

FUJITSU Software

Technical Computing Suite V2.0



Runtime Information Output Function

User's Guide

(PRIMEHPC FX100)

J2UL-1879-02ENZ0(00)
November 2015

Preface

Purpose of This Manual

This manual describes Fortran, C, and C++ program Runtime Information Output Function (called "this function" hereafter).

Intended Readers

The intended readers of this manual are those using this function to measure the performance of the executable program. In addition to knowledge of performance information (e.g., MFLOPS) and of programming in Fortran, C, and C++, readers need to have a basic knowledge of Linux commands.

Organization of This Manual

This manual is organized as follows:

[Chapter 1 Overview](#)

An overview of this function

[Chapter 2 Usage](#)

How to use this function

[Chapter 3 Measurement Range](#)

How to output information that fetched a part of the program as a measurement range

[Chapter 4 Output](#)

How to control the output format

[Chapter 5 Parallelization Information](#)

Explanation about Parallelization Information

[Chapter 6 Cost Information](#)

Explanation about Cost Information

[Chapter 7 Input-Output Information](#)

Explanation about Input-Output Information

[Chapter 8 Hardware Monitor Information](#)

Explanation about Hardware Monitor Information

[Chapter 9 Message](#)

Explanation about execution messages

[Appendix A Attention in The Use](#)

Restrictions on this function

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

Explanation about notes when migrating from FX10 to FX100

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

- OpenMP is a trademark of the OpenMP Architecture Review Board.
- 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-1879-02ENZ0(00)
October 2014, 1st Version	J2UL-1879-01ENZ0(00)

Copyright

Copyright FUJITSU LIMITED 2014-2015

Update History

Changes	Location	Version
Fixed the error in writing.	-	2nd Version

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

Contents

Chapter 1 Overview.....	1
Chapter 2 Usage.....	2
2.1 Execution Environment.....	2
2.2 Compiler Options.....	2
2.3 Runtime Environment Variables.....	3
2.4 Range of Obtainable Information.....	5
2.4.1 Information Fetched for Each Routine.....	5
2.4.2 Information Fetched for Each Loop.....	6
2.5 Output Information List.....	6
Chapter 3 Measurement Range.....	13
3.1 Entire Program.....	13
3.2 Range Specification.....	13
3.2.1 Fortran.....	13
3.2.1.1 Error processing.....	15
3.2.2 C/C++.....	15
3.2.2.1 Error processing.....	16
Chapter 4 Output.....	18
4.1 Output format.....	18
4.1.1 Text format.....	20
4.1.2 CSV format.....	20
4.2 Serial Programs or Multi-Thread Programs.....	21
4.3 MPI Programs or Hybrid Programs.....	21
Chapter 5 Parallelization Information.....	23
5.1 Overview of Parallelization Information.....	23
5.2 Details of Parallelization Information.....	23
5.2.1 Information for Each Routine.....	25
5.3 Output Example of Parallelization Information.....	26
Chapter 6 Cost Information.....	27
6.1 Overview of Cost Information.....	27
6.2 Details of Cost Information.....	27
6.2.1 Range from Start to End.....	27
6.2.2 Parallelized regions.....	27
6.2.3 Each Routine.....	28
6.2.4 Each Loop.....	29
6.3 Output Example of Cost Information.....	29
Chapter 7 Input-Output Information.....	34
7.1 Specification of Input Output Information.....	34
7.2 Fortran Input-Output Information.....	34
7.2.1 Overview of Fortran Input-Output Information.....	34
7.2.2 Details of Fortran Input-Output Information.....	35
7.2.2.1 Output specification.....	35
7.2.2.2 Unit Information.....	35
7.2.2.3 Statement Information.....	36
7.2.2.4 System Call Cumulative Information.....	37
7.2.2.5 Information of Asynchronous Input-Output.....	37
7.2.2.6 Range Specification.....	38
7.2.3 Output Example of Fortran Input-Output Information.....	38
7.3 C/C++ Input-Output Information.....	40
7.3.1 Overview of Fortran Input-Output Information.....	40
7.3.2 Details of Fortran Input-Output Information.....	40

7.3.3 Output Example of C/C++ Input-Output Information.....	42
Chapter 8 Hardware Monitor Information.....	44
8.1 Overview of Hardware Monitor Information.....	44
8.2 Details of Hardware Monitor Information.....	44
8.2.1 Fetching Information in Each Category.....	44
8.2.2 Switching Collection Mode.....	46
8.2.3 Fetching for Each Routine.....	47
8.3 Calculation Expressions for Hardware Monitor Information.....	47
8.3.1 "Statistics" Specified for Category Name.....	47
8.3.2 "Instructions_SIMD" Specified for Category Name.....	48
8.3.3 "Instructions_NOSIMD" Specified for Category Name.....	50
8.3.4 "Cache" Specified for Category Name.....	51
8.3.5 "MEM_access" Specified for Category Name.....	52
8.3.6 "Performance" Specified for Category Name.....	53
8.3.7 "TLB" Specified for Category Name.....	55
8.4 Output Example of Hardware Monitor Information.....	56
Chapter 9 Message.....	58
Appendix A Attention in The Use.....	64
Appendix B Notes on Migration from FX10 system to FX100 system.....	66
B.1 Change in Category of Hardware Monitor Information.....	66

Chapter 1 Overview

This chapter describes an overview of this function.

This function outputs Parallelization Ratio and Hardware Monitor Information, that is information (called "Runtime Information" hereafter) for speedup and performance confirmation of the program, by specifying the compiler option and setting the environment variable, when the program is executed.

The information output by this function is as follows.

- Parallelization Information
- Cost Information
- Input-Output Information
- Hardware Monitor Information



The routine supported this function is called "service routine" in this manual.

However, it is called service subroutine at the description that corresponds only to Fortran, and called service function at the description that corresponds only to C and C++.

In addition, the term "routine" used in this manual has the same meaning as a procedure in Fortran or a function in C and C++.

Chapter 2 Usage

This chapter describes how to use this function.

2.1 Execution Environment

The settings below are required used to this function.

Table 2.1 Append the following path name to user environment variable PATH and LD_LIBRARY_PATH

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

2.2 Compiler Options

This section describes the compiler options for use this function.

-N{ rt_tune | rt_notune }

- This option specifies whether or not Runtime Information is output. The default is `-Nrt_notune`.
- Even when only link edit, if the object program compiled by this is included, this option must be specified.

`-Nrt_tune` Runtime Information is output.
`-Nrt_notune` Runtime Information is not output.

-Nrt_tune_io

- This option must specify to fetch Input-Output Information of C/C++
- `-Nrt_tune_io` option sets `-Nrt_tune` option. When the compiler option `-Nrt_notune` is specified after `-Nrt_tune_io` option, `-Nrt_tune_io` option becomes invalid.
- The source program compiled without specifying this option is not output I/O information on C/C++.

-Nrt_tune_func

- In addition to the `-Nrt_tune` output, Runtime Information about each routine is output.
- `-Nrt_tune_func` option sets `-Nrt_tune` option. When the compiler option `-Nrt_notune` is specified after `-Nrt_tune_func` option, Runtime Information is not output.

-Nrt_tune_loop[={ all | innermost }]

- In addition to the `-Nrt_tune` output, runtime information about each loop is output.
- `-Nrt_tune_loop` option sets `-Nrt_tune` option. When the compiler option `-Nrt_notune` is specified after `-Nrt_tune_loop` option, Runtime Information is not output.
- If `-Nrt_tune_loop` is specified without argument, `-Nrt_tune_loop=all` is effective.

`-Nrt_tune_loop=all` Runtime Information measured all loop is output.
`-Nrt_tune_loop=innermost` Runtime Information measured innermost loop is output.



The loop of this case means the thing which deletion was not carried out for by optimization among DO loop of the Fortran, loop (for/while/if-goto) for the optimization of the C. The loop which is not nested is included in the innermost loop.

Notes when Runtime Information is fetched are shown below.

- When FALSE is specified for environment variable FLIB_FASTOMP, information cannot be correctly fetched. Refer to following manuals for environment variable FLIB_FASTOMP.
 - "Fortran User's Guide"
 - "C User's Guide"
 - "C++ User's Guide"
- When there are goto statement, stop statement (Fortran), setjmp()/longjmp()(C/C++), throw statement (C++) during a program or an exceptions is checked, information may be not correctly fetched.
- There is a possibility of receiving the following influences by optimization of the compiler.
 - When fetched the information each loop, it may be different which loop becomes a target of the measurement.
 - Line number may not accord with line number of the source program.
 - The nest relations of the loop may not accord with a nest of the source program. By the following optimization, I may not recognize loop information definitely. By the following optimization, it may not be correctly recognized loop information.
 - inline expansion for procedures
 - loop unrolling
 - loop blocking
 - loop interchange
 - loop fusion
 - loop striping
 - loop splitting
 - loop peeling
 - loop unswitching
 - change the function to a multi-operation function, use SIMD instructions
- If the information is fetched each routine or loop by specifying Nrt_tune_func or -Nrt_tune_loop, fetched information is increases, and execute time may increase.
- When I/O information on C/C++ is acquired specifying Nrt_tune_io, the function prototype should be declared to the service function in a correct type (prototype declaration). When it doesn't become a measurement object when not declared in a correct type, I/O information is not output. Please do an appropriate standard header in include to declare the function prototype in a correct type.

2.3 Runtime Environment Variables

This section describes the runtime environment variables using this function.

The English small letter and the English capital letter specified for the value of the environment variable at execution time are distinguished.

If an incorrect value is specified for the operand of a variable, then the runtime message concerning the environment variables for which operand values can be specified is output, and the environment variable is disabled or the operand value is disabled. For environment variables that do not require operands, the specified operand value is ignored and the environment variable takes effect.

Table 2.2 Environment variables that control the information output

Runtime environment variable		Explanation
Variable name	Operand	
FLIB_RTINFO_RATIO	(None)	Outputs Parallelization Information.
FLIB_RTINFO_COST	(None)	Outputs Cost Information.
FLIB_RTINFO_IO	F	Outputs Input-Output Information.
	C	The default operand is A.

Runtime environment variable		Explanation
Variable name	Operand	
	A	Outputs Hardware Monitor Information. The default operand is Statistics.
	-	
FLIB_RTINFO_PA	Statistics	
	Instructions_SIMD	
	Instructions_NOSIMD	
	Cache	
	MEM_access	
	Performance	
	TLB	
	-	

 **Note**

When either of Parallelization Information, Cost Information or Input-Output Information is output, Hardware Monitor Information cannot be output.

Table 2.3 Environment variables that control the output format

Runtime environment variable		Explanation
Variable name	Operand	
FLIB_RTINFO_NOALL	(None)	Outputs information only for the specified range (*1) rather than outputting information for the entire program.
FLIB_RTINFO_NOREGION	(None)	Outputs information only for the entire program rather than outputting information for the specified range (*1).
FLIB_RTINFO_NOFUNC	(None)	Even if the compiler option -Nrt_tune_func is effective, the output of information for each routine is suppressed.
FLIB_RTINFO_NOLOOP	(None)	Even if the compiler option -Nrt_tune_loop is effective, the output of information for each loop is suppressed.
FLIB_RTINFO_CSV	<i>Filename</i>	Outputs the fetched information to a CSV file.
	-	Operand is optional. Any file name can be specified as operand of the environment variable. If operand is omitted, flib_rtinfor.csv is assumed.
FLIB_RTINFO_PROCESS	txt	The information output in standard output file is output to the file unique in each process when the program is executed by the multi-process.
	csv	
	-	Operand is optional. File format can be specified as operand of the environment variable. If operand is omitted, txt is assumed.
FLIB_RTINFO_RANKID	<i>Integer value</i> from 0 to 2147483647	It is output only information corresponding to process(rank) ID specified for operand when the program is executed by the multi-process. (*2)
FLIB_RTINFO_RANKID_MAX	<i>Integer value</i> from 0 to 2147483647	It is output only information corresponding to process(rank) ID less than value specified for operand when the program is executed by the multi-process. (*2)

Runtime environment variable		Explanation
Variable name	Operand	
FLIB_RTINFO_RANKID_MIN	<i>Integer value</i> from 0 to 2147483647	It is output only information corresponding to process(rank) ID more than value specified for operand when the program is executed by the multi-process. (*2)
FLIB_RTINFO_COSTLIMIT	<i>Integer value</i> from 0 to 2147483647	When the cost information is output, information on the number of cases specified by the integral value is output in order with a high cost in the measurement section.
FLIB_RTINFO_THDCOST	(None)	When the cost information is output, information for each thread is output.
FLIB_RTINFO_EXEC_ORDER	(None)	When the Parallelization Cost or the Cost Information for each routine is output, information is output in order that the routine is executed.
FLIB_RTINFO_THRESHOLD	<i>Integer value</i> from 1 to 2147483647	For output of Input-Output Information in statement units (*2), information of the statements that elapsed time which exceeded the integer value (milliseconds) is output.
FLIB_RTINFO_PA_MODE	a	Specifies the Hardware Monitor Information collection mode (*3)
	u	
	s	
FLIB_RTINFO_FUNCMAX_PA	<i>Integer value</i> from 0 to 2147483647	When Hardware Monitor Information on each routine is output, the maximum measurement frequency to acquire information on the routine is specified.

*1) Refer to "3.2 Range Specification" for information on how to fetch information for a specified range.

*2) Refer to "7.2.2.3 Statement Information" for information on how to output Input-Output Information in statement units.

*3) Refer to "8.2.2 Switching Collection Mode" for information about collection modes.

2.4 Range of Obtainable Information

This section describes the range of information that can be fetched using this function.

Table 2.4 Range of obtainable information

Fetch Information	Range between start and end	Parallelized locations (Auto-parallel and OpenMP)	Each routine	Each loop
Parallelization Information	Yes	-	Yes	-
Cost Information	Yes	Yes (thread units)	Yes	Yes
Input-Output Information (Fortran)	Yes	-	-	-
Input-Output Information (C/C++)	Yes (*)	-	-	-
Hardware Monitor Information	Yes	Yes (thread units)	Yes	-

*) It is measured within the start of the program to the end.

2.4.1 Information Fetched for Each Routine

When information is fetched for each routine, the information below is fetched in the cases shown below.

- If a routine is called from within a routine :
 - Information for the called routine is not included in the calling source routine information.
- If the same routine is called from multiple locations :
 - The total values are used as the information for that routine.
- Only user-defined routines are output. Information concerning system calls and library functions is not fetched.

2.4.2 Information Fetched for Each Loop

When information is fetched for each loop, the information below is fetched in the cases shown below.

- If a routine is called from within a loop :
 - Information for the called routine is included in the calling source loop information.
- If a loop is called from within a loop :
 - Information for the called loop is included in the calling source loop information.

2.5 Output Information List

This section describes the information output by this function.

Table 2.5 Output information

Fetched information	Runtime environment variables		Output information
	Variable name	Operand	
Parallelization Information	FLIB_RTINFO_RATIO	(None)	<p>The following information related to Parallelization Information is output</p> <ul style="list-style-type: none"> - Parallelization ratio (percentage of the program operated in parallel) - For each process parallelized in the program, the percentage of the elapsed time for the entire program that is accounted for by the time that is the sum of the shortest processing times of all the separate processes. - For each process parallelized in the program, the percentage of the elapsed time for the entire program that is accounted for by the time that is the sum of the longest processing times of all the separate processes. - For each of the processes parallelized in the program, the percentage of the elapsed time for the entire program that is accounted for by the time that is the sum of the processing times executed by master thread of all the separate processes. <p>The following Parallelization Information for each routine is output</p> <ul style="list-style-type: none"> - Parallelization ratio (percentage of the program operated in parallel) - For each of the processes parallelized in the routine, the percentage of the elapsed time for the entire routine that is accounted for by the time that is the sum of the shortest processing times of all the separate processes. - For each of the processes parallelized in the routine, the percentage of the elapsed time for the entire routine that is

Fetched information	Runtime environment variables		Output information
	Variable name	Operand	
			<p>accounted for by the time that is the sum of the longest processing times of all the separate processes.</p> <ul style="list-style-type: none"> - For each of the processes parallelized in the routine, the percentage of the elapsed time for the entire routine that is accounted for by the time that is the sum of the processing times executed by master thread of all the separate processes. - Routine start line number - Routine end line number - Call frequency of routine - Routine name
Cost Information	FLIB_RTINFO_COST	(None)	<p>The following information related to Cost Information is output</p> <ul style="list-style-type: none"> - Elapsed time - User CPU time - System CPU time <p>The following Cost Information for each routine in serial is output</p> <ul style="list-style-type: none"> - Routine cost - Percentage of total cost accounted for by the routine cost - Routine cost per called once - Routine start line number - Routine end line number - Call frequency of routine - Routine name <p>The following Cost Information for each routine in parallel is output</p> <ul style="list-style-type: none"> - Routine cost - Percentage of total cost accounted for by the routine cost - Routine cost per called once - Routine start line number - Routine end line number - Call frequency of routine - Routine name <p>The following load balance information for each routine in parallel is output</p> <ul style="list-style-type: none"> - Cost of the thread which cost has a highest in routine - Cost of the thread which cost has a lowest in routine - Subtract lowest cost from highest cost - The ratio of subtract highest cost from lowest cost for highest cost - Routine start line number

Fetched information	Runtime environment variables		Output information
	Variable name	Operand	
			<ul style="list-style-type: none"> - Routine end line number - Call frequency of routine - Routine name <p>The following Cost Information for each thread in routine in parallel is output</p> <ul style="list-style-type: none"> - Routine cost - Percentage of total cost accounted for by the routine cost - Wait cost for synchronization between threads - The ratio of wait cost for synchronization between threads - Routine start line number - Routine end line number - Routine name <p>The following Cost Information for each loop in serial is output</p> <ul style="list-style-type: none"> - Loop cost - Percentage of total cost accounted for by the loop cost - Loop cost per called once - Loop start line number - Loop end line number - Call frequency of Loop - Loop nest level - Generated procedure name <p>The following Cost Information for each loop that in parallel region by OpenMP is output</p> <ul style="list-style-type: none"> - Loop cost - Percentage of total cost accounted for by the loop cost - Loop cost per called once - Loop start line number - Loop end line number - Call frequency of Loop - Loop nest level - Generated procedure name <p>The following Cost Information for each loop that in parallel region by OpenMP is output</p> <ul style="list-style-type: none"> - Cost of the thread which cost has a highest in loop - Cost of the thread which cost has a lowest in loop - Subtract lowest cost from highest cost - The ratio of subtract highest cost from lowest cost for highest cost

Fetched information	Runtime environment variables		Output information
	Variable name	Operand	
			<ul style="list-style-type: none"> - Loop start line number - Loop end line number - Call frequency of loop - Generated procedure name <p>The following Cost Information for each thread in loop that in parallel region by OpenMP is output</p> <ul style="list-style-type: none"> - Loop cost - Percentage of total cost accounted for by the loop cost - Wait cost for synchronization between threads - The ratio of wait cost for synchronization between threads <ul style="list-style-type: none"> - Loop start line number - Loop end line number - Generated procedure name <p>The following Cost Information for each parallel region is output</p> <ul style="list-style-type: none"> - Cost of parallel region - Percentage of total cost accounted for cost of parallel region - Cost of parallel region per called once - Parallel region start line number - Parallel region stop line number - Call frequency of parallel region - Generated procedure name <p>The following load balance information for each parallel region is output</p> <ul style="list-style-type: none"> - Cost of the parallel region which cost has a highest in routine - Cost of the parallel region which cost has a lowest in routine - Subtract lowest cost from highest cost - The ratio of subtract highest cost from lowest cost for highest cost - Routine start line number - Routine end line number - Call frequency of routine - Generated procedure name <p>The following Cost Information for each thread in parallel region is output</p> <ul style="list-style-type: none"> - Cost of parallel region - Percentage of total cost accounted for cost of parallel region - Wait cost for synchronization between threads - The ratio of wait cost for synchronization between threads

Fetched information	Runtime environment variables		Output information
	Variable name	Operand	
			<ul style="list-style-type: none"> - Parallel region start line number - Parallel region stop line number - Generated procedure name
Input-Output Information	Fortran		
	FLIB_RTINFO_IO	F or A	<p>The following information related to Fortran Input/Out Information is output</p> <ul style="list-style-type: none"> - Filename - File modify date - Unit number - Buffer size fetched by input-output statements - Access type (sequential, direct, or stream) - Format type (formatted, unformatted, or binary) - File size - Number of records - Execution time - Input-output item size - Statement called count - Statement type - line number - Input-output statement call source routine name - Information related to read (total size, total count, total time) - Information related to write (total size, total count, total time) - Information related to read (total count) - Execution count for I/O buffer parallel transfer - Total time and the number of times that depended on a completion demand of the asynchronous input - Total time and the number of times that depended on a completion demand of the asynchronous output
	C/C++		
	FLIB_RTINFO_IO	C or A	<p>The following information related to C/C++ Input/Out is output</p> <ul style="list-style-type: none"> - Filename - File modify date - File descriptor - File system - File size - Process number - Information related to open (Elapsed time, System time, Average elapsed time, Execution frequency)

Fetched information	Runtime environment variables		Output information
	Variable name	Operand	
			<ul style="list-style-type: none"> - Information related to read (Elapsed time, System time, Average elapsed time, Throughput, Transfer size, Execution frequency) - Information related to write (Elapsed time, System time, Average elapsed time, Throughput, Transfer size, Execution frequency) - Information related to fsync (Elapsed time, System time, Average elapsed time, Execution frequency) - Information related to close (Elapsed time, System time, Average elapsed time, Execution frequency)
Hardware Monitor Information	FLIB_RTINFO_PA	Statistics	<p>The following information related to Hardware Monitor Information is output</p> <ul style="list-style-type: none"> - Elapsed time - MFLOPS - MFLOPS peak performance ratio - MIPS - MIPS peak performance ratio - Floating point operation ratio
		Instructions_SIMD	<p>The following information related to Hardware Monitor Information is output</p> <ul style="list-style-type: none"> - Elapsed time - Number of instructions executed - MIPS - MIPS peak performance ratio - Load store instruction ratio (SIMD) - Floating point operation instruction ratio (SIMD) - Floating point sum operation instruction ratio (SIMD) - Fixed point operation instruction ratio (SIMD) - SIMD instruction ratio - SIMD Load store instruction ratio - Number of pre-fetch instructions - Number of indirect pre-fetch instructions
		Instructions_NOSIM D	<p>The following information related to Hardware Monitor Information is output</p> <ul style="list-style-type: none"> - Elapsed time - Number of instructions executed - MIPS - MIPS peak performance ratio - Load store instruction ratio (NOSIMD)

Fetched information	Runtime environment variables		Output information
	Variable name	Operand	
			<ul style="list-style-type: none"> - Floating point operation instruction ratio (NOSIMD) - Floating point sum operation instruction ratio (NOSIMD) - Fixed point operation instruction ratio (NOSIMD) - NOSIMD instruction ratio - NOSIMD Load store instruction ratio - Number of pre-fetch instructions - Number of indirect pre-fetch instructions
		Cache	<p>The following information related to Hardware Monitor Information is output</p> <ul style="list-style-type: none"> - Elapsed time - Number of instructions executed - Level 1 instruction cache mistake ratio - Level 1 data cache mistake ratio - Level 2 cache mistake ratio - Level 2 cache demand mistake ratio - Level 2 cache pre-fetch mistake ratio
		MEM_access	<p>The following information related to Hardware Monitor Information is output</p> <ul style="list-style-type: none"> - Elapsed time - Load store instruction ratio (SIMD instruction ratio) - Load store instruction ratio (NOSIMD instruction ratio) - Memory access throughput
		Performance	<p>The following information related to Hardware Monitor Information is output</p> <ul style="list-style-type: none"> - Elapsed time - 2-4 instruction commit - 1 instruction commit - Operation wait - Cache access wait - Memory access wait - Instruction fetch wait - Other waits
		TLB	<p>The following information related to Hardware Monitor Information is output</p> <ul style="list-style-type: none"> - Elapsed time - Number of instructions executed - Micro data TLB mistake ratio - Data main TLB mistake ratio

Chapter 3 Measurement Range

This chapter describes how to fetched information that a part of the program as a measurement range.

3.1 Entire Program

When the `-Nrt_tune` compile option is specified, this function is enabled and information can be fetched from the start of the program to the end.

Also, if the `-Nrt_tune_func` environment variables or `-Nrt_tune_loop` environment are specified, additional information concerning each routine, each loop can be fetched.

Refer to "[2.2 Compiler Options](#)" for further details.

3.2 Range Specification

The range specification is a function that adds Information from the start to the end of a specified range to information on the entire program. To specify a range, a service routine must be inserted at the measurement start position and the measurement end position in the source program. If external name of the service routine processed by specifying the compiler option `-mldefault=cdecl` or `-ml=cdecl` or using `$pragma c` specifier, information cannot be correctly fetched.

Also, if the `-Nrt_tune_func` environment variables or `-Nrt_tune_loop` environment are specified, additional information concerning each routine, each loop can be fetched.

However, if specified range crosses routine or parallelized location, information concerning each routine, each loop and parallelized location may be not correctly fetched.

Refer to "[2.2 Compiler Options](#)" for further details.

If the environment variable `FLIB_RTINFO_NOALL` is specified, information is fetched from only the specified segment rather than from the entire program. This can reduce measurement costs because the segment being measured is restricted. If there is no specified range, it is output only header information (Refer to "[4.1 Output format](#)") of runtime information.

Table 3.1 Format of `FLIB_RTINFO_NOALL` environment variable

variable name	Operand
<code>FLIB_RTINFO_NOALL</code>	(None)

If the environment variable `FLIB_RTINFO_NOREGION` is specified, information is fetched from only the entire program rather than from the specified segment. This can reduce measurement costs because the segment being measured is restricted.

Table 3.2 Format of `FLIB_RTINFO_NOREGION` environment variable

variable name	Operand
<code>FLIB_RTINFO_NOREGION</code>	(None)

When both environment variable `FLIB_RTINFO_NOALL` and environment variable `FLIB_RTINFO_NOREGION` are specified, it is output only header information (Refer to "[4.1 Output format](#)") of runtime information.

How to use the service routine for Fortran, C, and C++ are described below.

3.2.1 Fortran

The format of the service subroutines used to specify a range is shown below.

Table 3.3 Format of service subroutines used to specify a range

Service subroutine name	Explanation	Argument	Definition
<code>START_COLLECTION</code>	measurement start	Default character type scalar	<code>fjcoll_lib / fjcoll_lib.h</code>
<code>STOP_COLLECTION</code>	measurement stop	Default character type scalar	<code>fjcoll_lib / fjcoll_lib.h</code>

- If argument is not default character type scalar, the execution result is ensured.
- A space in the argument is read as being one character.

Example

Example 1: "AB<space>C" and "ABC" are different argument values


Example 2: "ABC<space>" and "ABC" are different argument values

- The measurement range is from the START_COLLECTION to the STOP_COLLECTION service subroutine that specifies the same argument value.
- If service subroutines with the same argument value specified are called from multiple locations, the measured value is the total value for all the ranges.
- If the service subroutine is called in parallel region, the information may be not correctly fetched.

The following specifications can be made for a service subroutine

1. Range from the START_COLLECTION to the STOP_COLLECTION that has the same argument value specified


```
call START_COLLECTION("region_1")
call STOP_COLLECTION("region_1")
```



2. Nested range

Range specifications that have different arguments can be nested.


```
call START_COLLECTION("region_1")
call START_COLLECTION("region_2")
call STOP_COLLECTION("region_2")
call STOP_COLLECTION("region_1")
```



3. Overlapped range

Range specifications that have different arguments can be overlapped.

```
call START_COLLECTION("region_1")
call START_COLLECTION("region_2")
call STOP_COLLECTION("region_1")
call STOP_COLLECTION("region_2")
```



The figure below shows example of service subroutine usage.

Example

Example of service subroutine usage

```
USE FJCOLL_LIB

INTEGER I, J
REAL, DIMENSION(100,100) :: R
CALL START_COLLECTION("MAIN_LOOP")
```

```

DO I = 1, 100
  DO J = 1, 100
    R( I, J ) = I + J
  END DO
END DO
CALL STOP_COLLECTION( "MAIN_LOOP" )
END

```

3.2.1.1 Error processing

An error occurs in the following cases:

- After START_COLLECTION, if a START_COLLECTION with the same argument value specified is executed, but only if there is no STOP_COLLECTION corresponding to the START_COLLECTION that was executed first

At the point when the subsequent START_COLLECTION is executed, runtime message RTINF0201 is output and the subsequent START_COLLECTION is disabled.

Refer to "Chapter 9 Message" for runtime message.

- If there is no START_COLLECTION corresponding to STOP_COLLECTION

At the point when the STOP_COLLECTION is executed, runtime message RTINF0202 is output.

Refer to "Chapter 9 Message" for runtime message.

- If there is no STOP_COLLECTION corresponding to START_COLLECTION

When the program ends, runtime message RTINF0203 is output.

Refer to "Chapter 9 Message" for runtime message.

3.2.2 C/C++

The format of the service functions used to specify a range is shown below.

Table 3.4 Format of service functions used to specify a range

Service function name	Explanation	Argument	Definition
void start_collection	measurement start	const char * type	fjcoll.h
void stop_collection	measurement stop	const char * type	fjcoll.h

- A space in the argument is read as being one character.

Example

Example 1: "AB<space>C" and "ABC" are different argument values

Example 2: "ABC<space>" and "ABC" are different argument values

- The measurement range is from the START_COLLECTION to the STOP_COLLECTION function that specifies the same argument value.
- If functions with the same argument value specified are called from multiple locations, the measured value is the total value for all the ranges.
- If the service function is called in parallel region, the information may be not correctly fetched.

The following specifications can be made for the service function used to specify a range:

1. Range from the start_collection to the stop_collection that has the same argument value specified

```

start_collection("region_1");

stop_collection("region_1");

```



2. Nested range

Range specifications that have different arguments can be nested.

```
start_collection("region_1");  
  
start_collection("region_2");  
  
stop_collection("region_2");  
  
stop_collection("region_1");
```



3. Overlapped range

Range specifications that have different arguments can be overlapped.

```
start_collection("region_1");  
  
start_collection("region_2");  
  
stop_collection("region_1");  
  
stop_collection("region_2");
```



The figure below shows example of service function usage.



Example

Example of service function usage

```
#include <stdio.h>  
#include <fjcoll.h>  
  
int main(void) {  
    int i,j;  
    float arr[100][100];  
    start_collection("main_loop");  
    for (i=0; i<100; i++) {  
        for (j=0; j<100; j++) {  
            arr[i][j] = i+j;  
        }  
    }  
    stop_collection("main_loop");  
}
```

3.2.2.1 Error processing

An error occurs in the following cases:

- After `start_collection`, if a `start_collection` with the same argument value specified is executed, but only if there is no `stop_collection` corresponding to the `start_collection` that was executed first

At the point when the subsequent `start_collection` is executed, runtime message RTINF0201 is output and the subsequent `start_collection` is disabled.

Refer to "[Chapter 9 Message](#)" for runtime message.

- If there is no `start_collection` corresponding to `stop_collection`

At the point when the `stop_collection` is executed, runtime message RTINF0202 is output.

Refer to "[Chapter 9 Message](#)" for runtime message.

- If there is no stop_collection corresponding to start_collection

When the program ends, runtime message RTINF0203 is output.
Refer to "[Chapter 9 Message](#)" for runtime message.

Chapter 4 Output

This chapter describes output of this function.

4.1 Output format

The fetched information is output file in the formats below.

In addition to the fetched information, header information shown the fetched information name is output

Output information includes header information shows the name of information other than Runtime Information.

Even if there is no information fetched, environment variable FLIB_RTINFO_NOALL is specified without specifying service routine, header information is output.

```
++=====++  
||           EXECUTION PERFORMANCE INFORMATION           ||  
++=====++
```

Program

Environment variable FLIB_RTINFO_RATIO is specified

```
++=====++  
||           PARALLELIZATION INFORMATION           ||  
++=====++
```

Parallelization Information is output
:

Environment variable FLIB_RTINFO_COST is specified

```
++=====++  
||           COST INFORMATION           ||  
++=====++
```

Cost Information is output
:

Environment variable FLIB_RTINFO_IO is specified

```
++=====++  
||           I/O INFORMATION           ||  
++=====++
```

```
*****  
Fortran I/O INFORMATION  
*****
```

I/O Information on Fortran is output
:

```
*****
C/C++ I/O INFORMATION
*****
```

I/O Information on C/C++ is output (If the compiler option -Nrt_tune_io is specified)
:

Environment variable FLIB_RTINFO_PA is specified

```
++=====++
||                HARDWARE MONITOR INFORMATION                ||
++=====++
```

Hardware Monitor Information is output
:

region_1 ##### (If there is a range specification named "region_1")

Environment variable FLIB_RTINFO_RATIO is specified

```
++=====++
||                PARALLELIZATION INFORMATION                ||
++=====++
```

Parallelization Information is output
:

Environment variable FLIB_RTINFO_COST is specified

```
++=====++
||                COST INFORMATION                ||
++=====++
```

Cost Information is output
:

Environment variable FLIB_RTINFO_IO is specified

```
++=====++
||                I/O INFORMATION                ||
++=====++
```

```
*****
Fortran I/O INFORMATION
*****
```

I/O Information on Fortran is output
:

Environment variable FLIB_RTINFO_PA is specified

```

++=====++
||                HARDWARE MONITOR INFORMATION                ||
++=====++

```

```

Hardware Monitor Information is output
:

```

The above information can be output by the text format or CSV format.

4.1.1 Text format

Output runtime information in the shown format that is "4.1 Output format"

Refer to the following for details of the output form of information.

Parallelization Information	"5.3 Output Example of Parallelization Information"
Cost Information	"6.3 Output Example of Cost Information"
Input-Output Information(Fortran)	"7.2.3 Output Example of Fortran Input-Output Information"
Input-Output Information(C/C++)	"7.3.3 Output Example of C/C++ Input-Output Information"
Hardware Monitor Information	"8.3 Calculation Expressions for Hardware Monitor Information"

4.1.2 CSV format

Output runtime information in the shown format that is "4.1 Output format"

When runtime information is output by CSV, first column to fourth column of the data line follows the following compositions. It is depending on output information that the fifth and subsequent column data.

When there is no output information, "-" is output.

Table 4.1 Composition of CSV format

Column position	Output information
first	Identifier that shows type
second	Process ID
third	Thread ID
fourth	Range name (When Information on the entire program is output, "__PROGRAM__" is output.)
fifth and subsequent	(It is depending on output information.)

About Identifier that shows type are shown below.

Table 4.2 Identifier that shows type

Identifier	Output information
[RATIO]	Parallelization Information (information for the entire program or specified range)
[RATIO_ROUTINE]	Parallelization Information (information for each routine)
[COST]	Cost Information (information for the entire program or specified range)

Identifier	Output information
[COST_ROUTINE_SERIAL]	Cost Information (information for each routine that operates in serial processing)
[COST_ROUTINE_PRL]	Cost Information (information for each routine that operates in multi processing)
[COST_LOOP_SERIAL]	Cost Information (information for each loop that operates in serial processing)
[COST_LOOP_PRL]	Cost Information (information for each loop that in parallel region by OpenMP)
[COST_PRL]	Cost Information (information for parallel region)
[IO_UNIT]	Fortran Input-Output Information (information for units and system calls)
[IO]	Fortran Input-Output Information (information for statement units)
[IO_C_SYS]	C/C++ Input-Output Information
[PA]	Hardware Monitor Information (information for the entire program or specified range)
[PA_ROUTINE]	Hardware Monitor Information (information for each routine)

The method of specifying the text format or CSV format is different at a case of serial Programs or Multi-Thread Programs and MPI Programs or a Hybrid Programs.

A Hybrid program is MPI and a Multi-Thread program.

Each specification method is shown as follows.

4.2 Serial Programs or Multi-Thread Programs

Output runtime information in the shown format that is "4.1 Output format" to standard output file.

If the FLIB_RTINFO_CSV environment is specified, the fetched information can be output in CSV format. If a filename is specified in the operands, the fetched information is output to the specified file. If the specified file already exists, the information overwrites the existing file. If the file fails to open, runtime message RTINF0101 is output and the information is output to the standard output. Refer to "Chapter 9 Message" for runtime message.

Note that this environment variable is disabled for MPI Programs or a Hybrid Programs.

Table 4.3 Format of FLIB_RTINFO_CSV environment variable

variable name	Operand
FLIB_RTINFO_CSV	<i>Filename</i>
	-

4.3 MPI Programs or Hybrid Programs

Output runtime information in the shown format that is "4.1 Output format" to standard output file.

If the FLIB_RTINFO_PROCESS environment variable is specified, the fetched information can be output to file each process.

If txt is specified for operand, information is output to the file of the following names.

If csv is specified for operand, information is output to the file of the following names by CSV format.

If the operand value is omitting or incorrect, it is assumed that txt is specified as the value.

Static Process

flib_rtinfo_id_rank (txt is effective as the operand)

flib_rtinfo_id_rank.csv (csv is effective as the operand)

Dynamic Process

flib_rtinfo_id_rank_spawn (txt is effective as the operand)

flib_rtinfo_id_rank_spawn.csv (csv is effective as the operand)

id: Identification number to prevent output file name from overlapping with other jobs

rank: process(rank) ID

spawn: spawn number

The above-mentioned file is generated to each process. If the specified file already exists, the information overwrites the existing file. If the file fails to open, runtime message RTINF0102 is output and the environment variable is disabled. Refer to "[Chapter 9 Message](#)" for runtime message.

Note that this environment variable is disabled for serial Programs or Multi-Thread Programs.

Table 4.4 Format of FLIB_RTINFO_PROCESS environment variable

Variable name	Operand
FLIB_RTINFO_PROCESS	txt
	csv
	-

The information to each process is output for all processes, moreover the environment variable FLIB_RTINFO_RANKID, FLIB_RTINFO_RANKID_MAX or FLIB_RTINFO_RANKID_MIN can restrict information of process.

If environmental variable FLIB_RTINFO_RANKID is specified, it is output only information corresponding to process (rank) ID specified for operand.

If environmental variable FLIB_RTINFO_RANKID_MAX is specified, it is output only information corresponding to process (rank) ID less than value specified for operand.

If environmental variable FLIB_RTINFO_RANKID_MIN is specified, it is output only information corresponding to process (rank) ID more than value specified for operand.

When FLIB_RTINFO_RANKID, FLIB_RTINFO_RANKID_MAX or FLIB_RTINFO_RANKID_MIN is specified at the same time, only FLIB_RTINFO_RANKID becomes effective.

When FLIB_RTINFO_RANKID_MIN and FLIB_RTINFO_RANKID_MAX are specified at the same time and the value specified for FLIB_RTINFO_RANKID_MIN is larger than the value that specified for FLIB_RTINFO_RANKID_MAX, only FLIB_RTINFO_RANKID_MAX becomes effective.

The specification of the environment variable is invalid when the specified value is disabled, and information on all processes is output.

Table 4.5 Format of environment variable specified process fetched information.

Variable name	Operand
FLIB_RTINFO_RANKID	Integer value (0 to 2147483647)
FLIB_RTINFO_RANKID_MAX	Integer value (0 to 2147483647)
FLIB_RTINFO_RANKID_MIN	Integer value (0 to 2147483647)

Chapter 5 Parallelization Information

This chapter describes the Parallelization Information.

The parallel in Parallelization Information means thread parallelization.

5.1 Overview of Parallelization Information

The following information is output, as "Parallelization Information" by specifying the environment variable FLIB_RTINFO_RATIO.

(1) Parallelization ratio (percentage of the program operated in parallel)

Total parallel processing elapsed time for all threads / (Total parallel processing elapsed time for all threads + master thread sequential processing elapsed time) is output.

(2) For each of the processes parallelized in the program, the percentage of the elapsed time for the entire program that is accounted for by the time that is the sum of the shortest processing times of all the separate processes.

(3) For each of the processes parallelized in the program, the percentage of the elapsed time for the entire program that is accounted for by the time that is the sum of the longest processing times of all the separate processes.

(4) For each of the processes parallelized in the program, the percentage of the elapsed time for the entire program that is accounted for by the time that is the sum of the processing times executed by master thread of all the separate processes.

The information in (1) shows the ratio of serial processing to parallel processing within the measured range.

From the information in (2), it is possible to determine whether performance could be improved by increasing the number of parallel processes.



Example

80% Performance improvement can be expected from increasing the number of parallel processes.

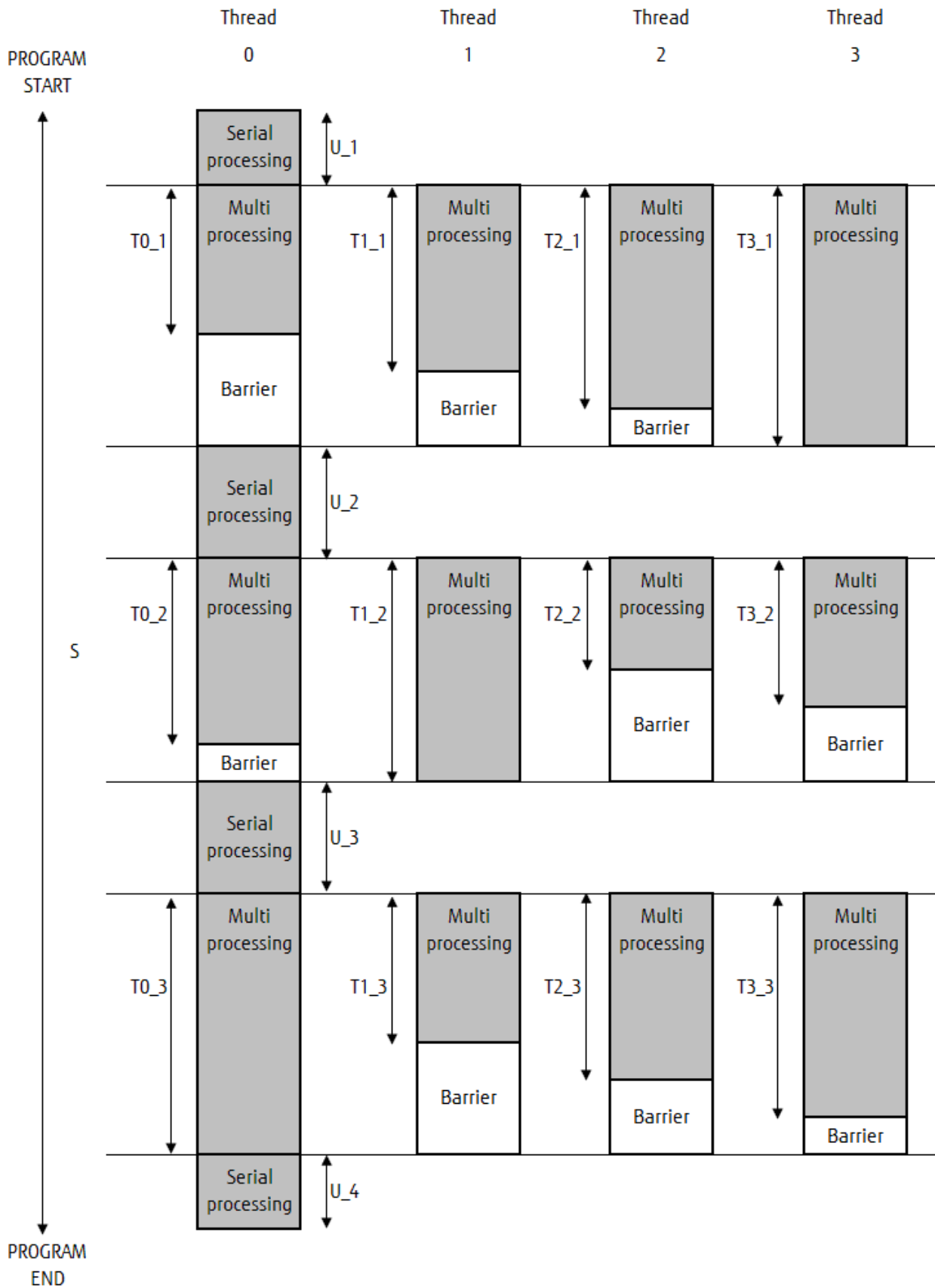
2% Performance improvement cannot be expected from increasing the number of parallel processes.

The difference between (2) and (3) indicates the load balance of the parallelized regions within the measured range. The parallel balance can be considered good if this difference is small.

5.2 Details of Parallelization Information

The figure below shows an overview of parallel execution processing. And the explanation below is the calculation method for (1)~(4) values in "5.1 Overview of Parallelization Information".

Figure 5.1 Overview of parallel execution processing



*) The bar graph shows the elapsed time for each thread.

Calculation method

(1) Parallelization ratio

The ratio can be obtained by dividing the parallel processing time by the shaded time (elapsed time except for barriers). The unit is a percentage (%).

The sum of T0_1~T3_3 is sumT, and the sum of U_1~U_3 is sumU.

$$\text{sumT} / (\text{sumT} + \text{sumU}) * 100 (\%)$$

(2) For each of the processes parallelized in the program, the percentage of the elapsed time for the entire program that is accounted for by the time that is the sum of the shortest processing times of all the separate processes

Obtain the lowest value for each of T0_1~T3_1, T0_2~T3_2, and T0_3~T3_3. Divide the total of these values by the elapsed time for the entire program S. The unit is a percentage (%).

The calculation expression for the parallel execution processing shown in the figure above is as follows:

$$(\text{T0}_1 + \text{T3}_2 + \text{T2}_3) / \text{S} * 100 (\%)$$

(3) For each of the processes parallelized in the program, the percentage of the elapsed time for the entire program that is accounted for by the time that is the sum of the longest processing times of all the separate processes

Obtain the highest value for each of T0_1~T3_1, T0_2~T3_2, and T0_3~T3_3. Divide the total of these values by the elapsed time for the entire program S. The unit is a percentage (%).

The calculation expression for the parallel execution processing shown in the figure above is as follows:

$$(\text{T3}_1 + \text{T1}_2 + \text{T0}_3) / \text{S} * 100 (\%)$$

(4) For each of the processes parallelized in the program, the percentage of the elapsed time for the entire program that is accounted for by the time that is the sum of the processing times executed by master thread of all the separate processes

Obtain the highest value for each of T0_1~T3_1, T0_2~T3_2, and T0_3~T3_3. Divide the total of these values by the elapsed time for the entire program S. The unit is a percentage (%).

The calculation expression for the parallel execution processing shown in the figure above is as follows:

$$(\text{T0}_1 + \text{T0}_2 + \text{T0}_3) / \text{S} * 100 (\%)$$

5.2.1 Information for Each Routine

When the -Nrt_tune_func compile option is specified, the Parallelization Information is output for separate routines.

Refer to "2.2 Compiler Options" for information concerning the compile options.

Parallelization ratios for each routine are output in order with large total value of ratio by Automatic parallelization and OpenMP.

If the FLIB_RTINFO_EXEC_ORDER environment variable is specified, the parallelization ratios are output in the execution order of the routines.

Table 5.1 Format of FLIB_RTINFO_EXEC_ORDER environment variable

Environment variable name	Operand
FLIB_RTINFO_EXEC_ORDER	(None)

If the FLIB_RTINFO_EXEC_ORDER environment variable is specified, the parallelization ratios are output in the execution order of the routines.

Even if compile option Nrt_tune_func is specified, information for each routine is not output when environment variable FLIB_RTINFO_NOFUNC is specified.

Table 5.2 Format of FLIB_RTINFO_NOFUNC environment variable

Environment variable name	Operand
FLIB_RTINFO_NOFUNC	(None)

5.3 Output Example of Parallelization Information

Output example of Parallelization Information is shown below.

Figure 5.2 Output example of Parallelization Information

```

=====
||                                     PARALLELIZATION INFORMATION                                     ||
=====

+ (1)-----+ (2)-----+ (3)-----+ (4)-----+
| All      | Lowest   | Highest  | Master   |
+ (5)-----+ (6)-----+
| Auto (%) | OpenMP (%) | Auto (%) | OpenMP (%) |
+-----+-----+
| 89.41 | 2.92 | 82.11 | 2.65 | 83.08 | 2.75 | 83.08 | 2.75 |
+-----+-----+

Routine (7)
+-----+-----+-----+-----+ (8)---+ (9)---+ (10)---+ (11)-----+
| All      | Lowest   | Highest  | Master   | Start | End  | Count | Routine name |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Auto (%) | OpenMP (%) | Auto (%) | OpenMP (%) | Auto (%) | OpenMP (%) | Auto (%) | OpenMP (%) |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 89.18 | 3.46 | 82.60 | 3.16 | 83.06 | 3.27 | 83.06 | 3.27 | 40 | 49 | 10 | sub1 |
| 91.64 | 0.00 | 82.44 | 0.00 | 84.74 | 0.00 | 84.74 | 0.00 | 51 | 56 | 2000 | sub2 |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

- (1) Percentage of the program operated in parallel
- (2) Percentage of the thread with the shortest elapsed time that is operated in parallel
- (3) Percentage of the thread with the longest elapsed time that is operated in parallel
- (4) Percentage of the master thread that is operated in parallel
- (5) Information for region into automatic parallelization
- (6) Information for region into OpenMP
- (7) Information for routine (-Nrt_tune_func is specified)
- (8) Start line number
- (9) End line number
- (10) Call frequency
- (11) Routine name
- (12) Information for routine

Remarks:

- If neither -Kparallel nor -Kvisimpact are specified, a hyphen (-) is displayed in the [Auto] column.
- If -Kopenmp is not specified, a hyphen (-) is displayed in the [OpenMP] column.
- For sequential programs, a hyphen (-) is displayed in the [Auto] and [OpenMP] columns.
- If there are no parallelized locations even though the option is specified, 0% is displayed in the [Auto] and [OpenMP] columns.

Figure 5.3 Output example of Parallelization Information (environmental variable FLIB_RTINFO_CSV is specified)

```

=====
PARALLELIZATION INFORMATION
=====
Type,Process ID,Thread ID,Range,All,,Lowest,,Highest,,Master,
..., Auto (%), OpenMP (%), Auto (%), OpenMP (%), Auto (%), OpenMP (%), Auto (%), OpenMP (%)
[RATIO], 0, -, "_PROGRAM_", 89.55, 2.90, 82.55, 2.68, 83.33, 2.70, 83.06, 2.70

Routine
Type,Process ID,Thread ID,Range,All,,Lowest,,Highest,,Master,, Start,End,Count,Routine name
..., Auto (%), OpenMP (%), Auto (%), OpenMP (%), Auto (%), OpenMP (%), Auto (%), OpenMP (%),...
[RATIO_ROUTINE], 0, -, "_PROGRAM_", 89.40, 3.44, 83.10, 3.19, 83.44, 3.22, 83.10, 3.22, 40, 49, 10, "sub1"
[RATIO_ROUTINE], 0, -, "_PROGRAM_", 91.46, 0.00, 81.77, 0.00, 84.48, 0.00, 84.48, 0.00, 51, 56, 2000, "sub2"

```

Chapter 6 Cost Information

This chapter describes the Cost Information.

6.1 Overview of Cost Information

Cost information for the ranges shown below can be fetched by specifying the FLIB_RTINFO_COST environment variable.

- Range from start to end
- Each routine
- Each loop

6.2 Details of Cost Information

6.2.1 Range from Start to End

The following cost information for the program or for the segment between the start and end of the specified range is output

- Elapsed time
- User CPU time
- System CPU time
- Parallelized regions

6.2.2 Parallelized regions

The following information is output as cost information on parallel regions.

- Information for each region parallelized
- Load balance information for each region parallelized
- Information for each thread in parallelized regions

Information for each thread in parallelized regions is output by specifying environment variable FLIB_RTINFO_THDCOST.

Table 6.1 Format of FLIB_RTINFO_THDCOST environment variable

Environment variable name	Operand
FLIB_RTINFO_THDCOST	(None)

As for Cost Information for each region parallelized, ten high ranks information is output in order with a high cost of the generation procedure.

Generation procedure costs are as the OpenMP instruction statement and automatically parallelized loop costs.

Moreover, the number of cases of output information can be changed by specifying environment variable FLIB_RTINFO_COSTLIMIT.

The integral value from 0 to 2147483647 is specified for an operand, and the number of cases of output information is specified. When 0 is specified, all information is output. When the value of the operand is unjustified, it is considered that 10 is specified.

Table 6.2 Format of FLIB_RTINFO_COSTLIMIT environment variable

Environment variable name	Operand
FLIB_RTINFO_COSTLIMIT	Integer value (0 to 2147483647)

The cost information for parallelized regions includes the overhead of thread parallelization (the cost of switching sequential regions and parallel regions). It is possible to work out how the cost will vary if the number of threads is changed from this information.

Load balance information for each region parallelized and information for each threads parallelized regions does not include parallelization overheads.

The table below shows the procedure names that correspond to generated procedures.

Table 6.3 Generated procedure names in thread parallel programs

Language type	Types of generated procedure names	Generated procedure name
Fortran	OpenMP	(Generated procedure name)._OMP_(numeric)_
	Automatic parallelization	(Generated procedure name)._PRL_(numeric)_
C/C++	OpenMP	(Generated procedure name)._OMP_(numeric)
	Automatic parallelization	(Generated procedure name)._PRL_(numeric)

If a routine or loop call is in a generated procedure, the cost for the called routine or loop is included in the generated procedure cost.

6.2.3 Each Routine

When the `-Nrt_tune_func` compile option is specified, Cost Information for each routine is output.

Refer to "2.2 Compiler Options" for further details.

The following information is output, as Cost Information for each routine.

- Information for each routine operates in serial processing.
- Information for each routine operates in multi processing.
- Load balance information for each routine operates in multi processing.
- Information for each thread in routine operates in multi processing.

Information for each thread in routine operates in multi processing is output by specifying environment variable `FLIB_RTINFO_THDCOST`.

Table 6.4 Format of `FLIB_RTINFO_THDCOST` environment variable

Environment variable name	Operand
<code>FLIB_RTINFO_THDCOST</code>	(None)

As for Cost Information for each routine, ten high ranks information is output in order with a high cost.

Moreover, the number of cases of output information can be changed by specifying environment variable `FLIB_RTINFO_COSTLIMIT`.

The integral value from 0 to 2147483647 is specified for an operand, and the number of cases of output information is specified. When 0 is specified, all information is output. When the value of the operand is unjustified, it is considered that 10 is specified.

Table 6.5 Format of `FLIB_RTINFO_COSTLIMIT` environment variable

Environment variable name	Operand
<code>FLIB_RTINFO_COSTLIMIT</code>	Integer value (0 to 2147483647)

If the `FLIB_RTINFO_EXEC_ORDER` environment variable is specified, Cost Information is output in the execution order of the routines. Then, Cost Information for all routine is output. Environment variable `FLIB_RTINFO_COSTLIMIT` specification doesn't influence.

Table 6.6 Format of `FLIB_RTINFO_EXEC_ORDER` environment variable

Environment variable name	Operand
<code>FLIB_RTINFO_EXEC_ORDER</code>	(None)

Even if compile option `Nrt_tune_func` is specified, information for each routine is not output when environment variable `FLIB_RTINFO_NOFUNC` is specified.

Table 6.7 Format of FLIB_RTINFO_NOFUNC environment variable

Environment variable name	Operand
FLIB_RTINFO_NOFUNC	(None)

The cost in the part where operates in serial processing in routine is output as information for each routine where operates in serial processing. The cost in the part where operates in multi processing in routine is output as information for each routine where operates in multi processing.

6.2.4 Each Loop

When the -Nrt_tune_loop compile option is specified, Cost Information for each loop is output.

Refer to "2.2 Compiler Options" for further details.

The following information is output, as Cost Information for each loop.

- Information for each loop operates in serial processing.
- Information for each loop operates in parallel region by OpenMP.
- Load balance information for each loop operates in parallel region by OpenMP.
- Information for each thread in loop operates in parallel region by OpenMP.

Information for each thread in loop operates in parallel region by OpenMP is output by specifying environment variable FLIB_RTINFO_THDCOST.

Table 6.8 Format of FLIB_RTINFO_THDCOST environment variable

Environment variable name	Operand
FLIB_RTINFO_THDCOST	(None)

As for Cost Information for each loop, ten high ranks information is output in order with a high cost.

Moreover, the number of cases of output information can be changed by specifying environment variable FLIB_RTINFO_COSTLIMIT.

The integral value from 0 to 2147483647 is specified for an operand, and the number of cases of output information is specified. When 0 is specified, all information is output. When the value of the operand is unjustified, it is considered that 10 is specified.

Table 6.9 Format of FLIB_RTINFO_COSTLIMIT environment variable

Environment variable name	Operand
FLIB_RTINFO_COSTLIMIT	Integer value (0 to 2147483647)

Even if compile option Nrt_tune_loop is specified, information for each loop is not output when environment variable FLIB_RTINFO_NOLOOP is specified.

Table 6.10 Format of FLIB_RTINFO_NOLOOP environment variable

Environment variable name	Operand
FLIB_RTINFO_NOLOOP	(None)

The cost of the serial loop is output as information for serial loop.

If the loop exists in parallel region by OpenMP, it is output as each loop in parallel region by OpenMP.

6.3 Output Example of Cost Information

Output example of Cost Information is shown below.

Figure 6.1 Output example of Cost Information

```

=====
|| COST INFORMATION ||
=====

+-----+-----+-----+
| (1) Elapsed | (2) User | (3) System |
| (sec) | (sec) | (sec) |
+-----+-----+-----+
| 2.4776 | 4.9400 | 0.0000 |
+-----+-----+-----+

Serial Routine (4)
+-----+-----+-----+-----+-----+-----+-----+
| (5) Cost(sec) | (6) % | (7) Once(sec) | (8) Start | (9) End | (10) Count | (11) Routine name |
+-----+-----+-----+-----+-----+-----+-----+
| 0.2491 | 82.78 | 0.0249 | 40 | 49 | 10 | sub1 |
| 0.0518 | 17.22 | 0.0000 | 51 | 56 | 2000 | sub2 |
+-----+-----+-----+-----+-----+-----+-----+
| 0.3010 | 100.00 | - | - | - | - | - |
+-----+-----+-----+-----+-----+-----+-----+

Parallel Routine (14)
+-----+-----+-----+-----+-----+-----+-----+
| Cost(sec) | % | Once(sec) | Start | End | Count | Routine name |
+-----+-----+-----+-----+-----+-----+-----+
| 1.6602 | 83.90 | 0.1660 | 40 | 49 | 10 | sub1 |
| 0.3185 | 16.10 | 0.0002 | 51 | 56 | 2000 | sub2 |
+-----+-----+-----+-----+-----+-----+-----+
| 1.9786 | 100.00 | - | - | - | - | - |
+-----+-----+-----+-----+-----+-----+-----+

Load Balance (15)
+-----+-----+-----+-----+-----+-----+-----+
| (16) Min(sec) | (17) Max(sec) | (18) Barrier | (19) % | Start | End | Count | Routine name |
| | | (sec) | | | | | |
+-----+-----+-----+-----+-----+-----+-----+
| 1.6517 | 1.6602 | 0.0084 | 0.51 | 40 | 49 | 10 | sub1 |
| 0.3160 | 0.3185 | 0.0025 | 0.79 | 51 | 56 | 2000 | sub2 |
+-----+-----+-----+-----+-----+-----+-----+
| 1.9677 | 1.9786 | 0.0110 | 0.55 | - | - | - | - |
+-----+-----+-----+-----+-----+-----+-----+

Thread 0 (20)
+-----+-----+-----+-----+-----+-----+-----+
| Cost | % | (21) Barrier | (22) % | Start | End | Routine name |
| (sec) | | (sec) | | | | |
+-----+-----+-----+-----+-----+-----+-----+
| 1.6517 | 83.94 | 0.0084 | 0.51 | 40 | 49 | sub1 |
| 0.3160 | 16.06 | 0.0025 | 0.79 | 51 | 56 | sub2 |
+-----+-----+-----+-----+-----+-----+-----+
| 1.9677 | 100.00 | 0.0110 | 0.55 | - | - | - |
+-----+-----+-----+-----+-----+-----+-----+

Thread 1
+-----+-----+-----+-----+-----+-----+-----+
| Cost | % | Barrier | % | Start | End | Routine name |
| (sec) | | (sec) | | | | |
+-----+-----+-----+-----+-----+-----+-----+
| 1.6602 | 83.90 | 0.0000 | 0.00 | 40 | 49 | sub1 |
| 0.3185 | 16.10 | 0.0000 | 0.00 | 51 | 56 | sub2 |
+-----+-----+-----+-----+-----+-----+-----+
| 1.9786 | 100.00 | 0.0000 | 0.00 | - | - | - |
+-----+-----+-----+-----+-----+-----+-----+

Serial Loop (23)
+-----+-----+-----+-----+-----+-----+-----+
| Cost(sec) | % | Once(sec) | Start | End | Count | (24) nest | Routine name |
+-----+-----+-----+-----+-----+-----+-----+
| 2.0980 | 43.21 | 2.0980 | 19 | 29 | 1 | 1 | sample1 |
| 1.8363 | 37.82 | 0.1836 | 41 | 42 | 10 | 2 | (sample1.sub1_) |
+-----+-----+-----+-----+-----+-----+-----+
| 4.8550 | 100.00 | - | - | - | - | - | - |
+-----+-----+-----+-----+-----+-----+-----+

```

Parallel Loop (27)

Cost(sec)	%	Once(sec)	Start	End	Count	nest	Routine name
0.0728	47.47	0.0073	45	47	10	2	(sample1.sub1._OMP_3_)
0.0669	43.63	0.0000	46	46	10000	3	(sample1.sub1._OMP_3_)
0.1534	100.00	-	-	-	-	-	-

Load Balance (28)

Min(sec)	Max(sec)	Barrier (sec)	%	Start	End	Count	Routine name
0.0728	0.0728	0.0000	0.04	45	47	10	(sample1.sub1._OMP_3_)
0.0667	0.0669	0.0002	0.28	46	46	10000	(sample1.sub1._OMP_3_)
0.1529	0.1534	0.0005	0.31	-	-	-	-

Thread 0 (29)

Cost (sec)	%	Barrier (sec)	%	Start	End	Routine name
0.0728	47.54	0.0000	0.04	45	47	(sample1.sub1._OMP_3_)
0.0669	43.71	0.0000	0.00	46	46	(sample1.sub1._OMP_3_)
0.1531	100.00	0.0003	0.18	-	-	-

Thread 1

Cost (sec)	%	Barrier (sec)	%	Start	End	Routine name
0.0728	47.53	0.0000	0.00	45	47	(sample1.sub1._OMP_3_)
0.0667	43.56	0.0002	0.28	46	46	(sample1.sub1._OMP_3_)
0.1532	100.00	0.0002	0.12	-	-	-

Parallel Region (30)

Cost(sec)	%	Once(sec)	Start	End	Count	Routine name
1.7534	74.41	0.0000	42	43	100000	sample1.sub1._PRL_1_ (31)
0.3492	14.82	0.0000	53	54	20000	sample1.sub2._PRL_1_ (32)
2.3564	100.00	-	-	-	-	-

Load Balance (33)

Min(sec)	Max(sec)	Barrier (sec)	%	Start	End	Count	Routine name
1.5789	1.5873	0.0084	0.53	42	43	100000	sample1.sub1._PRL_1_
0.3160	0.3185	0.0025	0.79	53	54	20000	sample1.sub2._PRL_1_
2.1324	2.1446	0.0122	0.57	-	-	-	-

Thread 0 (34)

Cost (sec)	%	Barrier (sec)	%	Start	End	Routine name
1.5789	74.04	0.0084	0.53	42	43	sample1.sub1._PRL_1_
0.3160	14.82	0.0025	0.79	53	54	sample1.sub2._PRL_1_
2.1324	100.00	0.0122	0.57	-	-	-

Thread 1

Cost (sec)	%	Barrier (sec)	%	Start	End	Routine name
1.5873	74.02	0.0000	0.00	42	43	sample1.sub1._PRL_1_
0.3185	14.85	0.0000	0.00	53	54	sample1.sub2._PRL_1_
2.1446	100.00	0.0000	0.00	-	-	-

- (1) Elapsed time
- (2) User CPU time
- (3) System CPU time

- (4) Information for each routine that operates in serial processing (-Nrt_tune_func is specified)
- (5) Cost (sec)
- (6) Percentage of the entire cost
- (7) Start line number
- (8) End line number
- (9) Call frequency
- (10) Cost called once
- (11) Routine name
- (12) Information for each routine
- (13) Total value for each routine
- (14) Information for each routine that operates in multi processing (-Nrt_tune_func is specified)
- (15) Load balance information for each routine that operates in multi processing (-Nrt_tune_func is specified)
- (16) Cost of the thread which cost has a highest
- (17) Cost of the thread which cost has a lowest
- (18) Subtract lowest cost from highest cost
- (19) The ratio of subtract highest cost from lowest cost for highest cost
- (20) Information for each thread in routine that operates in multi processing (-Nrt_tune_func is specified)
- (21) Wait cost for synchronization between threads (sec)
- (22) The ratio of wait cost for synchronization between threads to all
- (23) Information for each loop that operates in serial processing (-Nrt_tune_loop is specified)
- (24) Nest level
- (25) Information for each loop
- (26) Total value for each loop
- (27) Information for each loop that in parallel region by OpenMP (-Nrt_tune_loop is specified)
- (28) Load balance information for each loop that in parallel region by OpenMP (-Nrt_tune_loop is specified)
- (29) Information for each thread in loop that in parallel region by OpenMP (-Nrt_tune_loop is specified)
- (30) Information for each parallel region
- (31) Information for parallel region
- (32) Total value for each parallel region
- (33) Load balance information for each thread in parallel region
- (34) Information for each thread in parallel region

Remarks:

Information for automatically parallelized locations and locations changed to OpenMP are output as generated procedure information. An identifier is added at the end of the name to form the routine name.

When information is fetched for each loop, the generated procedure name is output as the routine name. The demangle name is output for C++. The routine name is enclosed with parentheses at the loop that is nested.

"[Table 6.3 Generated procedure names in thread parallel programs](#)" shows the generated procedure names for thread parallel programs.

Figure 6.2 Output example of Cost Information (environmental variable FLIB_RTINFO_CSV is specified)

```

=====
COST INFORMATION
=====
Type,Process ID,Thread ID,Range,Elapsed(sec),User (sec),System(sec)
[COST],0,-,"PROGRAM",2.5100,4.9900,0.0300

Serial Routine
Type,Process ID,Thread ID,Range,Cost(sec),% Once(sec),Start,End,Count,Routine name
[COST_ROUTINE_SERIAL],0,-,"PROGRAM",0.2591,82.11,0.0259,40,49,10,"sub1"
[COST_ROUTINE_SERIAL],0,-,"PROGRAM",0.0564,17.89,0.0000,51,56,2000,"sub2"
[COST_ROUTINE_SERIAL],0,total,"PROGRAM",0.3155,100.00,-,-,-,-

Parallel Routine
Type,Process ID,Thread ID,Range,Cost(sec),% Once(sec),Start,End,Count,Routine name
[COST_ROUTINE_PRL],0,-,"PROGRAM",1.6753,83.94,0.1675,40,49,10,"sub1"
[COST_ROUTINE_PRL],0,-,"PROGRAM",0.3206,16.06,0.0022,51,56,2000,"sub2"
[COST_ROUTINE_PRL],0,total,"PROGRAM",1.9959,100.00,-,-,-,-

Load Balance
Type,Process ID,Thread ID,Range,Min(sec),Max(sec),Barrier(sec),% Start,End,Count,Routine name
[COST_ROUTINE_PRL],0,-,"PROGRAM",1.6591,1.6753,0.0162,0.97,40,49,10,"sub1"
[COST_ROUTINE_PRL],0,-,"PROGRAM",0.3184,0.3206,0.0022,0.67,51,56,2000,"sub2"
[COST_ROUTINE_PRL],0,total,"PROGRAM",1.9775,1.9959,0.0184,0.92,-,-,-

Thread 0
Type,Process ID,Thread ID,Range,Cost(sec),% Barrier(sec),% Start,End,Routine name
[COST_ROUTINE_PRL],0,0,"PROGRAM",1.6753,83.94,0.0000,0.00,40,49,"sub1"
[COST_ROUTINE_PRL],0,0,"PROGRAM",0.3206,16.06,0.0000,0.00,51,56,"sub2"
[COST_ROUTINE_PRL],0,total,"PROGRAM",1.9959,100.00,0.0000,0.00,-,-,-

Thread 1
Type,Process ID,Thread ID,Range,Cost(sec),% Barrier(sec),% Start,End,Routine name
[COST_ROUTINE_PRL],0,1,"PROGRAM",1.6591,83.90,0.0162,0.97,40,49,"sub1"
[COST_ROUTINE_PRL],0,1,"PROGRAM",0.3184,16.10,0.0022,0.67,51,56,"sub2"
[COST_ROUTINE_PRL],0,total,"PROGRAM",1.9775,100.00,0.0184,0.92,-,-,-

Serial Loop
Type,Process ID,Thread ID,Range,Cost(sec),% Once(sec),Start,End,Count,nest,Routine name
[COST_LOOP_SERIAL],0,-,"PROGRAM",2.1237,43.17,2.1237,19,29,1,1,"sample1"
[COST_LOOP_SERIAL],0,-,"PROGRAM",1.8614,37.84,0.1861,41,42,10,2,"(sample1.sub1)"
[COST_LOOP_SERIAL],0,total,"PROGRAM",0.3155,100.00,-,-,-,-

Parallel Loop
Type,Process ID,Thread ID,Range,Cost(sec),% Once(sec),Start,End,Count,nest,Routine name
[COST_LOOP_PRL],0,-,"PROGRAM",0.0728,47.52,0.0073,45,47,10,2,"(sample1.sub1._CMP_3)"
[COST_LOOP_PRL],0,-,"PROGRAM",0.0669,43.67,0.0000,46,46,10000,3,"(sample1.sub1._CMP_3)"
[COST_LOOP_PRL],0,total,"PROGRAM",1.9959,100.00,-,-,-,-

Load Balance
Type,Process ID,Thread ID,Range,Min(sec),Max(sec),Barrier(sec),% Start,End,Count,Routine name
[COST_LOOP_PRL],0,-,"PROGRAM",0.0728,0.0728,0.0001,0.09,45,47,10,"(sample1.sub1._CMP_3)"
[COST_LOOP_PRL],0,-,"PROGRAM",0.0668,0.0669,0.0002,0.25,46,46,10000,"(sample1.sub1._CMP_3)"
[COST_LOOP_PRL],0,total,"PROGRAM",0.1530,0.1533,0.0003,0.19,-,-,-

Thread 0
Type,Process ID,Thread ID,Range,Cost(sec),% Barrier(sec),% Start,End,Routine name
[COST_LOOP_PRL],0,0,"PROGRAM",0.0728,47.54,0.0000,0.00,45,47,"(sample1.sub1._CMP_3)"
[COST_LOOP_PRL],0,0,"PROGRAM",0.0669,43.69,0.0000,0.00,46,46,"(sample1.sub1._CMP_3)"
[COST_LOOP_PRL],0,total,"PROGRAM",0.1532,100.00,0.0001,0.04,-,-,-

Thread 1
Type,Process ID,Thread ID,Range,Cost(sec),% Barrier(sec),% Start,End,Routine name
[COST_LOOP_PRL],0,1,"PROGRAM",0.0728,47.55,0.0001,0.09,45,47,"(sample1.sub1._CMP_3)"
[COST_LOOP_PRL],0,1,"PROGRAM",0.0668,43.63,0.0002,0.25,46,46,"(sample1.sub1._CMP_3)"
[COST_LOOP_PRL],0,total,"PROGRAM",0.1531,100.00,0.0002,0.15,-,-,-

Parallel Region
Type,Process ID,Thread ID,Range,Cost(sec),% Once(sec),Start,End,Count,Routine name
[COST_PRL],0,-,"PROGRAM",1.7746,74.43,0.0000,42,43,100000,"sample1.sub1._PRL_1"
[COST_PRL],0,-,"PROGRAM",0.3555,14.91,0.0000,53,54,20000,"sample1.sub2._PRL_1"
[COST_PRL],0,total,"PROGRAM",2.3844,100.00,-,-,-,-

Load Balance
Type,Process ID,Thread ID,Range,Min(sec),Max(sec),Barrier(sec),% Start,End,Count,Routine name
[COST_PRL],0,-,"PROGRAM",1.5863,1.6024,0.0161,1.01,42,43,100000,"sample1.sub1._PRL_1"
[COST_PRL],0,-,"PROGRAM",0.3184,0.3206,0.0022,0.67,53,54,20000,"sample1.sub2._PRL_1"
[COST_PRL],0,total,"PROGRAM",2.1425,2.1618,0.0193,0.89,-,-,-

Thread 0
Type,Process ID,Thread ID,Range,Cost(sec),% Barrier(sec),% Start,End,Routine name
[COST_PRL],0,0,"PROGRAM",1.6024,74.16,0.0000,0.00,42,43,"sample1.sub1._PRL_1"
[COST_PRL],0,0,"PROGRAM",0.3206,14.84,0.0000,0.00,53,54,"sample1.sub2._PRL_1"
[COST_PRL],0,total,"PROGRAM",2.1609,100.00,0.0009,0.04,-,-,-

Thread 1
Type,Process ID,Thread ID,Range,Cost(sec),% Barrier(sec),% Start,End,Routine name
[COST_PRL],0,1,"PROGRAM",1.5863,74.01,0.0161,1.01,42,43,"sample1.sub1._PRL_1"
[COST_PRL],0,1,"PROGRAM",0.3184,14.86,0.0022,0.67,53,54,"sample1.sub2._PRL_1"
[COST_PRL],0,total,"PROGRAM",2.1434,100.00,0.0184,0.85,-,-,-

```

Chapter 7 Input-Output Information

This chapter describes the Input-Output Information.

7.1 Specification of Input Output Information

Input-Output Information can be output by specifying the following environment variables.

Table 7.1 Format of FLIB_RTINFO_IO environment variable

Environment variable name	Operand
FLIB_RTINFO_IO	F
	C
	A
	-

- When F is specified for an operand, I/O information on Fortran is output.
- When C is specified for an operand, I/O information on C/C++ is output.
- When A is specified for an operand, I/O information on Fortran and C/C++ is output.
- If the operand value is omitting, it is assumed that A is specified as the value.
If the operand value is invalid, runtime message RTINF0020 is output and it is assumed that A is specified as the value. Refer to "[Chapter 9 Message](#)" for runtime message.

It is necessary to specify the compiler option `Nrt_tune_io` when Input-Output information for C/C++ is output.

Moreover, Input-Output Information on C/C++ for each range is not output.

7.2 Fortran Input-Output Information

7.2.1 Overview of Fortran Input-Output Information

The information shown below is fetched for each unit and connected file during program execution.

The following information is output, as "information of the units"

- Filename
- File modify date
- Unit number
- Buffer size
- Access type
- Format type
- File size
- Number of records

The following information is output, as "statement information"

- Execution time
- Input-output item size
- Statement called count
- Statement type

- Unit number (*1)
- line number
- Input-output statement call source routine name

The following information is output, as "system call cumulative information"

- Information related to read
 - read size
 - read count
 - read duration
- Information related to write
 - write size
 - write count
 - write duration
- Information related to seek
 - seek count
- I/O buffer parallel transfer count (*2)

The following information is output, as "information of asynchronous input-output"

- Information related to asynchronous input
 - Total time depended on a completion demand of the asynchronous input
 - The number of times demand of the asynchronous input
- Information related to asynchronous output
 - Total time depended on a completion demand of the asynchronous input
 - The number of times demand of the asynchronous input

*1) Only in the case of the information each specified range, it is output.

*2) Refer to "Fortran User's Guide" for details of I-O buffer parallel transfers.

7.2.2 Details of Fortran Input-Output Information

7.2.2.1 Output specification

Input-output Information for each connected file and unit is output, and information related to internal files and information related to external files is fetched eachly. Input-Output Information is output in the sequence in which those connections were closed.

However, statement information is not fetched for standard input-output files. And, information related to internal files is fetched only statement information.

7.2.2.2 Unit Information

The unit information is mainly related to file and unit connections, such as the unit number, access type, and so on. The items fetched as unit information are explained below.

- Filename
 - The absolute path of the file connected with the unit is output. However, when the standard file is connected, either stdin, stdout or stderr is output as a file name.
- File modify date
 - The date of the last modify to the file is output.

- Unit number
 - The unit number of the unit connected the file is output.
- buffer size
 - The size of the buffer fetched by the input-output statement is output. The unit is kilobytes.
- Access type
 - Either sequential, direct, or stream is output. These mean sequential seek, direct seek, and stream seek respectively.
- Format type
 - Either formatted, unformatted, or binary is output. These mean, formatted, unformatted, and binary format respectively.
- File size
 - The file size is output. The unit is bytes. If file size acquisition fails, or if stdout or stderr was specified, '***' is output.
- Number of records
 - If access type is direct, the number of Fortran records handled as one fixed-block is output.
 - If access type is sequential or stream, '***' is output.

7.2.2.3 Statement Information

The statement information is information concerning input-output statements, such as the size of items input and output when a READ statement or WRITE statement is executed for the unit, elapsed time, and so on. Information is fetched for only the number of input-output statements in the specified range.

The items fetched as statement information are explained below.

- Execution time
 - The elapsed time taken to execute the input-output statement is output. The unit is milliseconds.
 - If the same input-output statement is executed multiple times, the total elapsed time is output. On subsequent lines, the average value, maximum value, and minimum value are output.
- Input-output item size
 - The size of the input-output statement input-output item is output. The unit is bytes.
 - If the same input-output statement is executed multiple times, the total size is output. On subsequent lines, the average value, maximum value, and minimum value are output.
- Statement called count
 - The number of times the target input-output statement was called is displayed.
- Statement type
 - Either r or w is output. These mean a READ statement and a WRITE statement respectively.
- Input-output statement line number
 - The number of the line where the input-output statement is written is output.
- Input-output statement call source function name
 - The input-output statement call source function name is output.

Statement information is output in sequence from the first execution of the input-output statement in that line number.

If an input-output statement in the same line number is executed multiple times, information is consolidated as information for one input-output statement. In this case, for input-output item size and execution duration, total, average, maximum, and minimum values are fetched.

The total functions as an indicator showing to what extent the calling of that input-output statement affects execution of the entire program. The average functions as an indicator for comparing the cost of one execution with that of other input-output statements. The maximum

and minimum values function as an indicator of the dispersion of costs in accordance with the occasions when that input-output statement is called.

If the execution of the input-output statement is only once, average, maximum, minimum is not output.

However, if FLIB_RTINFO_CSV is specified, and output by CSV format, average, maximum, minimum is always output.

By setting the FLIB_RTINFO_THRESHOLD environment variable, information output can be restricted to the input-output statements for which the execution duration (total value) exceeds the value specified in the environment variable. The value is specified in milliseconds, and an integer value from 1 to 2147483647 can be specified. If the specified value is invalid, the environment variable specification is disabled. If this environment variable is not set, information is output for all input-output statements.

Table 7.2 Format of FLIB_RTINFO_THRESHOLD environment variable

Environment variable name	Operand
FLIB_RTINFO_THRESHOLD	Integer value (1 to 2147483647)

7.2.2.4 System Call Cumulative Information

System cumulative information is the cumulative total for the time taken for execution of the system calls issued internally during execution within input-output statements. The items fetched as system call cumulative information are explained below.

- read-related information
 - The total size of data that is input by means of read(2) and fread functions (includes fgets function) is output as the read size. The unit is bytes.
 - The read(2) and fread functions (includes fgets function) issue count is output as the read count.
 - The total time taken to execute read(2) and fread functions (includes fgets function) is output as the read duration. The unit is milliseconds.
- write-related information
 - The total size of data that is output by means of write(2) and fwrite functions (includes fputs function) is output as the write size. The unit is bytes.
 - The write(2) and fwrite functions (includes fputs function) issue count is output as the write count.
 - The total time taken to execute write(2) and fwrite functions (includes fputs function) is output as the write duration. The unit is milliseconds.
- seek-related information
 - The lseek(2) and fseek functions (includes rewind function) issue count is output as the seek count.
- I/O buffer parallel transfer count
 - The I/O buffer parallel transfer count is output.

7.2.2.5 Information of Asynchronous Input-Output

Information on asynchronous I/O is information on the completion demand of the asynchronous I/O processing. This information acquires each of the input process and the output processing. The explanation of each item acquired as information on asynchronous I/O is shown below.

- Information related to asynchronous input
 - The total of time that hangs to the completion demand at the asynchronous input process and the frequency of the completion demand are output.
- Information related to asynchronous output
 - The total of time that hangs to the completion demand at the asynchronous output process and the frequency of the completion demand are output.

The completion demand of the asynchronous I/O processing being done shows that the asynchronous I/O processing is effective.

7.2.2.6 Range Specification

Ranges from a `START_COLLECTION` to a `STOP_COLLECTION` that have the same argument value specified can be specified, as in the range specifications described in "3.2.1 Fortran".

Example

```
START_COLLECTION("region1")
STOP_COLLECTION("region1")
START_COLLECTION("region2")
STOP_COLLECTION("region2")
```

The input-output statement executed within the range of "Region1" is output as I/O information within the range of "Region1".

The input-output statement executed within the range of "Region2" is output as I/O information within the range of "Region2".

Note

- If a nested range is specified, information for that range is not output.

Example

```
START_COLLECTION("region1")
START_COLLECTION("region2")
STOP_COLLECTION("region2")
STOP_COLLECTION("region1")
```

The input-output statement executed within the range of "Region1" is output as I/O information within the range of "Region1".

The input-output statement executed within the range of "Region2" is not output as I/O information within the range of "Region2".

- If overlapping ranges are specified, information is output for the range from the earlier specified `START_COLLECTION` to the corresponding `STOP_COLLECTION`.

Example

```
START_COLLECTION("region1")
START_COLLECTION("region2")
STOP_COLLECTION("region1")
STOP_COLLECTION("region2")
```

The input-output statement executed within the range of "Region1" is output as I/O information within the range of "Region1".

The input-output statement executed within the range of "Region2" is not output as I/O information within the range of "Region2".

7.2.3 Output Example of Fortran Input-Output Information

Output example of Fortran Input-Output Information results is shown below.

Figure 7.1 Output example of Fortran Input-Output Information

```

+-----+
||                                     I/O INFORMATION                                     ||
+-----+
*****
Fortran I/O INFORMATION
*****

File name       : /export/home/user/work1.data (1)
Time of last access : Wed Apr  6 16:32:48 2011 (2)
+-----+-----+-----+-----+-----+-----+
| (3) | (4) | (5) | (6) | (7) | (8) |
| Unit | Buffer size | Access | Format | File size | The number |
|      | (K byte)  |       |       | (byte)   | of records |
+-----+-----+-----+-----+-----+-----+
| 10  | 8192 | sequential | unformatted | 5000079 | *** |
+-----+-----+-----+-----+-----+-----+

==== Input/output statement ====
Time(msec)   Size(byte)   Count   R/W   Line   Func name
+-----+-----+-----+-----+-----+-----+
(9)          (10)         (11)    (12) (13)   (14)
2.906        8             1       W     15    MAIN_
2.392        4             1       R     17    MAIN_
1.769       15            5       W     20    MAIN_
0.354        3             3       <---Ave
0.000        3             3       <---Min
1.767        3             3       <---Max
0.111       5000000       1       W     22    sample.subroutine01
+-----+-----+-----+-----+-----+-----+

==== System call ====
Time(msec)   Size(byte)   Count
+-----+-----+-----+
read/hread   2.381        16         1 (15)
write/fwrite 0.117       5000095    5 (16)
+-----+-----+-----+
seek count           : 4 (17)
I/O buffer parallel transfer count : 0 (18)

==== Asynchronous I/O suspend ====
Time(msec)   Count
+-----+-----+
read         0.000        0 (19)
write       4.097        1 (20)
+-----+-----+

*****

```

- (1) Filename
- (2) File access date
- (3) Unit number
- (4) Buffer size
- (5) Access type
- (6) Format type
- (7) File size
- (8) Number of records
- (9) Runtime elapsed time
- (10) Input-output item size
- (11) Statement called count
- (12) Statement type
- (13) Input-output statement line number
- (14) Input-output statement call source function name
- (15) read-related information (read size, count, duration)
- (16) write-related information (write size, count, duration)
- (17) seek-related information (seek count)
- (18) I/O buffer parallel transfer count

(19) Information related to asynchronous input (depended on a completion demand, the number of times demand)

(20) Information related to asynchronous output (depended on a completion demand, the number of times demand)

Figure 7.2 Output example of Fortran Input-Output Information (environmental variable FLIB_RTINFO_CSV is specified)

```
=====
I/O INFORMATION
=====
*****
Fortran I/O INFORMATION
*****
[IO_UNIT], 0, -, "_PROGRAM_", "/export/home/user/work1.data", Wed Apr 6 16:32:48 2011, 10, 8192, sequential, unformatted, 5000079, ***, 2.381, 16, 1,
0.117, 5000095, 5, 4, 0, 0.000, 0, 4.097, 1
[IO], 0, -, "_PROGRAM_", 2.906, 2.906, 2.906, 2.906, 0, 0, 0, 0, 1, W, 15, "MAIN_"
[IO], 0, -, "_PROGRAM_", 2.392, 2.392, 2.392, 2.392, 4, 4, 4, 4, 1, R, 17, "MAIN_"
[IO], 0, -, "_PROGRAM_", 1.769, 1.767, 0.000, 0.354, 15, 3, 3, 3, 5, W, 20, "MAIN_"
[IO], 0, -, "_PROGRAM_", 0.111, 0.111, 0.111, 0.111, 5000000, 5000000, 5000000, 5000000, 1, W, 22, "sample.subroutine01"
```

7.3 C/C++ Input-Output Information

7.3.1 Overview of Fortran Input-Output Information

It is necessary to specify the compiler option `Nrt_tune_io` when Input-Output information for C/C++ is output.

Input-Output information for C/C++ can acquire information of `open(2)` or `open64(2)` and `close(2)` and `read(2)`, `write(2)`, and `fsync(2)` between before close corresponding to practice of `open(2)` or `open64(2)`.

However, these Input-Output statements are limited to be described in the source program.

If `close(2)` corresponding to `open(2)` or `open64(2)` is not executed, Input-Output Information on the file that opens with previous `open(2)` or `open64(2)` is not output.

Moreover, I/O information is not output because it doesn't become a measurement object when the function prototype is not declared in the type with a correct service function (prototype declaration). To declare the function prototype in a correct type, should be include an appropriate standard header.

7.3.2 Details of Fortran Input-Output Information

Output information is described as follows.

- Filename

The absolute path name being regularized for the file name specified for `open(2)` or `open64(2)` is output.

- File modify date

The last time of modify file opened is output.

- File descriptor

The value of the file descriptor that `open(2)` or `open64(2)` returns is output.

- File system

The filesystem name where the file exists is output.

- File size

The size of the file acquired after `close(2)` is executed is output by Kbyte.

- Process number

The process number when `open(2)` or `open64(2)` operates is output.

- Information related to open (the following information acquired when open(2) or open64(2) is executed)
 - Elapsed time
Total time of the elapsed time when open(2) or open64(2) is executed is output by milliseconds.
 - System time
Total time of the system time when open(2) or open64(2) is executed is output by milliseconds.
 - Average elapsed time
Average time of the elapsed time when open(2) or open64(2) is executed is output by milliseconds.
 - Execution frequency
Execution frequency of open(2) or open64(2) is output.
- Information related to read (the following information acquired when read(2) is executed)
 - Elapsed time
Total time of the elapsed time when read(2) is executed is output by milliseconds.
 - System time
Total time of the system time when read(2) is executed is output by milliseconds.
 - Average elapsed time
Average time of the elapsed time when read(2) is executed is output by milliseconds.
 - Throughput
Throughput when read(2) is executed is output.
 - Transfer size
Total size of transfer when read(2) is executed is output by Kbyte.
 - Execution frequency
Execution frequency of read(2) is output.
- Information related to write (the following information acquired when write(2) is executed)
 - Elapsed time
Total time of the elapsed time when write(2) is executed is output by milliseconds.
 - System time
Total time of the system time when write(2) is executed is output by milliseconds.
 - Average elapsed time
Average time of the elapsed time when write(2) is executed is output by milliseconds.
 - Throughput
Throughput when write(2) is executed is output.
 - Transfer size
Total size of transfer when write(2) is executed is output by Kbyte.
 - Execution frequency
Execution frequency of write(2) is output.
- Information related to fsync (the following information acquired when fsync(2) is executed)
 - Elapsed time
Total time of the elapsed time when fsync(2) is executed is output by milliseconds.
 - System time
Total time of the system time when fsync(2) is executed is output by milliseconds.
 - Average elapsed time
Average time of the elapsed time when fsync(2) is executed is output by milliseconds.
 - Execution frequency
Execution frequency of fsync(2) is output.

- Information related to close (the following information acquired when close(2) is executed)
 - Elapsed time
Total time of the elapsed time when close(2) is executed is output by milliseconds.
 - System time
Total time of the system time when close(2) is executed is output by milliseconds.
 - Average elapsed time
Average time of the elapsed time when close(2) is executed is output by milliseconds.
 - Execution frequency
Execution frequency of close(2) is output.

When the filesystem name cannot be acquired, "Unknown" is output.

When the File modify date cannot be acquired, "*****" is output.

When filesize exceeds 9223372036854775807 byte, "9007199254740991KB" is output.

When elapsed time, system time or average elapsed time exceeds 9223372036854775807 micro second, "92233720368547.75" is output.

When execution frequency exceeds 9223372036854775807, "9223372036854775807" is output.

7.3.3 Output Example of C/C++ Input-Output Information

Output example of C/C++ Input-Output Information results is shown below.

Figure 7.3 Output example of C/C++ Input-Output Information

```

=====
||                                     I/O INFORMATION                                     ||
=====
*****
C/C++ I/O INFORMATION
*****

==== I/O System call Information ====
File Name      : /home/guest/test/x.data (1)
Access Time    : Fri Oct 14 18:05:09 2011 (2)
File Descriptor : 4 (3)
File System Type : ext4 (4)
File Size      : 4000000 KB (5)
Rank Id        : 0 (6)

I/O      Elapsed      System      Average      Throughput      Size (KB)      Count
(7)----(8)-----Time (ms)-----Time (ms)-----Time (ms)----- (MB/sec)-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
open      0.004          0.000          0.004          --              --              1
read      30180.956       1620.000       0.060          64.710         2000000         500000
write     14375.040       2120.000       0.029          135.860        2000000         500000
fsync     1.036           0.000          1.036          --              --              1
close     0.017           0.000          0.017          --              --              1
=====

```

- (1) Filename
- (2) File modify date
- (3) File descriptor
- (4) File system
- (5) File size
- (6) Process number
- (7) System call name
- (8) Elapsed time (total)
- (9) System time (total)
- (10) Elapsed time (average)
- (11) Throughput

(12) Transfer size (total)

(13) Execution frequency (total)

Remarks:

- Information for open64(2) is output as for open.
- Throughput and the transfer size are output only to information for read and write.

Figure 7.4 Output example of C/C++ Input-Output Information (environmental variable FLIB_RTINFO_CSV is specified)

```
=====
I/O INFORMATION
=====
*****
C/C++ I/O INFORMATION
*****
[IO_C_SYS],0,-,"__PROGRAM__",/home/guest/test/x.data",Fri Oct 14 18:05:09 2011,4,ext4,4000000,0.004,0.000,0.004,-,1,30180.956,1620.000,0.060
,64.710,2000000,500000,14375.040,2120.000,0.029,135.860,2000000,500000,1.036,0.000,1.036,-,1,0.017,0.000,0.017,-,1
```


Chapter 8 Hardware Monitor Information

This chapter describes the Hardware Monitor Information.

Measured values for the processor operating status during program execution, such as the number of instructions executed, the cache mistake ratio, and so on, are the Hardware Monitor Information.

8.1 Overview of Hardware Monitor Information

The Hardware Monitor Information is output in the format summarized related plural information in as a category.

Information to show in the next paragraph by specifying this category for environment variable FLIB_RTINFO_PA is output.

The table below shows how to specify a category name in FLIB_RTINFO_PA. Note that multiple categories cannot be specified.

If a category name is not specified in FLIB_RTINFO_PA, the information that is fetched is the same as when Statistics is specified as the category name. If there is a mistake in the category name, runtime message RTINF0001 is output and the information fetched is the same as when Statistics is specified.

Refer to "[Chapter 9 Message](#)" for runtime message.

Table 8.1 Format of FLIB_RTINFO_PA environment variable

Environment variable name	Operand
FLIB_RTINFO_PA	Statistics
	Instructions_SIMD
	Instructions_NOSIMD
	Cache
	MEM_access
	Performance
	TLB
	-

8.2 Details of Hardware Monitor Information

8.2.1 Fetching Information in Each Category

The Hardware Monitor Information that can be fetched in each category is shown in the table below.

Table 8.2 Hardware Monitor Information that can be fetched when a category name is specified

Category name	Display name	Information	Overview
Statistics	Time(sec)	Elapsed time	Time required to execute instructions (seconds)
	MFLOPS	MFLOPS	Execution efficiency for floating point operations (average number of floating point operations executed per second)
	MFLOPS/PEAK(%)	MFLOPS peak performance ratio	Ratio of actual value to MFLOPS logical peak value
	MIPS	MIPS	Instruction execution ratio (average number of instructions executed per second)
	MIPS/PEAK(%)	MIPS peak performance ratio	Ratio of actual value to MIPS logical peak value
	Float/operate (%)	Floating point operation ratio	Ratio of the number of floating point operations executed to the number of operations

Category name	Display name	Information	Overview
Instructions_S IMD	Time(sec)	Elapsed time	Time required to execute instructions (seconds)
	Inst	Number of instructions executed	Number of instructions executed
	MIPS	MIPS	Instruction execution ratio (average number of instructions executed per second)
	MIPS/PEAK(%)	MIPS peak performance ratio	Ratio of actual value to MIPS logical peak value
	SIMD-inst(%) - Load-store - Float - fma - Fixed - total	- Load/store instruction ratio - Floating point operation instruction ratio - Floating point sum operation instruction ratio - Fixed point operation instruction ratio - SIMD instruction ratio	The ratio of execution of the following SIMD to the number of instruction execution. - load/store instructions - floating point operation instructions - floating point sum operation instructions - fixed point operation instructions - Ratio of SIMD instructions to the number of instructions executed
	Load-store_SIMD/ Load-store (%)	SIMD load/store instruction ratio	Ratio of the number of SIMD load and SIMD store instructions to the number of load and store instructions executed
	prefetch_inst	Number of pre-fetch instructions	Number of pre-fetch instructions
	indirect_prefetch_inst	Number of indirect pre-fetch instructions	Number of indirect pre-fetch instructions
Instructions_ NOSIMD	Time(sec)	Elapsed time	Time required to execute instructions (seconds)
	Inst	Number of instructions executed	Number of instructions executed
	MIPS	MIPS	Instruction execution ratio (average number of instructions executed per second)
	MIPS/PEAK(%)	MIPS peak performance ratio	Ratio of actual value to MIPS logical peak value
	NOSIMD-inst(%) - Load-store - Float - fma - Fixed - total	- Load/store instruction ratio - Floating point operation instruction ratio - Floating point sum operation instruction ratio - Fixed point operation instruction ratio - NOSIMD instruction ratio	The ratio of execution of the following NOSIMD to the number of instruction execution. - load/store instructions - floating point operation instructions - floating point sum operation instructions - fixed point operation instructions - Ratio of NOSIMD instructions to the number of instructions executed
	Load-store_NOSIMD/ Load-store (%)	NOSIMD load/store instruction ratio	Ratio of the number of NOSIMD load and NOSIMD store instructions to the number of load and store instructions executed
	prefetch_inst	Number of pre-fetch instructions	Number of pre-fetch instructions
	indirect_prefetch_inst	Number of indirect pre-fetch instructions	Number of indirect pre-fetch instructions
Cache	Time(sec)	Elapsed time	Time required to execute instructions (seconds)
	Inst	Number of instructions executed	Number of instructions executed

Category name	Display name	Information	Overview
	L1I miss (%)	Level 1 instruction cache mistake ratio	Ratio of level 1 instruction cache mistake occurrences to the number of instructions executed
	L1D miss/ mem_access-inst(%)	Level 1 data cache mistake ratio	Ratio of level 1 data cache mistake occurrences to the number of memory access (load/store) instructions
	L2 miss(%) - demand - prefetch	Level 2 cache mistake ratio - Level 2 cache demand mistakes ratio - Level 2 cache prefetch mistakes ratio	Ratio of level 2 cache mistake issues to the number of memory access (load/store) instructions - accounted for by demand request mistakes - accounted for by pre-fetch request mistakes
MEM_access	Time(sec)	Elapsed time	Time required to execute instructions (seconds)
	Load-store_SIMD- inst(%)	Load/store instruction ratio (SIMD instruction ratio)	Ratio of the number of SIMD load and SIMD store instructions to the number of instructions executed
	Load- store_NOSIMD- inst(%)	Load/store instruction ratio (NOSIMD instruction ratio)	Ratio of the number of NOSIMD load and NOSIMD store instructions to the number of instructions executed
	Mem throughput _core(GB/S)	Memory access throughput	Average quantity of data transferred between memory and CPU per second (Gbytes per second)
Performance	Time(sec)	Elapsed time	Time required to execute instructions (seconds)
	2-4inst_commit	2-4 instruction commit	Time taken to execute a cycle that has 2 or more completion instructions (seconds)
	1inst_commit	1 instruction commit	Time taken to execute a cycle that has 1 completion instruction (seconds)
	operate_wait	Operation wait	Time taken to execute a cycle that has 0 completion instructions that is the oldest instruction within the instructions being executed (seconds)
	Cache_access_wait	Cache access wait	Time taken, due to a cache access wait, to execute a cycle that has 0 completion instructions (seconds)
	Mem_access_wait	Memory access wait	Time taken, due to a memory access wait, to execute a cycle that has 0 completion instructions (seconds)
	Inst_fetch_wait	Instruction fetch wait	Time taken, due to the CSE (Commit Stack Entry) being empty, to execute a cycle that has 0 completion instructions (seconds)
	Other_wait	Other waits	Time taken, for reasons other than the above, to execute a cycle that has 0 completion instructions (seconds)
TLB	Time(sec)	Elapsed time	Time required to execute instructions (seconds)
	Inst	Number of instructions executed	Number of instructions executed
	uTLB miss/ mem_access-inst (%)	Micro data TLB mistake ratio	Ratio of micro data TLB mistake issues to the number of memory access (load/store) instructions executed
	mTLB miss/ mem_access-inst (%)	Data main TLB mistake ratio	Ratio of data main TLB mistake issues to the number of memory access (load/store) instructions executed

8.2.2 Switching Collection Mode

The Hardware Monitor Information collection mode can be switched by specifying the FLIB_RTINFO_PA_MODE environment variable.

If u is specified as the value, Hardware Monitor Information is collected during execution of user programs.

If s is specified as the value, Hardware Monitor Information is collected during execution of tasks by the OS.

If a is specified as the value, total values for the above are fetched.

If FLIB_RTINFO_PA_MODE is not specified, or if the operand value is incorrect, it is assumed that a is specified as the value and total values are fetched.

Table 8.3 Format of FLIB_RTINFO_PA_MODE environment variable

Environment variable name	Operand
FLIB_RTINFO_PA_MODE	u, s or a

8.2.3 Fetching for Each Routine

Hardware Monitor Information can be fetched for each routine by specifying the `-Nrt_tune_func` compile option. Refer to "2.2 Compiler Options" for information concerning the compile options.

The maximum measurement frequency to acquire information on the routine by specifying environment variable `FLIB_RTINFO_FUNCMAX_PA` can be specified. The integral value from 0 to 2147483647 is specified for an operand. If 0 is specified, information on all call is output. If the value of the operand is invalid or the environment variable is not specified, it is considered that 1000 is specified.

If the call frequency of the routine exceeds the maximum measurement frequency, information to reaching to the maximum measurement frequency is output. In this case, "*" is output next to the call frequency of the routine.

Table 8.4 Format of FLIB_RTINFO_FUNCMAX_PA environment variable

Environment variable name	Operand
FLIB_RTINFO_FUNCMAX_PA	Integer value (0 to 2147483647)

Even when the compiler option `-Nrt_tune_func` is effective, if environment variable `FLIB_RTINFO_NOFUNC` is specified, the output of information on each routine is suppressed.

Table 8.5 Format of FLIB_RTINFO_NOFUNC environment variable

Environment variable name	Operand
FLIB_RTINFO_NOFUNC	(None)

8.3 Calculation Expressions for Hardware Monitor Information

This section gives the calculation expressions for the Hardware Monitor Information listed in "8.2 Details of Hardware Monitor Information".

8.3.1 "Statistics" Specified for Category Name

When Statistics is effective as a category of the Hardware Monitor Information, information based on the following calculation expressions is output.

Table 8.6 Hardware Monitor Information calculation expressions (Category name : Statistics)

Information	Calculation expression
MFLOPS	Number of all floating point operations / 1000000 / Elapsed time
MFLOPS peak performance ratio (%)	MFLOPS / (Number of executed cores * MFLOPS peak value for core units) * 100
MIPS	Number of instructions executed/1000000 / Elapsed time
MIPS peak performance ratio (%)	MIPS / (Number of executed cores * MIPS peak value for core units) * 100
Floating point operation ratio (%)	Number of all floating point operations / Number of operations * 100

Information on the entire process is calculated by the following calculation expressions.

Table 8.7 Hardware Monitor Information on the entire process calculation expressions (Statistics)

Information	Calculation expression
MFLOPS	Total number of all floating point operations in all threads / 1000000 / Elapsed time of a thread having the longest elapsed time
MFLOPS peak performance ratio (%)	MFLOPS in entire process / (Number of executed cores * MFLOPS peak value for core units) * 100
MIPS	Total number of instructions executed in all threads / 1000000 / Elapsed time of a thread having the longest elapsed time
MIPS peak performance ratio (%)	MIPS in entire process / (Number of executed cores * MIPS peak value for core units) * 100
Floating point operation ratio (%)	Total number of all floating point operations in all threads / Total number of operations in all threads * 100



Note

The value of PA event acquired in the calculation expression mentioned above includes cost waiting barrier of the parallel region.

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

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

Total number of all floating point operations	(2) + 2 * (3) + 4 * (4) + 8 * (5) + 16 * (6)
Number of instructions executed	(1)
Number of operations	(1) + (3) + 3 * (4) + 7 * (5) + 15 * (6)

Moreover, the value that depends on hardware is used as for the following value.

- Frequency in CPU core (GHz)
- MFLOPS peak value for core units (MFLOPS) = Frequency in CPU core (GHz) * Number of FMA operations * Number of floating point functional units * Number of operand processing of SIMD instruction * 1000
- MIPS peak value for core units (MIPS) = Frequency in CPU core (GHz) * Number of instructions that can be issued in 1 cycle * 1000

8.3.2 "Instructions_SIMD" Specified for Category Name

When Instructions_SIMD is effective as a category of the Hardware Monitor Information, information based on the following calculation expressions is output.

Table 8.8 Hardware Monitor Information calculation expressions (Category name : Instructions_SIMD)

Information	Calculation expression
Number of instructions executed	Number of instructions executed
MIPS	Number of instructions executed/1000000 / Elapsed time
MIPS peak performance ratio (%)	MIPS / (Number of executed cores * MIPS peak value for core units) * 100
Load/store instruction ratio (SIMD instruction ratio) (%)	Number of SIMD load and SIMD store instructions / Number of instructions executed * 100
Floating point operation instruction ratio (SIMD instruction ratio) (%)	Number of floating point operation SIMD instructions / Number of instructions executed * 100

Information	Calculation expression
Floating point sum operation instruction ratio (SIMD instruction ratio) (%)	Number of floating point sum operation SIMD instructions / Number of instructions executed * 100
Fixed point operation instruction ratio (SIMD instruction ratio) (%)	Number of fixed point operation SIMD instructions / Number of instructions executed * 100
SIMD instruction ratio (%)	Number of SIMD instructions / Number of instructions executed * 100
SIMD load/store instruction ratio (%)	Number of SIMD load and SIMD store instructions / Number of load and store instructions executed * 100
Number of pre-fetch instructions	Number of pre-fetch instructions
Number of indirect pre-fetch instructions	Number of indirect pre-fetch instructions

Information on the entire process is calculated by the following calculation expressions.

Table 8.9 Hardware Monitor Information on the entire process calculation expressions (Instructions_SIMD)

Information	Calculation expression
Number of instructions executed	Total number of instructions executed in all threads
MIPS	Total number of instructions executed in all threads / 1000000 / Elapsed time of a thread having the longest elapsed time
MIPS peak performance ratio (%)	MIPS in entire process / (Number of executed cores * MIPS peak value for core units) * 100
Load/store instruction ratio (SIMD instruction ratio) (%)	Total number of SIMD load and SIMD store instructions in all threads / Total number of instructions executed in all threads * 100
Floating point operation instruction ratio (SIMD instruction ratio) (%)	Total number of floating point operation SIMD instructions in all threads / Total number of instructions executed in all threads * 100
Floating point sum operation instruction ratio (SIMD instruction ratio) (%)	Total number of floating point sum operation SIMD instructions in all threads / Total number of instructions executed in all threads * 100
Fixed point operation instruction ratio (SIMD instruction ratio) (%)	Total number of fixed point operation SIMD instructions in all threads / Total number of instructions executed in all threads * 100
SIMD instruction ratio (%)	Total number of SIMD instructions in all threads / Total number of instructions executed in all threads * 100
SIMD load/store instruction ratio (%)	Total number of SIMD load and SIMD store instructions in all threads / Total number of load and store instructions executed in all threads * 100
Number of pre-fetch instructions	Total number of pre-fetch instructions in all threads
Number of indirect pre-fetch instructions	Total number of indirect pre-fetch instructions in all threads

Note

The value of PA event acquired in the calculation expression mentioned above includes cost waiting barrier of the parallel region.

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

- | | |
|--|-----------------------------------|
| (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 |

Number of instructions executed

(1)

Number of SIMD load and SIMD store instructions	(2)
Number of floating point operation SIMD instructions	(3)
Number of floating point sum operation SIMD instructions	(4)
Number of fixed point operation SIMD instructions	(5)
Number of SIMD instructions	(2) + (3) + (4) + (5)
Number of load and store instructions	(8) + (2)
Number of pre-fetch instructions	(6) + (7)
Number of indirect pre-fetch instructions	(7)

Moreover, the value that depends on hardware is used as for the following value.

- Frequency in CPU core (GHz)
- MIPS peak value for core units (MIPS) = Frequency in CPU core (GHz) * Number of instructions that can be issued in 1 cycle * 1000

8.3.3 "Instructions_NOSIMD" Specified for Category Name

When Instructions_NOSIMD is effective as a category of the Hardware Monitor Information, information based on the following calculation expressions is output.

Table 8.10 Hardware Monitor Information calculation expressions (Category name : Instructions_NOSIMD)

Information	Calculation expression
Number of instructions executed	Number of instructions executed
MIPS	Number of instructions executed/1000000 / Elapsed time
MIPS peak performance ratio (%)	MIPS / (Number of executed cores * MIPS peak value for core units) * 100
Load/store instruction ratio (NOSIMD instruction ratio) (%)	Number of NOSIMD load and NOSIMD store instructions / Number of instructions executed * 100
Floating point operation instruction ratio(NOSIMD instruction ratio) (%)	Number of floating point operation NOSIMD instructions / Number of instructions executed * 100
Floating point sum operation instruction ratio(NOSIMD instruction ratio) (%)	Number of floating point sum operation NOSIMD instructions / Number of instructions executed * 100
Fixed point operation instruction ratio(NOSIMD instruction ratio) (%)	Number of fixed point operation NOSIMD instructions / Number of instructions executed * 100
NOSIMD instruction ratio (%)	Number of NOSIMD instructions / Number of instructions executed * 100
NOSIMD load/store instruction ratio (%)	Number of NOSIMD load and NOSIMD store instructions / Number of load and store instructions executed * 100
Number of pre-fetch instructions	Number of pre-fetch instructions
Number of indirect pre-fetch instructions	Number of indirect pre-fetch instructions

Information on the entire process is calculated by the following calculation expressions.

Table 8.11 Hardware Monitor Information on the entire process calculation expressions (Instructions_NOSIMD)

Information	Calculation expression
Number of instructions executed	Total number of instructions executed in all threads
MIPS	Total number of instructions executed in all threads / 1000000 / Elapsed time of a thread having the longest elapsed time
MIPS peak performance ratio (%)	MIPS in entire process / (Number of executed cores * MIPS peak value for core units) * 100
Load/store instruction ratio (NOSIMD instruction ratio) (%)	Total number of NOSIMD load and NOSIMD store instructions in all threads / Total number of instructions executed in all threads * 100

Information	Calculation expression
Floating point operation instruction ratio (NOSIMD instruction ratio) (%)	Total number of floating point operation NOSIMD instructions in all threads / Total number of instructions executed in all threads * 100
Floating point sum operation instruction ratio (NOSIMD instruction ratio) (%)	Total number of floating point sum operation NOSIMD instructions in all threads / Total number of instructions executed in all threads * 100
Fixed point operation instruction ratio (NOSIMD instruction ratio) (%)	Total number of fixed point operation NOSIMD instructions in all threads / Total number of instructions executed in all threads * 100
NOSIMD instruction ratio (%)	Total number of NOSIMD instructions in all threads / Total number of instructions executed in all threads * 100
NOSIMD load/store instruction ratio (%)	Total number of NOSIMD load and NOSIMD store instructions in all threads / Total number of load and store instructions executed in all threads * 100
Number of pre-fetch instructions	Total number of pre-fetch instructions in all threads
Number of indirect pre-fetch instructions	Total number of indirect pre-fetch instructions in all threads

Note

The value of PA event acquired in the calculation expression mentioned above includes cost waiting barrier of the parallel region.

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

- | | |
|--|-----------------------------------|
| (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 |

Number of instructions executed	(1)
Number of NOSIMD load and NOSIMD store instructions	(2)
Number of floating point operation NOSIMD instructions	(3)
Number of floating point sum operation NOSIMD instructions	(4)
Number of fixed point operation NOSIMD instructions	(5)
Number of NOSIMD instructions	(2) + (3) + (4) + (5)
Number of load and store instructions	(2) + (8)
Number of pre-fetch instructions	(6) + (7)
Number of indirect pre-fetch instructions	(7)

Moreover, the value that depends on hardware is used as for the following value.

- Frequency in CPU core (GHz)
- MIPS peak value for core units (MIPS) = Frequency in CPU core (GHz) * Number of instructions that can be issued in 1 cycle * 1000

8.3.4 "Cache" Specified for Category Name

When Cache is effective as a category of the Hardware Monitor Information, information based on the following calculation expressions is output.

Table 8.12 Hardware Monitor Information calculation expressions (Category name : Cache)

Information	Calculation expression
Number of instructions executed	Number of instructions executed

Information	Calculation expression
Level 1 instruction cache mistake ratio (%)	Level 1 instruction cache mistake count / Number of instructions * 100
Level 1 data cache mistake ratio (%)	Level 1 data cache mistake count / Number of memory access instructions * 100
Level 2 cache mistake ratio (%)	Level 2 cache mistake count / Number of memory access instructions * 100
Proportion of demand mistakes that account for Level 2 cache mistakes (%)	Demand request Level 2 cache mistake count / Level 2 cache mistake count * 100
Proportion of pre-fetch mistakes that account for Level 2 cache mistakes (%)	Pre-fetch request Level 2 cache mistake count / Level 2 cache mistake count * 100

Information on the entire process is calculated by the following calculation expressions.

Table 8.13 Hardware Monitor Information on the entire process calculation expressions (Cache)

Information	Calculation expression
Number of instructions executed	Total number of instructions executed in all threads
Level 1 instruction cache mistake ratio (%)	Total Level 1 instruction cache mistake count in all threads / Total number of instructions in all threads * 100
Level 1 data cache mistake ratio (%)	Total Level 1 data cache mistake count in all threads / Total number of memory access instructions in all threads * 100
Level 2 cache mistake ratio (%)	Total Level 2 cache mistake count in all threads / Total number of memory access instructions in all threads * 100
Proportion of demand mistakes that account for Level 2 cache mistakes (%)	Total demand request Level 2 cache mistake count in all threads / Total Level 2 cache mistake count in all threads * 100
Proportion of pre-fetch mistakes that account for Level 2 cache mistakes (%)	Total pre-fetch request Level 2 cache mistake count in all threads / Total Level 2 cache mistake count in all threads * 100

Note

The value of PA event acquired in the calculation expression mentioned above includes cost waiting barrier of the parallel region.

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

- | | |
|----------------------------------|-----------------------------------|
| (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 |

Number of instructions executed	(1)
Number of memory access instructions	(2) + 2 * (3) + 4 * (4)
Level 1 instruction cache mistake count	(5)
Level 1 data cache mistake count	(6)
Level 2 cache mistake count	(7) + (8)
Demand request Level 2 cache mistake count	(7)
Pre-fetch request Level 2 cache mistake count	(8)

8.3.5 "MEM_access" Specified for Category Name

When MEM_access is effective as a category of the Hardware Monitor Information, information based on the following calculation expressions is output.

Table 8.14 Hardware Monitor Information calculation expressions (Category name : MEM_access)

Information	Calculation expression
Load/store instruction ratio (SIMD instruction ratio) (%)	Number of SIMD load and SIMD store instructions / Number of instructions executed * 100
Load/store instruction ratio (NOSIMD instruction ratio) (%)	Number of NOSIMD load and NOSIMD store instructions / Number of instructions executed * 100
Memory access throughput	Quantity of data transferred between memory and CPU / Elapsed time

Information on the entire process is calculated by the following calculation expressions.

Table 8.15 Hardware Monitor Information on the entire process calculation expressions (MEM_access)

Information	Calculation expression
Load/store instruction ratio (SIMD instruction ratio) (%)	Total number of SIMD load and SIMD store instructions in all threads / Total number of instructions executed in all threads * 100
Load/store instruction ratio (NOSIMD instruction ratio) (%)	Total number of NOSIMD load and NOSIMD store instructions in all threads / Total number of instructions executed in all threads * 100
Memory access throughput	Total quantity of data transferred between memory and CPU in all threads / Elapsed time of a thread having the longest elapsed time



Note

The value of PA event acquired in the calculation expression mentioned above includes cost waiting barrier of the parallel region.

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

- (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 instructions executed (1)
- Number of SIMD load and SIMD store instructions (2)
- Number of NOSIMD load and NOSIMD store instructions (3)
- Quantity of data transferred between memory and CPU ((4) + (5) + (6) + (7)) * 256 / 1000(GB)

8.3.6 "Performance" Specified for Category Name

When Performance is effective as a category of the Hardware Monitor Information, information based on the following calculation expressions is output.

Table 8.16 Hardware Monitor Information calculation expressions (Category name : Performance)

Information	Calculation expression
2-4 instruction commit	{ Number of CPU cycles - (Number of cycles in which the number of completion instructions is 0 + Number of cycles in which the number of completion instructions is 1) } / Frequency in CPU core
1 instruction commit	Number of cycles in which the number of completion instructions is 1 / Frequency in CPU core
Operation wait	Number of cycles in which the number of completion instructions is 0 in execution of operations / Frequency in CPU core

Information	Calculation expression
Cache access wait	(Number of cycles in which the number of completion instructions is 0 due to a data wait caused by memory access - Number of cycles with a wait caused by an L2 cache mistake) / Frequency in CPU core
Memory access wait	(Number of cycles with a wait caused by an L2 cache mistake + Number of cycles in which the number of completion instructions is 0 due to CSE being empty and store port being full) / Frequency in CPU core
Instruction fetch wait	(Number of cycles in which the number of completion instructions is 0 due to CSE being empty - Number of cycles in which the number of completion instructions is 0 due to CSE being empty and store port being full) / Frequency in CPU core
Other waits	{ Number of cycles in which the number of completion instructions is 0 - (Operation wait + Cache access wait + Memory access wait + Instruction fetch wait) } / Frequency in CPU core

Information on the entire process is calculated by the following calculation expressions.

Table 8.17 Hardware Monitor Information on the entire process calculation expressions (Category name : Performance)

Information	Calculation expression
2-4 instruction commit	{ Total number of CPU cycles in all threads - (Total number of cycles in which the number of completion instructions is 0 in all threads + Total number of cycles in which the number of completion instructions is 1 in all threads) } / Frequency in CPU core
1 instruction commit	Total number of cycles in which the number of completion instructions is 1 in all threads / Frequency in CPU core
Operation wait	Total number of cycles in which the number of completion instructions is 0 in execution of operations in all threads / Frequency in CPU core
Cache access wait	(Total number of cycles in which the number of completion instructions is 0 due to a data wait caused by memory access in all threads - Total number of cycles with waits due to L2 cache mistakes in all threads) / Frequency in CPU core
Memory access wait	(Total number of cycles with waits due to L2 cache mistakes in all threads + Total number of cycles in which the number of completion instructions is 0 due to CSE being empty and store port being full in all threads) / Frequency in CPU core
Instruction fetch wait	(Total number of cycles in which the number of completion instructions is 0 due to CSE being empty in all threads - Total number of cycles in which the number of completion instructions is 0 due to CSE being empty and store port being full in all threads) / Frequency in CPU core
Other waits	{Total number of cycles in which the number of completion instructions is 0 in all threads - (Total number of operation waits in all threads + Total number of cache access waits in all threads + Total number of memory access waits in all threads + Total number of instruction fetch waits in all threads) } / Frequency in CPU core

Note

The value of PA event acquired in the calculation expression mentioned above includes cost waiting barrier of the parallel region.

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 |

- Number of CPU cycles (1)
- Number of cycles in which the number of completion instructions is 0 (2)
- Number of cycles in which the number of completion instructions is 1 (3)
- Number of cycles in which the number of completion instructions is 0 in execution of operations (4)
- Number of cycles in which the number of completion instructions is 0 due to a data wait caused by memory access (5)
- Number of cycles with a wait caused by an L2 cache mistake (6)
- Number of cycles in which the number of completion instructions is 0 due to CSE being empty (7)
- Number of cycles in which the number of completion instructions is 0 due to CSE being empty and store port being full (8)

Moreover, the value that depends on hardware is used as for the following value.

- Frequency in CPU core(GHz)

8.3.7 "TLB" Specified for Category Name

When TLB is effective as a category of the Hardware Monitor Information, information based on the following calculation expressions is output.

Table 8.18 Hardware Monitor Information calculation expressions (Category name : TLB)

Information	Calculation expression
Number of instructions executed	Number of instructions executed
Micro data TLB mistake ratio (%)	Micro data TLB mistake count / Number of memory access instructions * 100
Data main TLB mistake ratio (%)	Data main TLB mistake count / Number of memory access instructions * 100

Information on the entire process is calculated by the following calculation expressions.

Table 8.19 Hardware Monitor Information on the entire process calculation expressions (TLB)

Information	Calculation expression
Number of instructions executed	Total number of instructions executed in all threads
Micro data TLB mistake ratio (%)	Total micro data TLB mistake count in all threads / Total number of memory access instructions in all threads * 100
Data main TLB mistake ratio (%)	Total data main TLB mistake count in all threads / Total number of memory access instructions in all threads * 100

Note

The value of PA event acquired in the calculation expression mentioned above includes cost waiting barrier of the parallel region.

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

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

- Number of instructions executed (1)
- Number of memory access instructions (2) + 2 * (3) + 4 * (4)
- Micro data TLB mistake count (5)
- Data main TLB mistake count (6)

Figure 8.2 Output example of Hardware Monitor Information (environmental variable FLIB_RTINFO_CSV is specified)

```

=====
HARDWARE MONITOR INFORMATION
=====
Type, Process ID, Thread ID, Range, Time(sec), MFLOPS, MFLOPS/PEAK(%), MIPS, MIPS/PEAK(%), Float/operate(%)
[PA], 0, 0, "_PROGRAM_", 2.0962, 110, 0.6858, 1175, 14.6819, 8.5603
[PA], 0, 1, "_PROGRAM_", 2.0957, 110, 0.6860, 1156, 14.4449, 8.6687
[PA], 0, total, "_PROGRAM_", 2.0962, 219, 0.6858, 2330, 14.5617, 8.6240

Routine : Count max = 1000
Type, Process ID, Thread ID, Range, Time(sec), MFLOPS, MFLOPS/PEAK(%), MIPS, MIPS/PEAK(%), Float/operate(%), Start, End, Count, Routine name
[PA_Routine], 0, total, "_PROGRAM_", 1.6027, 250, 0.7800, 2367, 14.7955, 9.5529, 40, 49, 10, "sub1"
[PA_Routine], 0, total, "_PROGRAM_", 0.1588, 0, 0.0000, 2190, 13.6876, 0.0000, 51, 56, 2000*, "sub2"
[PA_Routine], 0, total, "_PROGRAM_", 1.7615, 227, 0.7097, 2351, 14.6956, 8.8052, -, -, -, -

```

Chapter 9 Message

This chapter describes the messages output during execution.

The message is output to the standard error output file.

RTINF0001

The value of environment variable FLIB_RTINFO_PA is invalid. The default value (Statistics) is adopted.

[Message Explanation]

Environment variable FLIB_RTINFO_PA operand is invalid.

[System behavior]

Continues the processing, assuming that the value of environment variable FLIB_RTINFO_PA is Statistics.

[Programmer response]

Correct the value of environment variable.

RTINF0003

The value of environment variable FLIB_RTINFO_PROCESS is invalid. The default value (txt) is adopted.

[Message Explanation]

Environment variable The FLIB_RTINFO_PROCESS operand is invalid.

[System behavior]

Continues the processing, assuming that the value of environment variable FLIB_RTINFO_PROCESS is txt.

[Programmer response]

Correct the value of environment variable.

RTINF0004

The value of environment variable FLIB_RTINFO_THRESHOLD is invalid. The environment variable FLIB_RTINFO_THRESHOLD is ignored.

[Message Explanation]

The FLIB_RTINFO_THRESHOLD operand is invalid.

[System behavior]

Ignores the environment variable FLIB_RTINFO_THRESHOLD.

[Programmer response]

Specify 1 to 2147483648 for the environment variable.

RTINF0005

The value of environment variable FLIB_RTINFO_PA_MODE is invalid. The default value (a) is adopted.

[Message Explanation]

The FLIB_RTINFO_PA_MODE operand is invalid.

[System behavior]

Continues the processing, assuming that the value of environment variable FLIB_RTINFO_PA_MODE is a.

[Programmer response]

Correct the value of environment variable.

RTINF0009

The value of environment variable FLIB_RTINFO_RANKID is invalid. The environment variable FLIB_RTINFO_RANKID is ignored.

[Message Explanation]

The FLIB_RTINFO_RANKID operand is invalid.

[System behavior]

Ignores the environment variable FLIB_RTINFO_RANKID.

[Programmer response]

Specify 0 to 2147483648 for the environment variable.

RTINF0010

The value of environment variable FLIB_RTINFO_RANKID_MAX is invalid. The environment variable FLIB_RTINFO_RANKID_MAX is ignored.

[Message Explanation]

The FLIB_RTINFO_RANKID_MAX operand is invalid.

[System behavior]

Ignores the environment variable FLIB_RTINFO_RANKID_MAX.

[Programmer response]

Specify 0 to 2147483648 for the environment variable.

RTINF0011

The value of environment variable FLIB_RTINFO_RANKID_MIN is invalid. The environment variable FLIB_RTINFO_RANKID_MIN is ignored.

[Message Explanation]

The FLIB_RTINFO_RANKID_MIN operand is invalid.

[System behavior]

Ignores the environment variable FLIB_RTINFO_RANKID_MIN.

[Programmer response]

Specify 0 to 2147483648 for the environment variable.

RTINF0012

The value of environment variable FLIB_RTINFO_RANKID_MIN is larger than the value of environment variable FLIB_RTINFO_RANKID_MAX. The environment variable FLIB_RTINFO_RANKID_MIN is ignored.

[Message Explanation]

The specified value for environment variable FLIB_RTINFO_RANKID_MIN is bigger than for environment variable FLIB_RTINFO_RANKID_MAX.

[System behavior]

Ignores the environment variable FLIB_RTINFO_RANKID_MIN.

[Programmer response]

Specify the value for the environment variable FLIB_RTINFO_RANKID_MIN smaller than for FLIB_RTINFO_RANKID_MAX

RTINF0013

The value of environment variable FLIB_RTINFO_COSTLIMIT is invalid. The default value (10) is adopted.

[Message Explanation]

The FLIB_RTINFO_COSTLIMIT operand is invalid.

[System behavior]

Continues the processing, assuming that the value of environment variable FLIB_RTINFO_PA_MODE is 10.

[Programmer response]

Specify 0 to 2147483648 for the environment variable.

RTINF0014

The value of environment variable FLIB_RTINFO_FUNCMAX_PA is invalid. The default value (1000) is adopted.

[Message Explanation]

The FLIB_RTINFO_FUNCMAX_PA operand is invalid.

[System behavior]

Continues the processing, assuming that the value of environment variable FLIB_RTINFO_FUNCMAX_PA is 1000.

[Programmer response]

Specify 0 to 2147483648 for the environment variable.

RTINF0020

The value of environment variable FLIB_RTINFO_IO is invalid. The default value (A) is adopted.

[Message Explanation]

The FLIB_RTINFO_IO operand is invalid.

[System behavior]

Continues the processing, assuming that the value of environment variable FLIB_RTINFO_IO is A.

[Programmer response]

Correct the value of environment variable.

RTINF0101

Failed to open the file (*filename*) specified in the environment variable FLIB_RTINFO_CSV. The information is written to standard output.

[Message Explanation]

The FLIB_RTINFO_CSV output file failed to open.

[Parameters Explanation]

filename: file name

[System behavior]

Ignores the environment variable FLIB_RTINFO_CSV.

[Programmer response]

Confirm the state of the file.

RTINF0102

Failed to open the file (*filename*) specified in the environment variable FLIB_RTINFO_PROCESS. The information is written to standard output.

[Message Explanation]

The FLIB_RTINFO_PROCESS output file failed to open.

[Parameters Explanation]

filename: file name

[System behavior]

Ignores the environment variable FLIB_RTINFO_PROCESS.

[Programmer response]

Confirm the state of the file.

RTINF0201

The start_collection("rangename") is already being measured.

[Message Explanation]

After start_collection, another start_collection with the same argument value specified was executed.

[Parameters Explanation]

rangename: range name

[System behavior]

Ignores the service routine specified.

[Programmer response]

Specified the service routine correctly.

RTINF0202

The stop_collection("rangename") is invalid.

[Message Explanation]

Stop_collection has no corresponding start_collection.

[Parameters Explanation]

rangename: range name

[System behavior]

Ignores the service routine specified.

[Programmer response]

Insert the service routine correctly.

RTINF0203

The stop_collection("rangename") is not found.

[Message Explanation]

start_collection has no corresponding stop_collection

[Parameters Explanation]

rangename: range name

[System behavior]

The measurement is stopped.

[Programmer response]

Review the specification of the service routine.

RTINF0301

A part of value is not acquired, because hardware cannot be distinguished.

[Message Explanation]

The value that depends on hardware cannot acquire, because hardware was not identified

[System behavior]

Continues the processing, assuming that the value depends on hardware is 0.

RTINF0401

Failed to stop ExtCpuStat.

[Message Explanation]

It failed in the end of measurement ExtCpuStat.

[System behavior]

Processing is interrupted.

RTINF1xxx

Internal error.

[Message Explanation]

Internal module return value error, internal information conflict.

[Parameters Explanation]

xxx: Error number unique in each error part

[System behavior]

Processing is interrupted.

RTINF2xxx

Internal error. PAPI return code = xxx.

[Message Explanation]

The return value of the call of the PAPI (Performance Application Programming Interface) library is abnormal.

[Parameters Explanation]

xxx: Error number unique in each error part

[System behavior]

Processing is interrupted.

RTINF4xxx

Allocation error.

[Message Explanation]

It failed in the acquisition of the work area.

[Parameters Explanation]

xxx: Error number unique in each error part

[System behavior]

Processing is interrupted.

RTINF5001***syscall* : XXXX.****[Message Explanation]**

The error occurred by *syscall*. The message is XXXX.

[Parameters Explanation]

syscall: System call name

XXXX: Error message returned from system call

[System behavior]

Processing is continued.

[Programmer response]

Correct the program so that an error does not occur.

Appendix A Attention in The Use

- When FALSE is specified for environment variable FLIB_FASTOMP, information cannot be correctly fetched. Refer to following manuals for environment variable FLIB_FASTOMP.
 - "Fortran User's Guide"
 - "C User's Guide"
 - "C++ User's Guide"
- If the information is fetched each routine or loop by specifying -Nrt_tune_func or -Nrt_tune_loop, fetched information is increases, and execute time may increase. In that case, it is recommend to not specify -Nrt_tune_func and -Nrt_tune_loop and restrict measurement range by specifying range.
- When there are goto statement, stop statement (Fortran), setjmp()/longjmp() (C/C++), throw statement (C++) during a program or an exceptions is checked, information may be not correctly fetched.
- There is a possibility of receiving the following influences by optimization of the compiler.
 - When fetched the information for each loop, it may be different which loop is worked as a target of the measurement.
 - Line number may not accord with line number of the source program.
 - The nest relations of the loop may not accord with a nest of the source program. By the following optimization, I may not recognize loop information definitely. By the following optimization, it may not be correctly recognized loop information.
 - inline expansion for procedures
 - loop unrolling
 - loop blocking
 - loop interchange
 - loop fusion
 - loop striping
 - loop splitting
 - loop peeling
 - loop unswitching
 - change the function to a multi-operation function, use SIMD instructions
- When either of Parallelization Information, Cost Information or Input-Output Information is output, Hardware Monitor Information cannot be output.
- There are the following notes about Fortran Input-Output Information.
 - If a nested range is specified, information for that range is not output.
 - If overlapping ranges are specified, information is output for the range from the earlier specified START_COLLECTION to the corresponding STOP_COLLECTION.
- There are the following notes about C/C++ Input-Output Information.
 - If the source program was compiled without compiler option Nrt_tune_io, C/C++ Input-Output Information is not output.
 - When I/O information on C/C++ is acquired specifying Nrt_tune_io, the function prototype should be declared to the service function in a correct type (prototype declaration). When it doesn't become a measurement object when not declared in a correct type, I/O information is not output. Please do an appropriate standard header in include to declare the function prototype in a correct type.
 - If close(2) corresponding to open(2) or open64(2) is not executed, Input-Output Information on the file that opens with previous open(2) or open64(2) is not output.

- Please note the acquisition value of Hardware Monitor Information as follows.
 - The value of PA event acquired includes cost waiting barrier of the parallel region.

Appendix B Notes on Migration from FX10 system to FX100 system

This appendix describes modification as notes on migrating from the FX10 system to the FX100 system.

B.1 Change in Category of Hardware Monitor Information

a. Changes

Category names which can be specified for the Hardware Monitor Information and the output information of each category are changed.

Refer to "[Chapter 8 Hardware Monitor Information](#)" for the output information of each category.

[Previous version]

The following category names were able to be specified for environment variable FLIB_RTINFO_PA:

- Statistics
- Instructions
- Cache
- MEM_access
- Performance

[This version]

The following category names can be specified for environment variable FLIB_RTINFO_PA:

- Statistics (Output information is changed)
- Instructions_SIMD (It separates from Instructions to two categories, and output information is changed)
- Instructions_NOSIMD (It separates from Instructions to two categories, and output information is changed)
- Cache (Output information is changed)
- MEM_access (Output information is changed)
- Performance (Output information is changed)
- TLB (Category is added)

b. Influence

Category names which can be specified for the environment variable FLIB_RTINFO_PA and the output information of each category are changed. When the invalid category name is specified, runtime message RTINF0001 is output and the information fetched is the same as when Statistics is specified.

c. Coping

Please change to the category that contains information to acquire the category name specified for environment variable FLIB_RTINFO_PA.