Fujitsu Software
Technical Computing Suite V4.0L20

LLIO User's Guide

J2UL-2555-01ENZ0(08)
March 2023

Preface

Purpose of This Manual

This manual describes LLIO (Lightweight Layered IO-Accelerator) installation, operation management, and how to run jobs using LLIO. LLIO is included in Technical Computing Suite V4.0L20.

For details on FEFS, see the *FEFS User's Guide*.

For details on the Job Operation Software, see the following manuals:

- *Job Operation Software Overview*
- *Job Operation Software Administrator's Guide for Job Management*
- *Job Operation Software End-user's Guide*

Intended Readers

This manual is intended for the following readers:

- System administrators who install LLIO and manage operations
- End users who run jobs using LLIO

The manual assumes readers have the following knowledge:

- Basic Linux knowledge
- Knowledge about storage in general
- Overall knowledge of the Job Operation Software. obtained from the *Job Operation Software Overview*

Organization of This Manual

This manual is organized as follows. All chapters are intended for the system administrators. Chapters 1 and 2 are intended for the end users.

Chapter 1 LLIO Overview

This chapter provides an overview of LLIO and describes the LLIO configuration.

Chapter 2 LLIO Functions

This chapter describes LLIO functions.

Chapter 3 System Operation

This chapter describes LLIO operation.

Appendix A Reference

This appendix is the reference manual for the LLIO system calls, commands, and library.

Appendix B Messages

This appendix describes the messages output by LLIO.

Appendix C Statistical Information Output Items

This appendix describes statistical information output items.

Glossary

The glossary describes the main terms used with LLIO.

Notation Used in This Manual

Representation of Units

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

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

Model name notation

In this manual, the computer that based on Fujitsu A64FX CPU is abbreviated as "FX server", and FUJITSU server PRIMERGY as "PRIMERGY server" (or simply "PRIMERGY").

Prompts in Command Input Examples

The manual distinguishes prompts by administrator privileges required for command operations.

- # means that the command is executed with administrator privileges (superuser).
- $ means that the command is executed with privileges other than administrator privileges.

Symbols in This Manual

This manual uses the following symbols.

Note

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

See

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

Information

The Information symbol indicates a reference note related to LLIO.

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

- Lustre is a registered trademark of Seagate Technology LLC in the United States.
- Linux(R) is a registered trademark of Linus Torvalds in the U.S. and other countries.
- Red Hat and Red Hat Enterprise Linux are trademarks of Red Hat, Inc. registered in the U.S. and other countries.
- Intel is a trademark of Intel Corporation and its subsidiaries in the U.S. and other countries.
- Other company and product names in this manual are trademarks or registered trademarks of their respective owners.

Date of Publication and Version

Version	Manual Code
March 2023, Version 1.8	J2UL-2555-01ENZ0(08)
September 2022, Version 1.7	J2UL-2555-01ENZ0(07)

Version	Manual Code
March 2022, Version 1.6	J2UL-2555-01ENZ0(06)
November 2021, Version 1.5	J2UL-2555-01ENZ0(05)
August 2021, Version 1.4	J2UL-2555-01ENZ0(04)
December 2020, Version 1.3	J2UL-2555-01ENZ0(03)
September 2020, Version 1.2	J2UL-2555-01ENZ0(02)
June 2020, Version 1.1	J2UL-2555-01ENZ0(01)
February 2020, First version	J2UL-2555-01ENZ0(00)

Copyright

Update History

Changes	Location	Version
Added --sync option for the llio_transfer command.	A.2.3	1.8
Also fixed errata.	-	
Added a note of the cache area of second-layer storage.	2.1.1	1.7
Changed the description of the llio_transfer command return value.	A.2.3	
Added a specific example to explain when the cache in the cache area of the second-layer storage is deleted.	2.1.1	1.6
Added description of selecting LLIO area.	2.2.1	1.5
Added notes on collecting LLIO performance information.	2.7.2	
Added a note on the number of files that can be opened or written per job for three areas of first-layer storage.	2.1	1.4
Corrected the description of the node temporary area.	2.1.3	
Corrected the description of files and directories in the shared temporary area and cache area of second-layer storage remaining in the second-layer storage.	3.3.4 3.3.4.1 3.3.4.2	1.3
Corrected the description of the output item "Uncompleted-file" in LLIO performance information and added a note.	C.1	
Corrected the description of the shared temporary area.	2.1	1.2
Corrected the read and write size units with Direct I/O for first-layer storage to 64KB.	2.1	
Added a note about failure to write file data when asynchronous close function is enabled.	2.4.4	
Added a note for addressing the specification of the metadata consistency between client nodes.	3.4	
Corrected notes for the llio_transfer command.	A.2.3	
Improved description of consistency guarantees between client nodes.	2.1.1 2.1.2 2.1.3	1.1
Added description about the size consumed for file management in the shared temporary area and the node temporary area.	2.1.2 2.1.3	
Added description that the directory under the trash directory can be deleted even during job operation.	3.3.4.2	
The note for applying OS update packages are no longer needed and have been deleted.	3.4	

Changes	Location	Version
Improved description of LLIO performance information.	C.1	
Added message 6458 of the llio_transfer command.	B.2.3	

Contents

Chapter 1 LLIO Overview

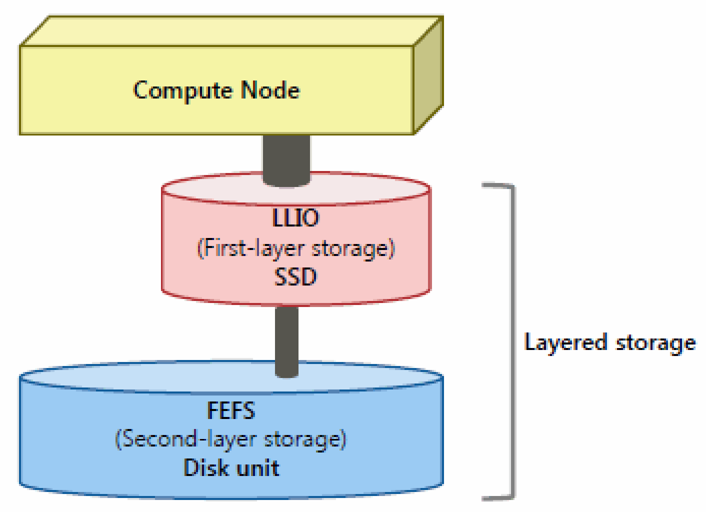
This chapter provides an overview of LLIO (Lightweight Layered IO-Accelerator). The overview is intended for system administrators and end users.

1.1 Features

Located between the FEFS parallel distributed file system and compute nodes, LLIO is a high-performance file system using high-speed flash memory. Higher speeds are realized by not writing temporary files for jobs to FEFS or by asynchronously writing from LLIO to FEFS during compute processing.

The combination of FEFS and LLIO takes advantage of their characteristics to make high-speed, large-capacity layered storage a reality. When viewed from compute nodes, LLIO at the higher layer is referred to as first-layer storage, and FEFS at the lower layer as second-layer storage. The combination of these two layers is called layered storage.

Figure 1.1 Layered Storage



Information

No modification of applications is necessary because LLIO supports the standard POSIX interface in Linux.

LLIO uses the following functions to implement a high-performance file system. Users can specify these functions when submitting a job.

First-Layer Storage Areas Suitably Organized for Jobs

First-layer storage has three areas, each with different characteristics. End users can specify suitable sizes for the respective areas when submitting jobs.

See

For details on the first-layer storage areas, see "2.1 Three Areas of First-Layer Storage."

Striping Function

The striping function is provided to distribute a single file over multiple storage I/O nodes and transfer it concurrently.

See

For details on the striping function, see "2.4.2 Striping Function for First-Layer Storage" and "2.4.3 Striping Function for Second-Layer Storage."

Common File Distribution Function

The common file distribution function is provided to implement load balancing through replication of files (a.out, input files, etc.) on second-layer storage, where the load is assumed to be concentrated on during job execution. The replicated files are created on first-layer storage.
As the load is assumed to be concentrated on second-layer storage, creating files from there to first-layer storage will distribute access from compute nodes to storage I/O nodes. You can deploy the function by writing a dedicated command at the beginning of a job script so that this processing begins before the start of an application.

See

For details on the common file distribution function, see "2.3 Common File Distribution Function."

Compute Node Cache Function

The compute node cache function provides a page cache using the free memory on compute nodes not used by applications.
Jobs that do not use much of the memory for compute node users are jobs executed with the free memory at hand. By allocating as much free memory as possible to the page cache, you can bring out better performance when writing with a smaller I/O size or when even reading the same file repeatedly. Using free memory as the page cache enables I/O processing from a job within a compute node, which enables higher-speed I/O. The compute node cache must be specified and take into account the area used by applications since the cache is allocated from the user memory area.

See

For details on the compute node cache function, see "2.5 Compute Node Cache Function."

Also, the following features are implemented as functions supporting the use of LLIO. Users can specify these functions when submitting a job.

Statistical Information

LLIO collects a lot of statistical information and provides it to the end users and system administrators who execute jobs. The system administrators and end users can browse this information, which is useful for I/O tuning and system trouble investigation.

See

- -For details on statistical information for end users, see "2.7 Statistical Information."

- -For details on statistical information for system administrators, see "2.8.2 System Statistical Information."

Job Handling When a Node Runs Abnormally

The Job Operation Software detects abnormalities that occur in compute nodes and storage I/O nodes while executing jobs, and isolates the abnormal nodes from job operation. The jobs running on these nodes at this time are abnormally terminated, and the end users are notified of any files that have not been written to second-layer storage.

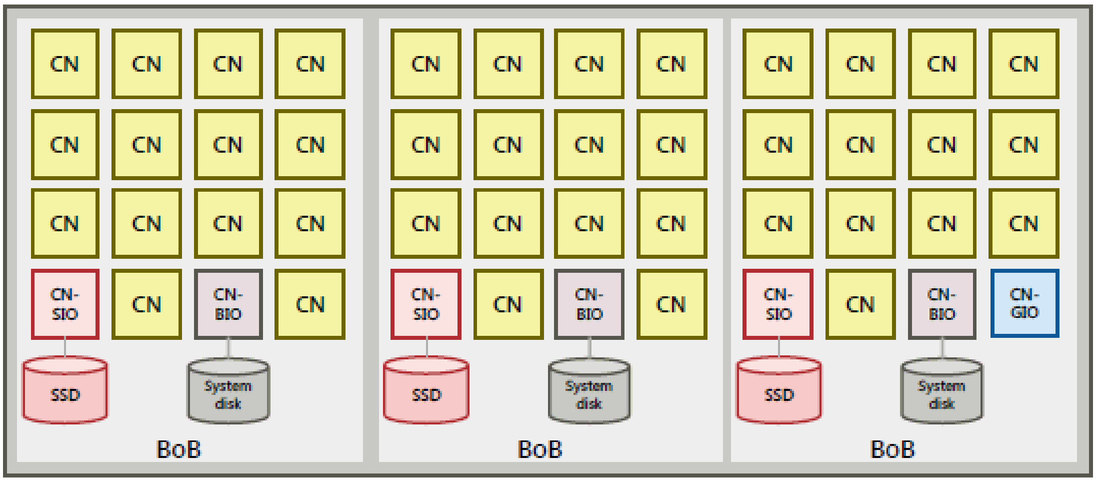
See

The function that notifies end users about files not written to second-layer storage is called the unwritten file list acquisition function.
For details on the unwritten file list acquisition function, see "2.6 Unwritten File List Acquisition Function."

1.2 System Configuration

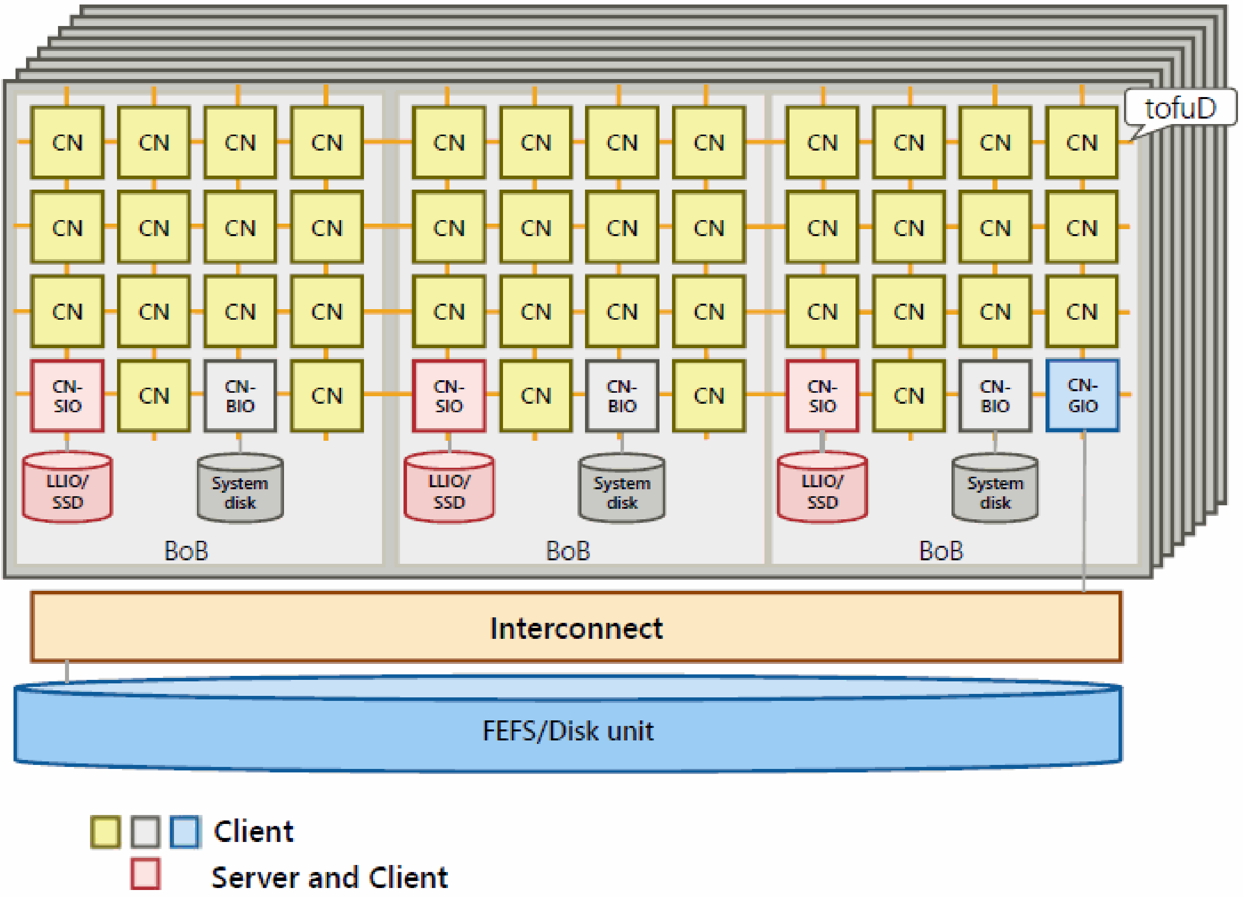
LLIO is a file system that consists of servers and clients. This section describes the LLIO system configuration, where a BoB (compute nodes controlled in a boot unit of 16 nodes) is the base component of a compute cluster. The following figure shows the LLIO physical configuration.

Figure 1.2 LLIO Physical Configuration



The LLIO system environment consists of compute nodes, storage I/O nodes, global I/O nodes, and boot I/O nodes.

Figure 1.3 LLIO System Configuration



See

The following table lists the node types used in LLIO. For details on each node, see the *Job Operation Software Overview* manual.

Table 1.1 Names and Meanings of Node Types

Node Name	Abbreviation	Description
Storage I/O node	SIO	I/O node responsible for input/output for first-layer storage. The storage I/O node is connected to an SSD comprising first-layer storage. The storage I/O node also serves as a compute node.
Global I/O node	GIO	Node that relays input/output for second-layer storage (global file system). The global I/O node also serves as a compute node.
Boot I/O node	BIO	I/O node that also serves as the boot server of nodes. The boot I/O node also serves as a compute node.
Compute node	CN	Node that runs jobs

1.2.1 Server Configuration

You can use the following type of node as an LLIO server:

- -Storage I/O node

Information

The storage I/O node also serves as a second-layer storage client.

1.2.2 Client Configuration

You can use the following types of node as an LLIO client:

- -Compute node
- -Storage I/O node
- -Global I/O node
- -Boot I/O node

1.2.3 Network Configuration

LLIO supports the following network:

- --Tofu interconnect D
    - -Network between the compute nodes, storage I/O node, global I/O node, and boot I/O node

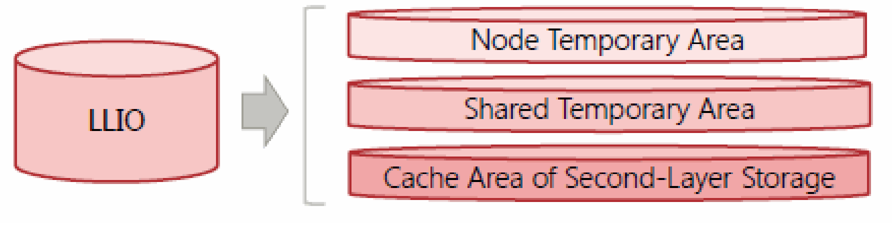
Chapter 2 LLIO Functions

This chapter describes the LLIO functions. The descriptions are intended for system administrators and end users.

2.1 Three Areas of First-Layer Storage

The following table lists the characteristics of the three areas of first-layer storage, which are the cache area of second-layer storage, shared temporary area, and node temporary area.

Figure 2.1 Form of Usage of First-Layer Storage



Cache Area of Second-Layer Storage

This area caches input/output data from jobs to FEFS to increase the speed of access.

Shared Temporary Area

This temporary area for a job can be shared by the compute nodes allocated to the job.
In this area, LLIO stores regular files' data on the first-layer storage, while metadata for all kinds of files is stored on the second-layer storage. Therefore, statfs (2) reports the value on the first-layer storage as the block information, and the value on the second-layer storage as the inode information. The maximum number of regular files containing data that can be created in the shared temporary area is "area size*25/100/300". If the stripe count for first-layer storage is greater than 1, the maximum number is divided by the stripe count.

Node Temporary Area

This temporary area for a job can be used by the individual compute nodes allocated to the job.
The maximum number of files in the node temporary area is "area size*25/100/300".

Table 2.1 Characteristics of the Three Areas

Area Name	Reference Range	Writable to Second-Layer Storage	Life span
Cache area of second-layer storage	All compute nodes	Yes	Not deleted at job end
Shared temporary area	Compute nodes in same job	No	Deleted at job end
Node temporary area	1 compute node	No	Deleted at job end

Information

The maximum number of subdirectories/files in a single directory and the maximum entry of ACL as LLIO file system conform to the FEFS specification.

Note

- To read or write with Direct I/O in the cache area of second-layer storage, the shared temporary area, or the node temporary area, execute the job in units of 64KB.
- When the number of files opened by a job is large, the file access might fail due to the memory shortage of the storage I/O node. The number of files a job can open can be estimated as follows.

  1,024 * <number of compute nodes allocated to the job>

* Please consider that one file is opened when you open the same file from two or more compute nodes.

When asynchronous close is enabled and you write data to a lot of files, the file access might fail due to the memory shortage of the storage I/O node.
The number of files a job can write can be estimated as follows.

1,024 * <number of compute nodes allocated to the job>

* Please consider that one file is written when you write data to the same file from two or more compute nodes.

Please consider the following when you estimate the number of files which your job can open or write.

a. Please evaluate the total of the number of files on cache area of second-layer storage, shared temporary area and node temporary area.

b. Please calculate each stripe as one file when the stripe count of LLIO is two or more. For example, when file-A is opened, the number of opened files about file-A is as follows.

   <the number of opened files about file-A> =
   minimum value of (<file size> / <stripe size>)(*1) and <stripe count>

   (*1) Please round up digits after the decimal point.

c. Please consider that the number of files is as follows when files are common files transferred by the llio_transfer command.

   <the number of common files> * <number of compute nodes allocated to the job>

   The consideration described in b is unnecessary for the common files.

The subsequent sections describe the three areas of first-layer storage in detail.

2.1.1 Cache Area of Second-Layer Storage

LLIO has prepared first-layer storage with a cached area for second-layer storage to implement high-speed access to second-layer storage. This area is called the cache area of second-layer storage. End users can treat this area the same way as the second-layer storage is treated. The second-layer storage cache writes the output results of jobs and other data to second-layer storage asynchronously from job calculation.

The following table describes specifications of the cache area of second-layer storage.

Table 2.2 Specifications of the Cache Area of Second-Layer Storage

Item	Characteristic
Reference range	The cache area can be referenced from all compute nodes.
Writable to second-layer storage	Data is written asynchronously for a write request from a compute node.
Life span	The cache area of second-layer storage is initialized before a job starts and deleted at job end. A job may be interrupted partway through writing to second-layer storage because it exceeds the job execution time limit or for other reasons. In this event, the cache area of second-layer storage is deleted even if writing to second-layer storage has not completed. The unwritten files are output as an unwritten file list, which is a listing of incompletely written files. Other cases in which the cache in the cache area of the second-layer storage is deleted include the following: - When you delete a file - When the cache area of second-layer storage becomes full (In this case, the old cache will be deleted by LRU policy.) - When Direct I/O (write or read) is performed (In this case, cache where the Direct I/O is done will be deleted.)
Inter-node consistency assurance	Equivalent to NFS(verion3) So, the following cases will occur like NFS.

Item	Characteristic
	- If some client nodes write the data to a same file concurrently and the offsets are overlapped, there is a case that the order of writing a data at server node is not same as the order at client nodes. Because the inter-node writing order is not guaranteed. - There is a case that a client node cannot detect immediately the change of file data which another client node executes. But after certain period of time it can be detected. - There is a case that a client node cannot detect immediately the result of file operation (creating and deleting file or directory, changing the file attribute, and so on) which another client node executes. But after certain period of time it can be detected.

When submitting jobs, end users can indirectly specify the size of the cache area of second-layer storage by specifying the sizes of the shared temporary area and node temporary area.

See

For details on how to use the area, see the *Job Operation Software End-user's Guide* manual.

Note

If you write to a file retained and managed by LLIO on the cache area of second-layer storage through second-layer storage directly, LLIO cannot recognize it and there is a case where you cannot read the data from the cache area of second-layer storage.

For example, the case that you execute interactive job and modify the data of file used by this job through second-layer storage directly on the login node, furthermore you use the file again in this job is applicable.

To avoid this problem, you should not read from/write to the file through both second-layer storage and the cache area of second-layer storage at same time.

2.1.2 Shared Temporary Area

LLIO provides the shared temporary area as a temporary area for the files used among multiple job processes in the same job. Multiple compute nodes for the same job can share this area among them. The shared temporary area is available from the beginning of a job and no longer available at the end of the job. Therefore, the files with job execution results and other files that the end user wants to save must be placed on second-layer storage.

The following table describes specifications of the shared temporary area.

Table 2.3 Specifications of the Shared Temporary Area

Item	Characteristic
Reference range	The shared temporary area can be referenced from the compute nodes allocated to a job. It can be referenced from multiple compute nodes allocated to the same job.
Writable to second-layer storage	Files exist only on first-layer storage and are not written to second-layer storage.
Life span	The shared temporary area is initialized before a job starts and deleted at job end.
Inter-node consistency assurance	Equivalent to NFS(verion3)

A dedicated MDT can be specified for the shared temporary area to separate it from meta-access to the second-layer storage.
Also end users can specify the size of the shared temporary area when submitting jobs.
Shared temporary area is consumed not only by data of files but also by meta data of files. Therefore, file access might fail with ENOSPC even if the total amount of data of files included in the filesystem does not exceed the specified amount of shared temporary area. The amount of meta data of files can be estimated as follows.

```
300 byte * <number of files>
```

See

For details on how to use the area, see the *Job Operation Software End-user's Guide* manual.

2.1.3 Node Temporary Area

LLIO provides the node temporary area as a temporary area for the files used locally among the processes in individual compute nodes allocated to a job. This area can be referenced only from a single compute node. The node temporary area is available from the beginning of a job and no longer available from the end of the job. Therefore, the files with job execution results and other files that the end user wants to save must be placed on second-layer storage.

The following table describes specifications of the node temporary area.

Table 2.4 Specifications of the Node Temporary Area

Item	Characteristic
Reference range	The node temporary area can be referenced only from the processes in individual compute nodes allocated to a job.
Writable to second-layer storage	Files exist only on first-layer storage and are not written to second-layer storage.
Life span	The node temporary area is initialized before a job starts and deleted at job end.

End users can specify the size of the node temporary area when submitting jobs.
Node temporary area is consumed not only by data of files but also by meta data of files. Therefore, file access might fail with ENOSPC even if the total amount of data of files included in the filesystem does not exceed the specified amount of node temporary area. The amount of meta data of files can be estimated as follows.

```
300 byte * <number of files>
```

See

For details on how to use the area, see the *Job Operation Software End-user's Guide* manual.

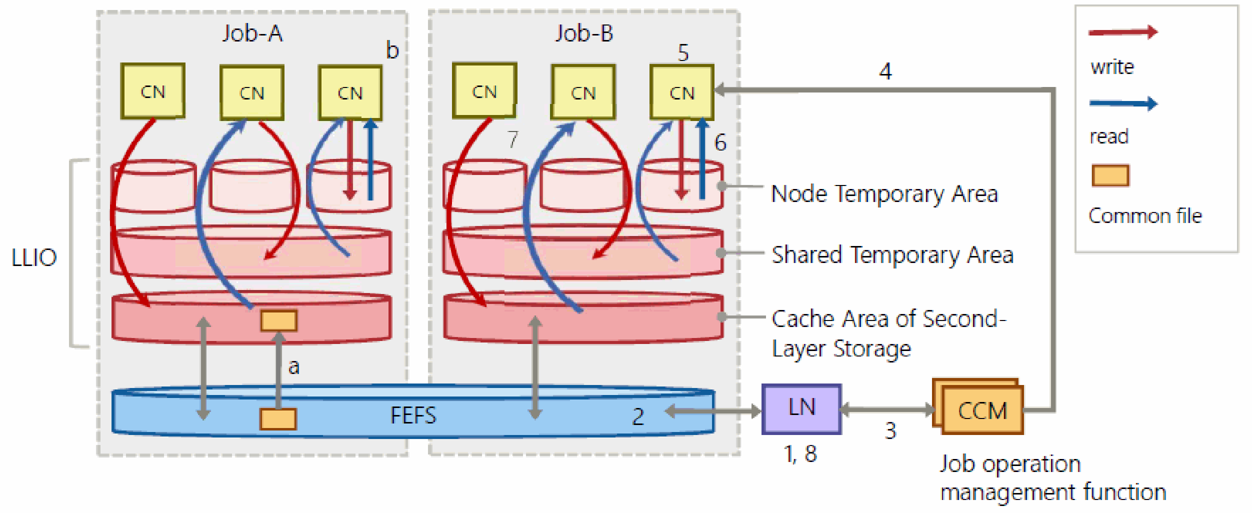
2.2 First-Layer Storage and Jobs

This section uses a diagram to describe the relationship between first-layer storage and jobs.

Note

Unless the system administrator configures the system so that jobs do not use LLIO, jobs always access the files on first-layer storage. Jobs will not access the files on second-layer storage.

Figure 2.2 First-Layer Storage and Jobs



The end user executes a program in the job operation environment provided by the Job Operation Software and the results are then generated using the specified parameters.

The processing flow is described below.

1. The end user logs in to the login node (LN), the entry point, to use the system.
2. The end user places a job script on FEFS, which is second-layer storage. The information in a job submitted with a command of the job operation management function is sent to the job operation management function for batch processing.

   a. In the job script, the end user specifies the files that the user wants placed on first-layer storage beforehand. The purpose is to split the load expected to be concentrated on the files due to the concentration of access from compute nodes to executable files, configuration files, and other files on second-layer storage.

   See

   For details, see "2.3 Common File Distribution Function."

   b. In the job script, the end user can specify a program that uses a library function to acquire compute node statistical information.

   See

   For details, see "2.7.3 Compute Node Statistical Information."

3. The end user requests job execution. The request is sent to the job operation management function of the Job Operation Software on the compute cluster management node (CCM). This is called a job submission.
   The end user can build a suitable LLIO environment for jobs by specifying the parameters in "Table 2.5 LLIO Parameters That Can be Specified at Job Submission" when submitting the jobs.
4. The job operation management function allocates the job to a compute node.
5. The compute node executes the job.
6. The job running on the compute node accesses the files required for executing the job on the node temporary area, the shared temporary area, and the cache area of second-layer storage.
7. The job outputs the results to second-layer storage via the cache area of second-layer storage, which is on first-layer storage.
8. From the login node, the end user views the output results of the job on second-layer storage.

Note

Files on the cache area of second-layer storage may be seen from compute nodes. Those files are in the cache and different from the files on second-layer storage.

Table 2.5 LLIO Parameters That Can be Specified at Job Submission

Classification	Application	For Description of Function
Specification of first-layer storage size	Specify the cache area of second-layer storage	2.1.1 Cache Area of Second-Layer Storage
	Specify the shared temporary area	2.1.2 Shared Temporary Area
	Specify the node temporary area	2.1.3 Node Temporary Area
Higher speed of first-layer storage	Specify that the files read from second-layer storage to a compute node are cached in first-layer storage	2.4.1 Cache Function for First-Layer Storage During Reading
	Specify the stripe size and stripe count for distributing files on first-layer storage	2.4.2 Striping Function for First-Layer Storage
	Specify the stripe size and stripe count for distributing files on second-layer storage	2.4.3 Striping Function for Second-Layer Storage
	Specify asynchronous closing of files on layered storage	2.4.4 Asynchronous Close Function
Use of compute node cache	Specify the size of memory allocated to the compute node cache from the amount of memory resources available to jobs	2.5.1 Compute Node Cache Memory Size
	Specify that files be cached on the compute node when the files are read from layered storage	2.5.2 Cache Function for a Compute Node Cache During Reading
	Specify that contiguous areas in layered storage are automatically read before areas in the compute node cache are read	2.5.3 Automatic Readahead Function for a Compute Node Cache
	Specify the write size threshold for switching between using and not using the compute node cache	2.5.4 Write Size Threshold for Switching Between Using and Not Using a Compute Node Cache
Job error handling	Specify that a file name list be output if files not written to second-layer storage remain on first-layer storage when a job is completed	2.6 Unwritten File List Acquisition Function
Statistical information	Acquire job statistical information	2.7.1 Job Statistical Information
	Acquire LLIO performance information	2.7.2 LLIO Performance Information
	Acquire compute node statistical information	2.7.3 Compute Node Statistical Information

2.2.1 LLIO Area selection

When submitting jobs, end users can select the LLIO area that they want to make accessible by specifying the environment variables in "Table 2.6 LLIO Environment variables That Can be Specified at Job Submission".

Table 2.6 LLIO Environment variables That Can be Specified at Job Submission

Environment variable name	Details	Precautions
PJM_LLIO_GFSCACHE	Specifies the cache area of second-layer storage to use. "pjsub -x PJM_LLIO_GFSCACHE=*pathname*" can be specified. To specify more than one, delimiting them with ":".	If this environment variable is not specified, the cache area of second-layer storage on the end user's home directory will be used. If this environment variable is set, the cache area of second-layer storage where the end user's home directory is located can be used even if it is not specified in this environment variable.

Environment variable name	Details	Precautions
PJM_LLIO_SHAREDTMP	Specifies second-layer storage to use as the MDT for shared temporary area. "pjsub -x PJM_LLIO_GFSCACHE=*pathname*" can be specified.	If this environment variable is not specified, second-layer storage on the end user's home directory will be used.

An example of specifying environment variables is shown below.

[pjsub example]

```
pjsub -x PJM_LLIO_GFSCACHE=/fefs1:/fefs2 -x PJM_LLIO_SHAREDTMP=/fefs3 ...
```

[Job script example]

```
:
#PJM -x PJM_LLIO_GFSCACHE=/fefs1:/fefs2
#PJM -x PJM_LLIO_SHAREDTMP=/fefs3
:
```

Note

Due to a file system failure, the file system may be disconnected in order to continue job operation. If you specify a disconnected file system in the environment variable, the job will fail. for file system disconnected, see the FEFS User's Guide manual.

2.3 Common File Distribution Function

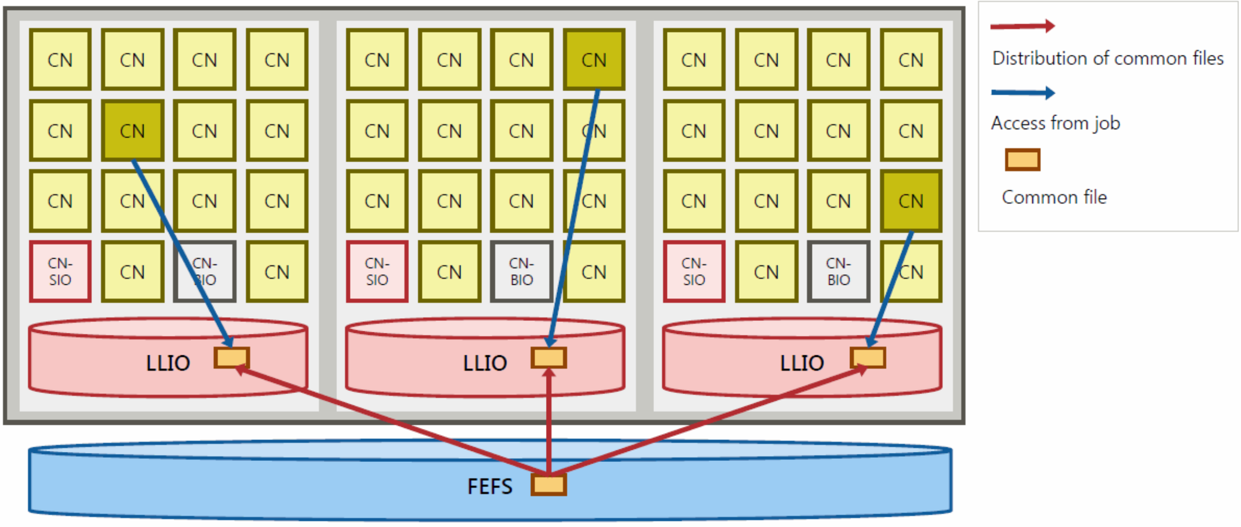
When a job begins, all compute nodes access files (common files) such as executable files and configuration files on second-layer storage, so access is concentrated on those files. This results in concentration on access on a specific storage I/O node, so performance may deteriorate. To prevent this situation, LLIO provides a function to balance access by distributing the common files to the cache area of second-layer storage, which is on first-layer storage. This is called the common file distribution function. End users can specify the distribution of the common files in a job script.

See

For details on how to use the function, see "A.2.3 llio_transfer Command" in the *Job Operation Software End-user's Guide* manual.

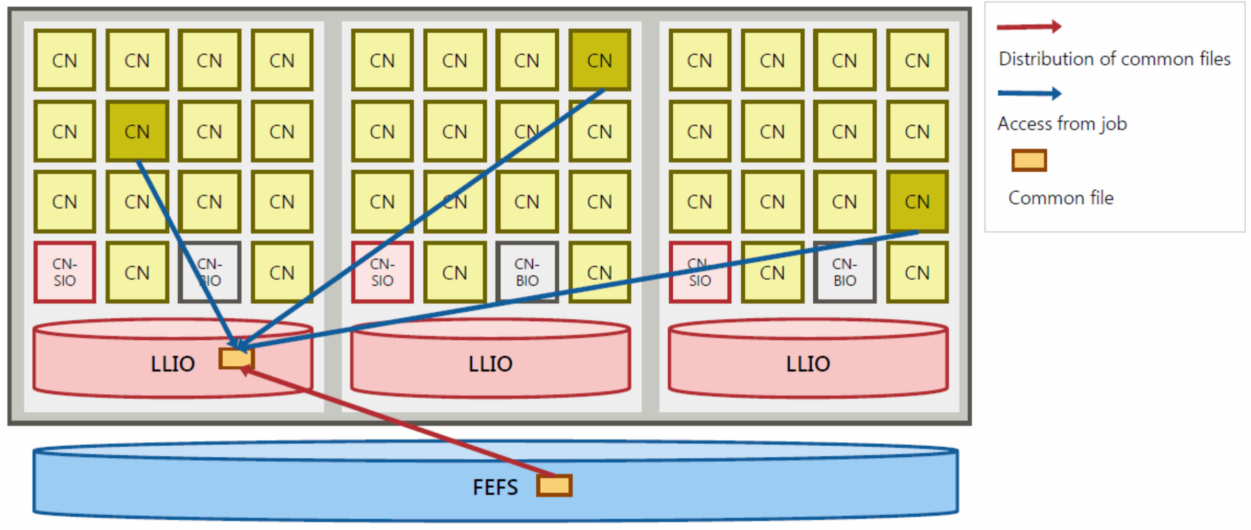
The following diagram shows execution of the function.

Figure 2.3 Common File Distribution Function Used



For every storage I/O node that will be allocated to jobs, the common file distribution function has distributed files to the cache area of second-layer storage, which is on first-layer storage. Each of the jobs on compute nodes accesses the storage I/O node that is physically closest, which balances the access load on the storage I/O nodes.

Figure 2.4 Common File Distribution Function Not Used



If the common file distribution function is not used, the jobs on the compute nodes concentrate access on a single storage I/O node, so performance may deteriorate.

2.4 Higher Speed of First-Layer Storage

2.4.1 Cache Function for First-Layer Storage During Reading

LLIO provides a cache function for the files that are read from second-layer storage to a compute node cache. While being read, the files are cached in first-layer storage. By caching files in first-layer storage in case that the same file is read multiple times within a job, this function implements performance improvements.

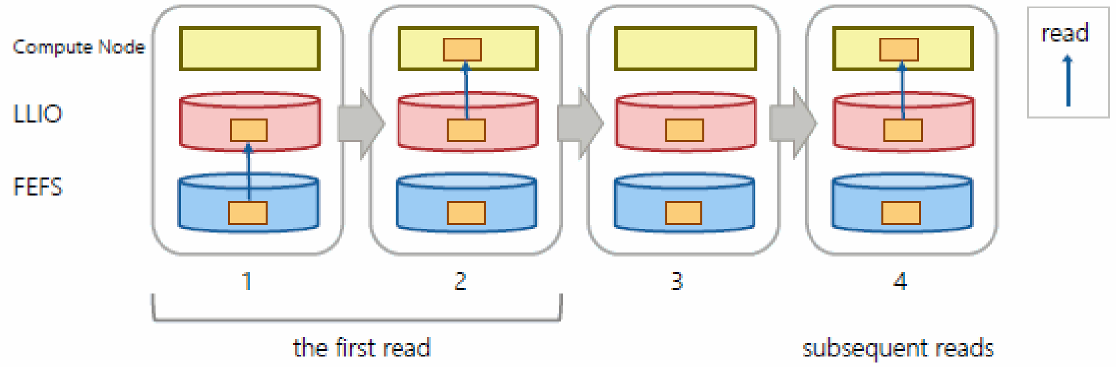
When submitting jobs, end users can specify caching to first-layer storage during reading.

See

- -For details on a compute node cache, see "2.5 Compute Node Cache Function."
- -For details on how to use the function, see the *Job Operation Software End-user's Guide* manual.

The following figure shows operation with the cache function for first-layer storage.

Figure 2.5 Operation With the Cache Function for First-Layer Storage



1. At the first read time, the files are read from second-layer storage to first-layer storage.
2. The files are read from first-layer storage to the compute node cache.
3. The files are cached in first-layer storage.
4. At the second and subsequent read times, the compute node reads the files from first-layer storage.
   The files on second-layer storage do not need to be read.

2.4.2 Striping Function for First-Layer Storage

LLIO provides a striping function for first-layer storage to distribute a single file over multiple storage I/O nodes. The striping function for first-layer storage has the following effects.

- -Users can create a file with a size exceeding the physical capacity of a single SSD.
- -The file access bandwidth improves when a single file is distributed over multiple storage I/O nodes of first-layer storage.

When submitting jobs, end users specify the stripe size and stripe count based on the I/O characteristics of the application. In this way, data can be effectively transferred to first-layer storage.

See

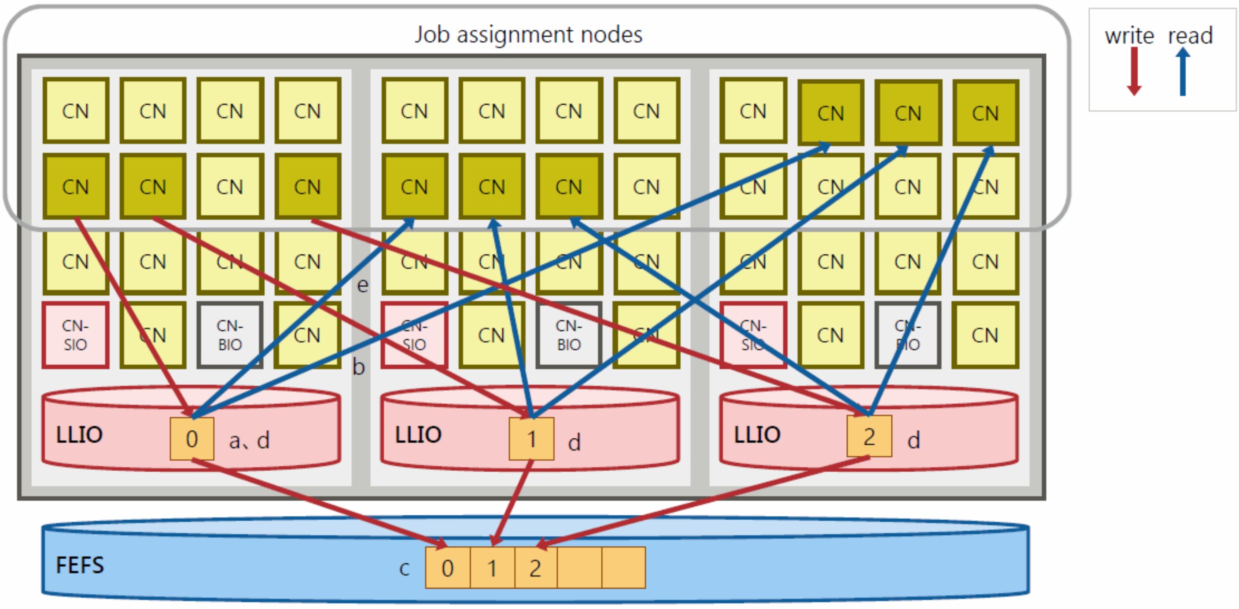
For details on how to use the function, see the *Job Operation Software End-user's Guide* manual.

Information

The striping function is available only on the shared temporary area and the cache area of second-layer storage. This function is not available on the node temporary area.

The following figure shows an example of using the striping function for first-layer storage. The example specifies a stripe size of 1 MiB and a stripe count of 3 for the cache area of second-layer storage.

Figure 2.6 Operation With the Striping Function for First-Layer Storage



a. LLIO determines the first storage I/O node.
b. LLIO writes the files in round-robin fashion to three storage I/O nodes according to the stripe size.
c. The files stored in first-layer storage on the three storage I/O nodes are written in the background as needed to second-layer storage.

d. The end user can specify whether first-layer storage keeps the files even after they are written to second-layer storage.

e. The compute nodes read the files written to first-layer storage.

2.4.3 Striping Function for Second-Layer Storage

LLIO provides a function to set and view striping for second-layer storage from compute nodes. When the stripe size and stripe count are specified for second-layer storage, data can be effectively transferred from the second-layer storage cache to second-layer storage.

End users can set striping and specify viewing for second-layer storage in a job script.

See

For details on how to use the function, see the *Job Operation Software End-user's Guide* manual.

Information

The setstripe subcommand and getstripe subcommand in the lfs command are provided to set and view striping for second-layer storage from a second-layer storage client. For details, see the *FEFS User's Guide* manual.

2.4.4 Asynchronous Close Function

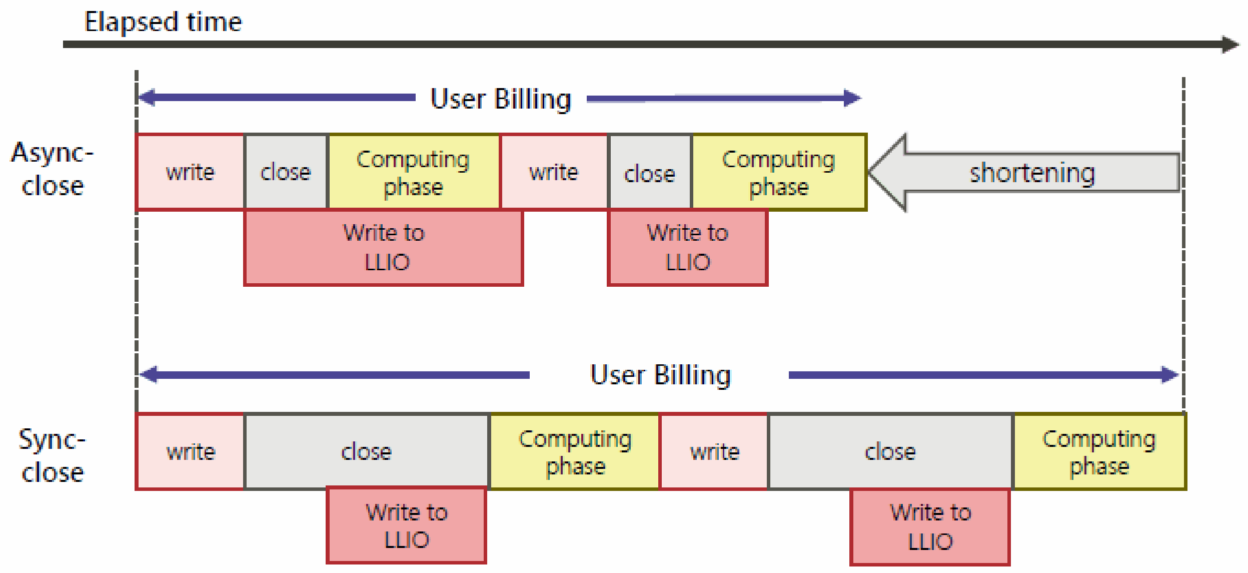
LLIO provides an asynchronous close function to asynchronously write from a compute node cache to first-layer storage or from the second-layer storage cache to second-layer storage when closing a file. The effect of using this function is that jobs end earlier without waiting for writing to complete. The writing of files by job end is guaranteed even when the function is enabled.

End users can specify the asynchronous close function when submitting jobs.

See

- -For details on a compute node cache, see "2.5 Compute Node Cache Function."
- -For details on how to use the function, see the *Job Operation Software End-user's Guide* manual.

Figure 2.7 Operation With the Asynchronous Close Function



Asynchronous close begins with writing from a compute node cache at synchronous close to first-layer storage and with writing to second-layer storage. Neither the writing to first-layer storage nor the writing to second-layer storage is guaranteed at end time of the close. The writing may or may not end in the next compute phase. However, the writing of files by job end is guaranteed, even when the asynchronous close function is enabled.

Synchronous close ensures that writing to first-layer storage and second-layer storage ends by the end time of the close.

Note

- When asynchronous close is enabled, if you open a file in O_WRONLY or O_RDWR and then execute execve(2) on the file immediately after closing it, the execve(2) may fail with ETXTBSY. To access the file in this way, turn off asynchronous close and execute the job.
- When asynchronous close is enabled, close(2) does not notify you of a failure to write file data from the compute node cache. For the cache area of the second-layer storage, the write failure can be confirmed by an unwritten file list (see "2.6 Unwritten File List Acquisition Function"). For shared and node temporary areas, however, the failures cannot be confirmed by the unwritten file list.
  If you attempt to write more than the space size in the shared temporary area or the node temporary area, you will get an ENOSPC error, and you may not be able to check for the write failures. Therefore, please estimate the size of files data stored in these areas and specify enough space size.

2.5 Compute Node Cache Function

2.5.1 Compute Node Cache Memory Size

LLIO provides a function to specify the compute node cache memory size based on the available amount of memory resources for a job. For applications and the like that do not use much memory to execute jobs, LLIO implements high-speed I/O by using the unused memory resources for applications as a compute node cache.

When submitting jobs, end users can specify the size of memory allocated to a compute node cache.

See

For details on how to use the function, see the *Job Operation Software End-user's Guide* manual.

2.5.2 Cache Function for a Compute Node Cache During Reading

LLIO provides a cache function for the files that are read from layered storage to a compute node. While being read, the files are cached in a compute node cache. By caching the files read from layered storage in the compute node cache in case that an application reads the same file multiple times, this function implements performance improvements.

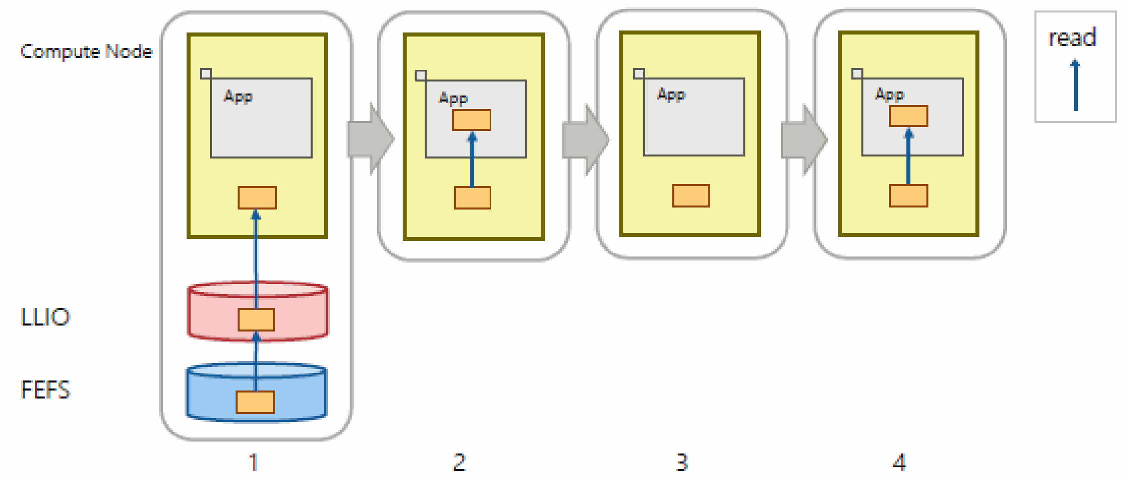
When submitting jobs, end users can specify the function to cache to a compute node cache during reading.

See

For details on how to use the function, see the *Job Operation Software End-user's Guide* manual.

The following figure shows operation with the function caching to a compute node cache.

Figure 2.8 Operation With the Function Caching to a Compute Node Cache



1. The files are read from second-layer storage to first-layer storage. These files are cached in a compute node cache.
2. The files read to the compute node cache are read to an application buffer.
3. The files are read from first-layer storage to the compute node. These files are cached in the compute node cache.
4. When reading the same files again, the application reads the files in the compute node cache.

2.5.3 Automatic Readahead Function for a Compute Node Cache

LLIO provides an automatic readahead function to specify whether or not to automatically read ahead from a compute node cache when a job tries to read contiguous areas in layered storage. By reading ahead from a compute node cache, this function implements higher speeds in read processing.
End users can specify the automatic readahead function for a compute node cache when submitting jobs.

See

For details on how to use the function, see the *Job Operation Software End-user's Guide* manual.

2.5.4 Write Size Threshold for Switching Between Using and Not Using a Compute Node Cache

LLIO provides a function to specify the write size threshold for switching between using and not using a compute node cache. This function implements high-speed transfer from a compute node to first-layer storage at high speed as follows. The files that are written from an application to first-layer storage are temporarily cached in a compute node cache. Then, the files are batch transferred in units of the maximum transfer size between the compute node and storage I/O node.

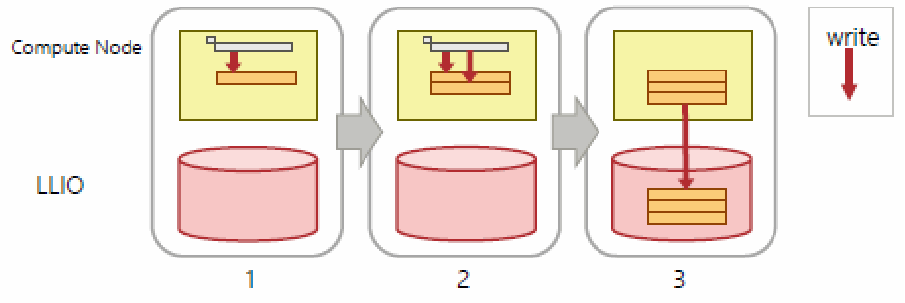
When submitting jobs, end users can specify the write size threshold for switching between using and not using a compute node cache.

See

For details on how to use the threshold, see the *Job Operation Software End-user's Guide* manual.

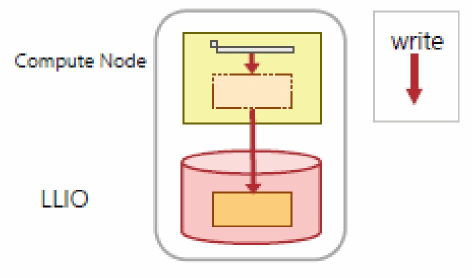
The following figures show operation with the function specifying the write size threshold for switching between using and not using a compute node cache.

Figure 2.9 Operation with a write Size Equal to or Less than the Specified Value



1. If the size of the data written from an application to first-layer storage is equal to or less than the specified value, the data is cached in the compute node cache.
2. If the total size of the data written from an application to first-layer storage is equal to or less than the specified value, the successive data is cached in the compute node cache.
3. The cache data is written to the first tier storage according to the fsync system call or the OS asynchronous write processing. If the total cache data being exported is greater than the size of the in-compute node cache, a synchronous export is performed.

Figure 2.10 Operation with a write Size Exceeding the Specified Value



If the size of the data written from an application to first-layer storage exceeds the specified value, the data is not written to a compute node cache but instead written to first-layer storage in synchronization with the application. If the compute node cache has files that must be written to first-layer storage, the data in the compute node cache is written to first-layer storage in advance.

2.6 Unwritten File List Acquisition Function

Suppose, for example, that an error occurs on a compute node while executing a job or that a job reaches the specified elapsed time limit for the job while write processing is in progress. Then, some files may remain unwritten. The writing of these files from a compute node cache to first-layer storage or from first-layer storage to second-layer storage could not be completed.

LLIO provides a function to output an unwritten file list, which is a listing of incompletely written files. The end user cannot only find out the incomplete files from the output but also get information for adjusting the possible execution times of the next executed job by analyzing the list.

End users can specify the unwritten file list acquisition function when submitting jobs.

See

For details on how to use the function, see the *Job Operation Software End-user's Guide* manual.

2.7 Statistical Information

LLIO provides three types of statistical information as a means for end users to get a grasp of file access for applications to first-layer storage.

Job statistical information

Users can check job submission conditions and the status of first-layer storage access.

LLIO performance information

Users can check meta-access information and input/output information for first-layer storage in more detail than job statistical information.

Compute node statistical information

By using a library in a job, users can check meta-access information and input/output information for first-layer storage by compute node.

2.7.1 Job Statistical Information

LLIO outputs some LLIO statistical information as job statistical information in linkage with the Job Operation Software. With a job submission command, end users can get the job statistical information as the output results of a job. By referring to the information, they can subsequently analyze the state when the job was executed.

See

- To output job statistical information as a result of a job, see "Specifying job statistical information output" in "Job Operation Procedures" of the *Job Operation Software End-user's Guide* manual.
- For details on the output items of job statistical information, see the man page pjstatsinfo(7) for the Job Operation Software.

2.7.2 LLIO Performance Information

LLIO provides LLIO performance information in linkage with the Job Operation Software. This function outputs the storage I/O node statistical information collected on the compute cluster management node as LLIO performance information at job end. The Job Operation Software has aggregated the statistical information in units of storage I/O nodes.

End users can specify at job submission that the LLIO performance information is obtained as the output of the job. By referring to the information, they can subsequently analyze the status of first-layer storage access related to their own jobs. The administrator can view the LLIO performance information for each job from the log file llioinfo of the Job Operation Software.

See

- To output LLIO performance information as a result of a job, see "Collecting LLIO performance information" in "Job Operation Procedures" of the *Job Operation Software End-user's Guide* manual.
- For details on the output items of LLIO performance information, see "C.1 Output Items of LLIO Performance Information."

Note

LLIO performance information is collected while the job is in the RUNOUT state.

As the number of nodes executed the job increases, the time of collecting LLIO performance information also increases. Therefore the time of RUNOUT state increases as well.

Collecting LLIO performance information is finished in a few minutes, if the number of nodes for the job is 10,000. But it takes 30 miniutes for 20,000 nodes, and about 2 hours for 50,000 nodes.

2.7.3 Compute Node Statistical Information

LLIO provides a library function to acquire I/O information from compute nodes to first-layer storage. End users can check the I/O status related to these compute nodes by using the library function to provide compute node statistical information.

See

For details on how to collect compute node statistical information and on the collected items, see "C.2 Collection Methods and Output Items of Compute Node Statistical Information."

Information

The statistical information can be acquired only from compute nodes that already acquired compute node statistical information.

2.8 Functions for the System Administrator

2.8.1 Checking the LLIO Status

Working together with the Job Operation Software, LLIO provides a function to monitor the status of the compute nodes, storage I/O nodes, and global I/O nodes running LLIO. The system administrator can check the status of each node by using the pashowclst command, which is the status monitoring function command.

See

For details on how to monitor the status of LLIO-related services, see "3.2.1 Monitoring the LLIO Status."

2.8.2 System Statistical Information

LLIO provides system statistical information as a means for the system administrator to get a grasp of the file I/O status, monitor the operation status, troubleshoot, and perform tuning. The system administrator can get statistical information about storage I/O nodes and global I/O nodes by configuring the collectl service, which is open source, to start automatically.

See

- -For details on how to collect system statistical information, see "3.2.3 Collecting System Statistical Information."
- -For details on the collected items of system statistical information, see "C.3 Output Items of System Statistical Information."

Chapter 3 System Operation

This chapter describes LLIO system operation. The descriptions are intended for system administrators.

3.1 Installation

LLIO is installed at the same time as FEFS installation.

Note

For details on FEFS installation, see the *FEFS User's Guide* manual.

Perform the following procedure to install LLIO:

1. Design the LLIO configuration
2. Apply the LLIO packages and collectl package
3. Create an FEFS design sheet
4. Build LLIO
5. Check the LLIO status
6. Configure the job ACL function
7. Configure linkage with the Job Operation Software

Note

- -The collectl package is procured separately and applied. If system statistical Information is not collected, installation of the package is not necessary.
- -collectl is supported only in version 4.3.0.

3.1.1 Designing the LLIO Configuration

Design the LLIO configuration specifically with the following points in mind:

- --To determine the SSD
  - -SSD capacity
  - -SSD device path name
- --To determine the configuration
  - -Number of storage I/O nodes
  - -Number of compute nodes
  - -Configuration with a storage I/O node and compute nodes

3.1.2 Applying the LLIO Packages

The LLIO packages are as follows.

LLIO Primary Packages

1. FJSVllio-*.aarch64.rpm
2. FJSVllio-modules-*.aarch64.rpm

3. FJSVllio-stats-*.aarch64.rpm

   * Enter a version number and release name at the asterisk (*).

The following table shows the relationship between the packages and the applying nodes.

Table 3.1 LLIO Primary Packages and Applying Nodes

Package Name	Node Type			
	SIO	GIO	BIO	CN
FJSVllio	Yes	Yes	Yes	Yes
FJSVllio-modules	Yes	Yes	Yes	Yes
FJSVllio-stats	Yes	Yes	Yes	Yes

LLIO Peripheral Package

1. FJSVllio-*.x86_64.rpm
2. FJSVllio-stats-*.x86_64.rpm

   * Enter a version number and release name at the asterisk (*).

The following table shows the relationship between the package and the applying nodes.

Table 3.2 Peripheral LLIO Package and Applying Nodes

Package Name	Node Type		
	CCM	LN	Multiuse *1
FJSVllio	Yes	Yes	Yes
FJSVllio-stats		Yes	

*1 Apply the package to the multiuse node only if it is used as the statistical information collection node. When using a node other than a multiuse node as the statistical information collection node, apply the package to that node.

Information

For details on the statistical information collection node, see "How to Output System Statistical Information for Multiple Storage I/O Nodes."

Note

Procure the collectl package separately and apply it to the following nodes.

Table 3.3 collectl Package and Applying Nodes

Package Name	Node Type		
	SIO	GIO	Multiuse *1
collectl	Yes	Yes	Yes

*1 Apply the package to the multiuse node only if it is used as the statistical information collection node. When using a node other than a multiuse node as the statistical information collection node, apply the package to that node.

3.1.3 Creating an FEFS Design Sheet

Write the LLIO configuration in the LLIO sheet in an FEFS design sheet. Create the FEFS design sheet on a Windows terminal. The FEFS product includes an FEFS design sheet template whose name is "FEFSDesignSheet.xlsm". Enable macros in Excel first before starting to work on creating the FEFS design sheet. The input items in the red cells are mandatory items. Be sure to fill with values all of the required items.

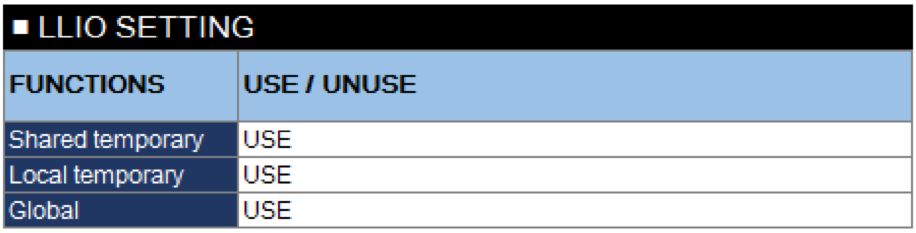
Note

The supported versions of Windows and Excel are Excel 2010 and 2013 on Windows 8, and Windows 10. For an environment other than the above, consult with a Fujitsu system engineer (SE) or the Fujitsu Support Desk.

Set LLIO installation information, the storage I/O node, and the first-layer storage device in the LLIO sheet.

1. LLIO SETTING section

Figure 3.1 Example of Entries in the LLIO SETTING Section



■ LLIO SETTING

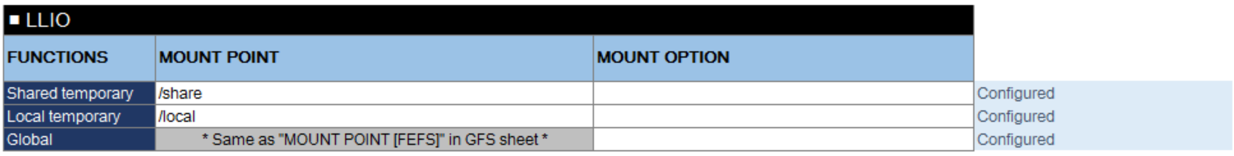
FUNCTIONS	USE / UNUSE
Shared temporary	USE
Local temporary	USE
Global	USE

When using LLIO, specify USE for all items.

If you do not use LLIO, leave the initial value as UNUSE, and you do not need to set LLIO in the FEFS design sheet afterwards.

2. LLIO section
   The descriptions below are based on "Example of Entries in the LLIO Section."

Figure 3.2 Example of Entries in the LLIO Section



■ LLIO

FUNCTIONS	MOUNT POINT	MOUNT OPTION	
Shared temporary	/share		Configured
Local temporary	/local		Configured
Global	* Same as "MOUNT POINT [FEFS]" in GFS sheet *		Configured

a. MOUNT POINT [Shared temporary]
   Specify the shared temporary mount point in a job.

b. MOUNT OPTION [Shared temporary]
   Specify the shared temporary mount option in a job. Normally, the option does not need to be changed.

c. MOUNT POINT [Local temporary]
   Specify the temporary mount point in a node.

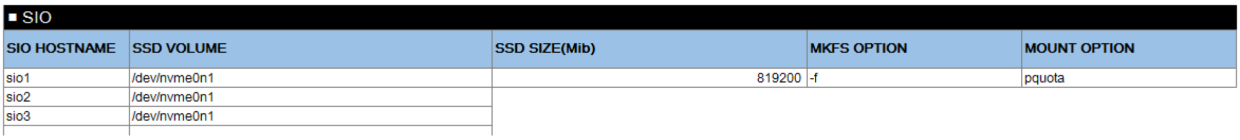
d. MOUNT OPTION [Local temporary]
   Specify the temporary mount option in a node. Normally, the option does not need to be changed.

e. MOUNT OPTION [Global]
   Specify the cache mount option of second-layer storage within a job. Normally, the option does not need to be changed.

The mount options cannot be defined without a mount point. As seen on the right side of the input field, "Configured" will be displayed if the mount option is properly defined.

3. SIO section
   The descriptions below are based on "Example of Entries in the SIO Section."

Figure 3.3 Example of Entries in the SIO Section



■ SIO

SIO HOSTNAME	SSD VOLUME	SSD SIZE(Mib)	MKFS OPTION	MOUNT OPTION
sio1	/dev/nvme0n1	819200	-f	pquota
sio2	/dev/nvme0n1			
sio3	/dev/nvme0n1			

a. SIO HOSTNAME
Enter the host name of a storage I/O node.

b. SSD VOLUME
Define the path to the first-layer storage connected to the applicable storage I/O node.

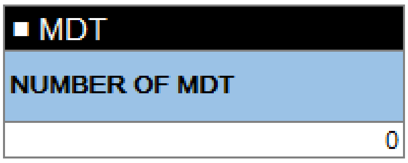
c. SSD SIZE (MiB)
Specify the device size used by the first-layer storage, in units of MiB.

d. MKFS OPTION
Specify the mkfs option of the first-layer storage. Normally, the option does not need to be changed.

e. MOUNT OPTION
Specify the mount option of the first-layer storage device. Normally, the option does not need to be changed.

4. MDT section
The following "Example of entries in the MDT section" is provided along with an explanation.

Figure 3.4 Example of entries in the MDT section



■ MDT

NUMBER OF MDT

0

a. NUMBER OF MDT
If you have dedicated MDTs for shared temporary use, specify the number of dedicated MDTs.
Specify 0 if you do not have a dedicated MDT for shared temporary. (Default 0)

Information

This section specifies the number of FEFS MDTs used by the shared temporary area. If specified, the FEFS MDT used by the shared temporary area will be allocated from the one with the highest index number. The system administrator uses the FEFS remote directory and stripe directory functions to set the FEFS MDT used for home directory, etc. to something other than the FEFS MDT used in the shared temporary area, thereby the FEFS MDT used for file access to the shared temporary area and the file access to the cache area of the second tier storage and the FEFS can be separated.

3.1.4 Building LLIO

After creating the FEFS design sheet, perform the following work concurrently with building FEFS:

- -Create a configuration definition file for the FEFS setup tool
- -Place the configuration definition file for the FEFS setup tool
- -Build FEFS

See

For details on FEFS build, see the *FEFS User's Guide* manual.

3.1.5 Checking the LLIO Status

In the FEFS build in "3.1.4 Building LLIO," starting the FEFS service automatically starts the LLIO services. The system administrator executes the pashowclst command to confirm that the LLIO services have started normally on storage I/O nodes and compute nodes.

```
[System management node]
# pashowclst -v --nodetype CN
```

The LLIO services have started normally when the LLIO status transitions to FEFSSR(o) and FEFS(o).

For details on the service status, see "3.2.1 Monitoring the LLIO Status."

Note

The cache area of second-layer storage, the shared temporary area, and the node temporary area are mounted when a job is submitted. These areas are not mounted at the service start time.

3.1.6 Configuring the Job ACL Function

Configure job ACL with the pmjacladm command. The job ACL function is a function of the Job Operation Software. It controls the upper limit on the amount of resources that can be specified for a job, the privileges for using commands related to job operations, and more.

See

For details on the concept and general performance of the job ACL function, see the *Job Operation Software Administrator's Guide for Job Management* manual.

3.1.7 Setting Linkage with the Job Operation Software

Set a "hook" for linkage with the Job Operation Software. The hook is a function of the Job Operation Software. The hook can be executed by an administrator incorporating arbitrary processing while the process of executing a job is in progress. The job uses the job manager exit function for the hook. LLIO prepares the job to use LLIO and performs post-processing.

See

For details on a hook, see the *Job Operation Software Administrator's Guide for Job Operation Manager Hook* manual.

This section describes how to configure the job manager exit function.

Configuring the Job Manager Exit Function

Configure the LLIO exit function library liblliohook.so in the ExitFunc subsection of the /etc/opt/FJSVtcs/Rscunit.d/*resource unit name*/pmpjm.conf file for each resource unit of the system management node.

```
ResourceUnit {
    ResourceUnitName = resource unit name
        ...
        ExitFunc {                          <- Setting for resource unit
            ExitFuncLib = liblliohook.so    <- Specifies exit function library
            ExitFuncPri = 200               <- Execution priority of exit function
            ExitFuncType = pjm              <- Exit function type
        }
}
```

Write the ExitFuncLib and ExitFuncType items as shown above. There is no specific condition for ExitFuncPri. If the execution priority duplicates that of another exit function library, the libraries are executed in the order described.

Execute the pmpjmadm command to integrate the LLIO exit function library into the Job Operation Software.

```
[System management node]
# pmpjmadm -c cluster name --set --rscunit resource unit name
```

3.2 Operation

3.2.1 Monitoring the LLIO Status

The system administrator can monitor the LLIO status with the pashowclst command. The FEFSSR monitoring service and FEFS monitoring service are the two types of status monitoring. The following table lists the target nodes and items of status monitoring.

Table 3.4 Target Nodes and Items of Status Monitoring

Monitoring Service	Status Monitoring Item	Status Monitoring Target Node
FEFSSR	LLIO server function	Storage I/O node
	Relay function of global I/O node	Global I/O node
FEFS	FEFS client function	Storage I/O node, global I/O node, compute node

Example: Check the status of service on the target node with the -n option in the pashowclst command.

```
[System management node]
# pashowclst -c clstname -n nodeid
[ CLST: clstname ]
[ NODE: nodeid ]
NODE        NODETYPE  STATUS    REASON     PWR_STATUS     ARCH_STATUS    SRV_STATUS
nodeid      SIO,CN    Running   -          os-running     ICC_Running
PLE(o),NRD(o),FEFSSR(o),FEFS(o),PWRD(o)
```

The following tables lists the meaning of each output status for services on the status monitoring target nodes.

Table 3.5 FEFSSR Status and Meaning for Storage I/O Nodes

Service Status	Meaning
o	All monitored items are normal.
!	The network became degraded or abnormal.
x	One of the monitored items is abnormal.
s	Post-startup setting. There is no status transition until each status is normal.
b	FEFS and LLIO are not set.

Table 3.6 FEFSSR Status and Meaning for Global I/O Nodes

Service Status	Meaning
o	All monitored items are normal.
x	The FEFS service is stopped or abnormal.
!	The network became degraded or abnormal.
s	Post-startup setting. There is no status transition until each status is normal.
b	FEFS and LLIO are not set.

Table 3.7 FEFS Status and Meaning for Storage I/O Nodes, Global I/O Nodes, and Compute Nodes

Service Status	Meaning
o	All monitored items are normal.
x	One of the monitored items is abnormal.
!	The network became degraded or abnormal.
s	Post-startup setting. There is no status transition until each status is normal.
a	The service is unavailable due to an abnormality in FEFS on a storage I/O node or global I/O node.
b	FEFS and LLIO are not set.

See

For details on the display by the pashowclst command, see the *Job Operation Software Administrator's Guide for System Management* manual.

3.2.2 Changing Job ACL Function Settings

Use the pmjacladm command to change job ACL function settings during operation. The job ACL function is a function of the Job Operation Software. It controls the upper limit on the amount of resources that can be specified for a job, the privileges for using commands related to job operations, and more.

See

For details on the concept and general performance of the job ACL function, see the *Job Operation Software Administrator's Guide for Job Management* manual.

3.2.3 Collecting System Statistical Information

3.2.3.1 Collecting System Statistical Information From a Storage I/O Node

Collect system statistical information from a storage I/O node with lliosv.ph, which is a plugin for collectl. The information can be automatically collected regularly when the collectl service is set to start automatically.

How to Collect System Statistical Information From a Storage I/O Node

In the following procedure, the system administrator sets the collectl configuration file (/etc/collectl.conf) to collect system statistical information from a storage I/O node.

1. Create a collectl.conf file, and write the following DaemonCommands line with settings for collectl daemon start options.

```
[System management node]
# vi collectl.conf
DaemonCommands = -i <systemstat_interval> -f <systemstat_dir> -r<time>,<systemstat_rolllogs> (*)
-m -F<flush_time> -s C --import /opt/FJSVllio/bin/lliosv.ph
```

Note: The places marked with "(*)" in the above example show line feeds inserted due to space limitations. Actually, the separated output appears on a single line.

The following table lists the setting items on the DaemonCommands line that has settings for collectl daemon start options in the collectl configuration file.

Table 3.8 DaemonCommands Setting Items

Setting Item	Description
-i <*systemstat_interval*>	Specifies the statistical information collection interval in seconds in <systemstat_interval>. The recommended value is 600. If nothing is specified, the default value in collectl is assumed.
-f <*systemstat_dir*>	Specifies the statistical information output directory in <*systemstat_dir*>. The recommended directory is /var/opt/FJSVllio/lliostat/. If nothing is specified, the default value in collectl is assumed.
-r<*time*>,<*systemstat_rolllogs*>	Specifies the time to rotate a log in <*time*> and the number of collection days in <*systemstat_rolllogs*>. Specify the time in the *hh:mm* format and a numeric value for the number of days. Suppose that the log file is switched every day and the specified number of days has elapsed. Then, the files are deleted in chronological order from oldest to newest. The recommended value for <*systemstat_rolllogs*> is 10. If nothing is specified, the default value in collectl is assumed.

Setting Item	Description
-m	Creates a collectl operation log in the directory specified in the -f option.
-F<*flush_time*>	Specifies the data write interval in seconds in <flush_time> for writing statistical information from memory to a disk. Specify 0 to write to the disk every time data is collected. If nothing is specified, the default value in collectl is assumed.
-s C	Collects the load on each CPU core.

2. Distribute the edited collectl.conf file to all storage I/O nodes.

```
[System management node]
# pmscatter -c clstname --nodetype SIO ./collectl.conf /etc/collectl.conf
```

3. The collectl service starts automatically.

```
[System management node]
# pmexe -c clstname --nodetype SIO "/usr/bin/systemctl enable collectl"
# pmexe -c clstname --nodetype SIO "/usr/bin/systemctl start collectl"
```

4. When changing the settings during operation, restart the collectl service as follows.

```
[System management node]
# pmexe -c clstname --nodetype SIO "/usr/bin/systemctl restart collectl"
```

How to Output System Statistical Information for a Storage I/O Node

Use a log file collected by collectl to output system statistical information for a storage I/O node.

This section shows how to output system statistical information for a storage I/O node.

```
[Storage I/O node]
# collectl -p <data file> -oD -s-C --plot [<collectl option>] --import (*)
/opt/FJSVllio/bin/lliosv.ph [,from=<value>] [,stat=<value>] [,jobid=<job ID>] [,kind=<value>] (*)
[,type=<value>] [,section=<value>]
```

Note: The places marked with "(*)" in the above example show line feeds inserted due to space limitations. Actually, the separated output appears on a single line.
In <*data file*>, specify a log file collected by collectl on the storage I/O node.

You can specify a separator between output items and an output time range in the collectl command options.

You can specify the following collectl extension plugin options for system statistical information for a storage I/O node.

See

For details on job ID, see the *Job Operation Software End-user's Guide* manual.

Table 3.9 collectl Extension Plugin for Storage I/O Nodes

Option	Description
from=<*value*>	Filters the output statistical information by date and time. You can specify the following in <*value*>: yyyymmdd:hh:mm:ss-yyyymmdd:hh:mm:ss
stat=<*value*>	Filters the output statistical information by statistical information type. You can specify the following in <*value*>: j: I/O information by job c: Information about the number of connections on the communication layer r: Information about request processing

Option	Description
jobid=<*job ID*>	Filters the output statistical information by job ID.
kind=<*value*>[/<*value*>]	Filters output information based on the type of I/O information by job. Only the statistical information specified in this option is output. You can specify multiple types of I/O information by delimiting them with "/". If the option is not specified, all information is output. You can specify the following in <*value*>: io: I/O information meta: Meta-access information rsc: Resource information async: Asynchronous transfer information
type=<*value*>[/<*value*>]	Filters I/O information by job based on the type of usage form. Only the statistical information specified in this option is output. You can specify multiple types of usage form by delimiting them with "/". If the option is not specified, all information is output. You can specify the following in <*value*>: local: Node temporary area information share: Shared temporary area information cache: Second-layer storage cache information
section=<*value*>[/<*value*>]	Filters I/O information by job based on the type of I/O location. Only the statistical information specified in this option is output. You can specify multiple I/O locations by delimiting them with "/". If the option is not specified, all information is output. You can specify the following in <*value*>: Total: Total I/O information CN-CBCN: I/O information between a compute node and the communication buffer CBCN-Dev: I/O information (for communication with a compute node) transferred between the communication buffer and first-layer storage CBGFS-Dev: I/O information (for communication with second-layer storage) transferred between the communication buffer and first-layer storage CBCN-GFS: I/O information (for Direct I/O) transferred between the communication buffer and second-layer storage CBGFS-GFS: I/O information (for Cached I/O) transferred between the communication buffer and second-layer storage

The following example shows output.

```
#Date Time Hostname j JOBID Cache-Total-WR(4K-)-Count Cache-Total-WR(1M-)-Count Cache-Total-WR(4M-)-
Count ... <Omitted>
#Date Time Hostname c Conn InProConn InProDisconn MaxConn AccumConn AccumDisconn
#Date Time Hostname r JOBID IO-Wait-Num IO-Wait-Min IO-Wait-Max IO-Wait-Sum IO-Qdepth-Num IO-Qdepth-
Min ... <Omitted>
20181113 11:40:00 sio-node1 j 744 0 0 0 0 0 0 0 0 0 0 0 0 ... <Omitted>
20181113 11:40:00 sio-node1 c 1 0 0 0 1 0
20181113 11:40:00 sio-node1 r 744 4 3025 7549 17256 4 0 0 0 4 1 1 4 47 2779 5587 152640 47 0 0 0 47 1
1 47
20181113 11:50:00 sio-node1 j 744 0 20 0 0 0 1310720 0 0 0 1310934 0 0 ... <Omitted>
20181113 11:50:00 sio-node1 r 744 39 -422 0 137259 39 0 0 0 39 0 0 39 2 0 0 7070 2 0 0 0 2 0 0 2
(Omitted)
```

From the first to the fifth row, header information is output (an item name is added after "#" at the beginning of a row). On the sixth and subsequent rows, statistical information is output at the timing of information collection.

See

For details on each output item, see "C.3.1 Output Items of System Statistical Information for a Storage I/O Node."

How to Output System Statistical Information for Multiple Storage I/O Nodes

Browsing log files one by one with the collectl command takes a while because system statistical information in the specified job is extracted from the collected log files of system statistical information on multiple storage I/O nodes. By collecting system statistical information from multiple storage I/O nodes onto a single node, the showsiostats command can batch output system statistical information in the specified job from the collected system statistical information. The node with the collected system statistical information is referred to below as the statistical information collection node.

See

For details on the showsiostats command, see "A.2.4 showsiostats Command."

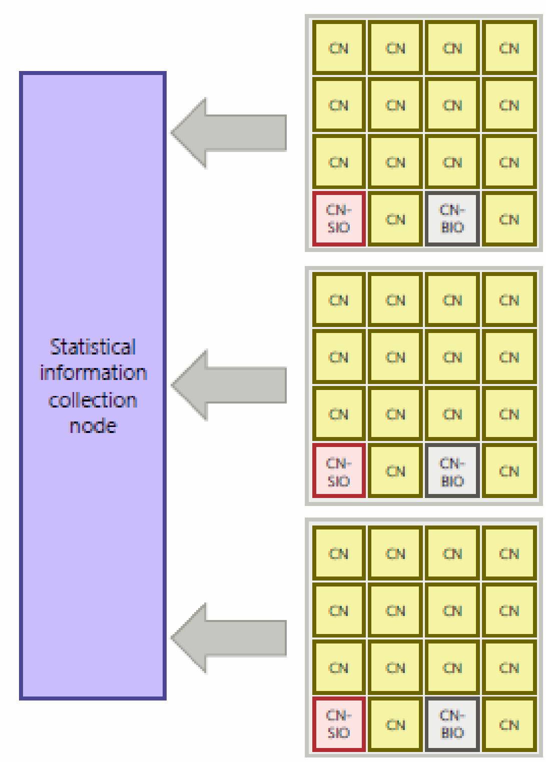
There are two ways to collect system statistical information from multiple storage I/O nodes: one method directly collects log files, and the other method collects system statistical information via nodes. Load balancing can be expected in a system with a large-scale configuration using the method of collecting system statistical information via nodes. The system administrator selects the collection method and whether to use the statistical information collection node or a relay node as appropriate for the system environment. The log collection node and the statistical information collection node are the same when the log collection node is in the system environment, which enables effective checking of logs and statistical information.

Note

The method shown below uses the cron command and rsync command. As a prerequisite, the rsync command without a passphrase can be executed between transferring nodes.

1. Method of directly collecting log files

Figure 3.5 Directly Collecting Log Files



1. For each of the target storage I/O nodes, create a directory for collecting the log files collected by collectl at any location on the statistical information collection node.

```
[Statistical information collection node]
# mkdir -p /var/opt/FJSVllio/rsync/<storage I/O node name>
```

2. For each of the target storage I/O nodes, create a command line for the rsync command.

```
rsync -auv --delete --log-file=<arbitrary log file name> (*)
<BIO IP>:/export/nfsroot/cnode<xxx>/var/opt/FJSVllio/lliostat (*)
/var/opt/FJSVllio/rsync/<storage I/O node name>
```

   Note: The places marked with "(*)" in the above example show line feeds inserted due to space limitations. Actually, the separated output appears on a single line.

   In <*BIO IP*>, specify the MNG_NET IP address displayed by the pashowclst command.

```
[System management node]
# pashowclst -v -l --nodetype BIO | grep "\." |  awk '{print $3}'
```

   In <*xxx*>, specify the CTRL_NET number shown in parentheses by the pashowclst command.

```
[System management node]
# pashowclst -v -l --nodetype SIO | grep "\." | awk -F'(' '{print $2}'| awk -F')' '{print
sprintf("cnode%03d", $1);}'
```

3. Set cron on the statistical information collection node.

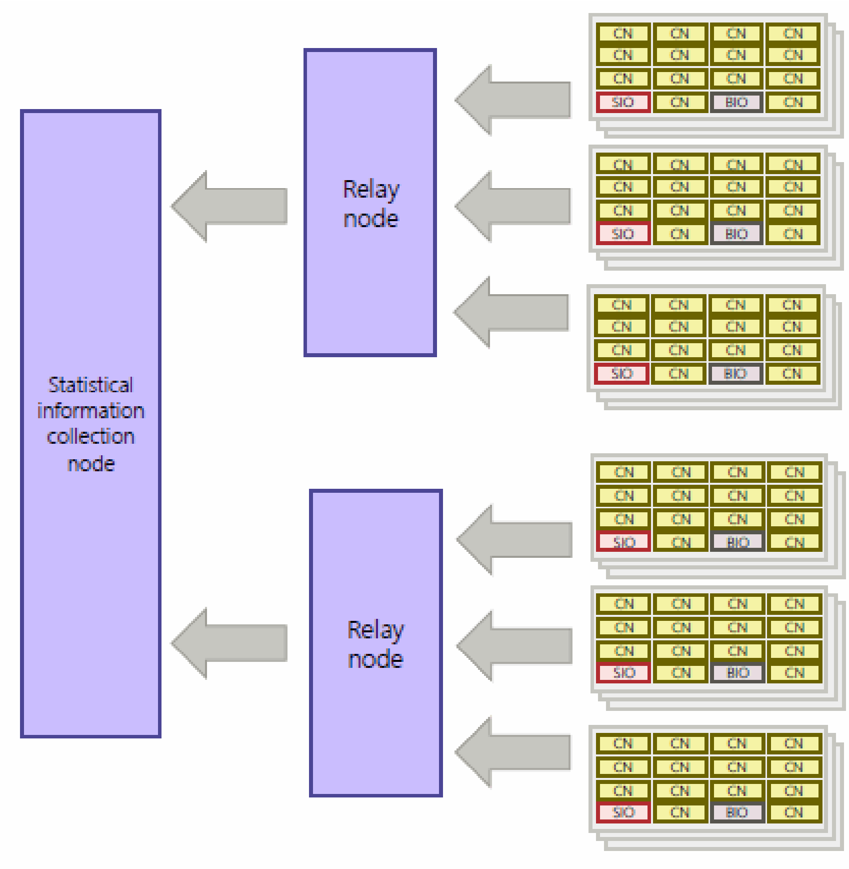
```
[Statistical information collection node]
# crontab -e
```

   - -For parallel execution
     Use the crontab command to set the command lines created in step 2.

   - -For sequential execution
     Create a script with the command lines created in step 2, and set the script in the crontab command.

2. Method of collecting log files via nodes

Figure 3.6 Collecting Log Files via Nodes



1. Create a directory for collecting the log files collected by collectl at an arbitrary location on the statistical information collection node.

```
[Statistical information collection node]
# mkdir -p /var/opt/FJSVllio/rsync
```

2. For each of the target storage I/O nodes, create a directory for collecting the log files collected by collectl at an arbitrary location on the relay node.

```
[Relay node]
# mkdir -p /var/opt/FJSVllio/rsync/<storage I/O node name>
```

3. Create a command line for the rsync command between the relay node and the target storage I/O nodes.

```
rsync -auv --delete --log-file=<arbitrary log file name> (*)
<BIO IP>:/export/nfsroot/cnode<xxx>/var/opt/FJSVllio/lliostat (*)
/var/opt/FJSVllio/rsync/<storage I/O node name>
```

   Note: The places marked with "(*)" in the above example show line feeds inserted due to space limitations. Actually, the separated output appears on a single line.

   In <*BIO IP*>, specify the MNG_NET IP address displayed by the pashowclst command.

```
[System management node]
# pashowclst -v -l --nodetype BIO | grep "\." |  awk '{print $3}'
```

   In <*xxx*>, specify the CTRL_NET number shown in parentheses by the pashowclst command .

```
[System management node]
# pashowclst -v -l --nodetype SIO | grep "\." | awk -F'(' '{print $2}'| awk -F')' '{print
sprintf("cnode%03d", $1);}'
```

4. Create a command line for the rsync command between the statistical information collection node and relay node.

```
rsync -auv --delete --log-file=<arbitrary log file name> /var/opt/FJSVllio/rsync/<storage I/
O node name> <statistical information collection node IP>:/var/opt/FJSVllio/rsync
```

5. Create a script by writing the command lines from steps 3 and 4 in the same order.

```
rsync -auv --delete --log-file=<arbitrary log file name> (*)
<BIO IP>:/export/nfsroot/<cnode>xxx/var/opt/FJSVllio/lliostat (*)
/var/opt/FJSVllio/rsync/<storage I/O node name>
rsync -auv --delete --log-file=<arbitrary log file name> (*)
/var/opt/FJSVllio/rsync/<storage I/O node name> (*)
<statistical information collection node IP>:/var/opt/FJSVllio/rsync
```

   Note: The places marked with "(*)" in the above example show line feeds inserted due to space limitations. Actually, the separated output appears on a single line.

6. Set the script from step 5 in cron on the relay node.

```
[Relay node]
# crontab -e
```

3.2.3.2 Collecting System Statistical Information From a Global I/O Node

Collect system statistical information from a global I/O node with lnetsv.ph, which is a plugin for collectl. The information can be automatically collected regularly when the collectl service is set to start automatically.

How to Collect System Statistical Information From a Global I/O Node

In the following procedure, the system administrator sets the collectl configuration file (/etc/collectl.conf) to collect system statistical information from a global I/O node.

1. Create a collectl.conf file, and write the following DaemonCommands line with settings for collectl daemon start options.

```
[System management node]
# vi collectl.conf
DaemonCommands = -i <systemstat_interval> -f <systemstat_dir> -r<time>,<systemstat_rolllogs> (*)
-m -F<flush_time> -s C --import /opt/FJSVllio/bin/lnetsv.ph
```

   Note: The places marked with "(*)" in the above example show line feeds inserted due to space limitations. Actually, the separated output appears on a single line.

   Referring to "Table 3.8 DaemonCommands Setting Items," set the DaemonCommands line with settings for collectl daemon start options.

2. Distribute the edited collectl.conf file to all global I/O nodes.

```
[System management node]
# pmscatter -c clstname --nodetype GIO ./collectl.conf /etc/collectl.conf
```

3. The collectl service starts automatically.

```
[System management node]
# pmexe -c clstname --nodetype GIO "/usr/bin/systemctl enable collectl"
# pmexe -c clstname --nodetype GIO "/usr/bin/systemctl start collectl"
```

4. When changing the settings during operation, restart the collectl service as follows.

```
[System management node]
# pmexe -c clstname --nodetype GIO "/usr/bin/systemctl restart collectl"
```

How to Output System Statistical Information for a Global I/O Node

Use a log file collected by collectl to output system statistical information for a global I/O node.

This section shows how to output system statistical information for a global I/O node.

```
[Global I/O node]
# collectl -p <data file> -oD -s-C --plot [<collectl option>] --import /opt/FJSVllio/bin/
lnetsv.ph[,stat=<value>]
```

In <*data file*>, specify a log file collected by collectl on the global I/O node.

You can specify a separator between output items and an output time range in the collectl command options.

You can specify the following collectl extension plugin options for system statistical information for a global I/O node.

Table 3.10 collectl Extension Plugin for Global I/O Nodes

Option	Description
stat=<*value*>	Filters the output statistical information by data type. You can specify the following in <*value*>: s: Information on the number of data transfers and data amount b: Transfer buffer information

The following example shows output.

```
#Date Time TransferCount TransferAmount MinFreeBufZero MinBufSend MinFreeBufRDMA
20181113 23:05:00 216 108288 2047 1531 96
20181113 23:10:00 284 59072 2048 1531 96
20181113 23:15:00 268 55744 2048 1531 96
(Omitted)
```

For details on each output item, see "C.3.2 Output Items of System Statistical Information for a Global I/O Node."

How to Output System Statistical Information for Multiple Global I/O Nodes

To collect system statistical information from multiple global I/O nodes onto a single node, see "How to Output System Statistical Information for Multiple Storage I/O Nodes." Then, replace <storage I/O node name> with <global I/O node name> to collect the information.

3.3 Maintenance

3.3.1 Changing the LLIO Configuration

For an LLIO configuration change that may involve hardware maintenance, such as an SIO configuration change, see the *Administrator's Guide for Maintenance* manual. Then, perform the following procedure.

1. Creating an FEFS design sheet
   Create an FEFS design sheet according to "3.1.3 Creating an FEFS Design Sheet."

2. Stopping the LLIO services
   Stop the LLIO services running on storage I/O nodes and compute nodes. Perform this operation on the active system management node.

   ```
   [System management node]
   # fefs_sync --stop --compute=<cluster>[,...] --llio
   ```

   Specify all the clusters in the LLIO configuration.

3. Rebuilding LLIO
   Perform the following work according to "3.1.4 Building LLIO":

   - -Create a configuration definition file for the FEFS setup tool

   - -Place the configuration definition file for the FEFS setup tool

4. Starting the LLIO services
   Start the LLIO services on storage I/O nodes and compute nodes. Perform this operation on the active system management node.

   ```
   [System management node]
   # fefs_sync --start --compute=<cluster>[,...] --llio
   ```

   Specify all the clusters that will build LLIO.

3.3.2 Rolling Update

You can update the LLIO packages without stopping the entire system.

Note

The packages contain information describing whether rolling updates are supported. Check the conditions and the information, including whether the work is supported, in advance. Use the rpm -qi command for an already applied package or the rpm -qpi command for a package scheduled to be applied. For details, see "Software Maintenance" in the *Job Operation Software Administrator's Guide for Maintenance* manual.

Perform these operations in units of SIO groups.

1. Preparation
   Perform the operations for isolation from the operation and switching to software maintenance mode.

   Information

   For details, see "Advance Preparation for Software Maintenance" in the *Job Operation Software Administrator's Guide for Maintenance* manual.

2. Stopping the LLIO services in the target range
   Perform the following operation on the active system management node.

```
[System management node]
# fefs_sync --stop --compute=<cluster> --llio --nodeid=<nodeid> --siogrp
```

   --compute: Specify a compute cluster name.

   --nodeid: Execute the command on the SIO group containing <*nodeid*>.

   Note

   Within the nodes in the specified range, do not perform operations for the nodes where the LLIO packages are not applied. (Example: LLIO service operation, etc. for the compute sub cluster management node when --nodegrp (node group range) is specified in the range specification options)

3. Maintenance work
   Apply the package in the range of the services stopped in step 1.

4. Starting the LLIO services in the target range
   Perform the following operation on the active system management node.

```
[System management node]
# fefs_sync --start --compute=<cluster> --llio --nodeid=<nodeid> --siogrp
```

   --compute: Specify a compute cluster name.

   --nodeid: Execute the command on the SIO group containing <*nodeid*>.

5. Incorporating into operation
   Incorporate the maintenance targets isolated in the preparation into operation.

   See

   For details, see "Incorporating Into Operation After Software Maintenance" in the *Job Operation Software Administrator's Guide for Maintenance* manual.

See

For details on how to apply the LLIO package and the FEFS client package together, see the "FEFS User's Guide" manual.

3.3.3 Troubleshooting

Collect the following materials when trouble occurs during operation.

Table 3.11 Materials Required When Trouble Occurs

Type of Material Collected	Target Node	Collected File/Collection Command
System log	All nodes	/var/log/messages*
PANIC DUMP	Nodes where DUMP is collected	Dump files collected by the padumpmgr command
System material	All nodes	OS investigation materials collected by the pasnap command
FEFS material	All nodes	<outputdir>/fefssnap_<time stamp>.tgz file created when the following command is executed: # /usr/sbin/fefssnap -d <outputdir> * <time stamp> is the command execution time (yyyymmddHHMMSS). FEFS investigation materials collected by the pasnap command

Type of Material Collected	Target Node	Collected File/Collection Command
LLIO material	All nodes	<outputdir>/lliosnap_<time stamp>.tgz file created when the following command is executed: # /usr/sbin/lliosnap -d <outputdir> * <time stamp> is the command execution time (yyyymmddHHMMSS). LLIO investigation materials collected by the pasnap command (--direct option specified)

See

- -The lliosnap command is executed when the fefssnap command is executed. For details on the lliosnap command, see "A.2.2 lliosnap Command."

- -For details on the fefssnap command, see the *FEFS User's Guide* manual.

- -For details on how to collect the materials of the pasnap command and how to collect dump files with the padumpmgr command, see the *Job Operation Software Administrator's Guide for Maintenance* manual.

Information

- -If the FEFS service does not have a problem, FEFS materials do not need to be collected.

- -fefs.log is an FEFS internal log that may be required for an investigation of a problem. Depending on the situation, fefs.log may not exist. Execute the ps command on the applicable node to investigate whether nothing was output to fefs.log or log output stopped. Running the following command shows that logs can be collected.

```
[Applicable node]
# ps -ef | grep fefslog
lctl fefslog start /var/opt/FJSVfefs/fefs.log
```

3.3.4 Files and directories in the shared temporary area and cache area of second-layer storage remaining in the second-layer storage

This section describes how to check and delete files and directories in the shared temporary area and cache area of second-layer storage remaining in the second-layer storage due to node down, etc. Files and directories created in the shared temporary space are created on the second-layer storage as well as the first-layer storage. Cache area of second-layer storage may leave deleted files on the second-layer storage temporarily. They are usually deleted at the end of the job, but should be deleted because they may remain on the second-layer storage due to node down, etc.

3.3.4.1 How to check for remaining files and directories

The remaining files and directories may exist under the following four directories "*<Job ID>*".

```
<Mount path of the second-layer storage(*1)>/.llio_share/<job000-1ff(*2)>/<jobID>
                                            /.llio_sillyrename/<job000-1ff(*2)>/<jobID>
<Mount path of the second-layer storage(*1)>/.llio_share/trash/<SIO Host Name(*3)>/<jobID>
                                            /.llio_sillyrename/trash/<SIO Host Name(*3)>/<jobID>
```

*1 : If multiple the second-layer storage are mounted, check each one.
*2 : Directories for each job are distributed among 512 directories, job000 through job1ff.
*3 : *<SIO Host Name>* is the host name of a storage I/O node.

3.3.4.2 How to delete remaining files and directories

The remaining files and directories should be deleted by the root user on the login node.
However, the following directories should not be deleted during job operation. They can be deleted during maintenance.

```
<Mount path of the second-layer storage>/.llio_share/<job000-1ff>
<Mount path of the second-layer storage>/.llio_share/<job000-1ff>/<jobID>
<Mount path of the second-layer storage>/.llio_sillyrename/<job000-1ff>
<Mount path of the second-layer storage>/.llio_sillyrename/<job000-1ff>/<jobID>
<Mount path of the second-layer storage>/.llio_share/trash/<SIO Host Name>
<Mount path of the second-layer storage>/.llio_sillyrename/trash/<SIO Host Name>
```

The following directories can be deleted even during job operation.

```
<Mount path of the second-layer storage>/.llio_share/trash/<SIO Host Name>/<jobID>
<Mount path of the second-layer storage>/.llio_sillyrename/trash/<SIO Host Name>/<jobID>
```

3.4 Precautions

File locking

Effective range of file lock

The effective range when using the LLIO file lock function is as follows.

- Node temporary area :
  Same as namespace range (within one CN)
- Shared temporary area :
  Same as namespace range (within job)
- Cache area of second-layer storage :
  Unlike namespace ranges, file lock is effective within a job. In other words, if you try to acquire a conflicting lock on the same file from more than one job, all jobs can acquire exclusiveness.

Support for mandatory lock

LLIO, like NFS, does not support mandatory lock.

Interaction between fcntl and flock

When using the file lock function of LLIO, the interaction between fcntl and flock has the following specifications for each file system.

- Node temporary area :
  According to Linux specifications, fcntl and flock lock do not interact with each other. In other words, even if a process that acquires an exclusive lock with fcntl and a process that acquires an exclusive lock with flock for the same file exist at the same time, the lock acquisition of either process succeeds.
- Shared temporary area :
  According to the NFS specification, fcntl and flock lock interact with each other.
- Cache area of second-layer storage :
  According to the NFS specification, fcntl and flock lock interact with each other.

Inhibiting access by updatedb command

When mlocate has been implemented, the updatedb command may be executed periodically, thereby causing unintentional accessing of the LLIO file system.

To inhibit accessing of the LLIO file system with the updatedb command, make the following setting to exclude the LLIO file system as a search target.

[Setting]

To exclude the LLIO file system as a search target of the updatedb command, add "lliofs" to "PRUNEFS" of the "/etc/updatedb.conf" file on the LLIO clients.

ex) Setting example for /etc/updatedb.conf

```
PRUNEFS = "auto afs gfs gfs2 iso9660 lliofs"
                                     ~~~~~~
```

Addressing the specification of the metadata consistency between client nodes

On the cache area of second-layer storage or the shared temporary area, if file_A is deleted or renamed another name and then recreated on compute node A, on compute node B, open(2) for file_A may fail or may open the file which is deleted or renamed on compute node A.

This specification can be addressed as follows.

- Execute ls command on compute node B against the parent directory of file_A before opening file_A.
- Wait 60 seconds after recreating file_A on compute node A before opening file_A on compute node B.

Appendix A Reference

A.1 System Calls

The following table shows the types of system calls supported by LLIO.

System Calls Supported by LLIO

System Call	Supported?
_llseek	Yes
access	Yes
bdflush	-
chdir	Yes
chmod	Yes
chown	Yes
chown32	Yes
chroot	Yes
close	Yes
creat	Yes
dup	-
dup2	-
execve	Yes
fchdir	Yes
fchmod	Yes
fchown	Yes
fchown32	Yes
fcntl	Partly Only advisory lock supported (mandatory lock not supported)
fcntl64	Partly Only advisory lock supported (mandatory lock not supported)
fdatasync	Yes
fgetxattr	Yes
flistxattr	Yes
flock	Yes
fremovexattr	Yes
fsetxattr	Yes
fstat	Yes
fstat64	Yes
fstatfs	Yes
fstatfs64	Yes
fsync	Yes
ftruncate	Yes

System Call	Supported?
ftruncate64	Yes
getdents	Yes
getdents64	Yes
getxattr	Yes
ioctl	Partly Only striping function supported
lchown	Yes
lchown32	Yes
lgetxattr	Yes
link	Yes
listxattr	Yes
llistxattr	Yes
lremovexattr	Yes
lseek	Yes
lsetxattr	Yes
lstat	Yes
lstat64	Yes
mkdir	Yes
mknod	Yes
mmap	Yes
mount	Yes
munmap	-
open	Yes
pipe	-
pivot_root	No
pread64	Yes
pwrite64	Yes
read	Yes
readdir	Yes
readlink	Yes
readv	Yes
removexattr	Yes
rename	Yes
rmdir	Yes
setrlimit	-
setxattr	Yes
stat	Yes
stat64	Yes
statfs	Yes
statfs64	Yes

System Call	Supported?
swapon	No
swapoff	No
symlink	Yes
sync	-
sysfs	No
truncate	Yes
truncate64	Yes
umount	Yes
umount2	Yes
unlink	Yes
utime	Yes
utimes	Yes
write	Yes
writev	Yes

```
Yes: Supported  Partly: Only some functions supported  -: Supported at VFS level  No: Not supported
TBD: Operation not guaranteed
```

A.2 Commands

A.2.1 lfs Command

lfs getstripe

NAME

lfs getstripe - Displays the stripe pattern for second-layer storage from compute nodes.

SYNOPSIS

```
/usr/local/bin/lfs getstripe <dirname | filename>
```

DESCRIPTION

The lfs getstripe command displays information on the stripe pattern of the specified file or directory. Specify <*dirname*> to display the stripe pattern of the directory. Specify <*filename*> to display the stripe pattern of the file.

lfs setstripe

NAME

lfs setstripe - Sets the stripe pattern for second-layer storage from compute nodes.

SYNOPSIS

```
/usr/local/bin/lfs setstripe [option] <dirname | filename>
```

DESCRIPTION

The lfs setstripe command creates a new file that will have the stripe pattern or sets the stripe pattern in an existing directory.

Specify <*dirname*> to set the stripe pattern in the directory. Specify <*filename*> to set the stripe pattern in the file.

You can specify stripe sizes and stripe counts between upper and lower limits. For the upper and lower limit values, check the second-layer storage specifications.

OPTIONS

[--stripe-size | -S stripe_size]

This option sets the stripe size.

You can set a size in units of KiB, MiB, or GiB by using -S #k, -S #m, or -S #g, respectively.

[--stripe-count | -c stripe_count]

This option sets the stripe count.

If -1 is set, data is written to all OSTs.

A.2.2 lliosnap Command

NAME

lliosnap - Collects the materials necessary for an investigation of first-layer storage.

SYNOPSIS

```
/usr/sbin/lliosnap -d <outputdir>
/usr/sbin/lliosnap --help
```

DESCRIPTION

The lliosnap command collects the necessary materials for troubleshooting.

The collected data is compressed in tar + gzip format and stored in a directory specified with the following name:

lliosnap_<time stamp>.tgz

* Time stamp: yyyymmddHHMMSS

Only users with administrator privileges can use this command.

OPTIONS

-d <outputdir>

This option specifies the storage directory for collected materials.

The option is a mandatory option when collecting materials.

--help

This option displays an explanation of usage and exits.

END CODE

The following end codes are returned:

0: Normal end

1: Abnormal end

A.2.3 llio_transfer Command

NAME

llio_transfer - Distributes or deletes a common file.

SYNOPSIS

```
Distribution operation (asynchronous mode):
/usr/bin/llio_transfer <path> [<path> ...]

Distribution operation (synchronous mode):
/usr/bin/llio_transfer --sync <path> [<path> ...]
```

```
Deletion operation:
/usr/bin/llio_transfer --purge <path> [<path> ...]

Displaying how to use:
/usr/bin/llio_transfer --help
```

DESCRIPTION

The llio_transfer command is used in the job script and distributes a common file on second-layer storage to the cache area of second-layer storage, which is on first-layer storage.
Using an option, this command also deletes a distributed common file from the cache area of second-layer storage, which is on first-layer storage.
<*path*> specifies the absolute or relative path to the common file on second-layer storage.
The common file is not striped across storage I/O nodes.
If a symbolic link file is specified, the file at the link destination is distributed to the cache area of second-layer storage, which is on first-layer storage.
If multiple files are specified, they are processed in order from first to last. Even if the processing to distribute or delete a file fails partway through, the command continues until the last file unless a serious error occurs, in which case the command cannot continue.
For a distribution operation (asynchronous mode), the command instructs the user to start distributing the file specified in <*path*> and the llio_transfer command returns.
For a distribution operation (synchronous mode), the llio_transfer command returns after distribution of the file specified in <*path*> is complete.

OPTIONS

--sync

This option distributes common files in synchronous mode.

--purge

This option deletes a distributed common file from the cache area of second-layer storage, which is on first-layer storage.

--help

This option outputs a message on how to use the command to the standard output, and ends normally.

Note

Files copied to the second-layer storage cache using the llio_transfer command are called common files. Note the following about common files:

- Deleting common files, or updating their data or attributes fail with an error via the cache area of the second-layer storage.
- File locking operations for common files are not supported.
- During job execution, do not change or delete the copy source file on second-layer storage. If the file is changed, the accuracy of the file contents copied to the second-layer storage cache cannot be guaranteed. Also, if it is deleted, the common file cannot be deleted with the --purge option of the llio_transfer command, and the cache space of the second-layer storage remains occupied by the common file until the job is finished.
- To delete a common file in the job script, specify the --purge option for the llio_transfer command. The rm command against the common file fails and it cannot be deleted.
- Shortly after deleting common files with llio_transfer --purge, operations to delete the copy source files or update their data or attributes may fail with an error. If this occurs, wait for 60 seconds, and then retry the operations.
- The "cache area of second-layer storage" located at the first-layer storage, which is the common file copy destination, must have enough space to store the entire common file. Common files copied to a cache area of second-layer storage are not deleted when this area is full. If you need space in the cache area of the second-layer storage, use the llio_transfer command with the --purge option to remove common files from the cache area during the job. Note that the common file in the cache area of the second-layer storage is automatically deleted after the job ends.

- If the cache data for the transfer source file exists in the first-layer storage before the command allocates the common file area in the job, the llio_transfer command fails. Do not open the file in the job before executing the llio_transfer command because the cache data may be created when the file is opened. Do not use the following commands, as they may also open files.

  - lfs setstripe

  - lfs getstripe

  Note that if a job continues with failure of the llio_transfer command, the job executes with no common files deployed.

- Files that can be specified in the llio_transfer command must meet the following conditions.

  - A file that can be accessed by the user executing this command, and

  - A file in the second-layer storage, and

  - A file type of a regular file, and

  - A file size of 1 byte or more

END CODE

The following end codes are returned:

0: Normal end

1: Abnormal end

If more than one file is specified, 0 is returned if all the files have been distributed or deleted successfully.

A.2.4 showsiostats Command

NAME

showsiostats - Outputs system statistical information related to a job from the log file of system statistical information.

SYNOPSIS

```
/opt/FJSVllio/bin/showsiostats -d <dirname> -j <jobid> [--from=<value>] [--stat=<value>]
                                  [--kind=<value>] [--type=<value>] [--section=<value>]
                               --help
```

DESCRIPTION

The showsiostats command outputs system statistical information related to the job specified in <jobid> from the log file of system statistical information for a storage I/O node under the directory specified in <dirname>.

Only users with system administrator privileges on the node that collected the system statistical information can use this command.

OPTIONS

-d <dirname>

This option specifies the directory to search for the source log file for extraction. The search range also includes the subdirectories of the specified directory.

-j <jobid>

This option specifies the job ID that is the extraction target.

--from=<value>

This option filters output information by date and time of statistical information. You can specify the following value in <value>:

yyyymmdd:hh:mm:ss-yyyymmdd:hh:mm:ss: Specified time range

--stat=<value>

This option filters output information by type of statistical information. You can specify the following value in <value>:

j: I/O information per job
c: Information related to the number of connections on the communication layer
r: Information related to request processing

--kind=<value>

This option filters output information by type of I/O information per job. Only the statistical information specified in the option is output.

You can specify multiple types by delimiting them with "/". If the option is not specified, all types of information are output.

You can specify the following value in <value>:

io: I/O information
meta: Meta-access information
rsc: Resource information
async: Asynchronous transfer information

--type=<value>

This option filters I/O information per job by type of usage form. Only the statistical information specified in the option is output. You can specify multiple types by delimiting them with "/". If the option is not specified, all types of information are output.

You can specify the following value in <value>:

cache: Cache information for second-layer storage
share: Information in the shared temporary area
local: Information in the node temporary area

--section=<value>

This option filters I/O information per job by type of I/O location. Only the statistical information specified in the option is output. You can specify multiple types by delimiting them with "/". If the option is not specified, all types of information are output.

You can specify the following value in <value>:

Total: Total I/O information
CN-CBCN: I/O information between a compute node and the communication buffer
CBCN-Dev: I/O information (for communication with a compute node) transferred between the communication buffer and first-layer storage
CBGFS-Dev: I/O information (for communication with second-layer storage) transferred between the communication buffer and first-layer storage
CBCN-GFS: I/O information (for Direct I/O) transferred between the communication buffer and second-layer storage
CBGFS-GFS: I/O information (for Cached I/O) transferred between the communication buffer and second-layer storage

--help

This option displays how to use this command.
Arguments and also other options are all invalid when this option is specified.

END CODE

The following end codes are returned:

0: Normal end

1: Abnormal end

A.3 Library

A.3.1 getlliostat

SYNOPSIS

```
#include <llio_stat.h>

getlliostat (lliostat_t * lliostat);
```

DESCRIPTION

getlliostat sets and returns the statistical information managed by first-layer storage in the lliostat argument on compute nodes.

Every time the library is called, the following two items return information: one returns the total cumulative value from the job start time until the function execution time, and the other returns the current value.

For example, an item that returns usage (unused cache space, occupied cache space, and cache space with nothing written) acquires values at the library execution time.

Other examples include an item that returns times and counts (total number of writes, total amount of write transfers, and total write processing time), which outputs the total cumulative value from the job start time.

Each item of statistical information is managed as the uint64_t type, and counts from 0 after an overflow.

ARGUMENTS

lliostat_t *lliostat

This argument specifies the structure for storing statistical information. For structure definitions, "C.2 Collection Methods and Output Items of Compute Node Statistical Information."

END CODE

The following end codes are returned:

0: Normal end

1: Abnormal end

LIBRARY PATH

/opt/FJSVllio/lib/liblliostat.so

To execute the job, specify /opt/FJSVllio/lib in LD_LIBRARY_PATH.

INCLUDE HEADER FILE PATH

/opt/FJSVllio/include/llio_stat.h

Appendix B Messages

B.1 Messages Output to the System Log

The messages output by LLIO to the system log have the following format.

```
[ERR.] [LLIO] 0001 jobid lliofs Job is already running.
(1)     (2)   (3)  (4)   (5)    (6)
```

1. Message type

   The message type represents the message output level. The message type is associated with the message ID shown at 3. The meaning of the contents is as follows:

   - [ERR.]: ERROR message (0001 to 5999)
   - [WARN]: WARNING message (6000 to 6999)
   - [NOTE]: NOTICE message (7000 to 7999)
   - [INFO]: INFO message (8000 to 8999)

2. LLIO prefix

   The LLIO prefix is an identifier showing that the output message relates to first-layer storage.

   The meaning of the contents is as follows:

   - [LLIO]: The message is output from the primary first-layer storage.

3. Message ID

   The message ID identifies the message. The message ID is associated with the message type shown at 1.

4. Job ID

   Job ID. "-" is output when the message does not have an applicable job ID.

5. Subcomponent name

   LLIO subcomponent name

6. Message contents

   Contents of the message

[ERROR Messages] (0001 to 5999)

[ERR.] [LLIO] 0001 *jobid* lliofs Job is already running.

Meaning

A job with first-layer storage specified is being executed.

Action

Job execution has possibly begun for multiple jobs that use first-layer storage and have the same job ID. Contact your Fujitsu system engineer (SE) or the Fujitsu Support Desk.

[ERR.] [LLIO] 0002 *jobid* lliofs Unable to unmount(*mountpoint*).

Meaning

Processing to unmount first-layer storage failed.

mountpoint: Mount point

Action

Contact your Fujitsu system engineer (SE) or the Fujitsu Support Desk.

[ERR.] [LLIO] 0005 *jobid* lliofs Cannot allocate memory.

[ERR.] [LLIO] 1000 *jobid* lliosv Cannot allocate memory.

[ERR.] [LLIO] 2000 *jobid* lliost Cannot allocate memory.

[ERR.] [LLIO] 2300 *jobid* lliolog Cannot allocate memory.

[ERR.] [LLIO] 2600 *jobid* lliosv_qos Cannot allocate memory.

[ERR.] [LLIO] 2800 *jobid* lliorpc Cannot allocate memory.

[ERR.] [LLIO] 2900 *jobid* lliolib Cannot allocate memory.

Meaning

Acquisition of the necessary memory for processing a first-layer storage module failed.

Action

Check the conditions for memory use on the node where the message was output. If the cause is unknown, contact your Fujitsu system engineer (SE) or the Fujitsu Support Desk.

[ERR.] [LLIO] 0007 jobid lliofs Mount(*path*) failed(*reason*).

Meaning

Mount to *path* failed for *reason*.

Action

If *reason* is as follows, check the specified mount option.

"unknown option"

An unsupported option is specified.

"incompatible filesystem type for lazystatfs option"

The lazystatfs option is specified other than shared temporary.

"invalid min/max relationship"

An invalid value is specified for the option that sets the maximum and minimum values.

"too long JOBID"

The specified JOBID string is too long.

"bad option value for attr_dir_min"

"bad option value for attr_dir_max"

"bad option value for attr_reg_min"

"bad option value for attr_reg_max"

The value specified in the option argument is invalid.

"lack of mandatory option"

The JOBID option is not specified.

"unknown filesystem type"

A file system other than local/share/global is specified as the file system to be mounted.

"invalid specified format for global filesystem"

The method for specifying the mount target of global is invalid.

"JOB not found"

The specified job has not been started.

If *reason* is as follows, contact a Fujitsu systems engineer (SE) or Fujitsu SupportDesk.

"insufficient memory"

A memory shortage occurred.

"system error"

An internal error has occurred.

[ERR.] [LLIO] 1001 *jobid* lliosv System error(*detail*).
[ERR.] [LLIO] 2001 *jobid* lliost System error(*detail*).
[ERR.] [LLIO] 2301 *jobid* lliolog System error(*detail*).
[ERR.] [LLIO] 2400 *jobid* llio_transfer System error(*detail*).
[ERR.] [LLIO] 2901 *jobid* lliolib System error(*detail*).

Meaning

A system error occurred in processing of a first-layer storage module.

detail: Detailed maintenance information

Action

Contact your Fujitsu system engineer (SE) or the Fujitsu Support Desk.

[ERR.] [LLIO] 1100 *jobid* lliosv Module load error. Specified(*filesystem*) mount point(*mountpoint*) is not an actual mount point.

Meaning

Loading of a first-layer storage module failed. The specified path in the design sheet is not a mount point.

filesystem: File system name. The output is either of the following:

2nd_layer storage: Mount point on second-layer storage

shared temporary namespace file system: Mount point on the file system for namespace management of the shared temporary area

mountpoint: Specified mount point

Action

Confirm that the mount point specified in the FEFS design sheet is correct.

[ERR.] [LLIO] 1101 *jobid* lliosv Module load error. Specified(*filesystem*) mount point(*mountpoint*) does not exist.

Meaning

Loading of a first-layer storage module failed. The specified path in the design sheet is a directory that does not exist in the file system.

filesystem: File system name. The output is either of the following:

2nd_layer storage: Mount point on second-layer storage

shared temporary namespace file system: Mount point on the file system for namespace management of the shared temporary area

mountpoint: Specified mount point

Action

Confirm that the mount point specified in the FEFS design sheet is correct.

[ERR.] [LLIO] 2002 *jobid* lliost 1st_layer storage device mkfs error(dev=*devname*).

Meaning

mkfs failed for the first-layer storage device.

devname: Device name

Action

Confirm that the device can be used. If the cause is unknown, contact your Fujitsu system engineer (SE) or the Fujitsu Support Desk.

[ERR.] [LLIO] 2200 *jobid* lliost The state of the 1st_layer storage device(*SSD_path*) is abnormal(*critical_warning*).

Meaning

The first-layer storage device is not in a normal state.

SSD_path: Path name of the first-layer storage device

critical_warning: *critical warning* value. The displayed value is any of the following values or a value obtained using OR.

Value	Description
0x02	Abnormal temperature
0x04	Reliability decreased
0x08	Transitioned to read-only mode
0x10	Abnormal backup device

Action

Check the status of first-layer storage. Perform maintenance as needed.

[WARNING Messages] (6000 to 6999)

[WARN] [LLIO] 6300 *jobid* lliost The available spare space of the 1st_layer storage device(*SSD_path*) is below the hardware threshold(current=*avail_spare*%, threshold=*spare_thresh*%).

Meaning

The remaining space in the reserved area on the first-layer storage device is under the hardware threshold.

SSD_path: Path name of the first-layer storage device

avail_spare: *avail_spare* value at message output

spare_thresh: Value of the threshold stipulated for hardware with regard to *avail_space*

Action

Perform maintenance as needed to prevent the reserved area from being depleted.

[WARN] [LLIO] 6400 *jobid* llio_transfer Failed to transfer common file(*path*).

Meaning

Transfer of the file specified in an argument of the llio_transfer command failed.

One of the following operations was possibly performed during common file transfer:

- Deleting a file on second-layer storage
- Updating a file on second-layer storage
- Executing the llio_transfer command with the --purge option specified in the same job script

path: Path to the file that failed to be transferred

Action

Confirm that the file exists on second-layer storage.
While transferring a common file, do not update the file on the second-layer storage. Also, while transferring a common file, do not execute the llio_transfer command with the --purge option specified in the same job script.

[NOTICE Messages] (7000 to 7999)

[NOTE] [LLIO] 7350 *jobid* lliolog Starting lliolog daemon.

Meaning

The log daemon of first-layer storage is starting.

Action

No action is necessary.

[NOTE] [LLIO] 7351 *jobid* lliolog Shutting down lliolog daemon.

Meaning

The log daemon of first-layer storage is terminated.

Action

No action is necessary.

[INFO Messages] (8000 to 8999)

[INFO] [LLIO] 8300 *jobid* lliost The available spare space of the 1st_layer storage device(*SSD_path*) is below the software threshold(current=*avail_spare*%, threshold=*spare_thresh*%).

Meaning

This message shows the remaining space in the reserved area on the first-layer storage device.

SSD_path: Path name of the first-layer storage device

avail_spare: *avail_spare* value at message output

spare_thresh: Value of the threshold stipulated for software with regard to *avail_space*

Action

No action is necessary.

B.2 Messages Output by Commands

If an error occurs when a command is being executed, the command outputs one of the messages below to the standard error output. The messages output by commands have the following format.

```
[ERR.] LLIO 2750 lliosnap -d: Outputdir: No such directory.
(1)    (2)  (3)  (4)      (5)
```

1. Message type

   The message type represents the message output level. The message type is associated with the message ID shown at 3. The meaning of the contents is as follows:

   - -[ERR.]: ERROR message (0001 to 5999)
   - -[WARN]: WARNING message (6000 to 6999)
   - -[NOTE]: NOTICE message (7000 to 7999)
   - -[INFO]: INFO message (8000 to 8999)

2. LLIO prefix

   The LLIO prefix is an identifier showing that the output message relates to first-layer storage.

   The meaning of the contents is as follows:

   - -LLIO: The message is output from the primary first-layer storage.

3. Message ID

   The message ID identifies the message. The message ID is associated with the message type shown at 1.

4. Command name

   Command name

5. Message content

   Contents of the message

B.2.1 lfs Command

Common

[ERR.] LLIO 3150 lfs : <*command*> is not supported.

Meaning

The specified subcommand is not supported.

Description of parameters

command : Specified subcommand

Action

The only subcommands that can be specified are setstripe and getstripe.
Check the specified subcommand.

lfs setstripe

[ERR.] LLIO 3151 lfs setstripe: Missing filename|dirname.

Meaning

A file name or directory name is not specified.

Action

Specify a file name or directory name.

[ERR.] LLIO 3152 lfs setstripe: Bad stripe size(*size*).

Meaning

The stripe size specified in the -S option is inappropriate.

size: Specified stripe size value

Action

Review the specified stripe size value.

[ERR.] LLIO 3153 lfs setstripe: <*path*> is not on a global file system cache.

Meaning

The specified path is not on the second-layer storage cache.

path : specified path name

Action

Check the specified path name.

[ERR.] LLIO 3154 lfs setstripe: Bad stripe size(*size*), must be multiple of 65536 bytes Invalid argument(22).

Meaning

The specified stripe size must be a multiple of 65,536 bytes.

size: Specified stripe size value

Action

Specify a multiple of 65,536 bytes for the stripe size.

[ERR.] LLIO 3155 lfs setstripe: Stripe size 4G or larger is not currently supported and would wrap Invalid argument (22).

Meaning

The specified value for the stripe size must be less than or equal to 4,194,240 KiB (4 GiB-64 KiB).

Action

Specify 4,194,240 KiB (4 GiB-64 KiB) or less for the stripe size.

[ERR.] LLIO 3156 lfs setstripe: bad stripe count(*count*).

Meaning

The stripe count specified in the -c option is inappropriate.

Count: Specified stripe count value

Action

Review the specified stripe count value.

[ERR.] LLIO 3157 lfs setstripe: Unable to open(*path*) *errmsg*(*err*).

Meaning

The file or directory specified in *path* could not be opened.

path: Specified path name
errmsg: Error message
err: Error code

Action

Review the specified path.

[ERR.] LLIO 3158 lfs setstripe: ioctl() failed for(*path*) *errmsg*.

Meaning

ioctl() failed to set stripe information.

path: Specified path name
errmsg: Error message

Action

If the error message is "stripe already set," a file already exists in the specified path. Review the path name, and try again.

For cases other than the above, contact your Fujitsu system engineer (SE) or the Fujitsu Support Desk.

[ERR.] LLIO 3159 lfs setstripe: setstripe failed <*errmsg*>.

Meaning

Setting stripe information failed.

errmsg: Error message

Action

Take action according to the message output in the error message.

lfs getstripe

[ERR.] LLIO 3160 lfs getstripe: Failed for(<*path*>).

Meaning

The directory name or file name is not specified correctly.

path: Specified path name

Action

Check the specified directory name or file name.

[ERR.] LLIO 3161 lfs getstripe: ioctl() failed for(<*path*>) <*errmsg*>.

Meaning

ioctl() failed to acquire stripe information.

Path: Specified path name
errmsg: Error message

Action

If the error message is "<*path*> is not on a global file system cache", the specified path name is not a file on the second-layer storage cache. Review the path name, and try again.

If the error message is "Stripe information is not set", set the stripe information.

For cases other than the above, contact your Fujitsu system engineer (SE) or the Fujitsu Support Desk.

[ERR.] LLIO 3162 lfs getstripe: getstripe failed <*errmsg*>.

Meaning

Acquisition of stripe information failed.

errmsg: Error message

Action

Take action according to the message output in the error message.

B.2.2 lliosnap Command

[ERR.] LLIO 2750 lliosnap -d: (*Outputdir*) No such directory.

Meaning

The specified directory was not found.

Outputdir: Directory path specified in the -d option

Action

Review the value specified in the -d option.

[ERR.] LLIO 2752 lliosnap Exist temporary directory(*Tmpdir*).

Meaning

The work area already exists.

Tmpdir: Path to the directory of the work area

Action

The directory path written in *Tmpdir* is the path to a directory used as a work area by this command. Change the directory path written in *Tmpdir* to another path name.

[ERR.] LLIO 2753 lliosnap Cannot create temporary directory(*Tmpdir*).

Meaning

The work area could not be created.

Tmpdir: Path to the directory of the work area

Action

Check the status to confirm that the work area can be created.

[ERR.] LLIO 2754 lliosnap Cannot create output file.

Meaning

An output file could not be created.

Action

Check the status of the output destination directory.

B.2.3 llio_transfer Command

[ERR.] LLIO 2450 llio_transfer Not enough disk space.

Meaning

The free capacity of first-layer storage available to jobs is insufficient.

Action

Increase the capacity of first-layer storage available to jobs.

[ERR.] LLIO 2451 llio_transfer Command is already running.

Meaning

The llio_transfer command is being executed.

Action

Confirm that the llio_transfer command is not being executed.

[ERR.] LLIO 2452 llio_transfer System error(*detail*).

Meaning

A system error occurred in processing with the llio_transfer command.

detail: Detailed maintenance information

Action

Contact your Fujitsu system engineer (SE) or the Fujitsu Support Desk.

[ERR.] LLIO 2453 llio_transfer Unrecognized option(*input*).

Meaning

The specified option is unknown.

Input: Unknown option

Action

Check the specified option.

[ERR.] LLIO 2454 llio_transfer Missing path operand.

Meaning

The path to a file is not specified.

Action

Specify the path to the file.

[ERR.] LLIO 2455 llio_transfer Missing option format.

Meaning

An option is specified in an incorrect way.

Action

Check the specified option.

[ERR.] LLIO 2456 llio_transfer System error(*detail*) detail code(*code*) \n.

Meaning

A system error occurred during processing with the llio_transfer command.

detail: Detailed maintenance information
code : Detailed error code for maintenance

Action

Contact a Fujitsu systems engineer (SE) or Fujitsu SupportDesk.

[WARN] LLIO 6450 llio_transfer Permission denied(*path*).

Meaning

The file privileges specified in an argument of the llio_transfer command are incorrect.

Path: File path with no access rights

Action

Confirm access privileges to the file specified in the argument of the llio_transfer command.

[WARN] LLIO 6451 llio_transfer File does not exist(*path*).

Meaning

The file specified in an argument of the llio_transfer command does not exist.

Path: Path to the non-existent file

Action

Confirm that an existing file is specified in the argument of the llio_transfer command.

[WARN] LLIO 6452 llio_transfer File was already cached(*path*).

Meaning

The file specified in an argument of the llio_transfer command was cached by an application in the job.

path: Path to a file cached by an application in the job

Action

Do not open the file in the job before executing the llio_transfer command because the cache data may be created when the file is opened. Do not use the following commands, as they may also open files.

- lfs setstripe
- lfs getstripe

[WARN] LLIO 6453 llio_transfer File is not supported(*path*) detail (*reason*).

Meaning

The common file distribution destinations do not include the file specified in an argument of the llio_transfer command.

path: File path not supported by the command
reason: Reason for determining unsupported

"file is not a global file": The file is not in second-layer storage.
"not regular file": The file type is not regular file.
"file size is zero": The file size is not more than 1 byte.

Action

Check the contents of reason and specify an appropriate file.

[WARN] LLIO 6454 llio_transfer File has already been distributed(*path*).

Meaning

The file specified in an argument of the llio_transfer command is an already distributed common file.

path: File path of an already distributed common file

Action

No action is necessary.

[WARN] LLIO 6455 llio_transfer There is no file to be deleted(*path*).

Meaning

The file specified in an argument of the llio_transfer command is not a common file to be deleted.

path: File path without a common file to be deleted

Action

No action is necessary.

[WARN] LLIO 6456 llio_transfer Pathname is not suitable(*path*).

Meaning

The file path name specified in the argument of the llio_transfer command is not valid.

path: Bad path name

Action

Make sure that the directory in the argument path name points to the correct directory.

Alternatively, reduce the use of symbolic links in path names.

[WARN] LLIO 6457 llio_transfer Filename is too long(*path*).

Meaning

The path name specified in the argument of the llio_transfer command is too long.

path: Too long path name

Action

Shorten the argument path name.

[WARN] LLIO 6458 llio_transfer Old file is being deleted(*path*).

Meaning

The file specified in an argument of the llio_transfer command was distributed as a common file and currently is being deleted.

path: File path of a common file being deleted

Action

Make sure that the file has already been closed. If it has not been closed yet, close it and then try the command again.
If async-close is on and the file was closed and this message is output, wait a few seconds and then try the command again.

B.2.4 showsiostats Command

[ERR.] LLIO 2510 showsiostats Invalid option specified(option=*Opt*).

Meaning

The specified option is unknown.

Action

Check the specified option.

[ERR.] LLIO 2511 showsiostats Required option not specified(option=*Opt*).

Meaning

A required option is not specified.

Action

Check the specified option.

[ERR.] LLIO 2512 showsiostats Directory not exist(path=*path*).

Meaning

The directory path specified in the -d option does not exist.

Action

Check the specified directory path.

[ERR.] LLIO 2513 showsiostats --from option format error(from=*from*).

Meaning

The format specified in the --from option is incorrect.

Action

Check the format specified in the --from option.

Appendix C Statistical Information Output Items

C.1 Output Items of LLIO Performance Information

The LLIO performance information is statistics for job input/output to first-layer storage (LLIO). There are two types of files to which LLIO performance information is output:

- The file that is output as a result of running the job (With the --llio perf option of the pjsub command when the job is submitted)

  The LLIO performance information is output as follows:

```
I/O     2ndLayerCache   NodeTotal                      Count          Amount(Byte)      Time(us)
                Write(Cached I/O)       Sum            <Value>        <Value>           <Value>
                                        [1,4Ki)        <Value>        <Value>           <Value>
                                        [4Ki,1Mi)      <Value>        <Value>           <Value>
                                        [1Mi,4Mi)      <Value>        <Value>           <Value>
                                        [4Mi+          <Value>        <Value>           <Value>
                Read(Cached I/O)        Sum            <Value>        <Value>           <Value>
                                        [1,4Ki)        <Value>        <Value>           <Value>
                                        ...
I/O     2ndLayerCache   ComputeNode<->CommBuf          Count          Amount(Byte)      Time(us)
                Write(Cached I/O)       Sum             -              -                <Value>
...
                        (*) Up to here, summary information of the job is output.
SIO Information         (*) From here, information for each storage I/O node.
Node ID : <NodeID>
I/O     2ndLayerCache   NodeTotal                      Count          Amount(Byte)      Time(us)
                Write(Cached I/O)       Sum            <Value>        <Value>           <Value>
                                        [1,4Ki)        <Value>        <Value>           <Value>
                                        [4Ki,1Mi)      <Value>        <Value>           <Value>
                                        [1Mi,4Mi)      <Value>        <Value>           <Value>
                                        [4Mi+          <Value>        <Value>           <Value>
                Read(Cached I/O)        Sum            <Value>        <Value>           <Value>
                                        [1,4Ki)        <Value>        <Value>           <Value>
                                        ...
I/O     2ndLayerCache   ComputeNode<->CommBuf          Count          Amount(Byte)      Time(us)
                Write(Cached I/O)       Sum             -              -                <Value>
...
Node ID : <NodeID>
I/O     2ndLayerCache   NodeTotal                      Count          Amount(Byte)      Time(us)
                Write(Cached I/O)       Sum            <Value>        <Value>           <Value>
                ...
```

  See "Table C.1 Output Items of LLIO Performance Information" for the meaning of each line.
  The output values <Value> are:

  - `Count` : Total number of times
  - `Amount(Byte)` : Total data amount (bytes)
  - `Time(us)` : Total processing time (microseconds)

- The log file of the Job Operation Software /var/opt/FJSVtcs/shared_disk/pjm/jsti/llioinfo

  The LLIO performance information is output in CSV format as follows:

```
JLLIO,<Job ID>,<Step number>,<Bulk number>,<#1>,<#2>,<#3>,<#4>,...
SLLIO,<Node ID>,<#1>,<#2>,<#3>,<#4>,...
SLLIO,<Node ID>,<#1>,<#2>,<#3>,<#4>,...
...
```

The line starting with "JLLIO" is the LLIO performance information for the job. The lines starting with "SLLIO" are the LLIO performance information for each storage I/O node that processed the job I/O. The "JLLIO" and "SLLIIO" lines are output sequentially for one job. The value *<#n>* corresponds to what is shown in "Table C.1 Output Items of LLIO Performance Information."

Table C.1 Output Items of LLIO Performance Information

`I/O 2ndLayerCache NodeTotal` I/O information (summary) to cache area of second-layer storage		Output Values <#n> in the llioinfo file ("-" items are not output)		
		Count	Amount	Time
`Write(Cached I/O)` Write without specifying O_DIRECT	`Sum` : Total	-	-	-
	`[1,4Ki)` : 1Byte to (4Ki-1)Bytes	<#1>	<#2>	<#3>
	`[4Ki,1Mi)` : 4KiBytes to (1Mi-1)Bytes	<#4>	<#5>	<#6>
	`[1Mi,4Mi)` : 1MiBytes to (4Mi-1)Bytes	<#7>	<#8>	<#9>
	`[4Mi+` : 4MiBytes or more	<#10>	<#11>	<#12>
`Read(Cached I/O)` Read without specifying O_DIRECT	`Sum` : Total	-	-	-
	`[1,4Ki)` : 1Byte to (4Ki-1)Bytes	<#13>	<#14>	<#15>
	`[4Ki,1Mi)` : 4KiBytes to (1Mi-1)Bytes	<#16>	<#17>	<#18>
	`[1Mi,4Mi)` : 1MiBytes to (4Mi-1)Bytes	<#19>	<#20>	<#21>
	`[4Mi+` : 4MiBytes or more	<#22>	<#23>	<#24>
`Write(Direct I/O)` Write with O_DIRECT specified	`Sum` : Total	-	-	-
	`[1,4Ki)` : 1Byte to (4Ki-1)Bytes	<#25>	<#26>	<#27>
	`[4Ki,1Mi)` : 4KiBytes to (1Mi-1)Bytes	<#28>	<#29>	<#30>
	`[1Mi,4Mi)` : 1MiBytes to (4Mi-1)Bytes	<#31>	<#32>	<#33>
	`[4Mi+` : 4MiBytes or more	<#34>	<#35>	<#36>
`Read(Direct I/O)` Read with O_DIRECT specified	`Sum` : Total	-	-	-
	`[1,4Ki)` : 1Byte to (4Ki-1)Bytes	<#37>	<#38>	<#39>
	`[4Ki,1Mi)` : 4KiBytes to (1Mi-1)Bytes	<#40>	<#41>	<#42>
	`[1Mi,4Mi)` : 1MiBytes to (4Mi-1)Bytes	<#43>	<#44>	<#45>
	`[4Mi+` : 4MiBytes or more	<#46>	<#47>	<#48>

`I/O 2ndLayerCache ComputeNode<->CommBuf` I/O information to cache area of second-layer storage (I/O information between compute node and communication buffer)		Output Values <#n> in the llioinfo file ("-" items are not output)		
		Count	Amount	Time
`Write(Cached I/O)` Write without specifying O_DIRECT	`Sum` : Total	-	-	-
	`[1,4Ki)` : 1Byte to (4Ki-1)Bytes	-	-	<#49>
	`[4Ki,1Mi)` : 4KiBytes to (1Mi-1)Bytes	-	-	<#50>
	`[1Mi,4Mi)` : 1MiBytes to (4Mi-1)Bytes	-	-	<#51>
	`[4Mi+` : 4MiBytes or more	-	-	<#52>
`Read(Cached I/O)` Read without specifying O_DIRECT	`Sum` : Total	-	-	-
	`[1,4Ki)` : 1Byte to (4Ki-1)Bytes	-	-	<#53>
	`[4Ki,1Mi)` : 4KiBytes to (1Mi-1)Bytes	-	-	<#54>
	`[1Mi,4Mi)` : 1MiBytes to (4Mi-1)Bytes	-	-	<#55>

I/O 2ndLayerCache ComputeNode<->CommBuf I/O information to cache area of second-layer storage (I/O information between compute node and communication buffer)		Output Values <*#n*> in the llioinfo file ("-" items are not output)		
		Count	Amount	Time
	[4Mi+ : 4MiBytes or more	-	-	*<#56>*
Write(Direct I/O) Write with O_DIRECT specified	Sum : Total	-	-	-
	[1,4Ki) : 1Byte to (4Ki-1)Bytes	-	-	*<#57>*
	[4Ki,1Mi) : 4KiBytes to (1Mi-1)Bytes	-	-	*<#58>*
	[1Mi,4Mi) : 1MiBytes to (4Mi-1)Bytes	-	-	*<#59>*
	[4Mi+ : 4MiBytes or more	-	-	*<#60>*
Read(Direct I/O) Read with O_DIRECT specified	Sum : Total	-	-	-
	[1,4Ki) : 1Byte to (4Ki-1)Bytes	-	-	*<#61>*
	[4Ki,1Mi) : 4KiBytes to (1Mi-1)Bytes	-	-	*<#62>*
	[1Mi,4Mi) : 1MiBytes to (4Mi-1)Bytes	-	-	*<#63>*
	[4Mi+ : 4MiBytes or more	-	-	*<#64>*

I/O 2ndLayerCache CommBuf<->1st&2ndStorageDev I/O information to cache area of second-layer storage (I/O information transferred between communication buffer and storage of either layer)		Output Values <*#n*> in the llioinfo file ("-" items are not output)		
		Count	Amount	Time
Write(Cached I/O) Write without specifying O_DIRECT	Sum : Total	-	-	-
	[1,4Ki) : 1Byte to (4Ki-1)Bytes	-	-	*<#65>*
	[4Ki,1Mi) : 4KiBytes to (1Mi-1)Bytes	-	-	*<#66>*
	[1Mi,4Mi) : 1MiBytes to (4Mi-1)Bytes	-	-	*<#67>*
	[4Mi+ : 4MiBytes or more	-	-	*<#68>*
Read(Cached I/O) Read without specifying O_DIRECT	Sum : Total	-	-	-
	[1,4Ki) : 1Byte to (4Ki-1)Bytes	-	-	*<#69>*
	[4Ki,1Mi) : 4KiBytes to (1Mi-1)Bytes	-	-	*<#70>*
	[1Mi,4Mi) : 1MiBytes to (4Mi-1)Bytes	-	-	*<#71>*
	[4Mi+ : 4MiBytes or more	-	-	*<#72>*
Write(Direct I/O) Write with O_DIRECT specified	Sum : Total	-	-	-
	[1,4Ki) : 1Byte to (4Ki-1)Bytes	-	-	*<#73>*
	[4Ki,1Mi) : 4KiBytes to (1Mi-1)Bytes	-	-	*<#74>*
	[1Mi,4Mi) : 1MiBytes to (4Mi-1)Bytes	-	-	*<#75>*
	[4Mi+ : 4MiBytes or more	-	-	*<#76>*
Read(Direct I/O) Read with O_DIRECT specified	Sum : Total	-	-	-
	[1,4Ki) : 1Byte to (4Ki-1)Bytes	-	-	*<#77>*
	[4Ki,1Mi) : 4KiBytes to (1Mi-1)Bytes	-	-	*<#78>*
	[1Mi,4Mi) : 1MiBytes to (4Mi-1)Bytes	-	-	*<#79>*
	[4Mi+ : 4MiBytes or more	-	-	*<#80>*

`I/O 2ndLayerCache 1stStorageDev->CommBuf(Async)` I/O information to cache area of second-layer storage (I/O information asynchronously transferred from first-layer storage to communication buffer)	Output Values <*#n*> in the llioinfo file ("-" items are not output)		
	Count	Amount	Time
`Sum` : Total	-	-	-
`[1,TS)` : 1Byte to (TS-1)Bytes (TS=Maximum transfer length)	*<#81>*	*<#82>*	*<#83>*
`[TS]` : Maximum transfer length	*<#84>*	*<#85>*	*<#86>*

`I/O 2ndLayerCache CommBuf->2ndStorageDev(Async)` I/O information to cache area of second-layer storage (I/O information asynchronously transferred from communication buffer to second-layer storage)	Output Values <*#n*> in the llioinfo file ("-" items are not output)		
	Count	Amount	Time
`Sum` : Total	-	-	-
`[1,TS)` : 1Byte to (TS-1)Bytes (TS=Maximum transfer length)	*<#87>*	*<#88>*	*<#89>*
`[TS]` : Maximum transfer length	*<#90>*	*<#91>*	*<#92>*

`I/O SharedTmp NodeTotal` I/O information (summary) in shared temporary area		Output Values <*#n*> in the llioinfo file ("-" items are not output)		
		Count	Amount	Time
`Write(Cached I/O)` Write without specifying O_DIRECT	`Sum` : Total	-	-	-
	`[1,4Ki)` : 1Byte to (4Ki-1)Bytes	*<#93>*	*<#94>*	*<#95>*
	`[4Ki,1Mi)` : 4KiBytes to (1Mi-1)Bytes	*<#96>*	*<#97>*	*<#98>*
	`[1Mi,4Mi)` : 1MiBytes to (4Mi-1)Bytes	*<#99>*	*<#100>*	*<#101>*
	`[4Mi+` : 4MiBytes or more	*<#102>*	*<#103>*	*<#104>*
`Read(Cached I/O)` Read without specifying O_DIRECT	`Sum` : Total	-	-	-
	`[1,4Ki)` : 1Byte to (4Ki-1)Bytes	*<#105>*	*<#106>*	*<#107>*
	`[4Ki,1Mi)` : 4KiBytes to (1Mi-1)Bytes	*<#108>*	*<#109>*	*<#110>*
	`[1Mi,4Mi)` : 1MiBytes to (4Mi-1)Bytes	*<#111>*	*<#112>*	*<#113>*
	`[4Mi+` : 4MiBytes or more	*<#114>*	*<#115>*	*<#116>*
`Write(Direct I/O)` Write with O_DIRECT specified	`Sum` : Total	-	-	-
	`[1,4Ki)` : 1Byte to (4Ki-1)Bytes	*<#117>*	*<#118>*	*<#119>*
	`[4Ki,1Mi)` : 4KiBytes to (1Mi-1)Bytes	*<#120>*	*<#121>*	*<#122>*
	`[1Mi,4Mi)` : 1MiBytes to (4Mi-1)Bytes	*<#123>*	*<#124>*	*<#125>*
	`[4Mi+` : 4MiBytes or more	*<#126>*	*<#127>*	*<#128>*
`Read(Direct I/O)` Read with O_DIRECT specified	`Sum` : Total	-	-	-
	`[1,4Ki)` : 1Byte to (4Ki-1)Bytes	*<#129>*	*<#130>*	*<#131>*
	`[4Ki,1Mi)` : 4KiBytes to (1Mi-1)Bytes	*<#132>*	*<#133>*	*<#134>*
	`[1Mi,4Mi)` : 1MiBytes to (4Mi-1)Bytes	*<#135>*	*<#136>*	*<#137>*
	`[4Mi+` : 4MiBytes or more	*<#138>*	*<#139>*	*<#140>*

I/O SharedTmp ComputeNode<->CommBuf I/O information in shared temporary area (I/O information between compute node and communication buffer)		Output Values <#*n*> in the llioinfo file ("-" items are not output)		
		Count	Amount	Time
`Write(Cached I/O)` Write without specifying O_DIRECT	`Sum` : Total	-	-	-
	`[1,4Ki)` : 1Byte to (4Ki-1)Bytes	-	-	*<#141>*
	`[4Ki,1Mi)` : 4KiBytes to (1Mi-1)Bytes	-	-	*<#142>*
	`[1Mi,4Mi)` : 1MiBytes to (4Mi-1)Bytes	-	-	*<#143>*
	`[4Mi+` : 4MiBytes or more	-	-	*<#144>*
`Read(Cached I/O)` Read without specifying O_DIRECT	`Sum` : Total	-	-	-
	`[1,4Ki)` : 1Byte to (4Ki-1)Bytes	-	-	*<#145>*
	`[4Ki,1Mi)` : 4KiBytes to (1Mi-1)Bytes	-	-	*<#146>*
	`[1Mi,4Mi)` : 1MiBytes to (4Mi-1)Bytes	-	-	*<#147>*
	`[4Mi+` : 4MiBytes or more	-	-	*<#148>*
`Write(Direct I/O)` Write with O_DIRECT specified	`Sum` : Total	-	-	-
	`[1,4Ki)` : 1Byte to (4Ki-1)Bytes	-	-	*<#149>*
	`[4Ki,1Mi)` : 4KiBytes to (1Mi-1)Bytes	-	-	*<#150>*
	`[1Mi,4Mi)` : 1MiBytes to (4Mi-1)Bytes	-	-	*<#151>*
	`[4Mi+` : 4MiBytes or more	-	-	*<#152>*
`Read(Direct I/O)` Read with O_DIRECT specified	`Sum` : Total	-	-	-
	`[1,4Ki)` : 1Byte to (4Ki-1)Bytes	-	-	*<#153>*
	`[4Ki,1Mi)` : 4KiBytes to (1Mi-1)Bytes	-	-	*<#154>*
	`[1Mi,4Mi)` : 1MiBytes to (4Mi-1)Bytes	-	-	*<#155>*
	`[4Mi+` : 4MiBytes or more	-	-	*<#156>*

I/O SharedTmp CommBuf<->1stStorageDev I/O information in shared temporary area (I/O information transferred between communication buffer and first-layer storage)		Output Values <#*n*> in the llioinfo file ("-" items are not output)		
		Count	Amount	Time
`Write(Cached I/O)` Write without specifying O_DIRECT	`Sum` : Total	-	-	-
	`[1,4Ki)` : 1Byte to (4Ki-1)Bytes	-	-	*<#157>*
	`[4Ki,1Mi)` : 4KiBytes to (1Mi-1)Bytes	-	-	*<#158>*
	`[1Mi,4Mi)` : 1MiBytes to (4Mi-1)Bytes	-	-	*<#159>*
	`[4Mi+` : 4MiBytes or more	-	-	*<#160>*
`Read(Cached I/O)` Read without specifying O_DIRECT	`Sum` : Total	-	-	-
	`[1,4Ki)` : 1Byte to (4Ki-1)Bytes	-	-	*<#161>*
	`[4Ki,1Mi)` : 4KiBytes to (1Mi-1)Bytes	-	-	*<#162>*
	`[1Mi,4Mi)` : 1MiBytes to (4Mi-1)Bytes	-	-	*<#163>*
	`[4Mi+` : 4MiBytes or more	-	-	*<#164>*
`Write(Direct I/O)` Write with O_DIRECT specified	`Sum` : Total	-	-	-
	`[1,4Ki)` : 1Byte to (4Ki-1)Bytes	-	-	*<#165>*

`I/O SharedTmp CommBuf<->1stStorageDev` I/O information in shared temporary area (I/O information transferred between communication buffer and first-layer storage)		Output Values <*#n*> in the llioinfo file ("-" items are not output)		
		Count	Amount	Time
	`[4Ki,1Mi)` : 4KiBytes to (1Mi-1)Bytes	-	-	*<#166>*
	`[1Mi,4Mi)` : 1MiBytes to (4Mi-1)Bytes	-	-	*<#167>*
	`[4Mi+` : 4MiBytes or more	-	-	*<#168>*
`Read(Direct I/O)` Read with O_DIRECT specified	`Sum` : Total	-	-	-
	`[1,4Ki)` : 1Byte to (4Ki-1)Bytes	-	-	*<#169>*
	`[4Ki,1Mi)` : 4KiBytes to (1Mi-1)Bytes	-	-	*<#170>*
	`[1Mi,4Mi)` : 1MiBytes to (4Mi-1)Bytes	-	-	*<#171>*
	`[4Mi+` : 4MiBytes or more	-	-	*<#172>*

`I/O LocalTmp NodeTotal` I/O information (summary) in node temporary area		Output Values <*#n*> in the llioinfo file ("-" items are not output)		
		Count	Amount	Time
`Write(Cached I/O)` Write without specifying O_DIRECT	`Sum` : Total	-	-	-
	`[1,4Ki)` : 1Byte to (4Ki-1)Bytes	*<#173>*	*<#174>*	*<#175>*
	`[4Ki,1Mi)` : 4KiBytes to (1Mi-1)Bytes	*<#176>*	*<#177>*	*<#178>*
	`[1Mi,4Mi)` : 1MiBytes to (4Mi-1)Bytes	*<#179>*	*<#180>*	*<#181>*
	`[4Mi+` : 4MiBytes or more	*<#182>*	*<#183>*	*<#184>*
`Read(Cached I/O)` Read without specifying O_DIRECT	`Sum` : Total	-	-	-
	`[1,4Ki)` : 1Byte to (4Ki-1)Bytes	*<#185>*	*<#186>*	*<#187>*
	`[4Ki,1Mi)` : 4KiBytes to (1Mi-1)Bytes	*<#188>*	*<#189>*	*<#190>*
	`[1Mi,4Mi)` : 1MiBytes to (4Mi-1)Bytes	*<#191>*	*<#192>*	*<#193>*
	`[4Mi+` : 4MiBytes or more	*<#194>*	*<#195>*	*<#196>*
`Write(Direct I/O)` Write with O_DIRECT specified	`Sum` : Total	-	-	-
	`[1,4Ki)` : 1Byte to (4Ki-1)Bytes	*<#197>*	*<#198>*	*<#199>*
	`[4Ki,1Mi)` : 4KiBytes to (1Mi-1)Bytes	*<#200>*	*<#201>*	*<#202>*
	`[1Mi,4Mi)` : 1MiBytes to (4Mi-1)Bytes	*<#203>*	*<#204>*	*<#205>*
	`[4Mi+` : 4MiBytes or more	*<#206>*	*<#207>*	*<#208>*
`Read(Direct I/O)` Read with O_DIRECT specified	`Sum` : Total	-	-	-
	`[1,4Ki)` : 1Byte to (4Ki-1)Bytes	*<#209>*	*<#210>*	*<#211>*
	`[4Ki,1Mi)` : 4KiBytes to (1Mi-1)Bytes	*<#212>*	*<#213>*	*<#214>*
	`[1Mi,4Mi)` : 1MiBytes to (4Mi-1)Bytes	*<#215>*	*<#216>*	*<#217>*
	`[4Mi+` : 4MiBytes or more	*<#218>*	*<#219>*	*<#220>*

I/O LocalTmp ComputeNode<->CommBuf I/O information in node temporary area (I/O information between compute node and communication buffer)		Output Values <#*n*> in the llioinfo file ("-" items are not output)		
		Count	Amount	Time
`Write(Cached I/O)` Write without specifying O_DIRECT	`Sum` : Total	-	-	-
	`[1,4Ki)` : 1Byte to (4Ki-1)Bytes	-	-	*<#221>*
	`[4Ki,1Mi)` : 4KiBytes to (1Mi-1)Bytes	-	-	*<#222>*
	`[1Mi,4Mi)` : 1MiBytes to (4Mi-1)Bytes	-	-	*<#223>*
	`[4Mi+` : 4MiBytes or more	-	-	*<#224>*
`Read(Cached I/O)` Read without specifying O_DIRECT	`Sum` : Total	-	-	-
	`[1,4Ki)` : 1Byte to (4Ki-1)Bytes	-	-	*<#225>*
	`[4Ki,1Mi)` : 4KiBytes to (1Mi-1)Bytes	-	-	*<#226>*
	`[1Mi,4Mi)` : 1MiBytes to (4Mi-1)Bytes	-	-	*<#227>*
	`[4Mi+` : 4MiBytes or more	-	-	*<#228>*
`Write(Direct I/O)` Write with O_DIRECT specified	`Sum` : Total	-	-	-
	`[1,4Ki)` : 1Byte to (4Ki-1)Bytes	-	-	*<#229>*
	`[4Ki,1Mi)` : 4KiBytes to (1Mi-1)Bytes	-	-	*<#230>*
	`[1Mi,4Mi)` : 1MiBytes to (4Mi-1)Bytes	-	-	*<#231>*
	`[4Mi+` : 4MiBytes or more	-	-	*<#232>*
`Read(Direct I/O)` Read with O_DIRECT specified	`Sum` : Total	-	-	-
	`[1,4Ki)` : 1Byte to (4Ki-1)Bytes	-	-	*<#233>*
	`[4Ki,1Mi)` : 4KiBytes to (1Mi-1)Bytes	-	-	*<#234>*
	`[1Mi,4Mi)` : 1MiBytes to (4Mi-1)Bytes	-	-	*<#235>*
	`[4Mi+` : 4MiBytes or more	-	-	*<#236>*

I/O LocalTmp CommBuf<->1stStorageDev I/O information in node temporary area (I/O information transferred between communication buffer and first-layer storage)		Output Values <#*n*> in the llioinfo file ("-" items are not output)		
		Count	Amount	Time
`Write(Cached I/O)` Write without specifying O_DIRECT	`Sum` : Total	-	-	-
	`[1,4Ki)` : 1Byte to (4Ki-1)Bytes	-	-	*<#237>*
	`[4Ki,1Mi)` : 4KiBytes to (1Mi-1)Bytes	-	-	*<#238>*
	`[1Mi,4Mi)` : 1MiBytes to (4Mi-1)Bytes	-	-	*<#239>*
	`[4Mi+` : 4MiBytes or more	-	-	*<#240>*
`Read(Cached I/O)` Read without specifying O_DIRECT	`Sum` : Total	-	-	-
	`[1,4Ki)` : 1Byte to (4Ki-1)Bytes	-	-	*<#241>*
	`[4Ki,1Mi)` : 4KiBytes to (1Mi-1)Bytes	-	-	*<#242>*
	`[1Mi,4Mi)` : 1MiBytes to (4Mi-1)Bytes	-	-	*<#243>*
	`[4Mi+` : 4MiBytes or more	-	-	*<#244>*
`Write(Direct I/O)` Write with O_DIRECT specified	`Sum` : Total	-	-	-
	`[1,4Ki)` : 1Byte to (4Ki-1)Bytes	-	-	*<#245>*

I/O LocalTmp CommBuf<->1stStorageDev I/O information in node temporary area (I/O information transferred between communication buffer and first-layer storage)		Output Values <#n> in the llioinfo file ("-" items are not output)		
		Count	Amount	Time
	[4Ki,1Mi) : 4KiBytes to (1Mi-1)Bytes	-	-	*<#246>*
	[1Mi,4Mi) : 1MiBytes to (4Mi-1)Bytes	-	-	*<#247>*
	[4Mi+ : 4MiBytes or more	-	-	*<#248>*
Read(Direct I/O) Read with O_DIRECT specified	Sum : Total	-	-	-
	[1,4Ki) : 1Byte to (4Ki-1)Bytes	-	-	*<#249>*
	[4Ki,1Mi) : 4KiBytes to (1Mi-1)Bytes	-	-	*<#250>*
	[1Mi,4Mi) : 1MiBytes to (4Mi-1)Bytes	-	-	*<#251>*
	[4Mi+ : 4MiBytes or more	-	-	*<#252>*

Meta 2ndLayerCache NodeTotal Meta-access information in cache area of second-layer storage	Output Values <#n> in the llioinfo file ("-" items are not output)	
	Count	Time
open	*<#253>*	*<#254>*
close	*<#255>*	*<#256>*
lookup	*<#257>*	*<#258>*
mknod	*<#259>*	*<#260>*
link	*<#261>*	*<#262>*
unlink	*<#263>*	*<#264>*
mkdir	*<#265>*	*<#266>*
rmdir	*<#267>*	*<#268>*
readdir	*<#269>*	*<#270>*
rename	*<#271>*	*<#272>*
getattr	*<#273>*	*<#274>*
setattr	*<#275>*	*<#276>*
getxattr	*<#277>*	*<#278>*
setxattr	*<#279>*	*<#280>*
listxattr	*<#281>*	*<#282>*
removexattr	*<#283>*	*<#284>*
statfs	*<#285>*	*<#286>*
sync	*<#287>*	*<#288>*
lock	*<#289>*	*<#290>*
getstripe	*<#291>*	*<#292>*
setstripe	*<#293>*	*<#294>*
transfer	*<#295>*	*<#296>*

`Meta SharedTmp NodeTotal` Meta-access information in shared temporary area	Output Values <*#n*> in the llioinfo file ("-" items are not output)	
	Count	Time
`open`	*<#297>*	*<#298>*
`close`	*<#299>*	*<#300>*
`lookup`	*<#301>*	*<#302>*
`mknod`	*<#303>*	*<#304>*
`link`	*<#305>*	*<#306>*
`unlink`	*<#307>*	*<#308>*
`mkdir`	*<#309>*	*<#310>*
`rmdir`	*<#311>*	*<#312>*
`readdir`	*<#313>*	*<#314>*
`rename`	*<#315>*	*<#316>*
`getattr`	*<#317>*	*<#318>*
`setattr`	*<#319>*	*<#320>*
`getxattr`	*<#321>*	*<#322>*
`setxattr`	*<#323>*	*<#324>*
`listxattr`	*<#325>*	*<#326>*
`removexattr`	*<#327>*	*<#328>*
`statfs`	*<#329>*	*<#330>*
`sync`	*<#331>*	*<#332>*
`lock`	*<#333>*	*<#334>*

`Meta LocalTmp NodeTotal` Meta-access information in node temporary area	Output Values <*#n*> in the llioinfo file ("-" items are not output)	
	Count	Time
`open`	*<#335>*	*<#336>*
`close`	*<#337>*	*<#338>*
`lookup`	*<#339>*	*<#340>*
`mknod`	*<#341>*	*<#342>*
`link`	*<#343>*	*<#344>*
`unlink`	*<#345>*	*<#346>*
`mkdir`	*<#347>*	*<#348>*
`rmdir`	*<#349>*	*<#350>*
`readdir`	*<#351>*	*<#352>*
`rename`	*<#353>*	*<#354>*
`getattr`	*<#355>*	*<#356>*
`setattr`	*<#357>*	*<#358>*
`getxattr`	*<#359>*	*<#360>*

Meta LocalTmp NodeTotal Meta-access information in node temporary area	Output Values <#n> in the llioinfo file ("-" items are not output)	
	Count	Time
setxattr	*<#361>*	*<#362>*
listxattr	*<#363>*	*<#364>*
removexattr	*<#365>*	*<#366>*
statfs	*<#367>*	*<#368>*
sync	*<#369>*	*<#370>*
lock	*<#371>*	*<#372>*

Resource 2ndLayerCache CacheOperation Resource information in cache area of second-layer storage	Output Values <#n> in the llioinfo file ("-" items are not output)		
	Count	Amount	Time
Hit : Cache hit	*<#373>*	-	-
Miss : Cache miss	*<#374>*	*<#375>*	*<#376>*
Shortage_wait : Cache write wait (cache I/O when cache area is insufficient)	*<#377>*	*<#378>*	*<#379>*
Nonssd_io_wait : Cache invalidation (read or direct I/O when sio-read-cache=off)	*<#380>*	*<#381>*	*<#382>*
Ssd_io_wait : Cache write wait (read or cache I/O when sio-read-cache=on)	*<#383>*	*<#384>*	*<#385>*
Invalidate : Cache invalidation (cache I/O when cache area is insufficient)	*<#386>*	*<#387>*	*<#388>*

Resource 2ndLayerCache CacheUsage Usage of cache area of second-layer storage	Output Values <#n> in the llioinfo file ("-" items are not output)
	Amount
MaximalUsedSpace : Maximum consumed capacity (*) Minimum free space is output for llioinfo file.	*<#389>*
Uncompleted-file : Amount of data not written	*<#390>*

[Note]
Amount of data not written output to the LLIO performance information are information of files that have been not written completely from a first-layer storage to second-layer storage.
On the other hand, unwritten file list described in "2.6 Unwritten File List Acquisition Function" outputs the following information.

- Files that have been not written completely from a compute node cache to first-layer storage
- Files that have been not written completely from a first-layer storage to second-layer storage

Therefore, total unwritten file sizes are as follows.

```
The value in unwritten file list >= Amount of data not written output to the LLIO performance
                                    information
```

Resource SharedTmp CacheUsage Usage of shared temporary area	Output Values <#n> in the llioinfo file ("-" items are not output)
	Amount
MaximalUsedSpace : Maximum consumed capacity	*<#391>*

`Resource LocalTmp CacheUsage` Usage of node temporary area	Output Values <*#n*> in the llioinfo file ("-" items are not output)
	Amount
`MaximalUsedSpace` : Maximum consumed capacity	*<#392>*

C.2 Collection Methods and Output Items of Compute Node Statistical Information

C.2.1 Collection Methods of Compute Node Statistical Information

Use the getlliostat library function to collect compute node statistical information from an application executed within a job.

See

For details on the getlliostat library function, see "A.3.1 getlliostat."

The format of the getlliostat library function is shown below.

```
#include <llio_stat.h>
int getlliostat(lliostat_t * lliostat):
```

The following example shows a program (sample.c).

```
#include <llio_stat.h>                                   <- Write process for including include files

int main(int argc, char *argv[]) {
    lliostat_t stat;
    <Omitted>
    memset(&stat, 0, sizeof(lliostat_t));
    int stat_status = getlliostat(&stat);
    if (stat_status == 0) {
        // Performance information output processing
        // Reference lliostat_t structure member to create output
    } else {
        // Error processing
    }
    <Omitted>
}
```

The following example shows how to compile sample.c.

```
# fcc -I/opt/FJSVllio/include -L/opt/FJSVllio/lib  -llliostat -o sample -D_GNU_SOURCE sample.c
      ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^  <- Add to fcc option
```

The following example shows a job script.

```
#!/bin/sh

export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/opt/FJSVllio/lib
/gfs/sample -f 4m -b 4m -w /gfs/aaa
```

After execution, the results are output to the normal output file of jobs.

The contents of the output are described at the performance information output processing part in the above example of a program.

C.2.2 Output Items of Compute Node Statistical Information

This section describes the structures and enumerated types used by the getlliostat library. They are defined in llio_stat.h.

```
enum LLIO_IOSIZERANGE {                                        // Enumerated types of I/O size ranges
collected by compute node
   LLIO_IOSIZERANGE_GE_1B,                                       // 1 B to 4 B-1
   LLIO_IOSIZERANGE_GE_4B,                                       // 4 B to 16 B-1
   LLIO_IOSIZERANGE_GE_16B,                                      // 16 B to 64 B-1
   LLIO_IOSIZERANGE_GE_64B,                                      // 64 B to 256 B-1
   LLIO_IOSIZERANGE_GE_256B,                                     // 256 B to 1 KiB-1
   LLIO_IOSIZERANGE_GE_1KIB,                                     // 1 KiB to 4 KiB-1
   LLIO_IOSIZERANGE_GE_4KIB,                                     // 4 KiB to 16 KiB-1
   LLIO_IOSIZERANGE_GE_16KIB,                                    // 16 KiB to 64 KiB-1
   LLIO_IOSIZERANGE_GE_64KIB,                                    // 64 KiB to 256 KiB-1
   LLIO_IOSIZERANGE_GE_256KIB,                                   // 256 KiB to 1 MiB-1
   LLIO_IOSIZERANGE_GE_1MIB,                                     // 1 MiB to 4 MiB-1
   LLIO_IOSIZERANGE_GE_4MIB,                                     // 4 MiB to 16 MiB-1
   LLIO_IOSIZERANGE_GE_16MIB,                                    // 16 MiB to 64 MiB-1
   LLIO_IOSIZERANGE_GE_64MIB,                                    // 64 MiB to 256 MiB-1
   LLIO_IOSIZERANGE_GE_256MIB,                                   // 256 MiB or larger
   LLIO_NUM_IOSIZERANGES                                       // Number of I/O size types collected
by compute node
};
```

```
enum LLIO_METAOPS{                                           // Enumerated types of meta-access types
for first-layer storage
   LLIO_METAOPS_OPEN,                                            // open
   LLIO_METAOPS_CLOSE,                                           // close
   LLIO_METAOPS_LOOKUP,                                          // lookup
   LLIO_METAOPS_MKNOD,                                           // mknod
   LLIO_METAOPS_LINK,                                            // link
   LLIO_METAOPS_UNLINK,                                          // unlink
   LLIO_METAOPS_MKDIR,                                           // mkdir
   LLIO_METAOPS_RMDIR,                                           // rmdir
   LLIO_METAOPS_READDIR,                                         // readdir
   LLIO_METAOPS_RENAME,                                          // rename
   LLIO_METAOPS_GETATTR,                                         // getattr
   LLIO_METAOPS_SETATTR,                                         // setattr
   LLIO_METAOPS_GETXATTR,                                        // getxattr
   LLIO_METAOPS_SETXATTR,                                        // setxattr
   LLIO_METAOPS_LISTXATTR,                                       // listxattr
   LLIO_METAOPS_REMOVEXATTR,                                     // removexattr
   LLIO_METAOPS_STATFS,                                          // statfs
   LLIO_METAOPS_SYNC,                                            // sync
   LLIO_METAOPS_LOCK,                                            // lock
   LLIO_METAOPS_GETSTRIPE,                                       // getstripe
   LLIO_METAOPS_SETSTRIPE,                                       // setstripe
   LLIO_METAOPS_TRANSFER,                                        // transfer
   LLIO_NUM_METAOPS                                          // Number of meta-access types for first-
layer storage
};
```

```
struct llio_cnstat_io {                                    // Structure of total I/O amount for first-
layer storage
   uint64_t write_count[LLIO_NUM_IOSIZERANGES];                 // Total number of writes
   uint64_t write_amount[LLIO_NUM_IOSIZERANGES];               // Total amount of write transfers (in
bytes)
   uint64_t write_time[LLIO_NUM_IOSIZERANGES];                  // Total write processing time (in
microseconds)
   uint64_t read_count[LLIO_NUM_IOSIZERANGES];                  // Total number of reads
   uint64_t read_amount[LLIO_NUM_IOSIZERANGES];                 // Total amount of read transfers (in
bytes)
   uint64_t read_time[LLIO_NUM_IOSIZERANGES];                   // Total read processing time (in
microseconds)
};
```

```
enum LLIO_IO_TYPE {                                       // Enumerated types of I/O types for first-
layer storage
   LLIO_IO_CACHED,                                            // Cache I/O
   LLIO_IO_DIRECT,                                            // Direct I/O
   LLIO_NUM_IO_TYPES                                          // Number of I/O types
};
```

```
struct llio_cnstat_meta {                                 // Structure of meta information for first-
layer storage
   uint64_t meta_count[LLIO_NUM_METAOPS];                     // Total number of meta-access times
   uint64_t meta_time[LLIO_NUM_METAOPS];                  // Total processing time (in microseconds)
};
```

```
struct llio_cnstat_cache {                                    // Structure of cache information for
first-layer storage
   uint64_t cache_hit_count;                            // Number of cache hits (cumulative total) *1
   uint64_t cache_miss_count;                         // Number of cache misses (cumulative total) *2
   uint64_t cache_wait_count;                                 // Number of cache clear wait times
(cumulative total)
   uint64_t cache_wait_amount;            // Amount of data evicted due to insufficient cache space
(cumulative total, in bytes)
   uint64_t cache_wait_time;                                 // Cumulative total period waiting for
cache clearing (in microseconds)
   uint64_t cache_free;                                    // Current space with no cache allocated
(in bytes)
   uint64_t cache_used;                                    // Current space used by cache (in bytes)
   uint64_t cache_dirty;                                  // Current space not yet written in cache
(in bytes)
};
```

Restriction: *1 and *2 above are not collected for files that have been memory mapped by mmap (2).

```
struct llio_cnstat_fs {                                      // Structure of all I/O information for
first-layer storage
   struct llio_cnstat_io io[LLIO_NUM_IO_TYPES];            // Structure of total I/O amount for first-
layer storage
   struct llio_cnstat_meta meta;                          // Structure of meta information for first-
layer storage
   struct llio_cnstat_cache cache;                       // Structure of cache information for first-
layer storage
};
```

```
enum LLIO_FS_TYPE {                                      // Enumerated types of forms of first-layer
storage usage
   LLIO_GFS_CACHE,                                           // Second-layer storage cache
   LLIO_SHARED_TEMP,                                         // Shared temporary area
   LLIO_LOCAL_TEMP,                                          // Node temporary area
   LLIO_NUM_FS_TYPES                                      // Number of types of forms of first-layer
storage usage
};
```

```
struct lliostat {                                            // lliostat structure
   struct llio_cnstat_fs fs[LLIO_NUM_FS_TYPES];             // Structure of all I/O information for
first-layer storage
};
#define lliostat_t struct lliostat
extern int getlliostat(lliostat_t *);
```

C.3 Output Items of System Statistical Information

C.3.1 Output Items of System Statistical Information for a Storage I/O Node

The following tables list the output items of system statistical information for a storage I/O node.

Table C.2 Types of System Statistical Information

Type	Description
j	I/O information per job
c	Information related to the number of connections on the communication layer
r	Request information per job

Table C.3 I/O Information Items

Item Name and Example	Description
<usage form>-<I/O location>-[Direct]<I/O type>(<size>)-<I/O item> [Example] Cache-CN-SIO-RD(4K)-Amount	<I/O item> per <size> of <I/O type> of Direct/Cached I/O at the <I/O location> intended for <usage form> * For Direct I/O, "Direct" is appended after <I/O location>.

Table C.4 Meta-Access Information Items

Item Name and Example	Description
<usage form>-<meta-access type>-<meta-access item> [Example] Share-close-Count	<meta-access item> per <meta-access type> intended for <usage form>

Table C.5 Resource Information Items

Item Name and Example	Description
<usage form>-<resource information item> [Example] Local-MinFreeSpace	<resource information item> intended for <usage form>

Table C.6 Usage Forms

Abbreviation	Description
Cache	Second-layer storage cache
Share	Shared temporary
Local	Node temporary

Table C.7 I/O Location Abbreviations and Meanings

Abbreviation	Meaning
Total	Total I/O
CN-CBCN	I/O information between a compute node and the communication buffer
CBCN-Dev	I/O information (for communication with a compute node) transferred between the communication buffer and a first-layer storage device
CBGFS-Dev	I/O information (for communication with second-layer storage) transferred between the communication buffer and a first-layer storage device
CBCN-GFS	I/O information (for Direct I/O) transferred between the communication buffer and second-layer storage
CBGFS-GFS	I/O information (for Cached I/O) transferred between the communication buffer and second-layer storage

Table C.8 I/O Abbreviations

Abbreviation	Description
RD	Read
WR	Write

Table C.9 Abbreviations for I/O Sizes

Abbreviation	Description
4K-	1 B to 4 KiB-1 B
1M-	4 KiB to 1 MiB-1 B
4M-	1 MiB to 4 MiB-1 B
4M+	4 MiB and up
TrSz-	1 B to <transfer size>-1 B
TrSz+	<transfer size> B

Table C.10 Abbreviations for I/O Items

Abbreviation	Description
Count	Total number of times
Amount	Total data amount
Time	Total processing time

Table C.11 Meta-Access Types

Abbreviation	Meta-Access
open	open
close	close
lookup	lookup
mknod	mknod
link	link
unlink	unlink
mkdir	mkdir
rmdir	rmdir
readdir	readdir
rename	rename
getattr	getattr
setattr	setattr
getxattr	getxattr
setxattr	setxattr
listxattr	listxattr
removexattr	removexattr
statfs	statfs
sync	sync
lock	lock
getstripe	getstripe

Abbreviation	Meta-Access
setstripe	setstripe
transfer	transfer
allocate	allocate
purge	purge

Table C.12 Meta-Access Items

Abbreviation	Meta-Access
Count	Total number of times
Time	Total processing time

Table C.13 Resource Information Items

Abbreviation	Description
CacheHitCount	Number of cache hits
CacheMissCount	Number of cache misses
CacheMissAmount	Amount of cache miss data
CacheMissTime	Cache miss time
CacheShortageWaitCount	Total number of cache evictions caused by insufficient cache capacity
CacheShortageWaitAmount	Total amount of data evicted from the cache because of insufficient cache capacity
CacheShortageWaitTime	Total time taken for cache evictions caused by insufficient cache capacity
CacheNonSSDIOWaitCount	Total number of cache evictions caused by I/O operations not using the SSD for a block that has old cache data
CacheNonSSDIOWaitAmount	Total amount of data evicted from the cache because of I/O operations not using the SSD for a block that has old cache data
CacheNonSSDIOWaitTime	Total time taken for cache evictions caused by I/O operations not using the SSD for a block that has old cache data
CacheSSDIOWaitCount	Total number of cache evictions caused by I/O operations using the SSD for a block that has old cache data
CacheSSDIOWaitAmount	Total amount of data evicted from the cache because of I/O operations using the SSD for a block that has old cache data
CacheSSDIOWaitTime	Total time taken for cache evictions caused by I/O operations using the SSD for a block that has old cache data
CacheInvalCount	Number of times that the cache was invalidated
CacheInvalAmount	Amount of invalidated cache data
CacheInvalTime	Cache invalidation time
MaximalUsage	Maximum consumed capacity
CacheUnflushedAmount	Amount of data not written

System statistical information type "c" indicates information related to the number of connections on the communication layer. "Table C.14 Abbreviations for Connection Count Items" lists the abbreviations for output items showing the number of connections or disconnections of an applicable node connection.

Table C.14 Abbreviations for Connection Count Items

Abbreviation	Description
Conn	Connection count of current established connections

Abbreviation	Description
InProConn	Connection count of connections in progress
InProDisconn	Connection count of disconnections in progress
MaxConn	Past maximum values for the total connection counts of established connections, connections in progress, and disconnections in progress
AccumConn	Cumulative total number of new connections in the past
AccumDisconn	Cumulative total number of disconnections in the past

Table C.15 Abbreviations for Request Items

Abbreviation	Description
<request type>-Wait-Num	Number of times that the wait time was recorded
<request type>-Wait-Min	Minimum wait time value
<request type>-Wait-Max	Maximum wait time value
<request type>-Wait-Sum	Total wait time value
<request type>-Qdepth-Num	Number of times that the queue length was recorded
<request type>-Qdepth-Min	Minimum queue length value
<request type>-Qdepth-Max	Maximum queue length value
<request type>-Qdepth-Sum	Total queue length value
<request type>-Active-Num	Number of times that simultaneous request processing was recorded
<request type>-Active-Min	Minimum value of simultaneous request processing
<request type>-Active-Max	Maximum value of simultaneous request processing
<request type>-Active-Sum	Total count of simultaneous request processing

IO or Meta is the entry in <request type>.

C.3.2 Output Items of System Statistical Information for a Global I/O Node

The following tables list the output items of system statistical information for a global I/O node.

Table C.16 Number of Data Transfers and Abbreviations for Data Amount Items

Abbreviation	Meaning
TransferCount	Number of transfers on LNET layer
TransferAmount	Amount of transfers on LNET layer (bytes)

Table C.17 Abbreviations for Transfer Buffer Items

Abbreviation	Meaning
MinFreeBufZero	Minimum number of free buffers for 0-byte transfers
MinBufSend	Minimum number of free buffers for SEND transfers
MinFreeBufRDMA	Minimum number of free buffers for RDMA transfers

Glossary

In addition to the terms below, see the *Job Operation Software Glossary* and glossary of *FEFS User's Guide* manual.

common file

Files that are read from all compute nodes at the start of the job, such as executable files and configuration files on the on second-layer storage. To avoid performance deterioration due to concentration of access, common files can be copied in advance to the cache area of second-layer storage (common file distribution function).

interconnect

An interconnect is a high-speed network that transfers the compute data and input/output data of programs between nodes. Tofu interconnect D and InfiniBand are interconnects.

layered storage

Layered storage is a configuration with two layers: first-layer storage and second-layer storage.

first-layer storage

First-layer storage is a high-speed temporary area for jobs managed by LLIO.

second-layer storage

Second-layer storage is a file system managed by FEFS.

rolling update

A rolling update performs partial maintenance on some compute nodes to apply a package without stopping the system or entire cluster while job operations continue within the cluster.

BoB (Bunch of Blades)

A BoB is a control unit for the FX server. The BoB consists of 16 nodes.

FEFS (Fujitsu Exabyte File System)

FEFS is a parallel distributed file system developed by Fujitsu.

LNET

LNET is a function for accessing a file system commonly using a number of different networks, such as Ethernet and InfiniBand.

LLIO (Lightweight Layered IO-Accelerator)

LLIO is a technology that makes high performance possible. LLIO establishes a storage layer using SSD between FEFS and compute nodes, and uses the layer as the FEFS cache area and a temporary area for jobs. LLIO is also an abbreviation for the file system implemented with this technology.

SIO group

An SIO group is a compute node group that uses the same storage I/O node within a compute cluster.

Tofu interconnect D

Tofu interconnect D is a high-speed network connecting nodes to one another in the FX server. With the Job Operation Software, a network that connects compute nodes to one another is called a compute network.