
NEC Network Queuing System V (NQS_V) User's Guide
[JobManipulator]

Proprietary Notice

The information disclosed in this document is the property of NEC Corporation (NEC) and/or its licensors. NEC and/or its licensors, as appropriate, reserve all patent, copyright and other proprietary rights to this document, including all design, manufacturing, reproduction, use and sales rights thereto, except to the extent said rights are expressly granted to others.

The information in this document is subject to change at any time, without notice.

Preface

The NEC Network Queuing System V (NQSV) User's Guide [JobManipulator] explains how to use NQSV/JobManipulator.

February	2018	1st edition
September	2022	14th edition
January	2023	15th edition
March	2023	16th edition
June	2023	17th edition
July	2024	18 th edition

Remarks

- (1) This manual conforms to Release 1.00 and subsequent releases of NEC Network Queuing System V(NQSV)/JobManipulator
- (2) All the functions described in this manual are program products.
The typical functions of them conform to the following product names and product series numbers:

Product Name	product series numbers
NEC Network Queuing System V (NQSV) /JobManipulator	UWAH00 UWHAH00 (Support Pack)

- (3) UNIX is a registered trademark of The Open Group.
- (4) Intel is a trademark of Intel Corporation in the U.S. and/or other countries.
- (5) OpenStack is a trademark of OpenStack Foundation in the U.S. and/or other countries.
- (6) Red Hat OpenStack Platform is a trademark of Red Hat, Inc. in the U.S. and/or other countries.
- (7) Linux is a trademark of Linus Torvalds in the U.S. and/or other countries.
- (8) Docker is a trademark of Docker, Inc. in the U.S. and/or other countries.
- (9) InfiniBand is a trademark or service mark of InfiniBand Trade Association.
- (10) Zabbix is a trademark of Zabbix LLC that is based in Republic of Latvia.
- (11) All other product, brand, or trade names used in this publication are the trademarks or registered trademarks of their respective trademark owners.

About This Manual

This manual consists of the following chapters:

Chapter	Title	Contents
1	Overview of JobManipulator	Overview
2	Environment Architecture	Setting of Install and Scheduling of JobManipulator
3	Operation Management	Basic Feature of Scheduling
4	Advanced Features	Advanced Feature of Scheduling
5	Functions for SX-Aurora TSUBASA	Functions for SX-Aurora TSUBASA
6	Command Reference	Command Reference

Related manuals that relate to this manual are as follows.

G2AD01E NQSV User's Guide [Introduction]

G2AD02E NQSV User's Guide [Management]

G2AD03E NQSV User's Guide [Operation]

G2AD04E NQSV User's Guide [Reference]

G2AD05E NQSV User's Guide [API]

G2AD07E NQSV User's Guide [Accounting & Budget Control]

Notation Conventions and Glossary

The following notation rules are used in this manual:

Omission Symbol	...	This symbol indicates that the item mentioned previously can be repeated. The user may input similar items in any desired number.
Vertical Bar		This symbol divides an option and mandatory selection item.
Brackets	{ }	A pair of brackets indicates a series of parameters or keywords from which one has to be selected.
Braces	[]	A pair of braces indicate a series of parameters or keywords that can be omitted.

Glossary

Term	Definition
Vector Engine (VE)	The NEC original PCIe card for vector processing based on SX architecture. It is connected to x86-64 machine. VE consists of more than one core and shared memory.
Vector Host (VH)	The x86-64 architecture machine that VE connected.
Vector Island (VI)	The general component unit of a single VH and one or more VEs connected to the VH.
Batch Server (BSV)	Resident system process running on a Batch server host to manage entire NQSV system.

Job Server (JSV)	Resident system process running on each execution host to manage the execution of jobs.
JobManipulator (JM)	JobManipulator is the scheduler function of NQSV. JM manages the computing resources and determines the execution time of jobs.
Accounting Server	Accounting server collects and manages account information and manages budgets.
Request	A unit of user jobs in the NQSV system. It consists of one or more jobs. Requests are managed by the Batch Server.
Job	A job is an execution unit of user job. It is managed by Job Server.
Logical Host	A logical host is a set of logical (virtually) divided resources of an execution host.
Queue	It is a mechanism that pools and manages requests submitted to BSV.
BMC	Board Management Controller for short. It performs server management based on the Intelligent Platform Management Interface (IPMI).
HCA	Host Channel Adapter for short. The PCIe card installed in VH to connect to the IB network.
IB	InfiniBand for short.
MPI	Abbreviation for Message Passing Interface. MPI is a standard for parallel computing between nodes.
NIC	Network Interface Card for short. The hardware to communicate with other node.

CONTENTS

Proprietary Notice	ii
Preface	iii
About This Manual	iv
CONTENTS.....	vi
Contents of Figures.....	x
Chapter 1. Overview of JobManipulator	1
1.1 Introduction	1
1.2 Features of JobManipulator.....	1
Chapter 2. Environment Architecture.....	2
2.1 Configuration of JobManipulator	2
2.2 Package Configuration	3
2.3 Basic Environment Architecture	3
2.3.1 Environment.....	3
2.3.2 Installation of Package	4
2.3.3 JobManipulator Start	4
2.3.4 Queue Setting.....	4
2.3.5 Setting of the Client Environment.....	5
2.3.6 JobManipulator Stop	5
2.4 Unit Management.....	6
2.5 Setting of JobManipulator Start.....	6
2.5.1 Configuration file	6
2.5.2 Starting of the multiple JobManipulator	7
2.5.3 Updating from R1.05 or earlier to R1.06 or later.....	10
2.5.4 Start Option of JobManipulator.....	13
2.5.5 Command environment file	14
2.6 Scheduler Log File Setting.....	14
2.7 Scheduling Parameter Setting.....	15
2.7.1 Run Limit	15
2.7.2 Assign Limit	21
2.7.3 Request Priority Order	25
2.7.4 Queue Type	26
2.7.5 Setting of Complex Queue Feature	27
2.7.6 Setting of Escalation Feature.....	32
2.7.7 No Overtaking Control at Pick-up	35
2.7.8 Setting of Assign Policy	37
2.7.9 Setting of Wait Time of Rescheduling.....	41
2.7.10 Set ON/OFF of Scheduling Feature	42
Chapter 3. Operation Management.....	43
3.1 Scheduling Basic Feature	43
3.1.1 Scheduler Map.....	43
3.1.2 Real Time Scheduling	50
3.1.3 Usage Data Collection and Adjustment.....	51
3.1.4 Scheduling Priority	54
3.1.5 Algorithm for Picking up Request.....	62
3.1.6 Algorithm for Starting Request.....	63
3.1.7 Elapse Margin	65
3.1.8 Assign Policy.....	68
3.1.9 Suspended Request	72
3.1.10 Job Condition.....	73
3.1.11 Exit Delay Scheduling	73
3.2 System Information Display	74

Chapter 4. Advanced Scheduling Features	75
4.1 Urgent Request/Special Request	75
4.1.1 Block of Assignment by Urgent Request	76
4.2 Interactive Request	77
4.3 Parametric Request	79
4.4 Workflow.....	79
4.5 Execution Time Reservation	80
4.5.1 Specify the Execution Start Time	80
4.5.2 Action for Failing in Time Specification	81
4.6 Advance Reservation (Resource Reservation Section)	81
4.6.1 Create the Reserved Section	81
4.6.2 Job Submission to Reserved Section.....	83
4.6.3 Deleting the Reserved Section	84
4.6.4 Job Assignment to the Resource Reservation Section	85
4.6.5 Display the Information of the Resource Reservation Section	85
4.6.6 Accounting for Resource Reservation Section Specifying Execution Queue ...	87
4.6.7 Set section for health-check and clean-up	87
4.6.8 Creation Function of the Resource Reservation Section Specifying Template	88
4.7 ShareDB Merge Feature	92
4.7.1 Overview of ShareDB Merge Feature.....	92
4.7.2 Set ShareDB Merge Feature	94
4.7.3 Display the Usage Data of ShareDB.....	96
4.7.4 ShareDB Merge Configuration File	97
4.8 Elapse Unlimited Feature.....	100
4.8.1 Set Elapse Unlimited Feature.....	100
4.8.2 Display the Setting of Elapse Unlimited	101
4.9 Scheduling with the change in the number of CPUs/GPUs.....	101
4.10 Support for Failover System	102
4.11 Scheduling in Problem on Node.....	102
4.11.1 Rescheduling at Node Problem	102
4.11.2 Forced Rerunning of Running Job	103
4.11.3 Waiting to Forced Rerunning on Connection with BSV	103
4.11.4 Keep Forward Schedule.....	104
4.11.5 Top Priority Execution of the Failure Encounter Request	105
4.12 Deadline Scheduling.....	106
4.12.1 Overview of Deadline Scheduling	106
4.12.2 Setting of Deadline Scheduling	106
4.12.3 Submission of Deadline Request.....	107
4.12.4 Scheduling of Deadline Request.....	107
4.12.5 Usage Data of Deadline Request.....	110
4.13 Incorporating External Policy.....	113
4.13.1 Overview of Incorporating External Policy	113
4.13.2 Setting of Incorporating External Policy feature	114
4.13.3 Connection to External Policy Daemon	115
4.13.4 External Policy on Submitting.....	115
4.13.5 External Policy on Request Priority	116
4.13.6 External Policy on Assignment	117
4.13.7 API Functions.....	119
4.14 Power-saving Function.....	123
4.14.1 Overview of Power-saving Function	123
4.14.2 Dynamic Power-saving Function	124
4.14.3 Scheduled Power- saving Function	131
4.15 Custom Resource Function	134
4.15.1 Overview of Custom Resource Function.....	134

4.15.2	Scheduling using Custom Resource Information	134
4.15.3	Examples of Using Custom Resource Function	135
4.16	Provisioning with OpenStack	136
4.16.1	Overview of Provisioning with OpenStack	136
4.16.2	Setting Re-scheduling Waiting Time at Failure of Start of Execution Host ..	137
4.16.3	Scheduling of the Execution Hosts at Provisioning	138
4.16.4	The Waiting time of Stage-out of the Request on Baremetal Server.....	140
4.17	Provisioning with Docker	140
4.17.1	Overview of Provisioning with Docker	140
4.17.2	Setting Re-scheduling Waiting Time at Failure of Start of Execution Host ..	140
4.17.3	Scheduling of the Execution Hosts at Provisioning	141
4.18	Setting Function of the First Stage-in Time	141
4.19	Pre-Staging Function	142
4.19.1	Overview of Pre-Staging Function.....	142
4.19.2	Setting of Stage-in Starting Time Threshold	143
4.20	Display the Detail of the Execution Host Information.....	143
4.21	Node group selection function for minimum network topology	146
4.21.1	Overview	146
4.21.2	Setting.....	148
4.22	FIFO Scheduling.....	150
4.23	Cloud Bursting Function.....	151
4.23.1	Overview of Cloud Bursting Function	151
4.23.2	Overview of Cloud Bursting Settings	153
4.23.3	Setting of Cloud Bursting Template	154
4.23.4	Setting of Cloud Bursting Node Group.....	157
4.23.5	Setting of Node Agent	161
4.23.6	Setting of Job Server on the instance	169
4.23.7	Overview of Cloud Bursting Policy	173
4.23.8	Setting of Scheduling	174
4.23.9	Submit a request.....	178
4.23.10	Policy of selection cloud	180
4.23.11	Forced Rerunning of Running Jobs for Cloud Bursting Requests	181
4.24	Request Assignment Mode	181
4.25	Indication of the scheduled start time of execution during the scheduling process	182
4.26	Caching of non-schedulable requests	183
Chapter 5.	Functions for SX-Aurora TSUBASA.....	184
5.1	Overview.....	184
5.2	VE Assignment Feature	184
5.3	Scheduling in VE Node Degradation.....	184
5.3.1	Overview of the Feature	184
5.3.2	Feature of Setting of Scheduling Method at VE Degradation	184
5.3.3	Display by sstat.....	186
5.4	HCA Assignment Feature	187
5.4.1	Overview of HCA Assignment Feature.....	187
5.4.2	HCA and the Information of Topology	189
5.4.3	Using HCA.....	195
5.4.4	Topology information and HCA.....	198
5.5	Scheduling with topology performance	198
5.5.1	Setting.....	199
5.5.2	Operation Considering Topology Performance.....	199
5.6	Suspend Jobs Using VEs.....	201
5.6.1	Suspend ve jobs with the 5.6.1 smgr(1M) command.....	201
5.6.2	Suspending VE Jobs to Run Urgent Requests	202

5.7	Dynamic JSV Priority.....	203
5.7.1	Overview	203
5.7.2	Settings	204
5.7.3	Calculation Method.....	204
5.7.4	Setting up calculation elements	205
Appendix.A	Update history	206
Index	207

Contents of Figures

Figure 2-1 JobManipulator component map	2
Figure 2-2 Example of Run Limit.....	17
Figure 2-3 Example of Assign Limit.....	25
Figure 2-4 Example of Complex Queue.....	27
Figure 2-5 The movement of a request to forward space on the scheduler map	33
Figure 2-6 Overtaking assignment for small-scale requests.....	36
Figure 3-1 Scheduler Map.....	43
Figure 3-2 Map Width and Pickup	46
Figure 3-3 Setting of the Map Width for each queue	47
Figure 3-4 The image of network topology node group definition	71
Figure 4-1 Image of Merge of ShareDB	93
Figure 4-2 Example of scheduling that prioritizes job execution.....	146
Figure 4-3 Example of network topology-first scheduling.....	147
Figure 4-4 Functional image of cloud bursting	151
Figure 4-5 Image of cloud startup	152
Figure 4-6 Cloud bursting configuration	153
Figure 4-7 Template and node group settings.....	153
Figure 4-8 The policy of cloud bursting.....	173
Figure 4-9 Setting of scheduling.....	174
Figure 5-1 SX-Aurora TSUBASA System	187
Figure 5-2 Execution of Program.....	188
Figure 5-3 Example of Topology Configuration.....	188
Figure 5-4 Example of Device Group with PCIeSW	189
Figure 5-5 Example of Device Group without PCIeSW	190
Figure 5-6 Assignment of VE at using HCA 1	197
Figure 5-7 Assignment of VE at using HCA 2	198
Figure 5-8 Example of the Operation Considering Topology Performance 1	200
Figure 5-9 Example of the Operation Considering Topology Performance 2	200

Chapter 1. Overview of JobManipulator

1.1 Introduction

JobManipulator is the job scheduler which is tailored to mixed operation of single and multi-node job execution on the large-scale cluster system. It is based on FIFO mechanism and enables scheduling that assigns the earliest time for job execution by managing unused amount of calculation resources (CPU, memory and others).

1.2 Features of JobManipulator

The main features of JobManipulator are as follows.

- Backfill scheduling which enables high and effective utilization of calculation resources based on the required resources of CPU, memory and others and the planned execution start time (ELAPSE time)
- Fair-share Scheduling which enables to control the priority of requests based on the resource usage and the distribution ratio of calculation resources per user and group
- The escalation which optimizes resource assignment of requests when a space of resource occurred by an end of execution before a plan or occurred by cancel of requests
- Advance Reservation feature (Resource Reservation Section) which enables to reserve the starting time of request execution and required calculation resources before execution
- Interrupting assignment managing which ensures assignment of calculation resource to the high-priority request(urgent request, special request) and enables immediate execution of the request
- Power-saving function which automatically power off execution host which does not have plan of execution of requests and Maximum Number of operation nodes can be set

In addition, JobManipulator has the following various scheduling functions, and it satisfies diverse user needs.

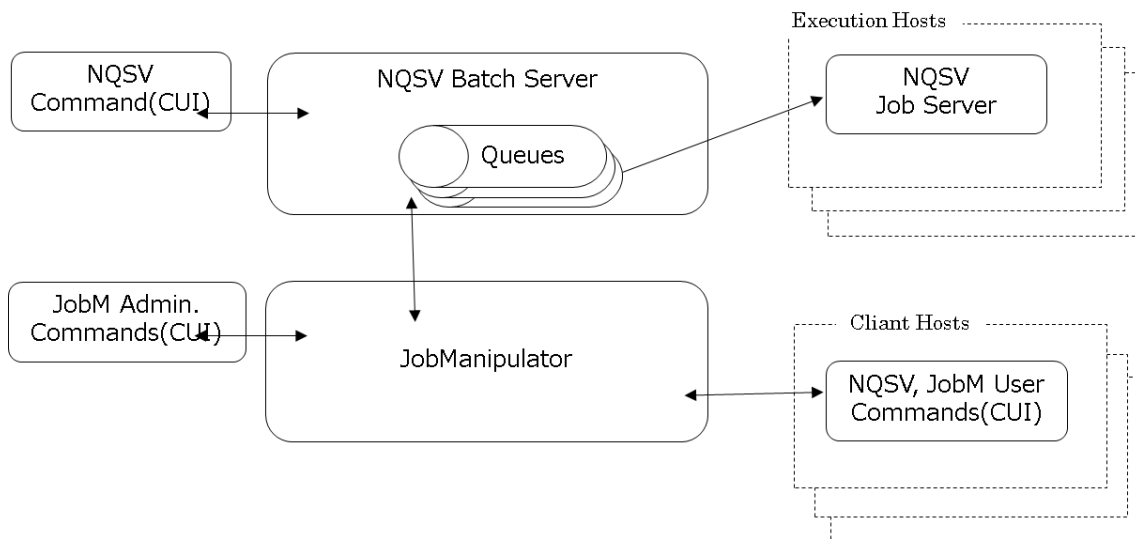
- The flexible scheduling setting functions by setting of run limit, the setting of assign limit, the setting of request priority order, the overtake control at pick-up, the setting of assign policy and the setting of a JSV assign policy, etc.
- Automatic setting function of the scheduling priority using more than 10 kinds of item and weighting of it.
- The Elapse Margin function which adds a margin time to elapsed time limit of a request so that the execution of a request does not overlap with other request
- The Custom Resource Function which defines a virtual resource and makes available in scheduling optionally
- Function of scheduling at node failure which reschedule request to normal node so that usage rates of calculation resources are maintained

Chapter 2. Environment Architecture

2.1 Configuration of JobManipulator

JobManipulator is job scheduler for NQSV exclusive use. It schedules requests which are submitted by each user managed by NQSV/Batch server.

Figure 2-1 JobManipulator component map



The following list shows the file configuration of JobManipulator.

files	explanation
/opt/nec/nqsv/sbin/nqs_jmd	JobManipulator scheduler
/opt/nec/nqsv/bin/sstat	The command to display scheduler information
/opt/nec/nqsv/sbin/smgr	The command to manage scheduler configuration
/opt/nec/nqsv/sbin/sushare	The command to manage user share
Default path: /etc/opt/nec/nqsv/nqs_jmd.conf	Configuration file The file defines the operation environment of JobManipulator. This file is a text file managed by the system administrator.
/etc/opt/nec/nqsv/jmtab	List of configuration file of JobManipulator. This file is a text file managed by the system administrator.
Default path: /etc/opt/nec/nqsv/jm_sharedb.conf	The file contains user share distribution value and usage data.

	This file is a text file managed by the system administrator.
Default path: /var/opt/nec/nqsv/nqs_jmd_<scheduler_id>.log	Log file The log file of JobManipulator.
/etc/opt/nec/nqsv/nqs_jmd_cmdapi.conf	Command environment file The file defines connection between JobManipulator commands and JobManipulator Scheduler (BatchServerHost). This file is a text file managed by the system administrator.

2.2 Package Configuration

The product of JobManipulator consists of following packages:

Product name	Package and the function contents
NEC Network Queuing System V/ ResourceManager	NQSV-Client-X.XX-X.x86_64.rpm A command interface function and user agent.(CUI)
NEC Network Queuing System V/ JobManipulator	NQSV-JobManipulator-X.XX-X.x86_64.rpm The batch scheduler.

Please refer to *NQSV User's Guide [Introduction]* for installation procedure of each software package.

2.3 Basic Environment Architecture

The minimum procedure for starting JobManipulator is described in this section.

2.3.1 Environment

The following installation environment is assumed for procedure of the creation of JobManipulator environment.

We assume that JobManipulator is installed in a batch server host.

Batch server host

Host name	IP address	Machine ID
bsv1.nec.co.jp	192.168.1.1	10

User

NQSV administrator user	root (batch server host)
General user	user (a batch server host and a client host)
Queue	
Execution queue name	execque1

2.3.2 Installation of Package

(1) Batch server host

Install the NQSV/JobManipulator package on the batch server host.

(2) Client hosts

Install the NQSV/Client package on the client hosts on which display the information of scheduler and do the management operation. sstat, smgr, and sushare commands are included in it. They are called JobManipulator command.

2.3.3 JobManipulator Start

JobManipulator starts if you execute following command with root privilege.

```
#systemctl start nqs-jmd
```

When JobManipulator is started first, the status of scheduling is stop.

Scheduling is started by execution of following command using **smgr**(1M) command after starting JobManipulator.

For details refer to "2.7.10 Set ON/OFF of Scheduling Feature".

```
#smgr -Po
Smgr: start scheduling
Start Scheduling.
```

2.3.4 Queue Setting

Queues to accept and execute a request on the NQSV system must be created. For creation of environment of NQSV system and creation and setting execution queues, refer to *NQSV User's Guide [Introduction]*.

For execution of requests, you need to bind execution queues to scheduler. Do bind with scheduler_id=1 because the default of scheduler ID is 1.

```
#qmgr -P m
Mgr: bind execution_queue scheduler execque1 scheduler_id=1
```

The execution queue bound once is bound automatically at the time of a next start of JobManipulator.

2.3.5 Setting of the Client Environment

To display information of JobManipulator and to do management operation of it you can use the JobManipulator command on a client host. The setting of it is as follows.

The file /etc/opt/nec/nqsv/nqs_jmd_cmdapi.conf is used for this setting. You should specify JobManipulator's running host name to jm_host in this file.

Add following line to /etc/opt/nec/nqsv/nqs_jmd_cmdapi.conf using editor with root privilege.

```
jm_host    bsv.nec.co.jp
```

The JobManipulator command and man data installed in following paths.

Command path

```
/opt/nec/nqsv/bin
/opt/nec/nqsv/sbin
```

man path

```
/opt/nec/nqsv/man (English)
/opt/nec/nqsv/man /ja (Japanese)
```

2.3.6 JobManipulator Stop

JobManipulator stops if you execute following command with root privilege.

```
#systemctl stop nqs-jmd
```

2.4 Unit Management

NQSV/JobManipulator has one unit as follows. For detail of unit, refer to the manual of systemd and systemctl.

Package Name	Unit	
	Target Unit Name	Service Unit Name
NQSV/JobManipulator	nqs-jmd.target	nqs-jmd.service

The unit which has .service extension is called service unit and manages daemon. The unit which has .target extension is called target unit, and controls multiple units. (NQSV/JobManipulator has one target unit)

Service unit is connected with target units. The connection become effective just after the installation of NQSV/JobManipulator. So NQSV/JobManipulator start automatically at starting of OS.

If you want to disable automatic starting of NQSV/JobManipulator at starting of OS, execute following command with root privilege. It makes ineffective the connection with nqs-jmd.target. .service extension can be omitted.

```
#systemctl disable nqs-jmd
```

If you want to enable automatic starting of NQSV/JobManipulator at starting of OS again, you need to execute following command to enable the connection with service unit.

```
#systemctl enable nqs-jmd
```

2.5 Setting of JobManipulator Start

2.5.1 Configuration file

You can specify scheduler ID, batch server host name, etc. in the configuration file on the host which NQSV/JobManipulator is installed. Default path of the configuration file is /etc/opt/nec/nqsv/nqs_jmd.conf. You need to add lines to configuration file as follows.

```
<directive>: <set value>
```


Settings of configuration file is as follows.

directive	set value	explanation
JM_SCHNO	scheduler ID	If you use scheduler ID except 1, you need to set this directive. The default is 1. You can specify an integer within the range of 0 to 15.
JM_SCHNAME	scheduler name	You can specify character string which is displayed by -D option of qstat(1) command etc.
BSV_HOST	batch server host name	When JobManipulator is installed on a different host from batch server host, you need to specify batch server host name to this directive. When this directive is omitted localhost is used.
JM_CMDPORT	port number	<JM_CMDPORT>+<JM_SCHNO> is port number which is used by JobManipulator command to connect to JobManipulator. The default is 13000.

Add directive to configuration file with root privilege if you need to.

The content in the configuration file (/etc/opt/nec/nqsv/nqs_jmd.conf) is loaded when starting JobManipulator. If configuration file contain error, JobManipulator stops.

If you want to use a different file from default configuration file, you need to edit the list of JobManipulator's configuration file which is written in /etc/opt/nec/nqsv/jmtab. Only "default" is written in /etc/opt/nec/nqsv/jmtab by default.

If you want to use a different file from default configuration file, you need to comment out the "default" line using "#" or delete it and add full path of different configuration file.

Example: in case of using /etc/opt/nec/nqsv/nqs_jmd_001.conf

```
# JobManipulator scheduler startup table
# "default" is /etc/opt/nec/nqsv/nqs_jmd.conf

# default
/etc/opt/nec/nqsv/nqs_jmd_001.conf
```

2.5.2 Starting of the multiple JobManipulator

The procedure when more than one JobManipulator where scheduling setting is different are connected to one batch server for batch requests and for interactive requests, etc., is explained in this section.

Firstly, you need to set unique scheduler ID to each JobManipulator.

That is, when you run multiple JobManipulator on one machine,

You need to make configuration files for each JobManipulator and specify different value to JM_SCHNO in each file.

Next, you create definition files of the service unit into /etc/systemd/system. /etc/opt/nec/nqsv/nqs-jmd0.service.sample is a sample file. You can use this file as a reference to create a definition file.

The following is a concrete example. In this example, the configuration file /etc/opt/nec/nqsv/nqs_jmd2.conf with 2 as scheduler ID is created. The following is an example nqs_jmd2.conf.

```
#
# Job scheduler configuration file
#

JM_SCHNO: 2
#JM_SCHNAME:
BSV_HOST: bsv.nec.co.jp
#JM_CMDPORT : 13000
#JM_RERUNWAIT: 600
```

You create a file that describes the startup parameters you give to JobManipulator. If you don't want to give JobManipulator a startup parameter, you don't have to give it to JobManipulator. In this example, assume that the file /etc/opt/nec/nqsv/nqs_jmd2.env is created. Specify the parameters to be given after JM_PARAM=.

```
# Environment variables for NQSV/JobManipulator
# Parameters to give NQSV/JobManipulator

JM_PARAM="-s ON"
```

You create a definition file of the service unit. In this case, this file is /etc/systemd/system/nqs-jmd2.service.

```
[Unit]
```

```
Description=NQSV JobManipulator - Job Scheduler
PartOf=nqs-jmd.target

[Service]
Type=forking
ExecStart=/opt/nec/nqsv/sbin/nqs_jmd -f /etc/opt/nec/nqsv/nqs_jmd2.conf $JM_PARAM
EnvironmentFile=/etc/opt/nec/nqsv/nqs_jmd2.env
PIDFile=/run/nec/nqsv/nqs_jmd.pid/02/jm_pid_file

[Install]
RequiredBy=nqs-jmd.target
```

ExecStart= specifies the name of the executable file and its startup parameters. You can specify the configuration file that is create by you in -f.

EnvironmentFile= specifies the file which contains the startup parameters. You can omit this line if you don't want to give any.

PIDFile= is a file that describes the process ID of JobManipulator. You replace the xx part below with the scheduler ID. If the scheduler ID is a single digit, please replace the 0 with a two digit ID.

```
PIDFile=/run/nec/nqsv/nqs_jmd.pid/xx/jm_pid_file
```

You execute the following command with root privilege after creating the definition file of service unit.

```
# systemctl daemon-reload
```

You execute the following command with root privilege to start JobManipulator with scheduler ID 2.

```
# systemctl start nqs-jmd2.service
```

You execute the following command with root privilege to start JobManipulator automatically at OS start up.

```
# systemctl enable nqs-jmd2.service
```

You execute the following command with root privilege to start JobManipulators that start automatically at OS start up.

```
# systemctl start nqs-jmd.target
```

You execute the following command with root privilege to stop JobManipulator with scheduler ID 2.

```
# systemctl stop nqs-jmd2.service
```

You execute the following command with root privilege to stop JobManipulators that start automatically at OS start up.

```
# systemctl stop nqs-jmd.target
```

You execute the following command with root privilege to deactivate that JobManipulator starts automatically at OS start up.

```
# systemctl disable nqs-jmd2.service
```

2.5.3 Updating from R1.05 or earlier to R1.06 or later

It is different to manage JobManipulator between R1.05 or earlier and R1.06 or later when multiple JobManipulators is running. Therefore, you need to update manually your environment to R1.06 or later if multiple JobManipulator is running in R1.05 or earlier. The procedure is shown below. You need to have root privileges to perform the following procedures.

In this case, if the following is written in /etc/opt/nec/nqsv/jmtab example.

```
# JobManipulator scheduler startup table
# "default" is /etc/opt/nec/nqsv/nqs_jmd.conf

/etc/opt/nec/nqsv/nqs_jmd.conf
/etc/opt/nec/nqsv/nqs_jmd2.conf
/etc/opt/nec/nqsv/nqs_jmd3.conf
```

In addition, suppose that each configuration file contained the following.

nqs_jmd.conf

```
#
# Job scheduler configuration file
#
```

```
JM_SCHNO: 1
#JM_SCHNAME:
BSV_HOST: bsv.nec.co.jp
#JM_CMDPORT : 13000
#JM_RERUNWAIT: 600
```

nqs_jmd2.conf

```
#
# Job scheduler configuration file
#

JM_SCHNO: 2
#JM_SCHNAME:
BSV_HOST: bsv.nec.co.jp
#JM_CMDPORT : 13000
#JM_RERUNWAIT: 600
```

nqs_jmd3.conf

```
#
# Job scheduler configuration file
#

JM_SCHNO: 3
#JM_SCHNAME:
BSV_HOST: bsv.nec.co.jp
#JM_CMDPORT : 13000
#JM_RERUNWAIT: 600
```

You update JobManipulator.

```
# rpm -Uvh NQSV-JobManipulator-1.06-xxx.x86_64.rpm
```

You create definition files of the service units based on `/etc/opt/nec/nqsv/nqs-jmd0.service.sample`. In this case, you create 3 files because 3 JobManipulators execute. Here are 3 definition files `nqs-jmd.service`, `nqs-jmd2.service` and `nqs-jmd3.service`.

nqs-jmd.service

```

[Unit]
Description=NQSV JobManipulator - Job Scheduler
PartOf=nqs-jmd.target

[Service]
Type=forking
ExecStart=/opt/nec/nqsv/sbin/nqs_jmd -f /etc/opt/nec/nqsv/nqs_jmd.conf $JM_PARAM
EnvironmentFile=/etc/opt/nec/nqsv/nqs_jmd.env
PIDFile=/run/nec/nqsv/nqs_jmd.pid/01/jm_pid_file

[Install]
RequiredBy=nqs-jmd.target

```

nqs-jmd2.service

```

[Unit]
Description=NQSV JobManipulator - Job Scheduler
PartOf=nqs-jmd.target

[Service]
Type=forking
ExecStart=/opt/nec/nqsv/sbin/nqs_jmd -f /etc/opt/nec/nqsv/nqs_jmd2.conf $JM_PARAM
EnvironmentFile=/etc/opt/nec/nqsv/nqs_jmd2.env
PIDFile=/run/nec/nqsv/nqs_jmd.pid/02/jm_pid_file

[Install]
RequiredBy=nqs-jmd.target

```

nqs-jmd3.service

```

[Unit]
Description=NQSV JobManipulator - Job Scheduler
PartOf=nqs-jmd.target

[Service]
Type=forking
ExecStart=/opt/nec/nqsv/sbin/nqs_jmd -f /etc/opt/nec/nqsv/nqs_jmd3.conf $JM_PARAM
EnvironmentFile=/etc/opt/nec/nqsv/nqs_jmd3.env
PIDFile=/run/nec/nqsv/nqs_jmd.pid/03/jm_pid_file

```

```
[Install]
```

```
RequiredBy=nqs-jmd.target
```

The points to note when creating the definition file are as follows.

1. The `-f` option of `nqs_jmd` for `ExecStart` is one of the configuration files to `jmtab` as a parameter.
2. Environment files is the name of the file containing the startup parameters given to `nqs_jmd`.
3. Match the scheduler ID in the configuration file to the directory specified in the `PIDFile`.

You make `systemd` reload the definition files of the service units.

```
# systemctl daemon-reload
```

You set to start automatically the service units if necessary.

```
# systemctl enable nqs-jmd.service nqs-jmd2.service nqs-jmd3.service
```

2.5.4 Start Option of JobManipulator

You can start JobManipulator with specifying IP address to perform failover or with specifying start/stop of scheduling feature or with specifying port number used for connection to BSV.

To perform failover, start JobManipulator with specifying the `-a` option. For details, refer to "4.11 Support for Failover System".

To specify scheduling status, start JobManipulator with specifying the `-s` option. Specifying `ON` to `-s` option means starting scheduling. Specifying `OFF` to `-s` option means stopping scheduling. Unspecifying of `-s` option means inheriting from status of previous starting. Unspecifying of `-s` option on first starting of JobManipulator means scheduling status is stop.

If you want to start JobManipulator by changing the port number to connect to BSV from the default (602), please specify the `-p` option.

You need to specify start option of JobManipulator to `JM_PARAM` in `/etc/opt/nec/nqsv/nqs_jmd.env`.

Example: in case of start JobManipulator with `-s ON`, `-a 192.168.1.1` and `-p 12345`.

```
# Environment variables for NQSV/JobManipulator
# Parameters to give NQSV/JobManipulator

JM_PARAM="--s ON -a 192.168.1.1 -p 12345"
```

2.5.5 Command environment file

The setting of the JobManipulator command using `/etc/opt/nec/nqsv/nqs_jmd_cmdapi.conf` file on client host is explained in this section.

By default, `sch_id` is 1 in `/etc/opt/nec/nqsv/nqs_jmd_cmdapi.conf` as follows. In this setting, JobManipulator commands connect to JobManipulator whose scheduler ID is 1.

When you specify other than 1 to `JM_SCHNO` in configuration file, you need to specify same number to `sch_id` in `/etc/opt/nec/nqsv/nqs_jmd_cmdapi.conf` to change default scheduler ID.

```
sch_id      1
```

When multiple JobManipulator runs, to change scheduler ID from default scheduler ID at using JobManipulator command you need to use `-s` option.

For details, please refer to NQSV User's Guide [Reference].

When you specify `JM_CMDPORT` directive at start of JobManipulator, to set port number to connect from JobManipulator command to JobManipulator you need to specify `jm_base_port` in `/etc/opt/nec/nqsv/nqs_jmd_cmdapi.conf`.

`<JM_CMDPORT>+<JM_SCHNO>` is port number which is used by JobManipulator command to connect to JobManipulator.

```
jm_base_port <port number which is specified to JM_CMDPORT>
```

Specify JobManipulator's running host name to `jm_host`.

```
jm_host <JobManipulator's running host name>
```

2.6 Scheduler Log File Setting

Set to output the log file of the scheduler. The following parameters can be set for the log.

The path of log file

The path name of the scheduler log file can be specified optionally. If not specified, the logs are output to the default path (/var/opt/nec/nqsv/nqs_jmd_<scheduler_id>.log). When the path name is changed while operating the scheduler, the file of previous path name is closed, a file of a new path name is created, and the logs are output.

Log level

A level from 1 to 5 can be specified. The default setting of the log level is 1, and it is recommended to use.

The size of log file

It is possible to set the log file size. The default setting of the logfile size is 2MB. In case the size is not set, it will be set to the current size. In case the size is set to 0, it will be set to unlimited.

The number of backup files

It is also possible to set the backup numbers of the log files and default is set to 1. In case the number of backup is not set, it will be set to the current numbers of backup. If the number of backup is set to 0, it will be 1. If it exceeds the set size when output the log file, it will make the backup files with the numbering and output the log files to the new files.

The **set logfile** subcommand of **smgr(1M)** sets these items.

```
# smgr -P m
Smgr:set logfile file =
/var/opt/nec/nqsv/nqs_jmd_<scheduler_id>.log size = 1000000 save
= 10
```

2.7 Scheduling Parameter Setting

This section describes how to set the parameters to schedule using JobManipulator.

2.7.1 Run Limit

"Run Limit" is the restriction value of request that can be executed simultaneously.

2.7.1.1 *Limits of the Number of Requests that can be Executed Simultaneously*

It is possible to limit the number of requests that can be executed simultaneously. If it exceeds the limit, a request cannot be assigned to the same time. The items and descriptions are as below.

This number is the amount of the requests which are assigned into scheduler map.

Item	Description
Per scheduler	
Request run limit for scheduler global_run_limit	This limits the number of requests which can be executed simultaneously in the scheduler.
Request run limit per users or for each user global_user_run_limit	This limits the number of requests which one user can execute simultaneously in the scheduler for all users or each user. The limit for each user is set by specifying a user or multiple users.
Request run limit per groups or for each group global_group_run_limit	This limits the number of requests which one group can execute simultaneously in the scheduler for all groups or each group. The limit for each group is set by specifying a group or multiple groups.
Per queue	
Request run limit in a queue queue_run_limit	This limits the number of requests which can be executed simultaneously in a queue.
Request run limit per user or for each user in a queue queue_user_run_limit	This limits the number of requests which one user can execute simultaneously in a queue for all users or each user. The limit for each user is set by specifying a user or multiple users.
Request run limit per group or for each group queue_group_run_limit	This limits the number of requests which one group can execute simultaneously in a queue for all groups or each group. The limit for each group is set by specifying a group or multiple groups.
Per complex queue	
Request run limit in a complex queue complex_queue_run_limit	This limits the number of requests which can be executed simultaneously in a complex queue.
Request run limit per user in a complex queue complex_queue_user_run_limit	This limits the number of requests which one user can execute simultaneously in a complex queue. Note that this limit cannot be set for each user. The same limit value is used for all users. When 0 is specified, the value will be unlimited. (Default: unlimited)

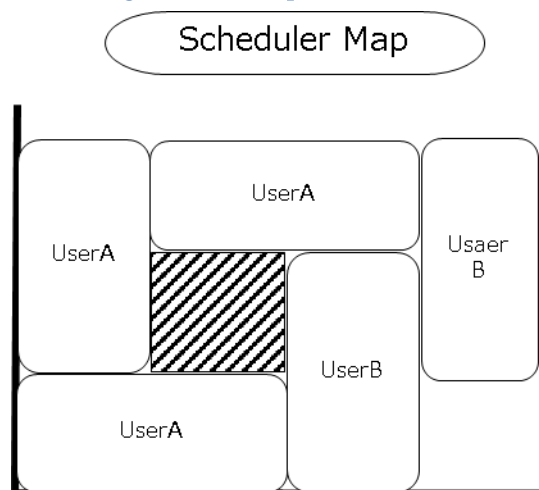
(Refer to "[2.3.5 Setting of Complex Queue Feature](#)" for details of complex queue.)


- * The limit for each user/group isn't set by default and other limit value is 0 (unlimited) by default.
- * When the limit for each user/group is set, it is limited by this value but not the limit for users/groups.

These limit values are set by using the set subcommand of **smgr(1M)**. If the limit is not necessary, the limit values can be ignored by setting to 0.

```
# smgr -P m
Smgr: set global_run_limit = 100
Smgr: set global_user_run_limit = 3
Smgr: set global_user_run_limit = 2 users = (userA,userB)
Smgr: set global_group_run_limit = 10 groups = groupA
Smgr: set queue_run_limit = 100 bq1
Smgr: set queue_user_run_limit = 2 bq1
Smgr: set queue_user_run_limit = 3 users = userA bq1
Smgr: set queue_group_run_limit = 15 bq1
Smgr: set queue_group_run_limit = 5 groups = groupA bq1
Smgr: set complex_queue_run_limit = 100 cq1
Smgr: set complex_queue_user_run_limit = 2 cq1
```

Figure 2-2 Example of Run Limit



Empty area of  is considered below.

- When the request run limit is 2:
There is no request which can be assigned to this area.

- When the request run limit per user is 2:
A userA's request cannot be assigned to this area but a userB's request can be assigned to this area.
- When the UserA's request run limit for each user is 3 and UserB's request run limit for each user is 2:
Both userA's request and userB's request can be assigned to this area.

The setting of per scheduler can be displayed by using **sstat(1)** with the **-S,-f** option. The setting of per queue can be displayed by using **sstat(1)** with the **-Q,-f** option. And the setting of each user/group can be displayed with **--limit** extra specified

The setting of each user/group can be deleted by using the delete subcommand of **smgr(1M)**.

```
# smgr -P m
Smgr: delete global_group_run_limit groups = groupA
Smgr: delete global_user_run_limit users = (userA,userB)
Smgr: delete queue_group_run_limit groups = groupA bql
Smgr: delete queue_user_run_limit users = userA bql
```

2.7.1.2 Limits of the Number of CPUs that can be Executed Simultaneously

It is possible to limit the number of CPUs that can be executed simultaneously. If it exceeds the limit, a request cannot be assigned to the same time.

CPU number that is limited by this function is calculated using limit on the number of CPUs that can be executed simultaneously of a request. This value can be displayed by using **qstat(1)** with **-f** options("CPU Number" of "Logical Host Resources" item). Refer to *NQSV User's Guide [Operation]* for details.

The items and descriptions are as below.

Item	Description
Per scheduler	
CPU run limit per users or for each user global_user_cpu_run_limit	This limits the number of CPUs which one user can execute simultaneously in the scheduler for all users or each user. The limit for each user is set by specifying a user or multiple users.
CPU run limit per groups or for each group global_group_cpu_run_limit	This limits the number of requests which one group can execute simultaneously in the scheduler for all groups or each group. The limit for each group is set by specifying a group or multiple groups.

Per queue	
CPU run limit per user or for each user in a queue queue user_cpu_run_limit	This limits the number of CPUs which one user can execute simultaneously in a queue for all users or each user. The limit for each user is set by specifying a user or multiple users.
CPU run limit per group or for each group queue group_cpu_run_limit	This limits the number of CPUs which one group can execute simultaneously in a queue for all groups or each group. The limit for each group is set by specifying a group or multiple groups.

* The limit for each user/group isn't set by default and other limit value is 0 (unlimited) by default.

* When the limit for each user/group is set, it is limited by this value but not the limit for users/groups.

These limit values are set by using the set subcommand of **smgr(1M)**. If the limit is not necessary, the limit values can be ignored by setting to 0.

```
# smgr -P m
Smgr: set global_user_cpu_run_limit = 150
Smgr: set global_user_cpu_run_limit = 100 users = (userA,userB)
Smgr: set global_group_cpu_run_limit = 1500
Smgr: set global_group_cpu_run_limit = 1000 groups = groupA
Smgr: set queue user_cpu_run_limit = 150 bq1
Smgr: set queue user_cpu_run_limit = 100 users = userA bq1
Smgr: set queue group_cpu_run_limit = 1500 bq1
Smgr: set queue group_cpu_run_limit = 1000 groups = groupA bq1
```

The setting of per scheduler can be displayed by using **sstat(1)** with the **-S -f** option. The setting of per queue can be displayed by using **sstat(1)** with the **-Q -f** option. And the setting of each user/group can be displayed with **--limit** extra specified.

UNLIMITED is displayed if the setting is 0(unlimited).

The setting of each user/group can be deleted by using the delete subcommand of **smgr(1M)**.

```
# smgr -P m
Smgr: delete global_user_cpu_run_limit users = (userA,userB)
Smgr: delete global_group_cpu_run_limit groups = groupA
Smgr: delete queue user_cpu_run_limit users = userA bq1
```

```
Smgr: delete queue group_cpu_run_limit groups = groupA bql
```

2.7.1.3 Limits the Number of VEs that can be Executed Simultaneously

It is possible to limit the number of VEs that can be executed simultaneously. If it exceeds the limit, a request cannot be assigned or escalation to the same time.

VE number that is limited by this function is calculated using limit on the number of VEs that can be executed simultaneously of a request. This value can be displayed by using `qstat(1)` with `-f` options("VE Node Number" of "Logical Host Resources" item). Refer to *NQSV User's Guide [Operation]* for details.

This function can be used on the environment that execution host is SX-Aurora TSUBASA system.

The items and descriptions are as below.

Item	Description
Per scheduler	
VE run limit per users or for each user global_user_ve_run_limit	This limits the number of VEs which one user can execute simultaneously in the scheduler for all users or each user. The limit for each user is set by specifying a user or multiple users.
VE run limit per groups or for each group global_group_ve_run_limit	This limits the number of requests which one group can execute simultaneously in the scheduler for all groups or each group. The limit for each group is set by specifying a group or multiple groups.
Per queue	
VE run limit per user or for each user in a queue queue_user_ve_run_limit	This limits the number of VEs which one user can execute simultaneously in a queue for all users or each user. The limit for each user is set by specifying a user or multiple users.
VE run limit per group or for each group queue_group_ve_run_limit	This limits the number of VEs which one group can execute simultaneously in a queue for all groups or each group. The limit for each group is set by specifying a group or multiple groups.

* The limit for each user/group isn't set by default and other limit value is 0 (unlimited) by default.

* When the limit for each user/group is set, it is limited by this value but not the limit for users/groups.

These limit values are set by using the set subcommand of **smgr(1M)**. If the limit is not necessary, the limit values can be ignored by setting to 0.

```
# smgr -P m
Smgr: set global_user_ve_run_limit = 150
Smgr: set global_user_ve_run_limit = 100 users = (userA,userB)
Smgr: set global_group_ve_run_limit = 1500
Smgr: set global_group_ve_run_limit = 1000 groups = groupA
Smgr: set queue_user_ve_run_limit = 150 bql
Smgr: set queue_user_ve_run_limit = 100 users = userA bql
Smgr: set queue_group_ve_run_limit = 1500 bql
Smgr: set queue_group_ve_run_limit = 1000 groups = groupA bql
```

The setting of per scheduler can be displayed by using **sstat(1)** with the **-S -f** option. The setting of per queue can be displayed by using **sstat(1)** with the **-Q -f** option. And the setting of each user/group can be displayed with **--limit** extra specified.

UNLIMITED is displayed if the setting is 0(unlimited).

The setting of each user/group can be deleted by using the delete subcommand of **smgr(1M)**.

```
# smgr -P m
Smgr: delete global_user_ve_run_limit users = (userA,userB)
Smgr: delete global_group_ve_run_limit groups = groupA
Smgr: delete queue_user_ve_run_limit users = userA bql
Smgr: delete queue_group_ve_run_limit groups = groupA bql
```

The change of run limit does not make an impact on assigned requests. Even if they exceed the changed limit, assignment of the assigned requests does not be changed. The changed limit becomes effective after the next scheduling (scheduling per interval or escalation)

2.7.2 Assign Limit

It is possible to set the number of requests that can be assigned simultaneously. The items and descriptions are shown below. This number is the amount of the requests which are assigned into scheduler map. It includes the number of running requests.

* There are no priorities among following limits. They are checked by each limit value, and it will stop assignment when it conflicts with any limit.

Item	Description
Per scheduler (core)	
Request assign limit for user global_user_assign_limit	<p>This limits the number of requests which can be assigned simultaneously for one user in the scheduler. If it exceeds this assign limit, a request cannot be assigned.</p> <p>Note that this limit cannot be set for each user. The same limit value is used for all users.</p> <p>When 0 is specified, the value will be unlimited. (Default: unlimited)</p>
Per queue	
Request assign limit for user in a queue queue user_assign_limit	<p>This limits the number of requests which can be assigned simultaneously for one user in a queue. If it exceeds this assign limit, a request cannot be assigned.</p> <p>Note that this limit cannot be set for each user. The same limit value is used for all users.</p> <p>When 0 is specified, the value will be unlimited. (Default: unlimited)</p>
Per complex queue	
Request assign limit for user in a complex queue complex_queue user_assign_limit	<p>This limits the number of requests which can be assigned simultaneously for one user in a complex queue. If it exceeds this assign limit, a request cannot be assigned.</p> <p>Note that this limit cannot be set for each user. The same limit value is used for all users.</p> <p>When 0 is specified, the value will be unlimited. (Default: unlimited)</p>
Per host	
Limit of the usable ratio of CPUs on the host executionhost cpunum_limit_ratio	<p>This limit controls the usable ratio of the number of CPUs on the host.</p> <p>This limits the ratio for simultaneous use of the total number of CPUs on the host, and the value is specified by the percent value divided by 100.</p> <p>When 1 (= 100%) is set for the CPU limit, jobs for the total number of CPUs on the machine are assigned. If the host has 8 CPUs and 2 (= 200%) is set for this limit, jobs for 16 CPUs can be assigned.</p> <p>Setting 0 (= 0%), this limit will be invalid and the number of CPUs is not checked for assigning jobs.</p>
Limit of the usable ratio of memory size on the host	<p>This limit controls the usable ratio of memory size on the host.</p> <p>This limits the ratio of total memory size which can be</p>

executionhost memsz_limit_ratio	used simultaneously on the host, and the value is specified by the percent value divided by 100. When 1 (= 100%) is set for the memory limit, jobs for the total memory of the machine are assigned. If the host has 10 GB of memory and 2 (= 200%) is set for this limit, jobs for 20 GB of memory can be assigned. Setting 0 (= 0%), this limit will be invalid and the memory size is not checked for assigning jobs.
Per RSG	
Limit of the usable ratio of CPUs per RSG executionhost rsg_cpunum_limit_ratio	This limit controls the usable ratio of the number of CPUs per RSG. This limits the ratio for simultaneous use of the number of CPUs set per RSG (Icpu), and the value is specified by the percent value divided by 100. When 1 (= 100%) is set for the CPU limit, jobs for the number of CPUs set per RSG (Icpu) are assigned. If Icpu = 4 and 2 (= 200%) is set for this limit, jobs for 8 CPUs can be assigned. Setting 0 (= 0%), this limit will be invalid and the number of CPUs is not checked for assigning jobs.
Limit of the usable ratio of memory size per RSG executionhost rsg_memsz_limit_ratio	This limit controls the usable ratio of memory size per RSG. This limits the ratio for simultaneous use of the memory size per RSG (Imem), and the value is specified by the percent value divided by 100. When 1 (= 100%) is set for the memory limit, jobs for the memory size set per RSG (Imem) are assigned. If Imem is 10 GB and 2 (= 200%) is set for this limit, jobs for 20 GB of memory can be assigned. Setting 0 (= 0%), this limit will be invalid and the memory size is not checked for assigning jobs.

RSG (Resource Sharing Group) is the name of each divided unit by resource division of execution host by CPuset function. Refer to *NQSV User's Guide [Management]* for details of the CPuset function.

-
- If you change RSG of a queue, it is necessary to delete the requests submitted in the queue and submit these requests again.
 - When using the CPU ratio (cpunum_limit_ratio) and Socket Scheduling function together, if the CPU ratio is set to a value other than 1, the number of CPUs that can be assigned to the job differs from the number installed on the execution host, so CPU core can not be assigned by core unit to the job. Therefore, enter the following contents in the job server environment setting file (/etc/opt/nec/nqsv/nqs_jsv.env) and start the job server.

JSV_PARAM = '--no-cpu-topology'

In this case, CPU core is assigned by socket unit to the job. Refer to 18.1 Socket Scheduling function in the *NQSV User's Guide [Management]* for details of the Socket Scheduling function.

These limit values are set by using the set subcommand of **smgr**(1M). When the resource limit is not necessary, the limit values can be ignored by setting to 0.

```
# smgr -P m
Smgr: set global_user_assign_limit = 10
Smgr: set queue user_assign_limit = 0 bql
Smgr: set complex_queue user_assign_limit = 0 cql
Smgr: set executionhost cpunum_limit_ratio = 2 hostname
Smgr: set executionhost memsz_limit_ratio = 0 hostname
Smgr: set executionhost rsg_cpunum_limit_ratio = 1.5 rsg_number =
0 hostname
Smgr: set executionhost rsg_memsz_limit_ratio = 0 rsg_number = 0
hostname
```

By specifying a node group instead of an execution host, the limit values can be set to all execution hosts in the specified node group.

```
# smgr -P m
Smgr: set executionhost cpunum_limit_ratio = 2 node_group = GrpA
Smgr: set executionhost memsz_limit_ratio = 0 node_group = GrpA
Smgr: set executionhost rsg_cpunum_limit_ratio = 1.5 rsg_number =
0 node_group = GrpA
Smgr: set executionhost rsg_memsz_limit_ratio = 0 rsg_number = 0
node_group = GrpA
```

If an execution host is added to a node group in *BSV*, to apply the settings that have been specified for the node group to the added execution host, specify the same settings to the added execution host individually, or specify the settings to the node group again. If an execution host is deleted from a node group, the settings specified for the node group remains as is. Therefore, it is necessary to specify the settings to each execution host of the node group.

The above settings can be specified only for the execution hosts that have been registered (attached) to the system.

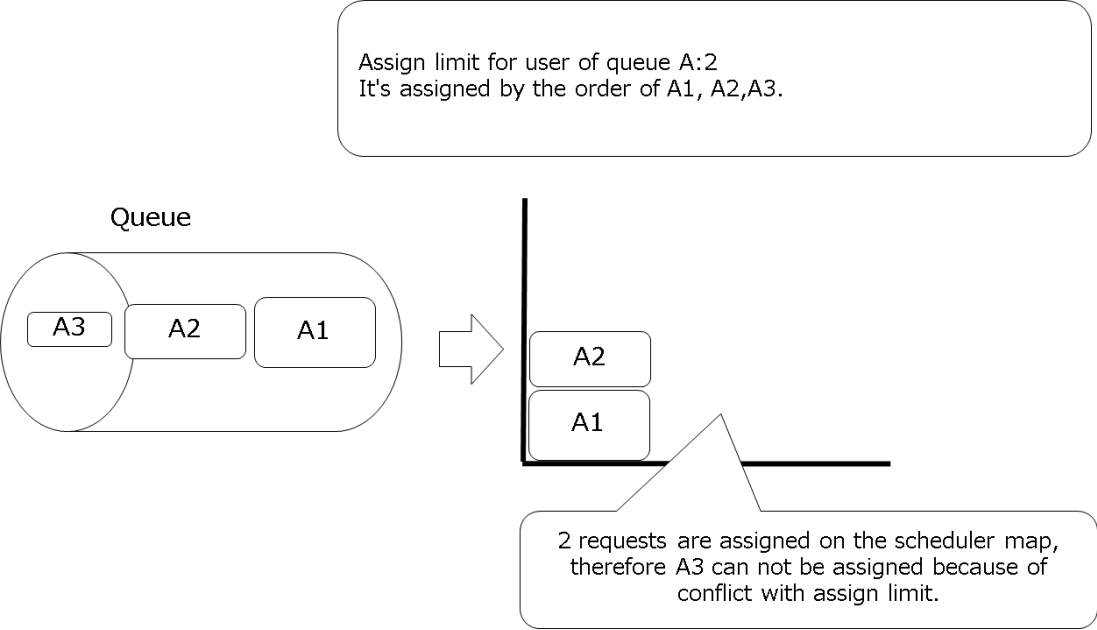
If an execution host is deleted (detached) from the system, the settings of the deleted execution host are also deleted from the DB.

The settings specified for the execution host can be displayed by using **sstat**(1) with **-E [-a] -g node_group** option displays the limit for available resources of the execution host belonging to the specified node group.

#sstat -E -g node_groupA		
ExecutionHost	CPUNRatio	MemRatio

hostA	2.000000	0.00000
(RSG 0)	1.500000	0.00000
(RSG 1)	0.500000	0.00000
hostB	2.000000	0.00000
(RSG 0)	1.500000	0.00000
(RSG 1)	0.500000	0.00000

Figure 2-3 Example of Assign Limit



2.7.3 Request Priority Order

It is possible to set the parameters to tune the order of priority for scheduling requests. (3.1.3 Scheduling Priority) The weight coefficients for parameters are specified by using the set subcommand of **smgr**(1M). The followings are the parameters which can be set.

Parameter Name	Description
weight_request_priority	weighted coefficient of request priority

weight_cpu_number	weighted coefficient of declared number of CPUs
weight_elapse_time	weighted coefficient of declared ELAPSE time
weight_memory_size	weighted coefficient of declared memory size
weight_job_number	weighted coefficient of number of jobs
weight_ve_number	weighted coefficient of declared number of VEs
weight_run_wait_time	weighted coefficient of period of waiting for execution from being submitted
weight_assign_wait_time	weighted coefficient of period of waiting for execution from becoming assignable
weight_restart_wait_time	weighted coefficient of period of waiting for restart from being suspended
weight_user_share	weighted coefficient of user share value
baseup_interrupted	based up value for a request suspended by urgent request
baseup_reschedule	based up value for rescheduled requests
baseup_user_definition	based up value for user definition
pastusage_weight_request_priority	weighted coefficient for past usage data of request priority
pastusage_weight_cpu_number	weighted coefficient for past usage data of number of CPU
pastusage_weight_elapse_time	weighted coefficient for past usage data of elapse time
pastusage_weight_memory_size	weighted coefficient for past usage data of memory size
pastusage_weight_ve_number	weighted coefficient for past usage data of number of VE

2.7.4 Queue Type

To use [4.1 Urgent Request](#) and [4.2 Special Request](#), set "urgent" or "special" to the queue type of the execution queue to start the request immediately by interrupting the running request. The queue type is specified by using the **set queue type** subcommand of **smgr(1M)**. Note that the setting above is valid only for JobManipulator, and it has no influence on the attribute of the execution queue.

```
# smgr -P m
Smgr: set queue_type = urgent bq1    set bq1 to an urgent queue
Smgr: set queue_type = special bq1   set bq1 to a special queue
```

```
Smgr: set queue_type = normal bql set bql to a normal queue
```

The queue type of a queue which has a request cannot be changed.

2.7.5 Setting of Complex Queue Feature

Outline of Functions

It is possible to set the following 3 limits for a group of multiple queues. That feature is called the complex queue feature and a group of multiple queues is called complex queue.

Request run limit

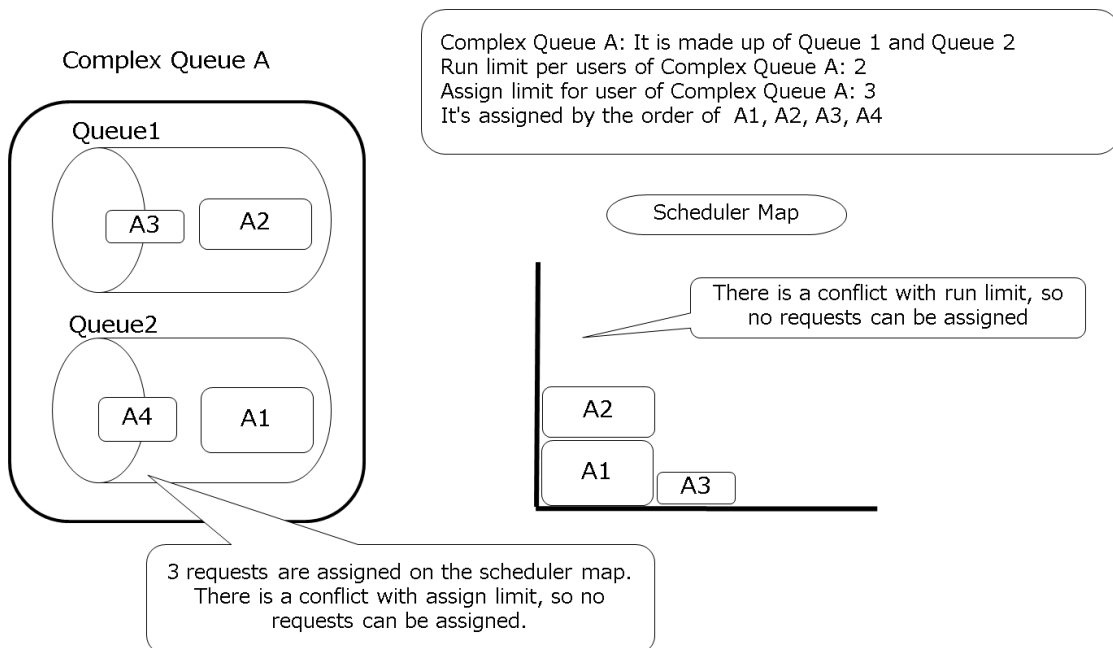
Request run limit for user

Request assign limit for user

This enables to set the limits not only for a queue but also for the complex queues. A queue is also able to belong to multiple complex queues and limits can be set more flexibly.

* The following is the image of complex queue.

Figure 2-4 Example of Complex Queue



It is set by using **smgr(1M)** command for setting the complex queue and adding/deleting the execution queues to/from complex queues. And it is possible to show the complex queue information by using **sstat(1)**. The setting of complex queue will be activated from scheduling after the setting is completed.

2.7.5.1 Creating Complex Queue

Create the complex queue by using **create complex queue** subcommand of **smgr(1M)**.

```
# smgr -P m
Smgr: create complex_queue = complex-queue-name queue = (queue-
name [,queue-name...])
```

- Specify the complex queue name to *complex-queue-name*.
- The longest name of complex queue is 63 characters.
- Each limit (Request run limit/Request run limit for user/Request assign limit for user) will be set to unlimited just after creating the complex queue.
- To *queue-name*, specify the name of execution queue that belongs to the created complex queue.
- The longest execution queue name is 15 characters.
- It is possible to specify the following queues as an execution queue.
 - The queue which belongs to other complex queues (The queues can belong to multiple complex queues.)
 - The execution queues whose queue type are different
 - The queue which is not controlled by JobManipulator.
- It is necessary to have the administrator privileges to create the complex queue.

In case there are any defects in the specified complex queue name or execution queue name, the complex queue will not be created.

* In following cases, it leads to an error and the complex queue is not created.

- In case the creating complex queue already exist
Error message: Specified complex queue already exists.
- In case the name of the creating complex queue exceeds 63 characters
Error message: Complex queue name too long.
- In case the name of the execution queue which belongs to complex queue exceeds 15 characters.
Error message: Execution queue name too long.
- In case a user who executed commands does not have the administrator privileges
Error message: Operation not permitted.

2.7.5.2 Deleting Complex Queue

Delete the complex queue by using the **delete complex_queue** subcommand of **smgr(1M)**.

```
# smgr -P m
Smgr: delete complex_queue = complex-queue-name
```

To *complex-queue-name*, specify the name of the complex queue to be deleted.
It is necessary to have the administrator privileges to delete the complex queue.

* In following cases, it leads to an error and the complex queue is not deleted.

- In case the deleting complex queue does not exist
Error message: Specified complex queue doesn't exist.
- In case a user who executed commands does not have the administrator privileges
Error message: Operation not permitted.

2.7.5.3 Adding Execution Queue to Complex Queue

Add the complex queue by using the **add complex_queue** subcommand of **smgr(1M)**.

```
# smgr -P m
Smgr: add complex_queue = (queue-name [,queue-name...]) complex-queue-name
```

- Specify the complex queue name to *complex-queue-name*.
- The longest name of execution queue is 15 characters.
- It is possible to specify the following execution queue to *queue-name*.
 - The queue which belongs to other complex queues (The queues can belong to multiple complex queues.)
 - The execution queue whose queue type are different
 - The queue can belong to complex queue in advance even if it is currently managed by other scheduler or it is to be managed by JobManipulator in the future.
- Execution queues can belong to multiple complex queues. And also, it is possible to activate all the complex queues.
- It is necessary to have the administrator privileges to add the complex queue.

In case the name of any specified execution queue exceeds the character limits, it does not add to any execution queue.

* In following cases, it leads to an error and execution queue is not added to the complex queue.

- In case the specified complex queue does not exist
Error message: Specified complex queue doesn't exist.
- In case a user who executed commands does not have the administrator privileges
Error message: Not permitted to modify attribute.
- In case the name of the specified execution queue exceeds 15 characters.
Error message : Execution queue name too long.

2.7.5.4 Removing Execution Queue from Complex Queue

Remove the execution queue by using the **remove complex_queue** subcommand of **smgr(1M)**.

```
# smgr -P m
Smgr: remove complex_queue = (queue-name [,queuename...])
complex-queue-name
```

* In following cases, it leads to an error and execution queue will not be removed from the complex queue.

- In case the specified complex queue does not exist.
Error message: Specified complex queue doesn't exist.
- In case the specified execution queue doesn't exist in complex queue.
Error message: Specified execution queue doesn't exist in complex queue.
- In case of specifying the same execution queue doubly
Error message: Same execution queue name were specified doubly.
- In case a user who executed commands does not have the administrator privileges
Error Message: Not permitted to modify attribute.

2.7.5.5 Setting of Complex Queue

It is possible to set limits to complex queue by using the following three subcommands of **smgr(1M)**.

[Request run limit]

```
# smgr -P m
Smgr: set complex_queue run_limit = run-limit complex-queue-name
```


- To *run-limit*, specify the request run limit to complex queue specified by *complex-queue-name*.
- It will be set to unlimited in case 0 is specified to *run-limit*.
- The defaults of these limits are unlimited.
- The maximum value of these limits are up to 2^{31} .

[Request run limit for user]

```
# smgr -P m
Smgr: set complex_queue user_run_limit = run-limit complex-queue-name
```

- To *run-limit*, specify the request run limits for user to complex queue specified by *complex-queue-name*.
- It will be set to unlimited in case 0 is specified to *run-limit*.
- The defaults of these limits are unlimited.
- The maximum value of these limits are up to 2^{31} .

[Request assign limit for user]

```
# smgr -P m
Smgr: set complex_queue user_assign_limit = assign-limit complex-queue-name
```

- To *assign-limit*, specify the request assign limit for user to complex queue specified by *complex-queue-name*.
- It will be set to unlimited in case 0 is specified to *run-limit*.
- The defaults of these limits are unlimited.
- The maximum value of these limits are up to 2^{31} .

* In following cases, it leads to be error and not to change the limits.

- In case the specified complex queue does not exist
Error message: Specified complex queue doesn't exist.
- In case the specified limits exceeds the maximum value of 2^{31}
Error message: Assign-limit out of bounds.
Run-limit out of bounds.
- In case a user who executed commands does not have the administrator privileges
Error message: Not permitted to modify attribute.

2.7.5.6 Showing Complex Queue Information

The information of the complex queue is displayed by using the **-C** option of **sstat(1)**.

```
# sstat -C
QueueName  Type      RL  URL  UAL  TOT  EXC  QUE  ASG  RUN  EXT  HLD  SUD
```

-												
Complex_1	-	ULIM	ULIM	ULIM	0	0	0	0	0	0	0	0
[jmq0]	Urgent	ULIM	ULIM	ULIM	0	0	0	0	0	0	0	0
[jmq1]	Special	ULIM	ULIM	ULIM	0	0	0	0	0	0	0	0
Complex_2	-	ULIM	ULIM	ULIM	0	0	0	0	0	0	0	0
[jmq2]	Normal	ULIM	ULIM	ULIM	0	0	0	0	0	0	0	0
[jmq4]	-	-	-	-	-	-	-	-	-	-	-	-

The displayed contents are followings.

- The name of complex queue
- The execution queue which belongs to the complex queue
- Request run limit
- Request run limit for user
- Request assign limit for user

* Regarding the queue which is not controlled by JobManipulator, only the queue name is displayed and other items displays "-" like as the above example of jmq4.

2.7.6 Setting of Escalation Feature

Early Execution

If a request finishes earlier than the scheduled execution time, the assigned space of node resources will be free. In order to fill this free space, the requests assigned backward on the same node are assigned if they can be executed immediately. The target request is selected with the following order.

1. A request with highest scheduling priority at the moving up moment
2. A request which was submitted earliest

JobManipulator performs "Early Execution" as default, and this feature is not influenced by the following settings of the escalation feature.

Setting the interval of escalation

JobManipulator supports the feature that checks free space on the scheduler map and moves requests to suitable spaces periodically at regular intervals. This feature is "Escalation". The value of interval of escalation can be set by **set escalation interval** subcommand of **smgr(1M)**. (Unit: scheduling interval)
There are the following two types of escalation.

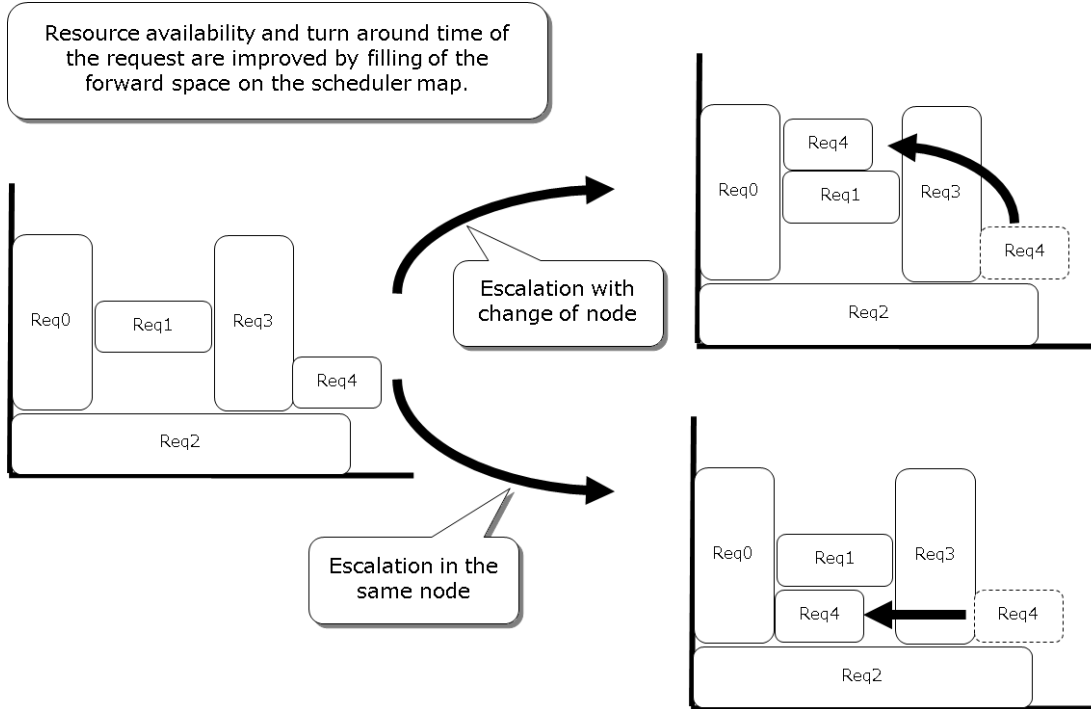
- **Forward Escalation**
The execution start time moves forward without change of node.

- **Side Escalation**

The execution start time moves forward with change of node.

If there are unfilled resources both forward (forward of the same node) and side (forward of the other node), forward escalation will be executed.

Figure 2-5 The movement of a request to forward space on the scheduler map



Note that a request with SUSPENDED status cannot be moved by escalation with node change.

By using the **set use_escalation** subcommand of **smgr(1M)**, it is possible to choose one of following three settings of escalation.

- **off** : Escalation is not executed
- **forward** : Forward Escalation.
- **all** : Forward Escalation or Side Escalation.

The default is **off**. (= not execute escalation)

Even if the escalation feature is set to off, early execution will be performed if a request finishes earlier and at the timing a request assigned backward can be moved.

	Early Execution	Escalation
Execution Timing	When a request is exiting	Executes with intervals defined by user
Target	The requests assigned on the same node with the finished request	All of the assigned requests

ON/OFF Setting	none	It can be set by smgr (1M)
-------------------	------	-----------------------------------

The following is an example of setting the escalation feature to off.

```
# smgr -P m
Smgr: set use_escalation = off
```

Specifying the conditions of selecting target requests to be escalated

Side Escalation is a high-load processing, because the batch job/jobs of target request need to be deleted once and then perform the process of staging. In order to avoid that Side Escalation happens frequently, following conditions of selecting the target request can be set in JobManipulator. The conditions can be set per queue.

- When the difference of scheduled start time between before escalation and after escalation is less than or equal to a limited time (Side Escalation Difference Limit), Side Escalation is not performed.
- When the planned start time is within a limited period from current time (Side Escalation Start Time Limit) and the number of jobs with execution host change is larger than a limited number (Side Escalation Number of Jobs Limit), Side Escalation is not performed.

The conditions of selecting target requests of escalation can be set by using the **set queue escalation_limit** subcommand of **smgr**(1M).

The conditions can be confirmed by **sstat -Q -f**.

- **Min Forward Time**: Side Escalation Difference Limit
- **No Escalation Period**: Side Escalation Start Time Limit
- **Max Side Escalation Jobs**: Side Escalation Number of Jobs Limit

Specifying adjusting time of estimated stage-in time

The stage-in time (the time of file staging) is considered when determining whether Side Escalation can be done to the request. The considered stage-in time is estimated by the largest value of stage-in time among the previous stage-in, however, the real stage-in time may fluctuate to a certain degree according to the operation. If the stage-in isn't completed by the scheduled start time of the request due to the fluctuation, the Side Escalation of this request will be canceled.

To reduce its impact, a feature adding a certain time to the estimated time of stage-in time is supported. The value should be set according to the degree of fluctuation of stage-in time of the requests in your system by system manager.

The value can be set by using **set stage-in_margin** subcommand of **smgr**(1M).

The setting can be displayed by using **sstat(1)** with the **-S,-f** option.

```
#sstat -S -f
JobManipulator Server Host: bsv.nec.co.jp
  JobManipulator Version    = R1.00
  JobManipulator Status     = Active
      :
  Keep Forward Schedule     = 0S
  Stage-in Margin = {
    Additional Margin for Escalation = 0S
    Stage-in Threshold = 0S
    First Stage-in Time = 0S
  }
  :
```

2.7.7 No Overtaking Control at Pick-up

The no overtaking control feature is supported in order to avoid that a large scale request is not executed eternally.

No overtaking control is set for each queue type with the **set use_overtake_priority** subcommand of the smgr (1M) command. If the setting is off, no overtaking control will not be performed

```
# smgr -P m
Smgr: set use_overtake_priority = on normal
```

In the above example, no overtaking control for normal queues is set to valid.

No overtaking control is realized by setting a scheduling priority threshold (threshold for overtaking control). If the scheduling priority of a request exceeds the no overtaking control threshold, scheduling of that request takes precedence. Until the execution start time of the request is determined, the request whose other scheduling priority is lower than the no overtaking control threshold cannot determine the scheduled execution start time.

The no overtaking control threshold is set with the **set overtake_priority** subcommand of smgr (1M) command. The value is set for each queue type.

The following is an example of setting the scheduling priority not to be overtaken for normal queues to 100.

```
# smgr -P m
Smgr: set overtake_priority = 100 normal
```

The no overtaking control setting does not affect requests submitted to queues that are more urgent than the set queue type. For example, even if the scheduling priority value of a request in the normal queue exceeds the threshold of no overtaking control of the normal queue, the request submitted to the special queue or the urgent queue can be overtaken.

No overtaking control is controlled from the timing when the request scheduling priority exceeds the no overtaking control threshold. Requests that overtake and determine the scheduled execution start time before the threshold is exceeded will start execution as scheduled.

2.7.7.1 Overtaking Assignment for Small-scale Requests

If you enable a scheduling priority value that disallows overtaking, and a large-scale request exceeds that priority value, then any requests submitted after that will not be assigned. Even if there is a small request that can be executed in the free space on the scheduler map, it will not be assigned and will wait for the large request to be assigned. It provides an overtaking assignment function for small-scale requests that allows assignment of requests that fit in the free space between the beginning of the scheduler map and the end of the scheduler map.

This feature allows small-scale requests to be executed in the gap first, without having to wait for large-scale requests to be assigned, leading to higher utilization and improved TAT (Turn Around Time) for jobs.

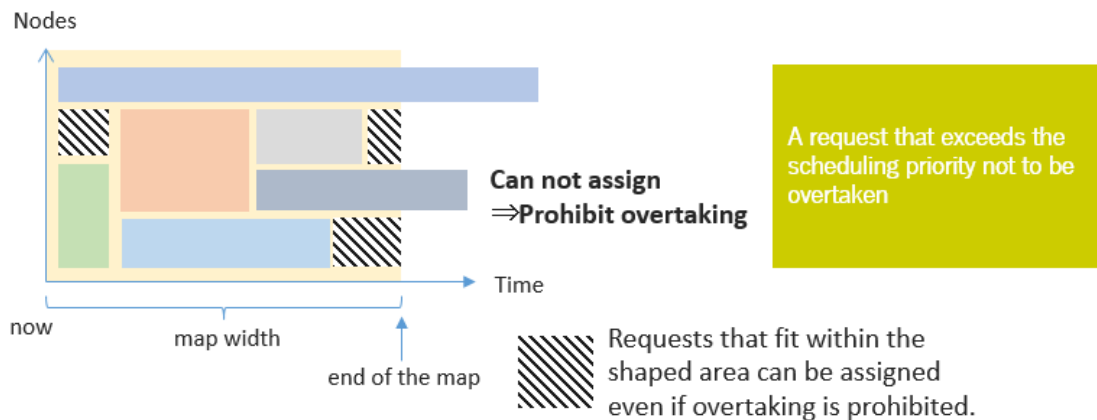


Figure 2-6 Overtaking assignment for small-scale requests

This function is turn on or off with the **set overtake allow_small_request** subcommand of the **smgr(1M)** command.

```
# smgr -P m
smgr: set overtake allow_small_request = on
```

When this function is on, small-scale requests that fit in the shaded area in Figure 2-6 will be assigned to overtake the yellow large-scale requests.

If a large-scale request is assigned and then escalated due to the termination of a forward job, the small request that was previously assigned to a gap in the scheduler map may be a disincentive to escalation.

2.7.8 Setting of Assign Policy

2.7.8.1 CPU number concentrated assignment or Resource balance assignment

JobManipulator supports the CPU number concentrated assignment policy to which jobs are assigned to one node until usable limit of the number of CPUs and the resource balanced assignment policy to which jobs are assigned so that number of using CPU may become uniform.

When the policy is "CPU number concentrated assignment", space nodes are secured as much as possible in order to make it easy to execute large scale request. When the policy is "Resource balanced assignment", it is possible to distribute load among nodes.

CPU number concentrated assignment (CPU_concentration)

Jobs are assigned to a node until usable limit of the number of CPUs. Jobs are not assigned to the other node until exceeds usable limit of the number of CPUs.
(Concentrated use of resources)

Resource balanced assignment (resource_balance)

Jobs are assigned to a node whose CPU usage is least at the assignment timing.

This assignment policy per scheduler can be set by the **set assign_policy** subcommand of **smgr(1M)**.

The default is **CPU_concentration**.

```
# smgr -P m
Smgr: set assign_policy = CPU_concentration
```

In the above example, the "CPU number concentrated assign" policy is set as the assignment policy of the scheduler.

The operator privilege or higher is required for this setting.

The assignment policy per queue can be set by the **set queue assign_policy** subcommand of **smgr(1M)**.

```
# smgr -P m
Smgr: set queue assign_policy=CPU_concentration bq1
```

In the above example, the "CPU number concentrated assignment" policy is set as the request assign policy of the queue "bq1".

The operator privilege or higher is required for this setting.

The assignment policy per queue is not set by default. In this case, the assignment policy per scheduler is applied. When the setting of the assignment policy per queue and the assignment policy per scheduler is different the assignment policy per queue is applied.

In the operation by which one job occupies a node and executes, the result of "CPU number concentrated assignment" and "resource balance assignment" are same. In such operation, it is recommended that you set the "CPU number concentrated assignment" to get relatively higher scheduling performance.

-
- This assignment policy per queue can be set only for the queues managed by JobManipulator.
 - When this assignment policy is changed, rescheduling is not performed and it is applied to the requests waiting to be assigned.
-

When "CPU number concentrated assignment" policy is set, request requiring GPU is assigned to a node by "GPU number concentrated assignment". That is, such request are assigned to a node in the smallest usable quantity of GPU.

When usable quantity of GPU is same, "CPU number concentrated assignment" is used.

2.7.8.2 Setting the Order of Execution Host Assignment

In JobManipulator, priority order of job servers (JSV Assign Priority) can be set for each queue, so that execution hosts can be assigned to requests based on it.

JSV Assign Priority can be set per job server of each queue by using the **set queue jsv assign_priority** subcommand of **smgr(1M)**.

```
Smgr: set queue jsv_assign_priority = 100 job_server_id = 1 bq1
```

In the above example, 100 is set to JSV Assign Priority of the job server whose ID is 1 of queue "bq1".

JSV Assign Priority can be set only to the queues that are bound with JobManipulator. JSV Assign Priority can be set to job servers on the attached execution host regardless of their bind state. The operator privileges or higher is required for specifying this setting. The default value is 0.

-
- The JSV Assign Priority set by this feature is used after job condition when selecting job servers. Therefore, when JSV Assign Priority is different between job servers, other lower assign policies will not be applied when selecting job servers. In order to make other lower assign policies effective among the execution hosts not shared with other queues, a same value should be set to these execution hosts as JSV Assign Priority.
 - By specifying a node group with the **set queue jsv assign_priority** subcommand, the JSV Assign Priority of JobServers included in the node group can be set all at once.
-

All JSV Assign Priorities of JobManipulator can be displayed by using **sstat -J**.

```
# sstat -J
```

JSVNO	Queue	Priority
----	-----	-----
0	bq1	200
1	bq1	100
1	bq2	100
2	bq2	200

In the above example, bq1 and bq2 share JSV 1. In this case, set a lower JSV Assign Priority to JSV 1.

JSV Assign Priorities of a queue can be displayed by using **sstat -Q -f -j**. Only JSV Assign Priorities of job servers that are bound with the queue are displayed. To display JSV Assign Priorities of job servers that are not bound with the queue, execute **sstat -Q -f -a**.

```
# sstat -Q -f -j bq1
```

Execution Queue: bq1			
...omission...			
JSV Assign Priority{			
JSV	0	=	200
JSV	1	=	100
}			
Request Statistical information:			
...omission...			

2.7.8.3 Setting of Priority or Disablement of Assignment Policy

The priority of either following assignment policies can be set and these assignment policies can be disabled as well.

- The assignment which is considered about network topology.
(Refer to [3.1.7.2 The assignment which considered a network topology](#))
- Preferential assignment policy of the node without staging job whose scheduled start time has been canceled.
(Refer to [3.1.7.3 Preferential Assignment Policy of the Node without Staging Job](#))

The priority and disablement can be set per scheduler by using the **set assign_policy_priority** subcommand of **smgr(1M)**.

```
#smgr -P m
Smgr : set assign_policy_priority = priority assign_policy =
assign_policy
```

Following policies can be set as "assign_policy".

network_topology	The assignment which is considered about network topology
staging_job	Preferential assignment policy of the node without any staging job whose scheduled start time has been canceled.

The following can be set as "priority".

low	The priority is low.
high	The priority is high.
disable	The assignment policy is disabled.

The defaults of above assignment policies are as follows.

network_topology	: high
staging_job	: low

Operator privilege is needed.

Please refer to 3.1.8.1 Priority of Assignment Policy for the criteria to determine the priority.

The setting can be displayed by using **sstat**(1) with the **-S,-f** option.

```
#sstat -S -f
JobManipulator Server Host: bsv.nec.co.jp
JobManipulator Version   = R1.00
JobManipulator Status    = Active
:
Request Assign Policy     = CPU concentration
Assign Policy Priority = {
  Network Topology = high
  Staging Job      = low
}
Global Run Limit          = 10
:
```

2.7.9 Setting of Wait Time of Rescheduling

By specifying a wait time of rescheduling, it is possible to wait a certain period of time from rescheduling a request if a stage-in or PRE-RUNNING (starting request execution) processing failed after assigning the request. This feature prevents request rescheduling from being repeated immediately after a stage-in or PRE-RUNNING processing failed. A wait time of rescheduling can be set to each queue by using the **set_queue_retry_time** subcommand of **smgr**(1M).

```
# smgr -P o
Smgr: set queue retry_time staging = 600 pre-running = 300 bq1

# 600 seconds is set as waiting time of rescheduling at Stage-in
processing failure and 300 seconds is set as waiting time of
rescheduling at PRE-RUNNING processing failure to queue "bq1".
```

In the above example, the following are set to queue "bq1".

- A wait time of rescheduling at Stage-in processing failure is set to 600 seconds.
- A wait time of rescheduling at PRE-RUNNING processing failure is set to 300 seconds.

The operator privileges or higher is required for specifying this setting. The default is 0 seconds.

In addition, if the request is made to wait for rescheduling because a stage-in or PRE-RUNNING processing failed, it is possible to release the job from such a state and specify the request as the rescheduling target again. This can be performed by using the **stop waiting_retry** subcommand of **smgr**(1M).

```
# smgr -P o
Smgr: stop waiting_retry request = 123.bsv.nec.co.jp # stop the
request 123.bsv.nec.co.jp to wait rescheduling.
```

The operator privileges or higher is required for specifying this setting.

2.7.10 Set ON/OFF of Scheduling Feature

You can set start and stop the scheduling by JobManipulator.

The **start_scheduling/stop_scheduling** subcommand of **smgr** (1M) sets this feature. Using the start scheduling subcommand loads starting scheduling. Using the stop scheduling subcommand loads stopping scheduling. The setting at immediate after installing of JobManipulator is stop scheduling

```
# smgr -P m
Smgr: start scheduling
Smgr: stop scheduling
```

The operator privileges or higher is required for specifying this setting.

The scheduling by JobManipulator for a queue starts by making the state of the queue active. However, the priority order among queues like prioritizing by queue priority may be ignored because of the setting order of activation.

In this case, stop the scheduling by JobManipulator using this feature, make the state of all queues active and start the scheduling by JobManipulator using this feature all at once, so that the priority order among queues is effective.

Chapter 3. Operation Management

3.1 Scheduling Basic Feature

This section describes the basic operation of JobManipulator.

3.1.1 Scheduler Map

JobManipulator uses scheduler map for assignment of the execution start time and resources. This enables planned distribution of calculation resources to jobs.

The scheduler map is an aggregation of cells (*i.e.* the pieces of calculation resources divided time-specially for each job server). The cell is minimum unit of width of the scheduler map (*i.e.* map width).

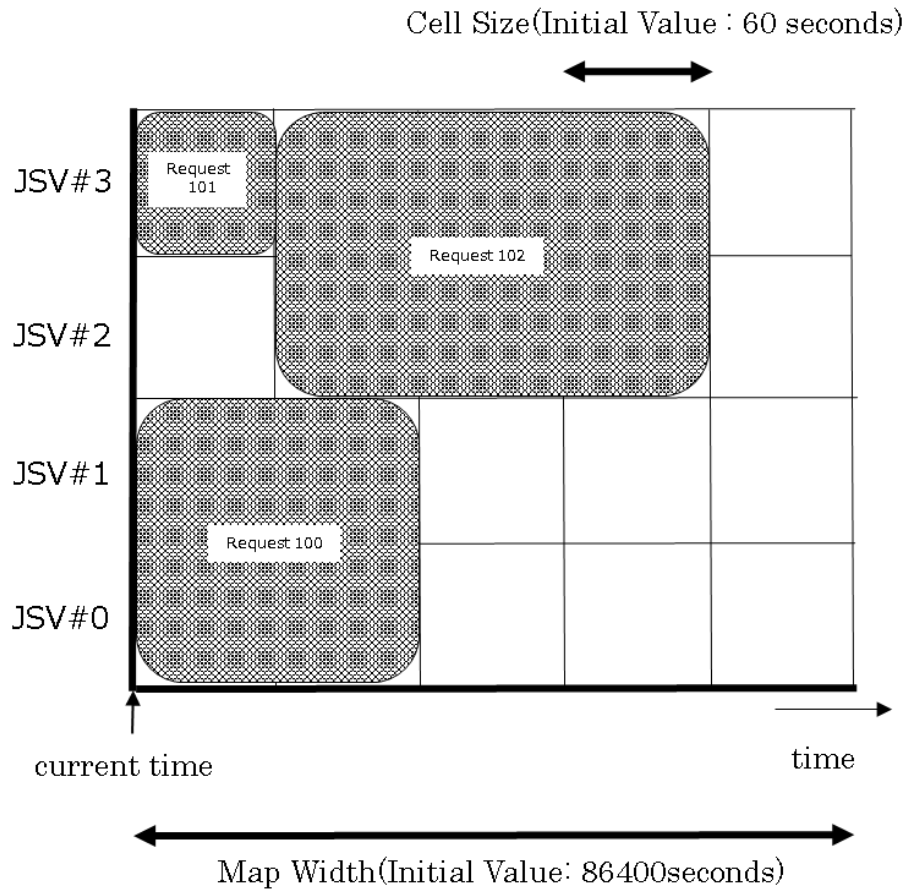
The size of a cell (cell size) is also called the scheduling interval. JobManipulator schedules at intervals of the scheduling interval.

The initial value of scheduling interval is 60 seconds. The initial value of map width is 1 day (86400 sec).

JobManipulator assigns jobs to the map. It depends on the setting of map width how many cells can be controlled in the future. For example, the number of cells per job server is 1440 ($= 86400/60$) when the value of map width is 1 day (86400 sec) and the value of scheduling interval is 1 minute (60 sec).

The following is a simple image of the scheduler map.

Figure 3-1 Scheduler Map



* In the above image, "Request: 100" and "Request: 101" are executing. After finishing executing "Request: 101", "Request: 102" will start to execute.

The Backfill scheduling is realized effectively by setting of long map width.

The Fair-Share scheduling is realized effectively by setting of short map width.

The Current Scheduling is realized by setting of enough short map width (than the declaration elapsed time of the request).

3.1.1.1 Map Width Set Up

It is possible to set the map width by the following two ways.

- A) Set the map width for each scheduler
- B) Set the map width for each queue

* Refer to the followings for details.

A. Set the map width for each scheduler

How to set up

The values of scheduling interval and map width can be set by the **set mapsize** subcommand of **smgr(1M)**. The minimum value of the map width is scheduling interval.

When the scheduling interval is changed, the scheduler map information is reconfigured and the scheduled start times of requests which were assigned on the map are deleted from the map.

In the case of increasing map width without changing scheduling interval, more requests can be assigned as map width increases. Conversely, when map width is decreased, the requests that doesn't fit in the decreased map will be targets of rescheduling and canceled on the map.

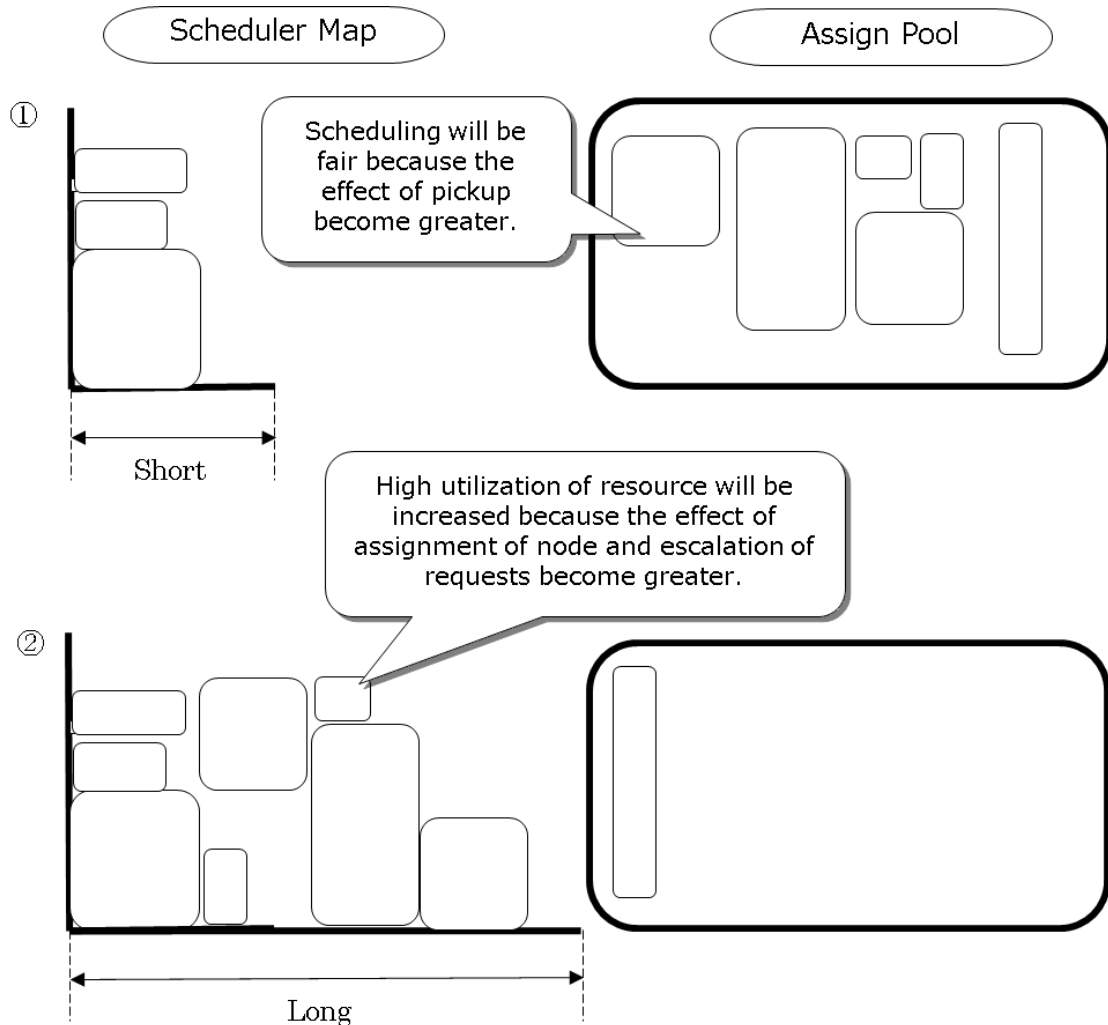
The map size must be set larger than the scheduling interval.

Relation of map width and request pick-up

The following picture is an image of map width and pick-up.

* The requests in the assign pool are aligned in order of scheduling priority (which is the calculated priority).

Figure 3-2 Map Width and Pickup



Assign Pool : The group of the requests which are not assigned on the map yet
(i.e. the request whose planned start time is not decided yet.)

Pick-up : Select the request in order to assign on the map.

It is possible to change the scheduling feature by setting the map width of JobManipulator.

Short map width: The fair-Share scheduling is conducted effectively

Long map width: The Backfill scheduling conducted effectively(=improvement of the resource usage)

B. Set the map width for each queue

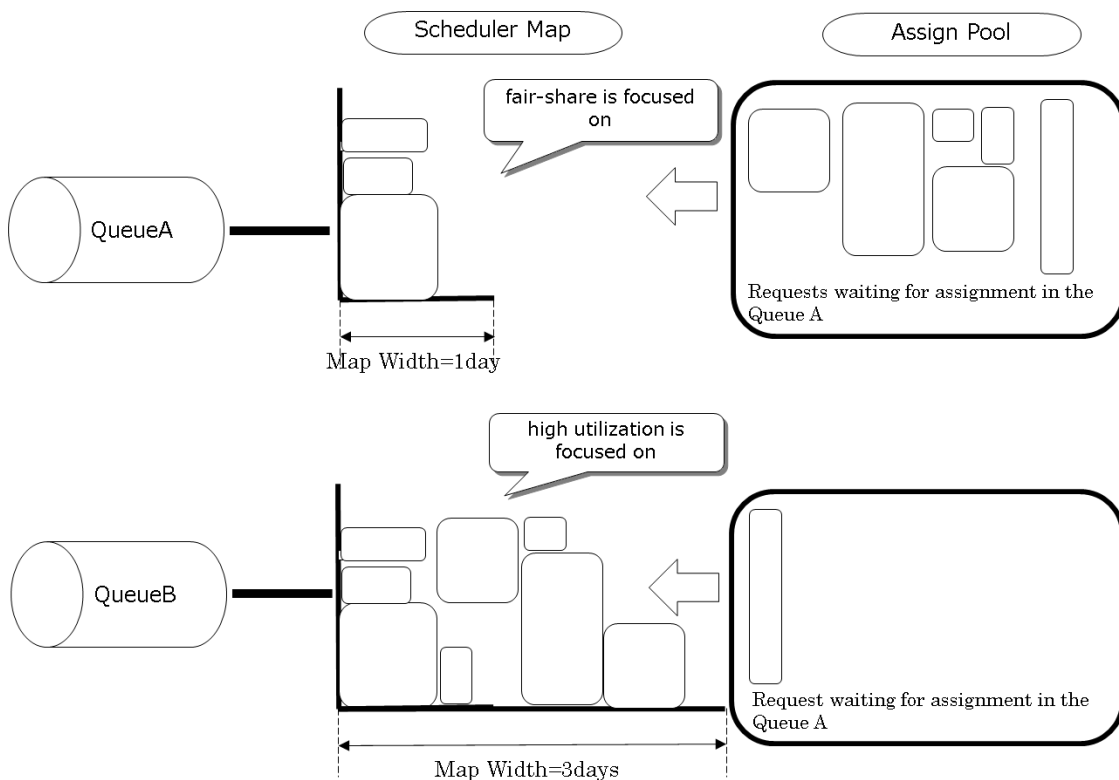
Map width can be set by each queue. By setting map width for each queue, it enables to have an appropriate scheduling operation feature (Fair-share or Backfill) for each queue. It can be more thorough and detailed scheduling operation than setting by each scheduler.

The following picture is an image of setting the map width by each queue.

* The scheduling feature can be set by each queue in one JobManipulator. The following operation is conducted in the picture below.

- In order to submit small scale jobs in "Queue: A", fair-share focused scheduling is conducted by setting map width to be short.
- In order to submit large scale jobs in "Queue:B", backfill focused scheduling (which increases resource usage rate) is conducted by setting map width to be long.

Figure 3-3 Setting of the Map Width for each queue



* In "Queue:A", map width are set to be short and fair-share focused scheduling is conducted. In "Queue:B", map width are set to be long and backfill focused scheduling is conducted.

It can be set one scheduling interval for each scheduler. The scheduling interval size cannot be set for each queue.

How to set up

The value of map width for each queue can be set by the **set queue mapsize sched_time** subcommand of **smgr**(1M). The scheduling interval cannot be set for each queue. The scheduling interval size which is set for each scheduler is used.

```
#smgr -P m
Smgr : set queue mapsize sched_time = sched_time queue-name

[Example] # In this example, it set the mapsize of "execqueue1" to
10000 seconds.
Smgr: set queue mapsize sched_time = 10000 execqueue1
Set queue Mapsize.
```

- Specify map width, which is set to the queue specified by *queue-name*, to *sched_time*.
- The map width is specified by seconds. The minimum value that can be set is the scheduling interval.
- In case the value specified to *sched_time* exceeds map width set by scheduler, it will be an error.
- The maximum value of map width set by each queue is the map width set by scheduler.
- The unset map width of the queue will be the map width set by scheduler.
- In case of changing the map width, all the jobs except executing jobs will be reassigned.
- The name of the queue whose map width is changed to the map width set by scheduler will be output to the log file (Default file name: `/var/opt/nec/nqsv/nqs_jmd_<scheduler_id>.log`).
- It is necessary to have the operator privileges or higher to set map width for each queue.

In case the smaller value than map width of each queue is specified to the map width of scheduler, the map width of the corresponding queue will be changed to map width set by scheduler.

Message : Some queues were changed to the mapsize of the system.

The name of the queue which map width was changed will be output to the log file (Default file name : `/var/opt/nec/nqsv/nqs_jmd_<scheduler_id>.log`).

* The following cases, it leads to an error and the map width is not changed.

- In case the map width specified for each queue is larger than the map width of scheduler
Error message: Mapsize too large. (Range of value = xx - xx)
- In case the queue which is not managed by JobManipulator is specified
Error message: No such queue. (name: <queue-name>).
- In case a user who executed commands does not have the operator privileges or higher
Error message: Not permitted to modify attribute.
- In case the map width specified for each queue is smaller than the scheduling interval.
Error message: Mapsize too small. (Range of value = xx - xx).

In case more than two queues share the same execution host, pay attention to the followings.

In case the map width of equal to or more than two queues which use the same host and the same RSG are changed, it will cause that the requests will be barely assigned to the queue which has a short map width. In order to avoid this situation, we recommend the operation as follows.

- In the operation changing the map width for each queue, we recommend the operation that each queue manage the different hosts.
- In case the queues manage the same host, we recommend managing the host resources divided by RSG to avoid confliction of the resource.

In case of not managing by RSG, the resource confliction also can be avoided by setting the CPU limit rate and memory limit rate of JobManipulator.

3.1.1.2 Map Width and Scheduling Interval Display Feature

A. Set the map width for each scheduler

The map width and the scheduling interval for each scheduler is shown by using **-S,-f** option of **sstat(1)**.

```
#sstat -S -f
JobManipulator Server Host: bsv.nec.co.jp
  JobManipulator Version   = R1.00
  JobManipulator Status    = Active
  Scheduler ID              = 1
  Schedule Interval       = 10S
  Schedule Time          = 86400S
:
```

B. Set the map width for each queue

The map width for each queue is shown by using **-Q,-f** option of **sstat(1)** command.

```
#sstat -Q -f
Execution Queue: jmq0
  Queue Type           = Normal
  Schedule Time        = 86400S
:
```

3.1.1.3 Notes on settings

JobManipulator schedules at intervals of the scheduling interval. When there is a large number of requests, scheduling may not be completed within the scheduling interval and processing may be terminated.

Adjust the scheduler map and scheduling interval according to the amount of requests at the site you are operating so that requests whose scheduled start time cannot be determined do not languish.

3.1.2 Real Time Scheduling

JobManipulator schedules when a request is submitted and starts executing when it is ready to start execution. As described in the **3.1.1 Scheduler Map**, JobManipulator determines the width of the scheduling interval as the smallest unit, but it is possible to start without waiting for the scheduling interval if the resource is free.

This feature has been enhanced with NQSV/JobManipulator R1.06. If you want to the scheduling and execution of requests at scheduling interval as before, the system administrator specify **off** in the **set realtime_scheduling** subcommand of the **smgr(1M)** command.

3.1.2.1 Request Realtime Scheduling Mode

In the scheduling interval, there is a dedicated time period for processing events such as JSV LINKDOWN and LINKUP. This time period is called the "event time". During the event time, we do not perform realtime scheduling of requests in order to concentrate on event processing. Therefore, requests submitted during the event time will be scheduled after the next interval, even if there is no event processing. If you change the request immediate scheduling mode to **always** using the **set realtime_scheduling mode** subcommand of the **smgr(1M)** command, requests will be scheduled and executed immediately, even during event times. By default, realtime scheduling of requests is not performed during event times. If you specify the request realtime scheduling mode to **default** in the **set realtime_scheduling mode** subcommand of the **smgr(1M)** command, the setting returns to the default value.

The value of the request realtime scheduling mode can be confirmed in the "Realtime Scheduling Mode" column of **sstat -Sf**.

3.1.3 Usage Data Collection and Adjustment

3.1.3.1 Collection of usage data

JobManipulator collects the amount of actual used system resources for each batch request and stores the accumulated value after calculating for each user.

Following system resources are collected for calculating usage data:

Number of CPU	Number of CPU (declared value by user) x Time (Elapsed)	Calculated by each request
Elapse Time	Elapse time (Usage value)	Calculated by each request
Memory amount used	Memory amount used (Measured) x Time (Elapsed)	Calculated by each request
Request Priority	Request Priority (declared value by user) x Time (Elapsed)	Calculated by each request
Number of VE	Number of VE (declared value by user) x Time (Elapsed)	Calculated by each request

Usage data is accumulated together with adjusted past usage data by half decay time.

Usage data is accumulated while reducing usage data values accumulated for each user at every request termination.

It is possible to set half-life decay time by the **set half_reduce_period** subcommand of **smgr(1M)**.

3.1.3.2 Reduction of usage data values

JobManipulator accumulates usage data while reducing past usage data values accumulated for each user at every request termination.

```
New usage data value = Usage data (accumulated) * 0.5 ^ (( current  
time - previous time ) / Half life decay time ) + usage data value  
obtained at current time
```

3.1.3.3 Reflection of usage data values to the scheduling priority

The weight can be specified to each component used for usage data values such as the number of CPU and elapsed time and the values are compared relatively with a scale set by system. These weight coefficients can be specified by set subcommand of **smgr(1M)**. The parameters are as below.

Parameter name	Description
pastusage_weight_request_priority	weight coefficient for usage data of request priority
pastusage_weight_cpu_number	weight coefficient for usage data of number of CPU
pastusage_weight_elapse_time	weight coefficient for usage data of elapse time
pastusage_weight_memory_size	weight coefficient for usage data of memory size
pastusage_weight_ve_number	weight coefficient for usage data of number of VE

Normalized past usage is used to calculate scheduling priority. The normalization formulas are as follows.

(a) Number of CPU

The declared value is taken as the number of CPU. Usage data is accumulated together with adjusted past usage data by half decay time.

It will be the value of 1 when the standard CPU number is (assumed to be) used without limit.

Normalization formula:

CPU usage data (accumulated value) / (Standard Number of CPUs / loge2 * Half life decay time)

(b) Elapse Time

Usage data is accumulated together with adjusted past usage data by half decay time.

It will be the value of 1 when the standard CPU number is (assumed to be) used without limit.

Normalization formula:

Elapse time usage data (accumulated value) / (Standard Number of CPUs / loge2 * Half life decay time)

(c) Used memory amount

Usage data is accumulated together with adjusted past usage data by half decay time.

It will be the value of 1 when standard all installed memory is (assumed to be) used without limit.

Normalization formula:

Memory usage data (accumulated value) / (Standard total memory size / loge2 * Half life decay time)

(d) Request Priority

The declared value is taken as the request priority. Usage data is accumulated together with adjusted past usage data by half decay time.

It will be the value of 1 when a request whose priority is 1023 is (assumed to be) kept executing unlimitedly.

Normalization formula:

Request priority usage data value(accumulated value) / (1023 / loge2 * Half life decay time)

(e) Number of VE

The declared value is taken as the number of VE. Usage data is accumulated together with adjusted past usage data by half decay time.

It will be the value of 1 when the standard VE number is (assumed to be) used without limit.

Normalization formula:

VE usage data (accumulated value) / (Standard Number of VEs / loge2 * Half life decay time)

3.1.3.4 Display of usage data values

Usage data values can be displayed by -S option of **sushare**(1). The usage data of each user and the total usage data of each group are displayed hierarchically by group. "*" is displayed at the beginning of group name as follows, if the displayed data is the total usage data of a group.

Parameter name	Description
Group Name	Display group name. It is displayed at the beginning when usage data of a group is displayed.
User	Display User name or group name. If it is group name, "*" will be displayed at the beginning of the group name.
Acctcode	Display account code of a user. If no account code for the user, "none" is displayed. "none" is displayed for usage data of a group.

Share	Display share distribution ratio of each user or group. Refer to User Share Value for share distribution ratio.
PU_cpunum	Display a user's or group's CPU usage data and its percentage of the system total.
PU_memsz	Display a user's or group's memory usage data and its percentage of the system total.
PU_elapstim	Display a user's or group's usage data of elapsed time and its percentage of the system total.
PU_reqpri	Display a user's or group's usage data of request priority and its percentage of the system total.
PU_venum	Display a user's or group's VE usage data and its percentage of the system total.

An example is shown as follows.

[Group Name : TOP_GROUP] <== #group name							
User	Acctcode	Share	PU_cpunum (%)	PU_venum (%)	PU_memsz (%)	PU_elapstim (%)	PU_reqpri (%)
*nec	none	0.333	4.190M (50.002)	4.190M (50.002)	3.996M (50.002)	1163:58:00 (50.002)	4.190M (50.002)
#group total usage data of each							
*nqs	none	0.667	4.190M (49.998)	4.190M (49.998)	3.996M (49.998)	1163:53:44 (49.998)	4.190M (49.998)
#group total usage data of each							
[Group Name : nec]							
User	Acctcode	Share	PU_cpunum (%)	PU_venum (%)	PU_memsz (%)	PU_elapstim (%)	PU_reqpri (%)
necusr1	none	0.167	2.095M (25.001)	2.095M (25.001)	1.998M (25.001)	581:59:01 (25.001)	2.095M (25.001)
#usage data of each user							
necusr2	none	0.167	2.095M (25.001)	2.095M (25.001)	1.998M (25.001)	581:58:58 (25.001)	2.095M (25.001)
[Group Name : nqs]							
User	Acctcode	Share	PU_cpunum (%)	PU_venum (%)	PU_memsz (%)	PU_elapstim (%)	PU_reqpri (%)
nqsusr1	none	0.167	1.048M (12.500)	1.048M (12.500)	1022.954K (12.500)	290:58:24 (12.500)	1.048M (12.500)
nqsusr2	none	0.167	1.048M (12.500)	1.048M (12.500)	1022.955K (12.500)	290:58:25 (12.500)	1.048M (12.500)
nqsusr3	none	0.167	1.048M (12.500)	1.048M (12.500)	1022.956K (12.500)	290:58:26 (12.500)	1.048M (12.500)
nqsusr4	none	0.167	1.048M (12.500)	1.048M (12.500)	1022.956K (12.500)	290:58:26 (12.500)	1.048M (12.500)

3.1.4 Scheduling Priority

3.1.4.1 Scheduling Priority

The Scheduling Priority is used to decide the order of execution host assignment (picking up of request) or the order of escalation in the execution queue. The elements for calculation of the scheduling priority are shown below.

The requests are picked up in order of the priority of the execution queue to which the requests are submitted. When multiple requests are existent in the execution queue, the order depends on the value of scheduling priority of each request. The scheduling priority is calculated based on the following elements.

- User share value
- Usage data value
- User rank
- Request priority
- Amount of required resources of the request
- Wait time for execution from being submitted
- Wait time for execution from becoming assignable

3.1.4.2 Formula of the Scheduling Priority

The formula for calculation of the scheduling priority is as follows.

```
Scheduling Priority =
User Share Value          x weight coefficient (User Share)
+ Usage Data Value (Total)
+ User Rank (Normalized) x weight coefficient (User Rank)
+ Request Priority (Normalized)
  x weight coefficient (Request Priority)
+ Declared Number of CPUs (Normalized)
  x weight coefficient (Declared Number of CPUs)
+ Declared Elapsed Time (Normalized)
  x weight coefficient (Declared Elapsed Time)
+ Declared Memory Size (Normalized)
  x weight coefficient (Declared Memory Size)
+ Number of Jobs (Normalized)
  x weight coefficient (Number of Jobs)
+ Declared Number of VEs (Normalized)
  x weight coefficient (Declared Number of VEs)
+ Wait Time for Execution from being submitted
  x weight coefficient (Wait Time for Execution from being
  submitted)
+ Wait Time for Execution from becoming assignable
  x weight coefficient (Wait Time for Execution from becoming
  assignable)
+ Wait Time for Restart
  x weight coefficient (Wait Time for Restart)
(+ base-up for a request suspended by urgent request)
(+ base-up for a rescheduled request)
(+ base-up defined by user)
```

The details of each item are described below.

User Share Value

The "User Share Value" is calculated by the scheduler, using a configuration file which sets the share ratio. (Share distribution ratio configuration file)

The share distribution ratio configuration file is read by **sushare**(1) command. If it isn't specified the configuration file when using **sushare**(1), the default path of this configuration file is `/etc/opt/nec/nqsv/jm_sharedb.conf`. The following is the format of the file.

```
TOP_GROUP = {
    (G:Group-name | U:User-name[:Account-name]) = Share-distribution-
    ratio
    (G:Group-name | U:User-name[:Account-name]) = Share-distribution-
    ratio
    ...
}
Group-name = {
    (G:Group-name | U:User-name[:Account-name]) = Share-distribution-
    ratio
    (G:Group-name | U:User-name[:Account-name]) = Share-distribution-
    ratio
    ...
}
...
```

A user belongs to one of *Group-names*, and each group is managed by the tree structure.

The top group of the tree structure is TOP_GROUP, and *Share-distribution-ratio* sets the distribution ratio in the group.

When *Account-name* is omitted, users are not distinguished according to the account code.

The user share value of a user who does not exist in the share distribution ratio configuration file is 0.

The following is a setting example.

```
/etc/opt/nec/nqsv/jm_sharedb.conf
TOP_GROUP = {
    U:root=50
    G:GroupA=30
    G:GroupB=20
}
GroupA = {
    U:User1=20
    U:User2=10
```

```

}
GroupB = {
U:User10=10
U:User11=10
U:User12=10
U:User13=10
U:User14=10
}

```

Usage Data Value(Total)

```

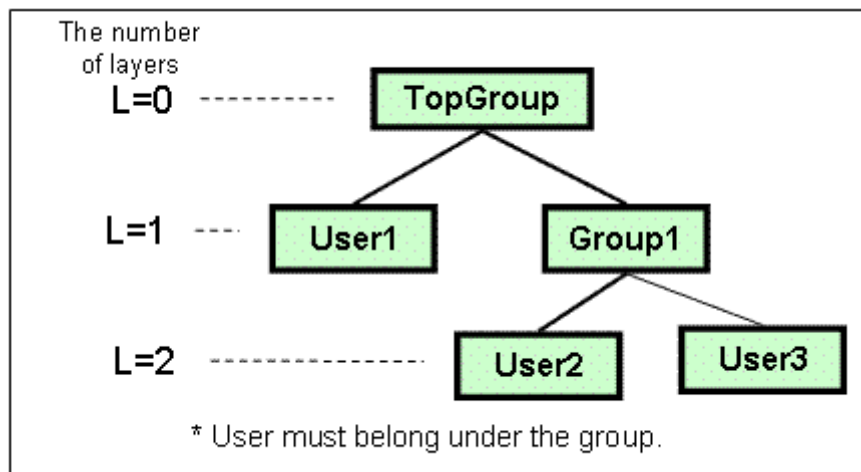
UaActual Usage Data Value (Total) =
Number of CPUs (Normalized)x weight coefficient (for usage data of Number of CPUs)
+ Elapsed Time (Normalized) x weight coefficient (for usage data of Elapsed Time)
+ Memory Size (Normalized) x weight coefficient (for usage data of Memory Size)
+ Request Priority (Normalized) x weight coefficient (for usage data of Request
Priority)
+ Number of VEs (Normalized)x weight coefficient (for usage data of Number of VEs)

```

User Rank

A user rank is a value calculated according to an actual usage and a predetermined share value, and used to decide the order(priority) among users located hierarchically.

- Calculation method of the User rank
Users are managed with a hierarchical structure. The share and usage data of a lower layer are managed in total by the parent node to which it belongs. The high-ranked share has stronger influence than the lower-ranked share. In particular, the share of the highest layers is given priority.



- Calculation Method and Formulas
 1. Calculates the rank value of each user.

- (i) The user share value divided by the total usage data is used in order that the ranking value of all users can be compared relatively.
- (ii) Above value is divided by coefficient which is composed of the number of users and layers(the number of the hierarchy) in order to correct it to the balanced value in hierarchical user structure. In other words, the logarithmic value of the total number of users(log N) from the higher layers and the layer to which the user belongs multiplied with the number of layers(= L : the top layer is assumed to be 0) is the coefficient.
- log N: The value will be greater as the number of users (N) is greater. The more users who share resource exist, the greater the denominator is. Then, the usage data (which is calculated at (i)) is corrected to be smaller.
- L : The value multiplied by the number of layers will be the coefficient for the purpose that the high-ranked share is given priority and the share value among the highest ranked sites will have much influence.

2. Calculates the user rank of the user located at a hierarchical position. This means the amount of the value of all the direct higher users calculated with the method described at 1.

3. Normalizes the value to be the value from 0 to 1. The denominator at normalization is different each layer to which the user belongs.

- (i) $r = (\log R) / (\log N * L)$
 $R = \text{User share value} / \text{Total of Usage data}$
 $(0.01 \leq R \leq 100. \text{ The value of out of the range will be the maximum or the minimum value.})$
 $N = \text{The number of users (The amount of users in the layers from top to the user. The top layer is not included.)}$
 $L = \text{The number of layers (The top layer is assumed to be 0.)}$
- (ii) $\text{UserRank} = r_1 + r_2 + r_3 + \dots + r_{(L-1)}$
- (iii) The maximum value of the numerator of r of (i) is +2,
and the minimum value is -2.
Therefore, the maximum value of r is $+2/(\log N * L)$,
and the minimum value is $-2/(\log N * L)$.
The maximum value of the total amount is equal to $+2(1/(\log N_1 * 1) + 1/(\log N_2 * 2) + \dots + 1/(\log N_L * L))$
The minimum value of the total amount is equal to $-2(1/(\log N_1 * 1) + 1/(\log N_2 * 2) + \dots + 1/(\log N_L * L))$
Normalization Formula = $0.5 + \text{UserRank} / (2 * 2(1/(\log N_1 * 1) + 1/(\log N_2 * 2) + \dots + 1/(\log N_L * L)))$

The "Request Priority" is specified by the **-p** option of **qsub**. It will be the value of 1 in the case request priority = 1023, and will be the value of 0 in the case request priority = -1024.

Normalization Formula $(\text{Request Priority} + 1024) / (1023 + 1024)$

Required Resource Usage of Requests

The following resource limits are used for required resource usage.

Number of CPUs	The number of CPUs that can be used simultaneously per logical host qsub --cpunum-lhost
Elapsed Time	The elapsed time per request qsub -l elapstim_req
Memory (optional)	The memory size per logical host qsub --memsz-lhost
Number of Jobs	The number of jobs qsub -b
Number of VEs	The number of VEs that can be used simultaneously per logical host qsub --venum-lhost If The total number of VEs is specified with qsub --venode , the default number of incorporated VE nodes in the queue is used.

- **Number of CPUs**
It will be a value from 0 (physical number of CPUs) to 1 (about 1 CPU) according to the number of CPUs declared by a user.

Normalization Formula $1 - (\text{Declared number of CPUs} / \text{Physical number of CPUs})$
--

- **Elapsed Time**
It will be a value from 0 (unlimited) to 1 (about 1 second) according to the elapsed time declared by a user.

Normalization Formula $0.5 ^ { (\text{Elapsed Time} / \text{Half-life decay time})}$

- **Memory (optional)**
It will be a value from 0 (maximum size of memory) to 1 (about 1 byte) according to the memory size declared by a user.

Normalization Formula

$$1 - (\text{Declared size of memory} / \text{Maximum size of memory})$$

- Number of Jobs

It will be a value from 0 (number of jobs = standard number of jobs) to 1 (number of jobs = 1) according to the number of jobs declared by a user.

Normalization Formula

$$1 - (\text{Declared number of jobs} / \text{Standard number of jobs})$$

- Number of VEs

It will be a value from 0 (physical number of VEs) to 1 (about 1 VE) according to the number of VEs declared by a user.

Normalization Formula

$$1 - (\text{Declared number of VEs} / \text{Physical number of VEs})$$

Wait time for execution from being submitted to a queue

The wait time for execution per half-life decay time will be the value of 1.

$$\text{Wait time for execution from being submitted} / \text{Half-life decay time}$$

Wait time for execution from becoming assignable

The wait time for execution per half-life decay time will be the value of 1.

$$\text{Wait time for execution from becoming assignable} / \text{Half-life decay time}$$

The factor of wait time for execution from becoming assignable is reset by binding JobManipulator and queue, restarting JobManipulator and batch server.

If you have set a high weighting for the wait time for execution from becoming assignable, you may want to adjust the priority of the other factors for requests that you want to give preference to even after binding JobManipulator and queue, restarting JobManipulator and batch server.

Wait time for restart from SUSPENDED

The wait time for restart per half-life decay time will be the value of 1.

$$\text{Wait time for restart} / \text{Half-life decay time}$$

Base-up for a request suspended by urgent request

Set the base-up value of the scheduling priority for requests forced to be SUSPENDED status because a special request was submitted. This base-up value is set to all applicable requests equally. The value is able to be set for each scheduler.

Base-up for a rescheduled request

Set the base-up value of the scheduling priority for requests rescheduled in execution or requests which cannot be started on schedule. This base-up value is set to all applicable requests equally. The value is able to be set for each scheduler.

Base-up defined by user

Set this base-up value in the case the manager wants to change the scheduling priority. This base-up value can be dynamically set to each request by the **smgr(1M)** command.

3.1.4.3 Calculation Timing of the Scheduling Priority

The timing to calculate the scheduling priority is described below.

- When a request is submitted
- When a request attribute is changed by the **qalter** command.

The scheduling priority including waiting time is recalculated at the timing of picking up a request.

3.1.4.4 Processes Using the Scheduling Priority

The scheduling priority is used to pick up a request in the following processing.

- Assignment of the execution host
- Escalation
- Control of overtaking

3.1.4.5 Subcommands for Weight Coefficients

Set the value of weight coefficient to each item of scheduling priority items by using set subcommand of **smgr(1M)**. The subcommands for each item are described below. The operator privilege is required to specify.

Item	smgr(1M) subcommand
Weight Coefficient	
weight coefficient for request priority	set priority weight_request_priority
weight coefficient for number of CPU	set priority weight_cpu_number
weight coefficient for elapse time	set priority weight_elapse_time
weight coefficient for memory size	set priority weight_memory_size

weight coefficient for job number	set priority weight_job_number
weight coefficient for number of VE	set priority weight_ve_number
weight coefficient for wait time for execution from being submitted	set priority weight_run_wait_time
weight coefficient for wait time for execution from becoming assignable	set priority weight_assign_wait_time
weight coefficient for wait time for restarting	set priority weight_restart_wait_time
weight coefficient for user share value	set priority weight_user_share
weight coefficient for user rank	set priority weight_user_rank
Base-Up	
base-up for a request suspended by urgent request	set priority baseup_interrupted
base-up for a rescheduled request	set priority baseup_reschedule
base-up for defined by user (Specifies for each request)	set request baseup_user_definition
Weight Coefficient for Usage data	
weight coefficient for usage data of request priority	set priority pastusage_weight_request_priority
weight coefficient for usage data of number of CPU	set priority pastusage_weight_cpu_number
weight coefficient for usage data of elapse time	set priority pastusage_weight_elapse_time
weight coefficient for usage data of memory size	set priority pastusage_weight_memory_size
weight coefficient for usage data of number of VE	set priority pastusage_weight_ve_number

The following is an example setting to set weight coefficient for request priority to 1 at assignment.

```
# smgr -Pm
Smgr: set priority weight_request_priority = 1 processing_pattern
= assign
```

3.1.5 Algorithm for Picking up Request

When multiple queues and multiple requests exist, the request to be scheduled is picked up according to the following policies.

1. Queue type is higher when request is submitted. (in the order of urgent, special and normal queue)
2. Queue priority is higher when the request is submitted
3. Scheduling priority is higher
4. The time submitted to a queue is earlier
5. In case one request cannot be decided with the conditions above, the scheduler picks up one of rest requests

Thus, the order of priority of the request is decided, and the request with higher priority will be processed as a scheduling object.

[Attention]

As a special case, when the map is full of urgent or special requests and the next urgent or special request cannot be assigned, a request submitted to a queue with lower priority will be scheduled. In such a case, execution of the request may be stopped by another urgent or special request even if assigned on the map once.

[Example] In case of submitting the following jobs, the request of "queue type: Special / queue priority: 100" will be assigned first to the resource effectively.

- queue type: Special/ queue priority:100
- queue type: Normal/ queue priority:100

3.1.6 Algorithm for Starting Request

JobManipulator assigns job servers and set execution start time to the request selected by the "[3.1.4 Algorithm for Picking Up Request](#)".

In case [job condition](#) is specified to a request, job servers applied to the job condition will be the target of the job assignment. In case no job condition is specified, all of the job servers bound to the execution queue to which a request was submitted will be the target of the job assignment.

In a job condition, a condition sentence is specified to "condition" and a target job number that job condition is applied is specified to "job_number". Refer to NQSV User's Guide [Operation] for details. Assignment method of job servers by the value of the condition are as follows.

condition	assign method of the job of job_number
-----------	--

JSV(Job Server Number)	one of them of a job server of the JSV number specified in "condition"
HW(Hardware)	one of them of a job server of the hardware specified in "condition"
NGRP(Name of Node Group)	one of them of a job server in the node group specified in condition

The declaration items that the user must specify in order to use backfill scheduling are as follows. It is specified by -l option of **qsub(1)** command.

Mandatory option

- Elapsed time (option -l ,sub-option elapstim_req)
- * In case "[4.8 Elapse Unlimited Feature](#)" is set on, it will be selectable option to specify Elapse time.
- The number of CPUs that can be executed simultaneously per job (option -l ,sub-option cpunum_job)
- * To specify cpunum_job is not required when specifying the --exclusive option with the qsub(1) command.
- The number of GPU Limit per job (option -l sub-option gpunum_job) if the request use GPU.
- Requests that use VE nodes must specify the number of VE nodes per logical host (--venum-lhost) or the number of VE nodes (--venode) option.

Requests to which these declaration values are not set (unlimited) will not be target for scheduling. In this case, the error message is output to the log file (Default file name :/var/opt/nec/nqsv/nqs_jmd_<scheduler_id>). Even after submitted with these items unlimited, the request can be target for scheduling by specifying these values by **qalter** command.

[Example] The following is the log message in case of not setting Elapse Time Limit.

```
Judge_assignable : Request cannot be scheduled. (Elapse time
unlimited) <Request-ID>
```

If the number of available CPUs is specified to "cpunum_job" or --exclusive option is set, the execution hosts also can be assigned by host.

Also, the user must specify by option the declaration item below by option in case of performing the scheduling using memory size.

Selectable option

- Memory size per job (option -l ,suboption memsz_job)

Requests to which these declaration values are not set (unlimited) will not be target for scheduling though the scheduling uses memory size. And also, in case of performing the scheduling with using memory size, it is necessary to set the limit of memory usage (memsz_limit_ratio) on the execution host by using **smgr(1M)** .

The priority items in choosing the execution host for job assignment are as shown as below and the space that satisfies all of them is selected. When there is no space to assign a job (the scheduled start time is out of the range of the map), it will be suspended until the next assign processing of assignment.

1. The resources for calculation can be reserved (Elapsed time, CPU and GPU(when number of GPU is specified))
2. Memory can be reserved (Optional)
3. The scheduled start time is the earliest

In backfill scheduling, utilization of node resource is considered as highest priority at assignment of jobs. So the order of execution of jobs will not always match to the assigned order.

3.1.7 Elapse Margin

Elapse Margin is a function to give margin to a request until following request is executed by adding a margin time to its elapsed-time limit value. When Elapse Margin is set, the resource occupation time in the scheduler map is decided based on the sum of the elapsed-time limit and the margin time. When it is not set, the resource occupation time is decided based on its elapsed-time limit value.

When Elapse Margin is not set, the elapsed-time limit of a request is the resource occupation time in the scheduler map. However, the time taken in following states is not counted up to the elapsed time limit of the request.

- PRE-RUNNING
- POST-RUNNING

Therefore, if the sum of the time taken in above states and the elapsed time of the request exceeds the elapsed-time limit, the request will be executed with exceeding its resource occupation time and then overlaps with the resource occupation time of following request. If the requests take long time in above states in your operation, it is recommended to set Elapse Margin, so that the execution of a request does not overlap with other request.

3.1.7.1 Setting Elapse Margin

Elapse Margin is set by queue. If the sizes of requests are different for each queue in you site, Elapse Margin can be set corresponding to the size of the request.

Setting method

Elapse Margin can be set by **set queue elapse_margin** a subcommand of **smgr(1M)**.

```
#smgr -P m
Smgr: set queue elapse_margin = elapse_margin queue-name
```

- The initial value of elapse margin is 0.
- The value of Elapse Margin can be set with *elapse_margin* to a queue specified with *queue-name*.
- The value of Elapse Margin is set in seconds. And values in 0 to 2147483647 can be specified.
- When the value of Elapse Margin is changed, requests other than running ones will be reassigned.
- Operator privileges is needed.

*In following cases, an error will occur and Elapse Margin will not be set or changed.

1. When specified Elapse Margin value beyond the range of value that can be specified.
Error message: Elapse margin value out of bounds.
2. When a queue which is not managed by JobManipulator is specified.
Error message: No such queue. (name: <queue-name>)
3. When the user execute the command has no operator privileges or higher ones.
Error message: Not permitted to modify attribute.

The time taken in PRE-RUNNING/POST-RUNNING of requests should be taken into consideration when setting Elapse Margin. The time is depended on the operational environment such as the system performance, user EXIT script set to the queue and so on. Note following facts when setting Elapse Margin.

- If a too large value of Elapse Margin is set, the number of requests which can be assigned to scheduler map will reduce.
 - If a too small value of Elapse Margin is set, the resource occupation time of requests cannot be guaranteed.
-

3.1.7.2 Display Elapse Margin

A. Elapse Margin set to a queue

The value of Elapse Margin set to a queue can be displayed by **-Q,-f** option of **sstat(1)**.

```
#sstat -Q -f
Queue Name: jmq0
Queue Type           = Normal
```

```

Schedule Time      = DEFAULT
Run Limit          = UNLIMITED
User Run Limit     = UNLIMITED
User Assign Limit  = UNLIMITED
Elapse Margin      = 600S <== #Elapse Margin
...

```

B. Elapse Margin of a request

The value of Elapse Margin of a request can be displayed by **-f** option of **sstat(1)**

The value of Elapse Margin of a request is the value of Elapse Margin set to the queue to which the request is submitted.

Planned End Time and Elapse Time include the value of Elapse Margin.

```

Request ID: 5208.batch_serverhost
Request Name = test_jm
User Name = nqs_user
User ID = 2019
Group ID = 500
Current State = Running
Previous State = Pre-running
State Transition Time = 2008-05-01 16:17:55
State Transition Reason = PRERUN_SUCCESS
Queue = testq
Reservation ID = -1
Scheduling Priority (Assign) = 0.998855
User Share = 0.000000
User Rank = 0.000000
Request Priority = 0.000000
CPU Number = 0.000000
VE Number = 0.000000
Elapse Time = 0.998855
Memory Size = 0.000000
Job Number = 0.000000
Run Wait Time = 0.000000
Restart Wait Time = 0.000000
Baseup Interrupted = 0.000000
Baseup Reschedule = 0.000000
Baseup User Definition = 0.000000
PastUsage Request Priority = 0.000000
PastUsage CPU Number = 0.000000
PastUsage Elapse Time = 0.000000
PastUsage Memory Size = 0.000000

```

```

Scheduling Priority (Escalation) = 0.500244
  User Share                      = 0.000000
  User Rank                      = 0.000000
  Request Priority                = 0.500244
  CPU Number                    = 0.000000
  VE Number                     = 0.000000
  Elapse Time                   = 0.000000
  Memory Size                   = 0.000000
  Job Number                    = 0.000000
  Run Wait Time                 = 0.000000
  Restart Wait Time             = 0.000000
  Baseup Interrupted            = 0.000000
  Baseup Reschedule             = 0.000000
  Baseup User Definition        = 0.000000
  PastUsage Request Priority     = 0.000000
  PastUsage CPU Number          = 0.000000
  PastUsage VE Number           = 0.000000
  PastUsage Elapse Time         = 0.000000
  PastUsage Memory Size         = 0.000000
Planned Start Time = (Already Running...)
Planned End Time   = 2008-05-01 16:44:35 <== #Planned End
Time with Elapse Margin included
  Elapse Margin      = 600S <== #Elapse Margin
Job Server a Job belongs to (Job No.:JSV No.):
  0:500
Resources Limits:
  Elapse Time = 1600S <== #The sum of the Elapse Margin and
the elapsed-time limit value
  CPU Number = 8
  Memory Size = 256MB

```

3.1.8 Assign Policy

3.1.8.1 Priority of Assignment Policy

As a normal assignment policy, the following policies are applied in following order to select the nodes for assigning a request. The priority of some of these policies can be adjusted by setting the priority. (Refer to 2.7.8.3 Setting of Priority or Disablement of Assignment Policy for details)

1. The node on which a request can be assigned earliest.
2. The node with the highest JSV Assign Priority.
3. Preferential assignment policy of the node without staging job whose scheduled start time is canceled. (When 'high' is set as the priority.)

(Refer to [3.1.7.3 Preferential Assignment Policy of the Node without any Staging Job](#))

4. The assignment which is considered about network topology. (When 'high' is set as the priority.)
(Refer to [3.1.7.2 The assignment which considered a network topology](#))
5. Topology performance scheduling (if the execution host is SX-Aurora TSUBASA system) (Refer to 5.5 Assignment with topology performance scheduling)
6. CPU Number Concentrated Assignment or Resource Balanced Assignment
7. Assignment looking at ahead and behind
8. Preferential assignment policy of the node without staging job whose scheduled start time is canceled. (When 'low' is set as the priority.)
9. Preferential assignment policy of the node with the fewest queues bound.
10. The assignment which is considered about network topology. (When 'low' is set as the priority.)

As an interrupting assign policy, in addition to the above order, a request is assigned to the node in consideration of the following.

1. The node with high Dynamic JSV Priority (when the execution host is SX-Aurora TSUBASA).
2. The node which does not have request(s) in the RUNNING state.
3. The node with fewest requests that will be re-scheduled by the interruption.

For details on Dynamic JSV Priority, refer to 5.7 Dynamic JSV Priority in Chapter 5. Functions for SX-Aurora TSUBASA.

The priority of above configurable assignment policies can be set by using the **set assign_policy_priority** subcommand of **smgr(1M)**. These assignment policies also can be disabled. Please refer to 2.7.8.3 Setting of Priority or Disablement of Assignment Policy for details.

3.1.8.2 The assignment which considered a network topology

In case of assigning a node for a request that performs communication between multiple nodes at the system configuration with which more than one node are connected by the network switch(NW-SW) of the multistep, the request is assigned to a group of nodes that are connected with same NW-SW (network switch) in order to maximize communication speed between nodes. This feature is called "the feature of the assignment which considered a network topology".

In order to use this assignment function in consideration of network topology, it is necessary to group nodes with low communication latency into a node group before starting JobManipulator.

In order to process a node group, mgrs. is used. (Refer to NQSV users guide for details)

The priority of this assignment policy can be set to "network_topology" by using the **set assign_policy_priority** subcommand. It is recommended that you set the priority as 'low' when you emphasize the system utilization.

(1) Usage of Assignment Considering Network Topology

It is necessary to group nodes with low communication latency.

- Node Group Creation
Create a node group. Note that type is "nw_topo".

```
#qmgr -Pm  
Mgr: create node_group = <ngrp_name> type = nw_topo [switch_layer = <layer>]
```

- node_group : any name of node group
 - type : nw_topo (fixed)
 - switch_layer : number of layers of network switch. Up to 2 layers can be scheduled.
- Node Registration to Node Group
Register nodes with low communication latency to a node group.
Note that a node cannot be registered to multiple node groups.
In case of this feature, node group cannot be nested.

```
#qmgr -Pm  
Mgr: edit node_group add job_server_id = <jsvid>-<jsvid> <ngrp_name>  
Mgr: edit node_group add job_server_id = (<jsvid>,<jsvid>,...) <ngrp_name>
```

The image of node grouping is as follows:

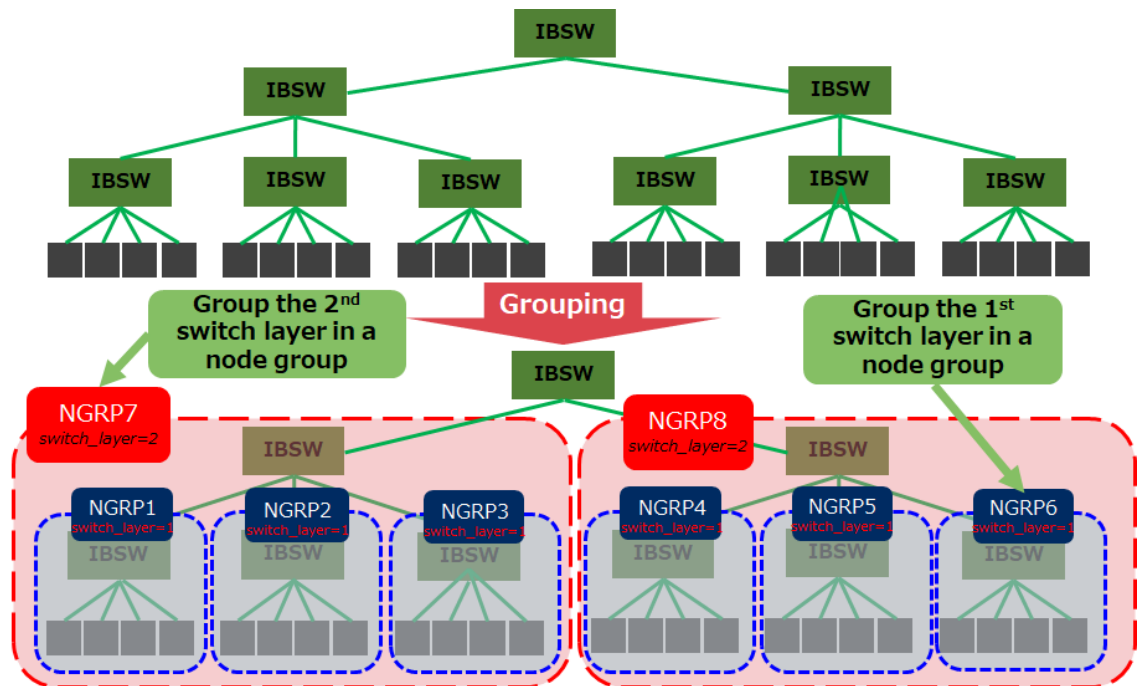


Figure 3-4 The image of network topology node group definition

(2) Stoppage of Assignment Considering Network Topology

In order to stop assignment considering network topology, it is necessary to set "disable" to "network_topology" by using the **set assign_policy_priority** subcommand of **smgr(1M)** or delete node group (with nw_topo type) created at the about (1) step for network topology.

```
#qmgr -Pm
Mgr: delete node_group = ngrp_name
```

3.1.8.3 Preferential Assignment Policy of the Node without any Staging Job

When the staging of files isn't finished in time, the scheduled start time of the request will be canceled and it will be re-assigned after the staging is finished. When assigning another request, the node without such staging job will be selected preferentially, so that the node with such staging job can be left to the request whose scheduled start time has been canceled.

In the operation of staging and emphasizing the TAT of the request whose scheduled start time has been canceled, it is recommended that you set the priority of this assignment policy as 'high'.

3.1.9 Suspended Request

- When a request is suspended by the **qsig** command
JobManipulator does not operate on the request in particular.
The resource gotten by the suspended request is held. In this case, the elapsed time progresses and the request will be terminated by the batch server when reaching the declared elapsed time. It therefore has no influence on the scheduling of the follow-on requests. This operation is constant regardless of the privileges that executed the **qsig** command. It can be confirmed by **-f** option of **sstat(1)** whether the request is suspended by the **qsig** command. If yes, **SIGSTOP** is displayed in **Suspend Reason** field.
- When a request is suspended by **smgr(1M)**
The memory is kept to be used but it is assumed that all resources held by the suspended request are released and the elapsed time of the request stops once and the request will not be reassigned. Only the user with manager privileges or operator privileges can suspend request by **smgr(1M)**. It is performed by the **suspend request** subcommand.

Whether the request is suspended by **smgr(1M)** can be confirmed by **-f** option of **sstat(1)**. If yes, **SMGR_SUSPEND** is displayed in **Suspend Reason** field. A resumption request for this request can be sent by the **resume request** subcommand of **smgr(1M)**, and then it is assigned based on the result of subtracting the executed period from required elapsed time and the request is resumed by the scheduler when reaching the rescheduled start time. Whether a resumption request has been sent for the request suspended by **smgr(1M)** can be confirmed by **-f** option of **sstat(1)**. If yes, **SMGR_RESUME** is displayed in **Suspend Reason** field.

- When a request is suspended by the scheduler due to interruption of an urgent or special request
The memory is kept to be used but it is assumed that all resources held by the suspended request are released. The elapsed time of the request stops once and the request is assigned based on the result of subtracting the executed period from all elapsed time. Reaching the rescheduled start time, the request will be resumed by the scheduler.

Whether the request is suspended by the scheduler due to interruption can be confirmed by **-f** option of **sstat(1)**. If yes, **INTERRUPT** is displayed in **Suspend Reason** field.

If the execution host is an SX-Aurora TSUBASA system, see also **5.6 Suspend Jobs Using VEs**.

About the request suspended by **smgr(1M)** or the interruption:

- CPU is released, however memory is kept because the process is remained.
Therefore, it should be ensured that enough memory or swap can be gotten even

- if other requests are executed while the request is suspended. If the memory becomes insufficient during executing of other requests, it can lead abort of jobs.
- The manager can resume the suspended request by the **qsig** command. Since the resumed request can be executed immediately, there is a possibility that it competes with other running requests for resources.
-

3.1.10 Job Condition

JobManipulator determines which host (job server) execute a job of a request submitted by users. However it would be necessary to execute a particular job on the specified host (job server) in some cases (based on user request types and site operations policy). In such case, you can specify the job condition to the job.

The job condition is specification of the execution condition such as executing a job on specific host or the job server. JobManipulator schedules based on the job condition.

The job condition is specified in with -B option of qsub(1), qlogin(1) and qrsh(1) commands. Refer to the command reference of qsub(1), qlogin(1) or qrsh(1) of NQSV User's Guide [Reference] for the description of specification of the job condition.

3.1.11 Exit Delay Scheduling

Sometimes a job does not finish due to I/O failure or other reasons, and the request execution time exceeds the resource occupancy time on the scheduler map.

To prevent resource duplication in this situation, this function excludes from scheduling the node allocated to the request whose completion is delayed. It also reschedules the affected requests. After the delayed termination request is completed, the node concerned returns to the scheduling target.

To enable this function, add “EXITDELAY_SCHEDULING” line to the NQSV/JobManipulator’s config file (/etc/opt/nec/nqsv/nqs_jmd.conf). After the settings, restart NQSV/JobManipulator.

EXITDELAY_SCHEDULING: ON

Running requests for which the elapsed time limit was increased to more than the resource occupation time by the qalter(1) command are subject to this function, and subsequent requests may be rescheduled. To prevent rescheduling, execute the qrerun(1) command in conjunction with the qalter(1) command.

3.2 System Information Display

Execute the **sstat**(1) command to see JobManipulator system information.

Each information is displayed by execution of the **sstat**(1) command with the following options.

Information	Option
Batch Request Information	no option
Map Information	-A
Resource Reservation Section Information	-B
Complex Queue Information	-C
Power-saving Schedule Information	-D
Execution Host Information	-E
JSV Assign Priority Information	-J
Information of Scheduling Priority	-M
Queue Information	-Q
Information of Scheduler Server Host	-S

Detailed information can be displayed for the batch request, the scheduler server and the queue. Execute the **sstat**(1) with each option and the **-f** option when more detailed information of them is required.

Chapter 4. Advanced Scheduling Features

4.1 Urgent Request/Special Request

An urgent request is a request that is assigned and executed in preference to a normal request.

There are three levels of queues: urgent, special, and normal, which are set on a per-queue basis. Requests submitted to queues of the urgent type are called urgent requests, requests submitted to queues of the special type are called special requests, and requests submitted to queue of the normal-type are called normal requests.

An urgent request is a request that is submitted to urgent queue. It is assigned and starts execution in preference to special requests and normal requests.

A special request is a request that is submitted to special queue. Special requests are assigned and started in preference to normal requests.

The type of queue can be set by the subcommands of the **smgr** (1M) command as below.

smgr(1M)

set queue type = urgent | special | normal <queue-name>

For urgent/special requests, you can specify an interrupt location. You can choose whether these requests interrupt the execution of a normal request to execute immediately, or to execute an urgent/special request after the normal request executed. You can set it to whole scheduler and each queue units, if there are no settings for each queue, the whole scheduler setting is applied.

If you want to interrupt the execution of a normal request, configure the interrupt position to "current".

If you want to execute an urgent/special request after the normal request has been executed, set the interrupt position to "next_run".

If you select the interrupt position to "current" and the running request at the time of urgent/special request is submitted belongs to the same level, it is assigned to the after of the running request.

You can choose to "suspend" and "rerun" for the way to interrupt the execution of a normal request. This setting is on a scheduler basis. This setting is enabled only when the interrupt position is set to "current".

If you choose "suspend" for a way to interrupt the execution of a normal request, the suspended request will be assigned to the after of the urgent/special request. It is resumed after the urgent/special request ends.

If you choose "rerun" for a way to interrupt the execution of a normal request, the re-run request is rescheduled. On this rescheduling, JobManipulator add the base-up for a rescheduled request to the scheduling priority to the request to schedule as a priority.

In addition, even if the interrupted request is set to re-run prohibited, it will be forced to re-run.

To configure the interrupt position and how to interrupt the running normal requests, use the following subcommand of the **smgr** (1M) command.

smgr(1M)

set interrupt to where: Setting interrupt position (scheduler units)

set queue interrupt to where: Setting interrupt positions (queue units)

set interruption method: How to interrupt the execution of a normal request

If the execution host is an SX-Aurora TSUBASA system, see also **5.6 Suspend Jobs Using VEs**.

Note when the method of interrupting the running request is **suspend**:

If there is a request that suspended by the **suspend request** subcommand of the **smgr**(1M) or the **qsig**(1) command, an urgent/special request cannot be used. If you run an urgent/special request, resource management for suspended requests is not guaranteed.

4.1.1 Block of Assignment by Urgent Request

An urgent request may not be executed immediately due to lack of resources and may be waiting to be executed. In this case, a small free resource may occur ahead of the time when the urgent request is assigned. It is possible to prohibit the assignment of newly submitted lower type requests to such free resources. When this setting is enabled, the entire execution host to which the urgent request waiting to be executed is assigned will become unassignable. The assignment prohibition will be removed when the urgent request starts executing.

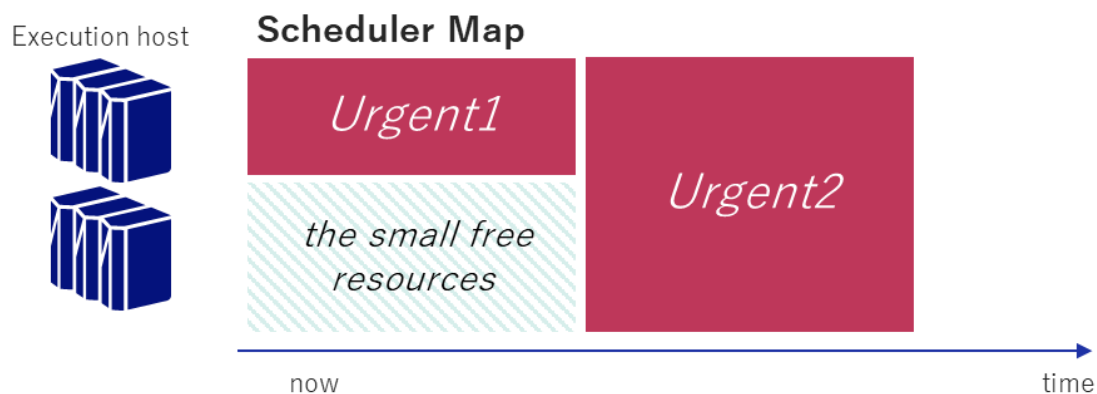


Figure 4-1 A small free resource that occurs before the time the urgent request is assigned

This feature makes it possible to initiate an urgent request as soon as the necessary resources are available for the urgent request waiting to be executed. This is the default behavior, as it gives the highest priority to the execution of urgent requests.

If the amount of urgent requests is large, it may be necessary to allow lower type requests submitted later to be assigned and executed before the urgent requests. In such a case, by setting this function to off, it is possible to allow the assignment and execution of lower type requests that are submitted later to overtake the urgent requests. Use the **set interrupt_assign_block** subcommand of the **smgr** (1M) command below to set the settings.

smgr(1M)

set interrupt_assign_block = *on* | *off*

When the execution host is an SX-Aurora TSUBASA system and the method to suspend the execution of a request is **suspend**, even if this function is set to off, the assignment of a newly submitted lower type request is not allowed until the suspended VE job is resumed. This is a behavior to reserve VE resources for the job to be resumed.

4.2 Interactive Request

The interactive request is a request that is mainly used in debugging and usually required to be executed immediately after it is submitted. By setting a small value to the scheduling interval and scheduler map width, the interactive request is immediately executed and assigned in the submitting order. The standard scheduling interval is two seconds, and the standard scheduler map width is three seconds. The interactive request supports the backfill scheduling function as well as the batch request.

-
- For the scheduler map width, be sure to specify a value that is one or more seconds greater than the scheduling interval.
 - When the interactive request and batch request are scheduled by using different scheduling intervals, they must be manipulated by different JobManipulator instances.
 - The parameters that must be specified to scheduling the interactive request are the same as those of the batch request. For details, refer to ["3.1.5 Algorithm for Starting Request"](#).
-

As well as the batch request, the interactive request is scheduled with the maximum number of usable CPUs and the memory usage limit of the execution host not exceeded.

When multiple queues and multiple interactive requests exist, the request to be scheduled is picked up by the following policies.

1. The priority of the queue to which the interactive request was submitted is higher.
2. The scheduling priority of the interactive request is higher.
3. The time when the interactive request was submitted to the queue is earlier.
4. When the priority could not be determined by the above three policies, any of the requests that are the same in the above policies is selected.

The scheduling priority is determined as follows:

Scheduling priority = User-defined base-up value

If the interactive request cannot be executed immediately, the behavior of the request differs depending on whether *submit_cancel* or *wait* is specified by set interactive_queue real_time_scheduling of **qmgr(1M)**.

- If *submit_cancel* is specified, the interactive request is deleted.
- If *wait* is specified, the interactive request will be executed at the scheduled execution start time if it is assigned. If the request is not assigned, it is scheduled at the specified scheduling interval.

Information of the interactive queue is displayed by using the **-Q** option of **sstat(1)**. When the **-Q -i** option is specified, information of only the interactive queue is displayed. When the **-f** option is also specified, detailed information of the queue is displayed.

```
#sstat -Q -i
[INTERACTIVE QUEUE]
=====
QueueName      RL  URL  UAL  TOT  EXC  QUE  ASG  RUN  EXT  SUD
-----
iq              ULIM  0    0    0    0    1    0    0
```

As well as the batch request, information of the interactive queue is displayed by using **sstat(1)**. By specifying the **-f** option, detailed information can be displayed.

The interactive request supports the basic scheduling function of the batch request, but does not support the urgent and special types and the deadline scheduling.

4.3 Parametric Request

The sub requests of the parametric request are treated and scheduled in the same way as the normal batch request. In the following operations, the subrequests in the parametric request can be displayed and operated by specifying them. In addition, by specifying the parametric request, the subrequests in the specified parametric request can collectively be displayed and operated. For the specification, see the description of each command.

- Displaying the sub requests in the parametric request by **sstat(1)**
- Setting the user-defined base-up value of the scheduling priority by using the **set request baseup_user_definition** subcommand of **smgr(1M)**
- Canceling the rescheduling waiting by using the **stop waiting_retry request** subcommand of **smgr(1M)**
- Suspending the request by the administrator by using the **suspend request** subcommand of **smgr(1M)**
- Resuming the request by the administrator by using the **resume_request** subcommand of **smgr(1M)**

Refer to *NQSV User's Guide [Operation]* for details of the parametric request.

4.4 Workflow

The requests in the workflow are assigned according to the time relationship (*) of the request execution order of the workflow. This request execution order of the workflow is also applied to rescheduling, escalation, and early execution of the requests.

* There are the following two types of the time relationship of the request execution order.

- Sequential execution
This is the relationship of the request which is specified by the **--after** option of the **qsub(1)** command. The preceding request is executed, and then the following requests are executed in order. To maintain this relationship, assign the requests in the execution order.
- Concurrent execution
This is the relationship of the request which is specified by the **--parallel** option of the **qsub(1)** command. Multiple requests are executed concurrently. (These requests are called *concurrent requests*.) To maintain this relationship, assign the concurrent requests to the same time so that these requests can be executed at the same time.

If the requests within the workflow are rescheduled, the requests within the concurrent request and the subsequent requests of the relevant request are also rescheduled.

The priority of the (assignment and escalation) scheduling of the requests within the workflow is the same as that of the normal batch request, with the following exceptions:

Even if the scheduling priority of the subsequent request is higher than that of the preceding request, the preceding request is scheduled, and then the subsequent request is scheduled immediately after the preceding request is assigned.

The scheduling priority of the concurrent requests is treated as the highest among the requests within the concurrent requests.

-
- Because the subsequent request refers to the execution result file of the preceding request, the files must be linked between the preceding and subsequent requests by using a shared file system.
 - It may take a certain amount of time to stage out the execution result file of the preceding request from a local disk to a shared file system, if it isn't written to the shared file system directly. Therefore, if the subsequent request is assigned right after the preceding request, the subsequent request may not refer to the execution result of the preceding request at the execution of the subsequent request starts. It is recommended to specify the stage-out wait time (the subsequent request is assigned after the scheduled execution end time of the preceding request at this interval) to ensure that the subsequent request is executed at the scheduled execution start time. To specify the stage-out wait time, use the **set queue wait_stageout** subcommand of **smgr(1M)**.
 - Because the concurrent requests have the same scheduling priority, they must be submitted in the same type of queue. If they are submitted in different type queues, they are not to be scheduled.
 - When a parametric request is specified as the preceding request, the following request is assigned after finish of the all subrequests. When subrequests are specified as the preceding request, the following request is assigned after assignment of the subrequests.
 - If a hybrid request (a request which requires a different resource for each job by using the **--job-separator** or **---** option for **qsub(1)** command) is submitted to execute concurrently, the request is not scheduled.
-

4.5 Execution Time Reservation

4.5.1 Specify the Execution Start Time

It is possible to start execution of request at the user's specified time by specifying the request execution start time using the **-s** option of **qsub(1)**. (Time Specification)
However the requests reserved by time specification will be controlled as follows in order to be executed at the specified time without fail.

- The request will not be escalated even if the forward escalation is possible.

The requests reserved by time specification can be interrupted by a request submitted to the queue of higher queue type.

- The normal request can be interrupted by the urgent or the special requests.
- The special request can be interrupted by the urgent requests.

4.5.2 Action for Failing in Time Specification

It is possible to select from the following actions in case of failing to assign at the specified time, although a request was submitted by time specification. This setting can be selected by using the **set treat_unbookable_request** subcommand of **smgr(1M)**.

- The request is deleted with a message notifying that time reservation was not successful.
- The request is assigned at the nearest time of the specified time.

4.6 Advance Reservation (Resource Reservation Section)

You can reserve resources for a specified section in advance.

There are two types of advance reservation functions: reserved sections for maintenance and reserved sections for job execution.

Type	Description
For maintenance	Used for maintenance of hardware, etc. Requests are not executed in the reserved section for maintenance. It is created by specifying the execution host name or node group. This reservation section is also called an execution host-specified reservation section.
For job execution	Used to ensure that user requests are executed. It is created by specifying an execution queue. This reservation section is also called an execution-queue-specified reservation section. A template-specified reservation section can also be created.

4.6.1 Create the Reserved Section

The amount of resource demanded and section are specified to make a Resource Reservation Section. An ID (from 0 to 9999) is assigned for it when making it. This ID

is used for the job submission to it and for deleting it.

It can be reserved only for the attached execution host and also outside of the scheduler map.

The Resource Reservation Section can be created by using the **create resource_reservation** subcommand of **smgr(1M)** and specifying following conditions.

- Start time of the Resource Reservation Section (mandatory)
- The period of the Resource Reservation Section (mandatory)
- The execution queue or the execution host. (It is necessary to specify either of them.)
- The number of the execution host (-optional condition when the execution queue is specified-)
- The number of CPU per execution host (-optional condition when the execution queue is specified-)
- Group name (-optional condition when the execution queue and the number of execution host is specified-)
This is to specify the group which can use the reservation. If group isn't specified, all users can use this reservation.

NQSV operator privileges or higher is required in order to demand the reservation.

-
- If you specify the execution host (hostname), the section is reserved for maintenance and the job cannot be executed. To execute a job, specify the execution queue(queue).
 - If a user/group does not have access permit to the execution queue, a reservation section cannot be created. For the detail of access limit of queue, refer to *NQSV User's Guide [Management]*.
-

Reservation policy

The Resource Reservation Section can be created except the following place.

- The time period when the job is already assigned on the execution host
- Time period when the reserved section has already been set on the execution host
- When a power-saving plan has already been set on the execution host

The reservation section with a queue specified can be created in the host and the section in which the request can be executed.

There are two types of reservation with a queue specified.

- Created with the number of execution hosts specified.
- Created without the number of execution host specified (reserve all execution hosts bound to the queue).

If no resource are available in the section to be reserved, an error occurs and the reservation section cannot be created.

If you create the reservation without specifying the number of execution hosts, the execution host that was added to the execution queue after the reserved section is created will also be reserved.

If the accounting function for the resource reservation section is set to ON in the batch server, it is necessary to specify the number of execution hosts when creating the reservation section.

Furthermore, if budget control is set to ON in the accounting server and the billing rate is set for the queue, it is necessary to specify the group name in addition to the number of execution hosts when creating the reserved section.

If the required specification is omitted, an error will occur and the reserved section cannot be created.

After creating reserved section, reservation section will be disabled if the reserved hosts will be removed from operation due to a failure or unbind. The remaining execution hosts are still available for the reservation.

Request can be executed in the reserved section specified by the execution queue. Please configure the duration of the reservation section by determining to the elapse margin and the stage-out wait time.

4.6.2 Job Submission to Reserved Section

By specifying the Reservation ID for the `-y` option of the **qsub(1)** command, the request is executed by using the resources that reserved by specifying the queue. You must have access permission to the queue to submit the job to in the reserved section. Even if the start time of the reserved section has passed, you can still submit and execute a new request as long as the reserved section ends. Multiple requests can be executed in the reserved section as long as the reserved resources are free. You can also submit a request by specifying the start time with `-s` option of the **qsub(1)** command.

For template-specified requests, it is possible to submit to the reserved section by specifying the execution queue in the case of submitting request with container templates. In the case of submitting request with OpenStack templates, it cannot be submitted. When provisioning VE jobs using Docker, use a queued specified reservation instead of a template specified reservation.

In order to execute a request in the reserved section, you must specify the elapsed time (`-l elapstim_req` option) even if you are using the Elapse unlimited feature.

The request is not checked whether it can be executed in the reserved section when **qsub(1)** command is executed. After submitting the request, JobManipulator determines whether it can be assigned to the reserved section. JobManipulator determines the following points: resources of execution hosts, whether the queue specified the reserved section matches the queue which the request is submitted, and checks the time of `-s` option specification for the **qsub(1)** command and the time related to the workflow.

You can check whether a request has been assigned to the reserved section by using the **-f** and **-B -f** options on the **sstat(1)** command. If it is assigned, it will be marked as Assigned, and if not, it will be marked as Queued.

4.6.3 Deleting the Reserved Section

There are two ways to delete the reserved section, one is deleting by the **smgr** (1M) command and the other is automatic deletion by the end of the request in the reserved section.

4.6.3.1 Delete by a smgr command

The reserved section can be deleted by specifying the Reservation ID in the **delete resource_reservation** subcommand of the **smgr**(1M) command.

If there is a request submitted with the **-y** option of **qsub** (1), the reserved section is not deleted. However, if you specify the **force** option to the **delete resource_reservation** subcommand, delete all requests that exist in the reserved section, and then delete the reserved section. JobManipulator will send an email to the owner of the deleted request to inform them that the request has been deleted.

To delete a reserved section, it requires the operator privilege of the NQSV privileges. However, to delete the reserved section where the job exists, administrator privilege is required.

4.6.3.2 Automatic deletion by end of request in the resource reservation

By default, the reserved section is still valid even if there are no requests submitted with the **-y** option of **qsub** (1) for the queued reserved section.

It is possible to delete the reserved section when any request exists in the reserved section after the start time of the reserved section. To enable this option, specify **on** in the **set auto_delete_resource_reservation** subcommand of the **smgr**(1M) command. The default value is **off**.

The reserved section will be deleted at the end time of reserved section. Before deleting the reserved section JobManipulator deletes all requests that exist in the reserved section. JobManipulator will send an email to the owner of the deleted request to inform them that the request has been deleted.

Also, when you DETACH a reserved execution host from the reserved section that is created by specifying the execution host, JobManipulator deletes the reserved section if all the execution host is detached.

4.6.4 Job Assignment to the Resource Reservation Section

The requests submitted to the reserved section is scheduled to the earliest start time that the request can be executed in the reserved section. Even if the submit time is already in the reserved section, JobManipulator assigns it to the time as possible as earliest time from the submit time. If a submitted request cannot be assigned within the reserved section, the request will remain in the Queued status.

4.6.5 Display the Information of the Resource Reservation Section

The information of the Resource Reservation Section is displayed. The information displayed is as follows.

- The Resource Reservation Section ID
- The Resource Reservation Section name (detail display)
- The group name (the Resource Reservation Section with a queue specified)
- The execution queue (the Resource Reservation Section with a queue specified)
- The start time of the Resource Reservation Section
- The period of the Resource Reservation Section
- The demanded number of the execution hosts of the Resource Reservation Section
- The demanded number of CPUs for each execution host of the Resource Reservation Section
- The execution host name in the Resource Reservation Section and its state (detail display)
- Information of Request that use Resource Reservation Section (detail display)

Commands

The information of the Resource Reservation Section is referred to by **sstat**(1) with the **-B** option.

```
# sstat -B
[Queue or Host Resource Reservations]
RES ID Start Time          End Time          NodeNum CPUNum  Queue
-----
27 2007-10-12 13:00:00 2007-10-12 13:20:00 1      0  execque1
```

To display group-specified Resource Reservation Section, specify the **--group** option in conjunction with the **-B** option. Only group-specified Resource Reservation Section information is displayed. The group name is also displayed.

```
# sstat -B --group
[Queue or Host Resource Reservations]
```

RES ID	Start Time	End Time	NodeNum	CPU Num	Queue	GrpName
27	2007-10-12 13:00:00	2007-10-12 13:20:00	1	0	execque1	groupA

The Resource Reservation Section with a queue specified is displayed as follows.

--group specification	Privilege	Scope of Display
specified	User Special user	The Resource Reservation Section of his/her own group
	Group manager	The Resource Reservation Section of his/her managed group
	Operator Manager	The Resource Reservation Section with a group specified
not specified	User Special user	The Resource Reservation Section without a group specified
	Group manager	The Resource Reservation Section of his/her managed group
	Operator Manager	All Resource Reservation Section

Note that the Resource Reservation Section of a queue to which you do not have access permit isn't displayed.

The Resource Reservation Section for maintenance is displayed to all users except that **--group** is specified with **sstat(1)**.

Also, in case of displaying more detail information, use **sstat(1)** with the **-B,-f** option.

```
# sstat -Bf
Resource Reservation ID: 27
  Resource Reservation Name = (none)
  Group Name = groupA
  Queue Name = execque1
  Reserve Start Time = 2007-10-12 13:00:00
  Reserve End Time  = 2007-10-12 13:20:00
  Execution Host Number = 1
  Reserve CPU Number by Host = ALL_CPU
Reserved Hosts (HOST_NAME : STATUS):
  hostserver : ACTIVE
Requests uses this reservation area:
```


none

4.6.6 Accounting for Resource Reservation Section Specifying Execution Queue

If accounting for Resource Reservation Section of batch server and accounting server is enabled, for the Resource Reservation Section specified by the execution queue and the number of execution host, the budget overrun check is performed at creation of the Resource Reservation Section, and the reservation accounting file is generated and accounting is performed based on it when ending or deleting the Resource Reservation Section.

In this case, specifying "**hostnum**" and "**group**" to "**create resource_reservation**" subcommand of **smgr**(1M) command is needed for creation of Resource Reservation Section.

For details of setting for the accounting for Resource Reservation Section, refer to *NQSV User's Guide [Accounting & Budget Control]*.

4.6.7 Set section for health-check and clean-up

For the operation performing health-check and clean-up respectively before or after the reservation, sections can be respectively set on the front and the back of the reservation, which is created by specifying an execution queue and the number of execution hosts, to enable it. Such sections are called PRE-MARGIN and POST-MARGIN respectively.

[PRE-MARGIN + demanded period + POST-MARGIN] is reserved. The health-check and clean-up are requested via BSV to the side of execution host respectively at the start of PRE-MARGIN and POST-MARGIN, so that the scripts for the health-check and clean-up can be executed, which are prepared in advance.

A job cannot be assigned to the section of PRE-MARGIN or POST-MARGIN.

Setting of PRE-MARGIN and POST-MARGIN

The setting of PRE-MARGIN and POST-MARGIN can be set by queue the **set queue resource_reservation** subcommand of **smgr**(1M).

<pre># smgr -P m Smgr : set queue resource_reservation pre-margin = seconds post- margin = seconds queue_name</pre>

- The unit is second.

- The initial value is 0 (not perform the health-check or clean-up).
- This value cannot be changed to a larger one, if there is a reservation of the queue. If it is changed to a smaller one, it will be applied to existing reservations of the queue.
- Operator privilege is needed.

The setting can be displayed by using `sstat(1)` with the `-Q,-f` option.

```
# sstat -Q -f testq
Execution Queue: testq
  Queue Type           = Normal
  Schedule Time        = DEFAULT
  ...omission...
  Wait_Stageout = 0S
  Min Operation Hosts = 10240
  Reservation Margin = {
    Pre-margin = 0S
    Post-margin = 0S
  }
```

Placing of the script for the health-check and clean-up

The script for the health-check and clean-up can be placed in `/opt/nec/nqsv/sbin/extscr/` of the execution hosts as the name of following.

- Health-check: `/opt/nec/nqsv/sbin/extscr/HealthCheck`
- Clean-up: `/opt/nec/nqsv/sbin/extscr/CleanUp`

A queue name as the first argument and a queue type ("batch" or "interactive") as the second argument are passed to the script when calling it, so that a processing can be defined per queue in it.

-
- When restarting JobManipulator, [PRE-MARGIN + demanded period + POST-MARGIN] is re-allocated for the reservation and it cannot be re-allocated for the reservation that has already started.
 - If there is any execution host failed in health-check before the start of reservation, an alternative one can be reserved and health check is performed for it. However, if it still fails, when the reservation has started, this unavailable reservation will be deleted automatically.
-

4.6.8 Creation Function of the Resource Reservation Section Specifying Template

This function is NOT available for the environment whose execution host is SX-Aurora TSUBASA system.

It is possible to make the reservation section which designated the number of machines which start by a template and a template in the reservation section making function of the execution queue designation. The number of machines is number of a virtual machine (VM), number of a baremetal server or number of a container here. Resource amount which is specified by template name and machine number is reserved at the start time by JobManipulator.

It is possible to make the reservation section which plural virtual machine (VM) or container will start running on the identical execution host in the single reservation section. In this case, execution hosts are reserved according to the assign policy. For details, please refer to [2.3.8 Setting of Assign Policy](#).

The reservation section for executing request of template designation is created by designating template for the reservation section which designated of the execution queue. The ID for the reservation section specifying template is designated and invested by **-y** option of **qsub (1)**. The template designated as a request in this case has to be parallel with a designation template of a reservation section. When those aren't identical, a request isn't scheduled.

-
- In case of provisioning, a health-check and a clean-up are not performed in the reservation area specifying template because OS or container is started and stopped at every execution of requests. Setting of PRE-MARGIN and POST-MARGIN for the queue that is specified at making reservation are ignored. For health check and cleanup at every execution of request health-check and clean-up procedure can be set in userexit script at PRERUNNING and POSTRUNNING for the virtual machine (VM) or container. For baremetal server they can be set in start script and stop script In this case elapse margin for virtual machine (VM) and container or timeout for booting and timeout for stopping for the baremetal server must be set by an appropriate time.
 - A reservation section specified by a template that specifies a container template in which VEs is defined cannot be created.
 - If the number of VEs of the container template specified at the time of creating the reservation section is changed to 1 or more by **qmgr (1M)**, the reservation section created with the template will be deleted. When submitting a job to the reservation section with the changed template, use the reservation section specifying a queue.
-

4.6.8.1 Creation of the Resource Reservation Section Specifying Template

A reservation section of template designation designates and makes the number of machines which start by the opening time, a period, a queue name, a template name and a template by the **create resource_reservation** sub-command of the **smgr(1M)** command.

```
#smgr -P m
```

```
Smgr: create resource_reservation starttime = <start_time> blocktime =  
<block_time> queue = <queue_name> template = <template_name> machinum =  
<machine_num> [name = <resource_reservation_name>] [group = <group_name>]
```

- Template name is specified to *template_name*.
- The number of machines that is started with template *template_name* is specified to *machine_num* with its value 1 to 10240.
- Specifying of *template_name* and *machine_num*, specifying of *hostnum* and specifying of *cpunum* cannot be set at the same time,
- When specify *group_name*, only the user who belongs to a specified group can use a reservation section.
- Operator privilege is needed.

When there are no spaces of a resource in the section I try to reserve, when making a reservation, it'll be an error.

4.6.8.2 Display the Resource Reservation Section Specifying Template

Summary information on a reservation section specifying template is displayed by using **sstat** (1) command with **-B** option.

```
$sstat -B
```

```
[Template Resource Reservations]
```

RES	ID	Start Time	End Time	Template	MacNum	Queue
2	2016-03-30 18:00:00	2016-03-30 19:00:00	ostmp1	6	tmp_que	

When specify **--group** with **-B** option, only reservation section information on group designation is displayed.

```
$sstat -B --group
```

```
[Template Resource Reservations]
```

RES	ID	Start Time	End Time	Template	MacNum	Queue	GrpName
3	2016-03-30 18:00:00	2016-03-30 19:00:00	ostmp2	2	tmp_que	group1	

All reservation section information is displayed by **-B** option. Only reservation section information specifying template is displayed which **-B** and **--template** are specified.

Detailed information of a reservation section specifying template is displayed when **-B -f** option are specified.

```

$ssstat -B -f
Resource Reservation ID: 1
:
Reserve End Time    = 2016-07-27 15:00:00
    Reserve Template = vm_1
    Reserve Machine Number = 6
Reserved Machines (HOST_NAME : STATUS):
    192.168.0.1 : ACTIVE
    192.168.0.1 : ACTIVE
    192.168.0.2 : ACTIVE
    192.168.0.2 : ACTIVE
    192.168.0.3 : ACTIVE
    192.168.0.4 : ACTIVE
Requests uses this reservation area:
:

```

4.6.8.3 Job Submission to the Resource Reserved Section Specifying Template

When you submit a request into resource reservation section specifying template, you can specify reservation ID to -y option and template to **--template** option of **qsub(1)** command. In this case the template must be the template which is specified at making reservation.

JobManipulator assigns one machine in the reservation area to one job of a request.

```
$qsub -y< reservation ID> --template=< template>
```

The request put in a reservation area of template designation indicates **-B -f** option of the **sstat** (1) command in addition to the **sstat** (1) command.

```

$ssstat -B -f
Resource Reservation ID: 1
:
Requests uses this reservation area:

```

RequestID	ReqName	UserName	Queue	Pri	STT	PlannedStartTime
1	sleep	user	vmque	500.2443/	0.5002	QUE -:

4.6.8.4 Accounting for Resource Reservation Section Specifying Template

If accounting for Resource Reservation Section specifying template of batch server and accounting server is enabled, for the Resource Reservation Section specified by the template, the budget overrun check is performed at creation of the Resource Reservation Section, and the reservation accounting file is generated and accounting is performed based on it when ending or deleting the Resource Reservation Section.

In this case, specifying "group" to "**create resource_reservation**" subcommand of **smgr**(1M) command is needed for creation of Resource Reservation Section specifying template.

For details of setting for the accounting for Resource Reservation Section, refer to *NQSV User's Guide [Accounting & Budget Control]*.

4.7 ShareDB Merge Feature

We recommend using "ShareDB Merge Feature" in order to conduct Fair-share scheduling on all the calculating clusters. Fair-share scheduling for each calculating cluster was supported.

On the user system which is operating the multiple computing clusters, if JobManipulator is operated on each computing cluster because job operation policy is different between clusters, ShareDB (= the file keeping share and usage data of each user) are kept by each JobManipulator.

ShareDB Merge Feature is the feature which merges the usage data by each user in ShareDB which stored by each JobManipulator and uses this merged data. All the JobManipulators keep the same merged data.

For example, it is possible to use the ShareDB data merged with usage data of a cluster and another cluster for calculating scheduling priority.

4.7.1 Overview of ShareDB Merge Feature

After collecting the usage data stored by each JobManipulator, these data are merged by each user. These merged usage data will be stored to ShareDB in each JobManipulator instance as the usage data used for calculating priority afterwards. The target usage data which will be merged is all the usage data stored in ShareDB as follows.

- Elapse time
- The number of CPU
- The amount of Memory usage
- Request priority
- The number of VE

For the calculation of these usages, it is possible to specify the merge rate for each JobManipulator. For example, it enables the operation that the usage data of each scheduler can be merged by the rate of "10 to 1" at the time of merging.

[Example] The following shows the differences in case of merging usage data by using two kinds of rates below for each scheduler in one of the system circumstances.

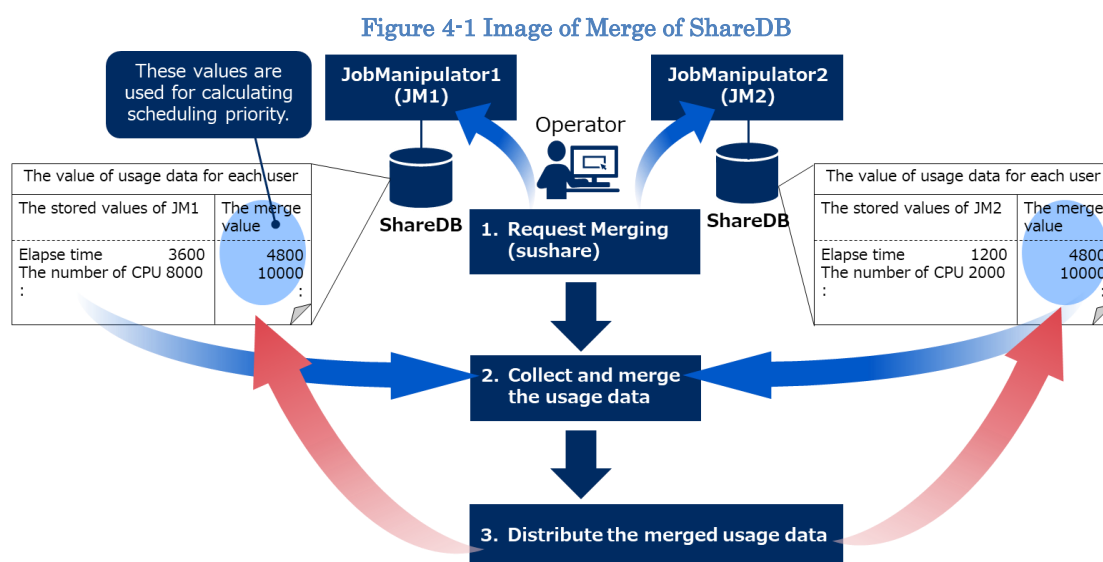
- A. cluster1 :cluster2 = 5 : 1
- B. cluster1 :cluster2 = 2 : 1

If the more resources of cluster1 are used than the one of cluster2, it enables to reflect the larger value to the merged usage data under operation of A than under operation of B.

At the time of merging, it is possible to specify scheduler flexibly in the following operations. (It is also possible to specify scheduler in other operations.)

- In case of operating the multiple schedulers on one host
- In case of operating one scheduler on one host and the multiple schedulers on another host
- In case of operating the multiple scheduler on the multiple hosts

The following picture is an image of ShareDB Merge processing.



* By using the **sushare**(1) command with the **-M** option, it processes (1)-(3) below at one time.

- (1) The operator requests merge processing to the target JobManipulator by using the **sushare**(1) command. After collecting the usage data in ShareDB stored by each JobManipulator instance, these data is merged.
- (2) The merged data is stored to each JobManipulator instance as the merged values and is registered to ShareDB. (These usage values are added up to both to the local value and the merged usage data.)
- (3) When calculating the scheduling priority, each JobManipulator instance use these merged data.

- By using the **sushare**(1) command with the **-M** option, the local usage data of each scheduler is collected and calculated according to configuration file (Refer to "[4.7.4 ShareDB Merge Configuration File](#)" for details.) Then, update the merged data at one time.
- Both data of local and merged are stored to database
`/var/opt/nec/nqsv/nqs_jmd/database /<scheduler_id>/pu_db` on the host managing JobManipulator.
- After merging, the usage data is added up to the local value and the merged value on each cluster.

4.7.2 Set ShareDB Merge Feature

Merge processing of usage data can be set by using the **-M** option of the **sushare**(1) command.

```
# sushare -Pm -M [file name of merge setting] -l [log file name]
```

- In case of specifying log file, specify the log file name just after the **-l** option.
- The log file is stored on the hosts which executes the **sushare**(1) command with the **-M** option.
- By using the **sushare**(1) command with the **-M** option, the data on the ShareDB file is merged by connecting to each JobManipulator instance specified in configuration file through TCP/IP connection.
- It is necessary to install the **sushare**(1) command to the appropriate host and on this host the **sushare**(1) command with the **-M** option can be executed.
- The data on ShareDB file is merged according to the contents specified in the configuration file (default file name: `/etc/opt/nec/nqsv/jm_merge_sharedb.conf`).
- It is necessary to locate the configuration file on the host executing the **sushare**(1) command because the **sushare**(1) command reads this file directly.
- In case of changing the [merge rate](#) in operation, this change will be updated when the executing **sushare**(1) command with the **-M** option next time. (This change is not updated to the merged usage data at the time of changing.)

- In case of rewriting the merged usage data to ShareDB, if the target user is not existent in ShareDB, this process will be ignored. (The new user will not be created.)
- It is necessary to have an operator privilege or higher to set ShareDB merge feature.

Without executing the **sushare(1)** command with the **-M** option, it never execute merge processing. In case of executing merge processing regularly, use "corn".

[Example] The following is an example of executing the **sushare(1)** command. This specifies "test2" as a configuration file name after the **-M** option and "test2.log" as a log file name after the **-l** option. When executing merge process, the following image will be output as standard output.

```
# sushare -P m -M test2 -l test2.log
sushare : 7 records were acquired from host1(sch_id=2) <==This
indicates that 7 records were read from server "host1" and
scheduler number"2".
sushare : 5 records were acquired from host1(sch_id=1)
sushare : 7 records are transmitted to all hosts.
sushare : Completed. <==Completed merge process.
```

The detail of merge process is output to the log file (default file name : **nqs_jmd_sharedb_merge.log**). The following is the output image of the log file. The red letters are explanation.

```
Tue Dec 11 20:44:27 2007 sushare : host1(2), user1[(none)],
CPU=0.000000, VE=0.000000, MEMORY=0.000000,
ELAPSE=0.000000, PRIORITY=0.000000
==>The above indicates that the data of user1(non-account code) was
received from server "host1" and scheduler number "2".
The value is the local value.

Tue Dec 11 20:44:27 2007 sushare : 7 records were acquired from
host1(sch_id=2)
==>The above indicates that 7 records were read from
server"host1" and scheduler number"2".

Tue Dec 11 20:44:27 2007 sushare : user1[(none)],
CPU=6722.278397, VE=5213.331234, MEMORY=6083.999465,
ELAPSE=6722.278397, PRIORITY=6083.999465
==>The above indicates that the data of user1(non-account code)
was merged. The value is the merged value.
```

* In the following cases, an error occurs and merge process is not executed with the **sushare(1)** command.

- The specified file name does not exist in the setting file.
- The host specified in the setting file is not in operation
- The host specified in the setting file is in operation but JobManipulator is not started

The merge process by the **sushare(1)** command can be executed while JobManipulator is running without stopping scheduling. If the system problem occurred and the running merge process was aborted, merge process will be executed including the uncompleted process at the next time of executing the **sushare(1)** command with the **-M** option.

4.7.3 Display the Usage Data of ShareDB

The usage data of ShareDB can be displayed by using following options of the **sushare(1)** command.

- **-S** option (the merged usage value)
- **-L** option (the merged usage value and the local usage value)

The followings are execution examples.

By using **sushare(1)** with the **-S(capital)** option, each usage value of scheduler is displayed as a merged usage value.

[Group Name : TOP_GROUP]								
User	Acctcode	Share	PU_cpunum (%)	PU_venum (%)	PU_memsz (%)	PU_elapstim (%)	PU_reqpri (%)	
*nec	none	0.333	4.190M (50.002)	4.190M (50.002)	3.996M (50.002)	1163:58:00 (50.002)	4.190M (50.002)	
*nqs	none	0.667	4.190M (49.998)	4.190M (49.998)	3.996M (49.998)	1163:53:44 (49.998)	4.190M (49.998)	
[Group Name : nec]								
User	Acctcode	Share	PU_cpunum (%)	PU_venum (%)	PU_memsz (%)	PU_elapstim (%)	PU_reqpri (%)	
necusr1	none	0.167	2.095M (25.001)	2.095M (25.001)	1.998M (25.001)	581:59:01 (25.001)	2.095M (25.001)	
necusr2	none	0.167	2.095M (25.001)	2.095M (25.001)	1.998M (25.001)	581:58:58 (25.001)	2.095M (25.001)	
[Group Name : nqs]								
User	Acctcode	Share	PU_cpunum (%)	PU_venum (%)	PU_memsz (%)	PU_elapstim (%)	PU_reqpri (%)	
nqsusr1	none	0.167	1.048M (12.500)	1.048M (12.500)	1022.954K (12.500)	290:58:24 (12.500)	1.048M (12.500)	
nqsusr2	none	0.167	1.048M (12.500)	1.048M (12.500)	1022.955K (12.500)	290:58:25 (12.500)	1.048M (12.500)	
nqsusr3	none	0.167	1.048M (12.500)	1.048M (12.500)	1022.956K (12.500)	290:58:26 (12.500)	1.048M (12.500)	
nqsusr4	none	0.167	1.048M (12.500)	1.048M (12.500)	1022.956K (12.500)	290:58:26 (12.500)	1.048M (12.500)	

By using **sushare**(1) with the **-L** option, each usage value of scheduler is displayed in a format of "merged usage value/local usage value".

[Group Name : TOP_GROUP]													
User	Acctcode	Share	PU_cpunum (%)		PU_venum (%)		PU_memsz (%)		PU_elapstim (%)		PU_reqpri (%)		
*nec	none	0.333	4.190M/	4.190M (50.002/50.002)	4.190M/	4.190M (50.002/50.002)	3.996M/	3.996M (50.002/ 50.002)	1163:58:19/	1163:58:19 (50.002/50.002)	4.190M/	4.190M (50.002/50.002)	
*nqs	none	0.667	4.190M/	4.190M (49.998/ 49.998)	4.190M/	4.190M (49.998/ 49.998)	3.996M/	3.996M (49.998/ 49.998)	1163:54:03/	1163:54:03 (49.998/ 49.998)	4.190M/	4.190M (49.998/ 49.998)	
[Group Name : nec]													
User	Acctcode	Share	PU_cpunum (%)		PU_venum (%)		PU_memsz (%)		PU_elapstim (%)		PU_reqpri (%)		
necusr1	none	0.167	2.095M/	2.095M (25.001/ 25.001)	2.095M/	2.095M (25.001/ 25.001)	1.998M/	1.998M (25.001/ 25.001)	581:59:10/	581:59:10 (25.001/ 25.001)	2.095M/	2.095M (25.001/ 25.001)	
necusr2	none	0.167	2.095M/	2.095M (25.001/ 25.001)	2.095M/	2.095M (25.001/ 25.001)	1.998M/	1.998M (25.001/ 25.001)	581:59:08/	581:59:08 (25.001/ 25.001)	2.095M/	2.095M (25.001/ 25.001)	
[Group Name : nqs]													
User	Acctcode	Share	PU_cpunum (%)		PU_venum (%)		PU_memsz (%)		PU_elapstim (%)		PU_reqpri (%)		
nqsusr1	none	0.167	1.048M/	1.048M (12.500/ 12.500)	1.048M/	1.048M (12.500/ 12.500)	1022.958K/	1022.958K (12.500/ 12.500)	290:58:29/	290:58:29 (12.500/ 12.500)	1.048M/	1.048M (12.500/ 12.500)	
nqsusr2	none	0.167	1.048M/	1.048M (12.500/ 12.500)	1.048M/	1.048M (12.500/ 12.500)	1022.960K/	1022.960K (12.500/ 12.500)	290:58:30/	290:58:30 (12.500/ 12.500)	1.048M/	1.048M (12.500/ 12.500)	
nqsusr3	none	0.167	1.048M/	1.048M (12.500/ 12.500)	1.048M/	1.048M (12.500/ 12.500)	1022.961K/	1022.961K (12.500/ 12.500)	290:58:31/	290:58:31 (12.500/ 12.500)	1.048M/	1.048M (12.500/ 12.500)	
nqsusr4	none	0.167	1.048M/	1.048M (12.500/ 12.500)	1.048M/	1.048M (12.500/ 12.500)	1022.961K/	1022.961K (12.500/ 12.500)	290:58:31/	290:58:31 (12.500/ 12.500)	1.048M/	1.048M (12.500/ 12.500)	

*The **"-s (small letter) option"** of the **sushare**(1) command can be used to specify a scheduler ID of JobManipulator. (Without specifying with the **-s** option, the default scheduler will be specified.)
(Refer to the **sushare**(1) command with the **-s** option in NQSV User's Guide [Reference] for details.)

4.7.4 ShareDB Merge Configuration File

The merge process is executed according to the configuration file (default file name: **/etc/opt/nec/nqsv/jm_merge_sharedb.conf**). The following contents can be specified in setting file.

- **Comment Line**
The line starting with '#' is comment.
- **HOST Line**
Specify the host name or IP address executing JobManipulator. HOST Line needs to be specified before SCH ID (scheduler ID) Line and Merge Rate Line.
- **SCH_ID Line**
Scheduler ID of JobManipulator
- **MERGE_RATE Line**
Merge Rate. The value which is the local usage value multiples by merge rate is merged.
- **CPU Line**
CPU Merge Rate. The value which is the CPU local usage value multiples by merge rate is merged. This content can be omitted and if it is omitted, MERGE_RATE will be used.

- **ELAPSE Line**
ELAPSE Merge Rate. The value which is the ELAPSE local usage value multiplied by merge rate is merged. This content can be omitted and if it is omitted, MERGE_RATE will be used.
- **MEMORY Line**
MEMORY Merge Rate. The value which is the MEMORY local usage value multiplied by merge rate is merged. This content can be omitted and if it is omitted, MERGE_RATE will be used.
- **PRIORITY Line**
PRIORITY Merge Rate. The value which is the PRIORITY local usage value multiplied by merge rate is merged. This content can be omitted and if it is omitted, MERGE_RATE will be used.
- **VE Line**
VE Merge Rate. The value which is the VE local usage value multiplied by merge rate is merged. This content can be omitted and if it is omitted, MERGE_RATE will be used.

Merge Rate is multiplying rate. Merge Rate for each resource is also multiplying rate.

[Example] The following is a calculating example of CPU usage value with using multiplying rate.

Scheduler : A	Scheduler : B
CPU usage value : 100	CPU usage value : 150
CPU Merge Rate : 2	CPU Merge Rate : 3

In the above case, the calculation will be "100 * 2 + 150 * 3" and the merged value will be 650.

[Example] The following is an example of configuration file that two JobManipulators are targets of merging cluster1 and cluster2.

The first half is the setting for cluster1. The second half is the setting for cluster2. The Merge rate is 10:1.

```
# JobM for cluster1
HOST=hostA
SCH_ID=1
MERGE_RATE=10

# JobM for cluster2
HOST=hostB
SCH_ID=11
```

* It is necessary to store the configuration file on the host executing the **sushare(1)** command because the **sushare(1)** command refer to this file directly.

* The following cases, it leads to an error and merge process is not executed with output the contents of error line and error type.

- In case HOST Line does not exist or HOST Line is specified after SCH_ID (Scheduler ID) Line or Merge Rate Line
Error message: "HOST" is not specified.
- In case SCH_ID Line does not exist
Error message: "SCH_ID" is not specified.
- In case MERGE_RATE is unset and CPU Line does not exist
Error message: "CPU" is not specified.
- In case MERGE_RATE is unset and ELAPSE Line does not exist
Error message: "ELAPSE" is not specified.
- In case MERGE_RATE is unset and PRIORITY Line does not exist
Error message: "PRIORITY" is not specified.
- In case MERGE_RATE is unset and MEMORY Line does not exist
Error message: "MEMORY" is not specified.
- In case MERGE_RATE is unset and VE Line does not exist
Error message: "VE" is not specified.
- In case the value except number is specified to SCH_ID
Error message: Only the numerical value can be specified for "SCH_ID"
- In case the value except number is specified to CPU
Error message: Only the numerical value can be specified for "CPU"
- In case the value except number is specified to ELAPSE
Error message: Only the numerical value can be specified for "ELAPSE"
- In case the value except number is specified to PRIORITY
Error Message: Only the numerical value can be specified for "PRIORITY"
- In case the value except number is specified to MEMORY
Error Message: Only the numerical value can be specified for "MEMORY"
- In case the value except number is specified to MERGE_RATE
Error message: Only the numerical value can be specified for "MERGE_RATE"
- In case the value except number is specified to VE
Error Message: Only the numerical value can be specified for "VE"
- In case invalid line was written
Error message: Unknown key word.
- In case HOST address specified in HOST Line was not transformed
Error message: Unknown host name.
- In case the multiple same hosts are specified
Error message: Host is doubly specified.
- In case SCH_ID Line is doubly specified
Error message: "SCH_ID" is doubly specified.
- In case CPU Line is doubly specified
Error message: "CPU" is doubly specified.
- In case ELAPSE Line is doubly specified
Error message: "ELAPSE" is doubly specified.
- In case PRIORITY Line is doubly specified
Error message: "PRIORITY" is doubly specified.

- In case MEMORY Line is doubly specified.
Error message: "MEMORY" is doubly specified.
- In case VE Line is doubly specified.
Error message: "VE" is doubly specified.

4.8 Elapse Unlimited Feature

Elapse Unlimited Feature enables to schedule requests without specifying the limitation value of elapse time (=Unlimited).

- * In case of specifying elapse time, refer to ["3.1.5 Algorithm for Starting Request"](#).
- * It is necessary to specify the [CPU number of run limit](#) for each job (by using

cpunum_job sub-option, -l option of the **qsub** command).

The following operation policies are set in the scheduling with activating Elapse Unlimited Feature.

Requests whose limit value of elapse time specified are also scheduled.

It is possible to assign requests with specifying elapse time or unlimited just after the request with specifying elapse time.

No request is assigned behind the unlimited request (= the request without specifying elapse time)

If the unlimited request finished running, the resource is released and other requests will be assigned.

4.8.1 Set Elapse Unlimited Feature

To set the Elapse Unlimited Feature (=scheduling the elapse time unlimited request can be set by the **set use_elapstim_unlimited** subcommand of **smgr(1M)**.

```
#smgr -P m
Smgr : set use_elapstim_unlimited = on | off
```

In case "on" is specified, it enables to schedule the requests that elapse time is specified to unlimited.

The initial set value is "off (=with elapse limit)".

It can be set by each scheduler.

It is necessary to have the operator privilege or higher to set "Elapse Unlimited Feature".

In case "on" is specified to the elapse limit, the unlimited request which was already submitted will be start scheduling.

In case "off "is specified to the elapse limit, the unlimited request which was not assigned yet will not be scheduled. The requests already assigned are kept assigned and started to run on planed schedule.

The unlimited request is not assigned to the host where [Advance Reservation \(Resource Reservation Section\)](#) is set because the Resource Reservation Section has higher priority than the Request Unlimited Feature.

4.8.2 Display the Setting of Elapse Unlimited

The set values (on/off) of elapse unlimited can be displayed by using `sstat(1)` with the `-S,-f` option.

```
#sstat -S -f
JobManipulator Server Host: bsv.nec.co.jp
  JobManipulator Version   = R1.00
  JobManipulator Status    = Active
  Scheduler ID              = 5
  Schedule Interval        = 60S
  Schedule Time             = 604800S
  Use Elapse Unlimited     = ON
  :
```

4.9 Scheduling with the change in the number of CPUs/GPUs

In cases of change in the number of available CPUs/GPUs, such as failure and recovery of CPU/GPU, setting change of RSG/RB etc., JobManipulator performs scheduling based on the updated number of available CPUs/GPUs and the requests that have been assigned to the scheduler map will be reassigned. The targets of reassignment are as follows.

- The requests that are assigned to the execution hosts with change in the number of available CPUs/GPUs and are waiting to run.
- The requests assigned behind of the multi-node request which is assigned to the execution hosts with change in the number of available CPUs/GPUs.

The order of reassigning the targets to the scheduler map is from the request with earlier planned start time determined when previous assignment.

This feature depends on Load Interval of NQSV batch server. When the value of Load Interval is set to 0, this feature does not work. Therefore, Load Interval should be set as a value larger than 1 to make this feature work. Load Interval controls the timing of updating available CPUs/GPUs. Consequently, when a large value is set to Load Interval, the interval of updating available CPUs/GPUs is large and it will take a bit of time to do scheduling based on the updated number of available CPUs/GPUs. Refer to *NQSV User's Guide [Management]* for Load Interval.

4.10 Support for Failover System

JobManipulator supports EXPRESSCLUSTER. By the redundant JobManipulator hosts configured with EXPRESSCLUSTER, it is possible to continue scheduling without system down.

By using the **-a** option at JobManipulator (nqs_jmd) starting up, it can specify the virtual IP address supplied by EXPRESSCLUSTER.

If the virtual-IP-address is specified, JobManipulator performs as follows.

- The JobManipulator server hostname displayed by **sstat(1)** is the hostname that corresponds to this IP address.

In case Fail-over occurs, the running requests will continue to run, and the scheduled start time of the requests which has already been assigned and is waiting to be executed is cleared and the requests is rescheduled.

4.11 Scheduling in Problem on Node

When node problem (which means unlink of the job server) occurred on the job server with assigned jobs, the jobs are cleared and rescheduled.

4.11.1 Rescheduling at Node Problem

The followings are request states which exist on the node.

1. Running request
2. Request waiting for execution
3. Request under stage-in

In case of node down due to the failures when these requests exist on the node, the requests are rescheduled as follows.

Refer to "[4.11.2 Forced Rerunning of Running Job](#)" for running requests.

A request waiting for execution will be in QUEUED status after purging its jobs, and rescheduled.

The operations above are valid in not only problems on the node but also in the case of unbinding the job server by the operator. Therefore, rescheduling requests can also work when a node is down for maintenance.

Using "Keep Forward Schedule function", it is possible to hold the number of requests to which the scheduled start time is changed to a minimum, by maintaining the scheduled start time of a request which begins to execute after fixation time from a node failure.

Refer to "[4.11.4 Keep Forward Schedule](#)" for Keep Forward Schedule function.

4.11.2 Forced Rerunning of Running Job

The job will be stalled when node problem occurs on node where a running job exists. This stalled job can be rerun forcibly by setting of scheduler. The running job which stalled will be rescheduled by executing rerun.

The forced rerunning of the running job is set by the **set forced_rescheduling** subcommand of **smgr**(1M). Operator privilege or higher is required to set. The default is OFF and a job is rescheduled after waiting for node recovery.

The state of the request subject to forced rerun is as follows:

- RUNNING
- SUSPENDING
- SUSPENDED
- RESUMING
- POST-RUNNING

4.11.3 Waiting to Forced Rerunning on Connection with BSV

If stalled jobs exist on connection of JobManipulator and batch server, JobManipulator will wait a period specified by **JM_RERUNWAIT** (default is 10 minutes) to force rerunning of the stalled jobs.

If the jobs recover from stalled state during the waiting time, forced rerunning will not be done. If the jobs are still in stalled state after the waiting time and the Forced Rerunning of Running Job function is set to ON, forced rerunning will be done to the jobs.

The waiting time can be specified in configuration file. The setting shown as follows can be added to configuration file (*/etc/opt/nec/nqsv/nqs_jmd.conf*) to customize it.

```
JM_RERUNWAIT: 600 #waiting time for waiting to forced
rerunning on start-up(specified in second)
```

This function only works on connection of JobManipulator and batch server. The jobs detected as stalled jobs during operation after completion of connection of JobManipulator and batch server will be forced to rerun immediately, when Forced Rerunning of Running Job function is set to ON.

4.11.4 Keep Forward Schedule

4.11.4.1 Overview of Keep Forward Schedule

This function enables that the schedule of requests after a time is maintained on node failure to minimize the schedule change. The schedule of requests assigned at earlier time than this time will be canceled and rescheduled. It is useful when you can fix the node failure as soon as possible after it happened and want to maintain the schedule as much as possible. If node failure is not fixed until the scheduled start time of the request, it will be rescheduled.

4.11.4.2 Setting of Keep Forward Schedule

The time can be configured by using the **set keep_forward_schedule** subcommand of **smgr(1M)**.

```
#smgr -P m
Smgr : set keep_forward_schedule = second
```

Set the time to determine to maintain the schedules of which requests when node problem (HW failure or only unlink down of the job server) occurred with specifying a time of period in *second* by second. The schedules of the requests whose scheduled start time is [time of HW failure occurrence + *second*] or a later one are maintained.

When 0 is specified in *second*, the schedule is not maintained. The initial value is 0. Operator privilege is needed.

When the state of the node by which failures occurred does not change to ACTIVE even if this setting time is passed, the requests which are assigned to the node are rescheduled.

4.11.4.3 Display of Setting of Keep Forward Schedule

The setting can be displayed by using **sstat(1)** with the **-S,-f** option.

```
#sstat -S -f
JobManipulator Server Host: bsv.nec.co.jp
  JobManipulator Version    = R1.00
  JobManipulator Status     = Active
  Scheduler ID              = 5
  :
  Keep Forward Schedule    = 3600S
  :
```

4.11.5 Top Priority Execution of the Failure Encounter Request

We do not recommend using this function because it greatly affects the execution of many requests and the utilization of the entire system. When using it, be sure to fully understand the notes before using it.

4.11.5.1 *Overview of Top Priority Execution of the Failure Encounter Request*

When a node failure occurs, you can rerun the running request that encountered the failure, assign it backwards, and re-execute it. However, in an operation where many requests are executed, it may take a long time for the failure encounter request to be executed again. Therefore, by using this function, it is possible to re-schedule and re-execute a running request that has encountered a failure with the highest priority. If you use this function, for a request scheduled to be executed, the scheduled start time will be canceled and the request will not be started even if there is space on the execution host until the failure encounter request is re-executed. We do not recommend using this function because it greatly affects the execution of many requests and the utilization of the entire system.

4.11.5.2 *Setting of Top Priority Execution of the Failure Encounter Request*

To use this function, the following line must be added to the configuration file (/etc/opt/nec/nqsv/nqs_jmd.conf). After adding, it will be enabled by rebooting JobManipulator.

```
HWFAILURE_OVERTAKE: ON
```

If there is no above description or if the value is OFF, this function will be disabled. In addition, to use this function, it is necessary to enable 4.11.2 Forced Rerunning of running jobs function.

4.11.5.3 *Notes when using this function*

We do not strongly recommend using this function. Please note the following when using this function.

-Until the re-execution schedule of the failure encounter request is decided, all the scheduled start times of other requests are canceled, and they will not be started even if there is space on the execution host. Therefore, it greatly affects the utilization of the entire system.

-If a request encounters a failure, all scheduled start times of other requests will be cancelled. Other requests will not be able to be executed at the scheduled start time that has already been determined.

-If a failure occurs and there are insufficient execution hosts or resources for the failure encounter request to be re-executed, all job execution will be stopped unless the failure is recovered.

4.12 Deadline Scheduling

4.12.1 Overview of Deadline Scheduling

JobManipulator assigns requests to the earliest possible time in backfill scheduling, while it assigns requests with deadline time specified to a time as close to the specified deadline time as possible, so that it can finish at the specified deadline time, and other requests can be assigned to the free resource at the head of the scheduler map first. The request with deadline time specified is named Deadline Request.

It is disadvantageous to Deadline Request in scheduling such as escalation because its priority is lower than non-deadline request. Therefore, JobManipulator supports a function to reduce the usage data of Deadline Request, which is used to calculate scheduling priority. It can give incentive to the user of Deadline Request. Deadline scheduling is enabled for Deadline Request which is submitted to the normal queue only, while urgent/special requests are not scheduled as Deadline Request. Deadline Request will be started to run immediately to prevent lowering of utilization of the system, when there is no request waiting to be assigned and there are free resources at current time for execution of the Deadline Request.

4.12.2 Setting of Deadline Scheduling

The setting of deadline scheduling for a queue should be set as **on** to enable deadline scheduling. This can be set by the **set queue deadline mode** subcommand of **smgr(1M)**. If it is set to **off**, the deadline time set to Deadline Request is disregarded and this request will be scheduled in backfill scheduling.

When deadline scheduling ON/OFF is changed during operation, Deadline Request is handled as follows.

- **OFF to ON**

Deadline time is displayed by the **qstat/sstat** command, and Deadline

Scheduling is applied at the next scheduling interval. Although Deadline Request which has already been assigned is not rescheduled at this time, it is rescheduled in deadline scheduling at the next escalation interval.

- **ON to OFF**

"(none)" is displayed in the field of Deadline Time by the **qstat/sstat** command, and Deadline Request which has already been assigned is not rescheduled, while the job of Deadline Request which has been assigned to outside of the scheduler map is deleted and this request is rescheduled as QUEUED request.

4.12.3 Submission of Deadline Request

Deadline Request is submitted by the **qsub** command with specifying a deadline time. The syntax is as below.

```
% qsub -Y deadline_time script

deadline_time : [[ [CC]YY]MM]DD]hhmm[.SS]

      CC: First two digits of year
      YY: Last two digits of year
      MM: Month(01-12)
      DD: Date(01-31)
      hh: Hours(00-23)
      mm: Minutes(00-59)
      SS: Seconds(00-61)

Specify deadline time to deadline_time.
```

Consistency between current time and specified deadline time is not checked at submitting Deadline Request. If a passed time is specified to deadline time, this request is scheduled as non-deadline request.

Deadline time can be confirmed with the **qstat -f** or **sstat -f** command. If deadline time is not specified, the command displays Deadline Time as "**(none)**".

4.12.4 Scheduling of Deadline Request

JobManipulator schedules Deadline Request to finish at deadline time to a maximum extent, when deadline scheduling setting of the queue is set as **on** and Deadline

Request is submitted. Deadline Request is handled from submission to starting execution as follows.

Pick up from assign pool

Deadline Request is picked up from assign pool in the order of scheduling priority like non-deadline requests.

Assigning

JobManipulator assigns Deadline Request to the scheduler map where the planned end time can be closest to the deadline time. The assignment time is decided in following order.

1. The planned end time is as same as the deadline time.
2. The planned end time is before the deadline time and is closest to the deadline time.
3. The planned end time is after the deadline time and is closest to the deadline time.

If there are always no free resources in scheduler map all of the time, Deadline Request cannot be assigned and it will cause Deadline Request always exceeds the deadline time. To avoid such situation, Deadline Request can be assigned to outside of the scheduler map.

Changing Assignment Location

In case Escalation is set to ON (Refer to "2.7.6 Setting of Escalation Feature" for details), Assignment Location of assigned deadline request is checked at every escalation interval whether it can be changed. At that time, nodes other than assigned one can be candidate for assignment location.

1. In case there are free resources (with which deadline request can be executed immediately) at the head of the scheduler map and there is no other assignable request in the assignment pool, escalation is performed for this request and then the request is assigned at the head of the scheduler map and started to run.
2. In case the planned end time of deadline request can be changed closer to the deadline time, the request will be re-assigned.

Running

When reaching the planned start time of Deadline Request, it starts to runs. Once its state becomes RUNNING, Deadline Request is handled as non-deadline request, and is

not scheduled in deadline scheduling while batch jobs exist. However, if it is rerun during running, it can be scheduled in deadline scheduling as deadline request.

- When a Deadline Request is interrupted by an urgent/special request.
The request is handled just like non-deadline request. (Refer to "4.1 Urgent / Special Request" for details)
- When a Deadline Request is hold by qhold command.
The request is returned to the assignment pool like non-deadline request, and assigned after released. After released, the request is not scheduled in deadline scheduling.
- When Deadline Request is suspended by **smgr**.
The request is returned to the assignment pool like non-deadline request, and assigned after resumed by **smgr**. After resumed, the request is not scheduled in deadline scheduling.
- When Deadline Request is suspended by the qsig command.
The request keeps occupying the resources on the scheduler map like non-deadline request.

Deadline Request is scheduled to finish by the deadline time, however, there are some cases that the planned end time exceeds the deadline time due to following reasons.

1. There are no free resources from current time to the deadline time.
 - Resource insufficient at assigning
 - Resource insufficient at rescheduling due to following reasons.
 - Delay of completion of stage-in
 - Execution host failure
 - Interruption by urgent/special request
 - Rerun by the **qrerun** command
 - Released by the **qrls** command
 - Resumed by the **smgr** command
 - Changing of the length of the scheduler map
 - Scheduling with the change in the number of CPUs function
2. The status of the request is unable to be scheduled.
 - The deadline time is exceeded as if the request was assigned at the head of the scheduler map.
 - The execution queue is stopped.
 - Job server is not bound to the execution queue.
 - Deadline Request exceeds the deadline while the scheduling is stopped.
 - Other request cannot be overtaken due to the overtake control.
 - Too many resources are specified to the request, or resources become insufficient due to the execution host down.

If Deadline Request exceeds the deadline time, it is scheduled to finish at the time closest to the deadline time.

4.12.5 Usage Data of Deadline Request

Deadline Request is disadvantageous in scheduling such as escalation because its priority is lower than non-deadline request. Therefore, JobManipulator supports a function to enable the manager to set some conditions to adjust the usage data which is used to calculate scheduling priority by reduce rate of usage data. Usage data is adjusted when the each job of the Deadline Request finishes. A usage data after subtracting the product of the real usage data and reduce rate from the real usage data is updated to the ShareDB. The same reduce rate is applied to the four kinds of usage data (elapsed time, number of CPUs, memory usage, request priority).

Reduce rate of usage data is not uniform. It is proportional to the difference of the deadline time and the planned end time of the requests, which is a time after required elapsed time and the elapse margin time added to the planned start time. Reduce rate can be adjusted as explained below.

The reduce rate when the request finishes just at the deadline time is as base value. If the request finishes before the deadline time, the reduce rate is decreased from the base value, while it is increased from the base value if the request finishes after the deadline time. The parameters for adjusting the reduce rate can be set per queue by the **set queue deadline reduce** subcommand of **smgr(1M)**. Operator privilege or higher is required to set these parameters.

The user with User privilege or higher can confirm the value of the parameters of the queue by the **sstat -Q -f** command.

Reduce rate adjustment parameters

Reduce rate is specified by following seven parameters for adjusting reduce rate. (The string in [] is short name).

[R3] Maximum reduce rate

[R2] Ontime reduce rate

[R1] Minimum reduce rate

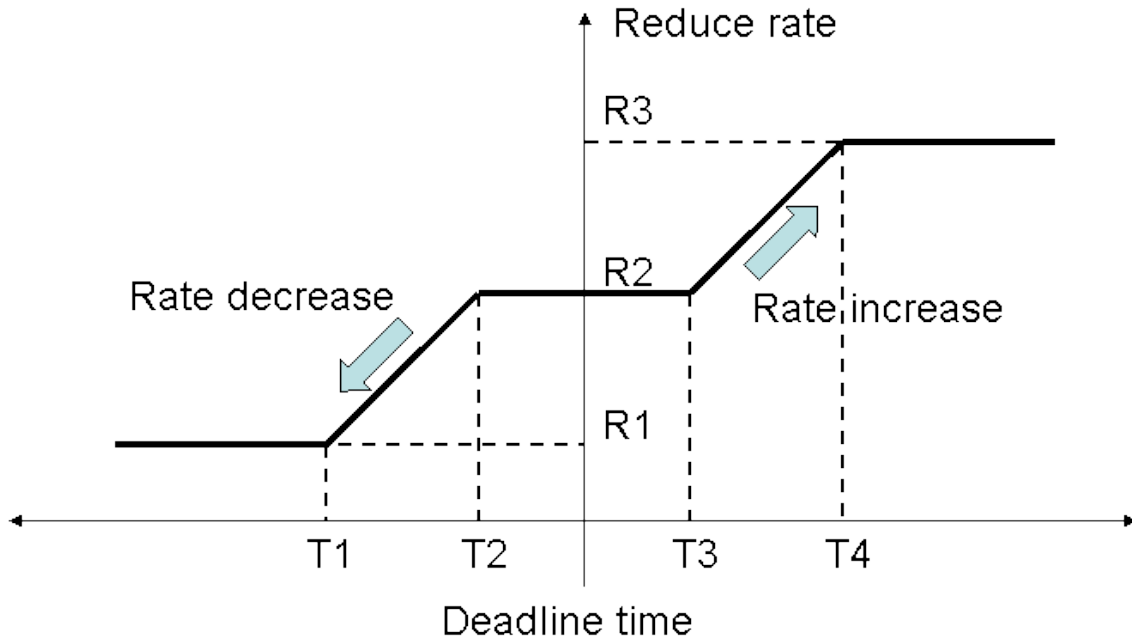
[T3] Start time of rate increase

[T4] End time of rate increase

[T2] Start time of rate decrease

[T1] End time of rate decrease

The time from T1 to T4 is set by relative time from deadline time by seconds. The specified value should be integer equal to 0 larger. The reduce rate from R1 to R3 is set by real number from 0 to 1.0.



How to calculate the reduce rate is explained below using above graph. In following formula, R_d means the reduce rate and T_r means the planned end time of a request, which is indicated by relative time from the deadline time.

When the request finishes before T_1 , R_d is equal to R_1 the minimum reduce rate uniformly.

$$R_d = R_1 \quad [\quad T_1 < T_r \quad]$$

When the request finishes between T_1 and T_2 , the more T_r increases, in other words, the more T_r closes to T_1 , the more R_d decreases proportionately. However, if T_1 is equal to T_2 , R_d is equal to R_1 .

$$R_d = ((R_2 - R_1) / (T_1 - T_2)) * T_r + ((T_1 * R_1 - T_2 * R_2) / (T_1 - T_2))$$

$$[\quad T_1 > T_2, T_1 \geq T_r > T_2 \quad]$$

$$R_d = R_1 \quad [\quad T_1 = T_2, T_1 \geq T_r > T_2 \quad]$$

When the request finishes between T_2 and T_3 in which the deadline time is included), R_d is equal to R_2 the ontime reduce rate uniformly.

$$R_d = R_2 \quad [\quad T_2 \geq T_r, T_r < T_3 \quad]$$

When the request finishes between T3 and T4, the more Tr increases, in other words, the more Tr closes to T4, the Rd increases proportionately. However, if T3 is equal to T4, Rd is equal to R3.

$$R_d = ((R_3 - R_2) / (T_4 - T_3)) * T_r + ((T_4 * R_2 - T_3 * R_3) / (T_4 - T_3))$$

$$[T_4 > T_3, T_4 \geq T_r > T_3]$$

$$R_d = R_3 [T_4 = T_3, T_4 \geq T_r > T_3]$$

When the request finishes after T4, Rd is equal to R3 the maximum reduce rate uniformly.

$$R_d = R_3 [T_4 < T_r]$$

The initial value of the parameters for adjusting the reduce rate, the range of the values, and limitations are as follows.

	Parameter name	Initial value	Maximum value	Minimum value	Limitations
R3	Maximum reduce rate	1.0	1.0	0	$R_3 \geq R_2$
R2	Ontime reduce rate	1.0	1.0	0	none
R1	Minimum reduce rate	1.0	1.0	0	$R_1 \leq R_2$
T3	Start time of rate increase	0	$2^{31}-1$	0	$T_3 \leq T_4$
T4	End time of rate increase	0	$2^{31}-1$	0	none
T2	Start time of rate decrease	0	$2^{31}-1$	0	$T_2 \leq T_1$
T1	Start time of rate decrease	0	$2^{31}-1$	0	none

The parameters for adjusting the reduce rate can be set per execution queue by a manager with Operator privilege or higher with **smgr**(1M). When the value of a parameter is changed during operation, the reduce rate calculated from the value after change modified is applied to the jobs that finishes after the changing.

Applying reduce rate to usage data

The reduce rate is applied to usage data of following four kinds of resources at the same rate at job termination.

- Elapse Time
- Number of CPUs
- Memory Usage
- Request Priority

The usage data of above 4 kinds of resources after applying the reduce rate is added to ShareDB of the user.

4.13 Incorporating External Policy

4.13.1 Overview of Incorporating External Policy

This feature enables you to customize the scheduling based on your own site policy (called External Policy below). JobManipulator performs scheduling based on External Policy by using the APIs created by your site, which are shown in following table. The following three External Policies can be incorporated into JobManipulator.

1. External Policy on submitting
JobManipulator can control the submitting on submitting a request based on External Policy such as limiting the resource usage per user/group.
2. External Policy of request priority
JobManipulator can adjust the priority of requests based on External Policy by setting a value determined by External Policy to a request as the request priority on submitting a request.
3. External Policy on assignment
JobManipulator can control the assignment on assigning a request based on External Policy such as limiting the number of CPUs that can be assigned simultaneously to a user/group.

The APIs for Incorporating External Policy feature are as follows.

API	Function
RLIM_connect	Establish the connection with External Policy Daemon
RLIM_disconnect	Disconnect the connection with External Policy Daemon
RLIM_chkresource	Check External Policy on submitting
RLIM_getpriority	Retrieve the request priority by External Policy
RLIM_chkrunlimit	Check External Policy on assignment
RLIM_relrunklimit	Release the check of External Policy on assignment

4.13.2 Setting of Incorporating External Policy feature

Set the following parameters in the configuration file (/etc/opt/nec/nqsv/nqs_jmd.conf or the file specified with the '-f' option on starting JobManipulator), and restart it. Each of above-mentioned three features can be enabled/disabled and target request type can be set to each feature.

4.13.2.1 *Enable Incorporating External Policy feature*

Each of the three features can be enabled or disabled.

- **API_SUBREQ_CHK:** *ON| OFF*
It enables or disables External Policy on submitting.
- **API_SET_PRI:** *ON| OFF*
It enables or disables External Policy on request priority.
- **API_ASSIGN_CHK:** *ON| OFF*
It enables or disables External Policy on assignment.

ON: enable
OFF: disable

It is OFF when the parameter is omitted. If a string other than ON/OFF is set or nothing is set behind ":", nqs_jmd outputs an error message to the standard output and does not start. If ON is set, the path of shared library of the APIs must be set.

4.13.2.2 *Set the type of target request*

The type of target request for each feature can be specified.

- **API_SUBREQ_CHK_TYPE:** *request_type [,request_type...]*
It sets the type of target request of External Policy on submitting.
- **API_SET_PRI_TYPE:** *request_type [,request_type...]*
It sets the type of target request of External Policy for request priority.
- **API_ASSIGN_CHK_TYPE:** *request_type [,request_type...]*
It sets the type of target request of External Policy on assignment.

The following can be set to *request_type*

normal: The requests submitted to normal queue

special: The requests submitted to special queue

urgent: The requests submitted to urgent queue

all: the requests submitted to normal queue, special queue , and urgent queue.

The request submitted to normal queue is the target, if this parameter is omitted. One or more types must be set when this parameter is set. When specifying multiple types separate them by using a comma (.). If a string other than **normal**, **special**, **urgent**, **all**

is set to *request_type* or nothing is set behind ":", nqs_jmd outputs an error message to the standard output and does not start.

4.13.2.3 Set the path of shared library of the APIs for Incorporating External Policy feature

Following parameter sets the path of shared library of the APIs.

API_LIB_PATH: *library_path*

The path is set to *library_path*.

The path must be set when one of the three features (API_SUBREQ_CHK/API_ASSIGN_CHK/API_SET_PRI) is enabled. If this setting is omitted, nqs_jmd outputs an error message to the standard error output and does not start.

4.13.3 Connection to External Policy Daemon

When Incorporating External Policy is enabled, JobManipulator connects to External Policy Daemon by using the [RLIM_connect](#) function.

When it failed to connect to External Policy Daemon, it retries at every scheduling interval. External Policy will not be reflected to the scheduling until the connection is established.

On terminating of JobManipulator, the connection is disconnected by using the [RLIM_disconnect](#) function.

4.13.4 External Policy on Submitting

When ON is set to **API_SUBREQ_CHK**, JobManipulator controls the submitting on submitting the request submitted to the queue specified with

API_SUBREQ_CHK_TYPE based on External Policy such as limiting the resource usage per user/group by using the [RLIM_chkresource](#) function. It performs the processing according to the return value of the **RLIM_chkresource** function as shown in following table.

Meaning of return value: return value	Processing
Allowed to submit: 0	JobManipulator performs scheduling for the request.
Disallowed to submit: -3 System error: -1	JobManipulator deletes the request and sends an e-mail to the address set to the request. The e-mail is as follows. Subject: NQSV request: <i>request_id.machine_id</i> is deleted. Message body:

	Reason: RLIM_chkresource error Detail: <i>the message returned by RLIM_chkresource function</i>
Connection error: -2	JobManipulator clears the target requests in ASSIGNED state from the scheduler map and stops scheduling them until it successfully reconnects to External Policy Daemon after retrying at the scheduling interval.

The timing checking External Policy on submitting is as follows.

- When a request is submitted.
- When JobManipulator starts.
- When JobManipulator successfully reconnects to External Policy Daemon.
- When an execution queue is bound to JobManipulator.

In addition, the check is not performed to following requests.

- a request in HELD state submitted with the Request Connection function
- a request in HELD state submitted with "**qsub -h**"
- a request in WAITING state submitted with "**qsub -a**"

4.13.5 External Policy on Request Priority

When ON is set to **API_SET_PRI**, for the request submitted to the queue specified with **API_SET_PRI_TYPE**, JobManipulator retrieves a value determined by External Policy and sets it to the request to adjust the priority of requests by using the [RLIM_getpriority](#) function. It performs the processing according to the return value of the **RLIM_getpriority** function as shown in following table.

Meaning of return value: return value	Processing
Retrieving success: 0	JobManipulator sets the request priority value to the request.
System error: -1	JobManipulator deletes the request and sends an e-mail to the address set to the request. The e-mail is as follows. Subject: NQSV request: <i>request_id.machine_id</i> is deleted. Message body: Reason: RLIM_getpriority error Detail: <i>the message returned by RLIM_getpriority function</i>
Connection error: -2	JobManipulator clears the target requests in ASSIGNED state from the scheduler map and stops scheduling them until it successfully

	reconnect s to External Policy Daemon after retrying at the scheduling interval.
--	--

The timing calling **RLIM_getpriority** is as follows.

- When a request is submitted.
- When JobManipulator starts.
- When JobManipulator successfully reconnects to External Policy Daemon.
- When an execution queue is bound to JobManipulator.

In addition, it is not performed to retrieve and set the request priority for the following request.

- a request in HELD state submitted with the Request Connection function
- a request in HELD state submitted with "**qsub -h**"
- a request in WAITING state submitted with "**qsub -a**"

In the following cases, the request is deleted and an e-mail is sent to the address set for the request.

- When the retrieved request priority value is out of the range (from -1024 to 1023).

The e-mail is as follows.

Subject: NQSV request: *request_id.machine_id* is deleted.

Message body: Reason: Request priority exceeds limit.

- When setting the request priority is failed.

The e-mail is as follows.

Subject: NQSV request: *request_id.machine_id* is deleted.

Message body: Reason: Request priority cannot be set.

4.13.6 External Policy on Assignment

4.13.6.1 Check External Policy on Assignment

When ON is set to **API_ASSIGN_CHK**, JobManipulator controls the assignment on assigning the request submitted to the queue specified with **API_ASSIGN_CHK_TYPE** based on External Policy such as limiting the number of CPUs that can be assigned simultaneously to a user/group, by using the [RLIM_chkrunlimit](#) function. It performs the processing according to the return value of the **RLIM_chkrunlimit** function as shown in following table. The timing checking External Policy on assignment is just before stage-in of the request and after the nodes have been determined which should be assigned to the request.

Meaning of return value: return value	Processing
---	------------

Allowed to assign: 0	JobManipulator assigns the request.
Disallowed to assign: -3	JobManipulator retries to assign it at the scheduling interval until it is allowed to assign.
System error: -1	JobManipulator deletes the request and sends an e-mail to the address set to the request. The e-mail is as follows. Subject: NQSV request: <i>request_id.machine_id</i> is deleted. Message body: Reason: RLIM_chkrunlimit error Detail: <i>the message returned by RLIM_chkrunlimit function</i>
Connection error: -2	JobManipulator clears the target requests in ASSIGNED state from the scheduler map and stops scheduling them until it successfully reconnects to External Policy Daemon after retrying at the scheduling interval.

4.13.6.2 Release checking External Policy on Assignment

When the jobs of the target request of checking External Policy on assignment terminate or deleted, [RLIM_relrunlimit](#) is executed, so that the state of the request can be managed in External Policy Daemon. The timing to release checking External Policy on assignment is as follows.

- When the request terminates.
- When the jobs of the request are canceled by unbinding the job server and so on.

Meaning of return value: return value	Processing
Release success: 0	None
System error: -1	JobManipulator deletes the request and sends an e-mail to the address set to the request. The e-mail is as follows. Subject: NQSV request: <i>request_id.machine_id</i> is deleted. Message body: Reason: RLIM_relrunlimit error Detail: <i>the message returned by RLIM_relrunlimit function</i>
Connection error: -2	JobManipulator clears the target requests in ASSIGNED state from the scheduler map and stops scheduling them until it successfully reconnects to External Policy Daemon after retrying at the scheduling interval.

4.13.7 API Functions

JobManipulator realizes Incorporating External Policy feature by calling the following API functions which you defines for your own site.

(1) RLIM_connect

Format

```
int RLIM_connect(char *msg)
```

Function

It establishes the connection with External Policy Daemon.
When it gets an error, it sets a description on the reason to *msg*.

Arguments

char *msg<OUT>: The buffer for an error message (within 128 characters).

Return value

Connection success :0
System error :-1
Connection error :-2

(2) RLIM_disconnect

Format

```
int RLIM_disconnect(char *msg)
```

Function

It disconnects the connection with External Policy Daemon.
When it gets an error, it sets a description on the reason to *msg*.

Arguments

char *msg<OUT>:The buffer for an error message (within 128 characters).

Return value

Disconnection success :0
System error :-1
Connection error :-2

(3) RLIM_chkresource

Format

int RLIM_chkresource([ReqID](#) *reqid, uid_t uid, gid_t gid, char *qname, Resources *resources, char *msg)

Function

It checks External Policy on submitting the request specified by the arguments, and returns the result.

When it gets an error, it sets a description on the reason to *msg*.

Arguments

ReqID *reqid<IN> : Request ID

```
typedef struct {
    int mid;          /* Machine ID */
    int seqno;        /* Sequential number */
    int subreq_no;    /* Subrequest number */
} ReqID;
```

uid_t uid<IN> : User ID of the request owner

gid_t gid<IN> : Group ID of the request owner

char *qname<IN> : Queue name of the request

Resources
*resources<IN> : Declared resources of the request

```
typedef struct {
    int job_number;
    int elapse;
    int cputime_per_job;
    long disk_per_job;
    int cpunum_per_job;
} Resources;
```

job_number: Number of jobs of the request

elapse: Declared elapse time (second) of the request

cputime_per_job: Declared CPU time per job (second) of the request

disk_per_job: An integer

cpunum_per_job: Declared CPU number per job of the request

char *msg<OUT> : The buffer for an error message (within 128 characters)

Return value

Allowed to submit : 0 (It returns "Allowed to submit" when the request is not over the limits of External Policy.)

System error :-1

Connection error :-2

Disallowed to submit :-3 (It returns "Allowed to submit" when the request is over the limit of External Policy.)

If the request is a hybrid request, which is submitted by **qsub(1)** command with **--job-separator** option or **---** option to require the different resources to each jobs, the CPU time of the first job group and the number of CPUs are set.

(4) RLIM_getpriority

Format

int RLIM_getpriority([ReqID](#)

*reqid, uid_t uid, gid_t gid, int *pri, char *msg)

Function

It retrieves the request priority determined based on External Policy for the request specified by the arguments, and returns the result.

When it gets an error, it sets a description on the reason to *msg*.

Arguments

ReqID *reqid<IN> :Request ID

uid_t uid<IN> :User ID of the request owner

gid_t gid<IN> :Group ID of the request owner

int *pri<IN/OUT> :Address in which the request priority is stored

char *msg<OUT> :The buffer for an error message (within 128 characters)

Return value

Success :0 (The retrieved value is set to *pri*.)

System error :-1

Connection error :-2

(5) *RLIM_chkrunlimit*

Format

```
int RLIM_chkrunlimit(ReqID *reqid, uid_t uid, gid_t gid, char *qname, char *msg)
```

Function

It checks the External Policy on assignment to the request specified by the arguments, and returns the result.

When it gets an error, it sets a description on the reason to *msg*.

Arguments

ReqID *reqid<IN> : Request ID

uid_t uid<IN> : User ID of the request owner

gid_t gid<IN> : Group ID of the request owner

char *qname<IN> : Queue name of the request

char *msg<OUT> : The buffer for an error message (within 128 characters)

Return value

Allowed to assign :0

System Error :-1

Connection Error :-2

Disallowed to assign :-3

(6) *RLIM_relrunlimit*

Format

```
int RLIM_relrunlimit(ReqID *reqid, uid_t uid, gid_t gid, char *msg)
```

Function

It changes the state of the request in External Policy Daemon and so on, and returns the result. For example, it can exclude the request from the targets checked by External Policy.

When it gets an error, it sets a description on the reason to *msg*.

Arguments

ReqID *reqid<IN> : Request ID

uid_t uid<IN> : User ID of the request owner

gid_t gid<IN> : Group ID of the request owner
char *msg<OUT> : The buffer for an error message (within 128 characters)

Return value

Success : 0
System error : -1
Connection error : -2

4.14 Power-saving Function

4.14.1 Overview of Power-saving Function

As power saving function, the following two functions are provided.

- Dynamic power saving function to control active nodes optimally according to state of running requests.
- Scheduled power saving function to control nodes based on schedule in which time period to stop a node is registered in advance.

Those functions enable to control power supply according to running state of execution nodes and to save unnecessary power consumption.

Power saving function can be used for execution hosts that meet all of the following conditions.

- Execution hosts of BMC (Baseboard Management Controller)
- Execution hosts of both queue bound to JobManipulator and JSV bound to JobManipulator
- Execution hosts which has never encountered failures
- Execution hosts ever linked-up after the operation is started, in which the JSV is bound to a queue and the queue is bound to JobManipulator

Setting for Execution Host:

- Set BMC to enable it.
- Install ipmitool to the Node Agent host.
- Start the Node Agent.

Refer to *NQSV User's Guide [Management]* for details of Node Agent.

The job server on the execution host must be set to start automatically after the execution host starts. For details on starting the job server automatically, refer to 3.5 *Job Server Startup* in *NQSV User's Guide [Management]*.

Note that only set either the job server startup by a launcher daemon of NQSV or the job server startup by systemd. If both are set, the job server startup may fail when the execution host is started by the power-saving function.

The eco-status of nodes can be displayed by **sstat -E --eco-status**.

```
#sstat -E --eco-status
```

ExecutionHost	EcoStatus	StateTransitionTime	OFF(D)	ACCUM
-----	-----	-----	-----	-----
Host1	PEAKCUT	2015-05-26 16:30:00	1	100
Host2	EXCLUDED	2015-06-30 12:00:00	1	101
Host3	-	-	1	98

The reason why the node has been excluded from the targets of DC power control can be displayed by **sstat -E --eco-status -f**.

```
%sstat -E --eco-status -f Host2
```

Execution Host: Host2	
Eco Status	= EXCLUDED
State Transition Time	= 2015-06-30 12:00:00
Exclude Reason	= START_FAIL
DC-OFF Times (Day)	= 1
DC-OFF Times (ACCUM)	= 101

4.14.2 Dynamic Power-saving Function

Dynamic power-saving function is a function to turn on/off the DC power dynamically in accordance with the operating state of the nodes, which is also called Dynamic DC Control. It enables peak cut of power consumption by adjusting the maximum number of operation nodes with setting maximum number of operation nodes per scheduler. JobManipulator powers off a part of nodes properly to make operation nodes not more than this value. One of following modes on urgency of peak out can be selected.

- (1) Power off a node after the running request in it is finished.
- (2) Power off a node immediately with rerunning the running request.

The nodes without requests assigned in a period from current time are powered off. However, if too many nodes are powered off, it will affect the operation. In order to avoid this, minimum number of operations for each queue should be set, so that operation nodes are not less than this value.

When there is a request waiting to be assigned, the nodes will be powered on.

At that time, the total number of operating node of each queue is kept under "the maximum number of operation nodes". When nodes will power on by urgent request,

"the maximum number of operation nodes" is ignored for guarantee of execution of the urgent request.

When this function is set as ON, all job servers bound to the queue of the JobManipulator instance are targets of power control, so if you want to exclude some node from power control such as for maintenance, it need to be unbound from all queues of the JobManipulator instance.

4.14.2.1 *Setting of Dynamic Power-saving Function*

The dynamic power-saving function can be enabled or disabled per scheduler by using the **set dynamic_dc_control** subcommand of **smgr(1M)**.

```
#smgr -P m
Smgr : set dynamic_dc_control = on | off
```

on Start Dynamic Dc Control

off Stop Dynamic Dc Control

When changing it from **on** to **off**, the nodes bound with the queue bound with JobManipulator except the node with HW failure and the node stopped according to Eco Schedule will be started immediately, and then Dynamic Dc Control is stopped.

The initial value is **off**. Operator privilege is needed.

The setting of dynamic power-saving function can be displayed by using **sstat(1)** with the **-S,-f** option.

```
#sstat -S -f
JobManipulator Server Host: bsv.nec.co.jp
  JobManipulator Version   = R1.00
  JobManipulator Status    = Active
Scheduler ID               = 1
      :
  Auto Delete Resource Reservation = OFF
  Forced Re-Scheduling      = OFF
  Dynamic DC Control        = OFF
:
```

4.14.2.2 *Setting of the Maximum Number of operation nodes*

The maximum number of operation nodes can be set per scheduler by using the **set max_operation_hosts** subcommand of **smgr(1M)**.

```
#smgr -P m
Smgr : set max_operation_hosts = number_of_hosts
```

The DC power supplies of a part of nodes are turned off so that the nodes in operation are not more than the maximum number of operation nodes.

- The range of the value is 0-10240.
- The initial value is 10240.
- Operator privilege is needed.

The setting of the maximum number of operation nodes can be displayed by using **sstat(1)** with the **-S,-f** option.

```
#sstat -S -f
JobManipulator Server Host: bsv.nec.co.jp
  JobManipulator Version   = R1.00
  JobManipulator Status    = Active
  Scheduler ID              = 1
  :
Auto Delete Resource Reservation = OFF
Forced Re-Scheduling        = OFF
Dynamic DC Control          = OFF
Max Operation Hosts        = 10240
:
```

4.14.2.3 **Setting of the Mode on Urgency of Peak Cut**

The mode on urgency of peak cut can be set per scheduler by using the **set peak_cut_urgency** subcommand of **smgr(1M)**.

```
#smgr -P m
Smgr : set peak_cut_urgency = wait_run | right_now
```

Set whether to power off a node immediately when the node will be powered off by the function of adjusting maximum operation of Dynamic DC Control.

wait_run

The node is powered off after the running request is finished.

right_now

The running request is rerun, the assigned requests are rescheduled and then the node is powered off immediately.

The initial value is **wait_run**. Operator privilege is needed.

The setting of the mode on urgency of peak cut can be displayed by using **sstat(1)** with the **-S, -f** option.

```
#sstat -S -f
JobManipulator Server Host: bsv.nec.co.jp
  JobManipulator Version   = R1.00
      :
  Auto Delete Resource Reservation = OFF
  Forced Re-Scheduling      = OFF
  Dynamic DC Control        = OFF
  Max Operation Hosts       = 10240
  Peak Cut Urgency         = wait_run
      :
```

4.14.2.4 **Setting of the Minimum Number of Operation Nodes of A Queue**

The minimum number of operation nodes can be set per queue by using the **set queue min_operation_hosts** subcommand of **smgr(1M)**.

```
#smgr -P m
Smgr : set queue min_operation_hosts = number_of_hosts queue_name
```

Set the minimum number of operation nodes of the queue specified by *queue_name* to *number of hosts*. The DC power of a node can be turned off by Dynamic DC Control so as not to make the number of operation nodes of the queue less than this value.

- The initial value is 10240
- The range of value is 0-10240.
- Operator privilege is needed.

The setting of the minimum number of operation nodes can be displayed by using **sstat(1)** with the **-Q, -f** option.

```
#sstat -Q -f
Execution Queue: bq1
  ...omission...
  Min Operation Hosts = 10240
  Request Statistical information:
  ...omission...
```

4.14.2.5 Setting of the DC Power Off Limit

This feature is to limit the number of times of stopping a node by Dynamic Power-saving function per day in since frequent stop-start of node may cause a HW failure. The number of times to stop the node per day is limited to the number of times that is set by using **set dc-off_limit** subcommand of **smgr(1M)**.

```
#smgr -P m
Smgr : set dc-off_limit = number_of_times
```

Set DC Power Off Limit to *number_of_times*.

- The range of value is 1-200.
- The default value is 5.
- Operator privilege is needed.

The setting of the DC Power Off Limit can be displayed by using **sstat(1)** with the **-S,-f** option.

```
# sstat -S -f
JobManipulator Server Host: bsv.nec.co.jp
  JobManipulator Version   = R1.00
  JobManipulator Status    = Active
  Scheduler ID              = 1
  :
  Dynamic DC Control        = OFF
  Max Operation Hosts       = 10240
  Peak Cut Urgency          = wait_run
  Min Idle Time             = 300S
  Estimated DC-OFF Time     = 3600S
  DC-OFF Limit              = 5
  Use Overtake Priority = {
    Normal = OFF
    Special = OFF
  }
  :
```

4.14.2.6 Setting of the Minimum Idle Time

This feature is to stop a node after the elapse of a certain period of time (Minimum Idle Time) from following time in order to avoid stopping the node right after it becomes the target of operation or the job in it is finished. If a job is executed in the node during this period, it will not be stopped.

The start of this period is the latest one of following time.

- When there is no running job in the node.
- When the node is started.
- When JobManipulator is started.
- When you enable the Dynamic Power-saving function by smgr.
- When you bind the Job Server to a queue which is bound with JobManipulator.
- When you bind JobManipulator to the queue with the node bound.

The Minimum Idle Time can be set per scheduler by using the **set min_idle_time** subcommand of **smgr(1M)**.

```
#smgr -P m
Smgr : set min_idle_time = seconds
```

Set the Min Idle Time to *seconds*.

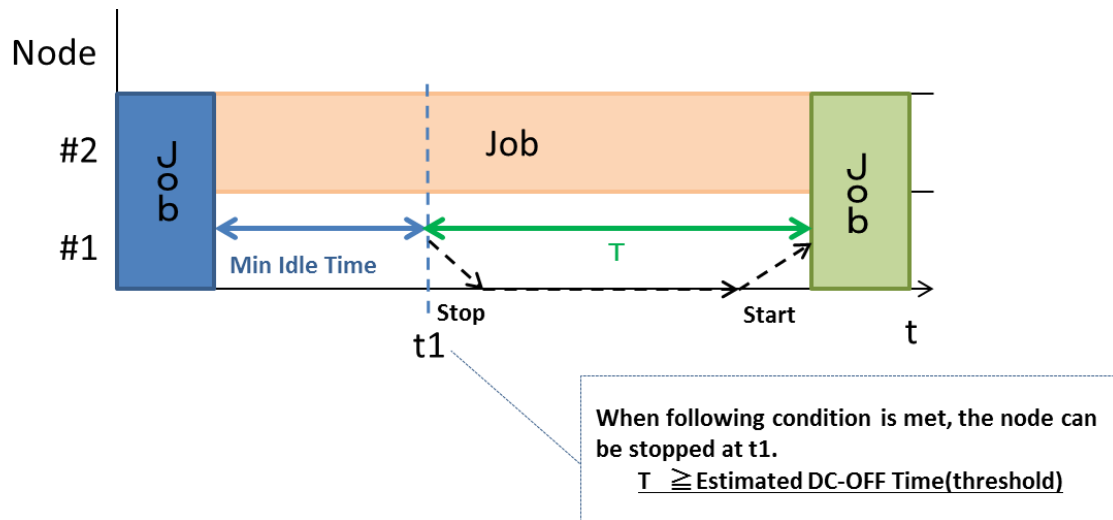
- The range of value is 0-2147483647.
- The default value is 300.
- Operator privilege is needed.

The setting of Min Idle Time can be displayed by using **sstat(1)** with the **-S,-f** option.

```
# sstat -S -f
JobManipulator Server Host: bsv.nec.co.jp
  JobManipulator Version   = R1.00
  JobManipulator Status    = Active
  Scheduler ID              = 1
  :
  Dynamic DC Control        = OFF
  Max Operation Hosts       = 10240
  Peak Cut Urgency          = wait_run
  Min Idle Time             = 300S
  Estimated DC-OFF Time     = 3600S
  :
```

4.14.2.7 **Setting of the Estimated DC-OFF Time**

This feature is to stop a node when it is possible to stop for not less than a certain period of time, in other words, when there is no job scheduled from current time to Estimated DC-OFF Time (threshold) later in the node as shown in following figure. It is to avoid unnecessary stop of the node such as that the node is stopped but is started immediately after the stopping of it.



The Estimated DC-OFF Time (threshold) can be set per scheduler by using the **set estimated_dc-off_time** subcommand of **smgr(1M)**.

```
#smgr -P m
Smgr : set estimated_dc-off_time = seconds
```

Set the Estimated DC-OFF Time to *seconds*.

- The unit is second.
- The range of value is 2-2147483647.
- It must be equal to or larger than the sum of the margin for stopping a node and the margin for starting a node. (Refer to 4.16.2.8 Setting of the Margin for Stopping a Node and the Margin for Starting a Node)
- The default value is 3600.
- Operator privilege is needed.

The setting of Min Idle Time can be displayed by using **sstat(1)** with the **-S,-f** option.

```
# sstat -S -f
JobManipulator Server Host: bsv.nec.co.jp
JobManipulator Version   = R1.00
JobManipulator Status    = Active
Scheduler ID              = 1
:
Dynamic DC Control        = OFF
Max Operation Hosts       = 10240
Peak Cut Urgency          = wait_run
Min Idle Time             = 300S
```

<code>Estimated DC-OFF Time = 3600S</code> :

4.14.2.8 *Setting of the Margin for Stopping a Node and the Margin for Starting a Node*

Because of taking time of several minutes for stopping a node, the margin for stopping a node is provided as expected time of stopping a node. And because of taking time of several minutes for starting a node, the margin for starting a node is provided as expected time of starting a node. The minimum time between stopping a node and starting the node for power-saving is "the margin for stopping a node" + "the margin for starting a node".

The margin for stopping a node and the margin for starting a node can be set in the configuration file (/etc/opt/nec/nqsv/nqs_jmd.conf).

<code>MARGIN_FOR_STOP_HOST:300</code> <code>MARGIN_FOR_START_HOST:600</code>

Specify the margin for stopping a node with MARGIN_FOR_STOP_HOST, and the margin for starting a node with MARGIN_FOR_START_HOST.

- The unit is second.
- The range of value is 1-2147483647.
- The default value of MARGIN_FOR_STOP_HOST is 300.
- The default value of MARGIN_FOR_START_HOST 600.
- These two parameters can be omitted. If omitted, the values are the default.

These parameters are also applied to Scheduled Power-saving function.

-
- In the dynamic power-saving function, a node is excluded from power saving control targets if it remains powered on after the margin to stop the node has elapsed.
 - The node that is excluded from power saving control targets returns to the target by LINKUP the JSV.
 - The node status can be checked with the -E --eco-status option of the sstat(1) command or EcoStatus with the -E -f option.
-

4.14.3 Scheduled Power- saving Function

Scheduled power-saving function is a function to turn on/off the DC power of execution host according to on/off schedule (scheduled power-saving period) that administrator determines if there is disproportionate operating rate of the nodes. (e.g. High on weekdays and low on weekends. There exists seasonality in operating rate. Etc.)

Scheduled power-saving function begins to stop the execution host after schedule start time of scheduled power-saving period (Eco Schedule), and to start the execution host so that job operation can be re-started at ending time of Eco Schedule. When Dynamic Power-saving function is enabled, whether to start the execution host is determined by Dynamic Power-saving function.

During the period of Eco Schedule, any request cannot be assigned.

However, as for urgent request, if it can be assigned and executed on the execution host that is stopped according to Eco Schedule after starting this execution host, then the execution host is started to execute it after deleting the Eco Schedule.

4.14.3.1 **Create Eco Schedule**

Eco Schedule is created by **smgr**(1M) with create eco_schedule sub-command. The operator privilege or higher is required for this creation.

```
create eco_schedule starttime = start_time endtime= end_time  
hostname = host_name
```

- Specify the start time of Scheduled power-saving period with starttime.
- Specify the end time of Scheduled power-saving period with endtime.
- Specify the target host name with hostname.

Eco Schedule ID (from 0 to 9999) is assigned. This Eco Schedule ID is used to delete it.

Note that the interval between starttime and endtime needs to be equal to or larger than following.

[Margin for stopping a node + Margin for starting a node.]

Multiple Eco Schedule can be created but any of periods for the same execution host cannot overlap each other.

Additionally, in case of the following, Eco Schedule cannot be created.

- During the specified period, there has existed assigned request in the specified execution host.
- During the specified period, a Reservation Section is set with specified queue in the specified execution host.

4.14.3.2 *Delete Eco Schedule*

Eco Schedule is deleted by **smgr**(1M) with delete eco_schedule sub-command. The operator privilege or higher is required for this deletion.

```
delete eco_schedule = eco_schedule_id
```

4.14.3.3 *Display Eco Schedule*

Eco Schedule ID, start time of Eco Schedule, end time of Eco Schedule and execution host are displayed by **sstat -D** command.

```
$sstat -D
EcoID EcoStartTime      EcoEndTime      ExecutionHost
-----
  0 2014-12-06 18:00:00 2014-12-06 23:00:00 host1
  1 2014-12-06 18:00:00 2014-12-06 23:00:00 host2
```

Additionally, detail information can be displayed by **sstat -Df**.

```
$sstat -Df
Eco Schedule ID: 0
    Scheduled Start Time    = 2014-12-06 18:00:00
    Scheduled End Time      = 2014-12-06 23:00:00
    Number of Scheduled Hosts = 1
    Scheduled Hosts:
        host1

Eco Schedule ID: 1
    Scheduled Start Time    = 2014-12-06 18:00:00
    Scheduled End Time      = 2014-12-06 23:00:00
    Number of Scheduled Hosts = 1
    Scheduled Hosts:
        host2
```

4.15 Custom Resource Function

4.15.1 Overview of Custom Resource Function

In scheduling based on defined custom resource information, the custom resource function is the function which controls the use amount of the custom resource used at the same time. A system administrator defines a virtual resource optionally. This is called "custom resource information. A custom resource name, and a unit for which a resource are spent, the reach of the target where the resource amount used at the same time is controlled and the upper limit value are set as custom resource information.

The user specify the use amount as each custom resource name in `--custom` option by the submit command (**qsub(1)**, **qlogin(1)** or **qrsh(1)**) at the time of request submitting. JobManipulator refers to this value and totals the use amount of the custom resource used at the same time, and schedules so that there isn't that beyond the upper limit value of the defined custom resource.

Refer to *NQSV User's Guide [Management]* for details of a custom resource function, setting method of the custom resource information, a setting method of a queue. Refer to *NQSV User's Guide [Operation]* for details of a request submitting method with the custom resource function.

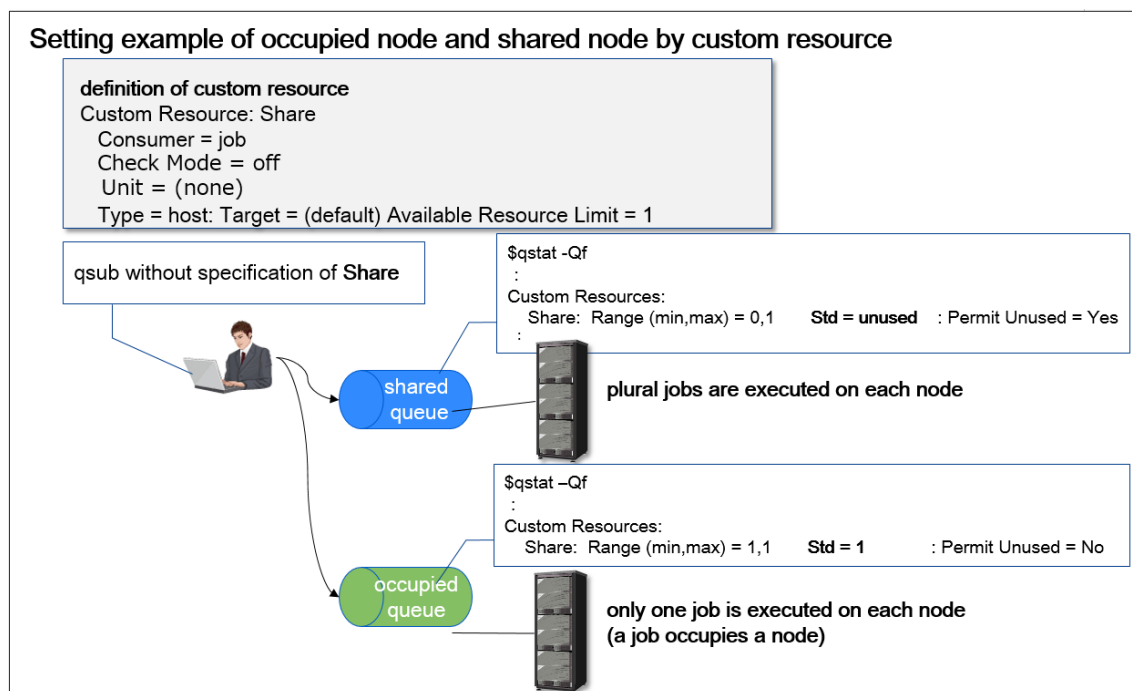
4.15.2 Scheduling using Custom Resource Information

The use amount of the custom resource specified in the request can be displayed by using **qstat(1)** with **-f** option("Custom Resources" item). Refer to *NQSV User's Guide /Operation/* for details.

When a request submitted with the use amount of the custom resource, JobManipulator counts the use amount of the custom resource of a request by the consumption unit of the custom resource, and a job is assigned in whichever time on the scheduler map also not to exceed the maximum of the simultaneous available resource in the reach of the target classification of the use amount control (batch server or execution host).

4.15.3 Examples of Using Custom Resource Function

4.15.3.1 Setting of occupied nodes and shared nodes



4.15.3.2 Scheduling by Electric power

setting example of scheduling by electric energy

definition of custom resource
Custom Resource: Power
Consumer = job
Check Mode = moment
Terminate Job = on
Unit = kW
Type = host: Target = (default) Available Resource Limit = 300

qsub with resource consumption pf power

\$qstat -Qf
:
Custom Resources:
Power: Range (min,max) = 10,50 Std = 30 : Permit Unused = No



queue A



It is scheduled not to exceed Available Resource Limit(=300) of custom resource on the each node.

4.15.3.3 Scheduling by Software License of ISV software

setting example of scheduling by license

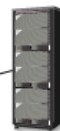
definition of custom resource
Custom Resource: License
Consumer = request
Check Mode = off
Terminate Job = off
Unit = (none)
Type = bsv: Available Resource Limit = 10

qsub with resource consumption License



queue A

queue B



It is scheduled not to exceed Available Resource Limit(=10) of custom resource in BSV

4.16 Provisioning with OpenStack

4.16.1 Overview of Provisioning with OpenStack

Virtual machine (VM) and baremetal server are supported as provisioning with OpenStack. Please refer to *NQSV User's Guide [Management]* for detail of provisioning with OpenStack. Please refer to *NQSV User's Guide [Operation]* for detail the method of

submitting of provisioning with OpenStack. JobManipulator does scheduling for virtual machine (VM) and baremetal server.

This function is NOT available for the environment whose execution host is SX-Aurora TSUBASA system.

4.16.2 Setting Re-scheduling Waiting Time at Failure of Start of Execution Host

At failing of starting virtual machine (VM) and baremetal server under environment of provisioning with OpenStack, all request assigned to such execution host are re-scheduled and starting is retried for beginning of request according to situation of scheduling after re-scheduling.

Execution host of which set the waiting time of the re-scheduling and failed in a start fixes re-scheduling by the template which failed in a start. This time is called the re-scheduling waiting time which is at the time of execution host start failure. Incorrect of a template is considered as the failed cause of the start as which such template was designated. There is a possibility that a retry of a start is failed once again in that case. Execution host which is set the re-scheduling waiting time by the template which failed in a start by this function, and maintenance is done during that mean time and becomes possible to prevent repeating start failure.

Re-scheduling waiting time can be set retry waiting time (second) by using **provisioning_start_retry_time** subcommand of **smgr(1M)**.

```
#smgr -P m
Smgr: set provisioning_start_retry_time = <seconds>
```

- The unit is second.
- The initial value is 0. In this case, re-scheduling is done immediately
- The value after changing of this setting is apply execution host that waiting re-scheduling from before changing of thin setting
- Operator privilege is needed.

The setting of this function can be displayed using **sstat(1)** with **-S -f** option.

```
$ sstat -S -f
:
  Stage-in Margin = {
    Additional Margin for Escalation = 0S
    Stage-in Threshold                = 0S
    First Stage-in Time               = 0S
  }
  Provisioning Start Retry Time = 0S <- re-scheduling retry time
Request Statistical Information:
:
```

Waiting of re-scheduling is released by using **stop waiting_retry** subcommand of **smgr** (1M)

```
#smgr -P m
Smgr: stop waiting_retry executionhost = <hostname>
```

- Execution host name of provisioning is specified to hostname.
- Operator privilege is needed.

Scheduling with specifying template for the execution host specified to hostname is re-started.

4.16.3 Scheduling of the Execution Hosts at Provisioning

When provisioning of virtual machine (VM) and baremetal server in the environment of provisioning with OpenStack requests are submitted with specifying template. In this case it is set so that start and stop time of virtual machine (VM) is included in Elapse margin, Also it is set so that start and stop time is timeout for booting and timeout for stopping of template. Please refer to *NQSV User's Guide [Management]* for detail of timeout for booting and timeout for stopping of template.

When a request is executed on virtual machine (VM), the request is executed after starting of virtual machine (VM) and after finishing of the request the virtual machine (VM) is stopped. When a request is executed on baremetal server the request is executed after starting of the baremetal server and after finishing of the request the baremetal server is stopped.

At failure of stopping of virtual machine (VM) and baremetal server which is started under the environment of provisioning with OpenStack, such host is omitted from operation. Such host is displayed by using **sstat(1)** with **-E --hw-failure** option.

```
$ sstat -E --hw-failure
ExecutionHost Status          V
-----
executionhost1 EXCLUDED      -
```

Execution host which is omitted from scheduling is added to scheduling by unbind from all queue (bind with JobManipulator) and bind to any queue after solving problem.

The execution host of virtual machine is target of power saving function but baremetal server is not target of power saving function because baremetal server is started and ended when starting and ending of request that is assigned to such execution hist.

Situation as baremetal server is omitted from power saving function is displayed by using **sstat(1)** with **-E --eco-status**.

```
$ sstat -E --eco-status
ExecutionHost EcoStatus StateTransitionTime OFF (D) ACCUM
-----
BareMetal host EXCLUDED 2016-07-13 09:07:30 0 0
```

FIFO scheduling is scheduling that runs in the order in which requests are submitted. To use, it is available to the system administrator by setting the set queue `schedule_type` subcommand of the `smgr (1M)` command to `fifo`. Fifo scheduling settings are per queue.

In a queue that you set up for FIFO scheduling, the start time of the submitted request is determined immediately if you have the resources needed to execute at that time. Also, if the start time of the previously submitted request is determined, the next request is scheduled, and the start time is determined if there is a necessary resources for execution.

Requests that cannot be started immediately due to the reservation interval, time specification, or workflow will not be displayed until the resource is available and the execution time is possible.

Exceptions include requests that are in the HELD state for a collaboration request, requests that hold requests by the `qhold(1)` command, and requests that are in waiting state by the `-a` option of the `qalter(1)` command, which are over bound by requests submitted later.

When FIFO scheduling is applied, overtaking control on pickup is disabled. In addition, FIFO scheduling cannot be used in conjunction with the first stage-in time.

4.16.4 The Waiting time of Stage-out of the Request on Baremetal Server

When execution host is a baremetal server, a stage out doesn't put it into effect concurrently with execution starting of other requests, and after a stage out of a request has been completed, I begin to restart a baremetal server and carry out a request of following. Therefore it is possible to consider and schedule stage out time by setting time to have a stage out of a request (the stage out waiting time).

The stage out waiting time is set by the **set queue wait_stageout** sub-command of the **smgr(1M)** command.

```
# smgr -P m
Smgr: set queue wait_stageout = <second> < queue - name>.
```

- The stage out waiting time is set in *second*.
- The unit is a second.
- The initial value is 0.
- Operator privilege is needed

A following request is assigned as the one which restarts a baremetal server after the time when the stage out waiting time was emptied from the execution end scheduled time of the request carried out by a baremetal server. The set stage out waiting time can be confirmed by **-Q -f** option of the **sstat(1)** command.

4.17 Provisioning with Docker

4.17.1 Overview of Provisioning with Docker

Container is supported as provisioning with Docker. Please refer to *NQSV User's Guide [Management]* for detail of provisioning with Docker. Please refer to *NQSV User's Guide [Operation]* for detail the method of submitting of provisioning with Docker.

JobManipulator does scheduling for container.

This function is NOT available for the environment whose execution host is SX-Aurora TSUBASA system.

4.17.2 Setting Re-scheduling Waiting Time at Failure of Start of Execution Host

At failing of starting container under environment of provisioning with Docker, all request assigned to such execution host are re-scheduled and starting is retried for beginning of request according to situation of scheduling after re-scheduling.

Execution host of which set the waiting time of the re-scheduling and failed in a start fixes re-scheduling by the template which failed in a start. This time is called the re-scheduling waiting time which is at the time of execution host start failure. The scheduling waiting time is same the case of [4.17 Provisioning with OpenStack](#). Incorrect of a template is considered as the failed cause of the start as which such template was designated. There is a possibility that a retry of a start is failed once again in that case. Execution host which is set the re-scheduling waiting time by the template which failed in a start by this function, and maintenance is done during that mean time and becomes possible to prevent repeating start failure.

For details of setting of re-scheduling waiting time, displaying of setting and releasing of setting please refer to [4.17.2 Setting Re-scheduling Waiting Time at Failure of Start of Execution Host](#).

4.17.3 Scheduling of the Execution Hosts at Provisioning

When provisioning of container in the environment of provisioning with Docker requests are submitted with specifying template. In this case it is set so that start and stop time of container is included in Elapse margin. Please refer to *NQS User's Guide [Management]* for detail of timeout for booting and timeout for stopping of template.

When a request is executed on container, the request is executed after starting of container and after finishing of the request the container is stopped.

At failure of stopping of container which is started under the environment of provisioning with Docker, such host is omitted from operation. Such host is displayed by using `sstat(1)` with `-E --hw-failure` option.

```
$ sstat -E --hw-failure
ExecutionHost Status          V
-----
executionhost1 EXCLUDED      -
```

Execution host which is omitted from scheduling is added to scheduling by unbind from all queue (bind with JobManipulator) and bind to any queue after solving problem.

The execution host of container is target of power saving function.

4.18 Setting Function of the First Stage-in Time

When the request which does file staging is assigned around the head of the scheduler map there is a possibility that its scheduled start time is cleared because of delay of the stage-in. So, you can set the estimated first stage-in time as First Stage-in Time per

scheduler. JobManipulator consider first stage-in time of a request to be it at scheduling.

When stage-in finish during First Stage-in Time, scheduled start time does not be cleared.

First Stage-in Time is set by using **set stage-in_margin first_stage-in_time** subcommand of **smgr(1M)**.

```
#smgr -P m
Smgr: set stage-in_margin first_stage-in_time = <value>
```

- First stage-in time is set to value.
- The unit is second.
- The initial value is 0.
- Operator privilege is needed.

It is possible to confirm the set value by **sstat(1)** with **-S -f** option.

```
$ sstat -S -f
:
JobManipulator Version   = R1.00
:
Keep Forward Schedule    = 0S
Stage-in Margin = {
    Additional Margin for Escalation = 0S
    Stage-in Threshold = 0S
    First Stage-in Time = 0S
}
:
```

4.19 Pre-Staging Function

4.19.1 Overview of Pre-Staging Function

The function to which a request can be assigned without staging is supported. The load of filesystem from simultaneous occurring of a lot of staging of request at assignment or escalation will be reduced by this function. Staging frequency between assignment and start of execution of a request will be reduced too. Stage-in will start when time to scheduled start time is less than stage-in starting time threshold set by **set stage-in_margin stage-in_threshold** subcommand of **smgr(1M)** command.

4.19.2 Setting of Stage-in Starting Time Threshold

Stage-in starting time threshold is set by using the **set stage-in_margin stage-in_threshold** subcommand of **smgr(1M)** command.

```
#smgr -P m
Smgr: set stage-in_margin stage-in_threshold = <value>
```

- Stage-in starting time threshold is set to value. The unit is second.
- The initial value is 0. In this case, staging start immediately after assignment of a request on scheduler map.
- Operator privilege is needed.
- It becomes effective by assignment after setting change at the time of setting change.

The setting of this function can be displayed using **sstat(1)** with the **-S -f** option.

```
$ sstat -S -f
:
JobManipulator Version   = R1.00
:
Keep Forward Schedule    = 0S
Stage-in Margin = {
    Additional Margin for Escalation = 0S
    Stage-in Threshold = 0S
    First Stage-in Time = 0S
}
:
```

4.20 Display the Detail of the Execution Host Information

Detailed information of the execution host can be displayed by using **sstat(1)** command with **-E -f** option. The information that is displayed by using **-E** option, **-E --eco-status -f** option and **-E --hw-failure** option are displayed collectively.

An image of execution of "**sstat -E -f**" is as follows.

```
$sstat -E -f
Execution Host: Host1
CPU Number Ratio = 1.000000
```

```

CPU Number Ratio of RSG = {
    RSG 0 = 1.000000
}
Memory Size Ratio = 0.000000
Memory Size Ratio of RSG = {
    RSG 0 = 0.000000
}
Eco Status = {
    Status = EXCLUDED
    State Transition Time = 2017-06-20 10:49:36
    Exclude Reason = HW_FAILURE
    DC-OFF Times (Day) = 0
    DC-OFF Times (ACCUM) = 0
}
Hardware Failure = {
    Status = CPUERR
}

Execution Host: Host2
CPU Number Ratio = 1.000000
CPU Number Ratio of RSG = {
    RSG 0 = 1.000000
}
Memory Size Ratio = 0.000000
Memory Size Ratio of RSG = {
    RSG 0 = 0.000000
}
Eco Status = {
    DC-OFF Times (Day) = 0
    DC-OFF Times (ACCUM) = 0
}
Hardware Failure = {
    Status = EXCLUDED
    Exclude Reason = VE_DEGRADATION
    VE Degradation = YES
}

```

An image of execution of "sstat -E -f -a" is as follows. Hardware Failure column is not displayed to unbound host. In this example Host3 is unbound and Host4 is bound.

```

$sstat -E -f -a
Execution Host: Host3
CPU Number Ratio = 1.000000
CPU Number Ratio of RSG = {
    RSG 0 = 1.000000
}

```

```

.....
RSG 31 = 1.000000
}
Memory Size Ratio = 0.000000
Memory Size Ratio of RSG = {
    RSG 0 = 0.000000
    .....
    RSG 31 = 0.000000
}
Eco Status = {
    Status = EXCLUDED
    State Transition Time = 2017-06-20 10:49:36
    Exclude Reason = UNBIND
    DC-OFF Times (Day) = 0
    DC-OFF Times (ACCUM) = 0
}
Execution Host: Host4
CPU Number Ratio = 1.000000
CPU Number Ratio of RSG = {
    RSG 0 = 1.000000
    .....
    RSG 31 = 1.000000
}
Memory Size Ratio = 0.000000
Memory Size Ratio of RSG = {
    RSG 0 = 0.000000
    .....
    RSG 31 = 0.000000
}
Eco Status = {
    DC-OFF Times (Day) = 0
    DC-OFF Times (ACCUM) = 0
}
Hardware Failure = {
    Status = EXCLUDED
    Exclude Reason = VE_DEGRADATION
    VE Degradation = YES
}

```

4.21 Node group selection function for minimum network topology

4.21.1 Overview

For the purpose of faster communication of MPI jobs, nodes are selected so that the number of network topology node groups to which the node belongs is minimized.

The JobManipulator's assignment policy prioritizes the nodes whose requests can be executed at the earliest time. Therefore, even if the assignment settings take the network topology into account, there is a possibility that the request will be assigned across the network switches.

For example, if you submit requests in the following order: Req1, Req2, Req3, Req4, Req5, Req3 will be scheduled across two network switches. The default assignment policy of JobManipulator gives priority to run jobs on free nodes earlier.

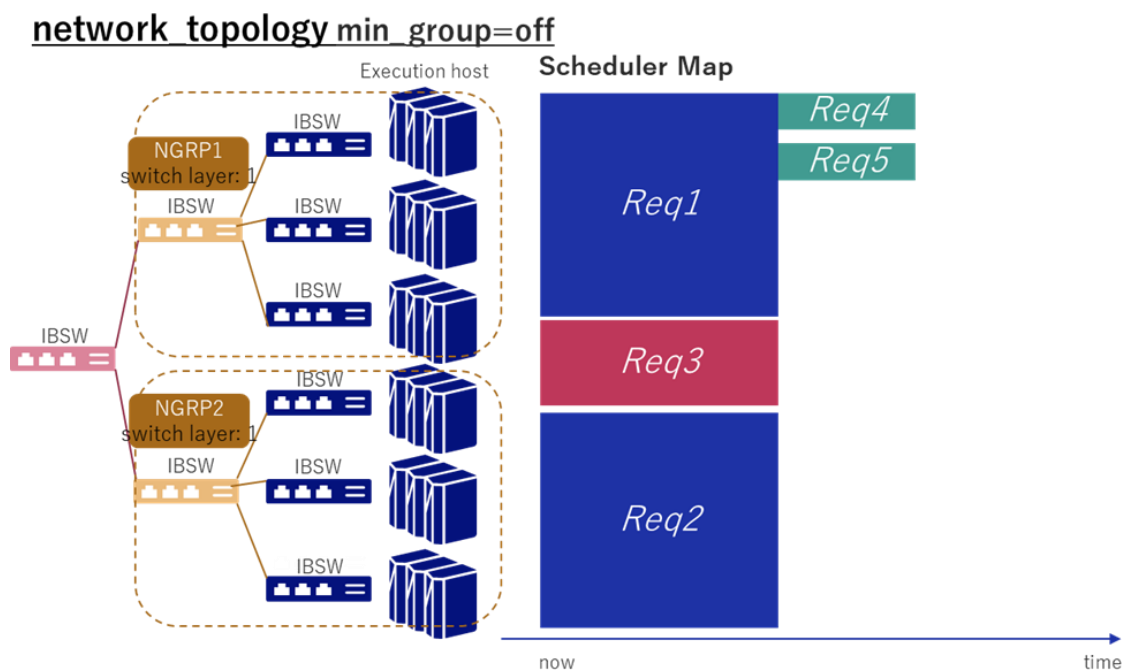


Figure 4-2 Example of scheduling that prioritizes job execution

Node group selection function for minimum network topology selects the smallest possible network topology group even if the start time is delayed.

It selects nodes within the minimum number of network topology groups for each request, even if there are nodes that can assign the request at a faster time.

If this function is applied in the previous example, the scheduling will be as follows: Since one group is the minimum for Req3, select nodes so that they are selected within one group, and assign them afterward. Req4 and Req5 will be scheduled in the free space in front.

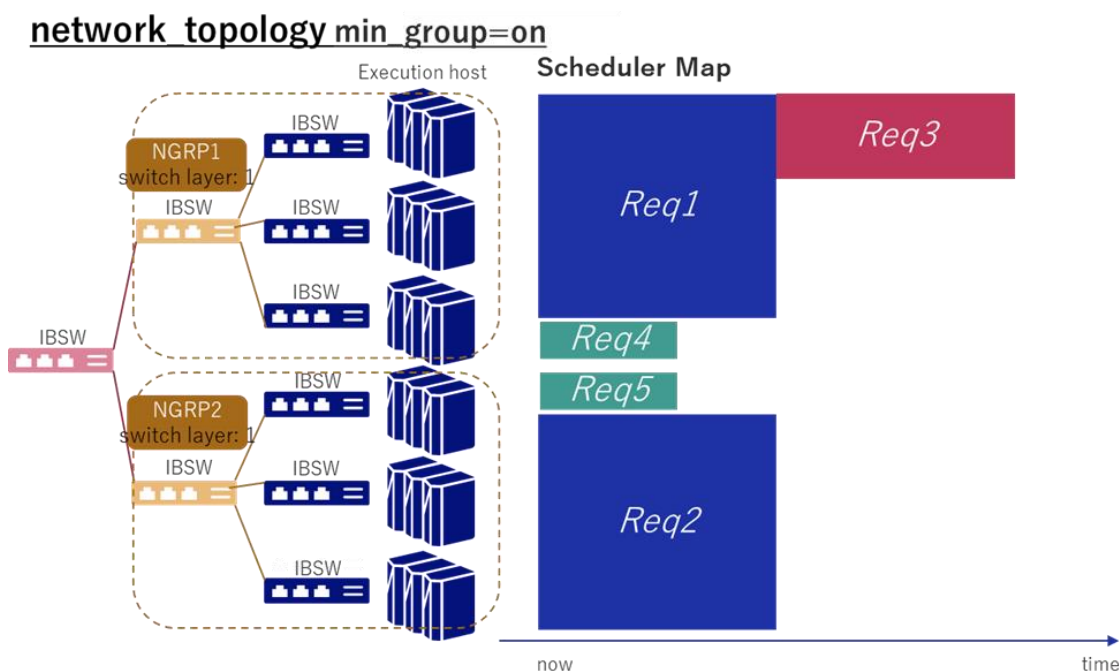


Figure 4-3 Example of network topology-first scheduling

- The purpose of this function is to provide faster communication for MPI jobs. Therefore, requests with the job topology Distribute are not applicable.
- This feature is applied to the smallest network topology node group of the switch layer level specified by the **create node_group switch_layer** or **set node_group switch_layer** subcommand of **qmgr(1M)**.
- If a JSV is specified to which jobs are assigned using job conditions, it is not guaranteed that the request will select the smallest network topology node group, since job conditions take precedence.
- If the order of jobs is important, please consider the order of job execution before turning this function on. Under certain conditions, it may be easier for a request submitted later to overtake a request submitted earlier. For example, if a group consists of 8 nodes and most of the requestss are 4-node jobs, then each time a request finishes, 4 nodes will be free. Therefore, a 4-node jobs submitted later may overtake a 5 or more node jobs. However, a request with 5 or more nodes will not start running later than the originally determined start time.

4.21.2 Setting

The target requests that uses Node group selection function for minimum network topology is set to a queue unit by "**set queue network_topology min_group**" subcommand of the **smgr(1M)** command. NQSV operator privileges or higher is required. The default value for a queue is off.

example:

```
# smgr -P o
Smgr: set queue network_topology min_group = on bq1
```

All requests submitted in bq1 are scheduled with Node group selection function for minimum network topology.

To find the minimum number of groups for a request, set the number of hosts to be executed per node group. The number of executing hosts per node group is set per queue by the **set queue network_topology host_per_group** subcommand of the **smgr(1M)** command. You must have operator or higher privileges to configure this setting.

example:

```
# smgr -P o
Smgr: set queue network_topology host_per_group = 512 execution_queue
```

For example, if the number of nodes per node group is set to 512, requests that use up to 512 nodes will always be assigned to one group. Even if there is a group with less than 512 nodes, or if the number of nodes is less than 512 due to failure or power saving factors, scheduling is performed assuming that each group has available 512 nodes.

It is recommended that you explicitly set the number of execution hosts per node group.

If not set, the default value is automatically selected the number of hosts to be used for each scheduling.

In the case of automatic selection by default, the number of execution hosts with the highest number of occurrences is selected among the number of execution hosts per network topology node group of the execution hosts to be scheduled that are BINDed in the request submission queue.

If you want to avoid the influence of the number of execution hosts changing due to failure encounters or queue configuration, explicitly set the number of execution hosts per node group.

The setting can be displayed by using **sstat(1)** with the **-Q -f** option.

```
#sstat -Q -f
Execution Queue: jmq0
  Queue Type      = Normal
  Schedule Time   = DEFAULT
  :
  Network Topology Control = {
    Network Topology Minimum Scheduling = ON
    Hosts per group = 4 (Default)
  }
:
```

The value of each request can be displayed by **sstat(1)** with **-f** option.

```
#sstat -f
Request ID: 1467.bsv0
  Request Name = batch job 1
  User Name = user1
  :
  Network Topology Control:
    Network Topology Minimum Scheduling = ON
    Hosts per group = 4 (Default)
    Jobs per host = 1 (Default)
:
```

4.22 FIFO Scheduling

The FIFO scheduling enables the execution of the requests in the same order of those submitted.

The system administrator can specify it with a subcommand of the **smgr(1M)** just like, **set queue schedule_type = fifo**. FIFO scheduling settings are per a queue.

JobManipulator immediately determines start time of the execution of the request submitted into the queue with the FIFO scheduling enables, if the resources for the request to run are available.

Also, if the start time of the previously submitted request is determined, the next request is scheduled, and the start time is determined, if necessary resources are available for execution.

Start time is not displayed for the request for which necessary resources are not available, due to the reservation section, the specified start time or the workflow. The start time can be displayed, when necessary resources are available.

Exceptionally a request submitted later overtakes the held request for the request connection, the request that is held by **qhold(1)** and the waiting request by **qalter(1) -a**.

When FIFO scheduling is applied, overtaking control on pickup is disabled. In addition, FIFO scheduling cannot be used in conjunction with the first stage-in time.

Both the queue that is set FIFO scheduling and the queue that is set Backfill scheduling must not have a same execution host if you use FIFO scheduling.

4.23 Cloud Bursting Function

4.23.1 Overview of Cloud Bursting Function

The cloud bursting function is a function that improves the TAT of a job by bursting the job to a computing resource in the cloud and executing it when there are more than a certain number of jobs waiting to be executed in the HPC cluster. The functional image is as follows:

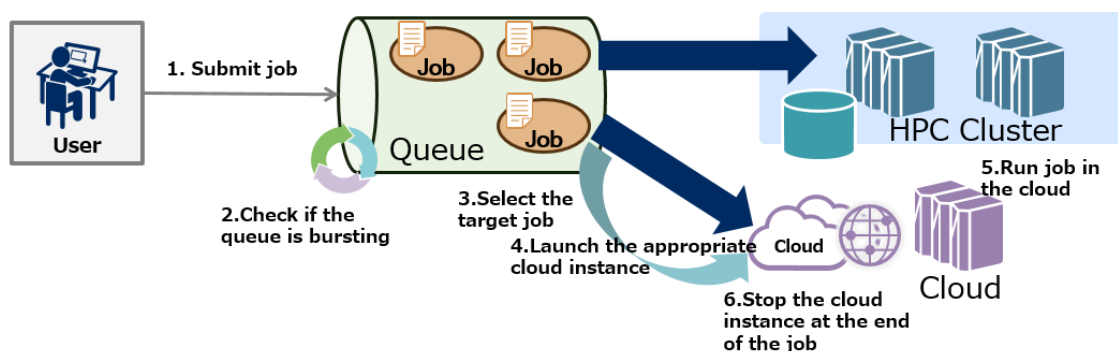


Figure 4-4 Functional image of cloud bursting

The overall flow of cloud bursting is as follows:

1. The user submits a request to the queue by specifying "Burst target possible".
2. NQSV decides whether to perform bursting according to the policy at each scheduling interval.
3. Calculate the bursting priority for each request to determine which requests to burst.
4. NQSV selects an image that is close to the requested amount of the request to be bursted and launches the cloud.
5. The request to be bursted is executed on the launched cloud environment.
6. NQSV will stop the cloud environment in sequence when the request for bursting is completed.

The cloud startup image is shown below.

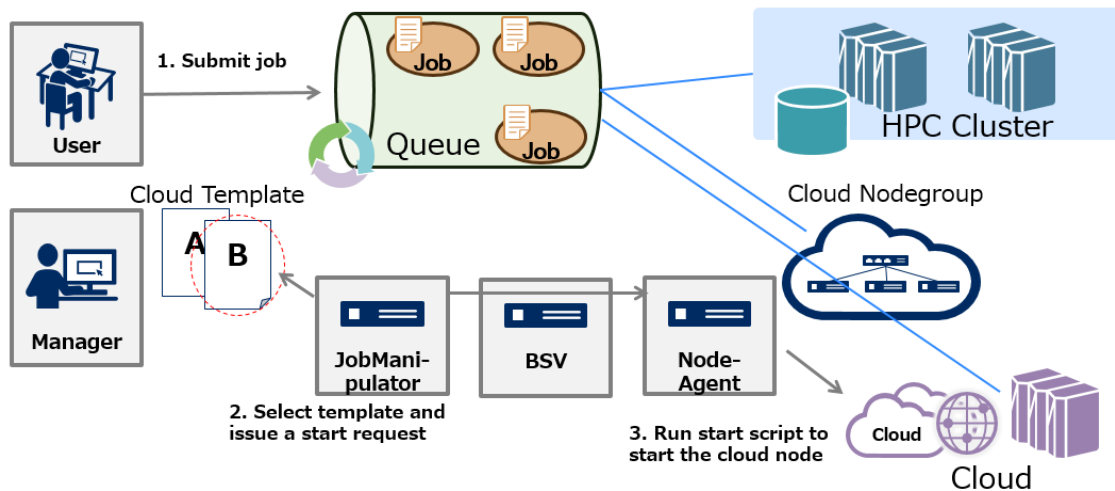


Figure 4-5 Image of cloud startup

The details of the cloud startup flow are as follows.

1. JobManipulator selects a cloud bursting template and cloud bursting node group for each request according to the policy and issues the required amount of cloud startup.
2. Node Management Agent launches the cloud via a shell script according to the requested cloud bursting template and cloud bursting node group information.
3. If the startup is successful, one JSV number will be assigned to one cloud node on BSV, and the execution host will be attached.
4. Since the cloud OS image has been created in advance so that JSV will start automatically, JSV will start automatically when the cloud instance starts.
5. When the JSV and BSV TCP link is established (JSV LINKUP), the queue bound to the specified cloud bursting node group and JSV are automatically bound.
6. The request will be scheduled and executed in the cloud launched with the selected cloud bursting template.
7. As soon as the request is complete, JobManipulator will request the cloud to stop. Node Management Agent will stop the cloud via a shell script according to the requested information.
8. When the stop is complete, the cloud node on the BSV and JSV will be automatically unbound from the queue and the execution host will be detached and unmanaged.

NQSV supports the following cloud environments.

- Amazon AWS (Amazon Web Service)
- Microsoft Azure
- Oracle OCI (Oracle Cloud Infrastructure)

4.23.2 Overview of Cloud Bursting Settings

This section describes the flow of settings for using cloud bursting.

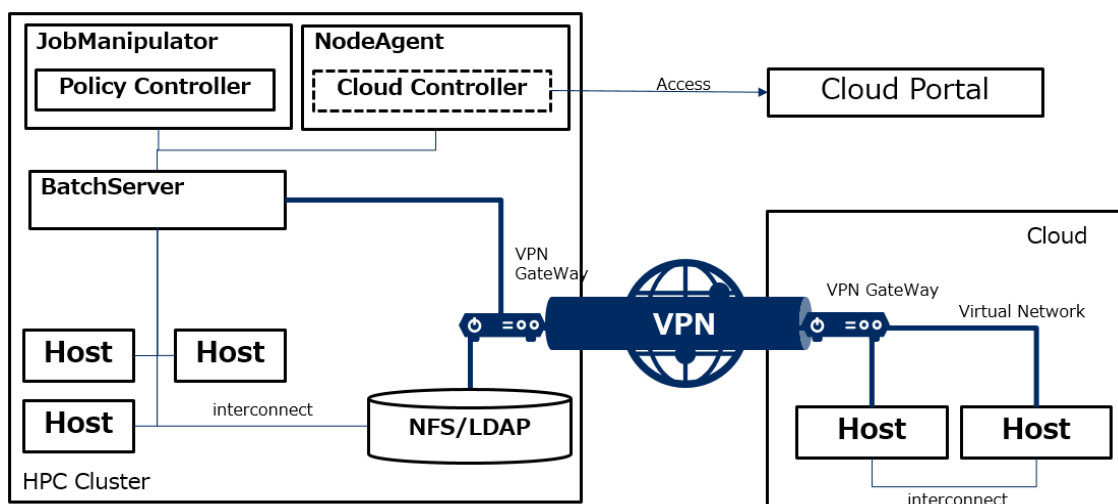


Figure 4-6 Cloud bursting configuration

Cloud bursting accesses the cloud portal via Node Management Agent and launches the cloud. Cloud Controller of the Node Management Agent can start, monitor, and stop the cloud with a shell script, so the administrator can start, monitor, and stop the cloud by customizing the shell script according to the cloud environment.

In order to perform cloud bursting, you first need to create a cloud bursting template and cloud bursting node group as shown in the figure below:

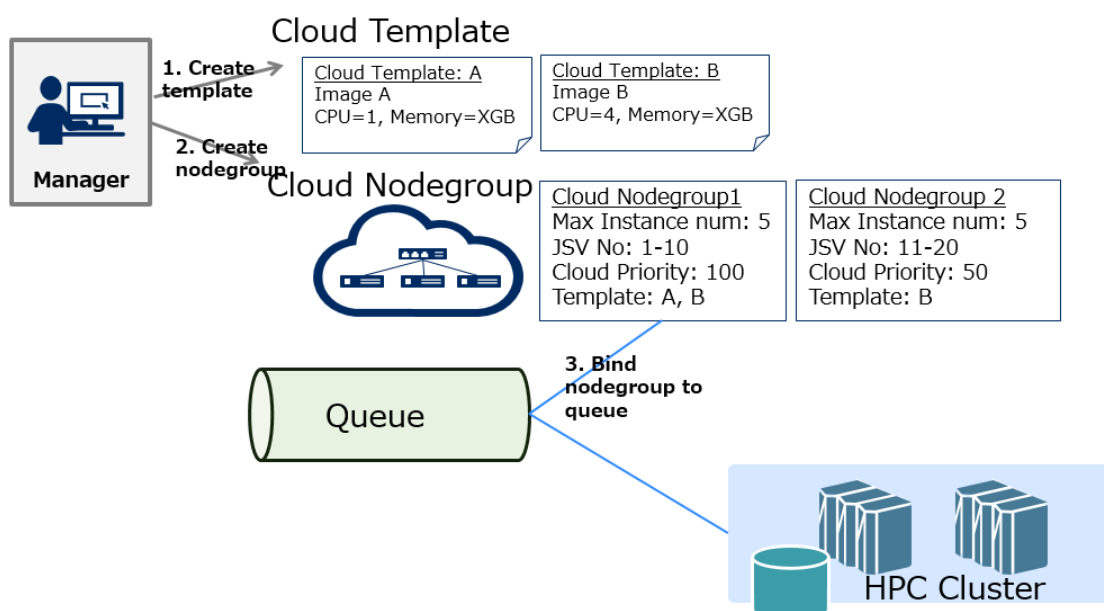


Figure 4-7 Template and node group settings

1. [Administrator] Create an image that can be used in the cloud environment as a **cloud bursting template**.

2. [Administrator] Create a **cloud bursting node group** that defines the cloud environment.
3. [Administrator] Bind the cloud bursting node group and the queue.

By submitting a request to the queue bound to the cloud bursting node group, JobManipulator decides whether to perform bursting according to the policy and selects the appropriate image from the templates added to the cloud bursting node group.

The queue can be an execution queue or an interactive queue.

The following sections explain the detailed setting method and job submission method for using the cloud bursting function.

4.23.3 Setting of Cloud Bursting Template

Information such as the OS image and resources of the instance to be started in the cloud environment is defined as a cloud bursting template (From now on, it is called a template).

By registering a template in the cloud bursting node group, NQSV automatically selects the template closest to the amount of job request resources and executes the job in the cloud environment indicated by that template. The user does not need to specify the template when submitting the request.

4.23.3.1 Defining a template

A template is defined by a system administrator. Multiple template can be defined. In a template, the following elements can be defined for each template as information of cloud environment.

Element name	Definition
Template name	<p>Template name.</p> <p>A name can consist of up to 47 characters.</p> <p>A space, double quotation mark ("), and @ symbol cannot be used in a template name.</p>
Image name (image)	<p>Image name of a cloud instance to start.</p> <p>This must be an image name that can be used to launch the cloud instance.</p> <p>This can consist of up to 127 characters.</p>
Number of CPUs (cpu)	<p>Number of CPUs to be assigned.</p> <p>This must be an integer of 1 or larger.</p> <p>This is also used as the limitation on the number of CPUs per job of the request for which the relevant template is specified.</p>
Memory size	<p>Memory size to be assigned.</p> <p>This must be an integer of 1 or larger followed by a unit (B, KB, MB,</p>

(memsz)	GB, TB, PB, EB). This is also used as the limitation on the memory size per job of the request for which the relevant template is specified.
Number of GPUs (gpu)	Number of GPUs to be assigned. This must be an integer of 0 or larger. Specify 0 if no GPU is assigned. The default is 0. This is also used as the limitation on the number of GPUs per job of the request for which the relevant template is specified.
Number of VEs (ve)	Number of VEs to be assigned. This must be an integer of 0 or larger. Specify 0 if no VE is assigned. The default is 0. This is also used as the limitation on the number of VEs per job of the request for which the relevant template is specified.
Estimated startup time (boot_timeout)	Timeout time until JSV LINKUP after starting the cloud instance. This must be an integer of 1 to 2147483647. The unit is seconds. The default is 300 seconds.
Estimated stop time (stop_timeout)	Timeout time to stop the cloud instance. This must be an integer of 1 to 2147483647. The unit is seconds. The default is 300 seconds.
Custom definition (custom)	If there is information uniquely defined as a startup environment, describe it. Up to 400 characters can be described.
Comment (comment)	Comments for a template can be described. Up to 255 characters can be described.

4.23.3.2 Using a template

(1) Creating a template

Start qmgr(1M) with the administrator privilege and use the "create cloud_template" subcommand below to create a new template.

```
create cloud_template=<template_name> image=<OS_image> cpu=<cpunum>
memsz=<memory_size> [gpu=<gpunum>] [ve= <venum>]
[custom="<custom_define>"] [comment="<comment>"] [boot_timeout=<timeout>]
[stop_timeout=<timeout>]
```

An example to create a template named AWS_MEDIUM_RHEL8 with the following settings configured is shown below.

- Image of the instance to start : ami-12ab34cd56ef78gh9
- Number of CPUs to be assigned : 8
- Memory to be assigned : 64GB
- Comment : For AWS medium instance

```
$ /opt/nec/nqsv/bin/qmgr -Pm
Mgr: create cloud_template=AWS_MEDIUM_RHEL8 image=ami-12ab34cd56ef78gh9 cpu=8
memsz=64gb comment="For AWS medium instance"
```

```
Cloud_Template AWS_MEDIUM_RHEL8 created.
```

(2) Deleting a template

Start qmgr(1M) with the administrator privilege and use the "delete cloud_template" subcommand below to delete the created template. However, if some requests use the target template, the template cannot be deleted.

```
delete cloud_template =<template_name>
```

(3) Editing a template

Start qmgr(1M) with the administrator privilege and use the "set cloud_template" subcommand below to edit the created template. However, if some requests use the target template, the template cannot be edited.

```
set cloud_template image=<OS_image> <template_name>
set cloud_template cpu=<cpunum> <template_name>
set cloud_template memsz=<memory_size> <template_name>
set cloud _template gpu=<gpunum> <template_name>
set cloud _template ve=<venum> <template_name>
set cloud_template custom="<custom_define>" <template_name>
set cloud_template comment="<comment>" <template_name>
set cloud_template boot_timeout=<timeout> <template_name>
set cloud_template stop_timeout=<timeout> <template_name>
```

(4) Locking and unlocking a template

A template can be locked to temporarily prevent use when the above "delete cloud_template" and "set cloud_template" subcommands are executed.

The locked template will no longer be selected when executing the request in the cloud environment. It has no effect on requests that are already running in the cloud environment.

For example, if you want to edit a template, first lock the target template, prohibit the use of the template for new requests, and then edit the template. After editing, unlock the template and make it available.

Start qmgr(1M) with the administrator privilege and use the "lock cloud_template" subcommand below to lock a template.

```
lock cloud_template =<template_name>
```

Start qmgr(1M) with the administrator privilege and use the "unlock cloud_template" subcommand below to unlock a template.

```
unlock cloud_template =<template_name>
```

4.23.3.3 Displaying a template

Use `qstat --template` to reference the information of the template defined in a system.

[Display example]

```
$qstat --template
[Cloud Template]
=====
Template  L Image      CPU  Memory GPU  VE  Custom          Comment
-----
AWS_MEDIUM - ami-12ab34    8  64.0G    0   0 (none)        For AWS medium insta*
```

Use `qstat --template -f` to reference the more detailed information of templates.

[Display example]

```
$qstat --template -f
Cloud Template: AWS_MEDIUM_RHEL8
Lock State   = UNLOCK
OS Image     = ami-12ab34cd56ef78gh9
CPU Number   = 8
Memory Size  = 64GB
GPU Number   = 0
VE Number    = 0
Boot Timeout = 300
Stop Timeout = 300
Custom       = (none)
Comment      = For AWS medium instance
```

4.23.4 Setting of Cloud Bursting Node Group

The cloud bursting node group (From now on, it is called a node group) is a node group required to start a cloud instance, and defines information such as the number of instances to be started (set within the range of Job Server numbers), the template for starting an instance, the network name of the cloud environment. If requests to able cloud bursting are submitted to the queue that the node group has been bound to, the cloud instances will be started with the template and Job Server number defined in the node group. Job servers are started on that instances and jobs are executed.

4.23.4.1 Create the node group

You can create the node group with `qmgr(1M)` command. You give "create node_group" sub-command the option "type=cloud".

The following is an example of how to create a node group named "cl_ng1". You must execute `qmgr(1M)` command with the manager privilege.

```
$ qmgr -Pm
Mgr: create node_group = cl_ng1 type=cloud
Node_group cl_ng1 created.
```

4.23.4.2 Add templates to the node group

At starting cloud instances the template that is added the node group is used. The node group requires the cloud bursting template. You can set the template to the node group at creating it. Or you can add the templates to the node group with "edit node_group add template" sub-command of **qmgr** (1M). A node group can have 20 templates. You must execute **qmgr** (1M) command to add the templates with the manager privilege.

```
$ qmgr -Pm
Mgr: edit node_group add template = AWS_MEDIUM_RHEL8 cl_ng1
Add template to Node_group (AWS_MEDIUM_RHEL8).
```

You can also add multiple templates at once, as following.

```
$ qmgr -Pm
Mgr: edit node_group add template =(AWS_MEDIUM_RHEL8.1,AWS_MEDIUM_RHEL8.2) cl_ng1
Add template to Node_group (AWS_MEDIUM_RHEL8.1).
Add template to Node_group (AWS_MEDIUM_RHEL8.2).
```

4.23.4.3 Delete templates from the node group

You can delete the added templates from the node group with "edit node_group delete template" sub-command of **qmgr** (1M). You must execute **qmgr** (1M) command to delete the templates with the manager privilege.

```
$ qmgr -Pm
Mgr: edit node_group delete template = AWS_MEDIUM_RHEL8 cl_ng1
Delete template to Node_group (AWS_MEDIUM_RHEL8).
```

You can also delete multiple templates at once, as following.

```
$ qmgr -Pm
Mgr: edit node_group delete template =(AWS_MEDIUM_RHEL8.1,AWS_MEDIUM_RHEL8.2) cl_ng1
Delete template to Node_group (AWS_MEDIUM_RHEL8.1).
Delete template to Node_group (AWS_MEDIUM_RHEL8.2).
```

4.23.4.4 Add Job Servers for instances to the node group

You add job servers to the node group so cloud instances to able to execute are added to the node group. The job servers as following cannot be added to the node group because they will be able to start in the cloud instances.

- Job servers belong to another node groups.
- Job servers have been already bound the queues.
- Job servers have been already attached to the execution hosts.

You cannot also add any node groups to the node group.

You can add job servers to the node group with "edit node_group add job_server_id" sub-command of **qmgr** (1M). How to use this sub-command is same case of common type node group.

4.23.4.5 Delete Job Servers for instances from the node group

You delete job servers from the node group so cloud instances to able to execute are deleted from the node group. In following case error occurs.

- These instances have jobs.
- These instances are starting or stopping.

In case of deleting multiple job servers if a job server is failed to delete, all specified job server are not deleted.

You can delete job servers from the node group with "edit node_group delete job_server_id" sub-command of **qmgr** (1M). How to use this sub-command is same case of common type node group.

4.23.4.6 Set priority to the node group

Some node groups can be bound to a queue for cloud bursting. When executing a request for cloud bursting, you can set which node group to prioritize in the node group priority. You can set the priority with "set node_group priority" sub-command of **qmgr** (1M) command. This sub-command requires operator privilege. The priority value is specified by a number from 0 to 63. The default value is 1.

```
$ qmgr -Po
Mgr: set node_group priority = 10 cl_ng1
Set priority to Node_group (cl_ng1).
```

4.23.4.7 Set network name to the node group

You can set the network name where the cloud instance started by the template set in the node group is located for each node group. Setting network name is not mandatory. If set, the information will be passed to the cloud instance startup script as an environment variable when the cloud instance is started. This setting allows you to launch cloud instances in different network environments in the same cloud (for example, different availability zone or different sub-network).

You can set the network name with "set node_group network" sub-command of **qmgr**(1M) command. This sub-command requires operator privilege. Maximum length of the network name is 255 characters.

```
$ qmgr -Po
```

```
Mgr: set node_group network = subnet-A cl_ng1
Set network to Node_group (cl_ng1).
```

4.23.4.8 Display node groups

qstat(1) command with "-G" option display status of node groups. If the node group is cloud type, the item "Type" is displayed as "cloud".

```
$ qstat -G
NodeGroup      Type      BatchServer    Comment      JSVs BindQueue
-----
cl_ng1         cloud     bsv1.example.co (none)      1 cloud_bq1
```

You can also display detailed information for each node group by adding the "-f" option to qstat(1) -G.

```
$ qstat -G -f
Node Group: cl_ng1
  Type = cloud
  Comment = (none)
  Bind Queue list = {
    cloud_bq1
  }
  Job Server number list = {
    100
  }
  Lock State = UNLOCK
  Priority = 1
  Instances = Live: 0 / Max: 1
  Network Name = (none)
  Template = {
    (none)
  }
```

Following items are only processed by cloud type node groups.

Priority: the priority to select the node group.

Instances: number of cloud instances. "Live" is number of instances currently running. "Max" is maximum number of the instances that can be started. "Max" equals number of job servers that are added to the node group.

Network Name: network name of the cloud.

Template: cloud template name to start cloud instances.

4.23.4.9 Lock/Unlock the node group

There may be cases where the instance cannot be started due to configuration errors. In this case you can lock the node group to disable it temporary. The locked node group isn't selected to start cloud instances. Already started cloud instances are not affected by locking the node group. The node group may be automatically locked by exit code of the cloud instance startup script, stop script, or monitoring script if an error occurs during the start, stop or execution of the cloud instance. Please refer to 4.23.5 for detail of the script.

You can lock the node group manually with "lock node_group" sub-command of **qmgr**(1M) command. This sub-command requires manager privilege.

```
qmgr -Pm
Mgr: lock node_group = cl_ng1
node_group cl_ng1 locked.
```

The locked node group isn't unlocked automatically. You can unlock the node group with "unlock node_group" sub-command of **qmgr**(1M) command. This sub-command requires manager privilege.

```
qmgr -Pm
Mgr: unlock node_group = cl_ng1
node_group cl_ng1 unlocked.
```

You can lock only cloud type node group. Common type or nw_topo type node group cannot be locked.

4.23.4.10 Delete the node group

You can delete the node group with "delete node_group" sub-command of **qmgr**(1M) command. This sub-command requires manager privilege. The node group that is bound to the queue cannot be deleted.

```
$ qmgr -Pm
Mgr: delete node_group = cl_ng1
Node_group cl_ng1 deleted.
```

4.23.5 Setting of Node Agent

4.23.5.1 Configure the node group to be managed

If you write names of the cloud bursting node group to the configuration file of Node Agent (/etc/opt/nec/nqsv/nag.conf), Node Agent manages them and can start, stop and monitor cloud instances. How to configure is following. Multiple names can be written by separating them with commas.

```
CLOUD_NGRP:<cloud_ngrp_name1>,<cloud_ngrp_name2>...
```

Node Agent must be restarted after you configure this. This configuration isn't applied by systemctl reload.

4.23.5.2 Cloud Instance Startup Script

It operates to start cloud instances by the script that is created by the system manager. This is allocated in /opt/nec/nqsv/sbin/cloud_prog/cloud_start.sh.

The script must operate following processes.

(1) Start cloud instances

The script creates and launches a cloud instance by executing CLI commands in the cloud environment. Information on the instance to be started are given by environment variables. The script is executed once per an instance.

(2) Wait for starting up to complete

The script waits for starting up to complete. The timeout period can be set freely by the script, but it cannot exceed the estimated startup time that the template has.

(3) Return startup result

The script writes the instance startup result to standard output.

The script is given following environment variables.

Environment variable	Value
NQSV_CLOUD_IMAGE	Image name for starting a cloud instance. This is OS image that selected cloud bursting template has.
NQSV_CLOUD_NETWORK_NAME	Network name of the cloud. This is network name that selected cloud bursting node group has.
NQSV_CLOUD_HOSTNAME	Host name of the cloud instance. This is one of the ID of job servers that were added to the selected cloud bursting node group. This value can be used as the host name of the instance.
NQSV_CLOUD_NAME	Name of the cloud. This is name of the cloud bursting node group that is selected.

The script writes the instance startup result as following format to standard output.
When startup is successful:

```
NQSV_INSTANCE_UP_RESULT:OK
NQSV_INSTANCE_ID:<instance-id>
NQSV_INSTANCE_IP:<ip-address>
```

The script writes above information to standard output one line at a time.

<instance-id> is identifier of the started instance. This will be used to stop this instance at terminating a job.

<ip-address> is IP address of the started instance. Either a public or private IP address is acceptable, as long as it is accessible from Batch Server host.

When startup fails:

```
NQSV_INSTANCE_UP_RESULT:NG  
  
NQSV_INSTANCE_UP_FAILSTOP:YES|NO  
NQSV_INSTANCE_UP_FAILLOCK: YES|NO
```

The script writes above information to standard output one line at a time.

NQSV_INSTANCE_UP_FAILSTOP indicates whether or not the instance needs to be stopped. Specify YES if stopping is required, and NO if stopping is not required. Normally, specify NO when some errors occur before launching the instance, and specify YES when some errors occur after launching the instance or when the launch timeout occurs.

If you specify YES, output the following information of the instance you want to stop line by line. If the following information is not output, the instance cannot be stopped. Set the IP address to a value within the valid range.

```
NQSV_INSTANCE_ID:<instance-id>  
NQSV_INSTANCE_IP:<ip-address>
```

NQSV_INSTANCE_UP_FAILLOCK indicates whether or not to lock the cloud bursting node group. Specify YES if locking is required, and NO if locking is not required.

4.23.5.3 Cloud Instance Monitoring Script

It operates to monitor cloud instances by the script that is created by the system manager. This is allocated in /opt/nec/nqsv/sbin/cloud_prog/cloud_watch.sh.

The script is executed by NQSV at regular intervals. The default interval is 60 seconds. The monitoring interval can be specified in the NQSV_CLOUD_WATCH_INTERVAL environment variable.

The script must operate following processes.

(1) Monitor cloud instances

The script monitors cloud instances by executing CLI commands in the cloud environment. Information on the instances to be monitored are given by environment variables.

(2) Return monitoring results

The script writes monitoring results of the instances to standard output.

The script is given following environment variable.

Environment variables	Value
NQSV_CLOUD_WATCH_INSTANCES	The cloud instance IDs to be monitored. Multiple IDs can be written by separating them with commas if multiple instances are monitored.

The script writes monitoring results of the instances as following format to standard output.

If no error occurs:

```
NQSV_INSTANCE_WATCH_RESULT:OK
```

The script writes above information to standard output.

If some errors occur:

```
NQSV_INSTANCE_WATCH_RESULT:NG  
NQSV_INSTANCE_WATCH_FAILLOCK:YES|NO  
NQSV_INSTANCE_ID:<instance-id>
```

The script writes above information to standard output one line at a time.

NQSV_INSTANCE_WATCH_FAILLOCK indicates whether or not the cloud bursting node group needs to be locked. Specify YES if locking is required, and NO if locking is not required.

<instance-id> is an identifier of the instance in which error occurred. If errors occurred in multiple instances, the script must write multiple lines for NQSV_INSTANCE_ID: <instance-id>.

4.23.5.4 Cloud Instance Stop Script

It operates to stop cloud instances by the script that is created by the system manager. This is allocated in /opt/nec/nqsv/sbin/cloud_prog/cloud_stop.sh.

The script must operate following process.

(1) Stop cloud instance

The script stops a cloud instance by executing CLI commands in the cloud environment. Information on the instance to be stopped are given by environment variables. The script is executed once per an instance. Also, make sure that the instance can be stopped in two modes: normal stop and forced stop. If an instance cannot be stop by normal stop mode, NQSV requires to stop the instance by forced stop mode.

(2) Wait for stopping to complete

The script waits for stopping to complete. The timeout period can be set freely by the script, but it cannot exceed the estimated stop time that the template has.

(3) Return stop result

The script writes the instance stop result to standard output.

The script is executed by following format. In forced stop mode case "force" as an argument is given.

```
/opt/nec/nqsv/sbin/cloud_prog/cloud_stop.sh [force]
```

The script is given following environment variable.

Environment variable	Value
NQSV_CLOUD_STOP_INSTANCE	Cloud instance ID to be stopped.

The script writes the instance stop result as following format to standard output.

When normal stop is successful:

```
NQSV_INSTANCE_DOWN_RESULT:OK
```

The script writes above information to standard output.

When normal stop fails:

```
NQSV_INSTANCE_DOWN_RESULT:NG
NQSV_INSTANCE_DOWN_FAILSTOP:YES|NO
NQSV_INSTANCE_DOWN_FAILLOCK:YES|NO
```

The script writes above information to standard output one line at a time.

NQSV_INSTANCE_DOWN_FAILSTOP indicates whether or not the instance needs to be forcibly stopped. Specify YES if stopping is required, and NO if stopping is not required.

NQSV_INSTANCE_DOWN_FAILLOCK indicates whether the cloud bursting node group needs to be locked. Specify YES if locking is required, and NO if locking is not required.

When forced stop is successful:

```
NQSV_INSTANCE_FORCEDOWN_RESULT:OK
```

The script writes above information to standard output.

When forced stop fails:

```
NQSV_INSTANCE_FORCEDOWN_RESULT:NG
NQSV_INSTANCE_FORCEDOWN_FAILLOCK:YES|NO
```

The script writes above information to standard output one line at a time.

NQSV_INSTANCE_FORCEDOWN_FAILLOCK indicates whether the cloud bursting node group needs to be locked. Specify YES if locking is required, and NO if locking is not required .

4.23.5.5 Sample Scripts

Sample scripts to start, to monitor and to stop for AWS, OCI and azure are installed on Batch Server host. Using these samples as a reference, system manager must create scripts that implement each of the above processes. and place them in the appropriate paths.

[Notes] These samples are provided to show a minimum implementation image of the functions, and isn't guaranteed to work in all cloud environments. When building the environment, please implement the appropriate processes according to the actual environment.

The summary of each sample script is as follows.

Sample cloud instance startup script

Installed path:

```
/opt/nec/nqsv/sbin/cloud_prog/aws_start.sh.sample
/opt/nec/nqsv/sbin/cloud_prog/azure_start.sh.sample
/opt/nec/nqsv/sbin/cloud_prog/oci_start.sh.sample
```

Summary of processing:

In AWS case:

The script start a cloud instance with the image identifier that environment variable NQSV_CLOUD_IMAGE has. Given environment variables are used as following.

NQSV_CLOUD_HOSTNAME: set as name tag of the instance.

NQSV_CLOUD_NETWORKNAME: set as sub-network identifier.

Type of the instance is t2.micro, availability zone is ap-northeast-1a.

The script monitors in 5 seconds intervals until status of the instance become running. Whole monitoring time is 300 seconds. If the instance doesn't become status running over 300 seconds, it is failed to start the instance.

If the instance starts normally, the script writes instance identifier and IP address of the instance to standard output. If the instance cannot start normally, the script write abnormal status to standard output.

NQSV doesn't give the script following information. The script specifies an appropriate value according to your actual environment.

- Key pair
- Security group identifier

In azure case:

The script start a cloud instance with the image identifier that environment variable NQSV_CLOUD_IMAGE has. Given environment variables are used as following.

NQSV_CLOUD_HOSTNAME: set as the computer name and VM name of the instance.

NQSV_CLOUD_NETWORKNAME: set as sub-network identifier.

Type of the instance is Standard_DS1_v2.

The script monitors in 5 seconds intervals until status of the instance become VM running. Whole monitoring time is 300 seconds. If the instance doesn't become status VM running over 300 seconds, it is failed to start the instance.

If the instance starts normally, the script writes instance identifier and IP address of the instance to standard output. If the instance cannot start normally, the script write abnormal status to standard output.

NQSV doesn't give the script following information. The script specifies an appropriate value according to your actual environment.

- Resource group

In OCI case:

The script start a cloud instance with the image identifier that environment variable NQSV_CLOUD_IMAGE has. Given environment variables are used as following.

NQSV_CLOUD_HOSTNAME: set as the host name and displaying name of the instance.

NQSV_CLOUD_NETWORKNAME: set as sub-network identifier.

Type of the instance VM.Standard.E2.1.Micro, availability domain is NTMs:AP-OSAKA-1-AD-1.

The script monitors in 5 seconds intervals until status of the instance become VM running. Whole monitoring time is 300 seconds. If the instance doesn't become status VM running over 300 seconds, it is failed to start the instance.

If the instance starts normally, the script writes instance identifier and IP address of the instance to standard output. If the instance cannot start normally, the script write abnormal status to standard output.

NQSV doesn't give the script following information. The script specifies an appropriate value according to your actual environment.

- Compartment identifier

Sample cloud instance monitoring script

Installed path:

```
/opt/nec/nqsv/sbin/cloud_prog/aws_watch.sh.sample  
/opt/nec/nqsv/sbin/cloud_prog/azure_watch.sh.sample  
/opt/nec/nqsv/sbin/cloud_prog/oci_watch.sh.sample
```

Summary of processing:

In AWS case:

The script checks the instances that are specified by instance identifier. If multiple identifier are given, the script checks all instances. If status of the instance is "running", the script determines that the instances are normal. In other case the script determines that the instances are abnormal.

The script writes status of monitoring to standard output. The script writes instance identifier to standard output if abnormal.

In Azure case:

The script checks the instances that are specified by instance identifier. If multiple identifier are given, the script checks all instances. If status of the instance is VM running, the script determines that the instances are normal. In other case the script determines that the instances are abnormal.

The script writes status of monitoring to standard output. The script writes instance identifier to standard output if abnormal.

In OCI case:

The script checks the instances that are designated by instance identifier. If multiple identifier are given, the script checks all instances. If status of the instance is "running", the script determines that the instances are normal. In other case the script determines that the instances are abnormal.

The script writes status of monitoring to standard output. The script writes instance identifier to standard output if abnormal.

Sample cloud instance stop script

Installed path:

```
/opt/nec/nqsv/sbin/cloud_prog/aws_stop.sh.sample  
/opt/nec/nqsv/sbin/cloud_prog/azure_stop.sh.sample  
/opt/nec/nqsv/sbin/cloud_prog/oci_stop.sh.sample
```

Summary of processing:

In AWS case:

The script stops and terminates the instances that are specified by instance identifier. If

"force" as an argument is given, the script directly terminates the instance without stopping.

The script monitors in 5 seconds intervals until status of the instance become stopping. Whole monitoring time is 180 seconds. If the instance doesn't become status stopping over 180 seconds, it is failed to stop the instance. If the status is already "stopped", "stopping", "shutting-down" and "terminated", the script has nothing to do for stopping the instance.

The script monitors in 5 seconds intervals until status of the instance become terminated. Whole monitoring time is 180 seconds. If the instance doesn't become status terminated over 180 seconds, it is failed to stop the instance.

The script writes status to stop the instance to standard output.

In Azure case:

The script stops and deletes the instances that are specified by instance identifier. If "force" as an argument is given, the script directly terminates the instance without deleting.

In stopping case the script monitors in 5 seconds intervals until status of the instance become VM stopped. Whole monitoring time is 180 seconds. If the instance doesn't become status VM stopped over 180 seconds, it is failed to stop the instance. If the status is already "VM stopped", "VM stopping", "VM deallocating" and "VM deallocated", the script has nothing to do for stopping the instance.

In deleting case the script monitors in 5 seconds intervals until status of the instance become terminated. Whole monitoring time is 180 seconds. If the instance doesn't become status VM deallocated over 180 seconds, it is failed to stop the instance.

The script writes status to stop the instance to standard output.

NQSV doesn't give the script following information. The script specifies an appropriate value according to your actual environment.

- Resource group

In OCI case:

The script stops and terminates the instances that are specified by instance identifier. If "force" as an argument is given, the script directly terminates the instance without stopping.

In stopping case the script monitors in 5 seconds intervals until status of the instance become STOPPED. Whole monitoring time is 180 seconds. If the instance doesn't become status STOPPE over 180 seconds, it is failed to stop the instance. If the status is already "STOPPED", "STOPPING", "TERMINATING" and "TERMINATED", the script has nothing to do for stopping the instance.

The script monitors in 5 seconds intervals until status of the instance become TERMINATED. Whole monitoring time is 180 seconds. If the instance doesn't become status TERMINATED over 180 seconds, it is failed to stop the instance.

The script writes status to stop the instance to standard output.

4.23.6 Setting of Job Server on the instance

NQSV/JobServer must be included in the OS image for the instance when the OS image is made to execute job server in the instance.

Job server to execute in the instance will have its number automatically assigned by Batch Server. Therefore nqs_shpd is started without job server number (option -n). The script (/opt/nec/nqsv/sbin/systemd_prog/nqs_jsv.sh) to start job server must be modified as following to start job server without its number. nqs_jsv.sh isn't modified as following at installing NQSV/JobServer.

before

```
#!/bin/sh
# NQSV JobServer Startup Script.

# Auto start config file (Don't change!)
JSV_START_CONF=/etc/opt/nec/nqsv/.jsv_start

PATH=/sbin:/usr/sbin:/bin:/usr/bin
export PATH

case $1 in
start)
    if [ "$JSV_NUMBER" = "AUTO" ]
    then
        if [ -f $JSV_START_CONF ]
        then
            cat $JSV_START_CONF | ¥
            while read line
            do
                set $line
                jsvno2=`printf "%04d" $1`
                /opt/nec/nqsv/sbin/nqs_shpd -h $2 -n $1 $JSV_PARAM
                if [ $? = 0 ]
                then
<snip>
            else
                for jsvno in $JSV_NUMBER
                do
                    jsvno2=`printf "%04d" $jsvno`
                    /opt/nec/nqsv/sbin/nqs_shpd -h $BSV_HOST_NAME -n $jsvno $JSV_PARAM >
/dev/null
                    if [ $? = 0 ]
                    then
                        RCNT=0
```

```
while [ $RCNT -lt 5 ]
do
```

after

```
#!/bin/sh
# NQSV JobServer Startup Script.

# Auto start config file (Don't change!)
JSV_START_CONF=/etc/opt/nec/nqsv/.jsv_start

PATH=/sbin:/usr/sbin:/bin:/usr/bin
export PATH

case $1 in
start)
    if [ "$JSV_NUMBER" = "AUTO" ]
    then
        if [ -f $JSV_START_CONF ]
        then
            cat $JSV_START_CONF | ¥
            while read line
            do
                set $line
                jsvno2=`printf "%04d" $1`
                /opt/nec/nqsv/sbin/nqs_shpd -h $2 $JSV_PARAM
                if [ $? = 0 ]
                then
<snip>
            else
                for jsvno in $JSV_NUMBER
                do
                    jsvno2=`printf "%04d" $jsvno`
                    /opt/nec/nqsv/sbin/nqs_shpd -h $BSV_HOST_NAME $JSV_PARAM > /dev/null
                    if [ $? = 0 ]
                    then
                        RCNT=0
                        while [ $RCNT -lt 5 ]
                        do
```

You can use `qstat(1)` command with option `-E` to check if the execution host is in the cloud or not. If the execution host is the cloud instance, "[C]" is displayed at the item "Execution Host". In `qstat -Ef` case "[Cloud]" is displayed at the item "Execution Host".

```
$ qstat -E host1
```

ExecutionHost	BatchServer	OS	Release	Hardware	VE	Load	Cpu
[C]host1	host0	Linux	4.18.0-477	x86_64	0	0.0	0.8

```

$ qstat -Ef host1
Execution Host: host1 [Cloud]
  Batch Server = host0
  Operating System = Linux (Rocky Linux release 8.8 (Green Obsidian))
  Version =
  Release = 4.18.0-477.15.1.el8_8.x86_64
  Hardware = x86_64
  Cloud Template = cl_tmpl
<snip>

```

You can also use `qstat -S` to check it. If the execution host is the cloud instance, "[C]" is displayed at the item "Execution Host". In `qstat -Sf` case "[Cloud]" is displayed at the item "Execution Host".

```
$ qstat -S 1000
```

JSVNO	JobServerName	BatchServer	ExecutionHost	LINK	BIND	Queue	Jobs	Load	Cpu
1000	JobServer1000	host0	[C]host1	UP	Y	bq, iq		1	0.0 0.0

```

$ qstat -Sf 1000
Job Server Name: JobServer1000
  Job Server Number = 1000
  Job Server Version = R1.08 (linux)
  Batch Server = host0
  Execution Host = host1 [Cloud]
  LINK Batch Server = UP
  BIND Queue = BIND
<snip>

```

The execution host and job server for the cloud bursting is automatically operated by NQSV. You cannot operate the following operations with **qmgr**(1M).

- Attach job server to the execution host (attach execution_host)
- Start job server (start job_server)
- Stop job server (stop job_server)
- Bind job server to the queue (bind execution_queue job_server / bind interactive_queue job_server)
- Unbind job server from the queue (unbind execution_queue job_server / unbind interactive_queue job_server)

You can detach job server from the execution host exceptionally. You can operate this to stop the cloud instance.

```
$ qmgr -Pm
Mgr: detach execution_host host=host1
```

4.23.7 Overview of Cloud Bursting Policy

The following policy is used to select the target request and cloud bursting node group.

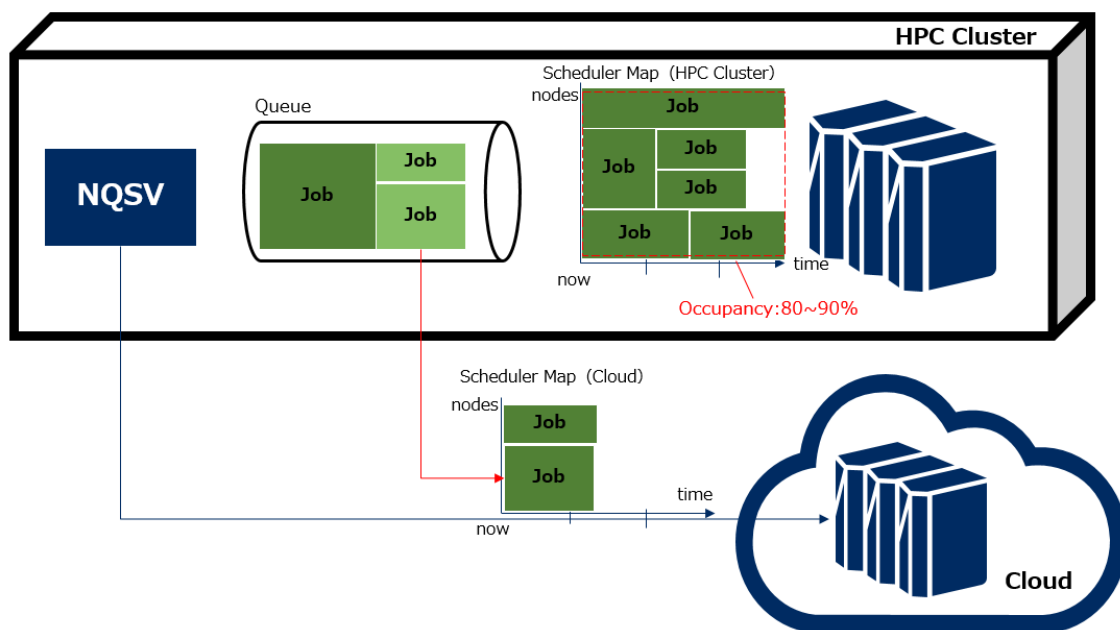


Figure 4-8 The policy of cloud bursting

1. Calculates the occupancy rate of the queue's scheduler map (HPC cluster) and checks if it exceeds the threshold at each scheduling interval.
2. If the occupancy of the queue's scheduler map (HPC cluster) exceeds the threshold, calculate the bursting priority of each request and determine the requests to be bursted.

3. Check the resources in the cloud bursting template set for each node group according to the priority of the cloud bursting node group bound to the queue, and select the most suitable node group and template for the target request for bursting, and start a cloud instance with the OS image of that template.
4. The request to be bursted is scheduled and executed on the cloud started with the selected OS image.
5. After all the requests are executed on the cloud, it will stop the cloud instances. Once all the cloud instances are stopped, the process will start from the beginning again.

The calculation of the occupancy rate is done at each scheduling interval. The next cloud bursting is performed after the request execution is finished and all cloud instances are stopped.

4.23.8 Setting of Scheduling

In order to bursting the submitted requests and execute them in the cloud, the following scheduling configuration is required.

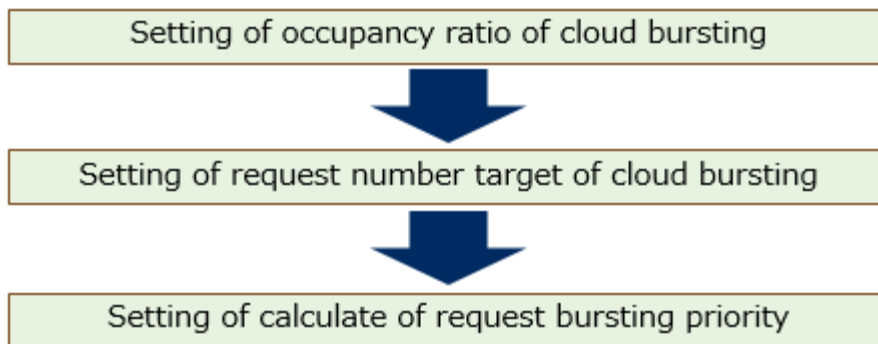


Figure 4-9 Setting of scheduling

4.23.8.1 Setting of occupancy ratio of cloud bursting

The threshold for cloud bursting when the scheduler map occupancy rate exceeds a certain percentage is set for each queue by the **set queue cloud_bursting occupancy_ratio** subcommand of the **smgr(1M)** command. The value is specified as a decimal number in the range of 0 to 1. 0 means that cloud bursting is always performed. 1 means that no bursting is performed. The default value is 1.

Scheduler map occupancy is the ratio of the sum of the resources assigned to a node for jobs, reserved sections, and planned power saving to the resources on the execution hosts (excluding execution hosts on cloud instances managed by NQSV) bound to the queue * map length. When the current scheduler occupancy exceeds the bursting threshold, the requests submitted to the queue can be bursted onto the cloud.

You can check the bursting threshold, the status of bursting or not, and the current scheduler map occupancy with **sstat -Qf** as follows.


```

$ sstat -Qf
Execution Queue: bq
:
  Elapse Margin          = 0S
  Cloud Bursting Occupancy Ratio = 1.000000
  Current Cloud Bursting Occupancy Ratio = 0.000000
  Current Cloud Bursting Status = DISABLE
  Display Cloud Bursting Priority = ON
:

```

4.23.8.2 Setting of request number target of cloud bursting

The number of requests to be selected for cloud bursting at a time is set by **set cloud_bursting_request_number** in the **smgr(1M)** command. The default value is 0 (not selected). The maximum number of nodes that can actually be used at one time is the number of instances defined in the cloud bursting node group. The number of requests should be the number of requests that can be executed within that number of nodes.

You can check the current setting values with the **sstat -Sf** command.

```

$ sstat -Sf
JobManipulator Server Host: bsv
:
  Device Group Topology = ON
  Cloud Bursting Request Number = 10
  Cloud Bursting Priority Weight = {
:

```

4.23.8.3 Setting of calculate of request bursting priority

When the occupancy rate of the queue's scheduler map exceeds the bursting threshold, the bursting priority of each request submitted to the queue is calculated. The number of requests to be bursted is selected in the order of increasing bursting priority.

The bursting priority is calculated for requests to be scheduled that do not have start time. It is not calculated for requests with scheduled start times, running requests, or requests that have jobs on HPC cluster nodes.

The bursting priority is calculated when the scheduler map occupancy threshold is exceeded, but it is always calculated if **set queue cloud_bursting display_bursting_priority** is set to on in the **smgr(1M)** command. The default value is off. Even if it is off, the bursting priority is calculated if the scheduler map occupancy threshold is exceeded. The bursting priority calculation is performed at each scheduling interval.

The bursting priority consists of four elements: requested resource, waiting time, scheduling priority, and custom resources. Each element can be assigned a weight. The

default value is that the bursting priority is calculated based on the waiting time only, and the longer the waiting time, the higher the priority.

The specific calculation formula is as follows

$$\begin{aligned} \text{bursting priority} = & (\text{requested resource elements (normalization)} + \\ & \text{waiting time elements} + \\ & \text{scheduling priority elements} + \\ & \text{custom resource 1 elements (normalization)} + \\ & [\text{custom resource 2 elements (normalization)...}] \\ & * \text{enable cloud bursting} \end{aligned}$$

If the bursting priority is less than or equal to 0, the request cannot cloud bursting. Note that a request may not be able to be burstable due to a negative value of scheduling priority or custom resource.

This section describes each element of the bursting priority.

1) Requested resource elements

The formula for calculating the requested resource elements is as follows

$$\begin{aligned} \text{requested resource elements} = & \\ & (\text{CPU number per request} * \text{elapse time (normalization)} + \\ & \text{GPU number per request} * \text{elapse time (normalization)} + \\ & \text{memory size per request} * \text{elapse time (normalization)}) \\ & * \text{weight for requested resource elements} \end{aligned}$$

The weight of the request resource element is set by **set cloud_bursting priority_weight_resource** in the **smgr(1M)** command. The default value is 0.

2) Wait time elements

The formula for calculating the wait time elements is as follows.

$$\begin{aligned} \text{wait time elements} = & (\text{now time} - \text{the request submitted time}) \\ & * \text{weight for wait time elements} \end{aligned}$$

The weight of the wait time element is set by **set cloud_bursting priority_weight_wait_time** in the **smgr(1M)** command. The default value is 1.

3) Scheduling priority elements

The formula for calculating the scheduling priority elements is as follows.

$$\text{scheduling priority elements} = \text{scheduling priority} * \text{weight for scheduling priority elements}$$

The weight of the scheduling priority elements is set by **set cloud_bursting priority_weight_scheduling_priority** in the **smgr(1M)** command. The default value is 0.

4)Custom Resource elements

The elements of a custom resource are calculated for each custom resource which defined. The weight is also set for each custom resource.

A custom resource whose unit of request is calculated as follows

$$\text{Custom Resource 1 element} = \text{The value Custom Resource 1 of the request} * \text{Weight for Custom Resource 1}$$

A custom resource whose unit of job is calculated as follows

$$\begin{aligned} \text{Custom Resource 2 element} = \\ \text{The value Custom Resource 2 of request} * \text{job number of request} \\ * \text{Weight for Custom Resource 2} \end{aligned}$$

The weighting of the elements of the custom resource is set for each custom resource by **set cloud_bursting priority_weight_custom_resource** in the **smgr(1M)** command. The default value is 0.

5)Enable cloud bursting elements

If the request disable cloud bursting, the bursting priority of the request is 0.

If **--enable-cloud-bursting=yes** is specified in **qsub(1)**, **qlogin(1)**, or **qcrsh(1)** when a request submitted, multiply the bursting priority by 1. If **--enable-cloud-bursting=no** is specified, or if it is not specified, multiply by 0.

You can use **sstat -Sf** to check the set value of weight for each element to calculate the request bursting priority.

```
$ sstat -Sf
JobManipulator Server Host: bsv
:
Device Group Topology = ON
Cloud Bursting Request Number = 10
Cloud Bursting Priority Weight = {
Resource              = 20
Wait Time             = 20
```

```

    Scheduling Priority = 30
    Custom Resource = {
        cr1 = 10
        cr2 = 20
    }
}
:

```

4.23.9 Submit a request

You give `--enable-cloud-bursting=yes` to `qsub(1)` command (if batch requests) or to `qlogin(1)` command or `qrsh(1)` command (if interactive requests) to submit cloud bursting available requests. Requests submitted with this option will be bounced to the cloud when they are ready to be bounced.

```

qsub --enable-cloud-bursting=yes
qlogin --enable-cloud-bursting=yes
qrsh --enable-cloud-bursting=yes

```

Requests that are submitted with `--enable-cloud-bursting=no` or without `--enable-cloud-bursting` will be never bounced to the cloud.

You can also use the `qalter(1)` command to change the bursting availability of requests that have already been submitted.

```

qalter --enable-cloud-bursting={yes|no} <rid>

```

If yes, the request will be changed to be available bursting on the cloud if necessary; if no, the request will be changed to never be bursting. `<rid>` is the request ID.

It doesn't take effect for requests that have already started executing.

You can check whether cloud bursting is enabled or disabled by using `qstat -f`. The **Enable Cloud Bursting** item will show whether it is enabled or disabled.

```

$ qstat -f 1.bsv
Request ID: 1.bsv
  Request Name = STDIN
  User Name = user
  Group Name = group
  User ID = 851
  Group ID = 701
  Current State = Running
  Previous State = Pre-running
  State Transition Time = Wed Apr 21 11:02:21 2021
  State Transition Reason = PRERUN_SUCCESS
  Queue = cloud@host0 (Execution Queue)
  Job Topology = Distribute Job

```

Request Priority = 0
Request Loglevel = 0
Rerunable = Yes
Holdable = Yes
Hold Type = (none)
Migratable = Yes
Suspend Type = (none)
Account Code = (none)
Stdout = host0:/home/user/STDIN.o1
Stderr = host0:/home/user/STDIN.e1
Reqlog = (none)
Shell = (none)
Mail Address = user@host0
Mail Option = (none)
Job Condition:
 Job NO: 0 ""
Number of Jobs = 1
Created Request Time = Wed Apr 21 11:01:03 2021
Entered Queue Time = Wed Apr 21 11:01:03 2021
Planned Start Time = Wed Apr 21 11:02:21 2021
Execute Request Time = (none)
Started Request Time = Wed Apr 21 11:02:21 2021
Ended Request Time = (none)
Requested Start Time = (none)
Deadline Time = (none)
UMASK = 022
Checkpoint Interval = 0
Restart File Directory = (none)
Reservation ID = (none)
qattach command = Enable
Attach = No
Cluster Type Select = NONE
Cloud Template = host1
UserPP Script = (none)
Exclusive = (none)
HCA Number = (none)
Accept Sigterm = No
Enable Cloud Bursting = Yes

```
Execution Hosts (JSVNO) :
    host1(1000)
<The following is omitted>
```

You can use `qstat -J` to check whether the jobs are actually running in the cloud or not. If they are running on the cloud, "[C]" will be added to the ExecutionHost item. In the detail view with `-f`, "[Cloud]" will be appended to the Execution Host item.

```
$ qstat -J 1.bsv
JNO RequestID      EJID  Memory    CPU JSVNO ExecutionHost  UserName Exit
-----
    0 1.bsv        14714  5.58M    0.00  1000  [C]host0        user    -

$ qstat -Jf 1.bsv
Request ID: 1.bsv
  Batch Job Number = 0
  Execution Job ID = 14714
  User Name = user
  User ID = 851
  Group ID = 701
  Job Server Number = 1000
  Job Server Name = JobServer1000
  Execution Host = host1 [Cloud]
  Exit Code = (none)
Resources Information:
<The following is omitted>
```

4.23.10 Policy of selection cloud

If the occupancy of the scheduler map exceeds the threshold, the bursting priority of each request is calculated, and the number of bursting target requests is selected in descending order of the bursting priority.

For each request, select the cloud that the request will execute from the cloud node group bound to the request submission queue. If multiple cloud bursting node groups are bound, select the cloud bursting node group in order of the cloud bursting node group's priority.

If there are multiple templates available in the cloud bursting node group, select the template with the closest resources per job or that request.

The following requests will not be selected for bursting regardless of the bursting priority value.

- Request with advance reservation(`qsub -y`)
 - Request with scheduled time(`qsub -s`)
 - Request with deadline time(`qsub -Y`)
 - Request which the preceding request specified `qsub --after` has not yet finished.
 - The number of request submitted in at a time with `--parallel` exceeds request number target of bursting.
 - Request with template of OpenStack or Container.
-

4.23.11 Forced Rerunning of Running Jobs for Cloud Bursting Requests

In a cloud computing environment, a network interruption may cause a linkdown of the job server. If the forced rerunning of running jobs (see **4.11.2 Forced Rerunning of Running Job** details) is set to **"on"**, the request will be rerunning every time there is a network interruption, and the cloud environment may stop and start up again.

Therefore, the forced rerunning of running jobs for cloud bursting requests is set differently from the forced rerunning of running jobs in the HPC cluster environment. **set forced_rescheduling = on** in the **set forced_rescheduling** command of the **smgr(1M)** command. In the **set forced_rescheduling** command of the **smgr(1M)** command, add **"cloud"** at the end like **"forced_rescheduling = on cloud"** to enable the setting only for cloud bursting requests. The default value is **off**, which means that cloud bursting requests will not be rerun due to the linkdown of the job server.

4.24 Request Assignment Mode

As described in **3.1.1 Scheduler Map 3.1.1.3 Notes on settings**, if there are too many requests, it is desirable to adjust the scheduler map and schedule interval to prevent requests from becoming backlogged.

This function is provided for cases where it is difficult to adjust the above parameters. This function allows the user to select whether or not to continue processing at the next scheduling interval, when processing has been terminated at this scheduling interval time. Note that this does not result in policy-aware scheduling.

This function can be set using the **set scheduling_method assign_mode** subcommand of the **smgr(1M)** command. The default value is **reset**, which means that if scheduling takes too long, it will stop after a certain amount of time; if it is set to **continue**, it will continue without terminating.

This setting can be confirmed in "Assign Mode" under "Scheduling Method" in `sstat -Sf`.

If set to **continue**, scheduling will not take into account policies such as scheduling priority at the current time. It will continue scheduling the request according to policies such as scheduling priority at the time of the scheduling interval in which the interruption first occurred.

4.25 Indication of the scheduled start time of execution during the scheduling process

When there is a large number of requests, scheduling may take a certain amount of time. During this time, it may take some time to refer to the scheduled start time of execution with the **sstat(1)** command.

In that case, please refer to the scheduled start time with the `--planned-start-time` option of the **qstat(1)** command.

If it is difficult to refer to the scheduled start time with the **qstat(1)** command, you can refer to the most recent scheduled start time with the **sstat(1)** command even during the scheduling process by using the following settings. This function is applicable to the **sstat(1)** command without options.

To enable this function, set the `FORK_WHILE_SCHEDULING` directive line in the NQSV/JobManipulator config file (`/etc/opt/nec/nqsv/nqs_jmd.conf`) to set the `FORK_WHILE_SCHEDULING` directive line. Then, specify the number of seconds from the start of the scheduling process until the scheduled execution start time can be referenced with the **sstat(1)** command.

The setting value can be specified from 0 to the scheduling interval -5 seconds.

After setting, restart NQSV/JobManipulator.

The following is an example of enabling the **sstat(1)** command to reference the scheduled execution start time when the scheduling process takes more than 30 seconds.

<code>FORK_WHILE_SCHEDULING:30</code>

-
- This function is available only when JM and the batch server to which JM is connected are operating on the same host.
 - If the **sstat(1)** command, **smgr(1M)** command, or **sushare(1M)** command with options are executed during the scheduling process, an error will occur.
 - When using this function, the **smgr(1M)** and **sushare(1M)** commands should be executed while all queues are stopped by the **qmgr(1M)** command.
-

4.26 Caching of non-schedulable requests

As described in **3.1.1 Scheduler Map 3.1.1.3 Notes on settings**, when there are too many requests, scheduling may not finish within the scheduling interval and processing may be terminated. If these requests continuously submitted by the same user with the same conditions and whose scheduled start time cannot be determined, Caching of non-schedulable requests function is effective.

This function is the scheduling results can be cached to increase the number of requests to be scheduled within the scheduling interval.

To enable this function, execute the following subcommand of the **smgr(1M)** command with operator or higher.

Smgr: set scheduling_method non_scheduled_request_cache = on
--

It has no effect if the scheduled start time has already been determined.

It has no effect if requests with different conditions or if they are not submitted consecutively.

Chapter 5. Functions for SX-Aurora TSUBASA

5.1 Overview

This chapter describes the functions for SX-Aurora TSUBASA of JobManipulator.

This function is available only for the environment whose execution host is SX-Aurora TSUBASA system.

5.2 VE Assignment Feature

When using VE, VE node number is specified by "--venum-lhost option" or "--venode option" of the qsub (1) command, the qlogin (1) command or the qrsh (1) command.

JobManipulator select the execution host (VI) to the request which requires VE nodes in order to satisfy required number of VE nodes.

5.3 Scheduling in VE Node Degradation

5.3.1 Overview of the Feature

In cases of change in the number of available VEs, such as failure and recovery of VE, you can select following Scheduling method.

(Such change of the number of available VEs is called VE node degradation)

1. "continue"
Schedule with the change in the number of VE node
2. "exclude"
Exclude VI with degraded VE nodes from the targets of scheduling
3. "auto"
Exclude VI with one or more degraded VE nodes from the target of scheduling. And when all of degraded VE nodes are recovered, the VI gets back to the target of scheduling automatically.

5.3.2 Feature of Setting of Scheduling Method at VE Degradation

This feature can be set per scheduler by using **set scheduling_method ve_degradation** subcommand of **smgr(1M)**. The operator privilege or higher is required for this setting.

The initial value is "continue". In this setting, JobManipulator schedules with the change in the number of VE node. When "exclude" is specified, JobManipulator excludes VI with degraded VE nodes from the targets of scheduling.

When "auto" is specified, if one or more VE nodes are degraded, the VI is excluded from scheduling. If all the degraded VEs are recovered, the VI will immediately return to the scheduling target.

The operation when the set value is changed is as shown in the table below.

Table 5-1 Operation of the scheduling method during VE node contraction

Before	After	Operation
continue	exclude	JobManipulator excludes immediately VI which degraded VE nodes.
continue	auto	JobManipulator excludes immediately VI which degraded VE nodes. If all the degraded VE nodes are recovered, the VI will immediately return to the scheduling target.
exclude	continue	VIs which are excluded from operation by degradation of VE nodes is returned to operation immediately.
exclude	auto	VIs that have been excluded from scheduling due to VE node degradation remain excluded. If all the degraded VE nodes are recovered, the VI will immediately return to the scheduling target.
auto	continue	VIs which are excluded from operation by degradation of VE nodes is returned to operation immediately.
auto	exclude	VIs that have been excluded from scheduling due to VE node degradation remain excluded. Even if all the degraded VE nodes are recovered, the VI concerned will not return to the scheduling target.

-
- This feature depends on "Load Interval" of BatchServer. This feature doesn't work if "Load Interval" is set 0. You need to set "Load Interval" to 1 or larger to detect increase and decrease of the number of VE nodes. The interval for updating the number of VE nodes is indicated by "Load Interval". Therefore, it will take some time for changing number of VE nodes to be reflected in the scheduling since it is slow to update number of VE nodes if "Load Interval" is set to a large value.
Please refer to *NQSV User's Guide [Management]* for "Load Interval" of BatchServer.
 - If auto is specified, the number of VE nodes is judged to be degenerated or restored by comparing it with the definition in the device resource definition file (/etc/opt/nec/nqsv/resource.def on the VI). In VIs where the device resource definition file does not exist, the behavior is equivalent to continue. For the device resource definition file, refer to **5.4 HCA Assignment Feature**.
-

5.3.3 Display by sstat

The setting can be displayed by using **sstat**(1) with the -S,-f option.

```
$sstat -S -f
JobManipulator Server Host: bsv.nec.co.jp
  JobManipulator Version   = R1.00
    :
  Stage-in Margin = {
    Additional Margin for Escalation = 0S
    Stage-in Threshold = 0S
    First Stage-in Time = 0S
  }
  Provisioning Start Retry Time = 0S
  Scheduling Method = {
    VE Degradation = Continue
  }
  :
```

The status of degradation of VE nodes can be displayed by using **sstat**(1) with "**-E --hw-failure**" option. Column "Status" shows status of VI and column "V" shows status of VE nodes have degraded in the past.

For "continue" settings, if VE nodes have degraded, "DEGRADED" is displayed at column "Status" and "D" is displayed at column "V".

```
$sstat -E --hw-failure
ExecutionHost  Status          V
-----
executionhost1 DEGRADED          D  <- under operation status of VI operation
with VE nodes degradation
```

For "exclude" or "auto" settings, if VE nodes have degraded, "EXCLUDED" is displayed at column "Status" and "D" is displayed at column "V".

```
$sstat -E --hw-failure
ExecutionHost  Status          V
-----
```

executionhost1	EXCLUDED	D	<- under exclusion status of VI operation with VE nodes degradation
----------------	----------	---	---

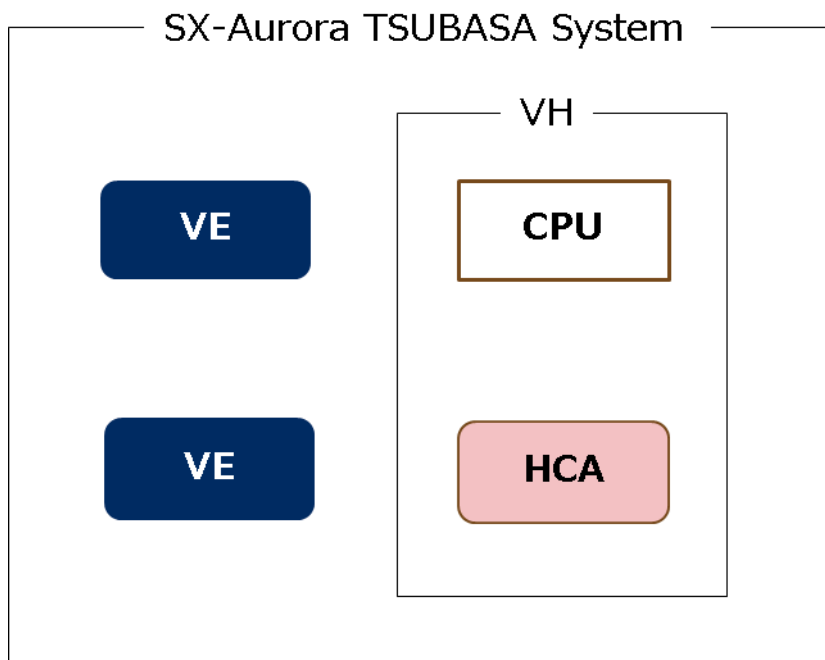
In order to reset VI degradation status you unbind JobManipulator from all bound queues and bind JobManipulator to the queues again.

5.4 HCA Assignment Feature

5.4.1 Overview of HCA Assignment Feature

Using the configuration below as an example, this section explains the SX-Aurora TSUBASA system that is used as an execution host.

Figure 5-1 SX-Aurora TSUBASA System



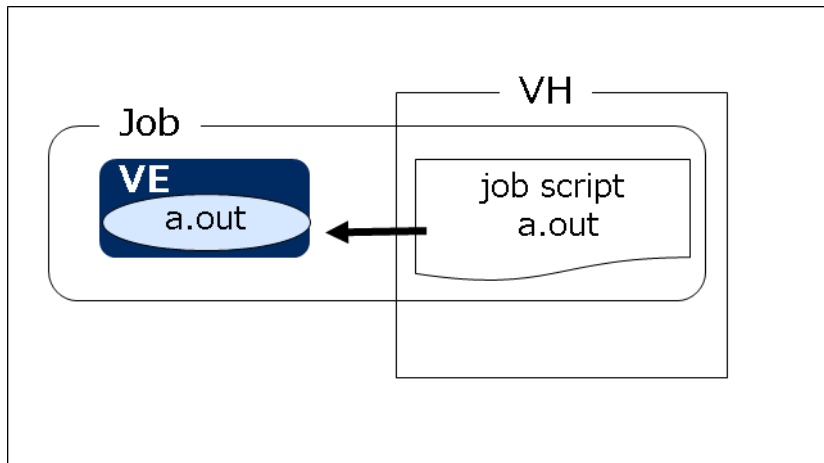
The vector engine (VE) is a core component of SX-Aurora TSUBASA and performs vector operation. The VE is a PCI Express card that is installed into an x86 server. The vector host (VH) is the x86 server(host computer) in which the VE is installed. Multiple VEs and an InfiniBand NIC (HCA) for communication between VEs may be installed in the VH depending on the VH model.

A host computer in which the VE is installed, the VE, and HCA are called a vector island (VI). It can be said that the VI and VH are the same for an execution host.

NQSV starts a job server and executes jobs on the VH. A program for the VE is run from a job script started on the VH. The VE and/or the HCA to run a VE program is assigned by NQSV. (In NQSV, the VE to be assigned to a job as a resource is called a VE node.) A VE program is run using the VE node assigned by NQSV.

The following shows an execution image of a VE program on the VH.

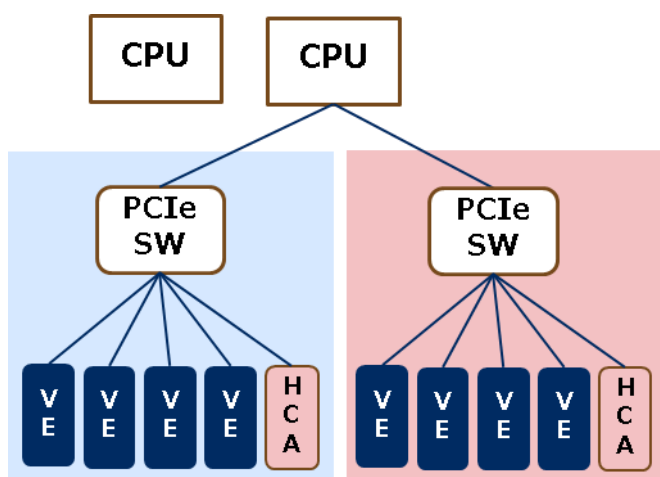
Figure 5-2 Execution of Program



Jobs can be executed with the appropriate VE node assigned to each job by inputting the `qsub(1)` command with the `--venode` (total number of VE nodes) or `--venum-lhost` (number of VE nodes per logical host) option specified into the queue bound with the VH execution host.

Depending on the SX-Aurora TSUBASA model, the topology configuration in the VH may be one in which the VE and HCA are connected to a CPU socket via a PCIe switch. The topology is the connection form of the CPU, VE and HCA. The following shows a topology configuration example.

Figure 5-3 Example of Topology Configuration



Administrators can define such topology configurations in advance, to enable NQSV to assign VE nodes and HCAs for jobs.

5.4.2 HCA and the Information of Topology

Administrators define use HCA per device and define topology information of CPU sockets, VE nodes and HCA in a file on execution host. This file is called a device resource configuration file. As use of HCA per devices, MPI (RDMA [Remote Direct Memory Access]) and I/O (Direct I/O) can be defined. Specifying multiple usages is also possible. It is not possible to change this file under operation. Restarting of JSV is needed at changing of this file.

VE and HCA connected to identical CPU socket and identical PCIeSW (CPU socket and PCIeSW connected) are grouped and it is called a “device group”.

5.4.2.1 Device Group

The examples of the device group are as follows.

Figure 5-4 Example of Device Group with PCIeSW

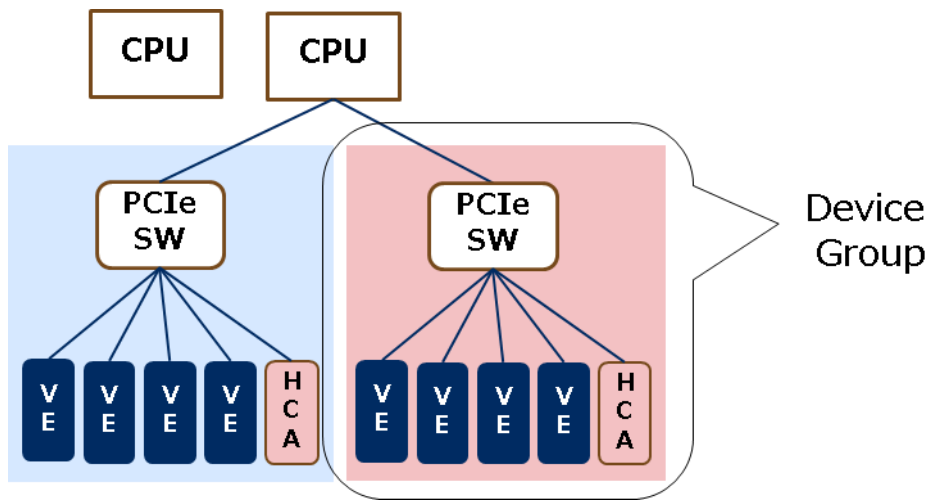
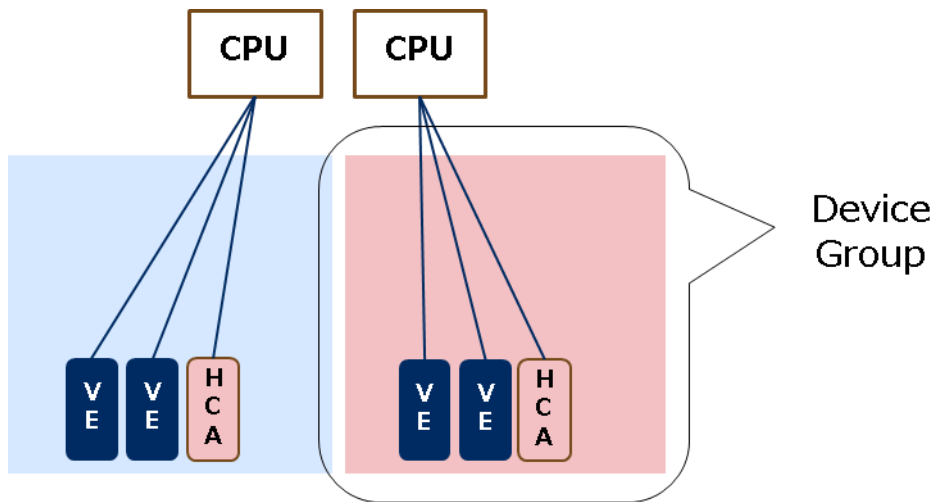


Figure 5-5 Example of Device Group without PCIeSW



5.4.2.2 Device Resource Configuration File

Device resource configuration file is /etc/opt/nec/nqsv/resource.def on the VI.

5.4.2.3 Format of the Device Resource Configuration File

Format of the device resource configuration file is as follows.

```
Format: <Resource>
      <Resource>: Resource information
Format:<Type> = { <List> }
```


<Type>: type of resource

Format: <Type> = Socket | PCIeSW | VE | Infiniband

The meaning of each character string are as follows

- Socket : CPU Socket
- PCIeSW : PCIeSW
- VE : VE node
- Infiniband:HCA

<List> : List of resource's detail. Nested descriptions of resource information express topology information.

Format: <Resource> | <Attribute>

<Attribute>: resource detailed information

Format:<Name> : <Value>

Possible resource detailed information for every <Type> is as follows.

All settings must be specified.

- Socket

<Name> : <Value>

Socket Number : socket number

- PCIeSW : PCI Switch

no resource detailed information

- VE

<Name> : <Value>

Number : physical VE number (It is possible to specify the range.)

- Infiniband

<Name> : <Value>

PCI ID : Identification number of PCI

Port Number : port number

Mode : use of HCA (IO and MPI can be specified. It is possible to specify multiple delimited by comma

IO : for direct communication of I/O

MPI : for direct communication of MPI

Both of capital letter and small letter are possible for setting character string.

Starting of JSV results in an error when one of the following condition is met.

- PCIeSW is defined as a resource outside Socket.
- VE and Infiniband is defined as a resource outside PCIeSW or Socket.
- The PCI ID which doesn't exist is specified.

5.4.2.4 Example of a Setting of the Device Resource Configuration File

Setting example of device group with PCIeSW when HCA is shared between IO and MPI is as follows. In this example 2 ports are installed to HCA and they can be referenced as independent HCA from VH.

```
Socket = {  
    Socket Number : 0  
}  
Socket = {  
    Socket Number : 1  
    PCIeSW = {  
        VE = {  
            Number : 0-3  
        }  
        Infiniband = {  
            PCI ID      : 0000:05:00.0  
            Port Number : 1  
            Mode        : IO, MPI  
        }  
        Infiniband = {  
            PCI ID      : 0000:07:00.0  
            Port Number : 2  
            Mode        : IO, MPI  
        }  
    }  
}  
PCIeSW = {  
    VE = {  
        Number : 4-7
```

```

    }
    Infiniband = {
        PCI ID      : 0000:0b:00.1
        Port Number : 1
        Mode        : IO, MPI
    }
    Infiniband = {
        PCI ID      : 0000:0d:00.1
        Port Number : 2
        Mode        : IO, MPI
    }
}
}
}

```

When PCIeSW is not included, the setting does not include PCIeSW's "{}".

```

Socket = {
    Socket Number : 0
    VE = {
        Number : 0-3
    }
    Infiniband = {
        PCI ID      : 0000:05:00.0
        Port Number : 1
        Mode        : IO, MPI
    }
}
}
Socket = {
    Socket Number : 1
    VE = {
        Number : 4-7
    }
    Infiniband = {
        PCI ID      : 0000:0b:00.1
        Port Number : 1
        Mode        : IO, MPI
    }
}

```

```
}
```

5.4.2.5 Display of the Setting Value of a Device Resource Configuration File

The setting of device resource configuration file can be displayed using `qstat(1)` command with `-F -f` options. In this case, the number next to the PCIeSW is ID of device group.

```
qstat -E -f
.....
Socket Resource Usage:
  NUMA Nodes = {
    Socket 0 (Cpus: 0-1) = Cpu: -/2 Memory: -/3.0GB
  }
Device Topology:
  Socket 0 = {
    (none)
  }
  Socket 1 = {
    PCIeSW 1 = {
      VE: 0-3
      HCA: 0000:05:00.0 0 (IO, MPI)
      HCA: 0000:07:00.0 1 (IO, MPI)
    }
    PCIeSW 2 = {
      VE: 4-7
      HCA: 0000:0b:00.1 0 (IO, MPI)
      HCA: 0000:0d:00.1 1 (IO, MPI)
    }
  }
}
```

When PCIeSW is not included, a part in PCIeSW is not displayed.

```
Device Topology:
  Socket 0 = {
    (none)
```

```

}
Socket 1 = {
    VE: 0-3
    HCA: 0000:05:00.0 0 (IO, MPI)
    HCA: 0000:07:00.0 1 (IO, MPI)
}

```

When there are no device resource configuration file following is displayed.

```

$ qstat -E -f
.....
Socket Resource Usage:
  NUMA Nodes = {
    Socket 0 (Cpus: 0-1) = Cpu: -/2 Memory: -/3.0GB
  }
Device Topology: (none)

```

5.4.3 Using HCA

5.4.3.1 Request Submission

You can submit a request specifying use for direct communication and number of HCA port using `--use-hca` option of `qsub(1)`, `qlogin(1)` and `qrsh(1)` command. In this case, the port number is the number necessary per device group to which VE belongs in logical host. You can specify `--use-hca` option in `#PBS` line in script. When `--use-hca` option and `--venode` option are not specified at the same time, submission error will occur. It will be also the same result is case of `--use-hca` option and `--venum-lhost` option.

The format of the `--use-hca` option is as follows.

```
format of <hca> : [<mode>:]<num>
```

`<num>` is the number of HCA port which is used by VE which is assigned to a logical host. Values in 0 to 32 can be specified. When specified value is beyond the range of value that can be specified or is not number submit error occurs.

`<mode>` is use the HCA. You can specify one of the following. If mode is not specified it is treated as "all". When a character string except the following is specified submit error occurs.

- `io` : I/O exclusive use

Only HCA that is specified IO in device resource configuration file is assigned.

- `mpi` : MPI exclusive use

Only HCA that is specified MPI in device resource configuration file is assigned.

- `all` : IO and MPI sharing use (initial value)

Only HCA that is specified IO and MPI in device resource configuration file is assigned.

It is possible to specify "io", "mpi" and "all" at the same time.

You cannot change the value of `--use-hca` by `qalter(1)` command.

[Example] When you submit a request that requires 4 VE and requires 1 HCA per device group to which belong VE.

```
qsub --venode=4 --use-hca=1 <script>
```

[Example] When you submit a request that requires 4 VE and requires 1 HCA that is I/O exclusive use and 1 HCA that is MPI exclusive use.

```
qsub --venode=4 --use-hca=io:1,mpi:1 <script>
```

[Example] When you submit a request that requires 2 VE per logical host, requires 1 HCA that is shared by IO and MPI, and requires 2 logical host.

```
qsub -b 2 --venum-lhost=2 --use-hca=1 <script>
```

5.4.3.2 *Display of the Information of a Request*

The information of the request which is submitted with `--use-hca` option can be displayed by using `qstat(1)` command with `-f` option.

[Example] When you submit a request that requires 4 VE and requires 1 HCA that is I/O exclusive use and 1 HCA that is MPI exclusive use.

```

$ qstat -f
.....
VE Node Number = 2
  HCA Number = {
    For I/O = 1    <- required number of HCA which is MPI exclusive use
    For MPI = 1    <- required number of HCA which is IO exclusive use
  }

```

Number of HCA which is required as IO and MPI sharing use is displayed as follows.

For ALL = <n>

When no HCA are required "HCA Number = (none)" is displayed.

5.4.3.3 Assignment of VE at using HCA

When a request is submitted with "--use-hca" option, VEs which belong to the same device group as much as possible are assigned to logical host.

However, it may not be always so because of emphasis of the request's TAT and the rate of operation.

[Example]

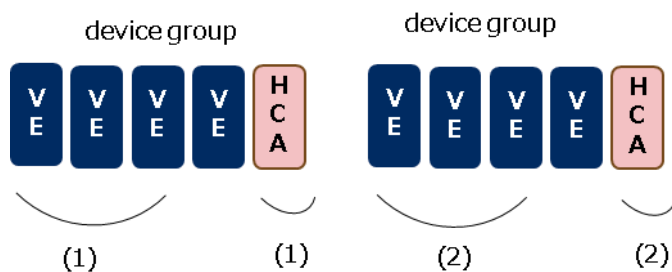
```

(1) qsub --venode=3 --use-hca=1
(2) qsub --venode=3 --use-hca=1

```

When the requests are submitted in numeric order, 3 VEs which belong to the same device group are assigned to "(1)", and 3 VEs which belong to another same device group are assigned to "(2)".

Figure 5-6 Assignment of VE at using HCA 1



Next, when a request (3) is submitted as follows:

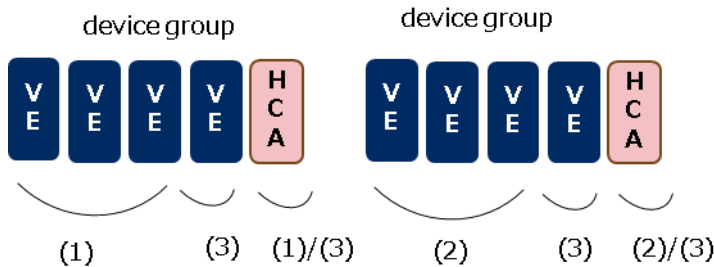
```

(3) qsub --venode=2 --use-hca=1

```

If there are no other empty VEs, 2 VEs which belong to different device group are assigned to "3".

Figure 5-7 Assignment of VE at using HCA 2



5.4.4 Topology information and HCA

VI without topology information is not target of scheduling of the request which is specified "--use-hca". The request which is specified "--venode" or "--venum-lhost" but is not specified "--use-hca" is target of this scheduling.

When VI with topology information and VI without topology information are mixed, VI without topology information is not target of scheduling of the request which is specified "--venode" or "--venum-lhost".

To maximize the execution performance, please bind VIs which have same topology configuration such as the numbers of CPU, VE, HCA and those connection form to a queue.

JobManipulator has following restriction about the request which uses topology information.

The request which is queued status at the timing of unbinding the JobManipulator from the queue, or rebooting JobManipulator or BSV, cannot be executed on the same execution host of other running request on that timing.

It start running after the end of a running request. Note that the running request contains the suspended request.

5.5 Scheduling with topology performance

By default, JobManipulator assigns jobs by packing ve nodes to the limit on the number of available VE nodes in the VI. You can minimize the number of VIs used because you do not assign jobs to other VIs until you exceed the limit on the number of available VE nodes. As a result, power saving effect can be expected.

If you want to assign ve node and HCA topology into account, you can do so by enabling scheduling with topology performance in mind.

5.5.1 Setting

The scheduling function for topology performance is set by the **set device_group_topology** subcommand of the **smgr(1M)** command. This needs operator or higher privilege. The default value is **off**, which does not emphasize the distance between ve nodes and HCA in VE or HCA assignments. If **on** is specified, ve and HCA are assigned taking into account the distance between the VE node and the HCA. For more information, see the 5.5.2 Operational with Topology Performance Considerations section.

In addition, when setting it to **off**, if the CPU concentration is enabled as the lower level policy, VEs in a VI, up to the limit of the available VE nodes, can be assigned to the request

In R1.05 or earlier this function was set by **VE_CONCENTRATION:** parameter in the configuration file. If there is **VE_CONCENTRATION:** parameter in the configuration file of R1.05 or earlier, this setting is automatically reflected at the first start JobManipulator after it is updated to R1.06 or later. After that, the **VE_CONCENTRATION:** parameter is ignored. Please set this function by **set device_group_topology** subcommand of **smgr(1M)** command.

5.5.2 Operation Considering Topology Performance

JobManipulator may assign VEs to some device groups in one logical host in order to emphasize turnaround time and the rate of operation for the request.

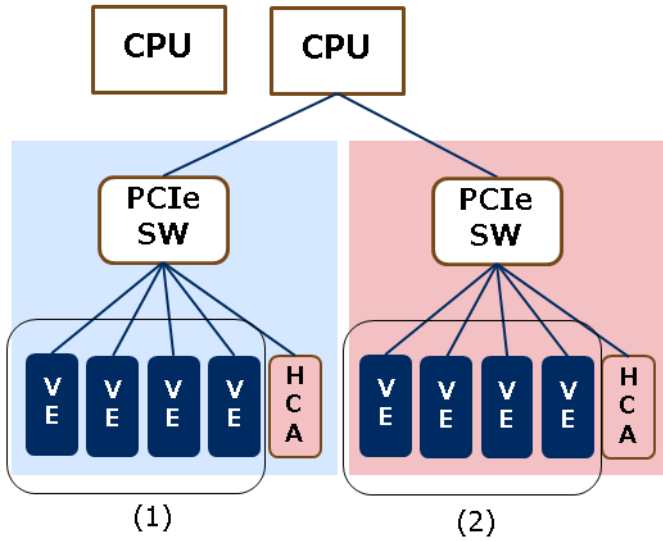
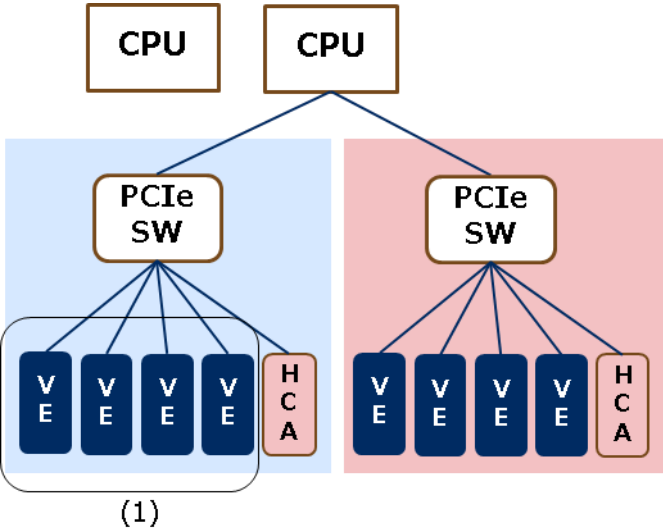
On the other hand, JobManipulator can also assign VEs to one device group in the logical host in order to emphasize topology performance. For it, all requests make require number of VEs that is multiplied VEs that one logical host has, regardless of the actual number of VEs used. As a result, VEs which are assigned to a logical host are included in one device group and you can always get a good performance.

[Example]

(1) qsub --venode=4 --use-hca=1
(2) qsub --venode=4 --use-hca=1

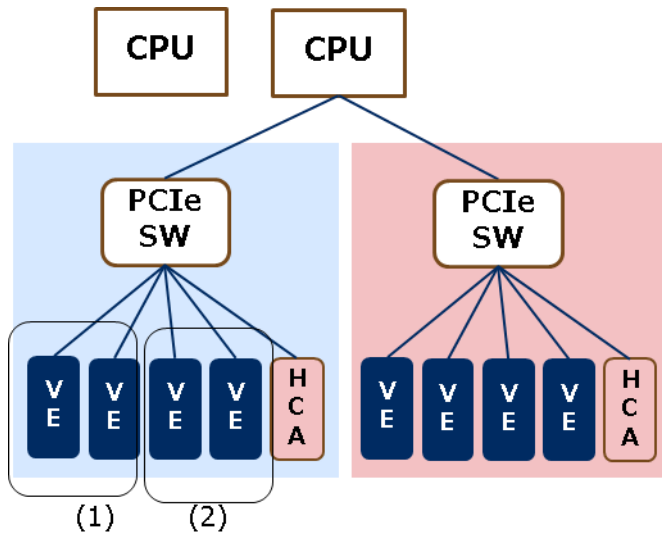
When the requests are submitted in numeric order, 4 VEs are assigned to logical host without dividing into device group and HCA closest from each VE are assigned. You can therefore use HCA that is good performance.

Figure 5-8 Example of the Operation Considering Topology Performance 1



If the number of VE which is included in a device group is 4, similar assignment is possible if all number of required VE is 2.

Figure 5-9 Example of the Operation Considering Topology Performance 2



5.6 Suspend Jobs Using VEs

It is able to swapping out the memory of the VE job to VH or storage, and the memory of the VE job can be returned to the VE from storage by using Partial Process Swapping (hereafter referred to as PPS) of VEOS. The feature for suspending the VE job is implemented by using this PPS feature.

See the SX-Aurora TSUBASA Installation Guide for information on how to configure the Partial Process Swapping feature in VEOS.

There are two ways to suspend using the PPS feature:

- The system administrator suspends a running VE job by using the **suspend request** subcommand of **smgr(1M)**
- NQSV suspends and interrupts normal requests which is using VE to execute an urgent/special request which uses VE.

If the execution host is SX-Aurora-TSUBASA, NQSV suspends the request by using the PPS feature by default. If you want to stop (SIGSTOP) process to suspend ve jobs without using the PPS function, the system administrator should set the **set execution_queue partial_process_swapping** sub-command of **qmgr (1M)** to **off**. It is configured to the queue that the request to be suspended is submitted. This configuration can be set for batch queues only. Interactive queue does not support this feature.

5.6.1 Suspend ve jobs with the 5.6.1 smgr(1M) command

The system administrator can suspend a running VE job by using **suspend request** sub-command of **smgr (1M)**. This operation requires prior configuration of the PPS function.

Requests that are suspended are resumed with the **resume request** sub-command of **smgr** (1M). The Elapse time stops during the request is in suspend status and start the measuring Elapse time again after the resume.

The request is rerun if the PPS function fails to swap out or swap-in when suspending or resuming with the **smgr**(1M) command.

To resume a request which is suspended by the **smgr**(1M) command, it requires there is no other running requests on the execution host that the request has been suspended. If running request exists, the resume may fail.

Multiple concurrent requests (requests submitted with the **--parallel** option of the **qsub**(1) command) running in a single execution host cannot be suspended with PPS by using **smgr**(1M) command.

5.6.2 Suspending VE Jobs to Run Urgent Requests

PPS function swap-outs the processes to suspend the normal requests which is using VE to execute an urgent/special request which uses VE. To do the urgent/special request wait for the completion of this swap-out and after that it starts running. After an urgent/special request finish the execution, the suspended normal requests are automatically swapped-in and resumed.

Special requests that suspends normal requests using VEs cannot be further suspended by the execution of urgent requests. The operation that suspends a request which is using VE, must be configured with either urgent request or special request.

5.6.2.1 Delete urgent requests by the lack of VE memory

There is a case that the VE memory that urgent request requires cannot be secured even if the processes of normal request are swapped-out. In this case, the default behavior is to rerun the interrupted normal request to execute the urgent request.

If you do not want to rerun the normal request in this case, set the **set execution_queue delete_failed_urgent_request** sub-command of **qmgr** (1M) to **on**. This configuration make interrupted normal request not to rerun and urgent request is deleted if the VE memory that urgent request requires are not available, an interrupted normal request is suspended first and then resumed after the urgent request has been deleted. Please configure this setting to the batch queue that urgent request is submitted.

5.6.2.2 Reduce of VE memory to swap-out

It is expectable to reduce the time for swapping-out by specifying the appropriate value to the limit on maximum VE memory size per VE node (`--vememsize-venode` option). For this purpose, all of the following conditions must be met.

- Version of both NQSV-ResourceManager and NQSV-JobServer is R1.14 or later.
- Version of VEOS is 3.1.1 or later.

In the case when these conditions are met

Please specify the value except "unlimited" to the limit on maximum VE memory size per VE node for the urgent request. This specification swaps out only the amount of memory required by the urgent request. It can be reduced to swap-out memory unnecessarily therefore It is expectable to reduce the time for swapping-out.

If "`--vememsize-venode`" option is not specified to the request, the default value of the queue submitted the request is applied to the request.

If "unlimited" is specified, all swappable VE memory is swapped-out regardless VE memory size required by the urgent request. It takes the same amount of time for swapping-out as before.

In the case when these conditions are not met

As in the past all swappable VE memory is swapped-out regardless VE memory size required by the urgent request. It takes the same amount of time for swapping-out as before.

5.7 Dynamic JSV Priority

5.7.1 Overview

As explained in **5.6 Suspend Jobs Using VEs**, when executing an urgent request that uses VE, the normal request is interrupted to temporarily free up VE memory. If the amount of VE memory required by the urgent request cannot be allocated, the interrupted normal request is rerun or the urgent request is deleted.

Enabling Dynamic JSV Priority allows node selection based on the status of running VE jobs when scheduling urgent requests that use VE. This enables the following

- Avoids nodes where VE jobs are running, reducing the number of interruptions to normal requests.
- It selects nodes that can release more VE memory in priority. Therefore rerunning and deleting of requests can be reduced.
- It selects nodes that has VE jobs with short elapsed execution time in priority. Therefore, reducing the impact of request reruns.

5.7.2 Settings

You can set Dynamic JSV Priority on a per queue. To enable this function, execute the following subcommand of the `smgr(1M)` command with operator privilege or higher. The default value is off.

If the queue type is set to a queue other than urgent, there is no effect.

`Smgr: set queue dynamic_jsv_priority control = on queue`

Dynamic JSV Priority is an assignment policy that has priority over JSV Assign Priority. See **3.1.8 Assign Policy** for details.

5.7.3 Calculation Method

The Dynamic JSV Priority value is the sum of the Priority values calculated by the five elements and up to 20 different custom resource elements. This value is calculated for each JSV when scheduling an urgent request. Details of the calculation method are as follows.

$$\text{Dynamic JSV Priority} = \sum \text{Priority}$$

The Priority value is calculated by the following formula.

You can choose to calculate the Priority value in ascending or descending order of element values.

Ascending order (asce)

Priority = ((Current Item Value * **max_item_priority**) / **max_item_value**) * **base**

Descending order (desc)

Priority = (**max_item_priority** - ((Current Item Value * **max_item_priority**) / **max_item_value**)) * **base**

base is the radix.

max_item_priority is the maximum value of Priority before multiplying by **base**.

Current Item Value is the value of the element at the time of scheduling.

For example, if a regular job using 5 VEs is running at the node, the Current Item Value is 5.

max_item_value is the maximum Current Item Value.

The value of Dynamic JSV Priority can be checked for each queue with the `-Q -f -j` option or the `-J --dynamic` option of the `sstat(1)` command. This is the value that was calculated when scheduling the urgent request. It may have changed from the situation at the time the command was executed.

5.7.4 Setting up calculation elements

The Priority calculation factor can be set on a per queue. Execute the set queue dynamic_jsv_priority item subcommand of the smgr(1M) command with operator privilege or higher. For details on the subcommand, see the reference section.

The Priority value is 0 by default.

Appendix.A Update history

14th edition

- Added 4.26 Caching of non-schedulable requests
- Added 5.7 Dynamic JSV Priority Extended

15th edition

- Modified 3.1.1 Scheduler Map
- Modified 4.6 Advance Reservation
- Modified 4.21 Node group selection function for minimum network topology

16th edition

- Added note to 4.14.2 Dynamic Power-saving Function.

17th edition

- Fixed 4.21 "set queue network_topology min_group" sub-command of the smgr(1M) command.
- Added 5.6.2.2 Reduce of VE memory to swap-out

18th edition

- Update qstat images in 4.23.6 Setting of Job Server on the instance.

Index

A

Adding Execution Queue to Complex Queue	29
Advance Reservation	81
Assign Limit	21
Assign Policy	68
Assign Pool	46

B

Backfill	47
Base-Up	62
Base-up defined by user	61, 62
Base-up for a request suspended by urgent request	60
Base-up for a rescheduled request	61
Basic Environment Architecture	3
BatchServerHost	2
BMC	v
BSV	iv

C

cell	43
change the scheduling feature	46
ClientHosts	2
Cloud Bursting Function	151
Command environment file	14
Configuration file	6
CPU number concentrated assign	37
Creating Complex Queue	28

D

Deleting Complex Queue	29
Deleting the Reserved Section	84
Device Group	189
Device resource configuration file resource.def	190
Display the Detail of the Execution Host Information	143
Display the Information of the Resource Reserved Section	85
Display the Information of the Resource Reserved Section (details)	88
Display the Setting of Elapse Unlimited	101
Dynamic Power-saving Function	124

E

Early Execution	32
-----------------------	----

Elapse Margin	65
Elapse Margin(Display format)	66
Elapse Margin(Setting method)	66
Elapse Unlimited Feature	100
Elapsed time	64
Escalation feature	32
Execution Hosts	2
Execution start time	63
Execution Time Reservation	80
Exit Delay Scheduling	73

F

Failover System	102
Feature of Setting of Scheduling Method at VE Degradation	184
FIFO Scheduling	150
First Stage-in Time	141
Forced Rerunning of Running Job	103
Formula of the Scheduling Priority	55
Forward escalation	32

H

HCA	v
HCA Assignment Feature	187

I

IB	v
Interrupting assign policy	69

J

JM	v
jmd.log	15
Job Assignment to the Resource Reserved Section	85
job condition	63
Job Condition	73
Job Submission to Reserved Section	83
JSV	v

L

limit of memory usage	65
Limits of the Number of CPUs that can be Executed Simultaneously	18, 20
Logfile	15

M

map width	43, 46
map width and request pick-up	46
Map Width Display Feature	49

Map Width Set Up	44
memory usage limit	65
merge rate	93, 98
MPI	v

N

NIC.....	v
Node group selection function for minimum network topology.....	146
normal queue.....	26
nqs_jmd.env.....	13

O

Operation Considering Topology Performance.....	199
Overtake Control at Pick-up	35

P

Pick-up	46
Power-saving Function	123
Provisioning with Docker	140
Provisioning with OpenStack.....	136

Q

queue type	26
------------------	----

R

Removing Execution Queue from Complex Queue	30
Request Assign Policy	37
Request Assignment Mode	181
Request Priority	58
Request Priority Order	25, 63
Request run limit	16
Reservation policy	82
Reserved Section Automatic delete.....	84
Reserved Section Delete by a command	84
Reserved Section ID.....	81
Resource balanced assignment	37
Resource Limit	59
Resource Reserved Section(Advance Reservation).....	81
RSG Limit of the usable ratio of CPUS.	23
RSG Limit of the usable ratio of memory size per RSG	23
Run Limit	15

S

Scheduled Power- saving Function.....	131
Scheduler logfile.....	14

Scheduler Map.....	43, 46
Scheduling by Software License of ISV software	136
Scheduling in Problem on Node	102
Scheduling in VE Node Problem	184
scheduling interval.....	43
Scheduling Parameter Setting	15
Scheduling Priority	54
Scheduling with the change in the number of CPUs.....	101
Scheduling with topology performance	198
Set Elapse Unlimited Feature	100
Set the Reserved Section.....	81
set weight coefficients of usage data to the scheduling priority	52
Setting of Complex Queue	30
Share distribution ratio configuration file	56
ShareDB.....	92
ShareDB Merge	92
Showing Complex Queue Information ..	31
Side escalation	33
special queue.....	26
Subcommands for Weight Coefficients..	61
Suspended Request	72
System Information Display	74

T

The number of CPUs that can be executed simultaneously per job.....	64
--	----

U

Unit Management	6
urgent queue.....	26
Urgent Request.....	75
usage data value	55
User Rank	57

V

VE.....	iv
VE Assignment Feature	184
VH.....	iv
VI	iv

W

Wait Time of Rescheduling	41
Waiting to Forced Rerunning on Start-up	103
Workflow	79

**NEC Network Queuing System V (NQSV)
User's Guide [JobManipulator]**

July 2024 18th edition

NEC Corporation

Copyright: NEC Corporation 2018

No part of this guide shall be reproduced, modified or transmitted without a written permission from NEC Corporation.

The information contained in this guide may be changed in the future without prior notice.