option is used.

However, the cost balance information in the loop cost distribution information will be output when the "-Icpu" and "-Ibalance" options are used.

The following figure shows the output format of the loop cost distribution information, while the following table describes the items output in the loop cost distribution information.

Figure 2.22 Output format of the loop cost distribution information

```

-----
Loops profile
*****
@level - loops
*****
-----
Cost      % Operation (S)  Barrier      % Nest  Kind  Exec  Start  End
-----
@cost @cost-rate      @ope  @barrier @barrier-rate      --  --  @level
-----
@cost @cost-rate      @ope  @barrier @barrier-rate  @nest @kind @exec @start @end @name
-----
MPL      % Communication (S)  Nest  Kind  Exec  Start  End
-----
@mpi @mpi-rate      @comm      --  --  @level
-----
@mpi @mpi-rate      @comm      @nest @kind @exec @start @end @name
-----

-- Parallel balance of cost --
@name @start - @end
+-----+
|           |           | @balance @cost @pno (*1)
+-----+

```

*1 : The cost balance information is output when the "-Ibalance" option is used.

Table 2.26 Output items of the loop cost distribution information

Output item	Description
@level	Information total level (Application, process number, or thread number)
@cost	Loop cost
@cost-rate	Rate of the cost of @level or @name to the cost of information of the total level (%)
@ope	Operation time (s)
@barrier	Synchronous waiting cost between threads (*1)

Output item	Description
@barrier-rate	Rate of synchronous waiting cost between threads to the loop cost (%) (*1)
@mpi	MPI cost (*2)
@mpi-rate	Rate of MPI cost to the loop cost (%) (*2)
@comm	Communication time (s) (*2)
@nest	Nested level
@kind	Loop type (DO, WHILE, UNTIL, ARRAY, FOR, GOTO, OTHER)
@exec	Compilation type of loop (SERIAL : Sequential, OpenMP : OpenMP, AUTO : Automatic parallel)
@start	Start line number of a loop
@end	End line number of a loop
@name	Procedure name
@balance	Cost balance of loop between processes or threads (%) (*3)
@pno	Process number or thread number

*1 : Output when a thread-parallel application is measured

*2 : Output when an MPI application is measured

*3 : Output when the "-Ibalance" option is used

Line cost distribution information

In the line cost distribution information, the line cost and the line number, including the procedure name that the line belongs to are output for each application, process, or thread.

This information is output when the "-Icpu" option is used.

The following figure shows the output format of the line cost distribution information, while the following table describes the items output in the line cost distribution information.

Figure 2.23 Output format of the line cost distribution information

```

Lines profile
*****
@level - lines
*****

-----
Cost          %   Operation (S)  Barrier          %   Line
-----
@cost @cost-rate          @ope  @barrier @barrier-rate  -- @level
-----
@cost @cost-rate          @ope  @barrier @barrier-rate  @line @name

MPI          % Communication (S)  Line
-----
@mpi @mpi-rate          @comm  -- @level
-----
@mpi @mpi-rate          @comm  @line @name

```

Table 2.27 Output items of the line cost distribution information

Output item	Description
@level	Information total level (Application, process number, or thread number)
@cost	Line cost
@cost-rate	Rate of the cost of @level or @name to the cost of information of the total level (%)
@ope	Operation time (s)

Output item	Description
@barrier	Synchronous waiting cost between threads (*1)
@barrier-rate	Rate of synchronous waiting cost between threads to the line cost (%) (*1)
@mpi	MPI cost (*2)
@mpi-rate	Rate of MPI cost to the line cost (%) (*2)
@comm	Communication time (s) (*2)
@line	Line number
@name	Procedure name

*1 : Output when a thread-parallel application is measured

*2 : Output when an MPI application is measured

2.4.6 Call Graph information

In the Call Graph information, the call route of a procedure and the cost of each call route is output.

This information is output when the "-lcall" option is used.

The following figure shows the output format of the Call Graph information, while the following table describes the items output in the Call Graph information.

Figure 2.24 Output format of the Call Graph information

```

Call graph
  Process      @pno - Thread      @thno
  -----+-----
          | @rate % <@nest> @name [ @cost / @accumulation ]

```

Table 2.28 Output items of the Call Graph information

Output item	Description
@pno	Process number
@thno	Thread number
@rate	Rate of the procedure cost to the cost of the whole thread (%)
@nest	Nested levels of procedure call
@name	Procedure name
@cost	Procedure cost
@accumulation	Procedure cost, including the cost of the called procedure

If any interruption by sampling of the Instant Profiler occurs during the execution of the input or output statement of the application, the Call Graph information may not be output correctly.

If "<???" is output for the nested level of the Call Graph information, it implies either of the following:

- The call route of a procedure is uncertain.
- The nested levels of the calling procedure are 128 or more.

2.4.7 Source code information

In the Source code information, the cost is added and output in each line of the source code.

This information is output when the "-lsrc" option is used.

The following figure shows the output format of the Source code information, while the following table describes the items output in the Source code information.

Figure 2.25 Output format of the Source code information

```

Sources profile
-----> @file-name
-----
   Line      Costs
-----
  @line      @cost  @source-code

```

Table 2.29 Output items of the Source code information

Output item	Description
@file-name	Source code file name
@line	Line number
@cost	Line cost
@source-code	Source code

Chapter 3 Advanced Profiler

This chapter describes the features and usage of the Advanced Profiler.

3.1 Overview of the Advanced Profiler

The Advanced Profiler collects and outputs the execution performance information for a specific section of an application. The Advanced Profiler can output the following information:

Basic information

Outputs the breakdown of the call count, the elapsed time, the user CPU time, and the system CPU time of the measurement section

MPI information

Outputs the MPI library information of the measurement section

Hardware monitor information

Outputs the Hardware monitor information of the measurement section

Largepage performance information

Outputs the Largepage performance information of the measurement section

The following commands must be used to use the Advanced Profiler.

fapp

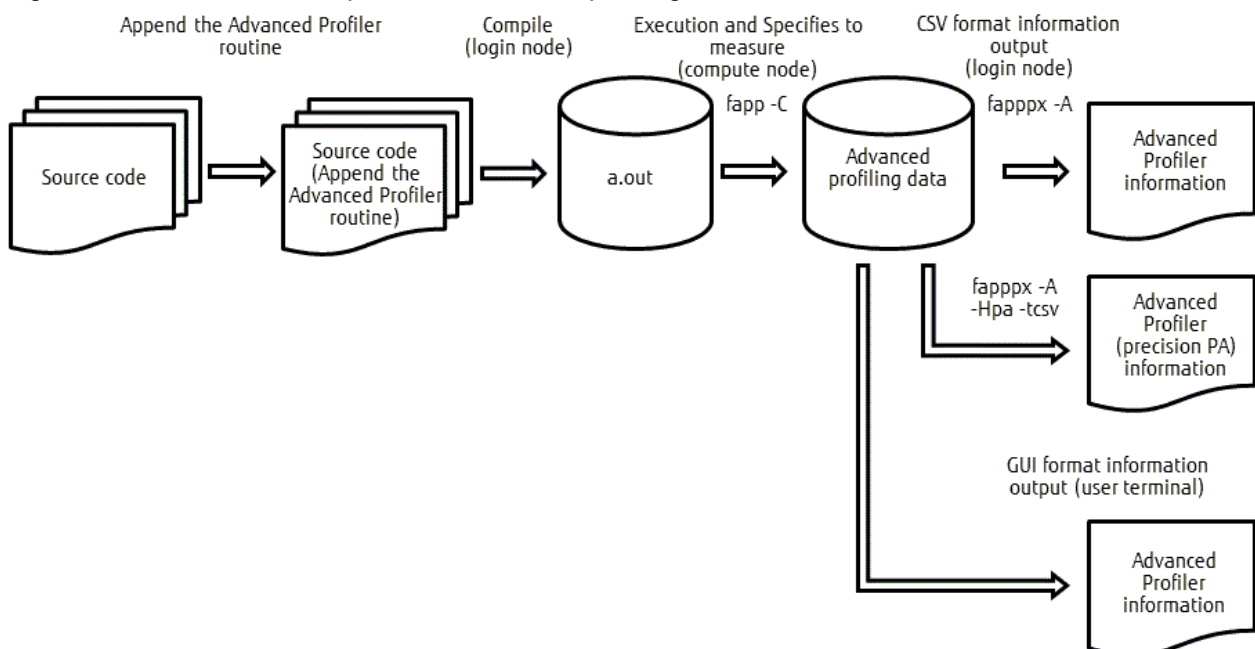
Collects the advanced profiling data of an application in the compute node of the FX100 system

fappx

Outputs the advanced profiling data in the text or CSV format on the login node at the front end

"Figure 3.1 Collection and output of the advanced profiling data" shows the relation between the collection and the output of the advanced profiling data.

Figure 3.1 Collection and output of the advanced profiling data



- fapp -C : Collects the advanced profiling data
- fappx -A : Outputs the Advanced Profiler information in the text format

fapppx -A -Hpa -tcsv : Outputs the Advanced Profiler information on the Hardware monitor information (precision PA) in the CSV format

Collection of advanced profiling data

The fapp command is used for collecting the advanced profiling data.

Output of the Advanced Profiler information

The Advanced Profiler information can be output in following formats:

GUI format

The Advanced Profiler information can be output in the GUI format.

However, the following Advanced Profiler information cannot be output in the GUI format.

- Hardware monitor information (precision PA)
- Largepage memory use information
- Largepage statistical information

Refer to "[3.3 Advanced Profiler information \(GUI format\)](#)" for information on how to output the GUI format.

Text format

The fapppx command is used to output the Advanced Profiler information in the text format. The fapppx command can output the Advanced Profiler information from the preserved advanced profiling data in the text format.

Refer to "[3.2.6 fapppx command](#)" for information on how to output the Advanced Profiler information in the text format.

CSV format

The fapppx command is used to output the Advanced Profiler information on the Hardware monitor information (precision PA) in the CSV format. The fapppx command can output the Advanced Profiler information from the preserved advanced profiling data in the CSV format.

Refer to "[3.2.6 fapppx command](#)" for information on how to output the Advanced Profiler information in the CSV format.

3.2 Using the Advanced Profiler

This section describes the usage of the Advanced Profiler.

3.2.1 Advanced Profiler routine

It is necessary to set the measurement section for using the Advanced Profiler.

Refer to "[3.3 Advanced Profiler information \(GUI format\)](#)" for the GUI formats for the executive summary of the detail profiler. Refer to "[3.4 Advanced Profiler information \(text/CSV formats\)](#)" for the text formats and CSV formats.

Setting the measurement section

Setting the measurement section involves inserting the Advanced Profiler routine at the measurement start position and the measurement end position in the source code. The Advanced Profiler routine can use Fortran subroutines or C/C++ functions. If a C/C++ function is used, the prototype of the function should be declared, or the header file of the Advanced Profiler section specification function should be included.

The table below provides an overview of the Advanced Profiler routine.

Table 3.1 Overview of the Advanced Profiler routine

Language	Header file	Function name (Advanced Profiler routine)	Function	Arguments
Fortran	-	fapp_start	Starts information measurement	(name, number, level)

Language	Header file	Function name (Advanced Profiler routine)	Function	Arguments
		fapp_stop	Ends information measurement	(name, number, level)
C/C++	fj_tool/fapp.h	void fapp_start	Starts information measurement	(const char *name, int number, int level)
		void fapp_stop	Ends information measurement	(const char *name, int number, int level)

Outline of arguments

name: Group name (basic character type scalar)

A group name can comprise alphabets, numbers, and underscores.

- Alphabets

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
a b c d e f g h i j k l m n o p q r s t u v w x y z

- Numbers

0 1 2 3 4 5 6 7 8 9

- Underscore

-

number: Detailed number (4 Bytes integer)

level: Priority level (4 Bytes integer of 0 or more)

The start and the end of the Advanced Profiler specified by a group name and a detailed number are specified.

(The name comprising a group name and a detailed number is distinguished as a measurement range name.)

If the priority level is greater than the value of "-L" option of the fapp command, it does not measure it.

When either the same beginning of the measurement of information on the measurement range name or the specification of the information measurement end is not described, the Advanced Profiler information is not output.

The Advanced Profiler information is output as information for the entire application by the measurement range name of group name ("all") and detailed number (0) besides the specified time base range.

Two or more time base ranges can be specified. A time base range can specify the nest and the overlap.

However, when the same measurement range name is specified for a nest and an overlap, the Advanced Profiler information in the time base range begun first is output.

As for the measurement of thread information, processing of the master thread is begun.

Make it to the time base range including the section where the master thread operates when you measure the range where threads other than the master thread are called.

The Advanced Profiler information in this time base range is not output if the Advanced Profiler routine under measurement does not do the measurement end and it becomes a process end.

An example of setting the Advanced Profiler routine is shown in the figure below.



Example

Example of setting the Advanced Profiler routine

```
#include <fj_tool/fapp.h>

#define SIZE 3000
double a[SIZE][SIZE],b[SIZE][SIZE],c[SIZE][SIZE];
```

```

main()
{
  int i,j;

  fapp_start("region",1,1);

  for(i=0;i<SIZE;i++){
    for(j=0;j<SIZE;j++){
      a[i][j]=(double)(i+j*0.5);
      b[i][j]=(double)(i+j*1.5);
      c[i][j]=a[i][j]+b[i][j];
    }
  }

  fapp_stop("region",1,1);
}

```

3.2.2 Advanced Profiler routine (precision PA)

It is necessary to set the measurement section for using the Advanced Profiler routine (precision PA).

Refer to "[3.5 Precision PA visibility function \(Excel format\)](#)" for the executive summary of the Precision PA visibility function.

Setting of measurement section

Setting the measurement section involves inserting the Advanced Profiler routine (precision PA) at the measurement start position and the measurement end position in the source code. The Advanced Profiler routine (precision PA) can use Fortran subroutines or C/C++ functions. If a C/C++ function is used, the prototype of the function should be declared, or the header file of the Advanced Profiler section specification function should be included.

The table below provides an overview of the Advanced Profiler routine (precision PA).

Table 3.2 Overview of the Advanced Profiler routine (precision PA)

Language	Header file	Function name (Advanced Profiler routine (precision PA))	Function	Arguments (*1)
Fortran	-	start_collection	Starts information measurement	Basic character type scalar
		stop_collection	Ends information measurement	Basic character type scalar
C/C++	fjcoll.h	void start_collection	Starts information measurement	Basic character type scalar
		void stop_collection	Ends information measurement	Basic character type scalar

*1: An argument is used to identify the time base range. The time base ranges between the start_collection and stop_collection functions with the same value as the argument.

Two or more time base ranges can be specified. A time base range can specify the nest and the overlap. An example of the measurement range specification routine is shown in the figure below.

Example

Example of the measurement range specification routine

```

#include <fjcoll.h>

#define SIZE 3000
double a[SIZE][SIZE],b[SIZE][SIZE],c[SIZE][SIZE];

```

```

main()
{
  int i,j;

  start_collection("region");

  for(i=0;i<SIZE;i++){
    for(j=0;j<SIZE;j++){
      a[i][j]=(double)(i+j*0.5);
      b[i][j]=(double)(i+j*1.5);
      c[i][j]=a[i][j]+b[i][j];
    }
  }

  stop_collection("region");
}

```

Even if a subroutine is inserted outside the parallel execution part (parallel region or loop that the loop slice is done), the Hardware monitor information on all threads of a thread-parallel application can be measured. Insert the subroutine in the part where all the processes operate to measure the Hardware monitor information on all the processes in a process-parallel application.

3.2.3 Environment variables

To use the Advanced Profiler, it is necessary to correctly set the environment variables described in the table below.

Table 3.3 Environment variables

Environment variable	Value
PATH	/opt/FJSVmxlang/bin
LD_LIBRARY_PATH	/opt/FJSVmxlang/lib64

To use the batch queuing system and the MPI processing system, setting additional values besides those mentioned above may be necessary.

Refer to the "Job Operation Software First Step Guide" for information on the batch queuing system. Refer to the "MPI User's Guide" for information on the MPI system.

3.2.4 Compilation

For using the Advanced Profiler, it is necessary to create an application linked with the tool library.

Refer to "[2.2.2 Compilation](#)" for information on the tool.

3.2.5 fapp command

The fapp command collects the advanced profiling data of an application.

Format

```
fapp -C -d profiling_data [ -I item ] [ -L level ] [ -H [hardmon] ] exec-file [ exec_option ... ]
```

Options

The table below describes the options that can be specified for the fapp command.

Table 3.4 fapp command options

Option	Description/Specified value (unit)
-C	Specifies the collection processing of the advanced profiling data This option is mandatory.

Option	Description/Specified value (unit)
-d <i>profiling_data</i>	<p>Specifies the name of the advanced profiling data (directory of the advanced profiling data file) to <i>profiling_data</i> by using the relative path or the absolute path.</p> <p>If the directory does not exist, it is newly created. If the directory exists, it must be empty.</p> <p>This option is mandatory.</p>
-I <i>item</i>	<p>Specifies the collecting items of the Advanced Profiler information</p> <p>Delimit using commas if two or more items are specified for <i>item</i>.</p> <p><i>item</i>: { { mpi nomp } { hwm nohwm } { lpgusage nolpgusage } { lpgstats nolpgstats } }</p> <p>The default is as follows:</p> <ul style="list-style-type: none"> - MPI application: -Impi,nohwm,nolpgusage,nolpgstats - Sequential application: -Inomp,nohwm,nolpgusage,nolpgstats
	<p>mpi nomp</p> <p>Specifies whether to collect the MPI information on the MPI application</p> <ul style="list-style-type: none"> - mpi: Collects the MPI information. This option cannot be specified in a sequential application. - nomp: Does not collect the MPI information. <p>"mpi" cannot be specified in a sequential application.</p> <p>The default is as follows:</p> <ul style="list-style-type: none"> - MPI application: mpi - Sequential application: nomp
	<p>hwm nohwm</p> <p>Specifies whether to collect the Hardware monitor information</p> <ul style="list-style-type: none"> - hwm: Collects the Hardware monitor information - nohwm: Does not collect the Hardware monitor information <p>The default is "nohwm".</p>
	<p>lpgusage nolpgusage</p> <p>Specifies whether to collect the Largepage memory use information.</p> <ul style="list-style-type: none"> - lpgusage: Collects the Largepage memory use information. - nolpgusage: Does not collect the Largepage memory use information. <p>The default is "nolpgusage".</p> <p>When the "-lpgusage" is specified, cannot be specified at the same time as "-Impi", "-Ihwm", "-lpgstats", or "-H" option.</p> <p>When the "-lpgusage" is specified in MPI applications, "-Impi" of the default value becomes ineffective, and MPI information is not collected.</p> <p>When the "-lpgusage" is specified, the Largepage memory use information cannot be output in the GUI format.</p>
	<p>lpgstats nolpgstats</p> <p>Specifies whether to collect the Largepage statistical information.</p> <ul style="list-style-type: none"> - lpgstats: Collects the Largepage statistical information. - nolpgstats: Does not collect the Largepage statistical information. <p>The default is "nolpgstats".</p> <p>When the "-lpgstats" is specified, cannot be specified at the same time as "-Impi", "-Ihwm", "-lpgusage", or "-H" option.</p> <p>When the "-lpgstats" is specified in MPI applications, "-Impi" of the default value becomes ineffective, and MPI information is not collected.</p>

Option	Description/Specified value (unit)
	When the "-Ilpgstats" is specified, the Largepage statistical information cannot be output in the GUI format.
-L <i>level</i>	<p>Specifies the start level of the measuring object</p> <p>The Advanced Profiler measures when the <i>level</i> is more than the priority level of the section specified for the Advanced Profiler routine.</p> <p>The default is 0.</p>
-H [<i>hardmon</i>]	<p>Specifies measurement of the Hardware monitor information</p> <p><i>hardmon</i> can be omitted (only "-H" can be specified).</p> <p>Delimit using commas if two or more items are specified for <i>hardmon</i>.</p> <p>The default is as follows:</p> <ul style="list-style-type: none"> - "-Ihwm" option is used: -Hevent=Statistics,mode=sys,method=normal - "-Inohwm" option is used: Does not collect the Hardware monitor information <p>If this option is specified, it is considered that "-Ihwm" was specified.</p> <p><i>hardmon</i>: { event=<i>event</i> event_number=<i>no</i> mode=<i>mode</i> pa=<i>no</i> method=<i>method</i> }</p>
event= <i>event</i> : { Cache Instructions_SIMD Instructions_NOSIMD MEM_access Performance Statistics TLB }	<p>Specifies the measurement event of the Hardware monitor information</p> <p>Any of the following can be specified for <i>event</i>:</p> <ul style="list-style-type: none"> - Cache: Cache miss rate - Instructions_SIMD : Execution instruction detail (SIMD) - Instructions_NOSIMD : Execution instruction detail (NOSIMD) - MEM_access: Memory access situation - Performance: Instruction execution efficiency - Statistics: CPU core operation situation - TLB : TLB miss rate <p>The default is "event=Statistics".</p>
event_number= <i>no</i>	<p>Specifies the measurement event number of Hardware monitor information.</p> <ul style="list-style-type: none"> - <i>no</i> : 0 - 127 <p>The measurement event number "<i>no</i>" is a value specified as "<i>no</i>" of the PA counter corresponding to eight PIC (picu0, picl0, picu1, picl1, picu2, picl2, picu3, picl3).</p> <p>Delimit using commas if two or more numbers are specified.</p> <p>For more information about the event number and PIC, refer to the SPARC64 XIfx Architecture Manuals.</p> <p>When this option is specified, Advanced Profiler routine (precision PA) becomes effective and the Advanced Profiler routine becomes ineffective. Refer to "3.2.2 Advanced Profiler routine (precision PA)" for the detail of the Advanced Profiler routine (precision PA).</p> <p>This option cannot be specified at the same time as "-Impi" option.</p> <p>Specify "-Hpa" of fapppx in the option to display the Hardware monitor information on the specified measurement event number.</p> <p>When this option is specified, the output profiling data cannot be analyzed by the GUI format.</p>
mode= <i>mode</i> : { sys usr }	<p>Specifies the measurement mode of the Hardware monitor information</p> <p>One of the following is specified for <i>mode</i>.</p>

Option	Description/Specified value (unit)
	<ul style="list-style-type: none"> - sys: Collects information on the kernel mode and the user mode - usr: Collects information on the user mode <p>The default is "mode=sys".</p>
pa= <i>no</i>	<p>The frequencies (times <i>no</i>) that Hardware monitor information (precision PA) collects are specified.</p> <ul style="list-style-type: none"> - <i>no</i>: 1 - 11 <p>When this option is specified, Advanced Profiler routine (precision PA) becomes effective and the Advanced Profiler routine becomes ineffective. Refer to "3.2.2 Advanced Profiler routine (precision PA)" for the detail of the Advanced Profiler routine (precision PA).</p> <p>This option cannot be specified at the same time as "-Impi" option.</p> <p>When this option is specified, the output profiling data cannot be analyzed by the GUI format.</p>
method= <i>method</i> : { raw normal }	<p>Specifies the measurement mode of Hardware monitor information.</p> <ul style="list-style-type: none"> - raw : This mode measures hardware information directly, thus performing highly precise measurement of Hardware monitor information. However, processing such as direct I/O to global file systems might fail in Hardware monitor information measurements. Refer to "Hardware monitor information" under "A.2 Advanced Profiler", in "Appendix A Considerations for Using the Profiler", for details. - normal : This mode makes measurements via the operating system to measure Hardware monitor information. <p>Please make "-Hpa=<i>no</i>" or "-Hevent_number=<i>no</i>" effective when you specify "raw".</p> <p>The default is "method=normal".</p>
exec-file [<i>exec_option</i> ...]	<p>Specifies the target execution file for the Advanced profiling data collection and the option.</p> <ul style="list-style-type: none"> - <i>exec-file</i> : Specifies mpiexec when MPI application is used. Specify the absolute path or the relative path containing the current directory (".") if specifying the execution file that starts in "-". The shell script cannot be specified. - <i>exec_option</i> ... : Specifies the option to <i>exec-file</i>. The character string following an execution file name is regarded as the option to an execution file.



Example

Example: An MPI application, a.out, of two parallels is executed, and the MPI information is obtained.

```
$ fapp -C -d FAPP_Example mpiexec -n 2 ./a.out
```

In this example, the advanced profiling data collected by the fapp command is stored in the FAPP_Example directory.

3.2.6 fappx command

The fappx command outputs the Advanced Profiler information in the text or CSV format.

Format

```
fappx -A [ -I item ] [ -o outfile ] [ -p p_no ] [ -H hardmon ] [ -l limit ] [ -t type ]  
-d profiling_data
```


Options

The table below describes the options that can be specified for the fapppx command.

Table 3.5 fapppx command options

Option	Description/Specified value (unit)
-A	Specifies the output processing of the Advanced Profiler information. This option is mandatory.
-I <i>item</i>	Specifies the output items of the Advanced Profiler information Delimit using commas if two or more items are specified for <i>item</i> . <i>item</i> : { { mpi nompi } { hwm nohwm } { lpg nolpg } } The default is as follows: - MPI application: -Impi,nohwm,nolpg - Sequential application: -Inompi,nohwm,nolpg
mpi nompi	Specifies whether to output of the MPI information on the MPI application - mpi: Outputs the MPI information. Nothing is output in a sequential application. - nompi: Does not output the MPI information. The default is as follows: - MPI application: mpi - Sequential application: nompi
hwm nohwm	Specifies whether to output of the Hardware monitor information - hwm: Outputs the Hardware monitor information - nohwm: Does not output the Hardware monitor information The default is "nohwm".
lpg nolpg	Specifies whether to output the Largepage (Largepage memory use information or Largepage statistical information) information. - lpg: Outputs the Largepage information - nolpg: Does not output the Largepage information The default is "nolpg".
-o <i>outfile</i>	Specifies the output destination of the Advanced Profiler information If stdout is specified for <i>outfile</i> , the Advanced Profiler information is output to a standard output. Specify the absolute path or the relative path containing the current directory (".") if specifying <i>outfile</i> that starts in "-". The default is "-ostdout".
-p <i>p_no</i>	Specifies the target process to be input and output with the Advanced Profiler information. The information of application unit of the Advanced Profiler information is calculated using the advanced profiling data of target process specified by this option. Delimit using commas if two or more target process numbers (<i>p_no</i>) are specified. The one specified later is valid, if two or more target process numbers (<i>p_no</i>) are specified. An error is detected, if <i>p_no</i> is omitted. <i>p_no</i> : { all <i>N</i> , <i>N</i> ... input= <i>n</i> limit= <i>m</i> } The default is "-pinput=0,limit=16".

Option	Description/Specified value (unit)
all $N_1, N_2 \dots$ input= n limit= m	<ul style="list-style-type: none"> - all : Data on all processes is input and information on all processes is output in order with a high cost. - $N_1, N_2 \dots$: Data on process number N is input and information on process number N is output ahead of information on the process with a high cost. When process number N does not exist, specification is disregarded. - input=n : Data on n processes is input. When the value that exceeds 0 or the number of processes is specified for n, data on all processes is input. The default is "input=0". - limit=m : Information on m processes is output. When 0 or the value that exceeds n is specified for m, information on n processes is output. The default is "limit=16".
-t <i>type</i>	Specifies the output format of the Advanced Profiler information The default is "-ttext".
csv text	<ul style="list-style-type: none"> - csv: Specifies to output the Advanced Profiler information in the CSV format. - text: Specifies to output the Advanced Profiler information in the text format.
-H <i>hardmon</i>	Output the Hardware monitor information (precision PA) or the measurement event number specification. This option only targets the advanced profiling data collected by the fapp command and analyzes it by specifying the "-Hpa=no" or "-Hevent_number=no" option.
pa	- pa: Outputs in the precise PA format or measurement event number specification.
-l <i>limit</i>	The output number in the time base range output to the form of Hardware monitor information (precision PA) is specified. - <i>limit</i> : Integers from 0 through 2,147,483,647 can be specified to define the range (output number) Everything is output in case of 0. The default is 10. Please specify "-Hpa" option to make this option effective.
-d <i>profiling_data</i>	Specifies the name of the advanced profiling data (name of the directory that stores the advanced profiling data file) to <i>profiling_data</i> by using the relative path or the absolute path. If specifying <i>profiling_data</i> that starts in "-", specify the absolute path or the relative path containing the current directory ("./"). When this option is specified at the end of an optional list of the fappx command, "-d" can be omitted. This option is mandatory.

3.3 Advanced Profiler information (GUI format)

This section explains the contents of the Advanced Profiler information.

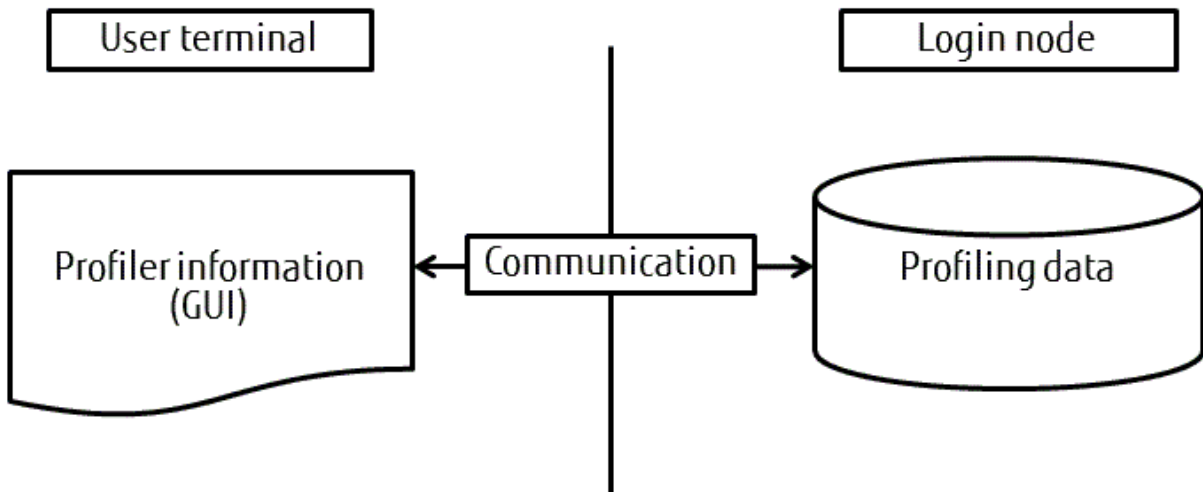
The Advanced Profiler information is displayed using the Profiler GUI.

3.3.1 Overview of the Profiler feature

The profiler function analyses and gives visibility to the profiling data collected by the Instant Profiler (fipp command) and the Advanced Profiler (fapp command).

The "Figure 3.2 Diagram of Profiler information" is shown below.

Figure 3.2 Diagram of Profiler information



Specify the profiling data on the login node to visualize it using the Profiler feature of the user terminal.

3.3.2 Starting the Profiler

Click the **Profiler** icon in the main window of FUJITSU Software Development Tools (FSDT) to start the Profiler. Refer to the "Programming Workbench User's Guide" for information on FSDT.

Figure 3.3 Profiler icon



3.3.3 Profiler information window

This section describes the profiler information window.

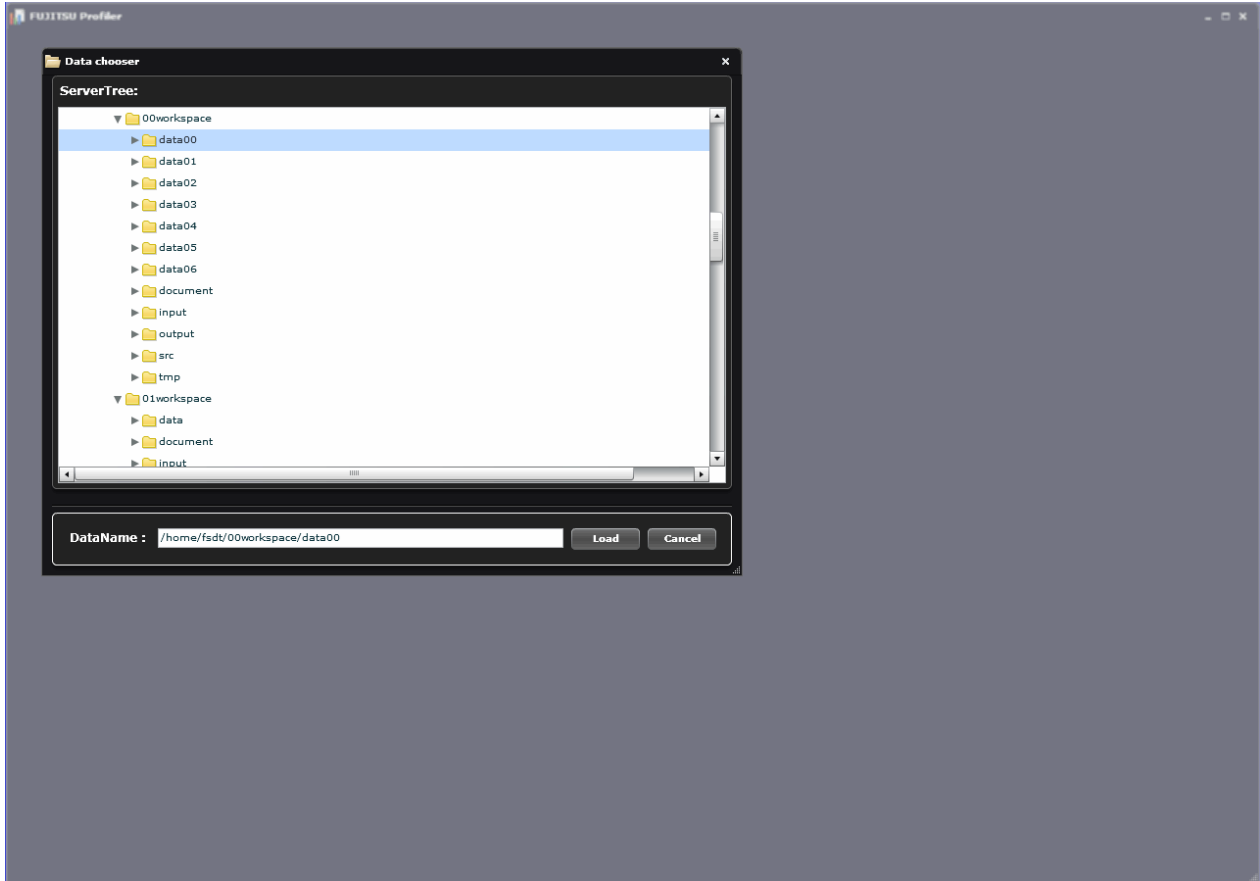
3.3.3.1 Profiling data selection window

When the Profiler is started, the profiling data selection window is displayed. The directories on the login node are displayed in the profiling data selection window in a tree structure. Select the advanced profiling data from the tree, and then click **Load** to start the data reading. When the data reading completes, the Advanced Profiler information window is displayed.

Note that the advanced profiling data is a directory.

Up to 9216 parallel processes can be displayed in the Profiler. Parallel profiling data that exceeds this count cannot be used by the Profiler.

Figure 3.4 Profiling data selection window

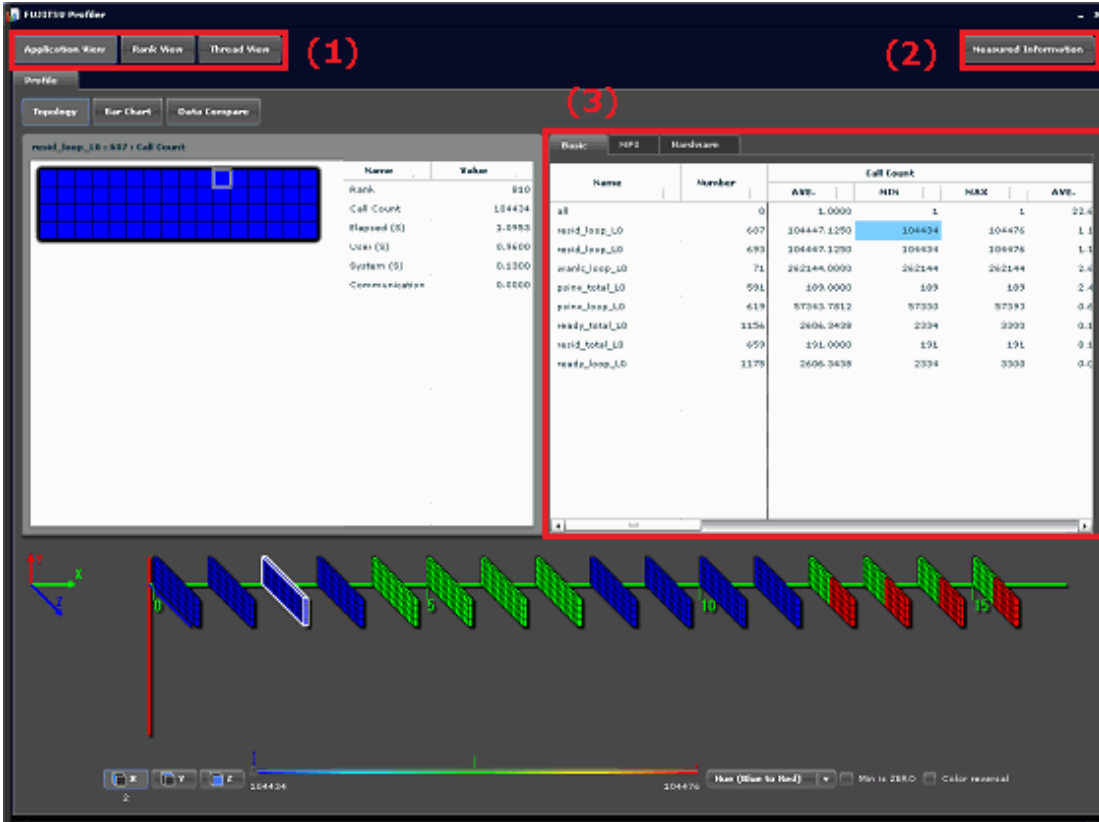


3.3.3.2 Overview of the Advanced Profiler information window

When the advanced profiling data is input, the Advanced Profiler information window is displayed. The Advanced Profiler information window comprises the following elements:

- (1) Display unit switching buttons
- (2) Measured Information button
- (3) Advanced Profiler information area

Figure 3.5 Advanced Profiler information window



(1) Display unit switching buttons

The unit of display shown in the Advanced Profiler information window can be switched.

Table 3.6 Display unit

Button	Display unit
Application View	Total information on each application is displayed
Rank View	Total information on a specific rank is displayed
Thread View	Information on a specific thread is displayed

The **Rank View** and the **Thread View** buttons are only enabled if the Hardware monitor information is measured. If the Hardware monitor information is not measured, these buttons are disabled.

If either **Rank View** or **Thread View** is selected, a box for rank selection is displayed. The rank to be displayed can be selected by using the box.

If **Thread View** is selected, a box for thread selection is displayed. The thread to be displayed can be selected using the box.

(2) Measured Information button

Click the **Measured Information** button to display the **Measured Information** window. The measured state of an application, such as the frequency of the machine, is displayed in the **Measured Information** window.

Figure 3.6 Measured Information window

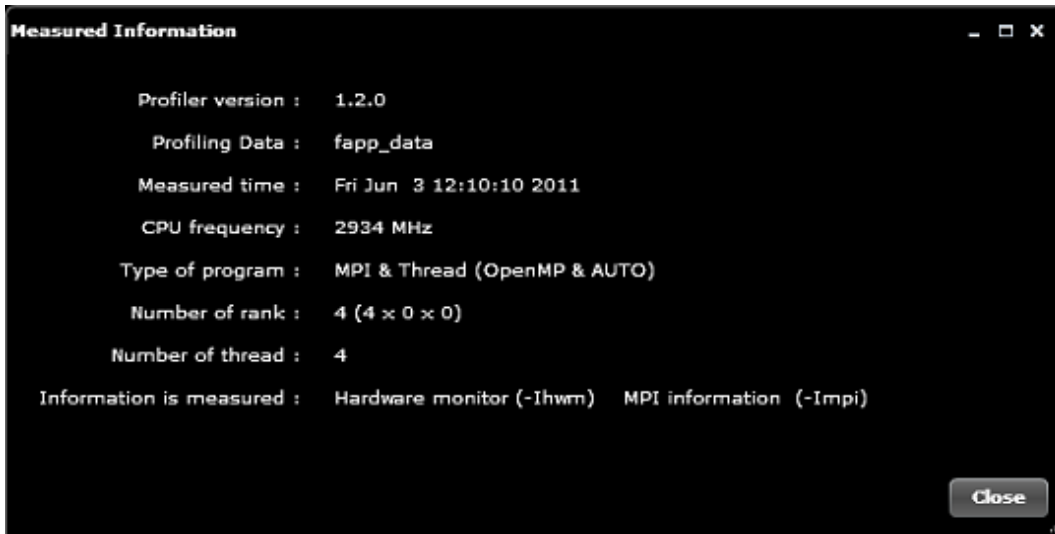


Table 3.7 Measured Information

Label	Description
Profiler version	Version of Profiler
Profiling Data	Name of the displayed profiling data
Measured time	Measured date and time
CPU frequency	Frequency of the measured machine
Type of program	Execution format of the application <ul style="list-style-type: none"> - SERIAL : Sequential - Thread : Thread parallel - Thread (OpenMP) : Thread parallel by OpenMP - Thread (AUTO) : Thread parallel by automatic parallel function of the compiler - MPI : MPI
Number of rank	Number of ranks, topology shape
Number of thread	Number of threads
Information is measured	Collected information <ul style="list-style-type: none"> - Default : the Basic information - Hardware monitor (-Ihwm) : the Basic information, and the Hardware monitor information - MPI information (-Impi) : the Basic information, and the MPI information - Hardware monitor (-Ihwm) MPI information (-Impi) : the Basic information, the Hardware monitor information, and the MPI information

(3)Advanced Profiler information area

The Advanced Profiler information is displayed.

3.3.3.3 Advanced Profiler information

The display can be switched using the display switch buttons in **Application View** and **Rank View**.

In **Thread View**, there is no display switch button and the information is displayed in the table format.

Table 3.8 Display methods

Button	Information
Topology	Distribution information to the rank on detailed performance information in the measurement section is displayed according to the topology shape of the application. This information can only be viewed in Application View .
Panel	Distribution information to the thread on detailed performance information in the measurement section is displayed. This information can only be viewed in Rank View .
Bar Chart	Distribution information to the rank or the thread on detailed performance information in the measurement section is displayed as a bar chart. This information can only be viewed in Application View and Rank View .
Data Compare	The whole graph of three times (for three items) of displayed by selecting the row in the Topology, Panel and Bar Chart is arranged vertically and displayed. This information is only available in Application View and Rank View .

3.3.3.3.1 Topology/Panel information

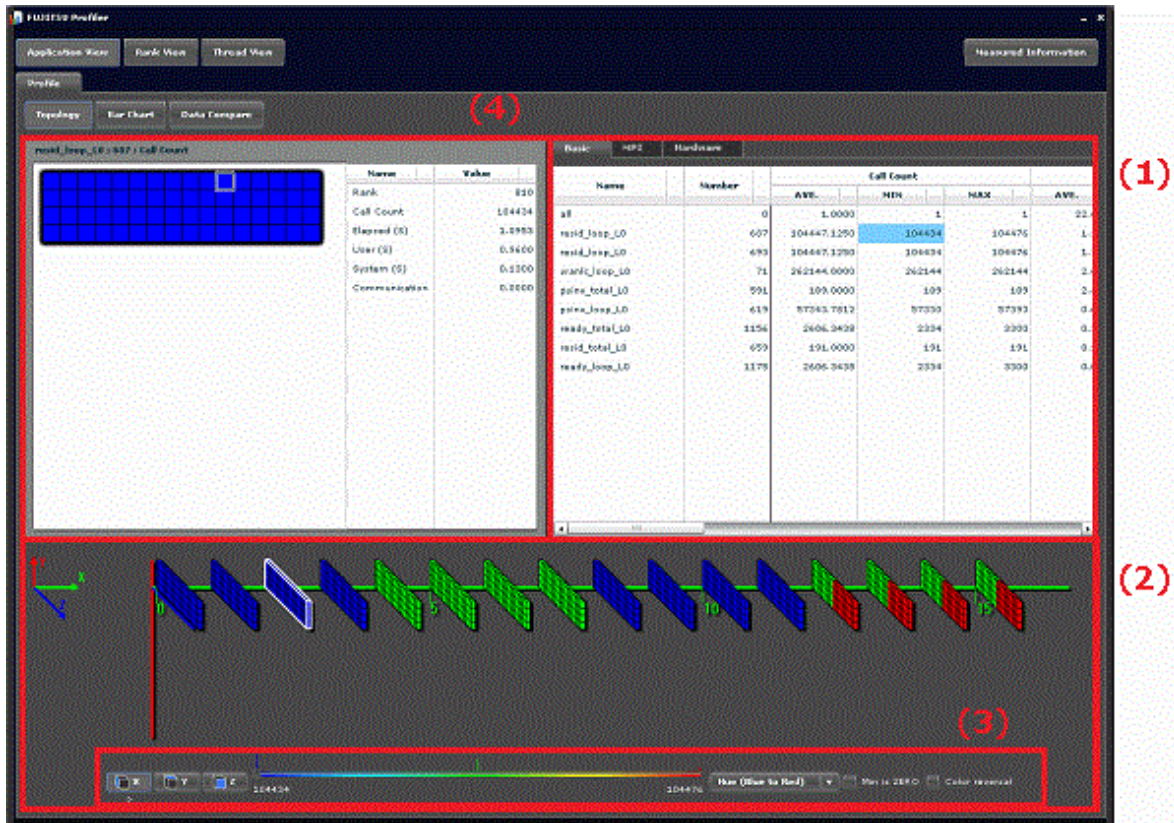
In the **Topology** information, the distribution shape to the rank of the cost information can be displayed by the form along the communication topology. In the **Panel** information, the distribution shape to the thread of the cost information can be displayed.

The display form of **Panel** information becomes one dimension shape and the same of Topology information.

The **Topology** or **Panel** information window comprises the following elements:

- (1) Advanced Profiler information list
- (2) Whole graph
- (3) Color histogram
- (4) Zoom information panel

Figure 3.7 Topology information window



(1) Advanced Profiler information list

The Advanced Profiler information list displays the cost information in the table format.

By clicking a cell, the distribution situation between parallel about item information on selection row concerning selection line are displayed in whole graph. Moreover, the selected unit name and item name are displayed in the title of the zoom information panel. The item of displayed distribution information becomes information as the group row of the selected cell. Even if AVE, MIN, or MAX is selected, it becomes a same deal. Moreover, it is an item that has distribution information that shows the performance. The Name, Number, and MPI columns do not have distribution information. It is assumed the one having been selected for the row with the distribution information at the right of the selection row when selected. It is assumed the one having been selected for the row with the first distribution information of the line when there is no row with the distribution information right.

In the initial state, the Basic tab is displayed with the distribution information selected in the first row.

The cost information is as follows:

a. Basic information

The basic information is displayed on the **Basic** tab.

Information on the call frequency in the measurement section and time is displayed.

Table 3.9 Items of Basic information

Label	Description
Name	Measurement section name
Number	Measurement section group number
Call Count	Call frequency for the measurement section
AVE.	Average value between ranks
MIN	Minimum value between ranks
MAX	Maximum value between ranks

Label		Description
Elapsed (S)		Elapsed time
	AVE.	Average value between ranks
	MIN	Minimum value between ranks
	MAX	Maximum value between ranks
User (S)		User CPU time
	AVE.	Average value between ranks
	MIN	Minimum value between ranks
	MAX	Maximum value between ranks
System (S)		System CPU time
	AVE.	Average value between ranks
	MIN	Minimum value between ranks
	MAX	Maximum value between ranks
Communication (S)		Communication time of processes (Only the whole application)
	AVE.	Average value between ranks
	MIN	Minimum value between ranks
	MAX	Maximum value between ranks

b. MPI information

The MPI information is displayed on the **MPI** tab.

It displays information about MPI functions called from the measurement section.

Table 3.10 Items of MPI information (parent node)

Label	Description
Name	Measurement section name
Number	Measurement section group number

Table 3.11 Items of MPI information (child node)

Label		Description
MPI		MPI function name
Call Count		Call frequency for the measurement section
	Total	Frequency for MPI functions called from the measurement section
	AVE.	Average value between ranks
	MIN	Minimum value between ranks
	MAX	Maximum value between ranks
0KB <= message < 4KB		Call frequency when transferring from 0 KB to less than 4 KB of message length
	AVE.	Average value between ranks
	MIN	Minimum value between ranks
	MAX	Maximum value between ranks
4KB <= message < 64KB		Call frequency when transferring from 4 KB to less than 64 KB of message length
	AVE.	Average value between ranks
	MIN	Minimum value between ranks
	MAX	Maximum value between ranks

Label		Description
	64KB <= message < 1024KB	Call frequency when transferring from 64 KB to less than 1024 KB of message length
	AVE.	Average value between ranks
	MIN	Minimum value between ranks
	MAX	Maximum value between ranks
	1024KB <= message	Call frequency when transferring 1024 KB or more of message length
	AVE.	Average value between ranks
	MIN	Minimum value between ranks
	MAX	Maximum value between ranks
	Elapsed (S)	Elapsed time
	AVE.	Average value between ranks
	MIN	Minimum value between ranks
	MAX	Maximum value between ranks
Wait (S)	Communication latency	
	AVE.	Average value between ranks
	MIN	Minimum value between ranks
	MAX	Maximum value between ranks
Average message (Byte)	Average forwarding message length	
	AVE.	Average value between ranks
	MIN	Minimum value between ranks
	MAX	Maximum value between ranks

c. Hardware monitor information

The operation situation of the processor during application execution is output.

The Hardware monitor information is classified into groups that comprise two or more items and is called an event.

The events of the Hardware monitor information are listed below. A group is specified on measuring.

1. Cache

The Cache event is used to check the cache miss and memory access.

Table 3.12 Items of Cache

Label		Description
Name		Measurement section name
Number		Measurement section group number
Elapsed (S)		Elapsed time
	AVE.	Average value between ranks in Application View
		Average value between threads in Rank View
	MIN	Minimum value between ranks in Application View
		Minimum value between threads in Rank View
	MAX	Maximum value between ranks in Application View
		Maximum value between threads in Rank View
Instructions		Executed number of instructions (number of instruction executions) Number of instruction executions : effective_instruction_counts

Label		Description
	AVE.	Average value between ranks in Application View Average value between threads in Rank View
	MIN	Minimum value between ranks in Application View Minimum value between threads in Rank View
	MAX	Maximum value between ranks in Application View Maximum value between threads in Rank View
L1 instruction miss (%)		First instruction cache miss rate generated in instruction executions Number of occurrences of first instruction cache misses / Number of instruction executions * 100 Number of occurrences of first instruction cache misses : L1I_miss Number of instruction executions : effective_instruction_counts
	AVE.	Average value between ranks in Application View Average value between threads in Rank View
	MIN	Minimum value between ranks in Application View Minimum value between threads in Rank View
	MAX	Maximum value between ranks in Application View Maximum value between threads in Rank View
L1 operand miss (%)		First data cache miss rate generated in memory access (loading store) instruction execution Number of occurrences of first data cache cache misses / Number of committed "load/store" instructions * 100 Number of occurrences of first data cache cache misses : L1D_miss Number of committed "load/store" instructions : load_store_instructions + 2 * SIMD_load_store_instructions + 4 * 4SIMD_load_store_instructions
	AVE.	Average value between ranks in Application View Average value between threads in Rank View
	MIN	Minimum value between ranks in Application View Minimum value between threads in Rank View
	MAX	Maximum value between ranks in Application View Maximum value between threads in Rank View
L2 cache miss (%)		Level 2 cache miss rate generated in memory access (loading store) instruction execution Number of level 2 cache miss / Number of committed "load/store" instructions * 100 Number of level 2 cache miss : L2_miss_dm + L2_miss_pf Number of committed "load/store" instructions: load_store_instructions + 2 * SIMD_load_store_instructions + 4 * 4SIMD_load_store_instructions
	AVE.	Average value between ranks in Application View Average value between threads in Rank View
	MIN	Minimum value between ranks in Application View Minimum value between threads in Rank View
	MAX	Maximum value between ranks in Application View Maximum value between threads in Rank View

Label	Description
L2 demand cache miss (%)	<p>Rate of demand demand misses in level 2 cache miss</p> <p>The demand demand is to access the level 2 cache with the resources necessary for memory access was able to be acquired.</p> <p>Number of level 2 cache misses caused by demand requests / Number of level 2 cache miss * 100</p> <p>Number of level 2 cache misses caused by demand requests : L2_miss_dm</p> <p>Number of level 2 cache miss : L2_miss_dm + L2_miss_pf</p>
AVE.	<p>Average value between ranks in Application View</p> <p>Average value between threads in Rank View</p>
MIN	<p>Minimum value between ranks in Application View</p> <p>Minimum value between threads in Rank View</p>
MAX	<p>Maximum value between ranks in Application View</p> <p>Maximum value between threads in Rank View</p>
L2 prefetch cache miss (%)	<p>Rate of prefetch demand misses in level 2 cache miss</p> <p>The prefetch demand is the hardware prefetch state that cannot acquire the resources necessary for memory access, and it accesses the level 2 cache.</p> <p>Number of level 2 cache misses caused by prefetch requests / Number of level 2 cache miss * 100</p> <p>Number of level 2 cache misses caused by prefetch requests : L2_miss_pf</p> <p>Number of level 2 cache miss : L2_miss_dm + L2_miss_pf</p>
AVE.	<p>Average value between ranks in Application View</p> <p>Average value between threads in Rank View</p>
MIN	<p>Minimum value between ranks in Application View</p> <p>Minimum value between threads in Rank View</p>
MAX	<p>Maximum value between ranks in Application View</p> <p>Maximum value between threads in Rank View</p>

2. Instructions_SIMD

The Instructions_SIMD event is used to check the instructions for processing floating-point data which SIMD conversion is applied to.

SIMD instruction processes two or more operands in one instruction.

Table 3.13 Items of Instructions_SIMD

Label	Description
Name	Measurement section name
Number	Measurement section group number
Elapsed (S)	Elapsed time
AVE.	<p>Average value between ranks in Application View</p> <p>Average value between threads in Rank View</p>
MIN	<p>Minimum value between ranks in Application View</p> <p>Minimum value between threads in Rank View</p>
MAX	<p>Maximum value between ranks in Application View</p> <p>Maximum value between threads in Rank View</p>

Label	Description
Instructions	Executed number of instructions (number of instruction executions) Number of instruction executions : effective_instruction_counts
AVE.	Average value between ranks in Application View Average value between threads in Rank View
MIN	Minimum value between ranks in Application View Minimum value between threads in Rank View
MAX	Maximum value between ranks in Application View Maximum value between threads in Rank View
MIPS	Instruction execution efficiency (average of the number of instruction executions per second) Number of instruction executions / Elapsed time / 1.0e+6 Number of instruction executions : effective_instruction_counts
AVE.	Average value between ranks in Application View Average value between threads in Rank View
MIN	Minimum value between ranks in Application View Minimum value between threads in Rank View
MAX	Maximum value between ranks in Application View Maximum value between threads in Rank View
MIPS/PEAK (%)	Rate of actual measurement values to the logical peak value of MIPS $MIPS / (\text{Number of execution cores} * \text{MIPS peak value of each core}) * 100$
AVE.	Average value between ranks in Application View Average value between threads in Rank View
MIN	Minimum value between ranks in Application View Minimum value between threads in Rank View
MAX	Maximum value between ranks in Application View Maximum value between threads in Rank View
Load/Store (SIMD) (%)	Rate of SIMD and 4SIMD load/store instruction number of instruction executions Number of committed "SIMD and 4SIMD load/store" instructions / Number of instruction executions * 100 Number of committed "SIMD and 4SIMD load/store" instructions : XSIMD_load_store_instructions Number of instruction executions : effective_instruction_counts
AVE.	Average value between ranks in Application View Average value between threads in Rank View
MIN	Minimum value between ranks in Application View Minimum value between threads in Rank View
MAX	Maximum value between ranks in Application View Maximum value between threads in Rank View
Floating operand (SIMD) (%)	Rate of SIMD and 4SIMD floating-point instruction number of instruction executions Number of committed "SIMD and 4SIMD floating-point" instruction / Number of instruction executions * 100

Label	Description
	<p>Number of committed "SIMD and 4SIMD floating-point" instruction : XSIMD_floating_instructions</p> <p>Number of instruction executions : effective_instruction_counts</p>
AVE.	<p>Average value between ranks in Application View</p> <p>Average value between threads in Rank View</p>
MIN	<p>Minimum value between ranks in Application View</p> <p>Minimum value between threads in Rank View</p>
MAX	<p>Maximum value between ranks in Application View</p> <p>Maximum value between threads in Rank View</p>
FMA (SIMD) (%)	<p>Rate of floating-point FMA instructions (SIMD execution) in number of instruction execution</p> <p>Number of committed "4SIMD floating point multiply and add/sub" and "4SIMD floating point trigonometric" instructions / Number of instruction execution * 100</p> <p>Number of committed "4SIMD floating point multiply and add/sub" and "4SIMD floating point trigonometric" instructions : XSIMD_fma_instructions</p> <p>Number of instruction executions : effective_instruction_counts</p>
AVE.	<p>Average value between ranks in Application View</p> <p>Average value between threads in Rank View</p>
MIN	<p>Minimum value between ranks in Application View</p> <p>Minimum value between threads in Rank View</p>
MAX	<p>Maximum value between ranks in Application View</p> <p>Maximum value between threads in Rank View</p>
Fixed operand (SIMD) (%)	<p>Rate of SIMD and 4SIMD fixed point partitioned add/sub and integer multiply add instructions in number of instruction executions</p> <p>Number of committed "SIMD and 4SIMD fixed point partitioned add/sub" and "integer multiply add" instructions / Number of instruction execution * 100</p> <p>Number of committed "SIMD and 4SIMD fixed point partitioned add/sub" and "integer multiply add" instructions : XSIMD_fixed_point_instructions</p> <p>Number of instruction executions : effective_instruction_counts</p>
AVE.	<p>Average value between ranks in Application View</p> <p>Average value between threads in Rank View</p>
MIN	<p>Minimum value between ranks in Application View</p> <p>Minimum value between threads in Rank View</p>
MAX	<p>Maximum value between ranks in Application View</p> <p>Maximum value between threads in Rank View</p>
SIMD (%)	<p>Rate of SIMD instructions to the number of total instruction executions</p> <p>(Number of committed "SIMD and 4SIMD load/store" instructions + Number of committed "4SIMD floating point" instructions + Number of committed "4SIMD floating point multiply and add/sub" and "4SIMD floating point trigonometric" instructions + Number of committed "SIMD and 4SIMD fixed point partitioned add/sub" and "integer multiply add" instructions) / Number of instruction executions * 100</p> <p>Number of committed "SIMD and 4SIMD load/store" instructions : XSIMD_load_store_instructions</p>

Label		Description
		<p>Number of committed "SIMD and 4SIMD floating point" instructions : XSIMD_floating_instructions</p> <p>Number of committed "4SIMD floating point multiply and add/sub" and "4SIMD floating point trigonometric" instructions : XSIMD_fma_instructions</p> <p>Number of committed "SIMD and 4SIMD fixed point partitioned add/sub" and "integer multiply add" instructions : XSIMD_fixed_point_instructions</p> <p>Number of instruction executions : effective_instruction_counts</p>
	AVE.	<p>Average value between ranks in Application View</p> <p>Average value between threads in Rank View</p>
	MIN	<p>Minimum value between ranks in Application View</p> <p>Minimum value between threads in Rank View</p>
	MAX	<p>Maximum value between ranks in Application View</p> <p>Maximum value between threads in Rank View</p>
SIMD Load/Store (%)		<p>Rate of SIMD and 4SIMD floating-point loading store instructions that occupies it to the number of floating-point loading store instructions</p> <p>Number of committed "SIMD and 4SIMD load/store" instructions / (Number of committed "SIMD and 4SIMD load/store" instructions + Number of committed "load/store" instructions) * 100</p> <p>Number of committed "SIMD and 4SIMD load/store" instructions : XSIMD_load_store_instructions</p> <p>Number of committed "load/store" instructions : load_store_instructions</p>
	AVE.	<p>Average value between ranks in Application View</p> <p>Average value between threads in Rank View</p>
	MIN	<p>Minimum value between ranks in Application View</p> <p>Minimum value between threads in Rank View</p>
	MAX	<p>Maximum value between ranks in Application View</p> <p>Maximum value between threads in Rank View</p>
Prefetch		<p>Number of prefetch instructions</p> <p>Number of committed "prefetch" instructions + Number of committed "NonSIMD, SIMD and 4SIMD of indirect prefetch" instructions</p> <p>Number of committed "prefetch" instructions : prefetch_instructions</p> <p>Number of committed "NonSIMD, SIMD and 4SIMD of indirect prefetch" instructions : nonSIMD_XSIMD_indirect_prefetch_instructions</p>
	AVE.	<p>Average value between ranks in Application View</p> <p>Average value between threads in Rank View</p>
	MIN	<p>Minimum value between ranks in Application View</p> <p>Minimum value between threads in Rank View</p>
	MAX	<p>Maximum value between ranks in Application View</p> <p>Maximum value between threads in Rank View</p>
Indirect Prefetch		<p>Number of committed "NonSIMD, SIMD and 4SIMD of indirect prefetch" instructions</p> <p>Number of committed "NonSIMD, SIMD and 4SIMD of indirect prefetch" instructions : nonSIMD_XSIMD_indirect_prefetch_instructions</p>
	AVE.	<p>Average value between ranks in Application View</p>

Label		Description
		Average value between threads in Rank View
	MIN	Minimum value between ranks in Application View Minimum value between threads in Rank View
	MAX	Maximum value between ranks in Application View Maximum value between threads in Rank View

3. Instructions_NOSIMD

The Instructions_NOSIMD event is used to check the instructions for processing floating-point data which SIMD conversion is not applied to.

NOSIMD instruction processes one operand in one instruction.

Table 3.14 Items of Instructions_NOSIMD

Label		Description
Name		Measurement section name
Number		Measurement section group number
Elapsed (S)		Elapsed time
	AVE.	Average value between ranks in Application View Average value between threads in Rank View
	MIN	Minimum value between ranks in Application View Minimum value between threads in Rank View
	MAX	Maximum value between ranks in Application View Maximum value between threads in Rank View
MIPS		Instruction execution efficiency (average of the number of instruction executions per second) Number of instruction executions / Elapsed time / 1.0e+6 Number of instruction executions : effective_instruction_counts
	AVE.	Average value between ranks in Application View Average value between threads in Rank View
	MIN	Minimum value between ranks in Application View Minimum value between threads in Rank View
	MAX	Maximum value between ranks in Application View Maximum value between threads in Rank View
MIPS/PEAK (%)		Rate of actual measurement values to the logical peak value of MIPS $MIPS / (\text{Number of execution cores} * MIPS \text{ peak value of each core}) * 100$
	AVE.	Average value between ranks in Application View Average value between threads in Rank View
	MIN	Minimum value between ranks in Application View Minimum value between threads in Rank View
	MAX	Maximum value between ranks in Application View Maximum value between threads in Rank View
Load/Store (NOSIMD) (%)		Rate of load/store instruction number of instruction executions (NOSIMD execution) in number of instruction executions

Label		Description
		Number of committed "load/store" instruction (NOSIMD execution) / Number of instruction executions * 100 Number of committed "load/store" instruction (NOSIMD execution) : load_store_instructions Number of instruction executions : effective_instruction_counts
	AVE.	Average value between ranks in Application View Average value between threads in Rank View
	MIN	Minimum value between ranks in Application View Minimum value between threads in Rank View
	MAX	Maximum value between ranks in Application View Maximum value between threads in Rank View
Floating operand (NOSIMD) (%)		Rate of floating-point instructions (NOSIMD execution) that occupies it to the number of instruction executions Number of committed "floating-point" instructions (NOSIMD execution) / Number of instruction executions * 100 Number of committed "floating-point" instructions (NOSIMD execution) : floating_instructions Number of instruction executions : effective_instruction_counts
	AVE.	Average value between ranks in Application View Average value between threads in Rank View
	MIN	Minimum value between ranks in Application View Minimum value between threads in Rank View
	MAX	Maximum value between ranks in Application View Maximum value between threads in Rank View
FMA (NOSIMD) (%)		Rate of floating-point multiply and add/sub and floating-point trigonometric instructions (NOSIMD execution) in number of instruction executions Number of committed "floating-point multiply and add/sub" and "floating-point trigonometric" instructions / Number of instruction executions * 100 Number of committed "floating-point multiply and add/sub" and "floating-point trigonometric" instructions : fma_instructions Number of instruction executions : effective_instruction_counts
	AVE.	Average value between ranks in Application View Average value between threads in Rank View
	MIN	Minimum value between ranks in Application View Minimum value between threads in Rank View
	MAX	Maximum value between ranks in Application View Maximum value between threads in Rank View
Fixed operand (NOSIMD) (%)		Rate of fixed point partitioned add/sub and integer multiply-add instruction (NOSIMD execution) in number of instruction executions Number of committed "fixed point partitioned add/sub and integer multiply-add" instruction (NOSIMD execution) / Number of instruction executions * 100 Number of committed "fixed point partitioned add/sub and integer multiply-add" instruction (NOSIMD execution) : fixed_point_instructions Number of instruction executions : effective_instruction_counts

Label		Description
	AVE.	Average value between ranks in Application View Average value between threads in Rank View
	MIN	Minimum value between ranks in Application View Minimum value between threads in Rank View
	MAX	Maximum value between ranks in Application View Maximum value between threads in Rank View
NOSIMD (%)		<p>Rate of NOSIMD instructions in number of instruction executions</p> $\frac{\text{Number of committed "floating-point loading store" instructions} + \text{Number of committed "floating-point" instructions} + \text{Number of committed "floating-point multiply and add/sub" and "floating-point trigonometric" instructions} + \text{Number of committed "fixed point partitioned add/sub and integer multiply-add" instructions}}{\text{Number of instruction executions}} * 100$ <p>Number of committed "floating-point loading store" instructions : <code>load_store_instructions</code></p> <p>Number of committed "floating-point" instructions : <code>floating_instructions</code></p> <p>Number of committed "floating-point multiply and add/sub" and "floating-point trigonometric" instructions : <code>fma_instructions</code></p> <p>Number of committed "fixed point partitioned add/sub and integer multiply-add" instruction (NOSIMD execution) : <code>fixed_point_instructions</code></p> <p>Number of instruction executions : <code>effective_instruction_counts</code></p>
	AVE.	Average value between ranks in Application View Average value between threads in Rank View
	MIN	Minimum value between ranks in Application View Minimum value between threads in Rank View
	MAX	Maximum value between ranks in Application View Maximum value between threads in Rank View
NOSIMD Load/Store (%)		<p>Rate of SIMD and 4SIMD floating-point loading store instructions that occupies it to the number of floating-point loading store instructions</p> $\frac{\text{Number of committed "load/store" instructions}}{\text{Number of committed "SIMD and 4SIMD load/store" instructions} + \text{Number of committed "load/store" instructions}} * 100$ <p>Number of committed "load/store" instructions : <code>load_store_instructions</code></p> <p>Number of committed "SIMD and 4SIMD load/store" instructions : <code>XSIMD_load_store_instructions</code></p>
	AVE.	Average value between ranks in Application View Average value between threads in Rank View
	MIN	Minimum value between ranks in Application View Minimum value between threads in Rank View
	MAX	Maximum value between ranks in Application View Maximum value between threads in Rank View
Prefetch		<p>Number of prefetch instructions</p> <p>Number of committed "prefetch" instructions + Number of committed "NonSIMD, SIMD and 4SIMD of indirect prefetch" instructions</p> <p>Number of committed "prefetch" instructions : <code>prefetch_instructions</code></p>

Label		Description
		Number of committed "NonSIMD, SIMD and 4SIMD of indirect prefetch" instructions : nonSIMD_XSIMD_indirect_prefetch_instructions
	AVE.	Average value between ranks in Application View Average value between threads in Rank View
	MIN	Minimum value between ranks in Application View Minimum value between threads in Rank View
	MAX	Maximum value between ranks in Application View Maximum value between threads in Rank View
Indirect Prefetch		Number of committed "NonSIMD, SIMD and 4SIMD of indirect prefetch" instructions Number of committed "NonSIMD, SIMD and 4SIMD of indirect prefetch" instructions : nonSIMD_XSIMD_indirect_prefetch_instructions
	AVE.	Average value between ranks in Application View Average value between threads in Rank View
	MIN	Minimum value between ranks in Application View Minimum value between threads in Rank View
	MAX	Maximum value between ranks in Application View Maximum value between threads in Rank View

4. MEM_access

The MEM_access event is used to check the execution status of instructions accessing the data transfer efficiency and the memory between CPU core and the memory.

Memory access throughput large enough, data transfer among many of the between memory and CPU core.

Table 3.15 Items of MEM_access

Label		Description
Name		Measurement section name
Number		Measurement section group number
Elapsed (S)		Elapsed time
	AVE.	Average value between ranks in Application View Average value between threads in Rank View
	MIN	Minimum value between ranks in Application View Minimum value between threads in Rank View
	MAX	Maximum value between ranks in Application View Maximum value between threads in Rank View
Load/Store (SIMD) (%)		Rate of SIMD and 4SIMD load/store instruction number of instruction executions Number of committed "SIMD and 4SIMD load/store" instructions / Number of instruction executions * 100 Number of committed "SIMD and 4SIMD load/store" instructions : XSIMD_load_store_instructions Number of instruction executions : effective_instruction_counts
	AVE.	Average value between ranks in Application View Average value between threads in Rank View

Label		Description
	MIN	Minimum value between ranks in Application View Minimum value between threads in Rank View
	MAX	Maximum value between ranks in Application View Maximum value between threads in Rank View
Load/Store (NOSIMD) (%)		Rate of load/store instruction number of instruction executions (NOSIMD execution) in number of instruction executions Number of committed "load/store" instruction (NOSIMD execution) / Number of instruction executions * 100 Number of committed "load/store" instruction (NOSIMD execution) : load_store_instructions Number of instruction executions : effective_instruction_counts
	AVE.	Average value between ranks in Application View Average value between threads in Rank View
	MIN	Minimum value between ranks in Application View Minimum value between threads in Rank View
	MAX	Maximum value between ranks in Application View Maximum value between threads in Rank View
Memory access throughput (core) (GB/s)		Average of the amount of data transfer per second between memory and CPU core Amount of data transfer between memory and CPU core * 256 / Elapsed time / 1.0e+9 Amount of data transfer between memory and CPU core : L2_miss_dm + L2_miss_pf + L2_wb_dm + L2_wb_pf
	AVE.	Average value between ranks in Application View Average value between threads in Rank View
	MIN	Minimum value between ranks in Application View Minimum value between threads in Rank View
	MAX	Maximum value between ranks in Application View Maximum value between threads in Rank View
Write back (Ratio)		Rate of writeback in memory access Memory CPU core writing data amount / Amount of data transfer between memory CPU cores * 100 Memory CPU core writing data amount : L2_wb_dm + L2_wb_pf Amount of data transfer between memory CPU cores : L2_miss_dm + L2_miss_pf + L2_wb_dm + L2_wb_pf
	AVE.	Average value between ranks in Application View Average value between threads in Rank View
	MIN	Minimum value between ranks in Application View Minimum value between threads in Rank View
	MAX	Maximum value between ranks in Application View Maximum value between threads in Rank View
Memory access throughput (peak) (core)(GB/s)		Memory access throughput peak value of each core - When the write backing rate is smaller than 50% 240 + 240 * ((Memory CPU core writing data amount / Amount of data transfer between

Label		Description
		<p>memory CPU cores) / (1 - Memory CPU core writing data amount / Amount of data transfer between memory CPU cores))</p> <p>- When the write backing rate is 50% or more $240 + 240 * ((1 - \text{Memory CPU core writing data amount} / \text{Amount of data transfer between memory CPU cores}) / (\text{Memory CPU core writing data amount} / \text{Amount of data transfer between memory CPU cores}))$</p> <p>Memory CPU core writing data amount : L2_wb_dm + L2_wb_pf Amount of data transfer between memory CPU cores : L2_miss_dm + L2_miss_pf + L2_wb_dm + L2_wb_pf</p>
	AVE.	<p>Average value between ranks in Application View</p> <p>Average value between threads in Rank View</p>
	MIN	<p>Minimum value between ranks in Application View</p> <p>Minimum value between threads in Rank View</p>
	MAX	<p>Maximum value between ranks in Application View</p> <p>Maximum value between threads in Rank View</p>
Memory access throughput / PEAK (core) (%)		<p>Rate of memory access throughput peak value in memory access throughput</p> <p>Memory access throughput (core) / memory access throughput (peak)(core) * 100</p>
	AVE.	<p>Average value between ranks in Application View</p> <p>Average value between threads in Rank View</p>
	MIN	<p>Minimum value between ranks in Application View</p> <p>Minimum value between threads in Rank View</p>
	MAX	<p>Maximum value between ranks in Application View</p> <p>Maximum value between threads in Rank View</p>

5. Performance

The Performance event is used to check the ratio at the time that is executed and waited for with the instruction row.

Moreover, the rate in which two or more instructions are executed in parallel and the rate at the execution waiting time according to the cause can be confirmed.

Execution efficiency can be improved wait time is small and 2-4 instruction commit is greater.

Table 3.16 Items of Performance

Label		Description
Name		Measurement section name
Number		Measurement section name
Elapsed (S)		Measurement section group number
	AVE.	<p>Average value between ranks in Application View</p> <p>Average value between threads in Rank View</p>
	MIN	<p>Minimum value between ranks in Application View</p> <p>Minimum value between threads in Rank View</p>
	MAX	<p>Maximum value between ranks in Application View</p> <p>Maximum value between threads in Rank View</p>

Label	Description
2-4 instruction commit (S)	<p>Time when number of completion instructions is two or more</p> <p>(CPU cycle number - (Number of cycles where no instructions are committed + Number of cycles where 1 instruction is committed)) / Frequency in CPU core</p> <p>CPU cycle number : cycle_counts</p> <p>Number of cycles where no instructions are committed : 0endop</p> <p>Number of cycles where 1 instruction is committed : 1endop</p>
AVE.	<p>Average value between ranks in Application View</p> <p>Average value between threads in Rank View</p>
MIN	<p>Minimum value between ranks in Application View</p> <p>Minimum value between threads in Rank View</p>
MAX	<p>Maximum value between ranks in Application View</p> <p>Maximum value between threads in Rank View</p>
1 instruction commit (S)	<p>Time when number of completion instructions is one</p> <p>Number of cycles where 1 instruction is committed / Frequency in CPU core</p> <p>Number of cycles where 1 instruction is committed : 1endop</p>
AVE.	<p>Average value between ranks in Application View</p> <p>Average value between threads in Rank View</p>
MIN	<p>Minimum value between ranks in Application View</p> <p>Minimum value between threads in Rank View</p>
MAX	<p>Maximum value between ranks in Application View</p> <p>Maximum value between threads in Rank View</p>
Operation wait (S)	<p>Oldest instruction in the instruction when being executing it is time when the operation of the floating-point number and the number of completion instructions are 0</p> <p>Number of cycles where no instructions are committed and the oldest / Frequency in CPU core</p> <p>Number of cycles where no instructions are committed and the oldest : eu_comp_wait</p>
AVE.	<p>Average value between ranks in Application View</p> <p>Average value between threads in Rank View</p>
MIN	<p>Minimum value between ranks in Application View</p> <p>Minimum value between threads in Rank View</p>
MAX	<p>Maximum value between ranks in Application View</p> <p>Maximum value between threads in Rank View</p>
Cache access wait (S)	<p>Time according to the cache access waiting when the number of completion instructions is 0</p> <p>(Number of cycles where no instructions are committed because the oldest - Number of cycle where waiting by level 2 cache miss) / Frequency in CPU core</p> <p>Number of cycles where no instructions are committed because the oldest : op_stv_wait</p> <p>Number of cycle where waiting by level 2 cache miss : op_stv_wait_sxmiss</p>
AVE.	<p>Average value between ranks in Application View</p> <p>Average value between threads in Rank View</p>
MIN	<p>Minimum value between ranks in Application View</p> <p>Minimum value between threads in Rank View</p>

Label		Description
	MAX	Maximum value between ranks in Application View Maximum value between threads in Rank View
Memory access wait (S)		Within the range of the measurement, time according to the memory access waiting when the number of completion instructions is 0 CSE is a buffer to hold the information on the instruction currently executing (it is issued, but not completed yet). (Number of cycle where waiting by level 2 cache miss + Number of cycles where no instructions are committed because the CSE is empty) / Frequency in CPU core (Number of cycle where waiting by level 2 cache miss : op_stv_wait_sxmiss Number of cycles where no instructions are committed because the CSE is empty : cse_window_empty_sp_full
	AVE.	Average value between ranks in Application View Average value between threads in Rank View
	MIN	Minimum value between ranks in Application View Minimum value between threads in Rank View
	MAX	Maximum value between ranks in Application View Maximum value between threads in Rank View
Fetch wait (S)		Time when number of completion instructions is 0 because CSE is empty CSE is a buffer to hold the information on the instruction currently executing (it is issued, but not completed yet). (Number of cycles where no instructions are committed because the CSE is empty - Number of cycles where no instructions are committed because the CSE is empty and the store ports are full) / Frequency in CPU core Number of cycles where no instructions are committed because the CSE is empty : cse_window_empty Number of cycles where no instructions are committed because the CSE is empty and the store ports are full : cse_window_empty_sp_full
	AVE.	Average value between ranks in Application View Average value between threads in Rank View
	MIN	Minimum value between ranks in Application View Minimum value between threads in Rank View
	MAX	Maximum value between ranks in Application View Maximum value between threads in Rank View
Other wait (S)		Time when number of completion instructions is 0 for reasons other than the above Time when number of cycles where no instructions are committed - (Operation waiting time + Cache access waiting time + Memory access waiting time + Instruction fetch waiting time) Time when number of cycles where no instructions are committed : 0endop / Frequency in CPU core
	AVE.	Average value between ranks in Application View Average value between threads in Rank View
	MIN	Minimum value between ranks in Application View Minimum value between threads in Rank View

Label	Description
MAX	Maximum value between ranks in Application View Maximum value between threads in Rank View

6. Statistics

The Statistics event is used to check the execution efficiency.

High performance as the MIPS and FLOPS is close to peak value.

Table 3.17 Items of Statistics

Label	Description
Name	Measurement section name
Number	Measurement section group number
Elapsed (S)	Elapsed time
AVE.	Average value between ranks in Application View Average value between threads in Rank View
MIN	Minimum value between ranks in Application View Minimum value between threads in Rank View
MAX	Maximum value between ranks in Application View Maximum value between threads in Rank View
MFLOPS	Floating-point arithmetic execution efficiency (average of the number of floating-point arithmetic executions per second) (Number of operations is obtained by multiplying by 1 + 2 * Number of operations is obtained by multiplying by 2 + 4 * Number of operations is obtained by multiplying by 4 + 8 * Number of operations is obtained by multiplying by 8 + 16 * Number of operations is obtained by multiplying by 16) / Elapsed time / 1.0e+6 Number of operations is obtained by multiplying by 1 : 1FLOPS_instructions Number of operations is obtained by multiplying by 2 : 2FLOPS_instructions Number of operations is obtained by multiplying by 4 : 4FLOPS_instructions Number of operations is obtained by multiplying by 8 : 8FLOPS_instructions Number of operations is obtained by multiplying by 16 : 16FLOPS_instructions
AVE.	Average value between ranks in Application View Average value between threads in Rank View
MIN	Minimum value between ranks in Application View Minimum value between threads in Rank View
MAX	Maximum value between ranks in Application View Maximum value between threads in Rank View
MFLOPS/PEAK (%)	Rate of actual measurement values to the logical peak value of MFLOPS $MFLOPS / (\text{Number of execution core} * MFLOPS \text{ peak value of each core}) * 100$
AVE.	Average value between ranks in Application View Average value between threads in Rank View
MIN	Minimum value between ranks in Application View Minimum value between threads in Rank View
MAX	Maximum value between ranks in Application View

Label		Description
		Maximum value between threads in Rank View
MIPS		Instruction execution efficiency (average of the number of instruction executions per second) Number of instruction executions / Elapsed time / 1.0e+6 Number of instruction executions : effective_instruction_counts
	AVE.	Average value between ranks in Application View Average value between threads in Rank View
	MIN	Minimum value between ranks in Application View Minimum value between threads in Rank View
	MAX	Maximum value between ranks in Application View Maximum value between threads in Rank View
MIPS/PEAK (%)		Rate of actual measurement values to the logical peak value of MIPS MIPS / (Number of execution cores * MIPS peak value of each core) * 100
	AVE.	Average value between ranks in Application View Average value between threads in Rank View
	MIN	Minimum value between ranks in Application View Minimum value between threads in Rank View
	MAX	Maximum value between ranks in Application View Maximum value between threads in Rank View
Floating point operations (%)		Rate of the number of floating-point arithmetic executions in operand of instruction (Number of operations is obtained by multiplying by 1 + 2 * Number of operations is obtained by multiplying by 2 + 4 * Number of operations is obtained by multiplying by 4 + 8 * Number of operations is obtained by multiplying by 8 + 16 * Number of operations is obtained by multiplying by 16) / (Number of instruction executions + Number of operations is obtained by multiplying by 2 + 3 * Number of operations is obtained by multiplying by 4 + 7 * Number of operations is obtained by multiplying by 8 + 15 * Number of operations is obtained by multiplying by 16) * 100 Number of operations is obtained by multiplying by 1 : 1FLOPS_instructions Number of operations is obtained by multiplying by 2 : 2FLOPS_instructions Number of operations is obtained by multiplying by 4 : 4FLOPS_instructions Number of operations is obtained by multiplying by 8 : 8FLOPS_instructions Number of operations is obtained by multiplying by 16 : 16FLOPS_instructions Number of instruction executions : effective_instruction_counts
	AVE.	Average value between ranks in Application View Average value between threads in Rank View
	MIN	Minimum value between ranks in Application View Minimum value between threads in Rank View
	MAX	Maximum value between ranks in Application View Maximum value between threads in Rank View

7. TLB

The TLB event is used to check the TLB miss at the data transfer between memory and CPU core.

Table 3.18 Items of TLB

Label	Description	
Name	Measurement section name	
Number	Measurement section group number	
Elapsed (S)	Elapsed time	
AVE.	Average value between ranks in Application View Average value between threads in Rank View	
	MIN	Minimum value between ranks in Application View Minimum value between threads in Rank View
	MAX	Maximum value between ranks in Application View Maximum value between threads in Rank View
Instructions	Executed number of instructions (number of instruction executions) Number of instruction executions : effective_instruction_counts	
AVE.	Average value between ranks in Application View Average value between threads in Rank View	
	MIN	Minimum value between ranks in Application View Minimum value between threads in Rank View
	MAX	Maximum value between ranks in Application View Maximum value between threads in Rank View
uDTLB miss (%)	Rate of data uTLB misses Number of uTLB misses / Number of memory access instructions * 100 Number of uTLB misses : uDTLB_miss Number of memory access instructions : load_store_instructions + 2 * SIMD_load_store_instructions + 4 * 4SIMD_load_store_instructions	
AVE.	Average value between ranks in Application View Average value between threads in Rank View	
	MIN	Minimum value between ranks in Application View Minimum value between threads in Rank View
	MAX	Maximum value between ranks in Application View Maximum value between threads in Rank View
DTLB miss (%)	Rate of data DTLB misses Number of DTLB write via the Hardware Tablealk caused by a data access DTLB miss / Number of memory access instructions * 100 Number of DTLB write via the Hardware Tablealk caused by a data access DTLB miss : DTLB_write Number of memory access instructions : load_store_instructions + 2 * SIMD_load_store_instructions + 4 * 4SIMD_load_store_instructions	
AVE.	Average value between ranks in Application View Average value between threads in Rank View	
	MIN	Minimum value between ranks in Application View Minimum value between threads in Rank View

Label		Description
	MAX	Maximum value between ranks in Application View Maximum value between threads in Rank View

(2)Whole graph

Distribution information between parallels on the item selected by the Advanced Profiler information list is displayed in the whole graph by the graph form.

A range selection frame is displayed when the cursor is moved to the whole graph area, and information that corresponds to the range selection frame is displayed in the zoom information panel.

The range selection frame is a fixed size that can be displayed in the zoom information panel for parallel numbers.

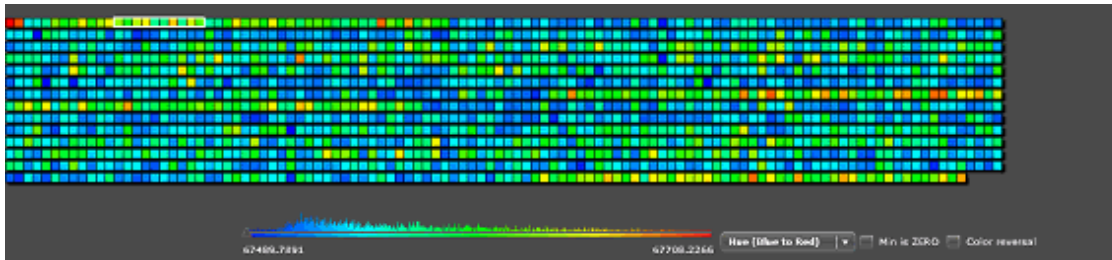
The range selection position can be fixed or released by clicking on the whole graph.

When all parallel numbers are installed on the zoom information panel, the range selection frame is not displayed.

Shape in the selection border for the color and three dimension shape used for the whole graph can be changed by the color histogram.

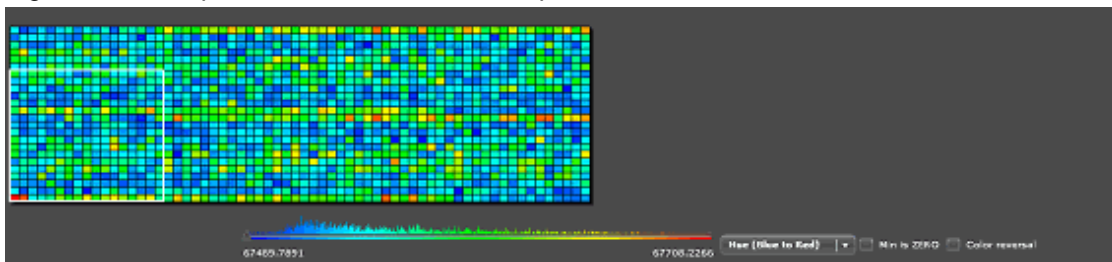
The example of a whole graph is shown as follows. A white frame in figure is a range selection frame.

Figure 3.8 Example of one-dimensional shape/Panel information



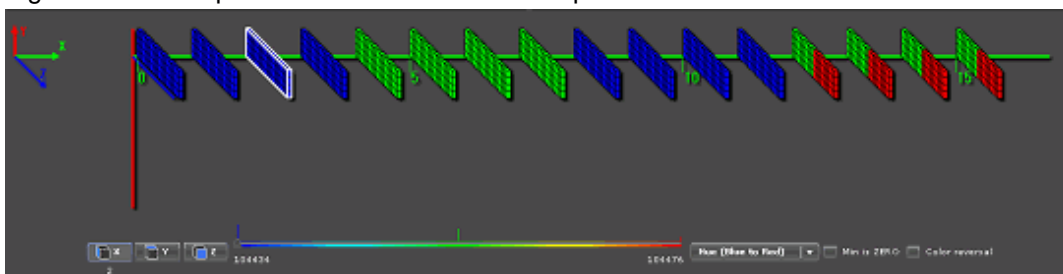
In case of a one-dimensional shape and the Panel information, a parallel unit is arranged from the left to the right. It moves to the next line if it is not possible to display in one line.

Figure 3.9 Example of a two-dimensional shape



In case of a two-dimensional shape, the bottom left is the starting point (0,0).

Figure 3.10 Example of a three-dimensional shape



In case of a three-dimensional shape, the intersection of the vertical Y axis (red axis line), the diagonal Z axis (blue axis line), and the horizontal X axis (green axis line) is the starting point.

(3)Color histogram

The color histogram shows the relation between the color and the value used for the whole graph and the graph in the zoom information panel. Note that the occurrence rate of a value is shown according to the height of the color bar chart.

The color histogram comprises the following elements:

1. Range selection frame buttons

Using the range selection frame buttons and layout switch buttons, the layout of the figure (only three-dimensional shapes) displayed in the zoom information panel and the selection border of the range selection frame can be changed.

The range selection frame button is displayed when displayed by three dimension shape. The range of the selection of the range selection frame is revocable because of the button that has been selected. Moreover, the coordinates location information about the dimension on the axis selected in Whole graph is displayed under the button that has been selected.

Table 3.19 Range selection frame

Button	Selected range
X	X axis is fixed. Aspect on the same X axis can be selected.
Y	Y axis is fixed. Aspect on the same Y axis can be selected.
Z	Z axis is fixed. Aspect on the same Z axis can be selected.

2. Histogram

It displays a histogram. The occurrence rate of a value and the relation between the value and the color is shown.

3. Color mode box

The color mode of the color histogram can be changed. The color modes are described below.

When the Advanced Profiler is started, "Hue (Blue to Red)" is selected by default.

Table 3.20 Color modes

Label	Description
Hue (Blue to Red)	The cost is allocated from blue to red in 256 colors. By default, the color becomes red from blue as the cost rises from low-cost.
Tone (Blue to Red)	The cost is allocated blue, white, and red tones. By default, the color becomes blue, white, and red as the cost rises from low-cost.
4 colors	The cost is allocated four colors (blue, green, yellow, and red). By default, the color becomes blue, green, yellow, and red as the cost rises from low-cost. The color allocation threshold can be changed by the operation of the thumb. The default position of thumb is as follows: First thumb: Center of the average value and the minimum value Second thumb: Average value Third thumb: Center of the average value and the maximum value

4. Color histogram controlled check boxes

The minimum value of the histogram and the color order can be changed.

Table 3.21 Check boxes

Label	Description
MIN is ZERO	Uses 0 as the minimum value for the color histogram
Color reversal	Reverses the order of the color histogram color scheme

(4)Zoom information panel

In the zoom information panel, information within the range selected in the whole graph is expanded and displayed. A white frame to select one unit is displayed when the cursor is moved to the zoom information panel. The cost information on the selected unit is displayed in a table in the panel. The displayed item is an item displayed in the Advanced Profiler information list. The range selection position can

be fixed or released by clicking in the zoom information panel. The unit that has been selected in the zoom information panel is the box where the rank of **Rank View** and **Thread View** is selected and the box and where the thread is selected.

3.3.3.3.2 Bar Chart information

In the Bar Chart information, the distribution between parallel units of the cost displayed in the cost information is displayed as a bar chart. The displayed items and the operation methods are the same as the Topology or Panel information.

However, there is no distribution information selected when the initial state of Rank view is displayed.

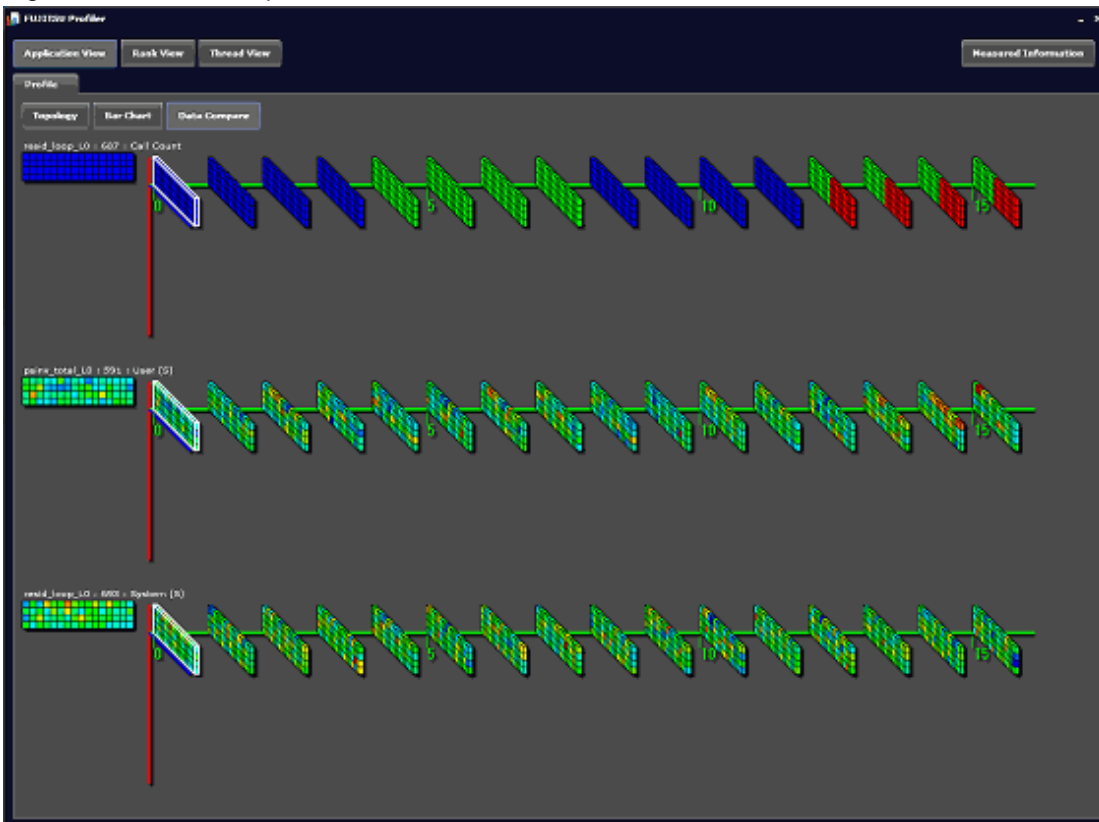
3.3.3.3.3 Data Compare information

In the Data Compare information, the whole graph of three times (for three items) of displayed by selecting the row in the Topology, Panel and Bar Chart is arranged vertically and displayed.

The graph of the same item with a different form is updated without being added.

When the cursor is moved to the whole graph, the range selection frame is displayed, and the information that corresponds to the selection frame is extracted to the left and displayed.

Figure 3.11 Data Compare information window



3.4 Advanced Profiler information (text/CSV formats)

This section describes the Advanced Profiler information (text/CSV formats) output by the fapppx command.

3.4.1 Overview of the Advanced Profiler feature

The Advanced Profiler information comprises the following information.

The output of each information can be controlled by using options of the fapppx command.

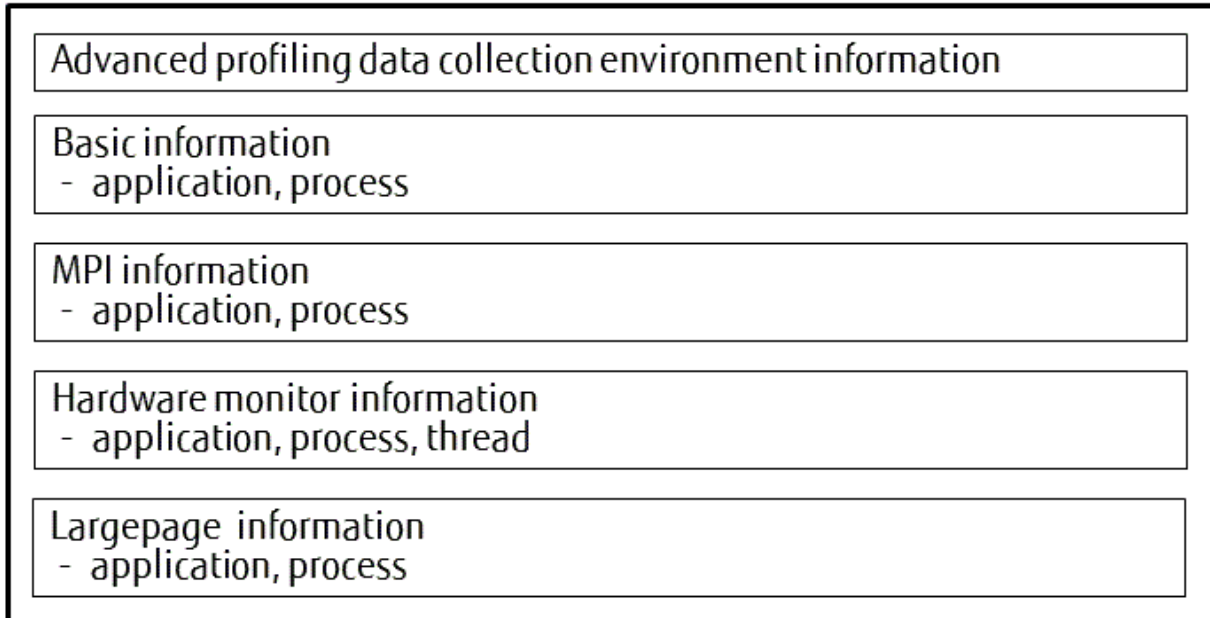
Refer to "[3.2.6 fapppx command](#)" for information on the options that control information.

- Environment information for advanced profiling data collection

- Basic information
- MPI information
- Hardware monitor information
- Largepage information

The following figure shows the composition of the Advanced Profiler information in the text format.

Figure 3.12 Composition of the Advanced Profiler information in the text format



3.4.2 Environment information for advanced profiling data collection

The environment information outputs the details of the execution environment where the advanced profiling data is collected.

Figure 3.13 Output format of the environment information for advanced profiling data collection

```

-----
Fujitsu Advanced Performance Profiler Version @vl
Measured time           : @date
CPU frequency           : @pno @frequency (MHz)
Type of program         : @type @thno
Virtual coordinate      : (@x, @y, @z)
-----

```

Table 3.22 Output items of the environment information for advanced profiling data collection

Output item	Description
@vl	Version level
@date	Date for advanced profiling data collection
@pno	Process number
@frequency	Frequency of execution processor
@type	Execution format of an application SERIAL : Sequential Thread(AUTO) : Automatic parallel Thread(OpenMP) : OpenMP MPI : MPI

Output item	Description
	MPI(AUTO) : MPI + Automatic parallel MPI(OpenMP) : MPI + OpenMP
@thno	Thread number (displayed in case of parallel threads)
@x, @y, @z	Logical shape of an MPI application on execution (displayed in case of MPIs)

3.4.3 Basic information

The average, minimum, and maximum values for the call count, the elapsed time, the user CPU time, and the system CPU time of each measurement section is output in the basic information.

Figure 3.14 Output format of the basic information

```

Basic profile
*****
Application
*****
(1)
-----
Kind      Elapsed(s)   User(s)      System(s)    Call
-----
AVG       @elapse      @user        @sys         @call  @name @no
MAX       @elapse      @user        @sys         @call
MIN       @elapse      @user        @sys         @call
(2)
*****
Process @pno
*****
-----
Elapsed(s)   User(s)      System(s)    Call
-----
(1) @elapse      @user        @sys         @call  @name @no

```

(1) The amount of @name and @no is repeated.
(2) The amount of @pno is repeated.

Table 3.23 Output items of the basic information

Output item	Description
@elapse	Elapsed time (s)
@user	User CPU time (s)
@sys	System CPU time (s)
@call	Call count
@name	Group name
@no	Advanced number
@pno	Process number

3.4.4 MPI information

The average, minimum, and maximum values for the call count, the message length, the elapsed time, and the wait time are output in the MPI information.

3.4.4.1 Formulas of the message length

The formulas of the message length of each MPI subroutine and the MPI function are shown in the table below:

Table 3.24 Formulas of the message length

MPI subroutine/function	Formula
MPI_SEND MPI_BSEND MPI_SSEND MPI_RSEND MPI_ISEND MPI_IBSEND MPI_SSEND MPI_IRSEND	Number of elements in the send buffer * Size of data type in the each send buffer element
MPI_RECV	Received number of elements * Size of data type in the each receive buffer element
MPI_IRECV	Number of elements in the receive buffer * Size of data type in the each receive buffer element
MPI_SENDRECV	(Number of elements in the send buffer * Size of type of elements in the send buffer) + (Number of elements in the receive buffer * Size of type of elements in the receive buffer)
MPI_SENDRECV_REPLACE	(Number of elements in the send and receive buffer * Size of type of elements in the send and receive buffer *2)
MPI_BCAST MPI_IBCAST	- For the root process Number of elements in the buffer * Size of data type in the buffer * 2 - Except the root process Number of elements in the buffer * Size of data type in the buffer
MPI_GATHER MPI_IGATHER	- For the root process (Number of elements in the send buffer * Size of data type in the send buffer elements) + (Number of total processes * Number of elements for any single receive * Size of data type in the receive buffer elements) - Except the root process Number of elements in the send buffer * Size of data type in the send buffer elements
MPI_GATHERV MPI_IGATHERV	- For the root process (Number of elements in the send buffer * Size of data type in the send buffer elements) + (Containing the number of elements that are received from each process * Size of data type in the receive buffer elements) - Except the root process Number of elements in the send buffer * Size of data type in the send buffer elements
MPI_SCATTER MPI_ISCATTER	- For the root process (Number of total processes * Number of elements sent to each process * Size of data type in the send buffer elements) + (Number of elements in the receive buffer * Size of data type in the receive buffer elements) - Except the root process Number of elements in the receive buffer * Size of data type in the receive buffer elements
MPI_SCATTERV MPI_ISCATTERV	- For the root process (Specifying the number of elements to send to each process * Size of data type in the send buffer elements) + (Number of elements in the receive buffer * Size of data type in the receive buffer elements)

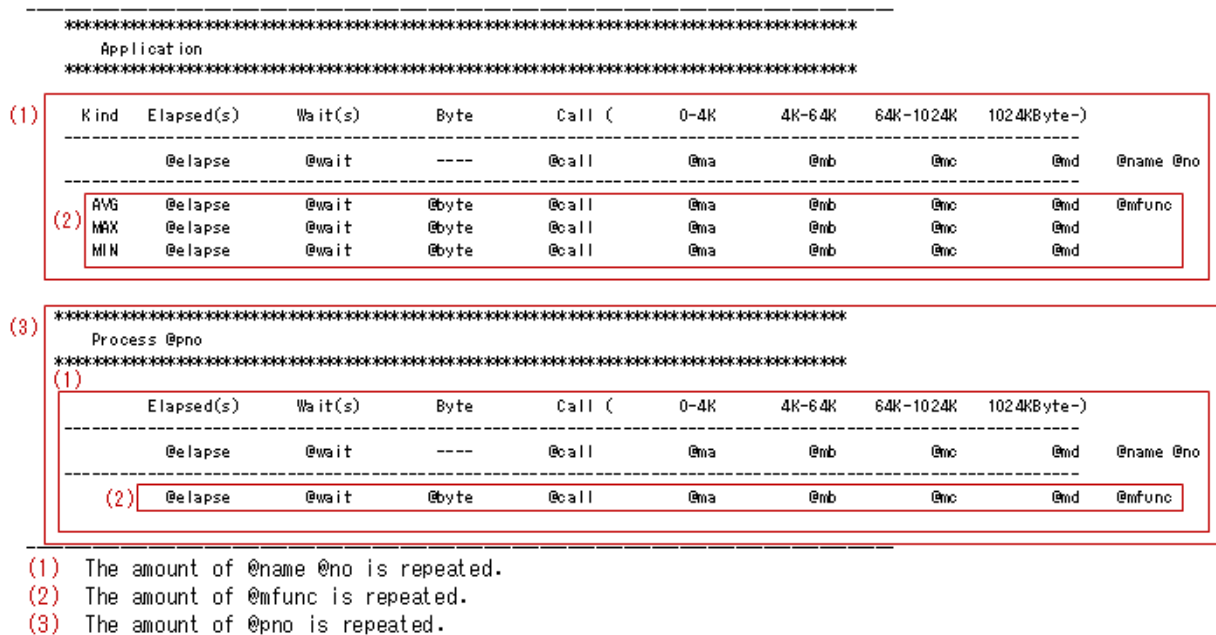
MPI subroutine/function	Formula
	- Except the root process Number of elements in the receive buffer * Size of data type in the receive buffer elements
MPI_ALLGATHER MPI_IALLGATHER	(Number of elements in the send buffer * Size of data type in the send buffer elements) + (Number of total processes * Number of elements in the receive buffer * Size of data type in the receive buffer elements)
MPI_ALLGATHERV MPI_IALLGATHERV	(Number of elements in the send buffer * Size of data type in the send buffer elements) + (Containing the number of elements that are received from each process * Size of data type in the receive buffer elements)
MPI_ALLTOALL MPI_IALLTOALL	(Number of total processes * Number of elements send to each process * Size of data type in the send buffer elements) + (Number of total processes * Number of elements received from one process * Size of data type in the receive buffer elements)
MPI_ALLTOALLV MPI_IALLTOALLV	(Total of number of elements in the data send to each process * Size of data type in the send buffer elements) + (Total of element in the data received from each process * Size of data type in the receive buffer elements)
MPI_REDUCE MPI_IREDUCE	- For the root process Number of elements in the send buffer * Size of data type in the send buffer elements * 2 - Except the root process Number of elements in the send buffer * Size of data type in the send buffer elements
MPI_ALLREDUCE MPI_IALLREDUCE	Number of elements in the send buffer * Size of data type in the send buffer elements * 2
MPI_REDUCE_SCATTER MPI_IREDUCE_SCATTER	(Total of number of elements in the data send to each process * Size of data type in the buffer elements) + (Number of receive elements * Size of data type in the buffer elements)
MPI_SCAN MPI_ISCAN	Number of elements in the input buffer * Size of data type in the input buffer * 2
MPI_PUT MPI_RPUT	Number of elements in the origin buffer * Size of data type in the origin buffer
MPI_GET MPI_RGET	Number of elements in the target buffer * Size of data type in the target buffer
MPI_ACCUMULATE MPI_RACCUMULATE	Number of elements in the origin buffer * Size of data type in the origin buffer
MPI_ALLTOALLW MPI_IALLTOALLW	(Number of elements of send buffers in the each process * Size of data type of send buffer in the each process) + (Number of elements of receive buffers in the each process * Size of data type of receive buffer in the each process)
MPI_EXSCAN MPI_IEXSCAN	Number of elements in the input buffer * Size of data type in the input buffer * 2
MPI_REDUCE_SCATTER_BLOCK MPI_IREDUCE_SCATTER_BLOCK	(Size of data type in the buffer elements * Number of total processes * Number of element per block) + (Size of data type in the buffer elements * Number of elements per block)
MPI_MRECV MPI_IMRECV	Size of data type in the each receive buffer element * Number of elements in receive buffer
MPI_COMPARE_AND_SWAP	Size of data type of elements in all buffers * 2
MPI_GET_ACCUMULATE MPI_RGET_ACCUMULATE	(Size of each entry in origin buffer * Number of entries in origin buffer)+(Size of each entry in target buffer * Number of entries in target buffer)

MPI subroutine/function	Formula
MPI_NEIGHBOR_ALLGATHER MPI_INEIGHBOR_ALLGATHER	(Number of elements in the send buffer * Size of data type in the send buffer elements)+ (Number of a edges becoming the input * Number of elements in the receive buffer * Size of data type in the receive buffer elements)
MPI_NEIGHBOR_ALLGATHERV MPI_INEIGHBOR_ALLGATHERV	(Number of elements in the send buffer * Size of data type in the send buffer elements)+ (Total of elements in the receive buffer of a edges becoming each input * Size of data type in the receive buffer elements)
MPI_NEIGHBOR_ALLTOALL MPI_INEIGHBOR_ALLTOALL	(Number of a edges becoming the output * Number of elements in the send buffer * Size of data type in the send buffer elements)+(Number of a edges becoming the input * Number of elements in the receive buffer * Size of data type in the receive buffer elements)
MPI_NEIGHBOR_ALLTOALLV MPI_INEIGHBOR_ALLTOALLV	(Total of elements in the receive buffer of a edges becoming each output * Size of data type in the send buffer elements)+(Total of elements in the receive buffer of a edges becoming each input * Size of data type in the receive buffer elements)
MPI_NEIGHBOR_ALLTOALLW MPI_INEIGHBOR_ALLTOALLW	(Total of elements in the receive buffer of a edges becoming each output * Size of data type in the send buffer elements of a edges becoming each output)+(Total of elements in the receive buffer of a edges becoming each input * Size of data type in the receive buffer elements of a edges becoming each input)

3.4.4.2 Output format of the MPI information

The output format of the MPI information is shown in the figure below.

Figure 3.15 Output format of the MPI information



The output items of the MPI information are shown in the table below.

Table 3.25 Output items of the MPI information

Output item	Description
@elapse	Elapsed time(s)
@wait	Wait time(s)
@byte	Average message length(Byte)
@call	Call count of an MPI function
@ma	Call count of MPI functions when the message length is more than 0 B but less than 4 KB

Output item	Description
@mb	Message length MPI functions when the message length is more than 4 KB but less than 64 KB
@mc	Call count of MPI functions when the message length is more than 64 KB but less than 1024 KB
@md	Call count of MPI functions when the message length is more than 1024 KB
@name	Group name
@no	Advanced number
@mfunc	MPI function name
@pno	Process number

3.4.5 Hardware monitor information

The Hardware monitor information is composed of two or more items.

Each item is classified into a group referred to as an event.

To collect specific information on the processor using the Advanced Profiler, the fapp command is executed specifying the event that contains the information to be measured.



Example

Example to the fapp command specifying an event

```
$ fapp -C -d FAPP_Example -Ihwm -Hevent=Statistics ./a.out
```

Refer to "3.2.5 fapp command" for information on how to collect the Hardware monitor information.

3.4.5.1 Events list

The following table describes the Hardware monitor information events.

Table 3.26 Hardware monitor information events

Event name	Measured information	Description
Cache	Elapsed time (s)	Elapsed time for executing instructions of the measured range
	Instructions	Executed number of instructions
	L1 instruction cache miss (%)	Level 1 instruction cache miss rate generated in memory access (loading store) instruction execution
	L1 operand cache miss (%)	Level 1 data cache miss rate generated in memory access (loading store) instruction execution
	L2 cache miss (%)	Level 2 cache miss rate generated in memory access (loading store) instruction execution
	L2 demand cache miss (%) (*2)	Rate of demand demand miss in level 2 cache miss
	L2 prefetch miss (%) (*3)	Rate of prefetch demand miss in level 2 cache miss
Instructions_SIMD	Elapsed time (s)	Elapsed time for executing instructions of the measured range
	Instructions	Executed number of instructions
	MIPS	Instruction execution efficiency (average of number of instructions executed per second)
	MIPS/PEAK (%)	Rate of actual measurement values to the logical peak value of MIPS

Event name	Measured information	Description
	Load/Store (SIMD) (%) (*1)	Rate of floating-point loading store SIMD or SIMD instruction that occupies it to the number of instruction executions
	Floating operand (SIMD) (%) (*1)	Rate of floating-point operation instructions (SIMD executions) that occupies it to the number of instruction executions
	FMA (SIMD) (%) (*1)	Rate of floating-point FMA instructions (SIMD execution) in number of instruction executions
	Fixed operand (SIMD) (%) (*1)	Rate of fixed-point operation instructions (SIMD executions) that occupies it to the number of instruction executions
	SIMD (%) (*1)	Rate of number of SIMD instructions in number of instruction executions
	SIMD load store instruction rate (%)	Rate of SIMD instruction that occupies it to load store instruction
	Prefetch	Number of prefetch instructions
	Indirect prefetch	Number of indirect prefetch instructions
Instructions_NOSIMD	Elapsed time (s)	Elapsed time for executing instructions of the measured range
	Instructions	Executed number of instructions
	MIPS	Instruction execution efficiency (average of number of instructions executed per second)
	MIPS/PEAK (%)	Rate of actual measurement values to the logical peak value of MIPS
	Load/Store (NOSIMD) (%) (*1)	Rate of floating-point operation instructions (NOSIMD execution) that occupies it to the number of instruction executions
	Floating operand (NOSIMD) (%) (*1)	Rate of floating-point operation instructions (NOSIMD execution) that occupies it to the number of instruction executions
	FMA (NOSIMD) (%) (*1)	Rate of floating-point FMA instructions (NOSIMD execution) in number of instruction executions
	Fixed operand (NOSIMD) (%) (*1)	Rate of fixed-point operation instructions (NOSIMD executions) that occupies it to the number of instruction executions
	NOSIMD (%) (*1)	Rate of NOSIMD instructions in number of instruction executions
	NOSIMD load store instruction rate (%)	Rate of NOSIMD instruction that occupies it to load store instruction
	Prefetch	Number of prefetch instructions
	Indirect prefetch	Number of indirect prefetch instructions
MEM_access	Elapsed time (s)	Elapsed time for executing instructions of the measured range
	Load/Store (SIMD) (%) (*1)	Rate of floating-point loading store SIMD or SIMD instructions that occupies it to the number of instruction executions

Event name	Measured information	Description
	Load/Store (NOSIMD) (%) (*1)	Rate of number of floating-point loading store instructions in number of instruction executions
	Memory access throughput (core) (GB/s)	Average data transfer between memory and CPU core per second
	Write back rate (%) (*5)	Rate of writing data amount to the data transfer between memory CPU cores
	Memory access throughput (PEAK) (*5)	Peak value of memory throughput calculated based on write backing rate
	Memory access throughput peak rate (%) (*5)	Rate of survey data to memory throughput peak value
Performance	Elapsed time (s)	Elapsed time for executing instructions of the measured range
	2-4 instruction commit	Time when number of completion instructions is two or more
	1 instruction commit	Time when number of completion instructions is one
	Operation wait	Oldest instruction in the instruction when being executing it is time when the operation of the floating-point number and the number of completion instructions are 0
	Cache access wait	Time according to the cache access waiting when the number of completion instructions is 0
	Memory access wait	Within the range of the measurement, time according to the memory access waiting when the number of completion instructions is 0
	Fetch wait	Time when number of completion instructions is 0 because CSE is empty (*4)
	Other wait	Time when number of completion instructions is 0 for reasons other than the above
Statistics	Elapsed time (s)	Elapsed time for executing instructions of the measured range
	MFLOPS	Floating-point arithmetic execution efficiency (average floating-point arithmetic executions per second)
	MFLOPS/PEAK (%)	Rate of actual measurement values to the logical peak value of MFLOPS
	MIPS	Instruction execution efficiency(average of number of instruction executions per second)
	MIPS/PEAK (%)	Rate of actual measurement values to the logical peak value of MIPS
	Floating operand	Rate of number of floating-point arithmetic execution in operand of instruction
TLB	Elapsed time (s)	Elapsed time for executing instructions of the measured range
	Instructions	Executed number of instructions
	uTLB miss (%)	Rate of the uTLB miss to the number of instruction executions
	mTLB miss (%)	Rate of the main TLB miss to the number of instruction executions

- *1 : SIMD instruction processes two or more operands in one instruction. NOSIMD instruction processes one operand in one instruction.
- *2 : The demand demand is to access the level 2 cache with the resources necessary for memory access was able to be acquired.
- *3 : The prefetch demand is the hardware prefetch state that cannot acquire the resources necessary for memory access, and it accesses the level 2 cache.
- *4 : CSE is a buffer to hold the information on the instruction currently executing (it is issued, but not completed yet).
- *5 : "Write back rate", "Memory access throughput (PEAK)", and "Memory access throughput peak rate" output only the unit of the application, and the power output of the unit of the process and each thread becomes 0.

3.4.5.2 Formulas of the Hardware monitor information

The following table describes the calculation formulas for events in the Hardware monitor information.

Table 3.27 Formulas of the Hardware monitor information (Cache)

Event name	Measured information (unit)	Formula
Cache	Elapsed time (s)	Elapsed time for executing instructions of the measured range
	Instructions	Executed number of instructions
	L1 instruction cache miss (%)	Level 1 instruction cache miss frequency / Executed number of instructions * 100
	L1 operand cache miss (%) (*1)	Level 1 data cache miss frequency / Number of memory access instructions * 100
	L2 cache miss (%) (*1)	Level 2 cache miss frequency / Number of memory access instructions * 100
	L2 demand cache miss (%)	Level 2 cache miss frequency of demand / Level 2 cache miss frequency * 100
	L2 prefetch miss (%)	Level 2 cache miss frequency of prefetch demand / Level 2 cache miss frequency * 100

*1 : Memory access instructions are calculated by the following expression:

Number of Integer or floating-point loading store instructions(load_store_instructions) + (Number of floating-point loading store SIMD instructions(SIMD_load_store_instructions) * 2) + (Number of floating-point loading store 4SIMD instructions(4SIMD_load_store_instructions) * 4)

The values in the calculation expressions are used from the following PA events.

- | | |
|----------------------------------|-----------------------------------|
| (1) effective_instruction_counts | (2) load_store_instructions |
| (3) SIMD_load_store_instructions | (4) 4SIMD_load_store_instructions |
| (5) L1I_miss | (6) L1D_miss |
| (7) L2_miss_dm | (8) L2_miss_pf |

- | | |
|---|-------------------------|
| Executed number of instructions | (1) |
| Number of memory access instructions | (2) + 2 * (3) + 4 * (4) |
| Level 1 instruction cache miss frequency | (5) |
| Level 1 data cache miss frequency | (6) |
| Level 2 cache miss frequency | (7) + (8) |
| Level 2 cache miss frequency of demand | (7) |
| Level 2 cache miss frequency of prefetch demand | (8) |

Table 3.28 Formulas of the Hardware monitor information (Instructions_SIMD)

Event name	Measured information (unit)	Formula
Instructions_SIMD	Elapsed time (s)	Elapsed time for executing instructions of the measured range
	Instructions	Executed number of instructions
	MIPS	Number of instruction executions / Elapsed time / 1.0e+6
	MIPS/PEAK (%)(*1)	MIPS / (Number of execution cores * MIPS peak value of each core) * 100
	Load/Store (SIMD) (%)	Number of floating-point loading store SIMD instructions (SIMD and 4SIMD) / Number of instruction executions * 100
	Floating operand (SIMD) (%)	Number of floating-point arithmetic SIMD instructions (SIMD and 4SIMD) / Number of instruction executions * 100
	FMA (SIMD) (%)	Number of floating-point FMA SIMD instructions (SIMD and 4SIMD) / Number of instruction executions * 100
	Fixed operand (SIMD) (%)	Number of fixed-point arithmetic SIMD instructions (SIMD and 4SIMD) / Number of instruction executions * 100
	SIMD (%)	(Number of floating-point loading store SIMD instructions (SIMD and 4SIMD) + Number of floating-point arithmetic SIMD instructions (SIMD and 4SIMD) + Number of floating-point FMA SIMD instructions (SIMD and 4SIMD) + Number of fixed-point arithmetic SIMD instructions (SIMD and 4SIMD)) / Number of instruction executions * 100
	SIMD load store instruction rate (%)	Number of floating-point loading SIMD store instructions (SIMD and 4SIMD) / (Number of floating-point loading store instructions + Number of floating-point loading store SIMD instructions (SIMD and 4SIMD)) * 100
	Prefetch	Number of prefetch instructions (exclude "indirect prefetch" instruction) + Number of indirect prefetch instructions
	Indirect prefetch	Number of indirect prefetch instructions

*1 : The MIPS peak value of each core is calculated by the following expression:

$\text{Frequency in CPU core} * \text{Number of instructions that can be issued per cycle} * 1000(\text{MIPS})$

The values in the calculation expressions are used from the following PA events.

- | | |
|--|-----------------------------------|
| (1) effective_instruction_counts | (2) XSIMD_load_store_instructions |
| (3) XSIMD_floating_instructions | (4) XSIMD_fma_instructions |
| (5) XSIMD_fixed_point_instructions | (6) prefetch_instructions |
| (7) nonSIMD_XSIMD_indirect_prefetch_instructions | (8) load_store_instructions |

- | | |
|---|-----|
| Executed number of instructions | (1) |
| Number of floating-point loading store SIMD instructions (SIMD and 4SIMD) | (2) |
| Number of floating-point arithmetic SIMD instructions (SIMD and 4SIMD) | (3) |
| Number of floating-point FMA SIMD instructions (SIMD and 4SIMD) | (4) |
| Number of fixed-point arithmetic SIMD instructions (SIMD and 4SIMD) | (5) |
| Number of floating-point loading store instructions | (8) |

Number of prefetch instructions (exclude "indirect prefetch" instruction) (6)

Number of indirect prefetch instructions (7)

Table 3.29 Formulas of the Hardware monitor information (Instructions_NOSIMD)

Event name	Measured information (unit)	Formula
Instructions_NOSIMD	Elapsed time (s)	Elapsed time for executing instructions of the measured range
	Instructions	Executed number of instructions
	MIPS	Number of instruction executions / Elapsed time / 1.0e+6
	MIPS/PEAK (%) (*3)	MIPS / (Number of execution cores * MIPS peak value of each core) * 100
	Load/Store (NOSIMD) (%)	Number of floating-point loading store instructions / Number of instruction executions * 100
	Floating operand (NOSIMD) (%)	Number of floating-point arithmetic instructions / Number of instruction executions * 100
	FMA (NOSIMD) (%)	Number of floating-point FMA instructions / Number of instruction executions * 100
	Fixed operand (NOSIMD) (%)	Number of fixed-point arithmetic instructions / Number of instruction executions * 100
	NOSIMD (%)	(Number of floating-point loading store instructions + Number of floating-point arithmetic instructions + Number of floating-point FMA instructions + Number of fixed-point arithmetic instructions) / Number of instruction executions * 100
	NOSIMD load store instruction rate (%)	Number of floating-point loading store instructions / (Number of floating-point loading store instructions + Number of floating-point loading store SIMD instructions (SIMD and 4SIMD)) * 100
	Prefetch	Number of prefetch instructions (exclude "indirect prefetch" instruction)+ Number of indirect prefetch instructions
Indirect prefetch	Number of indirect prefetch instructions	

The values in the calculation expressions are used from the following PA events.

(1) effective_instruction_counts

(2) load_store_instructions

(3) floating_instructions

(4) fma_instructions

(5) fixed_point_instructions

(6) prefetch_instructions

(7) nonSIMD_XSIMD_indirect_prefetch_instructions

(8) XSIMD_load_store_instructions

Executed number of instructions (1)

Number of floating-point loading store instructions (2)

Number of floating-point arithmetic instructions (3)

Number of floating-point FMA instructions (4)

Number of fixed-point arithmetic instructions (5)

Number of floating-point loading store SIMD instructions (SIMD and 4SIMD) (8)

Number of prefetch instructions (exclude "indirect prefetch" instruction) (6)

Number of indirect prefetch instructions (7)

Table 3.30 Formulas of the Hardware monitor information (MEM_access)

Event name	Measured information (unit)	Formula
MEM_access	Elapsed time (s)	Elapsed time for executing instructions of the measured range
	Load/Store (SIMD) (%)	Number of floating-point loading store SIMD instructions (SIMD and 4SIMD) / Number of instruction executions * 100
	Load/Store (NOSIMD) (%)	Number of floating-point loading store instructions / Number of instruction executions * 100
	Memory access throughput (core) (GB/s)	Amount of data transfer between memory and CPU core * 256 / Elapsed time / 1.0e+9
	Write back rate (%)	Memory CPU core writing data amount / Amount of data transfer between memory CPU cores * 100
	Memory access throughput (PEAK)	<ul style="list-style-type: none"> - When the write backing rate is smaller than 50% $240 + 240 * ((\text{Memory CPU core writing data amount} / \text{Amount of data transfer between memory CPU cores}) / (1 - \text{Memory CPU core writing data amount} / \text{Amount of data transfer between memory CPU cores}))$ - When the write backing rate is 50% or more $240 + 240 * ((1 - \text{Memory CPU core writing data amount} / \text{Amount of data transfer between memory CPU cores}) / (\text{Memory CPU core writing data amount} / \text{Amount of data transfer between memory CPU cores}))$
Memory access throughput peak rate (%)	Memory access throughput / Memory access throughput (PEAK) * 100	

The values in the calculation expressions are used from the following PA events.

- | | |
|----------------------------------|-----------------------------------|
| (1) effective_instruction_counts | (2) XSIMD_load_store_instructions |
| (3) load_store_instructions | (4) L2_miss_dm |
| (5) L2_miss_pf | (6) L2_wb_dm |
| (7) L2_wb_pf | |
-
- | | |
|---|-----------------------|
| Number of instruction executions | (1) |
| Number of floating-point loading store SIMD instructions (SIMD and 4SIMD) | (2) |
| Number of floating-point loading store instructions | (3) |
| Amount of data transfer between memory CPU cores | (4) + (5) + (6) + (7) |
| Memory CPU core writing data amount | (6) + (7) |

Table 3.31 Formulas of the Hardware monitor information (Performance)

Event name	Measured information (unit)	Formula
Performance	Elapsed time (s)	Elapsed time for executing instructions of the measured range
	2-4 instruction commit (S)	(CPU cycle number - (Cycle when number of completion instructions is 0 + Cycle when number of completion instructions is 1)) / Frequency in CPU core
	1 instruction commit (S)	Cycle when number of completion instructions is 1 / Frequency in CPU core
	Operation wait (S)	Cycle when number of floating-point numbers of operation execution completion instructions is 0 / Frequency in CPU core

Event name	Measured information (unit)	Formula
	Cache access wait (S)	(Cycle when the number of completion instructions is 0 numerical according to the data waiting by the memory access - Cycle of waiting by level 2 cache miss number) / Frequency in CPU core
	Memory access wait (S)	(Cycle of waiting by level 2 cache miss number + A store port full factor makes CSE empty and the number of completion instructions is cycle that 0 number) / Frequency in CPU core (*1),(*2)
	Fetch wait (S)	(CSE is cycle when the number of completion instructions is 0 because of emptiness number - A store port full factor makes CSE empty and the number of completion instructions is cycle that 0 number) / Frequency in CPU core (*1),(*2)
	Other wait (S)	Time when number of completion instructions is 0 - (Operation waiting time + Cache access waiting time + Memory access waiting time + Instruction fetch waiting time)

*1 : CSE is a buffer to hold the information on the instruction currently executing (it is issued, but not completed yet).

*2 : The store port is a queue that stores the store data.

The values in the calculation expressions are used from the following PA events.

- | | |
|----------------------|------------------------------|
| (1) cycle_counts | (2) 0endop |
| (3) 1endop | (4) eu_comp_wait |
| (5) op_stv_wait | (6) op_stv_wait_sxmiss |
| (7) cse_window_empty | (8) cse_window_empty_sp_full |

- | | |
|--|-----|
| CPU cycle number | (1) |
| Cycle when number of completion instructions is 0 | (2) |
| Cycle when number of completion instructions is 1 | (3) |
| Cycle when number of integer or floating-point numbers of operation execution completion instructions is 0 | (4) |
| Cycle when the number of completion instructions is 0 numerical according to the data waiting by the memory access | (5) |
| Cycle of waiting by level 2 cache miss number | (6) |
| CSE is cycle when the number of completion instructions is 0 because of emptiness number | (7) |
| A store port full factor makes CSE empty and the number of completion instructions is cycle that 0 number | (8) |

Table 3.32 Formulas of the Hardware monitor information (Statistics)

Event name	Measured information (unit)	Formula
Statistics	Elapsed time (s)	Elapsed time for executing instructions of the measured range
	MFLOPS	Number of floating-point arithmetic instructions / Elapsed time / 1.0e+6
	MFLOPS/PEAK (%) (*1)	MFLOPS / (Number of execution cores * MFLOPS peak value of each core) * 100
	MIPS	Number of instruction executions / Elapsed time / 1.0e+6
	MIPS/PEAK (%) (*2)	MIPS / (Number of execution cores * MIPS peak value of each core) * 100
	Floating operand (%)	Number of floating-point arithmetic instructions / Total operand * 100

*1 : The MFLOPS peak value of each core is calculated by the following expression:

Frequency in CPU core * Number of floating-point FMA SIMD instructions * Number of floating-point arithmetic instructions * Number of processing of operand of SIMD of floating-point instructions * 1000(MFLOPS)

*2 : The MIPS peak value of each core is calculated by the following expression:

Frequency in CPU core * Number of instructions that can be issued per cycle * 1000(MIPS)

The values in the calculation expressions are used from the following PA events.

- | | |
|----------------------------------|--------------------------|
| (1) effective_instruction_counts | (2) 1FLOPS_instructions |
| (3) 2FLOPS_instructions | (4) 4FLOPS_instructions |
| (5) 8FLOPS_instructions | (6) 16FLOPS_instructions |

- | | |
|--|--------------------------------------|
| Number of instruction executions | (1) |
| Number of floating-point arithmetic instructions | (2) + 2*(3) + 4*(4) + 8*(5) + 16*(6) |
| Total operand | (1) + (3) + 3*(4) + 7*(5) + 15*(6) |

Table 3.33 Formulas of the Hardware monitor information (TLB)

Event name	Measured information (unit)	Formula
TLB	Elapsed time (s)	Elapsed time for executing instructions of the measured range
	Instructions	Executed number of instructions
	uTLB miss (%)	uTLB miss frequency of data / Number of memory access instructions * 100
	mTLB miss (%)	Main TLB miss frequency of data / Number of memory access instructions * 100

The values in the calculation expressions are used from the following PA events.

- | | |
|----------------------------------|-----------------------------------|
| (1) effective_instruction_counts | (2) load_store_instructions |
| (3) SIMD_load_store_instructions | (4) 4SIMD_load_store_instructions |
| (5) uDTLB_miss | (6) DTLB_write |

- | | |
|--------------------------------------|---------------------|
| Executed number of instructions | (1) |
| uTLB miss frequency of data | (5) |
| Main TLB miss frequency of data | (6) |
| Number of memory access instructions | (2) + 2*(3) + 4*(4) |

3.4.5.3 Using the Hardware monitor information

The usage of each event of the Hardware monitor information is explained below.

(1) Cache

Cache is used to confirm the execution status of instructions accessing the memory and the occurrence of cache miss.

The execution performance of a program improves if the cache miss rate is low.

(2) Instructions_SIMD

Instructions are used to confirm the following execution status of instructions that process floating-point data.

- Execution rate of loading, store instruction, and floating-point operation instruction of floating-point data
- Execution rate of SIMD instruction

The operation performance of a program improves by the execution rate of the floating-point arithmetic and the execution rate of SIMD instruction large.

(3) Instructions_NOSIMD

Instructions are used to confirm the following execution status of instructions that process floating-point data.

- Execution rate of loading, store instruction, and floating-point operation instruction of floating-point data
- Execution rate of NOSIMD instruction

The operation performance of a program improves by the execution rate of the floating-point arithmetic and the execution rate of SIMD instruction large.

(4) MEM_access

MEM_access confirms the execution status of instructions accessing the data transfer efficiency and the memory between CPU core and the memory.

It is a program that executes a lot of data transfers by the memory access throughput large between CPU and the memory.

(5) Performance

Performance is used to confirm the rate at the time that is executed and waited for with the instruction row.

Moreover, the rate in which two or more instructions are executed in parallel and the rate at the execution waiting time according to the cause can be confirmed.

The execution performance of the program improves by the rate at 2-4 instruction committing time large, the rate at the execution waiting time small.

(6) Statistics

A program has a high execution performance and operation performance as the peak value of each performance gets closer to the MIPS value and MFLOPS value.

(7) TLB

TLB is used to confirm the occurrence situation of the TLB mistake between CPU core and the memory at the data transfer.

3.4.5.4 Output format of the Hardware monitor information

The Hardware monitor information is output for an entire application, processes, and threads.

The following figure shows the output format of the Hardware monitor information.

Figure 3.16 Output format of the Hardware monitor information

```

Performance monitor : @category

*****
Application
*****
(1)
Kind      @item1      @item2      ...      @item n
-----
AVG       @value1     @value2     ...      @value n  @name @no
MAX       @value1     @value2     ...      @value n
MIN       @value1     @value2     ...      @value n

*****
(2) Process @pno
*****
(1)
Kind      @item1      @item2      ...      @item n
-----
AVG       @value1     @value2     ...      @value n  @name @no
MAX       @value1     @value2     ...      @value n
MIN       @value1     @value2     ...      @value n

*****
(3) Thread @thno
*****
          @item1      @item2      ...      @item n
-----
(1)      @value1     @value2     ...      @value n  @name @no

```

- (1) The amount of @name, @no is repeated.
- (2) The amount of @pno is repeated.
- (3) The amount of @thno is repeated.

The output items of "Figure 3.16 Output format of the Hardware monitor information" are described in below:

Table 3.34 Output items of the Hardware monitor information

Output item	Description
@category	Category name of the Hardware monitor information
@item n	Output items of the Hardware monitor information
@name	Group name
@no	Advance number
@value n	Value of the Hardware monitor information
@pno	Process number
@thno	Thread number

The items that are output according to the category of the Hardware monitor information are shown below.

Figure 3.17 Output format of Cache

```

Performance monitor : Cache

*****
Application
*****

Kind    Elapsed(s)      Inst    L1I miss(%)    L1D miss(%)
-----
AVG     @elapsed        @inst   @11i_miss      @11d_miss  @name @no
MAX     @elapsed        @inst   @11i_miss      @11d_miss
MIN     @elapsed        @inst   @11i_miss      @11d_miss

Kind    Elapsed(s)      L2 miss(%)  L2 dm miss(%)  L2 pf miss(%)
-----
AVG     @elapsed        @l2_miss @l2_dm_miss    @l2_pf_miss @name @no
MAX     @elapsed        @l2_miss @l2_dm_miss    @l2_pf_miss
MIN     @elapsed        @l2_miss @l2_dm_miss    @l2_pf_miss

*****
Process @pno
*****

Kind    Elapsed(s)      Inst    L1I miss(%)    L1D miss(%)
-----
AVG     @elapsed        @inst   @11i_miss      @11d_miss  @name @no
MAX     @elapsed        @inst   @11i_miss      @11d_miss
MIN     @elapsed        @inst   @11i_miss      @11d_miss

Kind    Elapsed(s)      L2 miss(%)  L2 dm miss(%)  L2 pf miss(%)
-----
AVG     @elapsed        @l2_miss @l2_dm_miss    @l2_pf_miss @name @no
MAX     @elapsed        @l2_miss @l2_dm_miss    @l2_pf_miss
MIN     @elapsed        @l2_miss @l2_dm_miss    @l2_pf_miss

*****
Thread @thno
*****

      Elapsed(s)      Inst    L1I miss(%)    L1D miss(%)
-----
      @elapsed        @inst   @11i_miss      @11d_miss  @name @no

      Elapsed(s)      L2 miss(%)  L2 dm miss(%)  L2 pf miss(%)
-----
      @elapsed        @l2_miss @l2_dm_miss    @l2_pf_miss @name @no

```

Table 3.35 Output items of Cache

Output item	Description
@elapsed	Elapsed time (s)
@inst	Number of instruction executions
@11i_miss	First instruction cache miss rate (%)
@11d_miss	First data cache miss rate (%)
@l2_miss	Second data cache miss rate (%)
@l2_dm_miss	Second data cache demand miss rate (%)
@l2_pf_miss	Second data cache prefetch miss rate (%)
@name	Group name

Output item	Description
@no	Advance number
@pno	Process number
@thno	Thread number

Figure 3.18 Output format of Instruction_SIMD

```

Performance monitor event: Instruction_SIMD

*****
Application
*****

Kind  Elapsed(s)      Inst           MIPS    MIPS/PEAK(%)
-----
AVG   @elapsed        @inst          @mips   @mips/peak @name @no
MAX   @elapsed        @inst          @mips   @mips/peak
MIN   @elapsed        @inst          @mips   @mips/peak

Kind  Elapsed(s)  LS_SIMD-or(%) Float_SIMD-or(%) Fma_SIMD-or(%) Fixed_SIMD-or(%)
-----
AVG   @elapsed   @ls_simd      @float_simd    @fma_simd      @fixed_simd @name @no
MAX   @elapsed   @ls_simd      @float_simd    @fma_simd      @fixed_simd
MIN   @elapsed   @ls_simd      @float_simd    @fma_simd      @fixed_simd

Kind  Elapsed(s)      SIMD(%)  LS_SIMD_rate(%)  Prefetch  Indirect_pf
-----
AVG   @elapsed        @simd    @ls_simd_rate    @prefetch   @indirect_pf @name @no
MAX   @elapsed        @simd    @ls_simd_rate    @prefetch   @indirect_pf
MIN   @elapsed        @simd    @ls_simd_rate    @prefetch   @indirect_pf

*****
Process @pno
*****

Kind  Elapsed(s)      Inst           MIPS    MIPS/PEAK(%)
-----
AVG   @elapsed        @inst          @mips   @mips/peak @name @no
MAX   @elapsed        @inst          @mips   @mips/peak
MIN   @elapsed        @inst          @mips   @mips/peak

Kind  Elapsed(s)  LS_SIMD-or(%) Float_SIMD-or(%) Fma_SIMD-or(%) Fixed_SIMD-or(%)
-----
AVG   @elapsed   @ls_simd      @float_simd    @fma_simd      @fixed_simd @name @no
MAX   @elapsed   @ls_simd      @float_simd    @fma_simd      @fixed_simd
MIN   @elapsed   @ls_simd      @float_simd    @fma_simd      @fixed_simd

Kind  Elapsed(s)      SIMD(%)  LS_SIMD_rate(%)  Prefetch  Indirect_pf
-----
AVG   @elapsed        @simd    @ls_simd_rate    @prefetch   @indirect_pf @name @no
MAX   @elapsed        @simd    @ls_simd_rate    @prefetch   @indirect_pf
MIN   @elapsed        @simd    @ls_simd_rate    @prefetch   @indirect_pf

*****
Thread @thno
*****

Kind  Elapsed(s)      Inst           MIPS    MIPS/PEAK(%)
-----
      @elapsed        @inst          @mips   @mips/peak @name @no

Kind  Elapsed(s)  LS_SIMD-or(%) Float_SIMD-or(%) Fma_SIMD-or(%) Fixed_SIMD-or(%)
-----
      @elapsed   @ls_simd      @float_simd    @fma_simd      @fixed_simd @name @no

Kind  Elapsed(s)      SIMD(%)  LS_SIMD_rate(%)  Prefetch  Indirect_pf
-----
AVG   @elapsed        @simd    @ls_simd_rate    @prefetch   @indirect_pf @name @no

```


Table 3.36 Output items of Instruction_SIMD

Output item	Description
@elapsed	Elapsed time (s)
@inst	Number of instruction executions
@mips	MIPS
@mips/peak	MIPS peak performance rate (%)
@ls_simd	Loading store instruction rate (SIMD) (%)
@float_simd	Floating-point arithmetic instruction rate (SIMD) (%)
@fma_simd	Floating-point FMA instruction rate (SIMD) (%)
@fixed_simd	Fixed-point arithmetic instruction rate (SIMD) (%)
@simd	SIMD instruction rate (%)
@ls_simd_rate	SIMD loading store instruction rate (%)
@prefetch	Number of prefetch instructions
@indirect_pf	Number of indirect prefetch instructions
@name	Group name
@no	Advance number
@pno	Process number
@thno	Thread number

Figure 3.19 Output format of Instruction_NOSIMD

Performance monitor event: Instruction_NOSIMD

```

*****
Application
*****

Kind  Elapsed(s)      Inst           MIPS  MIPS/PEAK(%)
-----
AVG   @elapsed        @inst          @mips  @mips/peak @name @no
MAX   @elapsed        @inst          @mips  @mips/peak
MIN   @elapsed        @inst          @mips  @mips/peak

Kind  Elapsed(s)      LS-or(%)      Float-or(%)    Fma-or(%)      Fixed-or(%)
-----
AVG   @elapsed        @ls           @float        @fma           @fixed @name @no
MAX   @elapsed        @ls           @float        @fma           @fixed
MIN   @elapsed        @ls           @float        @fma           @fixed

Kind  Elapsed(s)      NOSIMD(%)     LS_rate(%)     Prefetch        Indirect_pf
-----
AVG   @elapsed        @nosimd       @ls_rate       @prefetch       @indirect_pf @name @no
MAX   @elapsed        @nosimd       @ls_rate       @prefetch       @indirect_pf
MIN   @elapsed        @nosimd       @ls_rate       @prefetch       @indirect_pf

*****
Process @pno
*****

Kind  Elapsed(s)      Inst           MIPS  MIPS/PEAK(%)
-----
AVG   @elapsed        @inst          @mips  @mips/peak @name @no
MAX   @elapsed        @inst          @mips  @mips/peak
MIN   @elapsed        @inst          @mips  @mips/peak

Kind  Elapsed(s)      LS-or(%)      Float-or(%)    Fma-or(%)      Fixed-or(%)
-----
AVG   @elapsed        @ls           @float        @fma           @fixed @name @no
MAX   @elapsed        @ls           @float        @fma           @fixed
MIN   @elapsed        @ls           @float        @fma           @fixed

Kind  Elapsed(s)      NOSIMD(%)     LS_rate(%)     Prefetch        Indirect_pf
-----
AVG   @elapsed        @nosimd       @ls_rate       @prefetch       @indirect_pf @name @no
MAX   @elapsed        @nosimd       @ls_rate       @prefetch       @indirect_pf
MIN   @elapsed        @nosimd       @ls_rate       @prefetch       @indirect_pf

*****
Thread @thno
*****

Kind  Elapsed(s)      Inst           MIPS  MIPS/PEAK(%)
-----
      @elapsed        @inst          @mips  @mips/peak @name @no

Kind  Elapsed(s)      LS-or(%)      Float-or(%)    Fma-or(%)      Fixed-or(%)
-----
      @elapsed        @ls           @float        @fma           @fixed @name @no

Kind  Elapsed(s)      NOSIMD(%)     LS_rate(%)     Prefetch        Indirect_pf
-----
AVG   @elapsed        @nosimd       @ls_rate       @prefetch       @indirect_pf @name @no

```

Table 3.37 Output items of Instruction_NOSIMD

Output item	Description
@elapsed	Elapsed time (s)
@inst	Number of instruction executions
@mips	MIPS
@mips/peak	MIPS peak performance rate (%)
@ls	Loading store instruction rate (NOSIMD) (%)
@float	Floating-point arithmetic instruction rate (NOSIMD) (%)
@fma	Floating-point FMA instruction rate (NOSIMD) (%)
@fixed	Fixed-point arithmetic instruction rate (SIMD) (%)
@nosimd	NOSIMD instruction rate (%)
@ls_rate	NOSIMD loading store instruction rate (%)
@prefetch	Number of prefetch instructions
@indirect_pf	Number of indirect prefetch instructions
@name	Group name
@no	Advance number
@pno	Process number
@thno	Thread number

Output item	Description
@mem_tp_peak	Memory access throughput (PEAK) (GB/s)
@mem_tp/peak	Memory access throughput / PEAK (%)
@name	Group name
@no	Advance number
@pno	Process number
@thno	Thread number

Figure 3.21 Output format of Performance

```

Performance monitor event: Performance

*****
Application
*****

Kind  Elapsed(s)  2-4icmit(S)  1i_cmit(S)  op_wait(S)  cache_wait(S)
-----
AVG    @elapsed    @2-4i_cmit   @1i_cmit    @op_wait    @cache_wait  @name @no
MAX    @elapsed    @2-4i_cmit   @1i_cmit    @op_wait    @cache_wait
MIN    @elapsed    @2-4i_cmit   @1i_cmit    @op_wait    @cache_wait

Kind  Elapsed(s)  Mem_wait(S)  fetch_wait(S)  other_wait(S)
-----
AVG    @elapsed    @mem_wait    @fetch_wait    @other_wait  @name @no
MAX    @elapsed    @mem_wait    @fetch_wait    @other_wait
MIN    @elapsed    @mem_wait    @fetch_wait    @other_wait

*****
Process @pno
*****

Kind  Elapsed(s)  2-4icmit(S)  1i_cmit(S)  op_wait(S)  cache_wait(S)
-----
AVG    @elapsed    @2-4i_cmit   @1i_cmit    @op_wait    @cache_wait  @name @no
MAX    @elapsed    @2-4i_cmit   @1i_cmit    @op_wait    @cache_wait
MIN    @elapsed    @2-4i_cmit   @1i_cmit    @op_wait    @cache_wait

Kind  Elapsed(s)  Mem_wait(S)  fetch_wait(S)  other_wait(S)
-----
AVG    @elapsed    @mem_wait    @fetch_wait    @other_wait  @name @no
MAX    @elapsed    @mem_wait    @fetch_wait    @other_wait
MIN    @elapsed    @mem_wait    @fetch_wait    @other_wait

*****
Thread @thno
*****

Elapsed(s)  2-4icmit(S)  1i_cmit(S)  op_wait(S)  cache_wait(S)
-----
@elapsed    @2-4i_cmit   @1i_cmit    @op_wait    @cache_wait  @name @no

Elapsed(s)  Mem_wait(S)  fetch_wait(S)  other_wait(S)
-----
@elapsed    @mem_wait    @fetch_wait    @other_wait  @name @no

```

Table 3.39 Output items of Performance

Output item	Description
@elapsed	Elapsed time (s)

Output item	Description
@2-4i_cmit	2-4 instructions committing time (S)
@1i_cmit	One instruction committing time (S)
@op_wait	Operation waiting time (S)
@cache_wait	Cache access waiting time (S)
@mem_wait	Memory access waiting time (S)
@fetch_wait	Instruction fetch waiting (S)
@other_wait	Other waiting time (S)
@name	Group name
@no	Advance number
@pno	Process number
@thno	Thread number

Figure 3.22 Output format of Statistics

```

Performance monitor event: Statistics

*****
Application
*****

Kind    Elapsed(s)      MFLOPS  MFLOPS/PEAK(%)
-----
AVG     @elapsed        @mflops @mflops/peak  @name  @no
MAX     @elapsed        @mflops @mflops/peak
MIN     @elapsed        @mflops @mflops/peak

Kind    Elapsed(s)      MIPS    MIPS/PEAK(%)  Floating-op(%)
-----
AVG     @elapsed        @mips  @mips/peak     @fl-op @name  @no
MAX     @elapsed        @mips  @mips/peak     @fl-op
MIN     @elapsed        @mips  @mips/peak     @fl-op

*****
Process @pno
*****

Kind    Elapsed(s)      MFLOPS  MFLOPS/PEAK(%)
-----
AVG     @elapsed        @mflops @mflops/peak  @name  @no
MAX     @elapsed        @mflops @mflops/peak
MIN     @elapsed        @mflops @mflops/peak

Kind    Elapsed(s)      MIPS    MIPS/PEAK(%)  Floating-op(%)
-----
AVG     @elapsed        @mips  @mips/peak     @fl-op @name  @no
MAX     @elapsed        @mips  @mips/peak     @fl-op
MIN     @elapsed        @mips  @mips/peak     @fl-op

*****
Thread @thno
*****

Kind    Elapsed(s)      MFLOPS  MFLOPS/PEAK(%)
-----
      @elapsed        @mflops @mflops/peak  @name  @no

Kind    Elapsed(s)      MIPS    MIPS/PEAK(%)  Floating-op(%)
-----
      @elapsed        @mips  @mips/peak     @fl-op @name  @no

```

Table 3.40 Output items of Statistics

Output item	Description
@elapsed	Elapsed time (s)
@mflops	MFLOPS
@mflops/peak	MFLOPS peak performance rate (%)
@mips	MIPS
@mips/peak	MIPS peak performance rate (%)
@fl-op	Floating-point arithmetic rate (%)
@name	Group name
@no	Advance number

Output item	Description
@pno	Process number
@thno	Thread number

Figure 3.23 Output format of TLB

```

Performance monitor event: TLB

*****
Application
*****

  Kind    Elapsed(s)      Inst    uTLB-op(%)      mTLB-op(%)
-----
  AVG     @elapsed          @inst   @ut lb         @mt lb  @name  @no
  MAX     @elapsed          @inst   @ut lb         @mt lb
  MIN     @elapsed          @inst   @ut lb         @mt lb

*****
Process @pno
*****

  Kind    Elapsed(s)      Inst    uTLB-op(%)      mTLB-op(%)
-----
  AVG     @elapsed          @inst   @ut lb         @mt lb  @name  @no
  MAX     @elapsed          @inst   @ut lb         @mt lb
  MIN     @elapsed          @inst   @ut lb         @mt lb

*****
Thread @thno
*****

  Kind    Elapsed(s)      Inst    uTLB-op(%)      mTLB-op(%)
-----
          @elapsed          @inst   @ut lb         @mt lb  @name  @no

```

Table 3.41 Output items of TLB

Output item	Description
@elapsed	Elapsed time (s)
@inst	Number of instruction executions
@uTLB-op	Micro data TLB miss rate of data (%)
@mTLB-op	Main TLB miss rate of data (%)
@name	Group name
@no	Advance number
@pno	Process number
@thno	Thread number

The output form when the measurement event number is specified ("-Hevent_number" Option is specified) is shown below.

Figure 3.24 Output form of measurement event number specification

```

Performance monitor - Procedures profile event:Internal

*****
Application - performance monitors
*****

Ev.00 = @picu0_no
Ev.01 = @picl0_no
Ev.02 = @picu1_no
Ev.03 = @picl1_no

-----
      Ev.00      Ev.01      Ev.02      Ev.03      Start      End
-----
@picu0_value  @picl0_value  @picu1_value  @picl1_value  --         --  @name

Ev.04 = @picu2_no
Ev.05 = @picl2_no
Ev.06 = @picu3_no
Ev.07 = @picl3_no

-----
      Ev.04      Ev.05      Ev.06      Ev.07      Start      End
-----
@picu2_value  @picl2_value  @picu3_value  @picl3_value  --         --  @name

*****
Process @pno - performance monitors
*****

Ev.00 = @picu0_no
Ev.01 = @picl0_no
Ev.02 = @picu1_no
Ev.03 = @picl1_no

-----
      Ev.00      Ev.01      Ev.02      Ev.03      Start      End
-----
@picu0_value  @picl0_value  @picu1_value  @picl1_value  --         --  @nan

Ev.04 = @picu2_no
Ev.05 = @picl2_no
Ev.06 = @picu3_no
Ev.07 = @picl3_no

-----
      Ev.04      Ev.05      Ev.06      Ev.07      Start      End
-----
@picu2_value  @picl2_value  @picu3_value  @picl3_value  --         --  @nan

*****
Thread @thno - performance monitors
*****

Ev.00 = @picu0_no
Ev.01 = @picl0_no
Ev.02 = @picu1_no
Ev.03 = @picl1_no

-----
      Ev.00      Ev.01      Ev.02      Ev.03      Start      End
-----
@picu0_value  @picl0_value  @picu1_value  @picl1_value  --         --  @name

Ev.04 = @picu2_no
Ev.05 = @picl2_no
Ev.06 = @picu3_no
Ev.07 = @picl3_no

-----
      Ev.04      Ev.05      Ev.06      Ev.07      Start      End
-----
@picu2_value  @picl2_value  @picu3_value  @picl3_value  --         --  @name

```

Table 3.42 Output item of measurement event number specification

Output item	Description
@picu0_no	Event number specified for picu0
@picl0_no	Event number specified for picl0
@picu1_no	Event number specified for picu1
@picl1_no	Event number specified for picl1
@picu2_no	Event number specified for picu2
@picl2_no	Event number specified for picl2
@picu3_no	Event number specified for picu3
@picl3_no	Event number specified for picl3
@picu0_value	Measurements of picu0
@picl0_value	Measurements of picl0
@picu1_value	Measurements of picu1
@picl1_value	Measurements of picl1
@picu2_value	Measurements of picu2
@picl2_value	Measurements of picl2
@picu3_value	Measurements of picu3
@picl3_value	Measurements of picl3
@name	Measurement range name
@pno	Process number
@thno	Thread number

3.4.6 Largepage information

3.4.6.1 Measurement information on Largepage memory use information

The table below shows measurement information on Largepage memory information.

Measured information	Description
All memory gain	Memory size acquired from System (Byte)
Number of arenas	Number of arenas used
Arena memory gain	Memory size acquired from System for arena (Byte)
Arena memory usage	Memory size allocated from arena in process (Byte)
Arena memory unused amount	Memory size that it is arena and unused (Byte)
Arena memory usage rate (At start)	Arena memory usage rate at measurement start (%)
Arena memory usage rate (At end)	Arena memory usage rate when measurement ends (%)
Number of mmaped chunks	Number of mmaped chunks
Mmapped chunk gain	Memory size acquired from System for mmaped_chunk (Byte)



Note

When region reduces between from the measurement beginning to the end, the value of "Measured information" might reach a minus value.

3.4.6.2 Output format of largepage memory use information

Figure 3.25 Output format of largepage memory use information

```

LPGUsage profile
*****
Application
*****
Kind  Memory_Total      Arena  Arena_Total  Arena_Inuse  Arena_Free  Arena_rate(S)  Arena_rate(E)  Mmapped  Mmapped_Total  @name no
-----
AVG   @memtotal           @arena @arenatotal  @aranainuse  @arenafree  @arena_rate_S  @arena_rate_E  @mmapped @mmappedtotal
MAX   @memtotal           @arena @arenatotal  @aranainuse  @arenafree  @arena_rate_S  @arena_rate_E  @mmapped @mmappedtotal
MIN   @memtotal           @arena @arenatotal  @aranainuse  @arenafree  @arena_rate_S  @arena_rate_E  @mmapped @mmappedtotal
*****
Process @pno
*****
Memory_Total      Arena  Arena_Total  Arena_Inuse  Arena_Free  Arena_rate(S)  Arena_rate(E)  Mmapped  Mmapped_Total  @name no
-----
@memtotal           @arena @arenatotal  @aranainuse  @arenafree  @arena_rate_S  @arena_rate_E  @mmapped @mmappedtotal

```

Table 3.43 Output item of Largepage memory use information

Measured information	Unit
@memtotal	All memory gain (Byte)
@arena	Number of arenas
@arenatotal	Arena memory gain (Byte)
@aranainuse	Arena memory usage (Byte)
@arenafree	Arena memory unused amount (Byte)
@arena_rate_S	Arena memory usage rate (At start) (%)
@arena_rate_E	Arena memory usage rate (At end) (%)
@mmapped	Number of mmapped chunks
@mmappedtotal	Mmapped chunk gain (Byte)
@pno	Process number
@name	Group name
@no	Advance number

3.4.6.3 Measurement information on Largepage statistical information

Measured information	Description
Counts	Memory acquisition number of issuances
Time (MIN)	Minimum value at memory acquisition processing time (microsecond)
Time (MAX)	Maximum value at memory acquisition processing time (microsecond)
Time (AVG)	Average value at memory acquisition processing time (microsecond)
Size (MIN)	Minimum value of demand size (Byte)
Size (MAX)	Maximum value of demand size (Byte)
Size (AVG)	Average value of demand size (Byte)
Arena generation frequency	Frequency in which mmapped arena is generated
Arena liberating frequency	Frequency in which mmapped arena is liberated

Measured information	Description
Heap area expansion frequency	Frequency in which heap area is extended with sbrk
Arena extended partition generation frequency	Frequency in which enhancing mmap region in main arena is generated
Mmapped chunk generation frequency	Mmapped chunk generation frequency
Mmapped chunk deletion frequency	Mmapped chunk deletion frequency
Seek frequency	Frequency that succeeds in free area acquisition because of acquisition of memory in less than 128 Bytes (high speed) (FASTBIN)
The maximum search frequency of free area	The maximum search frequency of free area
Average search frequency of free area	Average search frequency of free area
The maximum search frequency of free area	The maximum search frequency of free area (When succeeding in acquisition)
Average search frequency of free area	Average search frequency of free area (When succeeding in acquisition)
The maximum search frequency of free area	The maximum search frequency of free area (When failing in acquisition)
Average search frequency of free area	Average search frequency of free area (When failing in acquisition)

3.4.6.4 Output format of largepage statistical information

Figure 3.26 Output format of largepage statistical information

```

LPGStats profile
*****
Application
*****

Kind      Count      TimeMin     TimeMax     TimeAvg     SizeMin     SizeMax     SizeAvg
-----
AVG       @count     @time_min   @time_max   @time_avg   @size_min   @size_max   @size_avg   malloc      all 0
MAX       @count     @time_min   @time_max   @time_avg   @size_min   @size_max   @size_avg
MIN       @count     @time_min   @time_max   @time_avg   @size_min   @size_max   @size_avg
-----
AVG       @count     @time_min   @time_max   @time_avg   @size_min   @size_max   @size_avg   calloc
MAX       @count     @time_min   @time_max   @time_avg   @size_min   @size_max   @size_avg
MIN       @count     @time_min   @time_max   @time_avg   @size_min   @size_max   @size_avg
-----
AVG       @count     @time_min   @time_max   @time_avg   @size_min   @size_max   @size_avg   memalign
MAX       @count     @time_min   @time_max   @time_avg   @size_min   @size_max   @size_avg
MIN       @count     @time_min   @time_max   @time_avg   @size_min   @size_max   @size_avg
-----
AVG       @count     @time_min   @time_max   @time_avg   @size_min   @size_max   @size_avg   valloc
MAX       @count     @time_min   @time_max   @time_avg   @size_min   @size_max   @size_avg
MIN       @count     @time_min   @time_max   @time_avg   @size_min   @size_max   @size_avg
-----
AVG       @count     @time_min   @time_max   @time_avg   @size_min   @size_max   @size_avg   pvalloc
MAX       @count     @time_min   @time_max   @time_avg   @size_min   @size_max   @size_avg
MIN       @count     @time_min   @time_max   @time_avg   @size_min   @size_max   @size_avg
-----
AVG       @count     @time_min   @time_max   @time_avg   @size_min   @size_max   @size_avg   p_memalign
MAX       @count     @time_min   @time_max   @time_avg   @size_min   @size_max   @size_avg
MIN       @count     @time_min   @time_max   @time_avg   @size_min   @size_max   @size_avg
-----
AVG       @count     @time_min   @time_max   @time_avg   @size_min   @size_max   @size_avg   free
MAX       @count     @time_min   @time_max   @time_avg   @size_min   @size_max   @size_avg
MIN       @count     @time_min   @time_max   @time_avg   @size_min   @size_max   @size_avg
-----

Kind      NewHeap     De lHeap     Expbrk     Expnmap     Newnmap     Delnmap
-----
AVG       @newheap   @de lheap   @expbrk   @expnmap   @newnmap   @delnmap   all 0
MAX       @newheap   @de lheap   @expbrk   @expnmap   @newnmap   @delnmap
MIN       @newheap   @de lheap   @expbrk   @expnmap   @newnmap   @delnmap

Kind      Fastbin     AllMax     AllAvg     SuccMax     SuccAvg     FailMax     FailAvg
-----
AVG       @s_fastbin @s_allmax   @s_allavg @s_succmax @s_succavg @s_failmax @s_failavg   all 0
MAX       @s_fastbin @s_allmax   @s_allavg @s_succmax @s_succavg @s_failmax @s_failavg
MIN       @s_fastbin @s_allmax   @s_allavg @s_succmax @s_succavg @s_failmax @s_failavg

```

```

*****
Process @pno
*****

allocstats_total
-----
Count      TimeMin    TimeMax    TimeAvg    SizeMin    SizeMax    SizeAvg
-----
@count     @time_min  @time_max  @time_avg  @size_min  @size_max  @size_avg  malloc    all 0
@count     @time_min  @time_max  @time_avg  @size_min  @size_max  @size_avg  calloc
@count     @time_min  @time_max  @time_avg  @size_min  @size_max  @size_avg  memalign
@count     @time_min  @time_max  @time_avg  @size_min  @size_max  @size_avg  valloc
@count     @time_min  @time_max  @time_avg  @size_min  @size_max  @size_avg  pvalloc
@count     @time_min  @time_max  @time_avg  @size_min  @size_max  @size_avg  p_memalign
@count     @time_min  @time_max  @time_avg  @size_min  @size_max  @size_avg  free
-----

```

[Statistical information of each source library -- Information of the acquisition of the memory of each of the 32 libraries or less is output.]

```

Library 1
-----
Count      TimeMin    TimeMax    TimeAvg    SizeMin    SizeMax    SizeAvg
-----
@count     @time_min  @time_max  @time_avg  @size_min  @size_max  @size_avg  malloc    all 0
@count     @time_min  @time_max  @time_avg  @size_min  @size_max  @size_avg  calloc
@count     @time_min  @time_max  @time_avg  @size_min  @size_max  @size_avg  memalign
@count     @time_min  @time_max  @time_avg  @size_min  @size_max  @size_avg  valloc
@count     @time_min  @time_max  @time_avg  @size_min  @size_max  @size_avg  pvalloc
@count     @time_min  @time_max  @time_avg  @size_min  @size_max  @size_avg  p_memalign
@count     @time_min  @time_max  @time_avg  @size_min  @size_max  @size_avg  free
-----

Library 2
-----
Count      TimeMin    TimeMax    TimeAvg    SizeMin    SizeMax    SizeAvg
-----
@count     @time_min  @time_max  @time_avg  @size_min  @size_max  @size_avg  malloc    all 0
@count     @time_min  @time_max  @time_avg  @size_min  @size_max  @size_avg  calloc
@count     @time_min  @time_max  @time_avg  @size_min  @size_max  @size_avg  memalign
@count     @time_min  @time_max  @time_avg  @size_min  @size_max  @size_avg  valloc
@count     @time_min  @time_max  @time_avg  @size_min  @size_max  @size_avg  pvalloc
@count     @time_min  @time_max  @time_avg  @size_min  @size_max  @size_avg  p_memalign
@count     @time_min  @time_max  @time_avg  @size_min  @size_max  @size_avg  free
-----

```

[It is a repetition for a few minutes of the source library.]

```

NewHeap    DelHeap    Expbrk    Expmmap    Newmmap    Delmmap
-----
@newheap  @delheap  @expbrk   @expmmap  @newmmap  @delmmap  all 0
-----

Fastbin    AllMax    AllAvg    SuccMax    SuccAvg    FailMax    FailAvg
-----
@s_fastbin @s_allmax @s_allavg @s_succmax @s_succavg @s_failmax @s_failavg  all 0
-----

```

The value of allocstats_total (*1) is an additional value of the statistical information of each source library (*2).

Table 3.44 Output item of memory acquisition function (malloc, calloc, memalign, valloc, pvalloc, p_memalign, free)

Measured information	Unit
@library_name	Library name that outputs memory acquisition information
@count	Memory acquisition number of issuances
@time_min	Minimum time (microsecond)
@time_max	Maximum time (microsecond)
@time_avg	Average time (microsecond)
@size_min	Minimum size (Byte)
@size_max	Maximum size (Byte)
@size_avg	Average size (Byte)
@pno	Process number
@name	Group name
@no	Advance number



@size_min of free, @size_max, and @size_avg are always displayed by 0.

Table 3.45 Output item of arena information

Measured information	Description
@newheap	Arena generation frequency
@delheap	Arena liberating frequency
@expbrk	Heap area expansion frequency
@expmmap	Arena extended partition generation frequency
@newmmap	Mmapped chunk generation frequency
@delmmap	Mmapped chunk deletion frequency
@name	Group name
@no	Advance number

Table 3.46 Output item of seek information

Measured information	Description
@s_fastbin	Seek frequency
@s_allmax	The maximum search frequency of free area
@s_allavg	Average search frequency of free area
@s_succmax	The maximum search frequency of free area (When succeeding in acquisition)
@s_succavg	Average search frequency of free area (When succeeding in acquisition)
@s_failmax	The maximum search frequency of free area (When failing in acquisition)
@s_failavg	Average search frequency of free area (When failing in acquisition)
@name	Group name
@no	Advance number

3.5 Precision PA visibility function (Excel format)

This section explains the content of the precision PA visibility function (Excel format) output by the fappx command.



Note

It is necessary to install Excel to use the precision PA visibility function (Excel format).

3.5.1 Overview

The precise PA visibility function (Excel format) is a function to analyze the Advanced Profiler information output by CSV by using the Excel sheet in which the analysis procedure is defined by the macro beforehand, and to display the result in the graph and the table.

For this analysis, fapp must be executed 11 times. This is because 11 lots of hardware counter information are required. This means that differences might arise over the execution time. Minus numerical values might be output for some information due to these differences.

Follow the procedures below to perform these tasks:

1. Determine the measurement range, and incorporate the information collection routine in the program.
2. Compile.
3. Collect data.

4. Convert the data.
5. Use Excel sheets to analyze the data.

3.5.2 Collecting data (execution)

3.5.2.1 Specifying the measurement range

Determine the measurement range and incorporate the Advanced Profiler routine (precision PA) in the program.

Refer to "[3.2.2 Advanced Profiler routine \(precision PA\)](#)" for the method for incorporating the Advanced Profiler routine (precision PA).

3.5.2.2 Compiling/linking

Compile the program.

Refer to "[3.2.4 Compilation](#)" for the compilation method.

3.5.2.3 Collecting data

Use the fapp command to collect the data. Refer to "[3.2.5 fapp command](#)" for details of the fapp command.

Execute the command 11 times to collect data.

The "-H" option shown below must be added when collecting the data.

Table 3.47 -H option when collecting data

Collection count	-H option
First	-Hpa=1
Second	-Hpa=2
Third	-Hpa=3
Fourth	-Hpa=4
Fifth	-Hpa=5
Sixth	-Hpa=6
Seventh	-Hpa=7
Eighth	-Hpa=8
Ninth	-Hpa=9
Tenth	-Hpa=10
Eleventh	-Hpa=11

In addition the "-C" option and the "-d" option, specifying the data storage destination, must also be added.

Refer to "[3.2.5 fapp command](#)" for option details.



Example

Example of how to write a job launch script for collecting data

```
fapp -C -d pa1 -Hpa=1      mpiexec -n 8 ./a.out
fapp -C -d pa2 -Hpa=2      mpiexec -n 8 ./a.out
fapp -C -d pa3 -Hpa=3      mpiexec -n 8 ./a.out
fapp -C -d pa4 -Hpa=4      mpiexec -n 8 ./a.out
fapp -C -d pa5 -Hpa=5      mpiexec -n 8 ./a.out
fapp -C -d pa6 -Hpa=6      mpiexec -n 8 ./a.out
fapp -C -d pa7 -Hpa=7      mpiexec -n 8 ./a.out
fapp -C -d pa8 -Hpa=8      mpiexec -n 8 ./a.out
fapp -C -d pa9 -Hpa=9      mpiexec -n 8 ./a.out
```



```
fapp -C -d pa10 -Hpa=10 mpiexec -n 8 ./a.out
fapp -C -d pa11 -Hpa=11 mpiexec -n 8 ./a.out
```

3.5.3 Analyzing data

3.5.3.1 Converting data

Data must be converted to csv format before it is analyzed in Excel sheets.

Also, the data file names must be changed to output_prof_1.csv to output_prof_11.csv

Use the fappx command to convert the data. Refer to "3.2.6 fappx command" for details of the fappx command.



Example

Example of converting the collected data

```
$ fappx -A -d pa1 -o output_prof_1.csv -tcsv -Hpa
$ fappx -A -d pa2 -o output_prof_2.csv -tcsv -Hpa
$ fappx -A -d pa3 -o output_prof_3.csv -tcsv -Hpa
$ fappx -A -d pa4 -o output_prof_4.csv -tcsv -Hpa
$ fappx -A -d pa5 -o output_prof_5.csv -tcsv -Hpa
$ fappx -A -d pa6 -o output_prof_6.csv -tcsv -Hpa
$ fappx -A -d pa7 -o output_prof_7.csv -tcsv -Hpa
$ fappx -A -d pa8 -o output_prof_8.csv -tcsv -Hpa
$ fappx -A -d pa9 -o output_prof_9.csv -tcsv -Hpa
$ fappx -A -d pa10 -o output_prof_10.csv -tcsv -Hpa
$ fappx -A -d pa11 -o output_prof_11.csv -tcsv -Hpa
```

3.5.3.2 Excel operations

Use the Excel sheets to analyze the data.

Since the Excel sheets are under the login node, first transfer them to the user terminal.

```
/opt/FJSVmxlang/misc/CPUPA/FSDT_CPUPA_ENG.xlsm
```

Place the data files (output_prof_1.csv to output_prof_11.csv) in the same folder as the Excel sheets.

Double-click the Excel sheets to start Excel. This runs a macro and read starts.

3.5.3.2.1 Resolving security warnings

These Excel sheets use macros.

Therefore, security settings might stop the macros from starting and prevent operation.

If macros have been disabled, enable macros.

The method for enabling macros varies for different versions of Excel.

3.5.3.2.2 Specifying a process number

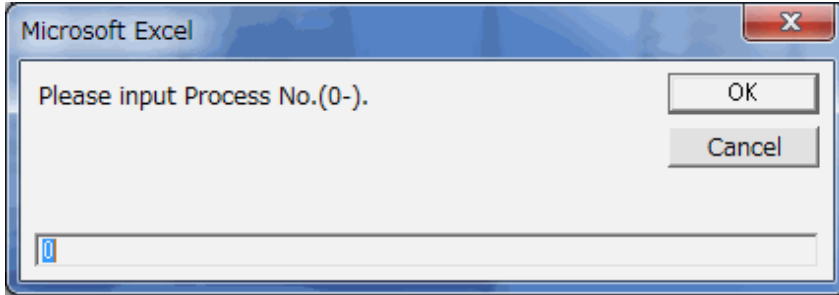
When the macro starts, a process number specification dialog is displayed automatically.

Enter the number of the process you want to analyze, then click the OK button.

If the specified process number is not in the data, a message indicating that is displayed and the process number specification dialog is displayed again.

If the Cancel button is clicked, processing stops and Excel ends.

Figure 3.27 Process number specification dialog



3.5.3.2.3 Specifying the segment name (measurement range)

If there are no problems with the process specification, a segment name specification dialog is displayed.

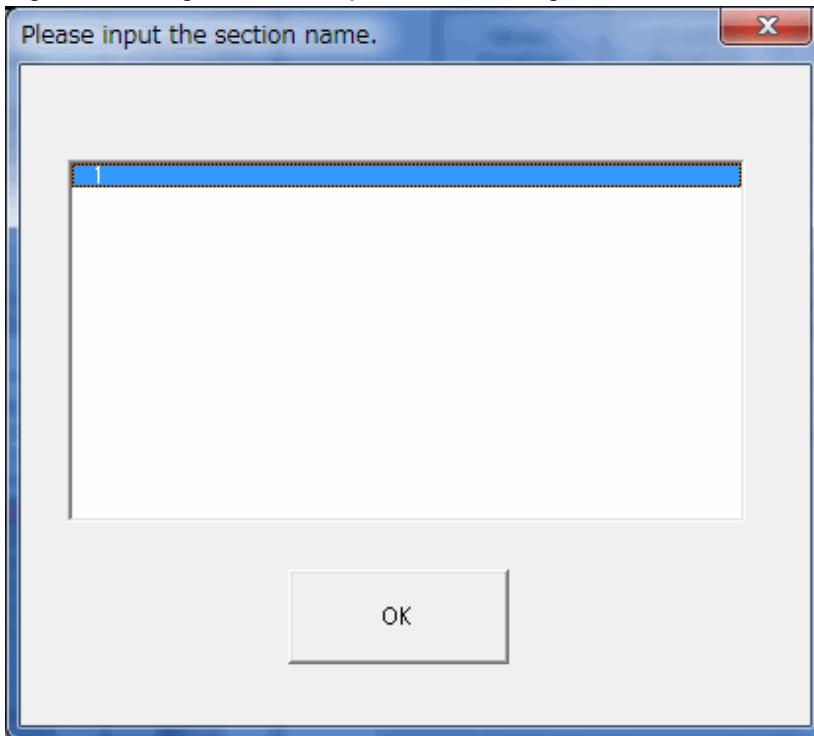
Enter the name of the segment you want to analyze, then click the OK button.

For the segment name, enter the segment name specified in "[3.5.2.1 Specifying the measurement range](#)".

If the specified segment name does not exist, a message indicating that is displayed and the segment name specification dialog is displayed again.

If the Cancel button is clicked, processing stops and Excel ends.

Figure 3.28 Segment name specification dialog



3.5.3.2.4 Generating Excel sheets

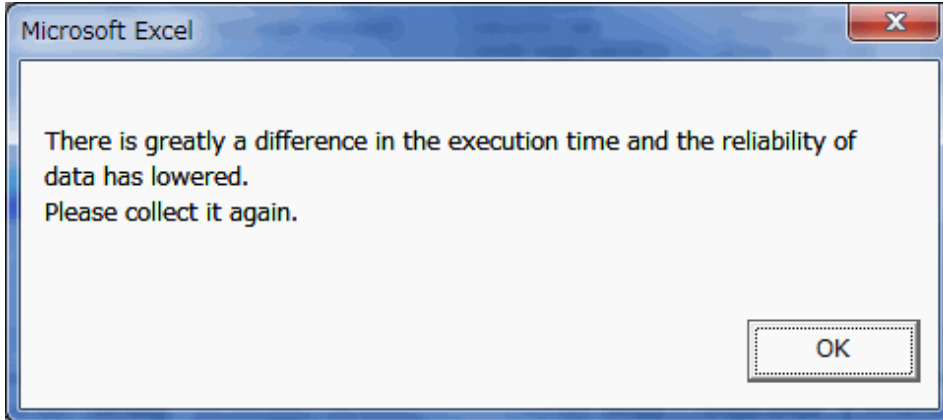
If there are no problems with the measurement range specification, Excel sheet generation starts.

Data collection is executed 11 times, and the execution time causes small differences. This difference is displayed in the lower part of the second Excel sheet.

A warning dialog is displayed if there is a difference of 5% or more either way (displayed as 95% or less, or as 105% or more) in the execution time compared with the first execution time.

If a high degree of precision is required, check this execution time. If there is a big difference, it is recommended to perform data collection again and use data that has small differences.

Figure 3.29 Warning dialog



3.5.4 Viewing the Excel sheets

The precision PA visibility function (Excel format) is comprised of the following information:

- Performance information
- Memory Cache information
- SIMD information
- Cache information
- Instruction information
- Balance information
- XFILL flag
- Time information

The Excel sheet displays information for 16 threads or less in the TOP sheet, and displays information for 32 threads or less in the TOP32 sheet. Please use the TOP32 sheet when you want to analyze data more than 17 threads.

The precision PA visibility function (Excel format) TOP sheet is organized as two pages of print output. The first page outputs Performance information, Memory Cache information, SIMD information, Cache information, Instruction information, Balance information, and XFILL flag. The second page displays Time information.

The precision PA visibility function (Excel format) TOP32 sheet is organized as four pages of print output.

The item cell of each measurement item with the possibility to be a bottleneck is displayed in pink.

The figure below shows the structure of the precision PA visibility function (Excel format) TOP sheet.

Figure 3.32 Performance information

Performance							
	Execution time (sec)	Floating-point operation peak rate	MFLOPS	MIPS	Integer operation performance (MINOPS)	Floating-point operation	Integer operation
Thread 0	0.23	69.33%	16640	2690	101	0.00E+00	2.28E+07
Thread 1	0.23	69.34%	16642	2690	101	0.00E+00	2.28E+07
Thread 2	0.23	69.32%	16636	2689	101	0.00E+00	2.28E+07
Thread 3	0.23	69.32%	16637	2689	101	0.00E+00	2.28E+07
Thread 4	0.23	69.31%	16635	2685	101	0.00E+00	2.26E+07
Thread 5	0.23	69.31%	16634	2684	100	0.00E+00	2.26E+07
Thread 6	0.23	69.30%	16632	2684	100	0.00E+00	2.25E+07
Thread 7	0.23	69.31%	16634	2684	100	0.00E+00	2.25E+07
Thread 8	0.23	69.30%	16633	2684	100	0.00E+00	2.26E+07
Thread 9	0.23	69.29%	16628	2684	100	0.00E+00	2.26E+07
Thread 10	0.23	69.29%	16630	2684	100	0.00E+00	2.26E+07
Thread 11	0.23	69.30%	16631	2684	100	0.00E+00	2.26E+07
Thread 12	0.23	69.29%	16630	2684	100	0.00E+00	2.26E+07
Thread 13	0.23	69.31%	16634	2684	100	0.00E+00	2.26E+07
Thread 14	0.23	69.30%	16632	2684	100	0.00E+00	2.26E+07
Thread 15	0.23	69.29%	16628	2681	97	0.00E+00	2.18E+07
Process	0.23	69.29%	266087	42957	1604	0.00E+00	3.61E+08

The items below are displayed.

Table 3.48 Performance information items output

Output item	Explanation of output item
Execution time (sec)	Displays the execution time for each thread. The execution time of the thread having the longest execution in the process line is displayed.
Floating-point operation peak rate	Displays the rate of the actual measured value against the MFLOPS logical peak value (%).
MFLOPS	Displays the floating point operation execution rate.
MIPS	Displays the instruction execution rate.
Integer operation performance (MINOPS)	Displays the performance of the integer arithmetic.
Floating-point operation	Displays the total number of floating point operations.
Integer operation	Displays the total number of integer operations. The instruction that calculates the memory address is included in the integer operand.

3.5.4.2 Memory Cache information

The Memory cache information displays the memory throughput and other information.

Figure 3.33 Memory Cache information

Memory Cache				
L1 busy rate	L2 busy rate	Memory busy rate	L2 throughput (GB/sec)	Memory throughput (GB/sec)
33%	1%	0%	0.23	0.00
38%			0.23	0.00
33%			0.23	0.00
37%			0.12	0.00
33%			0.34	0.00
39%			0.51	0.00
39%			0.23	0.00
38%			0.23	0.00
33%			0.23	0.00
38%			0.23	0.00
38%			0.23	0.00
33%			0.23	0.00
38%			0.23	0.00
38%			0.23	0.00
39%			0.23	0.00
39%			3.96	0.00

The items below are displayed.

Table 3.49 Memory Cache information items output

Output item	Explanation of output item
L1 busy rate (*1)	Displays the L1 engine busy rate (%).
L2 busy rate (*2)	Displays the L2 busy rate (%).
Memory busy rate	Displays the Memory busy rate (%).
L2 throughput (GB/sec)	Displays the value of the L2 throughput.
Memory throughput (GB/sec)	Displays the value of the memory throughput.

*1 : "L1 busy rate" is a capacity usage rate of the device that controls the forwarding between registers <->L1D and the forwarding between L1D <-> L2.

*2 : "L2 busy rate" is a capacity usage rate of the device that controls the forwarding between L1D <-> L2 and the forwarding between the L2 <-> memories.

3.5.4.3 SIMD information

The SIMD information displays the SIMD instruction rate and other information.

Figure 3.34 SIMD information

SIMD				
	SIMD instruction rate (/Effective instruction)	SIMD floating point instruction rate (/SIMD target floating point instruction)	SIMD integer instruction rate (/SIMD target integer instruction)	SIMD load-store instruction rate (/SIMD target load-store instruction)
Process	94.74%	99.76%	0.00%	97.41%

The items below are displayed.

Table 3.50 SIMD information items output

Output item	Explanation of output item
SIMD instruction rate (/Effective instruction)	Displays the rate (%) of SIMD instructions amongst the total number of effective instructions
SIMD floating point instruction rate (/SIMD target floating point instruction)	Displays the rate (%) of floating point instructions changed to SIMD amongst the number of floating point instructions to be changed to SIMD
SIMD integer instruction rate (/SIMD target integer instruction)	Displays the rate (%) of integer instructions changed to SIMD amongst the number of integer instructions to be changed to SIMD
SIMD load-store instruction rate (/SIMD target load-store instruction)	Displays the rate (%) of load-store instructions changed to SIMD amongst the number of load-store instructions to be changed to SIMD

3.5.4.4 Cache information

The cache information displays the cache miss rate and other information.

Figure 3.35 Cache information

Cache	L1I miss rate (/Effective instruction)	L1D miss rate (/Load-store instruction)	Load-store instruction	L1D miss	L1D miss dm rate (/L1D miss)	L1D miss hwpf rate (/L1D miss)	L1D miss swpf rate (/L1D miss)	L2 miss rate (/Load-store instruction)	L2 miss	L2 miss dm rate (/L2 miss)	L2 miss pfrate (/L2 miss)	uDTLB miss rate (/Load-store instruction)	mDTLB miss rate (/Load-store instruction)
Thread 0	0.00%	0.05%	4.23E+08	2.04E+05	99.69%	0.30%	0.00%	0.00%	3.95E+02	66.84%	33.16%	0.00151%	0.00000%
Thread 1	0.00%	0.05%	4.23E+08	2.02E+05	99.84%	0.16%	0.00%	0.00%	2.72E+02	76.10%	23.90%	0.00130%	0.00000%
Thread 2	0.00%	0.05%	4.23E+08	2.02E+05	99.82%	0.18%	0.00%	0.00%	1.41E+02	74.47%	25.53%	0.00130%	0.00000%
Thread 3	0.00%	0.02%	4.23E+08	1.02E+05	99.87%	0.33%	0.00%	0.00%	5.60E+01	87.50%	12.50%	0.00130%	0.00000%
Thread 4	0.00%	0.07%	4.23E+08	3.00E+05	99.90%	0.10%	0.00%	0.00%	6.30E+01	82.54%	17.46%	0.00128%	0.00000%
Thread 5	0.00%	0.11%	4.23E+08	4.48E+05	99.94%	0.06%	0.00%	0.00%	8.60E+01	80.23%	19.77%	0.00127%	0.00000%
Thread 6	0.00%	0.05%	4.23E+08	2.02E+05	99.87%	0.13%	0.00%	0.00%	1.04E+02	73.08%	26.92%	0.00126%	0.00000%
Thread 7	0.00%	0.05%	4.23E+08	2.02E+05	99.81%	0.13%	0.00%	0.00%	6.70E+01	80.60%	19.40%	0.00127%	0.00000%
Thread 8	0.00%	0.05%	4.23E+08	2.02E+05	99.76%	0.24%	0.00%	0.00%	8.40E+01	71.43%	28.57%	0.00129%	0.00000%
Thread 9	0.00%	0.05%	4.23E+08	2.02E+05	99.86%	0.14%	0.00%	0.00%	7.00E+01	78.57%	21.43%	0.00129%	0.00000%
Thread 10	0.00%	0.05%	4.23E+08	2.02E+05	99.86%	0.14%	0.00%	0.00%	6.40E+01	80.63%	3.38%	0.00129%	0.00000%
Thread 11	0.00%	0.05%	4.23E+08	2.02E+05	99.85%	0.15%	0.00%	0.00%	9.10E+01	72.53%	27.47%	0.00128%	0.00000%
Thread 12	0.00%	0.05%	4.23E+08	2.02E+05	99.80%	0.20%	0.00%	0.00%	8.90E+01	58.43%	41.57%	0.00129%	0.00000%
Thread 13	0.00%	0.05%	4.23E+08	2.02E+05	99.79%	0.20%	0.00%	0.00%	7.90E+01	81.01%	18.99%	0.00128%	0.00000%
Thread 14	0.00%	0.05%	4.23E+08	2.02E+05	99.90%	0.10%	0.00%	0.00%	1.10E+02	64.55%	35.45%	0.00130%	0.00000%
Thread 15	0.00%	0.05%	4.23E+08	2.02E+05	99.86%	0.14%	0.00%	0.00%	7.90E+01	67.09%	32.91%	0.00130%	0.00000%
Process	0.00%	0.05%	6.76E+08	3.48E+06	99.94%	0.16%	0.00%	0.00%	1.85E+03	73.24%	26.76%	0.00130%	0.00000%

The items below are displayed.

Table 3.51 Cache information items output

Output item	Explanation of output item
L1I miss rate (/Effective instruction)	Displays the rate (%) of level 1 instruction cache misses in the total number of effective instructions
L1D miss rate (/Load-store instruction)	Displays the rate (%) of level 1 data cache misses in the load/store instructions count
Load-store instruction	Displays the total number of load/store instructions
L1D miss	Displays the total number of level 1 data cache misses
L1D miss dm rate (/L1D miss)	Displays the rate (%) of level 1 data cache misses from load/store instructions in the level1 data cache misses
L1D miss hwpf rate (/L1D miss)	Displays the rate (%) of level 1 data cache misses from hardware prefetch in the level1 data cache misses
L1D miss swpf rate (/L1D miss)	Displays the rate (%) of level 1 data cache misses from software prefetch instructions in the level1 data cache misses
L2 miss rate (/Load-store instruction)	Displays the rate (%) of level 2 cache misses in the load/store count
L2 miss	Displays the total number of level 2 cache misses
L2 miss dm rate (/L2 miss)	Displays the rate (%) of level 2 cache demand misses
L2 miss pf rate (/L2 miss)	Displays the rate (%) of level 2 cache prefetch misses
uDTLB miss rate (/Load-store instruction)	Displays the micro data TLB miss rate (%)

Output item	Explanation of output item
Effective instruction rate	Displays the total of the above. In principle, this is 100%.

*1 : DSP is abbreviation of Dual Single Precision. The DSP operation instruction is an instruction that delimits the double precision register to the half and treats as two floating point of single precision data.

*2 : concatenate shift left instruction is an instruction to which effective all elements of the floating point of double precision register are connected and the left shifts.

3.5.4.6 Balance information

The balance information displays the load balance and other information.

Figure 3.37 Balance information

Balance		
	Load balance	Instruction balance
Thread 0	97%	100%
Thread 1	97%	100%
Thread 2	97%	100%
Thread 3	97%	100%
Thread 4	98%	100%
Thread 5	99%	100%
Thread 6	99%	100%
Thread 7	99%	100%
Thread 8	98%	100%
Thread 9	98%	100%
Thread 10	99%	100%
Thread 11	98%	100%
Thread 12	98%	100%
Thread 13	98%	100%
Thread 14	98%	100%
Thread 15	99%	100%

The items below are displayed.

Table 3.53 Balance information items output

Output item	Explanation of output item
Load balance	Displays comparison of the time of "Cycle_counts-barrier synchronization waiting" of each thread (rate to thread of the longest execution time).
Instruction balance	Displays the comparison of total number of effective instructions of each thread (rate to thread with most numbers of instructions).

3.5.4.7 XFILL flag

Please set 1(ON) when you are using XFILL instruction.

The default is 0(OFF).

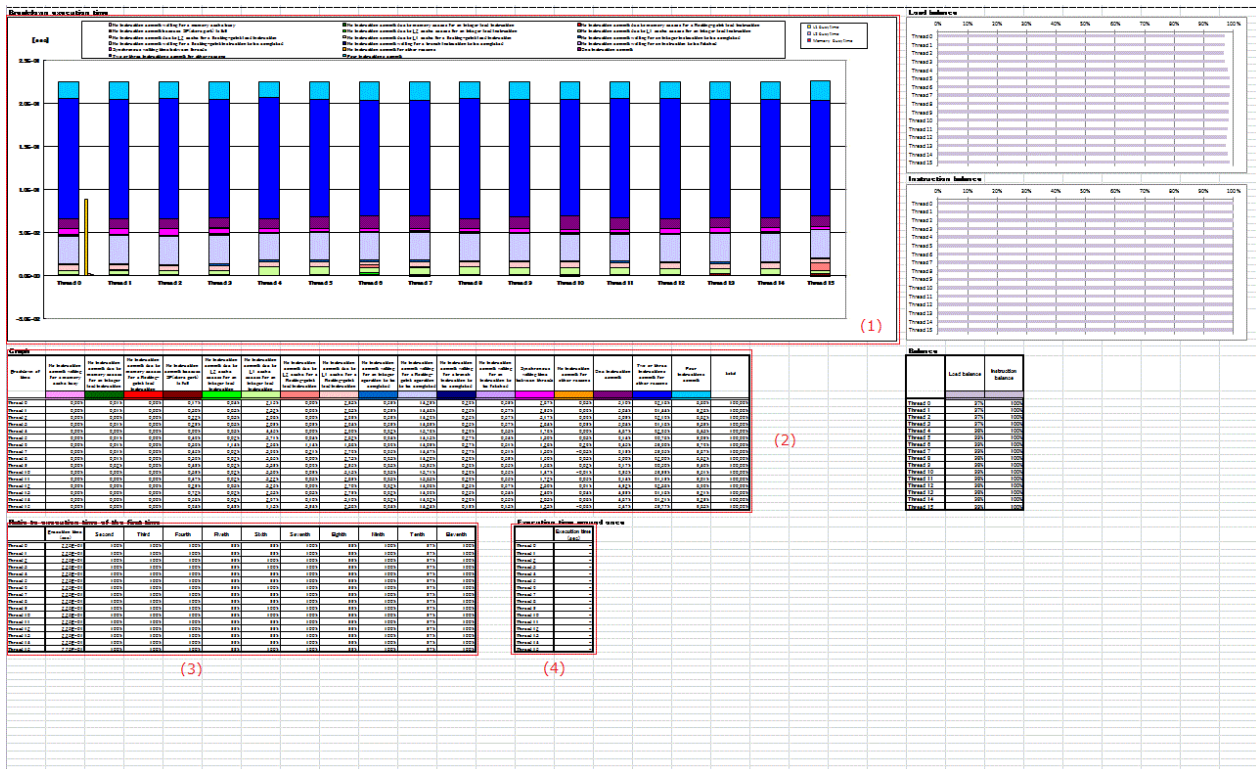
Figure 3.38 XFILL flag

XFILL flag	
0	0:OFF 1:ON

3.5.4.8 Time information

The second page of the print output is the time information.

Figure 3.39 Time information



(1) Graphs

The bar graphs of the time information are displayed for each thread.

L1 busy time, L2 busy time, and memory busy time are displayed at the right of the bar chart of thread 0.

Busy time is an indicator made a busy rate easy to compare than an actual execution time.

L1 busy time [second] is the maximum value of L1 busy rate of thread i * execution time of thread i . (i is a thread number.)

L2 busy time [second] is a value of calculating L2 busy rate * execution time of process.

Memory busy time [second] is a value of calculating memory busy rate * execution time of process.

Refer to "Table 3.54 Time information items and graph element detail items output" for details of the element on the bar graphs.

Refer to "3.5.4.2 Memory Cache information" for the detail of L1 busy rate, L2 busy rate, and memory busy rate.

(2) Graph element details

The graph element details display the times and rates (%) of the elements that make up the graphs.

The items below are displayed.

Table 3.54 Time information items and graph element detail items output

Output item	Explanation of output item
memory cache busy wait	Indicates the number of cycles where the number of completion instructions is 0 due to waits caused by memory cache busy
Integer load memory access wait	Indicates the number of cycles where the number of completion instructions is 0 due to data waits caused by integer load memory access by the oldest instruction amongst the instructions currently being executed

Output item	Explanation of output item
Floating point load memory access wait	Indicates the number of cycles where the number of completion instructions is 0 due to data waits caused by floating point load memory access by the oldest instruction amongst the instructions currently being executed
Store wait	Indicates the number of cycles where the number of completion instructions is 0 due to waits caused by no space in the store port
Integer load L2 cache access wait	Indicates the number of cycles where the number of completion instructions is 0 due to data waits caused by integer load L2 cache access by the oldest instruction amongst the instructions currently being executed
Integer load L1D cache access wait	Indicates the number of cycles where the number of completion instructions is 0 due to data waits caused by integer load L1D cache access by the oldest instruction amongst the instructions currently being executed
Floating-point load L2 cache access wait	Indicates the number of cycles where the number of completion instructions is 0 due to data waits caused by floating point load L2 cache access by the oldest instruction amongst the instructions currently being executed
Floating-point load L1D cache access wait	Indicates the number of cycles where the number of completion instructions is 0 due to data waits caused by floating point load L1D cache access by the oldest instruction amongst the instructions currently being executed
Integer operation wait	Indicates the number of cycles where the number of completion instructions is 0 because the oldest instruction amongst the instructions currently being executed is currently executing an integer operation
Floating-point operation wait	Indicates the number of cycles where the number of completion instructions is 0 because the oldest instruction amongst the instructions currently being executed is currently executing a floating point operation
Branch instruction wait	Indicates the number of cycles where the number of completion instructions is 0 because the oldest instruction amongst the instructions currently being executed is currently executing a Branch instruction
Instruction fetch wait	Indicates the number of cycles where the number of completion instructions is 0 because the CSE is empty The CSE is the buffer used to hold information for the instruction currently being executed (issued but not yet completed)
Barrier synchronization wait	Indicates the number of cycles where the number of completion instructions is 0 as a result of the oldest instruction amongst the instructions currently being executed using a SLEEP instruction to stop the instruction control part
Other waits	Indicates the processing of waits other than the above
1 instruction commit	Indicates number of cycles having one completion instruction
2/3 instruction commit (other)	Indicates number of cycles having two or three completion instruction
4 instruction commit	Indicates number of cycles having four completion instruction

(3) Comparison with first execution times

Display the comparison between the 11 lots of execution times performed to complete the Excel sheets

The closer the whole is to 100%, the more precise the information.

A warning dialog is displayed if there is a difference of 5% or more either way (displayed as 95% or less, or as 105% or more) in the execution time compared with the first execution time. Even after the warning dialog is closed, the cells are displayed in red.

Refer to "[3.5.3.2.4 Generating Excel sheets](#)" for information on the warning dialog.

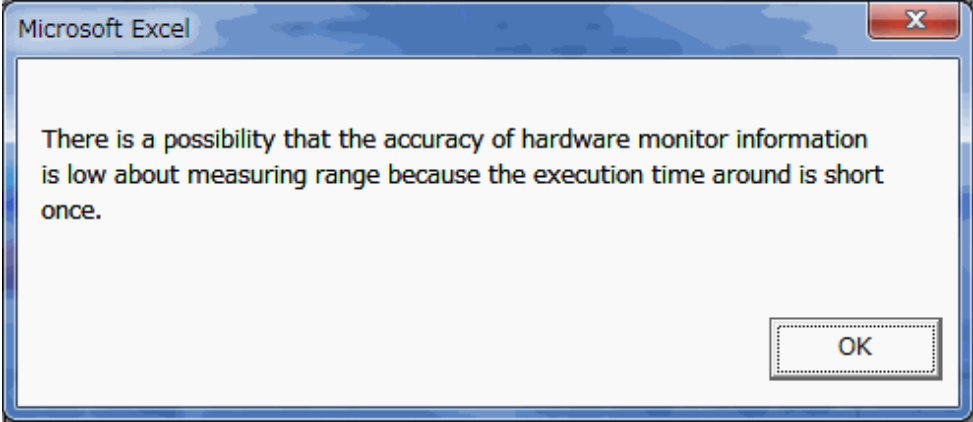
(4) Execution times for one occurrence

Display the execution times for one occurrence of the specified segment name (measurement range).

If the execution times for one occurrence are 150 microseconds or less, a warning dialog is displayed to notify that the precision of the Hardware monitor information is low.

Even after the warning dialog is closed, the cells are displayed in red.

Figure 3.40 Warning dialog when the execution times for one occurrence are low



Chapter 4 Tracer

This chapter describes the features and usage of the Tracer.

4.1 Overview of the Tracer

4.1.1 Overview of features

The Tracer collects the execution information on the function that the user called in the source code in the time series.

- The MPI library for high-end, technical computing server PRIMEHPC FX100 system
- User function
- I/O function defined in standard C library
- Memory function defined in standard C library

The execution information means the following, the performance of the application can be improved by tuning based on these information.

- start/end time of the MPI library and the wire traffic
- start/end time of the user function
- start/end time of the I/O function and number of bytes for data read/write
- start/end time of the Memory function and the number of bytes for allocated memory

The following features are provided by the Tracer.

VampirTrace is used for the realization of these features.

4.1.1.1 Information collection feature

The execution information for an application is collected. It is the information on the function that is called in the source code. The order of calling functions is arranged according to the time series, and the information regarding the start and end time of each function is maintained.

4.1.1.2 Local trace data files integration feature

Local trace data files are integrated, and a trace data file is created. Local trace data files signify the information collected by the information collection feature. Refer to "[4.2.3.5 Trace data files](#)" for information on local trace data files. The Vampir parallel program performance analysis tool can give GUI displays of the trace data file. Refer to "<http://www.vampir.eu>" for information on Vampir.

4.1.2 Preparation for using the Tracer

The FX100 system prepares the MPI described with the C language, the C++ language, or Fortran and translation/uniting command that one-by-one translates a multithreaded and a hybrid program, unites, and makes an executable program for the FX100 system. There are two kinds of translation/uniting command of own compiler (vtfcc/vtFCC/vtfrt) that operates by cross compiler (vtfccpx/vtFCCpx/vtfrtpx) and the compute node that operates by the login node in the front end.

Use this translation/uniting command according to the kind of MPI and each language that one-by-one describes a multithreaded and a hybrid program. Refer to "[4.2.1 Compilation](#)" for information on vtfccpx/vtFCCpx/vtfrtpx/vtfcc/vtFCC/vtfrt commands.

Set the following environment before using the Tracer.

4.1.2.1 Compilation/Integration environment

It is necessary to set the following environment variable on the login node at the front end to use the compilation feature of the Tracer by the vtfccpx/vtFCCpx/vtfrtpx (vtfcc/vtFCC/vtfrt for own compiler) commands.

Environment variable	Value
PATH	/opt/FJSVmxlang/bin

In addition to the above mentioned variable, it is necessary to set the following environment variable to use the integration feature on the login node at the front end by the vtunifypx command.

Environment variable	Value
LD_LIBRARY_PATH	/opt/FJSVmxlang/lib

Refer to "[4.2.3 Local trace data file integration feature](#)" for information on the vtunifypx command.

Additionally, the following settings besides those mentioned above are necessary:

- Environment variables, PATH and LD_LIBRARY_PATH, for using the MPI system
Refer to the "MPI User's Guide".
- Environment variables, PATH and LD_LIBRARY_PATH, for using the compiler and the library of each language
Refer to the relevant user's guide.

4.1.2.2 Compilation/Execution/Integration environment

When a job is turned on, it is necessary to specify the following environment variables in the job script to use the integration feature by execution and the vtunify-mpi application of the application (executable program) made by using the execution of the application (executable program) made by same command of the Tracer by the vtfcc/vtFCC/vtfrt command as the translation function,vtfccpx/vtFCCpx/vtfrtpx command.

Environment variable	Value
PATH	/opt/FJSVmxlang/bin
LD_LIBRARY_PATH	/opt/FJSVmxlang/lib64

Refer to "[4.2.3 Local trace data file integration feature](#)" for information on the vtunify-mpi command.

Refer to the "Job Operation Software First Step Guide" for information on job submission and the job script.

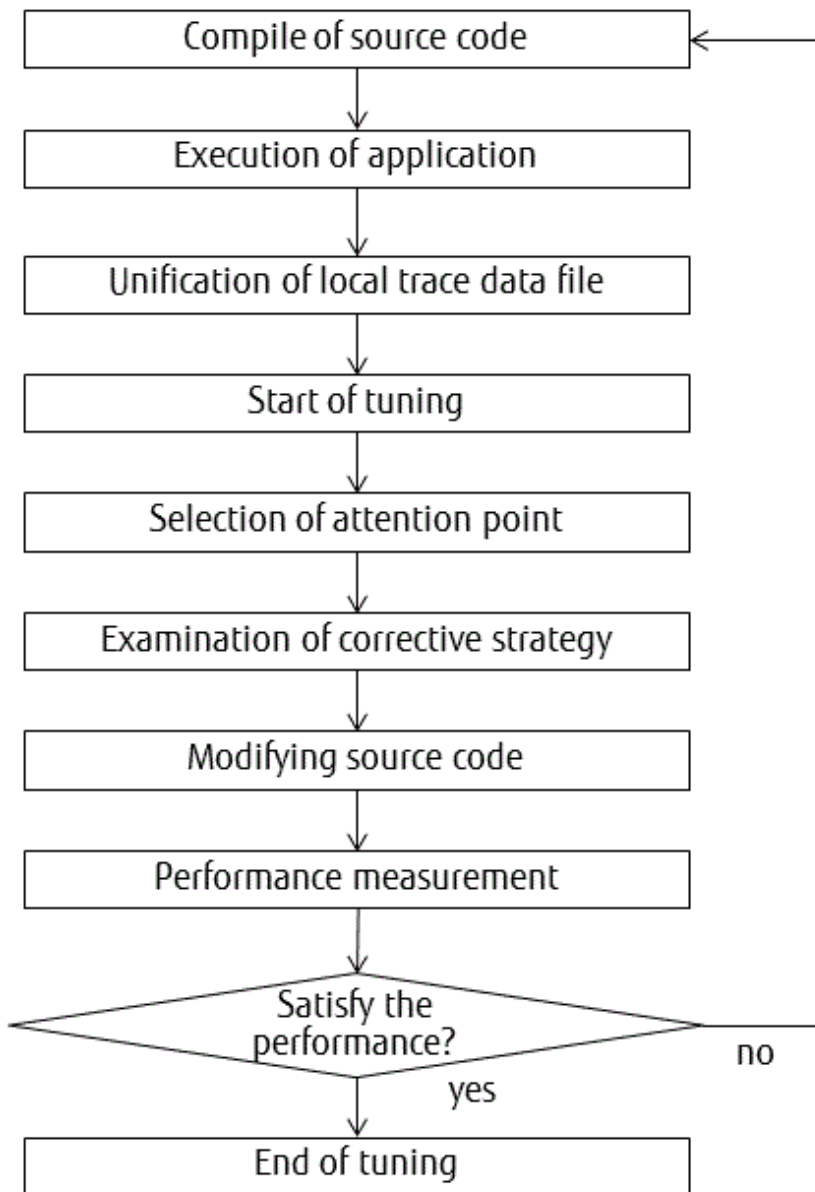
Additionally, the following settings besides those mentioned above are necessary:

- Environment variables, PATH and LD_LIBRARY_PATH, for using the MPI system
Refer to the "MPI User's Guide".
- Environment variables, PATH and LD_LIBRARY_PATH, for using the compiler and the library of each language
Refer to the relevant user's guide.

4.1.3 Flow for using the Tracer

The following figure shows the flow for using the Tracer.

Figure 4.1 Flow for using the Tracer



4.1.3.1 Compilation

An application (executable program) is created using the `vtfccpx/vtFCCpx/vtfrtpx` commands or the `vtfcc/vtFCC/vtfrt` commands.



Example

Compilation of the C language program "sample.c"

```
$ vtfccpx sample.c
```

Refer to "4.2.1 Compilation" for details.

4.1.3.2 Information collection

The application created by the `vtfccpx/vtFCCpx/vtfrtpx` commands or the `vtfcc/vtFCC/vtfrt` command is executed, and local trace data files are generated.

Local trace data files have the "def", "events", or "uctl" file name extensions.

The def and events files are created for each rank or thread.

Refer to "4.2.3.5 Trace data files" for information on each local trace data file. Refer to "4.2.2 Information collection" for details.



Example

- Execution by 2 processes and 4 threads of the application program "a.out"

```
$ mpiexec -n 2 ./a.out
```

- List of Local trace data files

Rank 0			Rank 1			Description
Process	Thread		Process	Thread		
a.1.def.z	0	-	a.2.def.z	0	-	List file
	1	a.10001.def.z		1	a.20001.def.z	
	2	a.10002.def.z		2	a.20002.def.z	
	3	a.10003.def.z		3	a.20003.def.z	
a.1.events.z	0	-	a.2.events.z	0	-	Time series record file
	1	a.10001.events.z		1	a.20001.events.z	
	2	a.10002.events.z		2	a.20002.events.z	
	3	a.10003.events.z		3	a.20003.events.z	
a.uctl						Unify control file

Information regarding thread 0 is included in the process file, so the thread 0 file does not exist.

4.1.3.3 Local trace data files integration

Vampir of the parallel program performance analysis tool reads the file with the otf file name extension as an input file, and starts drawing. Perform tuning by analyzing the performance with Vampir. The otf file is one of the trace data files created by the local trace data files integration feature.

Therefore, it is necessary to integrate local trace data files after they are created, and to create the trace data files. Local trace data files are integrated using the vtunify-mpi application that operates on the vtunifypx command or the compute node operating on the login node at the front end for making, and the trace data file is created.

Refer to "Table 4.1 Created trace data files" for an example of the trace data files created for two processes and four threads.

The trace data files with the file name extensions of def, otf, and events are created for each rank thread.

Refer to "4.2.3.5 Trace data files" for information on each trace data file.

Refer to "4.2.3 Local trace data file integration feature" for information on the vtunifypx command and the vtunify-mpi application.

Table 4.1 Created trace data files

File name	Description
a.0.def.z	List file
a.1.events.z	Time series recorded file of rank 0 and thread 0
a.2.events.z	Time series recorded file of rank 1 and thread 0
a.10001.events.z	Time series recorded file of thread 1 to rank 0
a.10002.events.z	Time series recorded file of thread 2 to rank 0
a.10003.events.z	Time series recorded file of thread 3 to rank 0

File name	Description
a.20001.events.z	Time series recorded file of thread 1 to rank 1
a.20002.events.z	Time series recorded file of thread 2 to rank 1
a.20003.events.z	Time series recorded file of thread 3 to rank 1
a.otf	Open trace format file (otf file)

4.2 Using the Tracer

This section explains compilation and the local trace data file integration feature using the Tracer.

4.2.1 Compilation

The Fujitsu compiler of fccpx/FCCpx/frtpx (fcc/FCC/frt for own compiler) is used for the compilation of a sequential program, and the Fujitsu compiler of mpifccpx/mpiFCCpx/mpifrtpx (mpifcc/mpiFCC/mpifrt for own compiler) is used for the compilation of MPI programs.

However, when a part of the include specification and the Fujitsu compiler of the header file of VampirTrace is translated in addition to the call of each Fujitsu compiler, it is necessary to specify some additional options when the Tracer is used and compiled. The wrapper command of the Fujitsu compiler is provided in the Tracer for compilation of programs so that users need not specify additional options. The Tracer provides the vtfccpx/vtFCCpx/vtfrtpx (vtfcc/vtFCC/vtfrt for own compiler) commands as the wrapper commands. When these commands are executed, the Fujitsu compiler of mpifccpx/mpiFCCpx/mpifrtpx (mpifcc/mpiFCC/mpifrt for own compiler) is internally called by default.

When a sequential program and a multithreaded program are compiled, it is necessary to change the compiler internally called to fccpx (fcc for own compiler).

The compiler internally called is optional, and can be changed by specifying the environment variable when compiling.

Refer to "4.2.1.2 Options" for information on the options. Refer to "4.2.1.4 Environment variables for compilation" for information on the environment variables used during compilation.

Users can use the vtfccpx/vtFCCpx/vtfrtpx (vtfcc/vtFCC/vtfrt for own compiler) command by a method similar to the Fujitsu compiler. As a result, users can compile programs by using the vtfccpx/vtFCCpx/vtfrtpx (vtfcc/vtFCC/vtfrt for own compiler) commands without considering various libraries used with the header file and VampirTrace of VampirTrace.

For compilation, it is necessary to specify the VT_PREFIX environment variable if the installation destination is changed from "/opt/FJSVmxlang".

Refer to "4.2.2.1 Environment variables for execution" for information on the environment variables used during execution.

The Fujitsu MPI compilers for each command are given below.

Table 4.2 Fujitsu MPI compilers

	Command	Compiler	Language
cross compiler	vtfccpx	mpifccpx	C
	vtFCCpx	mpiFCCpx	C++
	vtfrtpx	mpifrtpx	Fortran
own compiler	vtfcc	mpifcc	C
	vtFCC	mpiFCC	C++
	vtfrt	mpifrt	Fortran

4.2.1.1 Format

Cross compiler

```
{ vtfccpx | vtFCCpx | vtfrtpx }
[ -vt: { cc | cxx | f90 } cmd ]
[ -vt: { seq | mpi | mt | hyb } ]
```

```
[ -vt:inst { compinst | manual } ] [ -DVTRACE ]
[ -vt:version ] [ -vt: { verbose | show } ]
[ -vt:showme-compile ] [ -vt:showme-link ] [ -vt:help ]
[ compiler_arguments ]
file ...
```

Own compiler

```
{ vtfcc | vtFCC | vtfprt }
[ -vt: { cc | cxx | f90 } cmd ]
[ -vt: { seq | mpi | mt | hyb } ]
[ -vt:inst { compinst | manual } ] [ -DVTRACE ]
[ -vt:version ] [ -vt: { verbose | show } ]
[ -vt:showme-compile ] [ -vt:showme-link ] [ -vt:help ]
[ compiler_arguments ]
file ...
```

4.2.1.2 Options

The table below lists the options that can be specified for vtfccpx/vtFCCpx/vtfprtpx (vtfcc/vtFCC/vtfprt for own compiler).

Table 4.3 Options list

Option	Description
-vt:cc <i>cmd</i>	Changes the compiler, called internally by vtfccpx (vtfcc for own compiler), from mpifccpx (mpifcc for own compiler) to <i>cmd</i> . The possible value for <i>cmd</i> is fccpx (fcc for own compiler).
-vt:cxx <i>cmd</i>	Changes the compiler, called internally by vtFCCpx (vtFCC for own compiler), from mpiFCCpx (mpiFCC for own compiler) to <i>cmd</i> . The possible value for <i>cmd</i> is FCCpx (FCC for own compiler).
-vt:f90 <i>cmd</i>	Changes the compiler, called internally by vtfprtpx (vtfprt for own compiler), from mpifprtpx (mpifprt for own compiler) to <i>cmd</i> . The possible value for <i>cmd</i> is prtpx (prt for own compiler).
-vt:seq	Collects information of a sequential program The start and end time of a user function is collected.
-vt:mpi	Collects information of an MPI program The start and end time of an MPI function called in an MPI program, argument information, and the number of ranks is collected.
-vt:mt	Collects information of a multithreaded program Start/end time of the user function called in Start/end time of the user function called in the multithreaded program and the OpenMP instruction sentence is collected and the amounts of the number of threads are collected.
-vt:hyb	Collects information of a hybrid program The hybrid program is a parallel program that combines an MPI program with a multithreaded-parallel program. Start/end time of the user function called in start/end time of start/end time of the MPI function called in a hybrid program and argument information and the user function and the OpenMP instruction sentence is collected and the amounts of the number of threads are collected.
-vt:inst compinst	Trace user function Refer to " 4.3.2 User function trace " for information on the user function trace. It is the default value.
-vt:inst manual	Inhibits the trace function of a user function

Option	Description
	This option does not induce the "-g" option, and optimizes it effectively.
-DVTRACE	Trace VampirTrace API Refer to " 4.3.3 VampirTrace API trace " for information on the VampirTrace API.
-vt:verbose -vt:show	Displays the Fujitsu compiler that the vtfcpx/vtFCCpx/vtfrtpx (vtfcc/vtFCC/vtfrt for own compiler) called and the specified compiler options An application is created if "-vt:verbose" is specified, and if "-vt:show" is specified, an application is not created.
-vt:showme-compile	Displays the compiler options led from config file by tracer and arguments specified in command line (program name and so on). When this option is specified, an application is not created.
-vt:showme-link	Displays the linker options led from config file by tracer and arguments specified in command line (program name and so on). When this option is specified, an application is not created.
-vt:version	Outputs a version of VampirTrace
-vt:help	Outputs the help message
compiler_arguments	Specifies the options passed to the Fujitsu compiler by vtfcpx/vtFCCpx/vtfrtpx (vtfcc/vtFCC/vtfrt for own compiler) calls Refer to the manual of each Fujitsu compiler for information on the options that can be specified.

4.2.1.3 Operand

file

Specify the source code written in each language in *file*.

4.2.1.4 Environment variables for compilation

The operation of vtfcpx/vtFCCpx/vtfrtpx (vtfcc/vtFCC/vtfrt for own compiler) can be controlled by specifying environment variables during compilation. If an option and an environment variable functionally equivalent to the option are specified at the same time, the value of the option takes precedence. The environment variables for compilation are given below.

Table 4.4 Environment variables for compilation

Environment variable	Description
VT_INST	Same as the "-vt:inst" option. Either compinst or manual can be specified.
VT_CC	Same as the "-vt:cc" option. The value that can be specified is fcpx (fcc for own compiler).
VT_CXX	Same as the "-vt:cxx" option. The value that can be specified is FCCpx (FCC for own compiler).
VT_FC	Same as the "-vt:f90" option. The value that can be specified is frtpx (frt for own compiler).
VT_CFLAGS	Specify the option passed to the C compiler.
VT_CXXFLAGS	Specify the option passed to the C++ compiler.
VT_FCFLAGS	Specify the option passed to the Fortran compiler.
VT_LDFLAGS	Specify the option passed to the linker.
VT_LIBS	Specify the library passed to the linker.

4.2.1.5 Example of compilation

Examples of compiling a source code in each language are given below.

Example

C language

- MPI program

```
$ vtfccpx sample.c
```

Collects the start and end time of a user function, an MPI function, and the argument information called in an MPI program

- Sequential program

```
$ vtfccpx -vt:cc fccpx -vt:seq sample.c
```

Collects the start and end time of a user function called in a sequential program

- Multithreaded program

```
$ vtfccpx -Kopenmp -vt:cc fccpx -vt:mt sample.c
```

Start/end time of the user function called in start/end time of the user function called in the multithreaded program and the OpenMP instruction sentence is collected and the amounts of the number of threads are collected.

- Hybrid program

```
$ vtfccpx -Kopenmp -vt:hyb sample.c
```

Start/end time of the user function called in start/end time of start/end time of the MPI function called in a hybrid program and argument information and the user function and the OpenMP instruction sentence is collected and the amounts of the number of threads are collected.

C++ language

- MPI program

```
$ vtFCCpx sample.cc
```

Collects the start and end time of a user function, an MPI function, and the argument information called in an MPI program

- Sequential program

```
$ vtFCCpx -vt:cxx FCCpx -vt:seq sample.cc
```

Collects the start and end time of a user function called in a sequential program

- Multithreaded program

```
$ vtFCCpx -Kopenmp -vt:cxx FCCpx -vt:mt sample.cc
```

Start/end time of the user function called in Start/end time of the user function called in the multithreaded program and the OpenMP instruction sentence is collected and the amounts of the number of threads are collected.

- Hybrid program

```
$ vtFCCpx -Kopenmp -vt:hyb sample.cc
```

Start/end time of the user function called in start/end time of start/end time of the MPI function called in a hybrid program and argument information and the user function and the OpenMP instruction sentence is collected and the amounts of the number of threads are collected.

Fortran

- MPI program

```
$ vtftrtpx sample.f90
```

Collects the start and end time of a user function, an MPI function, and the argument information called in an MPI program

- Sequential program

```
$ vtftrtpx -vt:f90 frtpx -vt:seq sample.f90
```

Collects the start and end time of a user function called in a sequential program

- Multithreaded program

```
$ vtftrtpx -Kopenmp -vt:f90 frtpx -vt:mt sample.f90
```

Start/end time of the user function called in start/end time of the user function called in the multithreaded program and the OpenMP instruction sentence is collected and the amounts of the number of threads are collected.

- Hybrid program

```
$ vtftrtpx -Kopenmp -vt:hyb sample.f90
```

Start/end time of the user function called in start/end time of start/end time of the MPI function called in a hybrid program and argument information and the user function and the OpenMP instruction sentence is collected and the amounts of the number of threads are collected.



4.2.2 Information collection

The information collection feature executes applications created by the vtfccpx/vtFCCpx/vtftrtpx (vtfcc/vtFCC/vtftrt for own compiler) commands, and collects relevant information about the applications.

The Tracer can collect the following information:

- MPI trace (Refer to "4.3.1 MPI trace")
- User function trace (Refer to "4.3.2 User function trace")
- VampirTrace API trace (Refer to "4.3.3 VampirTrace API trace")
- I/O trace (Refer to "4.3.4 I/O trace")
- Memory trace (Refer to "4.3.5 Memory trace")

After an application is executed, the local trace data file of each rank thread is created. Refer to "4.2.3.5 Trace data files" for information on local trace data files.


When an application is executed, the collection operations can be controlled by specifying environment variables. Refer to "4.2.2.1 Environment variables for execution" for information on environment variables.

4.2.2.1 Environment variables for execution

The following table describes the environment variables that can be specified when an application is executed.

Table 4.5 Environment variables for execution

Environment variable	Description	Default value
VT_BUFFER_SIZE	Specifies the size of the buffer (unit of byte) where the time series collection information is recorded before it is written to the events file. If less than 100 KB, it is resized to 100 KB. A negative value cannot be specified.	32M
VT_CLEAN	Specifies whether to delete local trace data files temporarily output to "VT_PFORM_LDIR".	yes

Environment variable	Description	Default value
	Only "yes" or "no" can be specified.	
VT_COMPRESSION	Specifies whether to compresses a local trace data file when it is output; if compressed, a local trace data file is output with the file name extension of ".z". Only "yes" or "no" can be specified.	yes
VT_FILE_PREFIX	Adds a name at the top of local trace data files. The default is "a", or the application name specified by "-o" during translation.	a or application name
VT_FILE_UNIQUE	Prevents overwriting to a local trace data file. Specify "yes", "no", or a positive integer.	no
VT_GNU_NMFILE	Specifies the file that records the symbol information when the Tracer fails to acquire the symbol information of an application.  Example If an application is named "a.out": <pre>\$ nm --demangle --line-numbers a.out > a.nm \$ export VT_GNU_NMFILE="a.nm"</pre>	-
VT_MAX_FLUSHES	Specifies the upper bound of the frequency of the buffer flash. Either 0 or a positive value can be specified. It is considered that there is no upper bound in case 0 is specified.	1
VT_MAX_THREADS	Specifies the maximum number of threads per process for VampirTrace to secure a resource.	65536
VT_PFORM_GDIR	Specifies the directory that stores local trace data files.	Current directory
VT_PFORM_LDIR	Specifies the directory name that temporarily stores local trace data files.	Current directory
VT_THREAD_BUFFER_SIZE	Specifies the size (unit of byte) of the buffer per a thread to create the events file. If less than 100 KB, it is resized to 100 KB. If more than 1GB, it is resized to 1GB. A negative value cannot be specified. If you do not define this environment variable, as for the memory consumption of the tracer, 10% of the value specified with VT_BUFFER_SIZE is consumed. The calculating formula of the memory consumption is the following. - When VT_THREAD_BUFFER_SIZE is unspecified <pre>M = N * VT_BUFFER_SIZE * 0.7 + N * T * VT_BUFFER_SIZE * 0.1</pre> - When VT_THREAD_BUFFER_SIZE is specified <pre>M = N * VT_BUFFER_SIZE + N * T * VT_THREAD_BUFFER_SIZE</pre>	0

Environment variable	Description	Default value
	M : Value of total of memory allocation N : Number of processes T : Number of threads a process	
VT_VERBOSE	Specifies the level of the information message related to VampirTrace. A value of 0, 1, or 2 can be specified. - "Quiet: 0" or "Critical: 1" If 0 or 1 is specified, nothing is output. - "Information: 2" The making situation of the local trace data file is output.	1
VT_MPITRACE	Specifies whether an MPI function is traced. If "no" is specified, the information on the MPI function is not recorded in the events file. Only "yes" or "no" can be specified. Refer to " 4.3.1 MPI trace " for information on the MPI trace.	yes
VT_IOTRACE	Specifies whether the I/O function is traced. If "no" is specified, the information on the I/O function is not recorded in the events file. Only "yes" or "no" can be specified. Refer to " 4.3.4 I/O trace " for information on the I/O trace.	no
VT_MEMTRACE	Specifies whether the memory function is traced. If "no" is specified, the information on the memory function is not recorded in the events file. Only "yes" or "no" can be specified. Refer to " 4.3.5 Memory trace " for information on the memory trace.	no
VT_SYNC_FLUSH	Specifies whether the buffer flash synchronization is effectively done.	no
VT_SYNC_FLUSH_LEVEL	Specifies the minimum buffer level for the buffer flash synchronization in rate. A value from 0 through 100 can be specified.	80
VT_MAX_STACK_DEPTH	Specifies the maximum number of stack levels to be traced. 0 means unrestricted.	0
VT_PREFIX	Specifies the directory after the change if the installation destination of the Tracer is changed from "/opt/FJSVmxlang".	-

4.2.3 Local trace data file integration feature

This section explains the local trace data files integration feature.

This feature integrates local trace data files by using the vtunifypx command or the vtunify-mpi application after an application is executed, and the local trace data file is gathered of making by the vtfccpx/vtFCCpx/vtfrtpx commands (vtfcc/vtFCC/vtfrt for own compiler).

The vtunifypx command or the vtunify-mpi application is a tool that integrates local trace data files created for each rank, and generates the trace data file.

The vtunifypx command operates at the front end. The operation of the vtunifypx command should have the local trace data file of all ranks. As the vtunifypx command does not operate at the compute node, the CPU resource of the compute node is not consumed. The vtunify-mpi application operates by MPI parallel version on the compute node. The operation of the vtunify-mpi application should have

a local trace data file for each rank in each compute node. It is possible to say for high parallel execution because it integrates it more high-speed than vtunifypx though CPU resource of the compute node is consumed so that the vtunify-mpi application may operate by the compute node.

The method for creating trace data at the front end is shown in "Figure 4.2 Flow for using vtunifypx", and the method for creating trace data by the compute node is shown in "Figure 4.3 Flow for using vtunify-mpi".

Figure 4.2 Flow for using vtunifypx

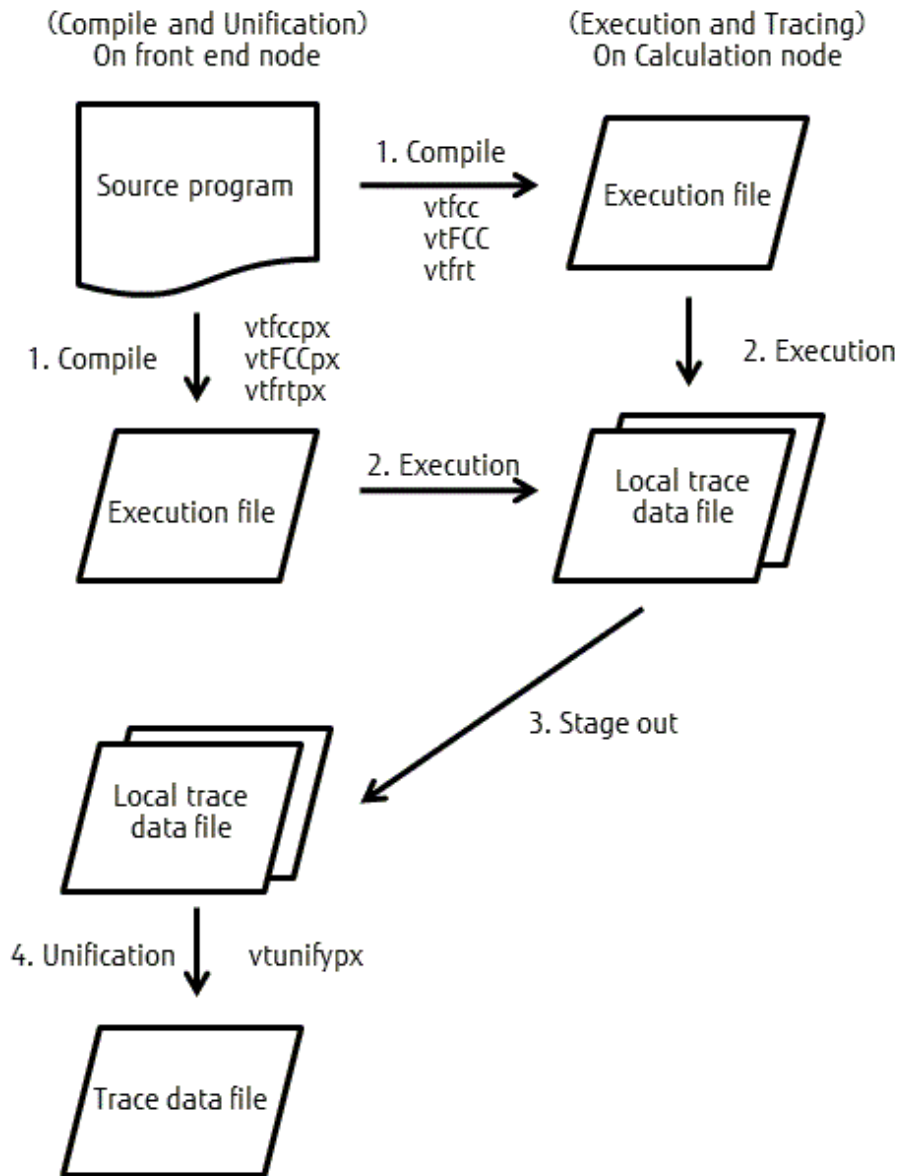
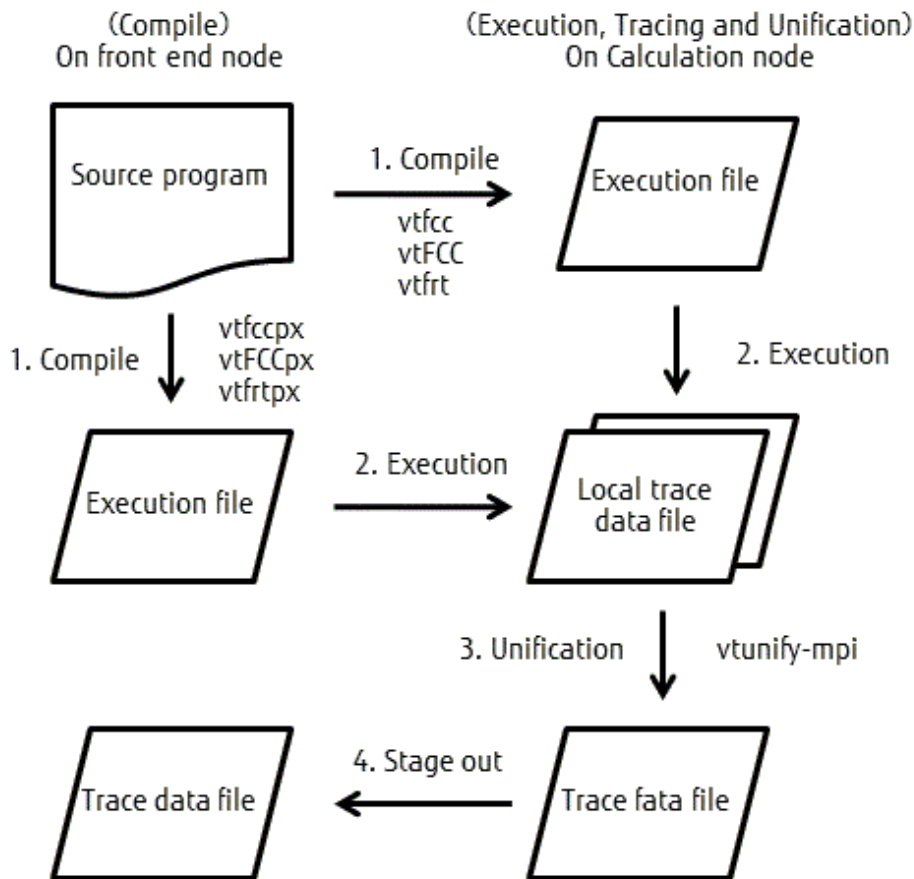


Figure 4.3 Flow for using vtunify-mpi



4.2.3.1 Format

vtunifypx

```
vtunifypx iprefix
[-o trace_filename] [{"-h" | "--help"}] [{"-V" | "--version"}] [--nocompress]
[{"-k" | "--keeplocal"}] [{"-p" | "--progress"}] [{"-q" | "--quiet"}] [{"-v" | "--verbose"}]
```

vtunify-mpi

```
vtunify-mpi iprefix
[-o trace_filename] [{"-h" | "--help"}] [{"-V" | "--version"}] [--nocompress]
[{"-k" | "--keeplocal"}] [{"-q" | "--quiet"}] [{"-v" | "--verbose"}]
```

4.2.3.2 Operand

iprefix

This is the name of the local trace data file.

For example, if an application named "sample" is executed, the name of the local trace data file starts with "sample".

In this case, "sample" is specified for *iprefix*.

Moreover, if the name of an application is "sample.out" or "sample.exe", "sample" is specified for *iprefix*.

If the storage location of a local trace data file is specified by the VT_PFORM_GDIR environment variable when an application is executed, the storage directory name, the slash ("/"), and the local trace data file name are specified for *iprefix*.

Refer to "[4.2.2.1 Environment variables for execution](#)" for information on environment variables.

4.2.3.3 Options

The options that can be specified for the vtunifypx command or the vtunify-mpi application are described in "Table 4.6 Options".

Table 4.6 Options

Option	Description
-o <i>trace_filename</i>	Assigns a unique value to the first name of the trace data file
-h or --help	Outputs the help message
-V or --version	Outputs the version of VampirTrace
--nocompress	Does not compress the def or events file Refer to "4.2.3.5 Trace data files" for information on the def and events files. The option takes precedence over the VT_COMPRESSION environment variable.
-k or --keeplocal	Creates an oft file without deleting the def, events, or uclt files If this option is specified, the def, the events, and the oft file that starts with "u_" are output.
-p or --progress	Displays the progress of the vtunifypx command This option is only valid for the vtunifypx command.
-q or --quiet	Controls the output of the execution log This option provides the same functionality as the environment variable VT_VERBOSE=0 or 1. The option takes precedence over the VT_VERBOSE environment variable. As for the priority level with "-v", the option specified at the end is given priority.
-v or --verbose	Outputs the execution log This option has the same functionality as the environment variable VT_VERBOSE=2. The option takes precedence over the VT_VERBOSE environment variable. As for the priority level with "-q", the option specified at the end is given priority.

4.2.3.4 Example of execution

An execution example is given below.



Example

- Example 1: Consider an application named "sample.out" where local trace data files that start with "sample" exist in the current directory.

- Example 1-1: When local trace data files of all ranks exist at the front end.

```
$ vtunifypx sample
```

- Example 1-2: When the local trace data file of each rank exists at each compute node after executing 1000 in parallel.

```
$ mpiexec -n 1000 vtunify-mpi sample
```

- Example 2: Consider an application named "sample.out", and two is executed in parallel, and the local trace data file that starts with "sample" exists at a location other than the current directory.

- Example 2-1: When the local trace data file of all ranks exists at the front end.

```
$ vtunifypx absolute_or_relative_path/sample
```

- Example 2-2: When the local trace data file of each rank exists at each compute node after executing 1000 in parallel.

```
$ mpiexec -n 1000 vtunify-mpi absolute_or_relative_path/sample
```

absolute_or_relative_path

The absolute or relative path of the directory where the local trace data file exists.

4.2.3.5 Trace data files

After an application is executed, the result of collecting the information is written to a file.

Moreover, the user creates the def, events, and the otf file using the vtunifypx command or the vtunify-mpi application.

These files are called trace data files.

A trace data file is usually output to the current directory. The output destination can be changed by VT_PFORM_GDIR.

Neither def nor the events file are compressed, and the file name extension of ".z" is removed if "no" is specified for VT_COMPRESSION.

The types and the names of trace data files that are created are given below.

Trace data files

DEF file

The whole definition file to function for each collection, user API, and the trace from which the identification number is distributed, and made a list respectively.

```
trace-filename_exec-count.0.def.z
```

EVENTS file

The trace result of the function called by the source program is recorded in the time series.

```
trace-filename_exec-count.parallel_or_thread-number.events.z
```

OTF file

This is the final file required for GUI display using Vampir.

```
trace-filename_exec-count.otf
```

Only local trace data files

UCTL file

This is a file necessary for the *.otf file creation.

```
trace-filename_exec-count.parallel-number.uctl
```

Trace data files and local trace data files commonness

LOCK file

This file records the execution frequency of an application when VT_FILE_UNIQUE=yes is specified.

```
trace-filename.lock
```

- *trace-filename*

This is the trace file name specified by the VT_FILE_PREFIX environment variable. The reference value becomes the application name specified by "-o" when compiling or "a".

- *exec-count*

If "yes" or an arbitrary numerical value is specified for the VT_FILE_UNIQUE environment variable, it becomes the execution frequency. The trace data file is overwritten and "_exec-count" is not added to the file name when VT_FILE_UNIQUE is not specified.

- *parallel_or_thread-number*

This is a parallel rank number or a thread number. It is expressed as a hexadecimal number.

- *parallel-number*

This is a parallel rank number. It is expressed as a hexadecimal number.

4.3 Trace information

4.3.1 MPI trace

The MPI trace collects the start time, the end time, and the argument information for an MPI function.

4.3.1.1 Compilation

The MPI trace compiles an MPI program using the `vtfccpx/vtFCCpx/vtfrtpx` (`vtfcc/vtFCC/vtfrt` for own compiler) command, and creates an application. A compilation example is given below.



Example

Compilation of the C language program "sample.c"

```
$ vtfccpx sample.c
```

4.3.1.2 Execution

Default need not consider and the user consider `VT_MPITRACE` specially for "yes" though he or she should specify "yes" for the `VT_MPITRACE` environment variable as for the MPI trace when executing it.

An execution example is given below.



Example

Execution by 2 processes of the application program "a.out"

```
$ mpiexec -n 2 ./a.out
```

4.3.1.3 MPI functions collected by the Tracer

MPI functions that the Tracer collects are given below.

The Tracer makes the function of MPI-1 and MPI-2 a collection object.

Refer to "[Table 4.7 MPI functions collected by the Tracer \(MPI 1\)](#)" for information on MPI-1.

Refer to "[Table 4.8 MPI functions collected by the Tracer \(MPI 2\)](#)" for information on MPI-2.

Table 4.7 MPI functions collected by the Tracer (MPI 1)

MPI_Abort	MPI_Error_string	MPI_Recv
MPI_Allgather	MPI_Finalize	MPI_Recv_init
MPI_Allgatherv	MPI_Gather	MPI_Reduce
MPI_Allreduce	MPI_Gatherv	MPI_Reduce_scatter
MPI_Alltoall	MPI_Get_count	MPI_Request_free
MPI_Alltoallv	MPI_Get_elements	MPI_Rsend
MPI_Attr_delete	MPI_Get_processor_name	MPI_Rsend_init
MPI_Attr_get	MPI_Graph_create	MPI_Scan
MPI_Attr_put	MPI_Graph_get	MPI_Scatter
MPI_Barrier	MPI_Graph_map	MPI_Scatterv
MPI_Bcast	MPI_Graph_neighbors	MPI_Send
MPI_Bsend	MPI_Graph_neighbors_count	MPI_Send_init

MPI_Bsend_init	MPI_Graphdims_get	MPI_Sendrecv
MPI_Buffer_attach	MPI_Group_compare	MPI_Sendrecv_replace
MPI_Buffer_detach	MPI_Group_difference	MPI_Ssend
MPI_Cancel	MPI_Group_excl	MPI_Ssend_init
MPI_Cart_coords	MPI_Group_free	MPI_Start
MPI_Cart_create	MPI_Group_incl	MPI_Startall
MPI_Cart_get	MPI_Group_intersection	MPI_Test
MPI_Cart_map	MPI_Group_range_excl	MPI_Test_cancelled
MPI_Cart_rank	MPI_Group_range_incl	MPI_Testall
MPI_Cart_shift	MPI_Group_rank	MPI_Testany
MPI_Cart_sub	MPI_Group_size	MPI_Testsome
MPI_Cartdim_get	MPI_Group_translate_ranks	MPI_Topo_test
MPI_Comm_compare	MPI_Group_union	MPI_Type_commit
MPI_Comm_create	MPI_Ibsend	MPI_Type_contiguous
MPI_Comm_dup	MPI_Init	MPI_Type_extent
MPI_Comm_free	MPI_Intercomm_create	MPI_Type_free
MPI_Comm_group	MPI_Intercomm_merge	MPI_Type_hindexed
MPI_Comm_rank	MPI_Iprobe	MPI_Type_hvector
MPI_Comm_remote_group	MPI_Irecv	MPI_Type_indexed
MPI_Comm_remote_size	MPI_Irsend	MPI_Type_lb
MPI_Comm_size	MPI_Isend	MPI_Type_size
MPI_Comm_split	MPI_Issend	MPI_Type_struct
MPI_Comm_test_inter	MPI_Keyval_create	MPI_Type_ub
MPI_Dims_create	MPI_Keyval_free	MPI_Type_vector
MPI_Errhandler_create	MPI_Op_create	MPI_Unpack
MPI_Errhandler_free	MPI_Op_free	MPI_Wait
MPI_Errhandler_get	MPI_Pack	MPI_Waitall
MPI_Errhandler_set	MPI_Pack_size	MPI_Waitany
MPI_Error_class	MPI_Probe	MPI_Waitsome

Table 4.8 MPI functions collected by the Tracer (MPI 2)

MPI_Accumulate	MPI_File_read_ordered_begin	MPI_Type_create_darray
MPI_Alltoallw	MPI_File_read_ordered_end	MPI_Type_create_f90_complex
MPI_Add_error_class	MPI_File_read_shared	MPI_Type_create_f90_integer
MPI_Add_error_code	MPI_File_seek	MPI_Type_create_f90_real
MPI_Alloc_mem	MPI_File_seek_shared	MPI_Type_create_hindexed
MPI_Comm_create_keyval	MPI_File_set_atomicity	MPI_Type_create_hvector
MPI_Comm_delete_attr	MPI_File_set_info	MPI_Type_create_indexed_block
MPI_Comm_free_keyval	MPI_File_set_size	MPI_Type_create_keyval
MPI_Comm_get_attr	MPI_File_set_view	MPI_Type_create_resized
MPI_Comm_get_name	MPI_File_sync	MPI_Type_create_struct

MPI_Comm_set_attr	MPI_File_write	MPI_Type_create_subarray
MPI_Comm_set_name	MPI_File_write_all	MPI_Type_delete_attr
MPI_Exscan	MPI_File_write_all_begin	MPI_Type_dup
MPI_File_close	MPI_File_write_all_end	MPI_Type_free_keyval
MPI_File_delete	MPI_File_write_at	MPI_Type_get_attr
MPI_File_get_amode	MPI_File_write_at_all	MPI_Type_get_contents
MPI_File_get_atomicity	MPI_File_write_at_all_begin	MPI_Type_get_envelope
MPI_File_get_byte_offset	MPI_File_write_at_all_end	MPI_Type_get_extent
MPI_File_get_group	MPI_File_write_ordered	MPI_Type_get_name
MPI_File_get_info	MPI_File_write_ordered_begin	MPI_Type_get_true_extent
MPI_File_get_position	MPI_File_write_ordered_end	MPI_Type_match_size
MPI_File_get_position_shared	MPI_File_write_shared	MPI_Type_set_attr
MPI_File_get_size	MPI_Free_mem	MPI_Type_set_name
MPI_File_get_type_extent	MPI_Get	MPI_Unpack_external
MPI_File_get_view	MPI_Get_version	MPI_Win_complete
MPI_File_iread	MPI_Grequest_start	MPI_Win_create
MPI_File_iread_at	MPI_Info_create	MPI_Win_create_keyval
MPI_File_iread_shared	MPI_Info_dup	MPI_Win_delete_attr
MPI_File_iwrite	MPI_Info_free	MPI_Win_fence
MPI_File_iwrite_at	MPI_Info_get_nkeys	MPI_Win_free
MPI_File_iwrite_shared	MPI_Info_get_nthkey	MPI_Win_free_keyval
MPI_File_open	MPI_Init_thread	MPI_Win_get_attr
MPI_File_preallocate	MPI_Is_thread_main	MPI_Win_get_group
MPI_File_read	MPI_Initialized	MPI_Win_get_name
MPI_File_read_all	MPI_Pack_external	MPI_Win_lock
MPI_File_read_all_begin	MPI_Pack_external_size	MPI_Win_post
MPI_File_read_all_end	MPI_Put	MPI_Win_set_attr
MPI_File_read_at	MPI_Query_thread	MPI_Win_set_name
MPI_File_read_at_all	MPI_Register_datarep	MPI_Win_start
MPI_File_read_at_all_begin	MPI_Request_get_status	MPI_Win_test
MPI_File_read_at_all_end	MPI_Status_set_cancelled	MPI_Win_unlock
MPI_File_read_ordered	MPI_Status_set_elements	MPI_Win_wait

4.3.2 User function trace

The user function trace collects the original beginning the function that the user defined in the source program and information at the end time.

4.3.2.1 Compilation

A compilation example is given below.



Example

Compilation of the C language program "sample.c"

```
$ vtfccpx sample.c
```

4.3.2.2 Execution

An execution example is given below.



Example

Execution by 2 processes of the application program "a.out"

```
$ mpiexec -n 2 ./a.out
```

4.3.3 VampirTrace API trace

Trace data can be generated using the VampirTrace API trace.

Users can manually insert VT_USER_START (for C language/Fortran), VT_USER_END (for C language/Fortran), and VT_TRACER (for the C++ language) in the source program, and then the section is specified, and the trace data is generated.

4.3.3.1 Usage

The method for specifying the VampirTrace API trace in each language is given below.



Example

C language and Fortran

For the C language

```
#include "vt_user.h"
VT_USER_START("name");
...
VT_USER_END("name");
```

For Fortran

```
#include "vt_user.inc"
VT_USER_START('name')
...
VT_USER_END('name')
```

VT_USER_START and VT_USER_END should pair for the C language and Fortran during execution.

Therefore, it is necessary to specify VT_USER_END for all exits if there is an exit in the block of each VT_USER_START and VT_USER_END.

It is necessary to specify if "-Cpp" of the Fortran compiler option is translated, if the VampirTrace API trace is used with FORTRAN77 and Fortran90, or to change the extension to ".F" and ".F90", and to call the preprocessor.

Refer to the "Fortran User's Guide" for information on the Fortran compiler.

C++ language

For C++ language

```
#include "vt_user.h"
{
VT_TRACER("name");
```

```
...  
}
```

Specify VT_TRACER only for the entrance of scope when you specify the section by the source program of the C++ language.

You cannot use the VT_TRACE function multiple times within a single scope. In the exit of scope, the trace data is automatically output.

4.3.3.2 Compilation

A compilation example is given below.

Example

- Example 1: When only VampirTrace API trace is used

```
$ vtfccpx -vt:inst manual -DVTRACE sample.c
```

If only the VampirTrace API trace is performed, it is necessary to specify "-vt:inst manual -DVTRACE" for compilation.

- Example 2: When VampirTrace API is traced, and a user function trace is used

```
$ vtfccpx -DVTRACE sample.c
```

If both the VampirTrace API trace and the user function trace are performed, it is necessary to specify the "-DVTRACE" option for compilation.

- Example 3: When neither the VampirTrace API trace nor the user function trace is used

```
$ vtfccpx -vt:inst manual sample.c
```

The VampirTrace API trace does not operate if the "-DVTRACE" option is not specified.

Refer to "4.2.1.2 Options" for information on options.

4.3.3.3 Execution

Example

Execution by 2 processes of the application program "a.out"

```
$ mpiexec -n 2 ./a.out
```

4.3.4 I/O trace

The I/O trace collects the start/end time of the I/O function and number of bytes for data read/write.

4.3.4.1 Compilation

The I/O trace compiles a program using the vtfccpx/vtFCCpx/vtfrtpx (vtfcc/vtFCC/vtfrt for own compiler) command, and creates an application.

A compilation example is given below.

Example

Compilation of the sequential program of C language with vtfcc


```
$ vtfccpx -vt:cc fccpx -vt:seq file_io.c
```

4.3.4.2 Execution

When executing, it is necessary to specify "yes" for environment variable VT_IOTRACE to make the I/O trace feature effective. The default is "no".

An execution example is given below.



Example

Execution of the sequential application program "a.out"

```
$ export VT_IOTRACE=yes
$ ./a.out
```

4.3.4.3 I/O functions collected by the Tracer

The I/O function that tracers collect is shown [Table 4.9 I/O functions collected by the Tracer](#) below.

Table 4.9 I/O functions collected by the Tracer

close	creat	creat64	dup	dup2
fclose	fdopen	fgetc	fgets	flockfile
fopen	fopen64	fputc	fputs	fread
fscanf	fseek	fseeko	fseeko64	fsetpos
fsetpos64	ftrylockfile	funlockfile	fwrite	getc
gets	lockf	lseek	lseek64	open
open64	pread	pread64	putc	puts
pwrite	pwrite64	read	readv	rewind
unlink	write	writev		

4.3.5 Memory trace

The memory trace collects the start time, the end time, the number of bytes for allocated memory.

The memory trace feature cannot be used with the multithreaded program (OpenMP and the automatic parallelization are included).

4.3.5.1 Compilation

The Memory trace compiles a program using the vtfccpx/vtFCCpx/vtfrtpx (vtfcc/vtFCC/vtfrt for own compiler) command, and creates an application.

A compilation example is given below.



Example

Compilation of the sequential program of C language with vtfcc

```
$ vtfccpx -vt:cc fccpx -vt:seq memory.c
```

4.3.5.2 Execution

When executing, it is necessary to specify "yes" for environment variable VT_MEMTRACE to make the memory trace feature effective. The default is "no".

An execution example is given below.



Example

Execution of the sequential application program "a.out"

```
$ export VT_MEMTRACE=yes  
$ ./a.out
```

4.3.5.3 Memory functions collected by the Tracer

The memory function that tracers collect is shown [Table 4.10 Memory functions collected by the Tracer](#) below.

Table 4.10 Memory functions collected by the Tracer

calloc	free	malloc	memalign
posix_memalign	realloc	valloc	

Chapter 5 Tofu PA

5.1 Overview of Tofu PA

This chapter describes the features and usage of the Tofu PA information.

5.1.1 Tuning and Tofu PA information acquisition feature

Tuning work related to the communication performance of an application comprises two stages of optimization: optimization of the overall communication performance and optimization of the local communication performance. In the former stage, the optimal process mapping is decided based on the communication pattern and the topology shape. It is possible to generate it by using the rank arrangement optimization tool, though the optimal process mapping can also be generated manually. Refer to the "Rank Map Automatic Tuning Tools User's Guide" for information how to optimize process mapping using this tool. In the latter stage, performance improvement is done by finding communication performance problems in a specific communication section or a specific node, and determining corrective strategies such as specifying runtime options or fine-tuning process mapping. If performance issues are not discovered, it is confirmed that performance issues have been resolved because of the optimal process mapping generated using the former stage. The cause of a communication performance problem in a specific communication section or a specific node can be identified by using the Tofu PA acquisition feature, and it is useful for determining the method for improving the performance and to verify the resolution of performance issues.

5.1.2 Overview of the feature

The Tofu PA acquisition feature acquires the Tofu PA information (statistical information regarding communication in the Tofu interconnect) using the Performance Analysis (PA) feature mounted on the Tofu interconnect.

The following information is output:

Tofu PA information

The elapsed time in the 0 state during data transfer (send port and receive port) and the amount of the remaining destination buffer is output.

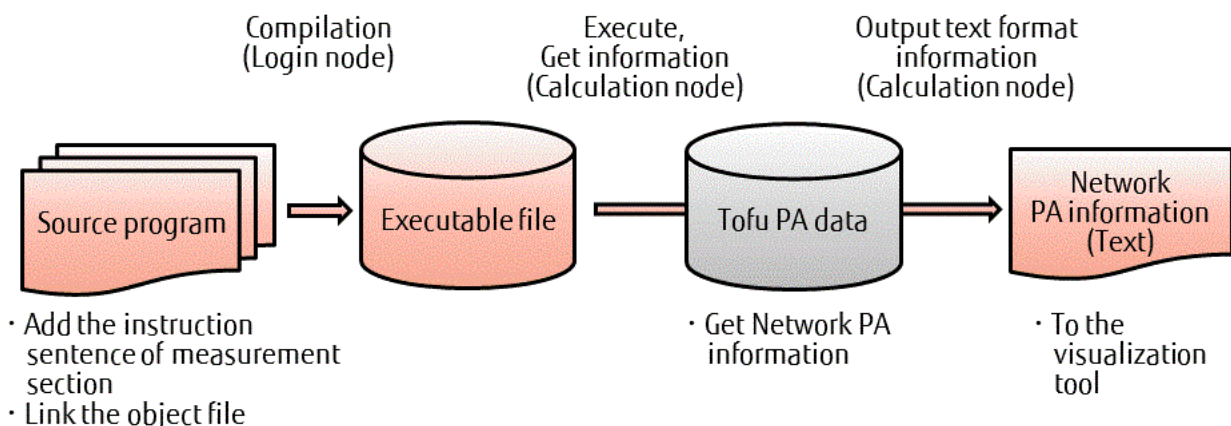
5.2 Using the Tofu PA information acquisition feature

This section describes the acquisition of the Tofu PA information and the specification of the measurement section in an application.

5.2.1 Overview of the Tofu PA information acquisition feature

"[Figure 5.1 Tofu PA information acquisition feature](#)" shows the relation between the collection and output of Tofu PA information.

Figure 5.1 Tofu PA information acquisition feature



Acquisition of the Tofu PA information

To collect the Tofu PA information, it is necessary to execute the application linked with the Tofu PA object file.

The following information is acquired on executing an application. It is also possible to select the information you want to acquire. However, you can only acquire the PA information for 10 ports of a network router, but the PA information for a network interface cannot be acquired.

- Elapsed time in the 0 state in amount of the destination buffer remainder: The total of the accumulation time in the state that becoming empty is lost in the receive buffer of the destination node is shown. The aggregate is calculated for each virtual channel.
- Number of sent and received TLP, and number of sent and received TLP bytes: The total number of packets transmitted or received, and total number of bytes transferred.

Depending on the timing of acquisition, the Tofu PA information is of two types as specified below. You can switch across these two types by using specific settings during operation.

- When the Tofu PA information is acquired at the start point and the end point of a communication section (refer to "[Figure 5.2 Operation example of Tofu PA information acquisition](#)" below)

The interface to specify the start point and the end point of a code section that acts as the measuring object is provided. The Tofu PA information is acquired at the time when these interfaces are called. The code section specified by this interface is called the measurement section. The measurement timing may differ across nodes.

- When the Tofu PA information is acquired at the start point and the end point of a communication section and the collection timing is set between nodes (refer to "[Figure 5.3 Operation example when the timing for Tofu PA information acquisition is set between nodes](#)" below)

When the Tofu PA information is acquired at the start point and the end point in a section that acts as the measuring object, the measurement timing of each node is set in accordance to one of the nodes. The node that acts as the standard for the measurement timing is specified. As the measurement timing is set by not synchronization but interrupt, change in the behavior of the program on execution can be suppressed to the minimum. (This feature is a limitation in the first edition and cannot be used.)

(This feature is a limitation in the first edition and cannot be used.)

Figure 5.2 Operation example of Tofu PA information acquisition

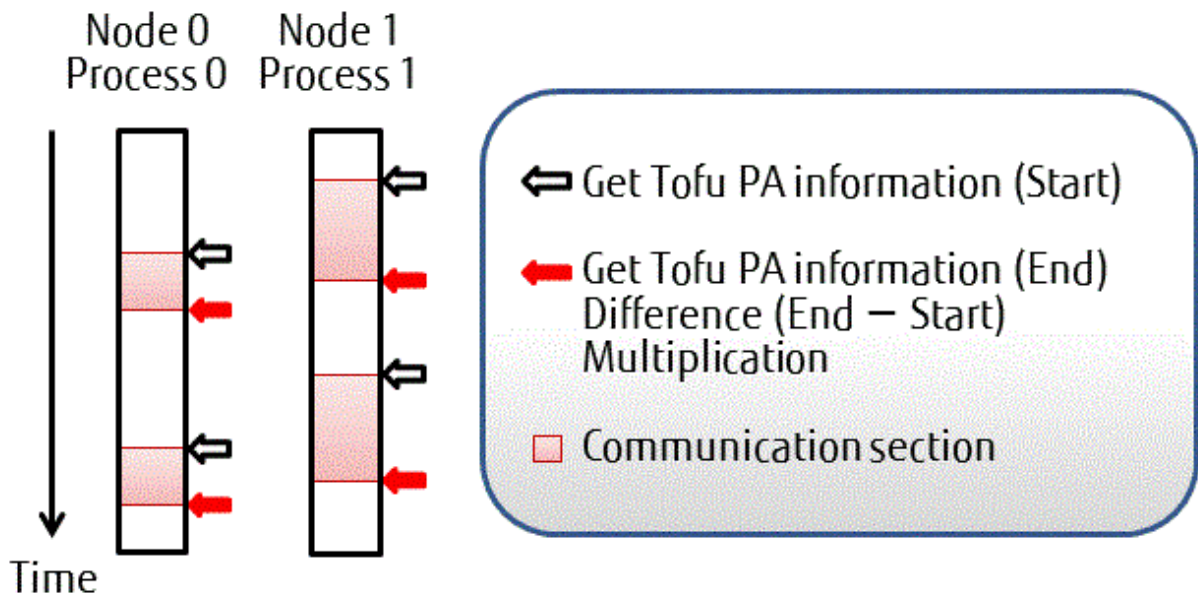
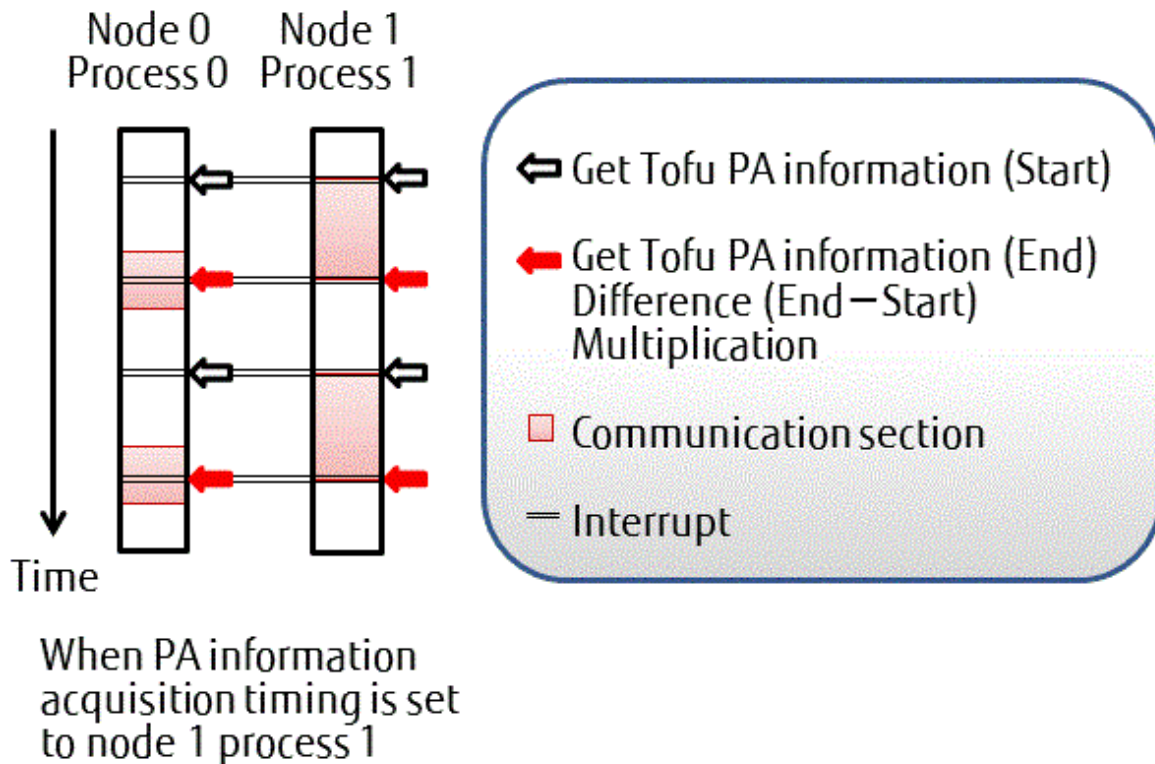


Figure 5.3 Operation example when the timing for Tofu PA information acquisition is set between nodes



The total of the Tofu PA information acquired for each measuring object section can be calculated by using the following two methods, and can be switched by using specific settings during operation.

- Aggregate of all measurement sections

The Tofu PA information is acquired for the start point and the end point of a measurement section, the difference value is requested, and each measurement section is aggregated.

- Separately record the information for each measurement section

When a measurement section is executed, the difference value of the Tofu PA information is requested. Each section for the measurement is not aggregated, and the difference value of each times of execution in each section for the measurement and the section for the measurement is recorded.

When two or more processes start in a node, the Tofu PA information is acquired only for one process automatically selected.

Output of the Tofu PA information

When an application terminates, the Tofu PA information is output to a file in the text format on each node. In addition to the Tofu PA information, the node address is output to each file. Refer to "5.2.6 File formats" for details.

Visualization of the Tofu PA information

It is possible to visualize it by loading the output file to a special tool. Additionally, it is possible to read the spreadsheet software, and generate graphs. Refer to "5.2.7 Visibility" for details.

5.2.2 Specifying the measurement section

Specifying a measurement section involves specifying a measurement range for the Tofu PA information. To specify a measurement range in the source code, insert subroutines at the start position and the end position for measuring the Tofu PA information. Functions of the C/C++ languages or Fortran subroutines can be used to specify a measurement section.

If using C/C++ functions, include the header file or function prototype declaration.

Overview of the functions used to specify a measurement section is given below.

Language	Header file	Function name (Procedure name)	Description	Arguments	Return value
C/C++	fj_tool/fjtofupa.h	fj_tofupa_start	Starts a measurement section	Described in the next table	-
		fj_tofupa_stop	Ends a measurement section		
Fortran	-	fj_tofupa_start	Starts a measurement section		
		fj_tofupa_stop	Ends a measurement section		

The arguments for each function (procedure) are described below.

Argument	Type (C/C++)	Type (FORTRAN)	Description
name	char *	CHARACTER	Group name (not used)
region	int	INTEGER	Number of measurement sections (Integer value from 1 - 4095)
level	int	INTEGER	Priority level (Integer value of 0 or more)

Take the following points into consideration when calling these functions (procedure).

- Call fj_tofupa_start or fj_tofupa_stop between MPI_Init (or MPI_Init_thread) and MPI_Finalize.
- In case of a multithreaded program, call fj_tofupa_start and fj_tofupa_stop by the thread where an MPI function is called.
- The method for calling MPI_Init, MPI_Init_thread, or MPI_Finalize must conform to the requirements of the MPI library. Refer to the "MPI User's Guide" for details.
- The section with the same number of the section for the measurement for the measurement overlaps or it cannot have the inclusion relation though can a part of the section with a different number of the section for the measurement for the measurement overlap or the other side be included in one.
- Tofu PA information in the section from MPI_Init (or, MPI_Init_thread) to MPI_Finalize is acquired as data of the measurement object section number 0 and the priority level 0 besides the specified section for the measurement. Please specify the value of 0 or more for environment variable FJ_TOFUPA_LEVEL to do the above-mentioned measurement effectively.



Example

- Example 1: The measurement section contains a point-to-point communication section of an application written in C.

```
#include <mpi.h>
#include "fj_tool/fjtofupa.h"

int main(int argc, char ** argv)
{
    int rank, size, l, r, j;
    int b[2][262144];
    MPI_Request Rq[2];
    MPI_Status St[2];
    MPI_Init(&argc, &argv);

    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);

    l = (rank - 1 + size) % size;
    r = (rank + 1) % size;

    MPI_Barrier(MPI_COMM_WORLD);

    fj_tofupa_start("", 1, 0);
    MPI_Isend(&b[0], 262144, MPI_INT, l, 0, MPI_COMM_WORLD, &Rq[0]);
    MPI_Irecv(&b[1], 262144, MPI_INT, r, 0, MPI_COMM_WORLD, &Rq[1]);
    MPI_Waitall(2, Rq, St);
}
```

```

fj_tofupa_stop("", 1, 0);

MPI_Finalize();
return 0;
}

```

- Example 2: The measurement section contains a point-to-point communication section of an application written in C++.

```

#include <mpi.h>
#include "fj_tool/fjtofupa.h"
using namespace std;

int main(int argc, char ** argv)
{
    int rank, size, l, r, j;
    int b[2][262144];
    MPI::Request Rq[2];
    MPI::Status St[2];
    MPI::Init(argc, argv);

    rank = MPI::COMM_WORLD.Get_rank();
    size = MPI::COMM_WORLD.Get_size();

    l = (rank - 1 + size) % size;
    r = (rank + 1) % size;

    MPI::COMM_WORLD.Barrier();

    fj_tofupa_start("", 1, 0);
    Rq[0] = MPI::COMM_WORLD.Isend(&b[0], 262144, MPI_INT, l, 0);
    Rq[1] = MPI::COMM_WORLD.Irecv(&b[1], 262144, MPI_INT, r, 0);
    Rq[0].Waitall(2, Rq, St);
    fj_tofupa_stop("", 1, 0);

    MPI::Finalize();
    return 0;
}

```

- Example 3: The measurement section contains a point-to-point communication section of an application written in Fortran 90.

```

program prg
    implicit none
    include 'mpif.h'
    integer myrank, mysize, l, r, j, ierr
    integer, dimension(262144,2) :: b
    integer Rq(2)
    integer St(MPI_STATUS_SIZE,2)
    call MPI_Init(ierr)

    call MPI_Comm_rank(MPI_COMM_WORLD, myrank, ierr)
    call MPI_Comm_size(MPI_COMM_WORLD, mysize, ierr)

    l = mod((myrank - 1 + mysize), mysize)
    r = mod((myrank + 1), mysize)

    call MPI_Barrier(MPI_COMM_WORLD, ierr)

    call fj_tofupa_start('', 1, 0)
    call MPI_Isend(b(1,1), 262144, MPI_INTEGER, l, 0, MPI_COMM_WORLD, Rq(1), ierr)
    call MPI_Irecv(b(1,2), 262144, MPI_INTEGER, r, 0, MPI_COMM_WORLD, Rq(2), ierr)
    call MPI_Waitall(2, Rq, St, ierr)
    call fj_tofupa_stop('', 1, 0)

```

```

call MPI_Finalize(ierr)
end

```

5.2.3 Compilation

It is necessary to create an application that links with a special object file to use the Tofu PA acquisition feature.

Specify the object file (/opt/FJSMxlang/lib64/libtofupa.o) in the options of the mpifccpx, mpiFCCpx, and mpifrtpx commands.

5.2.4 Execution

Environment variable

It is necessary to set the environment variable correctly.

Environment variable	Value
LD_LIBRARY_PATH	/opt/FJSMxlang/lib64

To use the batch queuing system and the MPI processing system, additional settings besides those mentioned above may be necessary. Refer to the "Job Operation Software First Step Guide" for information on the batch queuing system. Refer to the "MPI User's Guide" for information on the MPI system.

The operation of the Tofu PA acquisition feature can be controlled by specifying the following environment variables.

Environment variable	Value
FJ_TOFUPA_LEVEL	The priority level counter is set. The PA information is acquired if the "level" value specified for the measurement section is less than this value. The default is -1, and all PA information is not acquired.
FJ_TOFUPA_DIR	The file output destination directory is specified. The default is the current directory.
FJ_TOFUPA_PA	The Tofu PA information to be output is specified. The name of the Tofu PA information or the " Table 5.2 Tofu PA information groups " is specified for this environment variable (refer to " Table 5.1 Tofu PA information name " below). A Tofu PA information group contains two or more Tofu PA information. The default is "STD", and all Tofu PA information is acquired.
FJ_TOFUPA_MAXREGION	The maximum number of measurement sections is specified. However, because the number of measurement sections starts from 0, the value obtained by subtracting 1 from the value specified for this environment variable becomes the maximum number of measurement sections. The default is 4096, while a value from 1 through 4095 can be specified for the number of measurement sections. A value of 0 for the number of measurement sections is reserved, and disregarded even if specified.
FJ_TOFUPA_MAXITERATION	The maximum record frequency when Tofu PA information on each section for the measurement is not aggregated, and each times of execution in the section for the measurement are recorded is specified. The default is 0 that aggregates the Tofu PA information on each measurement section. The Tofu PA information is recorded at each execution of the measurement section if a value of 1 or more is specified. However, if the record frequency specified by this environment variable is exceeded, the Tofu PA information is discarded.

The different types of Tofu PA information that can be acquired with their description are described below. Each Tofu PA information is output for each port of a network router of each node. Refer to "[Chapter 7 Glossary](#)" for information on the glossary related to Tofu interconnect.

Table 5.1 Tofu PA information name

Tofu PA information	Description
TxVc0ZeroCreditCount	In the virtual channel 0 (VC0) of each port of the network router of each node, it shows between total time in 0 states in the amount of the destination buffer remainder. This time is expressed in "cycles". One cycle corresponds to 2.56 nanoseconds.
TxVc1ZeroCreditCount	In the virtual channel 1 (VC1) of each port of the network router of each node, it shows between total time in 0 states in the amount of the destination buffer remainder. This time is expressed in "cycles". One cycle corresponds to 2.56 nanoseconds.
TxVc2ZeroCreditCount	In the virtual channel 2 (VC2) of each port of the network router of each node, it shows between total time in 0 states in the amount of the destination buffer remainder. This time is expressed in "cycles". One cycle corresponds at 2.56 nanoseconds.
TxVc3ZeroCreditCount	In the virtual channel 3 (VC3) of each port of the network router of each node, it shows between total time in 0 states in the amount of the destination buffer remainder. This time is expressed in "cycle". One cycle corresponds at 2.56 nanoseconds.
NumSendTLP	The total number of packets (TLP) transmitted from each port of the network router of each node is displayed.
ByteSendTLP	The total number of bytes for packets (TLP) transmitted from each port of the network router of each node is displayed. The number of bytes includes the packet header and the MPI header.
NumReceiveTLP	The total number of packets (TLP) received at each port of the network router of each node is displayed.
ByteReceiveTLP	The total number of bytes for packets (TLP) received at each port of the network router of each node is displayed. The number of bytes includes the packet header and the MPI header.

TxVc[0-3]ZeroCreditCount displays the time duration when the destination buffer had been buried by the packet. Neither time that the packet that had to be transmitted did the transmission waiting nor the packet that should be transmitted exist and both at the time that nothing but passes are included at this time. When latter time is generally shorter than the former, it is also possible to consider this value to be an approximate value of the former because it is surmisable. When such how to catch is done, the situation of the occurrence of the packet forwarding waiting in the network router of each node can be presumed according to these values.

The Tofu PA information groups and the Tofu PA information included in each group is shown below.

Table 5.2 Tofu PA information groups

Tofu PA group	Description
STD	Specifies the following Tofu PA information names at the same time: TxVc0ZeroCreditCount, TxVc1ZeroCreditCount, TxVc2ZeroCreditCount, TxVc3ZeroCreditCount, NumSendTLP, ByteSendTLP, NumReceiveTLP, ByteReceiveTLP
CreditCount	Specifies the following Tofu PA information names at the same time: TxVc0ZeroCreditCount, TxVc1ZeroCreditCount, TxVc2ZeroCreditCount, TxVc3ZeroCreditCount
ZeroCreditCount	Same as CreditCount.
TLP	Specifies the following Tofu PA information names at the same time: NumSendTLP, ByteSendTLP, NumReceiveTLP, ByteReceiveTLP
SendTLP	Specifies the following Tofu PA information names at the same time: NumSendTLP, ByteSendTLP
ReceiveTLP	Specifies the following Tofu PA information names at the same time: NumReceiveTLP, ByteReceiveTLP
ByteTLP	Specifies the following Tofu PA information names at the same time:

Tofu PA group	Description
	ByteSendTLP, ByteReceiveTLP
NumTLP	Specifies the following Tofu PA information names at the same time: NumSendTLP, NumReceiveTLP

Options specified when a job is submitted

When the Tofu PA information is acquired, a job may be executed two or more times for tuning. To maintain constant physical shape of the node resources allocated at the time of job execution (shape in six-dimensional coordinate system), the following options must be specified, and for this, the node shape must not become three-dimensional and breakdown nodes must not be included. Two or more candidates are in physical shape in the specification of 1-2 dimensions, and physical shape might change at job execution.

Option	Description
--mpi "assign-online-node"	The breakdown node is not included in the allocated space. It is guaranteed that the alternative path is not included in routing.
-L "node=XxYxZ:strict"	The physical shape of the allocated resource is fixed. Any arbitrary rotation is prohibited.

To control the operations of the Tofu PA acquisition feature, the above options are executed for some environment variables. This is exemplified in the following example.

Example

An MPI application, a.out, is executed by 16 parallels (16 nodes are allocated, and one process per node is started) and the Tofu PA information is obtained

```
#!/bin/sh
#PJM -L "rscunit=unit1"
#PJM --mpi "assign-online-node"
#PJM -L "node=2x2x4:strict"

export LD_LIBRARY_PATH=/opt/FJSMxlang/lib64:$LD_LIBRARY_PATH
export FJ_TOFUPA_LEVEL=100
mpiexec -n 16 ./a.out
```

5.2.5 Output file name

The name of each file is decided based on the address of the node. The format of the file name is given below.

File name format

```
tofupa.<job ID>.<node address>.txt
```

5.2.6 File formats

Each file consists of the following lines.

Line	Description
Six-dimensional physical node address	Coordinates (X,Y,Z,A,B,C) of the six-dimensional physical node address are output by switching off the space district following the "# NODE_ADDR " character string. Only one line per file is output. Example: "# NODE_ADDR 3 4 5 0 2 1"
Three-dimensional logical address	Coordinates (X), (X,Y) or (X,Y,Z) of the node address in 1-3 dimensional logical space is output by switching off the space district following the "# LOGICAL_ADDR" character string.

Line	Description
	Only one line per file is output. Example: "# LOGICAL_ADDR 0 1 2"
Rank	The rank in MPI_COMM_WORLD is output following the "# RANK" character string. Only one line per file is output. Example: "# RANK 0"
Port	The number of ports and the port names are output following the "# PORT" character string. Only one line per file is output. Example: "# PORT 10 X+ X- Y+ Y- Z+ Z- A B+ B- C"
PA information kind	The number of output PA information (port) and the names of PA information is output following the "# PA" character string. Only one line per file is output. Example: "# PA 3 TxVc0ZeroCreditCount TxVc1ZeroCreditCount ByteSendTLP"
PA information output	The line of each measurement section is delimited and it is output. PA information in the number of the section for the measurement and the section for a measurement concerned is output to each line by switching off the space district. A colon is added after the number of the section for the measurement when Tofu PA information on each times of execution is recorded, and the record frequency is output. PA information is as many as 8 pieces or less port, and the number becomes 80 or less in ten ports. The PA information is output in the order of the ports (X+, X-, Y+, Y-, Z+, Z-, A, B+, B-, C). As for this line, the same number of lines as products of the number of sections for the measurement and the number are output about one file.
The final line	The "# EOF" character string is output. Only one line per file is output.

The port names are described below.

The length of X,Y,Z axis of the Tofu interconnect is assumed to be X_SIZE,Y_SIZE,Z_SIZE, and the operation symbol from which the remainder is requested is the rate symbol (%).

Port name	Description
X+	Port connected to the node adjacent to the + direction of the X axis (Node of ((X+1)%X_SIZE,Y,Z,A,B,C) when the six-dimensional node address is (X,Y,Z,A,B,C))
X-	Port connected to the node adjacent to the - direction of the X axis (Node of ((X-1+X_SIZE)%X_SIZE,Y,Z,A,B,C) when the six-dimensional address of the node is (X,Y,Z,A,B,C))
Y+	Port connected to the node adjacent to the + direction of the Y axis (Node of (X,(Y+1)%Y_SIZE,Z,A,B,C) when the six-dimensional address of the node is (X,Y,Z,A,B,C))
Y-	Port connected to the node adjacent to the - direction of the Y axis (Node of (X,(Y-1+Y_SIZE)%Y_SIZE,Z,A,B,C) when the six-dimensional address of the node is (X,Y,Z,A,B,C))
Z+	Port connected to the node adjacent to the + direction of Z axis (Node of (X,Y,(Z+1)%Z_SIZE,A,B,C) when the six-dimensional address of the node is (X,Y,Z,A,B,C))
Z-	Port connected to the node adjacent to the - direction of the Z axis (Node of (X,Y,(Z-1+Z_SIZE)%Z_SIZE,A,B,C) when the six-dimensional address of the node is (X,Y,Z,A,B,C))
A	Port connected to the node adjacent to the A axis (Node of (X,Y,Z,(1-A),B,C) when the six-dimensional address of the node is (X,Y,Z,A,B,C))
B+	Port connected to the node adjacent to the + direction of the B axis

Port name	Description
	(Node of (X,Y,Z,A,(B+1)%3,C) when the six-dimensional address of the node is (X,Y,Z,A,B,C))
B-	Port connected to the node adjacent to the - direction of the B axis (Node of (X,Y,Z,A,(B+2)%3,C) when the six-dimensional address of the node is (X,Y,Z,A,B,C))
C	Port connected to the node adjacent to the C axis (Node of (X,Y,Z,A,B,(1-C)) when the six-dimensional address of the node is (X,Y,Z,A,B,C))



Example

Contents of an output file

```
# NODE_ADDR 1 1 0 0 0 0
# LOGICAL_ADDR 0 0 0
# RANK 0
# PORT 10 X+ X- Y+ Y- Z+ Z- A B+ B- C
# PA 8 tx_vc0_zero_credit_count tx_vc1_zero_credit_count tx_vc2_zero_credit_count
tx_vc3_zero_credit_count num_send_tlp byte_send_tlp num_receive_tlp byte_receive_tlp
0 0 0 0 0 167 87176 157 65432 0 0 0 0 219 68584 227 86216 0 0 0 0 0 0 0 0 0 0 208 97104 208
97072 ...
```

5.2.7 Visibility

It is possible to convert the Tofu PA information output on each node into the spreadsheet format that is easy to read by using the following script. The file output on each node is read by a batch by executing the following script in the directory where the output file exists, and the node and each port are output to one file as a record of one row. In the spreadsheet, the statistics can be visually understood by graphs between total time 0 states in the transmission volume of data, the amount of data received, and the amount of each port of the destination buffer remainder. Moreover, the difference in the statistics due to the difference of the place can be confirmed by changing the sort order for the node address.

```
#!/bin/sh

cat tofupa.*.txt |
awk '\
BEGIN{
  init=0;
}
/# EOF/{
  next;
}
/# NODE_ADDR/{
  x=$3;
  y=$4;
  z=$5;
  a=$6;
  b=$7;
  c=$8;
  next;
}
/# PORT/{
  n_port = $3;
  for (i = 0; i < n_port; i++)
    namePort[i] = $(i+4);
  next;
}
/# PA/{
  n_pa = $3;
  for (i = 0; i < n_pa; i++)
```

```

    namePa[i] = $(i+4);
next;
}
{
if (init == 0) {
    init = 1;
    printf "6d_addr\taxis\trid\titer";
    for (i=0; i<n_pa; i++) {
        printf "\t%s", namePa[i];
    }
    printf "\n";
}

tmp = $1;
split(tmp, arr, ":");
rid = arr[1];
itr = arr[2];
for (i=0; i<n_port; i++){
    printf "%d,%d,%d,%d,%d,%d\t%s\t%d\t%d", x,y,z,a,b,c, namePort[i], rid, itr;
    for (j=0; j<n_pa; j++){
        printf "\t%s", $(i*n_pa + j + 2)
    }
    printf "\n";
}
}'

```

Chapter 6 Open Source Profiler

The profiler that is released as open source software is installed in this system.

This chapter describes the features and usage of mpiP that is the one of the open source profiler.

6.1 Overview of mpiP

mpiP is a lightweight profiling library for MPI applications. Because it only collects statistical information about MPI functions, mpiP generates considerably less overhead and much less data than tracing tools

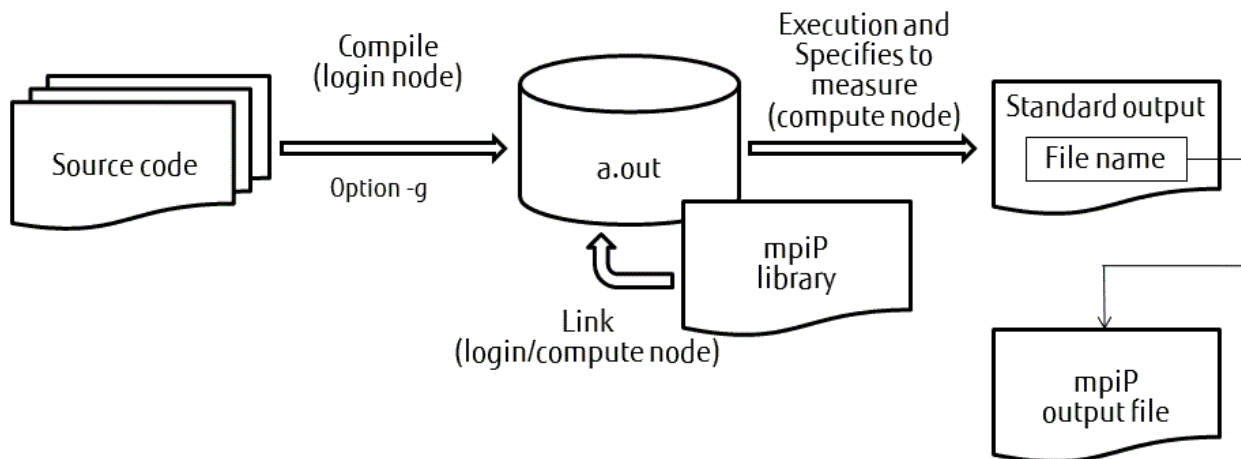
For more information about mpiP, refer to the following website.

<http://mpiP.sourceforge.net/>

6.2 Using mpiP

This section describes compiling steps and the execution method required to use the mpiP.

Figure 6.1 Collection and output of mpiP information



6.2.1 Compilation

You don't have to recompile your object file if you just use mpiP. But you might have to recompile to include the "-g" option if you need line number in a source code.

6.2.2 Linking

By specifying the following options in the linking, an application is linked with mpiP library (libmpiP.a). And mpiP information is output when the application runs.

```
-L${mpiP_root}/lib64 -lmpiP -lm -lbfd -liberty -Ntl_notrt
```

- \${mpiP_root} is the following directory that the FJSVmxlang package is installed.
 - /opt/FJSVmxlang
- "-lm", "-lbfd" and "-liberty" are libraries used by mpiP.
- "-Ntl_notrt" is an option to disable the feature of the tool library conflicting with mpiP.



Example

Example of linking C program

```
$ mpifccpx sample1.c -o sample1.out -L/opt/FJSMxlang/lib64 -lmpiP -lm -lbfd -liberty -Ntl_notrt
```

6.2.3 Execution

By the application linked with mpiP library is executed, mpiP information is collected.

Environment variables

When the application runs, mpiP output format is changed by specifying the environment variable MPIP such as the follows.

```
MPIP="-p -t 10.0"
```

The following table lists the options for the environment variable MPIP.

Table 6.1 Options for the environment variable MPIP

Option	Description	Note
-ba	MPI_Barrier is inserted after collective communication, and the beginning timing of the processing after collective communication is arranged between the ranks. As a result, the influence of the imbalance by collective communication of preceding is excluded, and the measurement of the clear time of following process is supported. For more information, refer to " Table 6.14 MPI function of target for options (-ba,-bb) of environment variable MPIP ".	
-bb	MPI_Barrier is inserted before collective communication, and the beginning timing of collective communication is arranged between the ranks. As a result, the influence of the imbalance by the processing of preceding (operation etc.) is excluded, and the measurement of the clear collective communication time is supported. For more information, refer to " Table 6.14 MPI function of target for options (-ba,-bb) of environment variable MPIP ".	
-c	Generates concise version of report, omitting callsite process-specific detail.	
-d	Suppresses printing of callsite detail sections.	
-e	Prints report data using floating-point format.	
-f <i>dir</i>	Records output file in directory <i>dir</i> .	When "-f <i>dir</i> " option or <i>dir</i> is not specified, the output file is stored in the current directory.
-g	Enables mpiP debug mode.	
-k <i>n</i>	Sets callsite stack traceback depth to <i>n</i> .	When "-k <i>n</i> " option or <i>n</i> is not specified, the depth is 1.
-l	Uses less memory to generate the report by using MPI collectives to generate callsite information on a callsite-by-callsite basis.	
-n	Do not truncate full pathname of filename in callsites.	
-o	Disable profiling at initialization. Application must enable profiling with MPI_Pcontrol(). For more information, refer to " 6.3.2 Control of Profiling Range for mpiP ".	
-p	Point-to-point Communication histogram reporting on message size and communicator used.	
-r	Generates the report by aggregating data at a single task.	When "-r" option is not specified, the report is generated.

Option	Description	Note
-s <i>n</i>	Sets hash table size to <i>n</i> .	When "-s <i>n</i> " option or <i>n</i> is not specified, the size is 256.
-t <i>x</i>	Sets print threshold for report, where <i>x</i> is the MPI rate of time for each callsite.	When "-t <i>x</i> " option or <i>x</i> is not specified, the rate is 0.0.
-v	Generates both concise and verbose report output.	
-x <i>exe</i>	Information on the line number of the source code output to the mpiP information is searched from the application specified for <i>exe</i> . The application name is specified for <i>exe</i> by the full path.	When "-x <i>exe</i> " option or <i>exe</i> is not specified, it is searched from the executed application.
-y	Collective Communication histogram reporting on message size and communicator used.	
-z	Suppresses printing of the report at MPI_Finalize.	

6.2.4 mpiP Output

When the application exits, the following message is printed in standard output.

```

mpiP:
mpiP: mpiP: mpiP V3.4.1 (Build May  7 2014/14:46:10)
mpiP: Direct questions and errors to mpip-help@lists.sourceforge.net
mpiP:
mpiP:
mpiP: Storing mpiP output in [./sample.out.2.8670.1.mpiP].
mpiP:

```

The above example shows that mpiP information is output to the new file sample.out.2.8670.1.mpiP.

The example of mpiP information output is the follows.

```

@ mpiP
@ Command : ./sample.out
@ Version      : 3.4.1
@ MPIP Build date   : May  7 2014, 14:46:10
@ Start time      : 2014 05 27 10:41:00
@ Stop time       : 2014 05 27 10:41:00
@ Timer Used      : PMPI_Wtime
@ MPIP env var    : -p -v -y
@ Collector Rank  : 0
@ Collector PID   : 9339
@ Final Output Dir : .
@ Report generation : Single collector task

-----
@--- Task Time Statistics (seconds) -----
-----
Max           AppTime      MPITime  MPI%   App Task  MPI Task
Mean          0.000564    0.000479  70.69  3         3
Min           0.000510    0.000360
Stddev        0.000385    0.000050
Aggregate     0.000084    0.000207
Aggregate     0.002040    0.001442  70.69

-----
@--- Aggregate Time (top twenty, descending, milliseconds) -----
-----
Call          Site      Time     App%   MPI%   COV
Reduce        1         1.44    70.69  100.00  0.57

-----
@--- Aggregate Sent Message Size (top twenty, descending, bytes) -----
-----

```



```

Call          Site      Count      Total      Avrg  Sent%
Reduce        1          4          16          4 100.00
-----
@--- Aggregate Collective Time (top twenty, descending) -----
-----
Call          MPI Time %      Comm Size      Data Size
Reduce        100            0 -           7            0 -           7
-----
@--- Aggregate Point-To-Point Sent (top twenty, descending) -----
-----
No point to point operations to report
-----
@--- Callsites: 1 -----
-----
ID Lev File/Address      Line Parent_Funct      MPI_Call
  1  0 sample.f           23 MAIN__            Reduce
-----
@--- Callsite Time statistics (all callsites, milliseconds): 1 -----
-----
Name          Site    Tasks      Max      Mean      Min MaxRnk MinRnk
Reduce        1       4      0.479    0.3605    0.05     3     0
-----
@--- End of Report -----
-----

```

6.3 Functional Detail

6.3.1 mpiP Report Information

This section describes the items in information of the mpiP report. Refer to "6.2.4 mpiP Output" for details.

6.3.1.1 Header Information

Basic information is output as the header at the top of the mpiP report.

Figure 6.2 Output format of header information

```

@ mpiP
@ Command : @commd
@ Version           : @ver
@ MPIP Build date  : @bdate
@ Start time       : @stime
@ Stop time        : @etime
@ Timer Used       : PMPI_Wtime
@ MPIP env var     : @env
@ Collector Rank   : @rank
@ Collector PID    : @pid
@ Final Output Dir : @outdir
@ Report generation : @repgen
@ MPI Task Assignment : @tid @node

```

Table 6.2 Output items of header information

Output item	Description
@commd	Executed command name
@ver	Version of mpiP (Fixed value; system-dependent)
@bdate	Build date of mpiP (Fixed value; system-dependent)
@stime	Start time

Output item	Description
@etime	Stop time
@env	Value of the environment variable MPIP
@rank	Rank number of the collector process
@pid	PID of the collector process
@outdir	Output directory of the mpiP report
@repgen	Generation type of the mpiP report
@tid	Task ID
@node	Node name that is assigned the task

6.3.1.2 MPI Time Information

An overview of the application's time in MPI is output.

Figure 6.3 Output format of MPI time information

```

-----
@--- MPI Time (seconds) -----
-----
Task   AppTime  MPITime  MPI%
@tid   @atime   @mtime   @mper
@tid   @atime   @mtime   @mper
@tid   @atime   @mtime   @mper

```

Table 6.3 Output items of MPI time information

Output item	Description
@tid	Task ID. The task number line displayed with asterisk (*) shows the output item information that totals the value for the entire application.
@atime	The wall-clock time from the end of MPI_Init until the beginning of MPI_Finalize. (Second)
@mtime	The wall-clock time for all the MPI calls contained within @atime. (Second)
@mper	@mtime/@atime (%)

6.3.1.3 Callsite Information

Information about all the MPI callsites within the application is output.

Figure 6.4 Output format of callsite information

```

-----
@--- Callsites: @num -----
-----
ID Lev File/Address      Line Parent_Funct      MPI_Call
@i @lv @fname              @ln @func              @mcall
@i @lv @fname              @ln @func              @mcall

```

Table 6.4 Output items of callsite information

Output item	Description
@num	Total number of callsites
@i	The callsite ID for this mpiP file
@lv	The callsite stack traceback depth
@fname	The filename that includes the callsite

Output item	Description
@ln	The line number of the callsite in the source code
@func	The function name that includes the callsite
@mcall	The type of MPI function (w/o MPI_ prefix) MPI_Barrier inserted after collective communication by specifying optional "-ba" is displayed as BarrierAfterColl. MPI_Barrier inserted before collective communication by specifying optional "-bb" is displayed as BarrierBeforeColl.

6.3.1.4 Aggregate Time Information

The top twenty MPI callsites that consume the most aggregate time are output.

Figure 6.5 Output format of aggregate time information

```

-----
@--- Aggregate Time (top twenty, descending, milliseconds) -----
-----
Call          Site      Time    App%   MPI%   COV
@mcall        @i       @time   @apr   @mper  @cov
@mcall        @i       @time   @apr   @mper  @cov

```

Table 6.5 Output items of aggregate time information

Output item	Description
@mcall	The type of MPI function (w/o MPI_ prefix) MPI_Barrier inserted after collective communication by specifying optional "-ba" is displayed as BarrierAfterColl. MPI_Barrier inserted before collective communication by specifying optional "-bb" is displayed as BarrierBeforeColl.
@i	The callsite ID for this mpiP file (as listed in the callsite section)
@time	The aggregate time for the callsite (Millisecond)
@apr	The aggregate time for the callsite / The total application time (%)
@mper	The aggregate time for the callsite / The total MPI time (%)
@cov	The coefficient of variation as calculated from the individual process times. The variation in times of individual processes for the callsite is indicated. A larger value indicates more variation between the process times.

6.3.1.5 Aggregate Sent Message Size

The top twenty MPI callsites for total sent message size are output.

Figure 6.6 Output format of aggregate sent message size

```

-----
@--- Aggregate Sent Message Size (top twenty, descending, bytes) -----
-----
Call          Site      Count   Total   Avrg   MPI%
@mcall        @i       @cnt    @tl    @avg   @mper
@mcall        @i       @cnt    @tl    @avg   @mper

```

Table 6.6 Output items of aggregate sent message size

Output item	Description
@mcall	The type of MPI function (w/o MPI_ prefix)

Output item	Description
@i	The callsite ID for this mpiP file (as listed in the callsite section)
@cnt	Number of times this function was executed
@tl	The total of sent message sizes (Byte)
@avg	The average of sent message sizes (Byte)
@mper	The total of sent message sizes / The total of sent message sizes for all MPI function (%)

6.3.1.6 Callsite Time Statics

The wall-clock time of callsites for each rank is output.

If the threshold is specified by "-t" option, the lines where MPI% was less than the threshold are not printed. The default value of the threshold is 0 (all aggregate lines are printed).

Figure 6.7 Output format of callsite time statistics

```

-----
@--- Callsite Time statistics (all, milliseconds): @num -----
-----
Name           Site Rank  Count    Max    Mean    Min    App%  MPI%
@mcall         @i  @r  @cnt   @max   @avg   @min  @apr  @mper
@mcall         @i  @r  @cnt   @max   @avg   @min  @apr  @mper

```

Table 6.7 Output items of callsite time statistics

Output item	Description
@num	Total number of callsites per rank
@mcall	The type of MPI function (w/o MPI_ prefix). MPI_Barrier inserted after collective communication by specifying optional "-ba" is displayed as BarrierAfterColl. MPI_Barrier inserted before collective communication by specifying optional "-bb" is displayed as BarrierBeforeColl.
@i	The callsite ID for this mpiP file (as listed in the callsite section)
@r	Task rank in MPI_COMM_WORLD The rank number line displayed with asterisk (*) shows the output item information that totals the value of all ranks.
@cnt	Number of times this function was executed
@max	Maximum wall-clock time for one call (Millisecond)
@avg	Arithmetic mean of the wall-clock time for one call (Millisecond)
@min	Minimum wall-clock time for one call (Millisecond)
@apr	The wall-clock time for this call / The wall-clock time of the overall application (%)
@mper	The wall-clock time for this call / The wall-clock time of the overall MPI (%)

6.3.1.7 Callsite Message Sent Statistics

The message sent size of callsites for each rank is output.

Figure 6.8 Output format of callsite message sent statistics

```

-----
@--- Callsite Message Sent statistics (all, sent bytes) -----
-----
Name           Site Rank  Count      Max      Mean     Min      Sum
@mcall         @i  @r  @cnt     @max    @avg    @min    @sum
@mcall         @i  @r  @cnt     @max    @avg    @min    @sum

```

Table 6.8 Output items of callsite messages sent statistics

Output item	Description
@mcall	The type of MPI function (w/o MPI_ prefix)
@i	The callsite ID for this mpiP file (as listed in the callsite section)
@r	Task rank in MPI_COMM_WORLD The rank number line displayed with asterisk (*) shows the output item information that totals the value of all ranks.
@cnt	Number of times this function was executed
@max	Maximum sent message size for one call (Byte)
@avg	Arithmetic mean of the sent message sizes for one call (Byte)
@min	Minimum sent message size for one call (Byte)
@sum	Total of all message sizes for the this operation and callsite (Byte)

6.3.2 Control of Profiling Range for mpiP

By using the following function in the application, it is possible to control the profiling range of the source code.

```
MPI_Pcontrol(const int flag)
```

The following values are specified as the argument "flag".

Table 6.9 Argument Value of MPI_Pcontrol

Argument	Behavior
0	Disable profiling.
1	Enable profiling.
2	Reset all callsite data.
3	Generate verbose report.
4	Generate concise report.

6.3.3 MPI functions collected by mpiP

The following table lists MPI functions profiled with mpiP.

Table 6.10 MPI functions profiled with mpiP

MPI_Allgather	MPI_File_seek	MPI_Rsend_init
MPI_Allgatherv	MPI_File_set_view	MPI_Scan
MPI_Allreduce	MPI_File_write	MPI_Scatter
MPI_Alltoall	MPI_File_write_all	MPI_Scatterv
MPI_Alltoallv	MPI_File_write_at	MPI_Send
MPI_Attr_delete	MPI_Gather	MPI_Send_init
MPI_Attr_get	MPI_Gatherv	MPI_Sendrecv

MPI_Attr_put	MPI_Graph_create	MPI_Sendrecv_replace
MPI_Barrier	MPI_Graph_get	MPI_Ssend
MPI_Bcast	MPI_Graph_map	MPI_Ssend_init
MPI_Bsend	MPI_Graph_neighbors_count	MPI_Start
MPI_Bsend_init	MPI_Graphdims_get	MPI_Startall
MPI_Buffer_attach	MPI_Group_compare	MPI_Test
MPI_Buffer_detach	MPI_Group_difference	MPI_Testall
MPI_Cancel	MPI_Group_excl	MPI_Testany
MPI_Cart_coords	MPI_Group_free	MPI_Testsome
MPI_Cart_create	MPI_Group_incl	MPI_Topo_test
MPI_Cart_get	MPI_Group_intersection	MPI_Type_commit
MPI_Cart_map	MPI_Group_translate_ranks	MPI_Type_free
MPI_Cart_rank	MPI_Group_union	MPI_Type_get_contents
MPI_Cart_shift	MPI_Ibrecv	MPI_Type_get_envelope
MPI_Cart_sub	MPI_Intercomm_create	MPI_Unpack
MPI_Cartdim_get	MPI_Intercomm_merge	MPI_Wait
MPI_Comm_create	MPI_Iprobe	MPI_Waitall
MPI_Comm_dup	MPI_Irecv	MPI_Waitany
MPI_Comm_group	MPI_Irrecv	MPI_Waitsome
MPI_Comm_remote_group	MPI_Isend	MPI_Win_complete
MPI_Comm_remote_size	MPI_Issend	MPI_Win_create
MPI_Comm_split	MPI_Keyval_create	MPI_Win_fence
MPI_Comm_test_inter	MPI_Keyval_free	MPI_Win_free
MPI_Dims_create	MPI_Pack	MPI_Win_get_group
MPI_Error_class	MPI_Probe	MPI_Win_lock
MPI_File_close	MPI_Recv	MPI_Win_post
MPI_File_open	MPI_Recv_init	MPI_Win_start
MPI_File_preallocate	MPI_Reduce	MPI_Win_test
MPI_File_read	MPI_Reduce_scatter	MPI_Win_unlock
MPI_File_read_all	MPI_Request_free	MPI_Win_wait
MPI_File_read_at	MPI_Rsend	

Table 6.11 MPI functions for which mpiP gather sent message size data

MPI_Allgather	MPI_Gatherv	MPI_Scan
MPI_Allgatherv	MPI_Ibrecv	MPI_Scatter
MPI_Allreduce	MPI_Irrecv	MPI_Send
MPI_Alltoall	MPI_Isend	MPI_Sendrecv
MPI_Bcast	MPI_Issend	MPI_Sendrecv_replace
MPI_Bsend	MPI_Reduce	MPI_Ssend
MPI_Gather	MPI_Rsend	

Table 6.12 MPI functions for which mpiP gather I/O data

MPI_File_close	MPI_File_read_all	MPI_File_write
MPI_File_open	MPI_File_read_at	MPI_File_write_all
MPI_File_preallocate	MPI_File_seek	MPI_File_write_at
MPI_File_read	MPI_File_set_view	

Table 6.13 MPI functions for which mpiP gather RMA origin data

MPI_Accumulate	MPI_Get	MPI_Put
----------------	---------	---------

Table 6.14 MPI function of target for options (-ba,-bb) of environment variable MPIP

MPI_Allgather	MPI_Allgatherv	MPI_Allreduce
MPI_Alltoall	MPI_Alltoallv	MPI_Bcast
MPI_Gather	MPI_Gatherv	MPI_Reduce
MPI_Reduce_scatter	MPI_Scatter	MPI_Scatterv

Chapter 7 Glossary

Application

An application is an executable program created by a user. Applications processed by the Profiler are categorized as sequential applications and parallel applications. Parallel applications are further categorized as process-parallel applications and thread-parallel applications. Process-parallel applications are categorized as MPI applications. Thread-parallel applications are categorized as OpenMP applications and auto-parallel applications. All applications are categorized based on their compilation format.

The following table shows application categorization.

Table 7.1 Application categorization for the Profiler

Class	Middle class	Subclass	Compilation mode
Sequential application			Sequential compilation
Parallel application	Process-parallel application	MPI application	MPI compilation
	Thread-parallel application	OpenMP application	-Kopenmp specification
		Auto-parallel application	-Kparallel specification

Primary cache

Primary cache refers to the level 1 cache. Primary cache has level 1 data cache (L1D) and level 1 instruction cache (L1I).

Kernel mode

The kernel mode is the mode in which the operating system executes on the processor.

Instant Profiler information

The Instant Profiler information is the information that has been edited to visualize the profiling data.

Instant profiling data

Refer to "[Profiling data](#)".

Cache miss

Cache miss signifies that the instruction or data used by the instruction does not exist in the cache. If a cache miss occurs, the data string, including pertinent data, is accessed (cache line) by a large-scale cache or memory.

Cache line

Cache line is the access unit of data to the cache. If a cache miss occurs, the data, including pertinent data, is accessed by a unit called cache line. Generally, the size of a cache line is from 64 bytes through 256 bytes.

Clock cycle

Clock cycle is the minimum unit for the internal processing time of the processor.

Cost

Cost is the hitting frequency of an instruction that corresponds to a process, thread, procedure, loop, or line of source code by interrupting an application at specific CPU intervals while the application is being executed.

Cost balance information

The cost balance information relates to costs between parallel execution units.

Cycle number

Cycle number is the total of the clock cycle required to execute an instruction line of the performance measurement section.

Sampling

Sampling is the collection of information related to processes, threads, procedures, loops, or lines by interrupting an application at specific user CPU intervals.

Advanced profiling data

Refer to "[Profiling data](#)".

Belonging procedure name

A belonging procedure name is the name of a procedure or a subroutine that the loop and line are constituted.

Measurement section

It is the code section specified by the following functions in the source code of an application.

- Instant Profiler: fipp_start, fipp_stop
- Advanced Profiler: fapp_start, fapp_stop
- Tofu PA: fj_tofupa_start, fj_tofupa_stop

Communication section

It is the code section that calls the communication functions of the MPI library and communicates between processes in an application.

Secondary cache

Secondary cache signifies the level 2 cache. Secondary cache has level 2 data cache (L2D) and level 2 instruction cache (L2I).

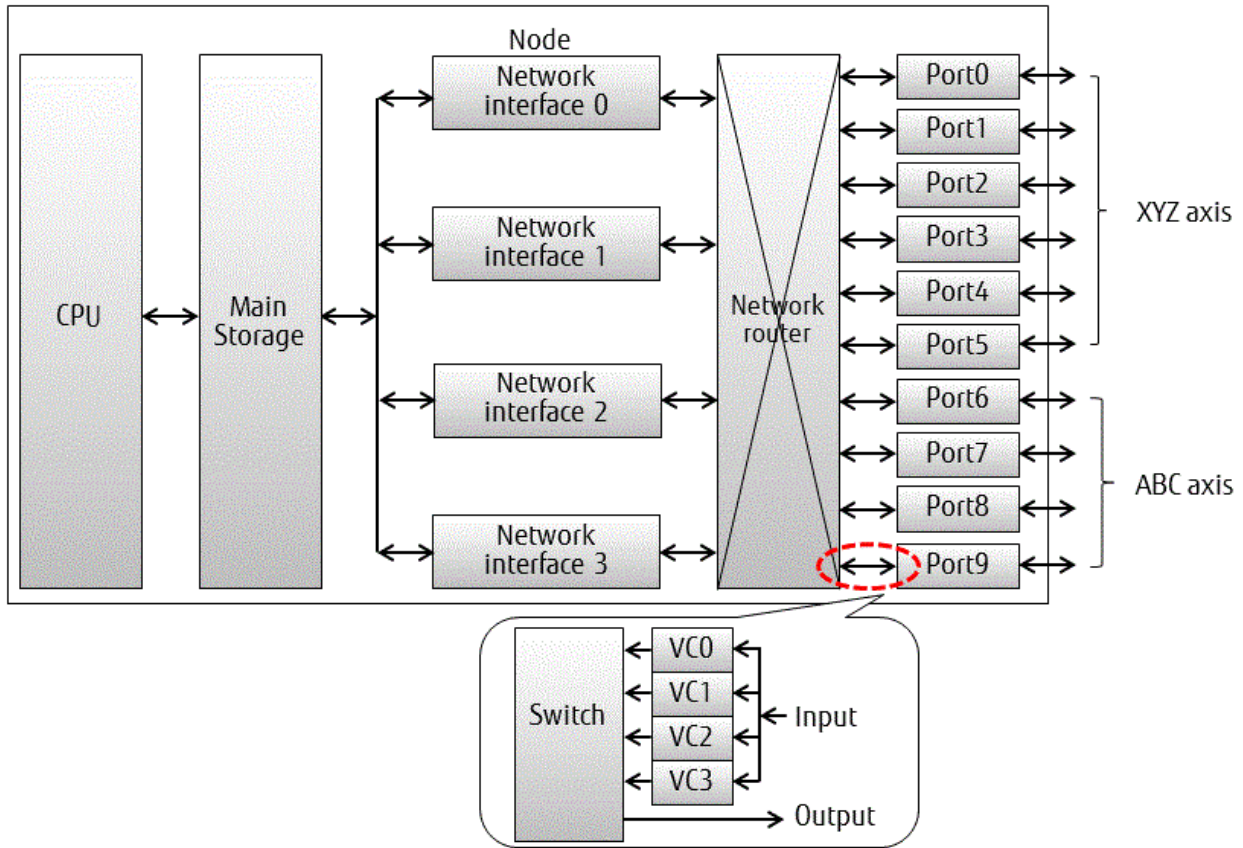
Network interface

It is the interface of a Tofu interconnection. It is connected with the network router of a Tofu interconnection. The network interfaces are installed by four a node (refer to "[Figure 7.1 Node imitative chart](#)").

Network router

It is the router of a Tofu interconnection. It is installed on one, single node (refer to "[Figure 7.1 Node imitative chart](#)").

Figure 7.1 Node imitative chart



 Note

It is pattern diagrams, and does not show the implementation correctly.

Processor frequency

Processor frequency is the total number of clock cycles executed per second.

Profiling data

Profiling data is the performance information collected during one execution of an application. Profiling data consists of one or more files stored in a directory, and these files are called "profiling data files". The profiler uses "profiling data name" as the directory name to identify the directory where profiling data files are stored. Profiling data includes the following:

- Instant profiling data
- Advanced profiling data

Parallel execution unit

Parallel execution unit refers to each thread in a thread-parallel application or each process in a process-parallel application.

Ports (send port, receive port)

It is the interface installed to connect network routers of different nodes with a special cable electrically (refer to "Figure 7.1 Node imitative chart"). In one node, ten ports (X+,X-,Y+,Y-,Z+,Z-,A,B+,B-,C) are installed in each axis of the six dimensions (X,Y,Z,A,B,C) and in each direction (+,- of X,Y,Z,B axis). Two-sided communication (send and receive) is done in one port on each axis and each direction. Two-sided communication is done in one port physically. However, it is considered that two ports with the function (send and receive) are installed individually for convenience, and are referred to as the send port and the receive port.

Memory access throughput

Memory access throughput is the data transfer efficiency between the processor and the memory system. This value increases in an application when data is consumed in large quantity.

User mode

User mode is the mode when an application executes on the processor.

L1 engine

The L1 engine is a device that controls the forwarding between the register and L1D and the forwarding between L1D and L2.

L2 engine

The L2 engine is a device that controls the forwarding between L1D and L2 and the forwarding between L2 and the memory.

MPI library

The MPI library includes MPI subroutines in Fortran or MPI functions in C/C++ language.

Performance Analysis (PA) feature

It is the feature that collects data for performance tuning, monitoring, and troubleshooting.

PA information

It is the data collected by the PA feature.

Translation Lookaside Buffer (TLB)

TLB comprises the data (main) TLB and the instruction TLB.

Transaction Layer Packet (TLP)

TLP is the packet that sends and receives data between transport layers of a Tofu interconnection. The maximum length, including the header, is 2048 bytes. TLP may be created by the communication of MPI and file access.

Virtual Channel (VC)

VC is a method that shows a channel virtually to prevent the deadlock of a packet by using two or more buffers in a torus system network. It is the buffer (refer to "[Figure 7.1 Node imitative chart](#)"). Four VCs per port are supported.

Appendix A Considerations for Using the Profiler

This appendix describes the considerations for using this system.

A.1 Instant Profiler

Programs with short execution time

A program that is executed within one second cannot be analyzed by the Instant Profiler.

Effects of applying modifications

If using a different version of the Instant Profiler or if applying any modification, the title, the items, and the display position of the Instant Profiler may not necessarily be the same. Therefore, the analytical result of the Instant Profiler may be different than before.

Instant Profiler workspace

When the instant profiling data is collected, information of the function included in the library is stored in the Instant Profiler workspace. If the Instant Profiler workspace has a shortage, the following message is output when an application terminated.

```
fipp: work memory overflowed. Specify memsize or more to -m option and retry.
```

In the above message, *memsize* is the recommended size of the Instant Profiler workspace. If this message is output, collect the instant profiling data again by specifying the "-m *memsize*" option of the fipp command. The default size is 3000 * process number * thread number. The unit is Kbyte.

Refer to "[Chapter 2 Instant Profiler](#)" for information on the fipp command and the "-m *memsize*" option.

Environment variable names

Environment variables that start with "FIPP_" or "PROF_" are used by the Instant Profiler. Do not use these environment variables.

The application for measurement

Do not use strip(1) for an executable program. If the symbol is deleted, the instant profiling data cannot be correctly collected. If the current directory of the application being analyzed is changed during execution, specify the absolute path of the instant profiling data to the "-d *profiling_data*" option of the fipp command.

Refer to "[Chapter 2 Instant Profiler](#)" for information on the fipp command and the "-d *profiling_data*" option.

Instant profiling data

Strictly adhere to the following guidelines for the instant profiling data and file. The operation cannot be guaranteed in such cases.

- Do not change the instant profiling data and file (adding , removing, or editing files)
- Do not change the instant profiling data file name

If the measurement program is interrupted during instant profiling data collection by the fipp command, incomplete instant profiling data may remain. In such cases, delete the instant profiling data (file name : DProf_XXXXX).

If there is no file system shared between processes, create the instant profiling data file from the local file system and collect the profiling data using the fipp command.

Sampling interval

The sampling interval for collecting the instant profiling data is rounded off to the nearest multiple of the timer interrupt interval of OS.

But, if the sampling interval that is smaller than the timer interrupt interval is specified, it is rounded up to the timer interrupt interval.

Interval of the timer interrupt depends on the measure against noise of the OS (*1), and it is from about 11 to 14 milliseconds.

The following is an example for comparing of the sampling interval and the timer interrupt interval.

If the timer interrupt interval is 14 milliseconds:

- When the value specified by "-i" option is 10, the sampling interval is 14 milliseconds.
- When the value specified by "-i" option is 25, the sampling interval is 14 milliseconds.
- When the value specified by "-i" option is 100, the sampling interval is 98 (=14*7) milliseconds.

*1 : It is the measure to reduce the impact of system daemons on the job.

SIGVTALRM

The Instant Profiler collects the instant profiling data by catching the signal SIGVTALRM.

When signal SIGVTALRM is caught or is issued by the application, the instant profiling data cannot be correctly collected.

Call Graph information

In case of the Call Graph information, the call route may not be analyzed for the following applications.

When the frame pointer register is not kept

When the nest level of the call route is one, the nest level is output as "<??>". When the nest level of the call route is two or more, Call Graph information is output by mistake.

For example, the made application was measured specifying "-Komitfp" (compiler option). Refer to "Fortran User's Guide", "C User's Guide", or "C++ User's Guide" for details.

Nest level of a call route is 128 or more

The nest level is output as "<??>".

If the cost is in the entry or exit of a procedure, the Call Graph information may be output incorrectly. If the following line is included in line cost information, this phenomenon may occur.

Fortran application

- SUBROUTINE statement
- FUNCTION statement
- ENTRY statement
- RETURN statement
- END statement

C/C++ language applications

- Brackets that show a function start
- Brackets that show a function end
- return statement

The horizontal scrollbar may not operate correctly in the Call Graph information window.

Output of loop information

For C/C++ language applications, specify the optimization option ("-O1" or more) of compilation to measure loop information on the application with the Instant Profiler.

Cost of the shared library

Regarding the cost of the shared library of the system, information on the unit of loop/line cannot be output on analyzing.

If you want to measure the cost of user's shared library with "-L shared" option, do not use strip(1) for the shared library.

When symbols are removed, the instant profiling data is not correctly collected.

Include file and inline expansion

In case of applications created by the source code containing the include file, the cost information in the include file is output by the line number of the include file. Additionally, for applications with inline expansion, the cost information in inline expansion is output by the line number of the procedure of inline expansion origin.

-p all option

If the "-p all" option is specified for the fippxx command, information on all processes is output. Especially, when a parallel process is large, it takes time for processing. Information gathering processing (fipp -C) does the information gathering for all processes, and the instant profiling data is preserved. In information analysis processing (fippxx -A), we will recommend limiting the process by using the instant profiling data that is not output information on all the processes at a time but is preserved, divided into plurals and outputting it.

-P userfunc option

If the "-P userfunc" option of the fipp command is used, the cost information cannot be output correctly for the following applications.

- Fortran/C/C++ language applications compiled with "-Nnoline" option

For such applications, confirm the actual cost by again executing with the "-P nouserfunc" option.

Maximum number of processes

The maximum number of processes supported by the Profiler GUI feature is 9216. The instant profiling data that exceeds this count cannot be displayed by the Profiler GUI feature.

Display limit of the Topology information window

The maximum number of directions available in the overall view of the Topology information window is given below. If the upper bound is exceeded by even one direction, it cannot be displayed.

- One-dimensional shape : Same as the maximum number of processes
- Two-dimensional shape : X=512, Y=64
- Three-dimensional shape : X=32, Y=18, Z=16

MPI applications

If an MPI function of a dynamic process creation is executed, the operation cannot be guaranteed.

Hardware monitor information

If the FLIB_FASTOMP environment variable is FALSE, the Hardware monitor information cannot be correctly acquired. Refer to the "Fortran User's Guide", "C User's Guide", or "C++ User's Guide" for FLIB_FASTOMP.

Color histogram

If the difference between the min and the max is less than 0.1, the thumb may not operate correctly.

Applications that generates processes

In an application, do not generate a process with, for example, fork, vfork, the popen system call, or system functions. If it is generated, the instant profiling data cannot be correctly collected.

Source code information

The source code information is not output in case of the following conditions:

- The Instant Profiler information is output in the text format.
- There is a cost in the user procedure.
- In procedure cost distribution information, the user procedure does not exist in the top five ranks.

The following message is output if any of the above conditions is met.

Symbol information up to the 5th do not include information which relates to the source code.

And, only one source code information is displayed to one user procedure in the output of the text format.

For example, in case of inline expansion, one user procedure might be composed by two or more source codes. In this instance, the source code which has smaller beginning line number of the procedure of inline function or its caller is only displayed. If the beginning line number of the procedure is the same, the source code information that has been expanded inline in first is displayed.

Runtime Information Output Function

It is not possible to use it at the same time as Runtime Information Output Function.

Please refer to "Runtime Information Output Function User's Guide" for Runtime Information Output Function.

MPI library cost

The MPI library cost of the cost information is not correctly output in the following applications.

- Fortran/C/C++ language applications compiled with "-Nnoline" option

DT_RPATH

If the DT_RPATH dynamic section attribute exists in an executable application and /opt/FJSVmxlang/lib64 is included in the DT_RPATH attribute, it fails that the Instant Profiler gathers information.

When environment variable LD_RUN_PATH or "-rpath" option is specified at compiling, DT_RPATH is set to the execution application. Confirm whether DT_RPATH exists in the execution application by outputting the readelf -d command. If "(RPATH)" exists in Type of output information, DT_RPATH is set to the execution application.

Please use the application where /opt/FJSVmxlang/lib64 is not included in the DT_RPATH attribute to gather information. (Relinking application and so on)

Node-sharing job

If the job type is node-sharing job, the operation of the profiler cannot be guaranteed.

It fails that the profiler collects the instant profiling data for MPI execution. And the processing of the MPI application halts.

Refer to the "Job Operation Software End-user's Guide" for more information on node-sharing job.

Cost information

Even if different entities are used on the function with the same name, the cost of the operation is calculated as for one function.

When the optimization option ("-O1" or more) is specified at compiling, line information on each instruction might be different from the line of the source according to the influence of optimization.

For example, the line information on each order will be the start position of a loop, and the cost is calculated according to this start position.

pthread

The Instant Profiler works in cooperation with the Fujitsu compiler.

Therefore, thread parallel information on "automatic parallelization by the Fujitsu compiler" and "OpenMP parallelization" is collected.

Thread parallel information on "pthread parallelization" is not collected.

MPI profiling interface

The tool library is given to priority more than user's libraries at linking when "-Ntl_notrt" is not specified (the default is "-Ntl_trt") because the Instant Profiler, the Advanced Profiler, the Tracer, and the mpiP functions are implemented by hooking MPI functions. Therefore, the tool functions cannot be used together with the MPI profiling interface.

COARRAY feature

Note the following when "-Ncoarray" (compiler option) is enabled.

- The value in which one is added to the rank number or the process number corresponds to the image index.

- The cost might be post as for the MPI library that COARRAY used.

Refer to the "Fortran User's Guide" and "Fortran User's Guide Additional Volume COARRAY" for "-Ncoarray" option.

Link Time Optimization

The following information which the profiler outputs may not be guaranteed, when "-Klto" (compiler option) is enabled.

- Call graph information.

The function names in call graph information may include the function name which is generated internally by link time optimization.

- Source code information.

The each line cost may not be displayed correctly.

A.2 Advanced Profiler

Effects of applying modifications

If using a different version of the Advanced Profiler or if applying any modification, the title, the items, and the display position of the Advanced Profiler may not necessarily be the same. Therefore, the analytical result of the Advanced Profiler may be different than before.

Environment variable names

Environment variables that start with "FAPP_" are used by the Advanced Profiler. Do not use these environment variables.

Group name

If the following characters are used, the Profiler GUI feature cannot be analyzed.

" ' < > &

The application for measurement

Do not use strip(1) for an executable program. If the symbol is deleted, the advanced profiling data cannot be correctly collected. If an application that changes the current directory in execution is analyzed, specify the absolute path of the advanced profiling data to the "-d *profiling_data*" option of the fapp command.

Refer to "[3.2.5 fapp command](#)" for information on the fapp command and the "-d *profiling_data*" option.

Advanced profiling data

Strictly adhere to the following guidelines for the advanced profiling data and file. The operation cannot be guaranteed in such cases.

- Do not change the advanced profiling data and file (File increasing, file decreasing, or editing)
- Do not change the advanced profiling data file name

If the measurement program is interrupted during advanced profiling data collection by the fapp command, incomplete advanced profiling data may remain. In such cases, delete the advanced profiling data (file name : EProf_XXXX). If there is no file system shared between processes, create the advanced profiling data file from the local file system and collect the profiling data using the fapp command.

Elapsed time information of the MPI library

In case of applications created by the mpiFCC command, the member function name of the C++ language is output to the MPI library name of the elapsed time information. The elapsed time information of an MPI function called from a child thread cannot be analyzed by the Advanced Profiler.

MPI applications

If an MPI function of a dynamic process creation is executed, the operation cannot be guaranteed. The MPI_THREAD_SERIALIZED and MPI_THREAD_MULTIPLE (level of thread support) are not supported.

Hardware monitor information

If the FLIB_FASTOMP environment variable is FALSE, the Hardware monitor information cannot be acquired correctly. Refer to the "Fortran User's Guide", "C User's Guide", or "C++ User's Guide" for information on FLIB_FASTOMP.

When the "-Hmethod=raw" option is specified, processing such as direct I/O to global file systems might fail in Hardware monitor information measurements. Either perform measurements outside global file systems, or avoid I/O instructions when making measurements.

When the "-Hmethod=raw" option is specified, information while the measured process is not being allocated to CPU with Sleep is measured. For instance, the value of execution time (second) of the Precision PA visibility function (Excel format) might become large compared with the case to specify "-Hmethod=normal" option.

Maximum number of processes

The maximum number of processes supported by the Profiler GUI feature is 9216. The advanced profiling data that exceeds this count cannot be displayed by the Profiler GUI feature.

Display limit of the Topology information window

The maximum number of directions available in the overall view of the Topology information window is given below. If the upper bound is exceeded by even one direction, it cannot be displayed.

- One-dimensional shape : Same as the maximum number of processes
- Two-dimensional shape : X=512, Y=64
- Three-dimensional shape : X=32, Y=18, Z=16

Color histogram

If the difference between the min and the max is less than 0.1, the thumb may not operate correctly.

Applications that generates processes

In an application, do not generate a process with, for example, fork, vfork, the popen system call, or system functions. If it is generated, the advanced profiling data cannot be collected correctly.

Runtime Information Output Function

It is not possible to use it at the same time as Runtime Information Output Function.

Please refer to "Runtime Information Output Function User's Guide" for Runtime Information Output Function.

DT_RPATH

If the DT_RPATH dynamic section attribute exists in an executable application and /opt/FJFSVmxlang/lib64 is included in the DT_RPATH attribute, it fails that the Advanced Profiler gathers information.

When environment variable LD_RUN_PATH or "-rpath" option is specified at compiling, DT_RPATH is set to the execution application. Confirm whether DT_RPATH exists in the execution application by outputting the readelf "-d" command. If "(RPATH)" exists in Type of output information, DT_RPATH is set to the execution application.

Please use the application where /opt/FJFSVmxlang/lib64 is not included in the DT_RPATH attribute to gather information. (Relinking application and so on)

Node-sharing job

If the job type is node-sharing job, the operation of the profiler cannot be guaranteed.

It fails that the profiler collects the advanced profiling data for MPI execution. And the processing of the MPI application halts.

Refer to the "Job Operation Software End-user's Guide" for more information on node-sharing job.

pthread

The Advanced Profiler works in cooperation with the Fujitsu compiler.

Therefore, thread parallel information on "automatic parallelization by the Fujitsu compiler" and "OpenMP parallelization" is collected. Thread parallel information on "pthread parallelization" is not collected.

MPI profiling interface

The tool library is given to priority more than user's libraries at linking when "-Ntl_notrt" is not specified (the default is "-Ntl_trt") because the Instant Profiler, the Advanced Profiler, the Tracer, and the mpiP functions are implemented by hooking MPI functions. Therefore, the tool functions cannot be used together with the MPI profiling interface.

COARRAY feature

Note the following when "-Ncoarray" (compiler option) is enabled.

- The value in which one is added to the rank number or the process number corresponds to the image index.
- The MPI library information that COARRAY used is also output to MPI information.

Refer to the "Fortran User's Guide" and "Fortran User's Guide Additional Volume COARRAY" for "-Ncoarray" option.

-H event_number option

If the "-Hevent_number" option is specified at the same time as "-Hmethod=normal" option, the same PA event cannot be specified for two or more PIC.

The following error message is output and processing ends.

```
RTINF2xxx : Internal error. PAPI return code = xxx.
```

A.3 Tracer

Temporary local trace data file

Operations when the current directory is changed, for example, by the chdir(2) system call, in a traced application cannot be secured.

If it is necessary to use, for example, the chdir(2) system call, specify the directory name using the full path to the VT_PFORM_LDIR environment variable.

Environment variable names

If environment variables that start with "VT_" other than those described in this guide are used, the operation cannot be guaranteed.

Function names

- If functions that start with "VT_" other than those described in this guide are used, the operation cannot be guaranteed.
- If functions that start with "PMPI_" are defined in an MPI program, the Tracer operations cannot be guaranteed.

MPI trace

- If a program is compulsorily ended by the MPI_Abort or MPI_Cancel functions, an incomplete local trace data file may be output. Refer to "4.2.3.5 Trace data files" for information on local trace data files.
- MPI_THREAD_SERIALIZED and MPI_THREAD_MULTIPLE (level of thread support) of the MPI_Init_thread function are not supported.
- If an MPI program of an invalid communicator or a group operation is executed, the operation cannot be guaranteed.
- When the MPI trace feature by Fortran is used, the following functions are not considered to be a trace object.

MPI_Comm_create_keyval	MPI_Comm_get_attr	MPI_Comm_set_attr
MPI_Type_create_hvector	MPI_Type_create_keyval	MPI_Type_get_attr

MPI_Type_match_size	MPI_Type_set_attr	MPI_Win_create_keyval
MPI_Win_get_attr	MPI_Win_set_attr	

I/O trace

Perform the following operation when you trace information on fscanf, getc and putc function by using the I/O trace feature.

- C language(only fscanf)

Specify "-ansi" option at compiling, and compile based on the C89 specification.

Refer to "C User's Guide" for "-ansi" option.

- C++ language(fscanf, getc, putc)

Add the "#undef function-name" directive behind the part where stdio.h is included in the application.



Example

Example of fscanf

```
#include <stdio.h>
#undef fscanf
int main(){
    :
```

The tracer does not trace the linguistic level of Fortran.

The tracer traces the call of the LIBC I/O function.

Therefore, one I/O processing to the file might be divided into two or more LIBC I/O functions by linguistic level Fortran and it be recorded in the events file.

Memory trace

The tracer does not trace the linguistic level of Fortran.

The tracer traces the call of the LIBC memory function.

Therefore, one memory processing might be divided into two or more LIBC memory functions by linguistic level and it be recorded in the events file.

COARRAY feature

Tracer is not intended for COARRAY feature. The following error message is output and processing ends.

```
VampirTrace: FATAL: Cannot find window
```

vtunify-mpi application

If you want to run the vtunify-mpi application, please set the environment variable FLIB_FASTOMP to FALSE. If you do not have this configuration, it is an error of jwe1041i-s and the vtunify-mpi execution ends.

Refer to the "Fortran User's Guide", "C User's Guide", or "C++ User's Guide" for FLIB_FASTOMP. Refer to the "Fortran/C/C++ Runtime Messages" for more information about the error.

A.4 Tofu PA

Simultaneous use with the Profiler

Tofu PA cannot be used concurrently with the Instant Profiler, Advanced Profiler, Tracer, and mpiP. Moreover, it cannot be used concurrently with the tool, library, and application that hook following functions:

- MPI_Init
- MPI_Init_thread

- MPI_Finalize function

Node-sharing job

If the job type is node-sharing job, the operation of the Tofu PA cannot be guaranteed.

Refer to the "Job Operation Software End-user's Guide" for more information on node-sharing job.

A.5 Open Source Profiler

MPMD model

Statistical information of the program executed by the MPMD model cannot be collected.

Simultaneous use with the Profiler

mpiP cannot be used concurrently with the Instant Profiler, Advanced Profiler, Tracer, and Tofu PA. Moreover, it cannot be used concurrently with the tool, library, and application that hook following functions:

- MPI_Init
- MPI_Init_thread
- MPI_Finalize function

Function names

If functions that start with "PMPI_" are defined in an MPI program, the mpiP operations cannot be guaranteed.

MPI applications

MPI_THREAD_SERIALIZED and MPI_THREAD_MULTIPLE (level of thread support) of the MPI_Init_thread function are not supported.

Options (-ba,-bb) of environment variable MPIP

When you execute the program specifying "-ba,-bb" in the option, MPI_Barrier is inserted before or after collective communication functions.

Therefore, the execution time of the program might be longer compared to the case where "-ba,-bb" is not specified.

COARRAY feature

The MPI function information that COARRAY internally uses cannot not be collected and be displayed.

MPI_Init/MPI_Finalize is inserted in the COARRAY program, and the MPI function information executed between those can be collected and be displayed.

Appendix B Troubleshooting

B.1 Instant Profiler

When the instant profiling data is collected, the execution time takes longer than usual

Specify a longer sampling interval using the "-i" option of the fipp command to decrease the execution time of the Instant Profiler. Refer to ["2.2.3 fipp command"](#) for information on the fipp command.

A procedure name (such as the library name) that does not exist in the source code is output in the Instant Profiler information

Specify the "-P userfunc" option of the fipp command to collect only user procedures in the Instant Profiler information. Refer to ["2.2.3 fipp command"](#) for information on the fipp command.

The instant profiling data file cannot be opened.

The application for which the instant profiling data is to be collected may not have ended normally. Collect the instant profiling data again.

The symbol "__?unknown" is output

The cost may not correspond to any procedure on collecting the instant profiling data. The cost is output as the "__?unknown" symbol. If the sampling interval is increased when collecting the instant profiling data, this symbol may not be output. Refer to ["2.2.3 fipp command"](#) for information on how to specify the sampling interval.

B.2 Advanced Profiler

When the advanced profiling data is collected, the execution time takes longer than usual

Reduce the number of measured sections or calls of the Advanced Profiler routine to decrease the execution time of the Advanced Profiler.

The advanced profiling data file cannot be opened

The application for which the advanced profiling data is to be collected may not have ended normally. Collect the advanced profiling data again.

B.3 Tracer

Failure to output the local trace data file

When an application is executed, the following error message may be output due to memory shortage of the compute node, and then the application terminates.

```
VampirTrace: FATAL: Failed to execute /usr/bin/nm --demangle --line-numbers $WORK_DIR/a.out
Please set the environment variable VT_GNU_NM to the 'nm' command including command line switches
which lists symbol/addresses of an object file in BSD-style or set VT_GNU_NMFILE to a pre-created
symbol list file.
```

Specify the file that records the symbol information for the VT_GNU_NMFILE environment variable.

Refer to ["4.2.2.1 Environment variables for execution"](#) for information on how to specify the environment variable.

The MPI application information is not collected

"no" may be specified for the VT_MPITRACE environment variable. Specify the value "yes". Refer to ["4.2.2.1 Environment variables for execution"](#) for information on the environment variable.

B.4 Tofu PA

The file is not output

Check if the FJ_TOFUPA_LEVEL environment variable is set correctly.

Appendix C Notes on Migration from FX10 system to FX100 system

This appendix provides notes on migrating from FX10 system on V1.0L30 (Generation number:09 or later) to FX100 system.

If migrating from FX10 system on V1.0L30 (Generation number:08 or earlier), refer to "[Appendix D Compatibility Information \(FX10 system\)](#)" also.

C.1 Measured information of Hardware monitor information in Instant Profiler is changed

a. Changes

The measured information of the Hardware monitor information in Instant Profiler is changed.

[Previous version]

The Hardware monitor information of Instant Profiler outputted "Elapsed time", "MFLOPS", "MFLOPS peak performance rate", "MIPS", "MIPS peak performance rate", "Memory access throughput (unit of chip)", "Memory access throughput (chip) / PEAK", and "SIMD instruction rate".

[This version]

The Hardware monitor information (Statistics) of Instant Profiler outputs "Elapsed time", "MFLOPS", "MFLOPS peak performance rate", "MIPS", "MIPS peak performance rate", and "Floating-point arithmetic instruction rate".

"Memory access throughput (unit of chip)" and "Memory access throughput (chip) /PEAK)" are output as memory throughput information in each core by specifying "-Hevent=MEM_access".

"SIMD instruction rate" is output by specifying "-Hevent=Instructions_SIMD".

b. Influence

The measurement items output in the Hardware monitor information in Instant Profiler

The measurement items output in the Hardware monitor information of Instant Profiler are measured separately for events "-Hevent=Statistics", "-Hevent=MEM_access" and "-Hevent=Instructions_SIMD".

"Memory access throughput (unit of chip)" is output as memory throughput information in each core by specifying "-Hevent=MEM_access".

c. Coping

Execute the fipp command specifying the event being included for measurement information to be acquired in "-Hevent option".

Refer to measurement information on Hardware monitor information for more information.

C.2 Measured information of Hardware monitor information in Advanced Profiler is changed

a. Changes

The measured information of Hardware monitor information in Advanced Profiler is changed.

[Previous version]

The following measurement events were in the measured information of Hardware monitor information.

- Cache
- Instructions
- MEM_access
- Performance

- Statistics

[This version]

The following measurement events are in the measured information of Hardware monitor information.

- Cache
- Instructions_SIMD
- Instructions_NOSIMD
- MEM_access
- Performance
- Statistics
- TLB

b. Influence

The event including necessary measurement information might change.

"Memory access throughput (unit of chip)" information collected with Statistics is output with MEM_access as information on the memory access throughput in each core.

c. Coping

Specifies the event name including necessary measurement information in "-Hevent" option and executes the fapp command.

C.3 Frequency of the collection data and the analyzing data for the precision PA visibility function (Excel format) in Advanced Profiler is changed

a. Changes

The frequency of the collection data and the analyzing data is changed from 7 to 11.

[Previous version]

The collecting data and the analyzing data were executed 7 times for displaying the precision PA visibility function (Excel format).

[This version]

The collecting data and the analyzing data are executed 11 times for displaying the precision PA visibility function (Excel format).

b. Influence

The collecting data and the analyzing data are executed 11 times for displaying the precision PA visibility function (Excel format).

c. Coping

[Collecting data]

Adds the 8th, 9th, 10th, and 11th data collection processing.

```
fapp -C -d pa8 -Hpa=8 mpiexec -n 8 ./a.out
fapp -C -d pa9 -Hpa=9 mpiexec -n 8 ./a.out
fapp -C -d pa10 -Hpa=10 mpiexec -n 8 ./a.out
fapp -C -d pa11 -Hpa=11 mpiexec -n 8 ./a.out
```

[Analyzing data]

Adds the 8th, 9th, 10th, and 11th data conversion processing, and prepare the csv file.

```
fapppx -A -d pa8 -o output_prof_8.csv -tcsv -Hpa
fapppx -A -d pa9 -o output_prof_9.csv -tcsv -Hpa
```



```
fappxx -A -d pa10 -o output_prof_10.csv -tcsv -Hpa
fappxx -A -d pa11 -o output_prof_11.csv -tcsv -Hpa
```

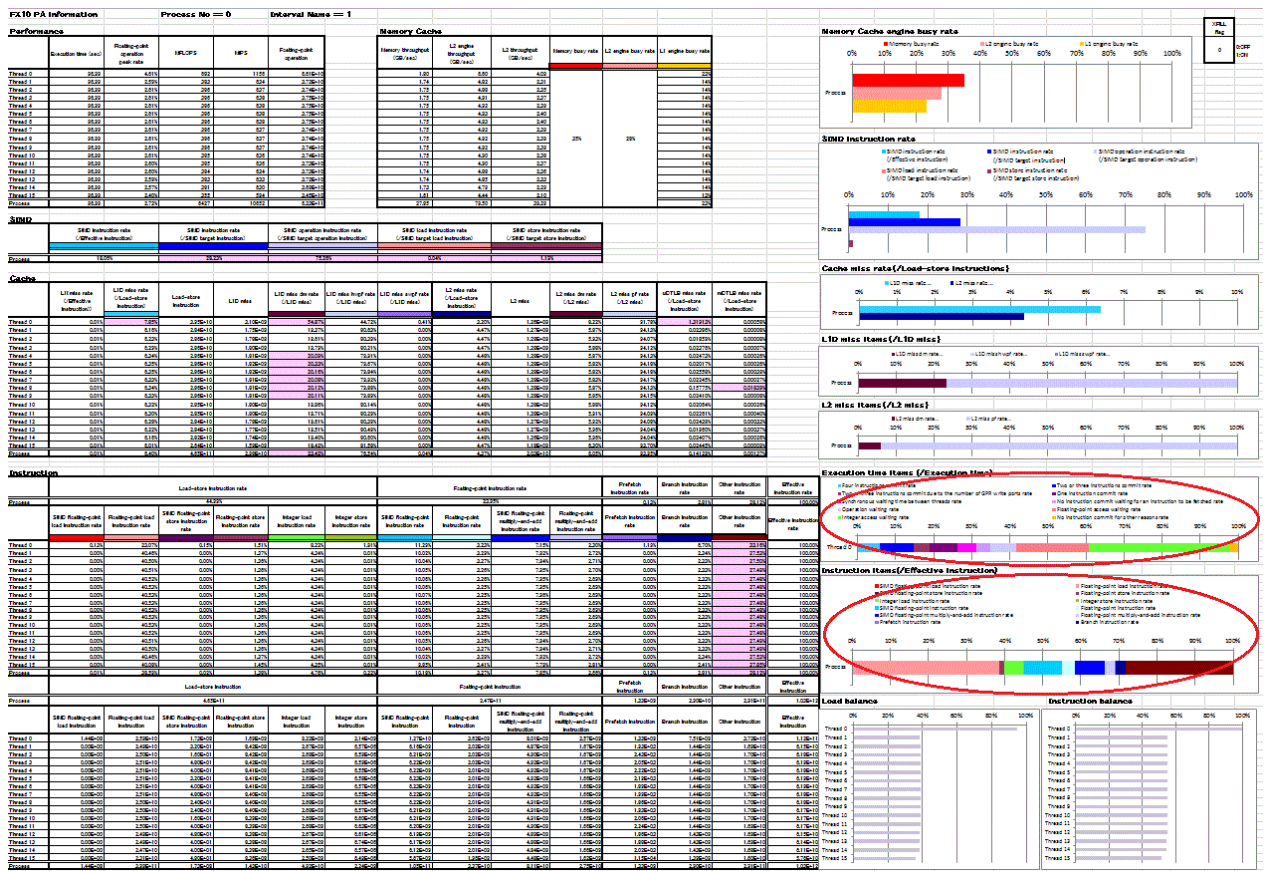
It becomes an error when there are no 8th, 9th, 10th, and 11th csv files.

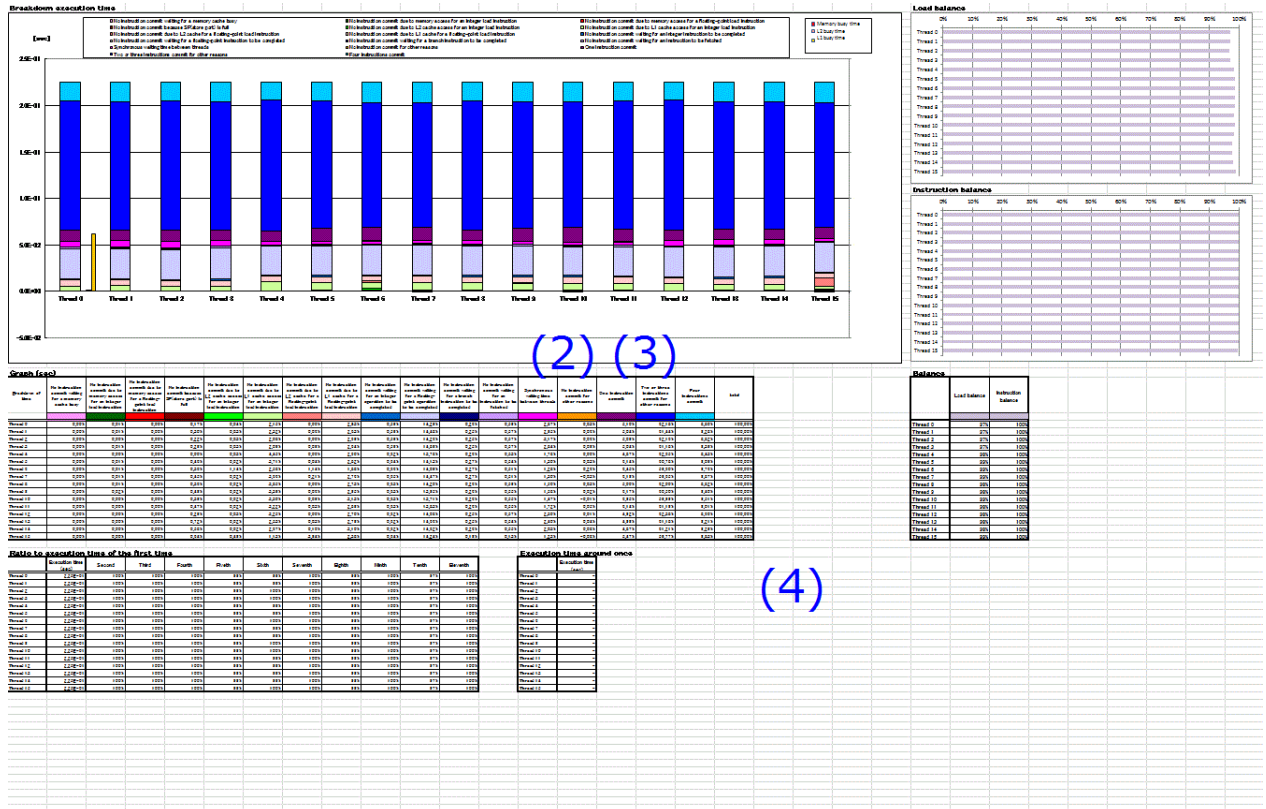
C.4 Presentation item of the precision PA visibility function (Excel format) in Advanced Profiler is changed

- Changes
- "Execution time items (/Execution time)", "No instruction commit waiting for a micro-operation to be completed", and "Two or three instructions commit due to the number of GPR write ports" are deleted from the display items of the precision PA visibility function (Excel format).

[Previous version]

The deleted part is shown in red.





c. Influence

- In (1), "Execution time items (/Execution time)" and "Instruction items (/Effective instruction)" are deleted, and "Load balance" and "Instruction balance" are moved to second page.
- In (2), "Breakdown of time" in "Graph element details" is deleted.
- In (2), "No instruction commit waiting for a micro-operation to be completed" is deleted from display items of the Graph.
- In (3), "Two or three instructions commit due to the number of GPR write ports" is deleted from display items of Graph.
- In (4), "Execution time items (outline)" is deleted.

d. Coping

- "Execution time items (/Execution time)" is confirmed from "Breakdown execution time".
- "Instruction items (/Effective instruction)" is confirmed from "Rate of time breakdown (%)".
- "Breakdown of time" of "Graph element details" is confirmed from "Rate of time breakdown (%)".
- The value counted in "No instruction commit waiting for a micro-operation to be completed" is included in "No instruction commit for other reasons".
- The value counted in "Two or three instructions commit due to the number of GPR write ports" is included in "Two or three instructions commit for other reasons".
- "Execution time items (outline)" is confirmed from "Rate of time breakdown (%)".

Appendix D Compatibility Information (FX10 system)

D.1 Migration to V1.0L30(Generation Number:09)

D.1.1 The -c option of the vtunifypx command and vtunify-mpi application of the tracer is abolished

a. Changes

The "-c" option of the vtunifypx command and vtunify-mpi application of the tracer is abolished.

[Previous version]

The trace data of non-compressed format was able to be output by the "-c" option specification of the vtunifypx command and vtunify-mpi application.

[This version]

If "-c" option of the vtunifypx command and vtunify-mpi application is specified, the trace data of the error end doing and non-compressed format cannot be output.

b. Influence

The following error messages are output and processing ends if "-c" option was specified.

vtunifypx

invalid option -- -c

vtunify-mpi

invalid option -- -c

c. Coping

Specify --nocompress option if you want to output the trace data file of non-compressed format.

D.1.2 VT_MAX_MPI_COMMS/VT_MAX_MPI_WINS of the environment variable for execution is abolished

a. Changes

VT_MAX_MPI_COMMS/VT_MAX_MPI_WINS of the environment variable for execution is abolished.

[Previous version]

The maximum number of communicator/window used by specifying environment variable for execution VT_MAX_MPI_COMMS/VT_MAX_MPI_WINS in the application (Both defaults are 100) was able to be changed.

[This version]

The limitation of the maximum number of communicator/window used in the application was removed therefore, the user does not consider the maximum number and create the application becomes possible.

Because environment variable for execution VT_MAX_MPI_COMMS/VT_MAX_MPI_WINS was abolished.

b. Influence

Even if environment variable VT_MAX_MPI_COMMS/VT_MAX_MPI_WINS is specified, it does not become effective.

Moreover, there is no influence on the user who has already specified these environment variables because the interruption of processing by the error end is not generated by specifying these environment variables.

c. Coping

There is no method of the object. However, there is no influence on the user like the description in "b. Influence").

D.1.3 The record in the trace data of the MPI_Address function is abolished

a. Changes

The MPI_Address function is excluded from the MPI function collect by the Tracer.

[Previous version]

The MPI_Address function called in the application could be recorded in the trace data.

[This version]

The MPI_Address function called in the application cannot be recorded in the trace data.

b. Influence

The MPI_Address function called in the application cannot be recorded in the trace data.

c. Coping

There is no alternate method for the change of the specification of OSS.