速石科技 | fastone 速石科技

fastone Scheduler

Fsched使用手册

上海速石信息科技有限公司 +86-21-31263638 www.fastonetech.com 上海 · 深圳 · 北京

目录

Introduction to Fsched

What is Fsched?

Fsched is Fastone's in-house HPC cluster resource management and job scheduling software, referred to as the scheduler. Fsched is pronounced /'ef,sked/.

Fsched can distribute computing workloads across heterogeneous, elastically scalable computing clusters, helping users complete computing jobs more efficiently on a shared computing cluster, while also helping enterprises improve resource utilization and reduce the business impact caused by single points of failure.

In high-performance computing (HPC) environments, an HPC scheduler is a key component responsible for managing and allocating computing resources, such as compute nodes, processor cores, and memory, to jobs waiting to run. The main functions of a scheduler include resource management, job scheduling, monitoring and reporting, and scheduling policy configuration.

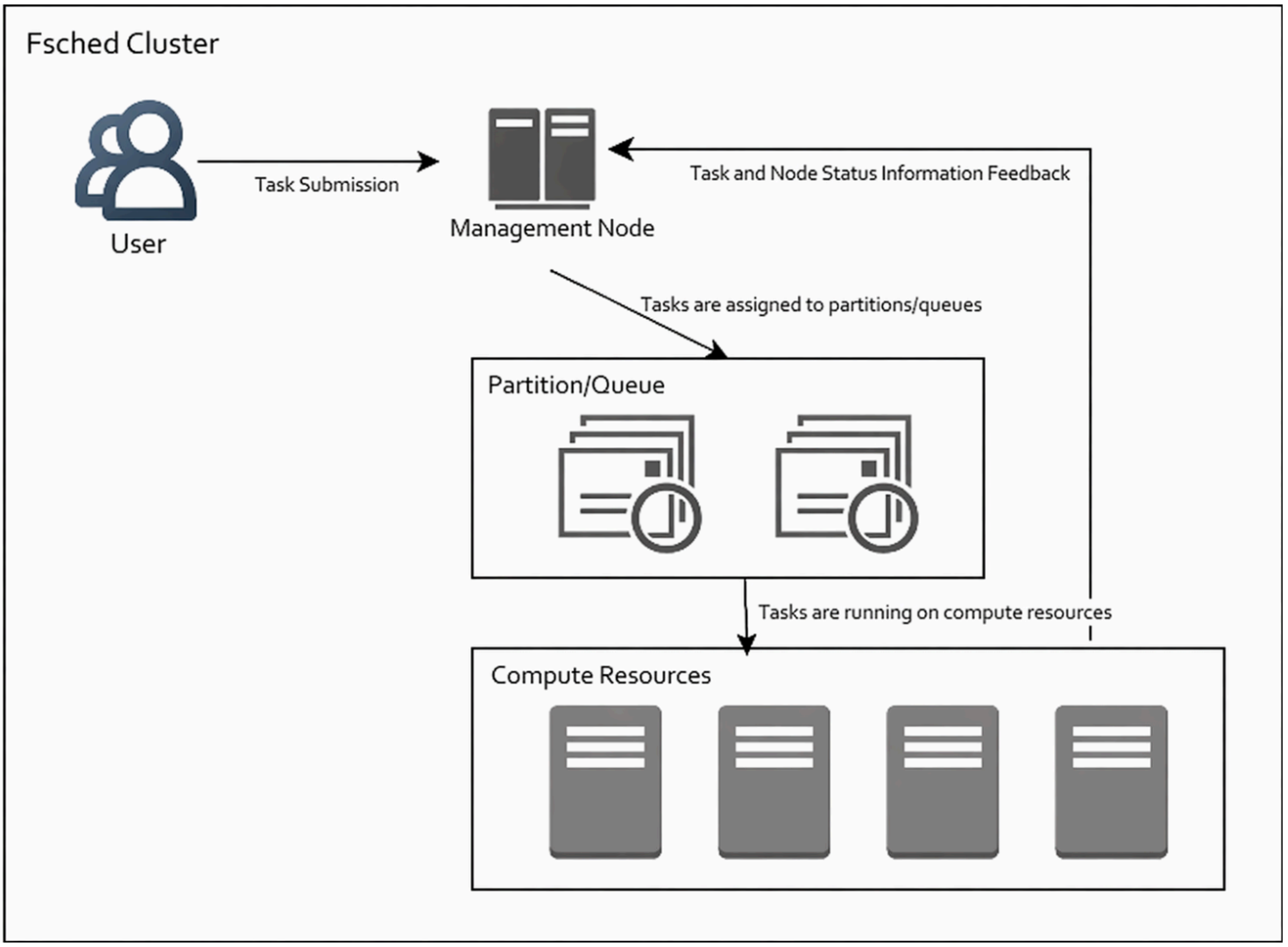
The Fsched scheduler has complete independent intellectual property rights and has been deployed in production environments across multiple industries, scales, and applications for many years. Its performance, ease of use, and stability have been validated in practice and have received extensive positive feedback.

Fsched Editions

Fsched is available in two editions:

- Fsched SE (Standard Edition): Unless otherwise specified, "Fsched" in this documentation refers to the standard edition of Fsched. Fsched is not released as a standalone product, but is built into other Fastone products, such as FCP and FCC-E.
- Fsched CE (Community Edition): Fsched CE is free for users and can be used freely on clusters with up to 6 nodes or 300 CPU cores. Fsched CE is supported through the community only and does not include commercial support.

Cluster Architecture



Key Concepts

Cluster

An Fsched cluster is a system composed of multiple nodes. These computers are connected through a network and work together to execute computing jobs. Fsched is used to manage these computing resources and ensure efficient job scheduling and resource allocation. A cluster can range in size from a few computers to thousands of computers.

Node

- Head node: The management node of an Fsched cluster, responsible for accepting user job submissions and managing node states and job states.
- Login node: On some HPC platforms, this is also called a submission node and is used only for submitting computing jobs.
- Compute node: The node on which user-submitted computing jobs actually run.

Partition

A partition is a logical grouping of jobs and nodes, used to divide jobs and nodes in a cluster into different subsets. Each partition can contain different types of nodes, allowing users to choose an appropriate partition for resource requests based on job requirements. Partition settings help manage resource allocation and access permissions, ensuring that different users or jobs can make effective use of cluster resources.

Depending on the administrator's configuration, nodes in different partitions can be either completely independent or overlapping. Jobs submitted to different partitions run at different priorities according to the configured scheduling policies.

In other HPC schedulers, a partition may also be referred to as a "queue" or "Queue".

Job

A job is a computing task submitted by a user to Fsched. It represents a set of resource allocation requests and process execution information. A job can contain multiple job steps and can run in parallel across multiple nodes. Job types include batch jobs and interactive jobs. Users can submit jobs with different commands, such as sbatch and srun, to obtain the required computing resources.

Inside Fsched

Fsched-Related Processes

Fsched is derived from the open-source Slurm scheduler. Based on Slurm 19.05, Fsched adds many improvements in functionality, performance, and stability. The daemons in an Fsched cluster include:

- `slurmctld`: the Fsched controller daemon, running on the head node. All scheduler management functions are provided by `slurmctld`. It is responsible for job queue management, scheduling, and monitoring the overall cluster state. It interacts with other daemons to allocate resources, start jobs, monitor job execution, and manage job dependencies.
- `slurmd`: the Fsched compute node daemon, running on all compute nodes. It manages local resource allocation, job execution, and communication with `slurmctld`. It ensures jobs run correctly on compute nodes, reports job status to `slurmctld`, and applies resource limits according to policy.
- `slurmdbd`: the Fsched database daemon, responsible for maintaining job accounting records in the database. It stores information about job submission, completion, failure, and resource usage. It supports queries and reporting, allowing users and administrators to track cluster usage and job history. In FCP or FCC-E environments, `slurmdbd` and its corresponding database service run on the FCP/FCC-E management node.
- `munged`: the authentication service used to create and validate credentials. It runs on all nodes in the Fsched cluster, including head nodes, login nodes, and compute nodes. It ensures communication between SLURM daemons and nodes is secure and trusted.
- `statesvc`: the Fsched state service, used to provide Fsched state and job information to the monitoring and analysis modules of FCP/FCC-E

How Fsched-Related Processes Communicate

- Registration and heartbeat: when `slurmd` starts on a compute node, it sends registration information to `slurmctld`, announcing its presence and providing the node's resource status. After that, `slurmctld` periodically sends heartbeat signals to each `slurmd` to confirm its state and availability.
- Request and response: when a user submits a job, `slurmctld` receives the request and schedules it based on the current resource state. After scheduling is decided, `slurmctld` sends the job information to the corresponding `slurmd`, which is responsible for executing the job on the local node.

- Monitoring and reporting: `slurmd` periodically reports node resource usage and job execution status to `slurmctld`, ensuring that the main control process has real-time visibility into the overall cluster state.

High Availability Design

Fsched's High Availability (HA) mechanism is designed to ensure that the cluster can continue operating when critical components fail, minimizing downtime and potential waste of computing resources. Fsched implements high availability in the following ways:

1. **`slurmctld`** failover

- Dual-controller configuration: Fsched supports running `slurmctld`, the control daemon, in a dual-controller mode. The cluster can be configured with one primary `slurmctld` and one backup `slurmctld`, usually running on different physical nodes.
- Automatic failover: When the primary `slurmctld` fails or becomes unavailable, the backup `slurmctld` automatically takes over and continues managing the job queue and cluster resources. This automatic failover mechanism ensures continuous cluster availability.
- Data synchronization: The primary `slurmctld` periodically synchronizes current cluster state information to the backup `slurmctld`, ensuring that the backup node has the latest state information during failover.

2. Job fault tolerance

Fsched is fault-tolerant and can continue operating in the event of node failures or network interruptions. By monitoring node health, Fsched can automatically detect failures and reschedule affected jobs, minimizing downtime. In extreme cases where the head node becomes completely unavailable, Fsched is designed so that jobs on unaffected compute nodes can continue running for up to 8 hours, maximizing business continuity in HPC environments.

3. **`slurmdbd`** high availability

- Dual-node configuration: Like `slurmctld`, `slurmdbd`, the SLURM database daemon, can also be configured in a primary-backup mode to ensure cluster job accounting data is not lost if the primary daemon fails.
- Database backup: Using a high-availability database management system, such as MySQL or MariaDB with primary-replica replication, can ensure that SLURM accounting data remains persistently available and avoid single points of failure.

4. Use of shared storage

- Shared file system: Fsched relies on a shared file system, such as NFS or Lustre, to store configuration files, state files, job scripts, and similar data. Using a shared file system ensures that when `slurmctld` fails over to a backup node, all nodes can access the same files and data.
- Cluster logs and state files: By storing logs and state files in shared storage, Fsched daemons can maintain consistency and availability across nodes.

User Authentication

Within an Fsched cluster, the best practice is to use an external authentication system, such as LDAP or NIS, to provide unified identity authentication for all nodes in the cluster. Fsched clusters use standard Linux users and groups.

Installation Requirements

Preparation Before Installation

- The installation package (not required in an FCP environment)
- At least one host that meets the requirements. In an FCP environment, this means at least one head node (management node), one compute node, and one login node.
- Network connectivity between all hosts used to install Fsched. Following common HPC practice, disabling all firewalls within the cluster is recommended.
- (Optional, recommended) Shared storage for data sharing between cluster nodes
- (Optional) Shared storage for head-node high availability

Node Requirements

Supported Architectures and Operating Systems

- x86: CentOS/RHEL 6/7, Rocky Linux 8, RHEL 8*, Ubuntu 18.04/20.04/22.04
- ARM: Amazon Linux 2*, UOS*, Kylin*
- LoongArch: Loongnix*, UOS*, Kylin*

*: Not publicly released yet. Contact the support team for more information.

Management Node Sizing

Cluster Size	Active Jobs	Minimum Memory (Recommended Memory)	Recommended CPU Cores
Small (1 - 50 nodes)	1,000	2 GB (16 GB)	2
	10,000	4 GB (32 GB)	2
Medium (50 - 500 nodes)	10,000	4 GB (32 GB)	4
	50,000	16 GB (64 GB)	8

Cluster Size	Active Jobs	Minimum Memory (Recommended Memory)	Recommended CPU Cores
Large (more than 500 nodes)	50,000	16 GB (128 GB)	8
	500,000	64 GB (256 GB)	16

Login Node and Compute Node Sizing

Fsched has no specific sizing requirements for login nodes and compute nodes. Any node that meets the operating system and processor architecture requirements can be used as an Fsched login node or compute node. Size compute nodes based on your workload requirements. For login nodes, decide the sizing based on actual usage, including but not limited to whether access is terminal-only or also provides remote desktops, whether the node is used only for job submission or also runs graphical applications, and whether it is dedicated to a single user or shared by multiple users.

Partition Planning

Partition planning is a critical step when building an Fsched cluster. It can be done during cluster deployment or after the cluster is built. Good partition planning can reduce the learning and operational cost for users, improve resource utilization, and shorten job wait times. Fsched administrators should refer to the relevant sections of the administrator guide when planning and configuring partitions.

Install Fsched SE

Because Fsched SE is built into FCP and FCC-E products, when users create an Fsched cluster through FCP or FCC-E, the installation and configuration of Fsched SE are completed automatically.

If necessary, refer to the following steps to install Fsched CE manually.

Manual Installation Steps

1. Install `fsched-{BUILDVERSION}`. Replace `{BUILDVERSION}` in the command with the version to be installed.

   ```
   sudo tar -xvf fsched-{BUILDVERSION}.tar.gz -C /opt
   ```

   ```
   sudo mkdir -p /var/{run,lib,log}/munge /etc/munge
   ```

   ```
   sudo /opt/fsched-{BUILDVERSION}/install.sh -t /usr/bin
   ```

   When the installation succeeds, the last line of output from the install script is:

   ```
   Successfully installed fsched from ...
   ```

   If not, the installation has failed. Check the failed step, correct the issue, and run the installation script again.

2. Restart the relevant services.

   i. On the head node, restart `slurmctld` and `fs-statesvc`.
   ii. On compute nodes, restart `slurmd`.

Other Notes

- Fsched supports mixing different versions within the same cluster, but with the following restrictions:

  - If head-node versions are inconsistent in HA mode, serious issues may occur.

    - If the head node and compute nodes use different versions, new features from later versions will be unavailable.

- Extracting to a non-standard directory

    i. Create a temporary directory.

    ```
    mkdir /tmp/fsched
    ```

    ii. Extract to the temporary directory.

    ```
    tar -xvf fsched-*.tar.gz -C /tmp/fsched ./opt
    ```

    iii. Copy the files under `/tmp/fsched/opt` to the corresponding directories.

Install Fsched CE

Fsched CE must be installed manually. This installation process uses Ansible-based scripts to quickly deploy an Fsched CE cluster.

Installation Steps

1. Confirm the installation prerequisites

   Confirm that the cluster nodes already have:

   - A shared home directory
   - Unified authentication
   - Time synchronization

   Confirm that the Ansible host already has:

   - Ansible installed
   - Access to the cluster nodes through the `root` user and an SSH key

2. Extract the Fsched remote installation package on the Ansible host

   Parameters:

   - `VERSION`: the version of the Fsched remote installation package

```
tar zxvf fsched-ce-${VERSION}-remote-install.tar.gz
cd fsched-ce-${VERSION}-remote-install/ce-install
```

3. Update the hostname and IP of each cluster node in `inv.yml` based on your cluster information

   Example:

```
vi inv.yml
```

```
head:
    hosts:
        partition-qhkjh-1:              # specify hostname of head
            ansible_user: root
            ansible_host: 10.0.10.2     # specify ip address of head
```

```
compute:
    hosts:
        partition-qhkjh-2:              # specify hostname of compute 1
            ansible_user: root
            ansible_host: 10.0.10.4     # specify ip address of compute
1
        partition-qhkjh-3:              # specify hostname of compute 2
            ansible_user: root
            ansible_host: 10.0.10.3     # specify ip address of compute
2
```

4. Run the installation script

   Parameters:

   - `SSH_KEY_PATH`: the path to the SSH key file used to log in to the cluster nodes
   - `SOURCE_PATH`: the full path of `fsched-ce-${VERSION}.tar.gz` under the `fsched-ce-${VERSION}-remote-install` directory

```
ansible-playbook --private-key ${SSH_KEY_PATH} --inventory inv.yml --
extra-vars "source_path=${SOURCE_PATH}" fsched-ce.yml
```

Upgrade Fsched

Scope

This document applies to the following scenarios. Follow the instructions for your scenario:

- Existing clusters
- FCP platform
- FCC-E (image)

> DANGER
>
> - Installing the same Fsched version already used by the current cluster may affect running jobs.

Steps

Existing Clusters

1. Copy the installation package (`fsched-*.tar.gz`) to the target machine.
2. Install `fsched-{BUILDVERSION}`. Replace `{BUILDVERSION}` in the command with the version to be installed.

   ```
   sudo tar -xvf fsched-{BUILDVERSION}.tar.gz -C /opt
   ```

   ```
   sudo /opt/fsched-{BUILDVERSION}/install.sh -t /usr/bin [-r]

   -r: Optional parameter. Attempts to restart services automatically.
   Effective in fsched-10.61 and later.
   ```

   When the installation succeeds, the last line of output from the install script is:

   ```
   Successfully installed fsched from ...
   ```

If not, the installation has failed. Check the failed step, correct the issue, and run the installation script again.

3. If step 2 does not restart services automatically, restart the relevant services manually.

    i. On the head node, restart `slurmctld` and `fs-statesvc`.

    ii. On compute nodes, restart `slurmd`.

4. If HA is configured, upgrade the standby node first and then the primary node.

FCP Platform

1. For the FCP platform, upgrade existing clusters by following the steps in Existing Clusters.
2. Copy the installation package (`fsched-*.tar.gz`) and overwrite `/opt/components/fsched.tar.gz`. This ensures that newly added nodes receive the updated Fsched component.

Image (FCC-E Only)

1. Create a virtual machine from the console.
2. Copy `fsched-*.tar.gz` to the virtual machine.
3. Install it by following the installation steps.
4. Create an image and register it with the API according to the image update procedure.

Other Notes

- Fsched supports mixing different versions within the same cluster, but with the following restrictions:
    - Head-node versions must be identical in HA mode.
    - If the head node and compute nodes use different versions, the head-node version must be newer than the compute-node version.
    - New features in later versions are unavailable on older nodes.

Extracting to a Non-Standard Directory

Each Fsched installation package contains its version number and is typically located at `/opt/fsched-xxx` (where `xxx` is the version). The path actually in use is `/opt/fsched`, which is a symbolic link to a specific version. Therefore, it is safe to extract over an installation package that is not currently in use, but it is not safe to extract over a version that is currently in use.

1. Create a temporary directory.

```
mkdir /tmp/fsched
```

2. Extract to the temporary directory.

```
tar -xvf fsched-*.tar.gz -C /tmp/fsched ./opt
```

3. Copy the files under `/tmp/fsched/opt` to the corresponding directories.

Access the Cluster

To submit compute jobs to an Fsched cluster, the user must already be connected to one of the cluster nodes (often the login node).

Common ways to access the cluster include:

- Via SSH: The user logs in to the Fsched cluster's login node via SSH and submits jobs from that node.
- Via desktop: Start a VNC session on the login node and submit jobs in the desktop environment provided by the VNC session.

Submit Compute Jobs

Before submitting a job, please understand two concepts: interactive jobs and batch jobs.

As the name implies, an interactive job allows users to interact with the job in real time while it runs. This type of job is similar to a remote session that logs directly into a compute node, where you can enter commands, view output, debug programs, and so on while the job is running. A batch job runs in the background and does not require real-time user interaction. Users write all instructions in a script file in advance and submit that script to run automatically on the specified resources.

Submit an Interactive Job

Example:

```
srun --ntasks=1 --cpus-per-task=4 --mem=4G --time=01:00:00 --pty bash
```

This command requests 1 task, 4 CPUs, 4 GB memory, 1 hour of runtime, and starts an interactive bash shell.

Submit a Batch Job

Example:

Suppose there is a batch script named job_script.sh with the following content:

```
#!/bin/bash
#SBATCH --job-name=example_job
#SBATCH --output=output.txt
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=4
#SBATCH --mem=4G
#SBATCH --time=01:00:00

module load my_software
srun my_program arg1 arg2
```

You can submit the job with the following command:

```
sbatch job_script.sh
```

Fsched will automatically schedule the job when resources are available and save the output to the output.txt file.

Manage Compute Jobs and Resources

squeue View queued or running jobs

Use the `squeue` command to get the current job status. If the job you want is not shown in the squeue output, it means the job has already exited.

In the output, the `ST` column is the job status. The status codes mean:

- R: Running
- PD: Pending
- CG: Completing
- S: Suspended

Common `squeue` command and option combinations are as follows:

Function	Command Example
Show the status of all jobs in the queue	`squeue`
View job information for job ID 11	`squeue -j 11`
View job information for user user1	`squeue -u user1`
View jobs submitted to partition01	`squeue -p partition01`
View jobs using node compute01	`squeue -w compute01`
View jobs in Pending state	`squeue --state=PENDING`
View detailed info for a job with custom output	`squeue -j 11 -o "%.18i %.9P %.8j %.8u %.2t %20V %.10M %.6D %R %Z"`
View detailed info for a partition with custom output	`squeue -p partition01 -o "%.18i %.9P %.8j %.8u %.2t %20V %.10M %.6D %R %Z"`

Other options can be viewed with `squeue --help`.

sinfo View partition information

The main function of `sinfo` is to view status information for partitions and nodes. Common command and option combinations are as follows:

Function	Command Example
Show status of all partitions in the cluster	`sinfo -Nl`
Show usage of a specified partition	`sinfo -p partition01`
Show detailed usage of a specified partition	`sinfo -p partition01 -N -o "%20N %15C %.5a %.6t"`

Node status meanings in `sinfo` output:

- alloc: Node is allocated
- drain: Node is drained/unresponsive; no new jobs will be assigned in this state
- idle: Node is idle
- mix: Node has partial resources allocated
- comp: Node is releasing resources; nodes in other states are unavailable

Example

```
[root@login1 ~]# sinfo -N -o "%20N %15C %.5a %.6t"
NODELIST       CPUS(A/I/O/T)   AVAIL  STATE
compute01      0/4/0/4            up   idle
compute02      0/4/0/4            up   idle
compute03      0/4/0/4            up   idle
```

In the second column `CPUS(A/I/O/T)`, A = CPUs used by jobs, I = idle CPUs, T = total CPUs on the node.

Common sinfo options

```
--help    # Show help for the sinfo command;
-d        # Show non-responsive nodes in the cluster;
-i <seconds>    # Refresh partition/node output every N seconds
-n <name_list>    # Show specified node(s); separate multiple nodes with
commas;
-N    # Display one line per node;
-p  # <partition> Show specified partition(s); separate multiple partitions
with commas;
-r    # Show only responsive nodes;
-R    # Show reasons for node issues;
```

Output in a specified format;

```
-o #<output_format>    Show specified output. The format is %[[.]size]type.
"." means right alignment; omitted means left alignment. size is the field
width; type is the item to display. Common items include:
%a Availability state
%A Show node counts as "allocated/idle"; do not use with "%t" or "%T"
%c Number of cores per node
%C Total cores as "allocated/idle/other/total"
%D Total number of nodes
%E Reason a node is unavailable
%m Memory per node (in M)
%N Node name
%O CPU load
%P Partition name; the default partition is marked with "*"
%r Only root can submit jobs (yes/no)
%R Partition name
%t Node state (compact form)
%T Node state (extended form)
```

scancel Cancel running or queued jobs and view job status

The `scancel` command can cancel running or pending jobs in the queue.

Common commands and parameter examples:

Function	Command Example
Cancel job ID 11	`scancel 11`

Function	Command Example
Cancel job named test-001	`scancel -n test-001`
Cancel jobs submitted to partition01	`scancel -p partition01`
Cancel pending jobs	`scancel -t PENDING`
Cancel jobs running on node compute01	`scancel -w -n compute01 -t RUNNING`

Other parameter options can be viewed with `scancel --help`.

Common scancel options:

```
--help                # Show help for the scancel command;
-A <account>        # Cancel jobs for the specified account; if no job_id
is specified, cancel all;
-n <job_name>        # Cancel jobs with the specified job name;
-p <partition_name> # Cancel jobs in the specified partition;
-q <qos>            # Cancel jobs with the specified qos;
-t <job_state_name> # Cancel jobs in the specified state, "PENDING",
"RUNNING" or "SUSPENDED";
-u <user_name>        # Cancel jobs for the specified user;
```

sacct View historical job information

The `sacct` command can view historical job start/end time, end status, job ID, job name, number of nodes used, node list, runtime, and more.

Example

View runtime information for a job:

```
sacct -j 29
```

The output includes: job ID, job name, partition, billing account, requested CPU count, status, and exit code.

```

```

```
[root@head ~]# sacct -j 9
       JobID    JobName  Partition    Account  AllocCPUS      State
ExitCode
------------ ---------- ---------- ---------- ---------- ---------- -------
-
9                 sleep partition+ _fsched_a+          1  COMPLETED
0:0
9.extern         extern            _fsched_a+          1  COMPLETED
0:0
9.0               sleep            _fsched_a+          1  COMPLETED
0:0
```

You can add output parameters to view detailed job information, for example:

```
[root@head ~]# sacct -j 9 -X -o
jobid,jobname%50,user,group,partition,submit,start,end,state,alloccpus,reqmem
       JobID                                              JobName      User
Submit               Start                 End      State  AllocCPUS     ReqM
WorkDir
------------ -------------------------------------------------- --------- ---
-------- ------------------- ------------------- ---------- ---------- ------
--------------------------------------------------
9                                                         sleep      cyan
03T00:25:02 2024-09-03T00:25:02 2024-09-03T00:26:02  COMPLETED          1
/fastone/users/cyan
```

View job history and runtime information since a specific time:

```
[root@head ~]# sacct -X -T -S2024-08-10-11:00:00 -o
jobid,jobname,user,partition,submit,start,end,state,alloccpus,reqmem,elapsed,
       JobID    JobName      User  Partition              Submit
End      State  AllocCPUS     ReqMem    Elapsed ExitCode              WorkDir
------------ ---------- --------- ---------- ------------------- ------------
---------- ---------- ---------- ---------- ---------- -------- ------------
2              hostname shaobing+ partition+ 2024-07-31T23:13:41 2024-07-31T2
31T23:13:42  COMPLETED          1        1Mn   00:00:01      0:0 /fastone/use
3             big_task1 cadservi+ partition+ 2024-08-01T06:47:18 2024-08-01T0
01T06:49:29 CANCELLED+          6        1Mn   00:02:10      0:0 /fastone/use
4             big_task1 cadservi+ partition+ 2024-08-01T06:53:05 2024-08-01T0
01T06:53:45 CANCELLED+          6        1Mn   00:00:40      0:0 /fastone/use
5             big_task1 cadservi+ partition+ 2024-08-01T22:04:58 2024-08-01T2
```

```
01T22:06:09 CANCELLED+          6        1Mn   00:01:10      0:0 /fastone/use
6             big_task1 cadservi+ partition+ 2024-08-01T22:08:04 2024-08-01T2
01T22:08:19 CANCELLED+          2        1Mn   00:00:14      0:0 /fastone/use
7             big_task1 cadservi+ partition+ 2024-08-01T22:11:50 2024-08-01T2
01T22:11:51     FAILED          2        1Mn   00:00:00    127:0 /fastone/use
8             Fano-slot shaobing+ partition+ 2024-08-01T22:50:31 2024-08-01T2
01T22:57:46     FAILED          4        1Mn   00:07:14    127:0 /fastone/use
9               sleep      cyan partition+ 2024-09-03T00:25:02 2024-09-03T0
03T00:26:02  COMPLETED          1        1Mn   00:01:00      0:0  /fastone/us
```

For more output fields, see `sacct -help`.

```
# For sacct output, -o can include the following fields:

--format=jobid,jobname,partition,maxvmsize,maxvmsizenode,
         maxvmsizetask,avevmsize,maxrss,maxrssnode,
         maxrsstask,averss,maxpages,maxpagesnode,
         maxpagestask,avepages,mincpu,mincpunode,
         mincputask,avecpu,ntasks,alloccpus,elapsed,
         state,exitcode,avecpufreq,reqcpufreqmin,
         reqcpufreqmax,reqcpufreqgov,consumedenergy,
         maxdiskread,maxdiskreadnode,maxdiskreadtask,
         avediskread,maxdiskwrite,maxdiskwritenode,
         maxdiskwritetask,avediskread,allocgres,reqgres

# If output is truncated, add "%field_length" after a format item to show
more, for example "workdir%300"
```

scontrol View Fsched configuration and status

`scontrol` is used to view or modify Fsched configuration, including jobs, job steps, nodes, partitions, reservations, and overall system configuration. Regular users can use `scontrol` to query and display many Fsched status details, while most modification commands can only be executed by the `root` user or administrators.

Common command and parameter examples for regular users:

Function	Command Example
View details of job ID 9	`scontrol show job 9`

Function	Command Example
View details of all running, queued, and just completed jobs	`scontrol show job`
View details of node compute03	`scontrol show node compute03`
View details of all nodes	`scontrol show node`
View details of all partitions	`scontrol show partition`
View Fsched configuration information	`scontrol show config`

Other options can be viewed with `scontrol --help`.

LSF/SGE Commands

Fsched provides most commonly used LSF and SGE commands and parameter combinations, so users can seamlessly switch to Fsched when their applications already integrate LSF/SGE.

Load the Environment

- If using cshell (`csh`):

Load the LSF environment

```
source /opt/fsched/wrappers/lsf/cshrc.lsf
```

Load the SGE environment

```
source /opt/fsched/wrappers/sge/cshrc.sge
```

- If using `sh`/`bash`:

Load the LSF environment

```
source /opt/fsched/wrappers/lsf/profile.lsf
```

Load the SGE environment

```
source /opt/fsched/wrappers/sge/profile.sge
```

Use Commands

After the environment is loaded, you can use LSF/SGE commands and parameters in the command line. For a detailed list of supported commands and parameters, see:

- LSF Command Usage
- SGE Command Usage

Partition Custom Parameters

Parameter	Description	fsched Version
AllocNodes	Comma separated list of nodes from which users can submit jobs in the partition.	
AllowAccounts	Comma separated list of accounts which may execute jobs in the partition. The default value is ""ALL"".	
AllowQos	Comma separated list of Qos which may execute jobs in the partition.	
Alternate	Partition name of alternate partition to be used if the state of this partition is ""DRAIN"" or ""INACTIVE.	
CpuBind	If a job step request does not specify an option to control how tasks are bound to allocated CPUs (--cpu-bind) and all nodes allocated to the job.	
DefCpuPerGPU	Default count of CPUs allocated per allocated GPU.	
DefMemPerCPU	Default real memory size available per allocated CPU in megabytes. Used to avoid over-subscribing memory and causing paging.	
DefMemPerGPU	Default real memory size available per allocated GPU in megabytes. Also see DefMemPerCPU, DefMemPerNode and MaxMemPerCPU.	
DefMemPerNode	Default real memory size available per allocated node in megabytes. Used to avoid over-subscribing memory and causing paging.	
DenyAccounts	Comma separated list of accounts which may not execute jobs in the partition.	

Parameter	Description	fsched Version
DenyQos	Comma separated list of Qos which may not execute jobs in the partition.	
DefaultTime	Run time limit used for jobs that don't specify a value. If not set then MaxTime will be used. Format is the same as for MaxTime.	
DisableRootJobs	If set to ""YES"" then user root will be prevented from running any jobs on this partition.	
ExclusiveUser	If set to ""YES"" then nodes will be exclusively allocated to users.	
GraceTime	Specifies, in units of seconds, the preemption grace time to be extended to a job which has been selected for preemption.	
Hidden	Specifies if the partition and its jobs are to be hidden by default. Hidden partitions will by default not be reported by the Slurm APIs or commands. Possible values are ""YES"" and ""NO"". The default value is ""NO"".	
LLN	Schedule resources to jobs on the least loaded nodes (based upon the number of idle CPUs).	
MaxCPUsPerNode	Maximum number of CPUs on any node available to all jobs from this partition. This can be especially useful to schedule GPUs.	
MaxMemPerCPU	Maximum real memory size available per allocated CPU in megabytes. Used to avoid over-subscribing memory and causing paging.	
MaxMemPerNode	Maximum real memory size available per allocated node in megabytes. Used to avoid over-subscribing memory and causing paging.	

Parameter	Description	fsched Version
MaxNodes	Maximum count of nodes which may be allocated to any single job. The default value is ""UNLIMITED"", which is represented internally as -1.	
MinNodes	Minimum count of nodes which may be allocated to any single job. The default value is 0.	
OverSubscribe	Controls the ability of the partition to execute more than one job at a time on each resource node, socket or core depending upon the value of.	
PreemptMode	Mechanism used to preempt jobs from this partition when PreemptType=preempt/partition_prio is configured.	
PriorityJobFactor	Partition factor used by priority/multifactor plugin in calculating job priority. The value may not exceed 65533. Also see PriorityTier.	
PriorityTier	Jobs submitted to a partition with a higher priority tier value will be dispatched before pending jobs in partition with lower priority tier value.	
QOS	Used to extend the limits available to a QOS on a partition. Jobs will not be associated to this QOS outside of being associated to the partition. They will still be associated to their requested QOS.	
ReqResv	Specifies users of this partition are required to designate a reservation when submitting a job.	
RootOnly	Specifies if only user ID zero (i.e. user root) may allocate resources in this partition.	
SelectTypeParameters	Partition-specific resource allocation type. This option replaces the global SelectTypeParameters value.	

Parameter	Description	fsched Version
TRESBillingWeights	TRESBillingWeights is used to define the billing weights of each TRES type that will be used in calculating the usage of a job.	
LoadSchedMem	Stop scheduling job to the node if node's available memory is lower than LoadSchedMem (in MiB)	fsched-10.25 +
LoadStopMem	Stop jobs on the node if node's available memory is lower than LoadStopMem (in MiB)	fsched-10.25 +
LoadSchedUt	Stop scheduling job to the node if node's cpu utilization is higher than LoadSchedUt (in %)	fsched-10.25 +
LoadStopUt	Stop jobs on the node if node's cpu utilization is higher than LoadStopUt (in %)	fsched-10.25 +
OverMemoryKill	Kill jobs that are being detected to use more memory than requested	fsched-10.25 +
LoadSchedR15s	Stop scheduling job to the node if node's 15-second average CPU run queue length is larger than LoadSchedR15s (float)	fsched-10.70 +
LoadStopR15s	Stop jobs on the node if node's 15-second average CPU run queue length is larger than LoadStopR15s (float)	fsched-10.70 +
LoadSchedR1m	Stop scheduling job to the node if node's 1-minute average CPU run queue length is larger than LoadSchedR1m (float)	fsched-10.70 +
LoadStopR1m	Stop jobs on the node if node's 1-minute average CPU run queue length is larger than LoadStopR1m (float)	fsched-10.70 +
LoadSchedR15m	Stop scheduling job to the node if node's 15-minute CPU run queue length is larger than LoadSchedR15m (float)	fsched-10.70 +

Parameter	Description	fsched Version
LoadStopR15m	Stop jobs on the node if node's 15-minute CPU run queue length is larger than LoadStopR15m (float)	fsched-10.70 +
AdaptSchedInterval	Interval of adaptive scheduling (in minutes)	fsched-dev(FIXME) +
AdaptMinJobElapsed	The minumal elapsed time of job which will be handled in adaptive scheduling (in seconds)	fsched-dev(FIXME) +
AdaptCpuMode	The adapt mode of CPU (can be INC_DEC/INC/DEC)	fsched-dev(FIXME) +
AdaptMemMode	The adapt mode of memory (can be INC_DEC/INC/DEC)	fsched-dev(FIXME) +
AdaptMemBasis	The adapt basis of memory (can be MAX/AVE)	fsched-dev(FIXME) +

Cluster Custom Parameters

Parameter	Description	fsched Version
AccountingStoreJobComment	If set to ""YES"" then include the job's comment field in the job complete message sent to the Accounting Storage database. The default is ""YES"".	
AcctGatherNodeFreq	The AcctGather plugins sampling interval for node accounting. For AcctGather plugin values of none, this parameter is ignored.	
AllowSpecResourcesUsage	If set to 1, Slurm allows individual jobs to override node's configured CoreSpecCount value.	
AuthAltTypes	Command separated list of alternative authentication plugins that the slurmctld will permit for communication.	
AuthInfo	Additional information to be used for authentication of communications between the Slurm daemons (slurmctld and slurmd) and the Slurm clients.	
BatchStartTimeout	The maximum time (in seconds) that a batch job is permitted for launching before being considered missing and releasing the allocation.	
BurstBufferType	The plugin used to manage burst buffers.	
CheckpointType	The system-initiated checkpoint method to be used for user jobs.	
CliFilterPlugins	A comma delimited list of command line interface option filter/modification plugins. The specified plugins will be executed in the order listed.	

Parameter	Description	fsched Version
CliFilterPluginsFile	The content of clifilter lua script file.	
CompleteWait	The time, in seconds, given for a job to remain in COMPLETING state before any additional jobs are scheduled.	
CoreSpecPlugin	Identifies the plugins to be used for enforcement of core specialization.	
CpuFreqDef	Default CPU frequency value or frequency governor to use when running a job step if it has not been explicitly set with the --cpu-freq option.	
CpuFreqGovernors	List of CPU frequency governors allowed to be set with the salloc, sbatch, or srun option --cpu-freq.	
CredType	The cryptographic signature tool to be used in the creation of job step credentials.	
DebugFlags	Defines specific subsystems which should provide more detailed event logging. Multiple subsystems can be specified with comma separators.	
DefCpuPerGPU	Default count of CPUs allocated per allocated GPU.	
DefMemPerCPU	Default real memory size available per allocated CPU in megabytes. Used to avoid over-subscribing memory and causing paging.	
DefMemPerGPU	Default real memory size available per allocated GPU in megabytes. The default value is 0 (unlimited). Also see DefMemPerCPU and DefMemPerNode.	
DefMemPerNode	Default real memory size available per allocated node in megabytes. Used to avoid over-subscribing	

Parameter	Description	fsched Version
	memory and causing paging.	
DefaultStorageHost	The default name of the machine hosting the accounting storage and job completion databases.	
DefaultStorageLoc	The fully qualified file name where accounting records and/or job completion records are written when the DefaultStorageType is ""filetxt"".	
DefaultStoragePass	The password used to gain access to the database to store the accounting and job completion data.	
DefaultStoragePort	The listening port of the accounting storage and/or job completion database server.	
DefaultStorageType	The accounting and job completion storage mechanism type. Acceptable values at present include ""filetxt"", ""mysql"" and ""none"".	
DefaultStorageUser	The user account for accessing the accounting storage and/or job completion database.	
DisableRootJobs	If set to ""YES"" then user root will be prevented from running any jobs. The default value is ""NO"", meaning user root will be able to execute jobs.	
EioTimeout	The number of seconds srun waits for slurmstepd to close the TCP/IP connection used to relay data between the user application and srun when the.	
EpilogMsgTime	The number of microseconds that the slurmctld daemon requires to process an epilog completion message from the slurmd daemons.	
FairShareDampeningFactor	Dampen the effect of exceeding a user or group's fair share of allocated resources.	

Parameter	Description	fsched Version
FederationParameters	Used to define federation options. Multiple options may be comma separated.	
FirstJobId	The job id to be used for the first submitted to Slurm without a specific requested value.	
GetEnvTimeout	Controls how long the job should wait (in seconds) to load the user's environment before attempting to load it from a cache file.	
GroupUpdateForce	If set to a non-zero value, then information about which users are members of groups allowed to use a partition will be updated periodically, even.	
GroupUpdateTime	Controls how frequently information about which users are members of groups allowed to use a partition will be updated, and how long user group.	
GpuFreqDef	Default GPU frequency to use when running a job step if it has not been explicitly set using the --gpu-freq option.	
HealthCheckInterval	The interval in seconds between executions of HealthCheckProgram. The default value is zero, which disables execution.	
HealthCheckNodeState	Identify what node states should execute the HealthCheckProgram. Multiple state values may be specified with a comma separator.	
HealthCheckProgram	Fully qualified pathname of a script to execute as user root periodically on all compute nodes that are not in the NOT_RESPONDING state.	
JobAcctGatherType	The job accounting mechanism type.	

Parameter	Description	fsched Version
JobAcctGatherFrequency	The job accounting and profiling sampling intervals.	
JobAcctGatherParams	Arbitrary parameters for the job account gather plugin Acceptable values at present include:.	
JobCheckpointDir	Specifies the default directory for storing or reading job checkpoint information.	
JobCompHost	The name of the machine hosting the job completion database. Only used for database type storage plugins, ignored otherwise.	
JobCompLoc	The fully qualified file name where job completion records are written when the JobCompType is ""jobcomp/filetxt"" or the database where job com-.	
JobCompPass	The password used to gain access to the database to store the job completion data.	
JobCompPort	The listening port of the job completion database server. Only used for database type storage plugins, ignored otherwise.	
JobCompType	The job completion logging mechanism type.	
JobCompUser	The user account for accessing the job completion database. Only used for database type storage plugins, ignored otherwise.	
JobContainerType	Identifies the plugin to be used for job tracking. The slurmd daemon must be restarted for a change in JobContainerType to take effect.	
JobFileAppend	This option controls what to do if a job's output or error file exist when the job is started.	

Parameter	Description	fsched Version
JobRequeue	This option controls the default ability for batch jobs to be requeued.	
JobSubmitPlugins	A comma delimited list of job submission plugins to be used. The specified plugins will be executed in the order listed.	
JobSubmitPluginsFile	A lua script file for job submission plugins.	
KeepAliveTime	Specifies how long sockets communications used between the srun command and its slurmstepd process are kept alive after disconnect.	
KillOnBadExit	If set to 1, a step will be terminated immediately if any task is crashed or aborted, as indicated by a non-zero exit code.	
KillWait	The interval, in seconds, given to a job's processes between the SIGTERM and SIGKILL signals upon reaching its time limit.	
NodeFeaturesPlugins	Identifies the plugins to be used for support of node features which can change through time.	
LaunchParameters	Identifies options to the job launch plugin.	
Licenses	Specification of licenses (or other resources available on all nodes of the cluster) which can be allocated to jobs.	
MailDomain	Domain name to qualify usernames if email address is not explicitly given with the ""--mail-user"" option.	
MailProg	Fully qualified pathname to the program used to send email per user request.	

Parameter	Description	fsched Version
MaxArraySize	The maximum job array size. The maximum job array task index value will be one less than MaxArraySize to allow for an index value of zero.	
MaxJobCount	The maximum number of jobs Slurm can have in its active database at one time.	
MaxJobId	The maximum job id to be used for jobs submitted to Slurm without a specific requested value.	
MaxMemPerCPU	Maximum real memory size available per allocated CPU in megabytes. Used to avoid over-subscribing memory and causing paging.	
MaxMemPerNode	Maximum real memory size available per allocated node in megabytes. Used to avoid over-subscribing memory and causing paging.	
MaxStepCount	The maximum number of steps that any job can initiate. This parameter is intended to limit the effect of bad batch scripts.	
MaxTasksPerNode	Maximum number of tasks Slurm will allow a job step to spawn on a single node. The default MaxTasksPerNode is 512. May not exceed 65533.	
MCSParameters	MCS = Multi-Category Security MCS Plugin Parameters. The supported parameters are specific to the MCSPlugin.	
MCSPlugin	MCS = Multi-Category Security : associate a security label to jobs and ensure that nodes can only be shared among jobs using the same security.	

Parameter	Description	fsched Version
MessageTimeout	Time permitted for a round-trip communication to complete in seconds. Default value is 10 seconds.	
MinJobAge	The minimum age of a completed job before its record is purged from Slurm's active database.	
MpiDefault	Identifies the default type of MPI to be used. Srun may override this configuration parameter in any case.	
MpiParams	MPI parameters. Used to identify ports used by older versions of OpenMPI and native Cray systems.	
MsgAggregationParams	Message aggregation parameters.	
OverTimeLimit	Number of minutes by which a job can exceed its time limit before being canceled.	
PowerParameters	System power management parameters. The supported parameters are specific to the PowerPlugin.	
PreemptMode	Enables gang scheduling and/or controls the mechanism used to preempt jobs.	
PreemptType	This specifies the plugin used to identify which jobs can be preempted in order to start a pending job.	
PreemptExemptTime	Global option for minimum run time for all jobs before they can be considered for preemption.	
PriorityCalcPeriod	The period of time in minutes in which the half-life decay will be re-calculated. Applicable only if PriorityType=priority/multifactor.	

Parameter	Description	fsched Version
PriorityDecayHalfLife	This controls how long prior resource use is considered in determining how over- or under-serviced an association is (user, bank account and cluster).	
PriorityFavorSmall	Specifies that small jobs should be given preferential scheduling priority. Applicable only if PriorityType=priority/multifactor.	
PriorityFlags	Flags to modify priority behavior. Applicable only if PriorityType=priority/multifactor. The keywords below have no associated value	
PriorityMaxAge	Specifies the job age which will be given the maximum age factor in computing priority.	
PriorityParameters	Arbitrary string used by the PriorityType plugin.	
PrioritySiteFactorParameters	Arbitrary string used by the PrioritySiteFactorPlugin plugin.	
PrioritySiteFactorPlugin	The specifies an optional plugin to be used alongside "priority/multifactor".	
PriorityType	This specifies the plugin to be used in establishing a job's scheduling priority.	
PriorityUsageResetPeriod	At this interval the usage of associations will be reset to 0. This is used if you want to enforce hard limits of time usage per association.	
PriorityWeightAge	An integer value that sets the degree to which the queue wait time component contributes to the job's priority.	

Parameter	Description	fsched Version
PriorityWeightAssoc	An integer value that sets the degree to which the association component contributes to the job's priority.	
PriorityWeightFairshare	An integer value that sets the degree to which the fair-share component contributes to the job's priority.	
PriorityWeightJobSize	An integer value that sets the degree to which the job size component contributes to the job's priority.	
PriorityWeightPartition	Partition factor used by priority/multifactor plugin in calculating job priority. Applicable only if PriorityType=priority/multifactor.	
PriorityWeightQOS	An integer value that sets the degree to which the Quality Of Service component contributes to the job's priority.	
PriorityWeightTRES	A comma separated list of TRES Types and weights that sets the degree that each TRES Type contributes to the job's priority.	
PrivateData	This controls what type of information is hidden from regular users. By default, all information is visible to all users.	
ProctrackType	Identifies the plugin to be used for process tracking on a job step basis.	
PrologEpilogTimeout	The interval in seconds Slurms waits for Prolog and Epilog before terminating them. The default behavior is to wait indefinitely.	
PropagatePrioProcess	Controls the scheduling priority (nice value) of user spawned tasks.	

Parameter	Description	fsched Version
PropagateResourceLimitsExcept	A list of comma separated resource limit names.	
ReconfigFlags	Flags to control various actions that may be taken when an ""scontrol reconfig"" command is issued.	
RequeueExit	Enables automatic requeue for batch jobs which exit with the specified values.	
RequeueExitHold	Enables automatic requeue for batch jobs which exit with the specified values, with these jobs being held until released manually by the user.	
ResumeFailProgram	The program that will be executed when nodes fail to resume to by ResumeTimeout.	
ResumeProgram	Slurm supports a mechanism to reduce power consumption on nodes that remain idle for an extended period of time.	
ResumeTimeout	Maximum time permitted (in seconds) between when a node resume request is issued and when the node is actually available for use.	
ResvOverRun	Describes how long a job already running in a reservation should be permitted to execute after the end time of the reservation has been reached.	
RoutePlugin	Identifies the plugin to be used for defining which nodes will be used for message forwarding and message aggregation.	
SallocDefaultCommand	Normally, salloc(1) will run the user's default shell when a command to execute is not specified on the salloc command line.	

Parameter	Description	fsched Version
SbcastParameters	Controls sbcast command behavior. Multiple options can be specified in a comma separated list.	
SchedulerParameters	The interpretation of this parameter varies by SchedulerType. Multiple options may be comma separated.	
SchedulerTimeSlice	Number of seconds in each time slice when gang scheduling is enabled (PreemptMode=SUSPEND,GANG).	
SchedulerType	Identifies the type of scheduler to be used.	
SelectTypeParameters	The permitted values of SelectTypeParameters depend upon the configured value of SelectType.	
SlurmdParameters	Parameters specific to the Slurmd. Multiple options may be comma separated.	
SlurmctldDebug	The level of detail to provide slurmctld daemon's logs. The default value is info.	
SlurmctldHost	The short, or long, hostname of the machine where Slurm control daemon is executed (i.e. the name returned by the command ""hostname -s"").	
SlurmctldLogFile	Fully qualified pathname of a file into which the slurmctld daemon's logs are written. The default value is none (performs logging via syslog).	
SlurmctldParameters	Multiple options may be comma-separated.	
SlurmctldPlugstack	A comma delimited list of Slurm controller plugins to be started when the daemon begins and terminated when it ends.	

Parameter	Description	fsched Version
SlurmctldPort	The port number that the Slurm controller, slurmctld, listens to for work. The default value is SLURMCTLD_PORT as established at system build time.	
SlurmctldSyslogDebug	The slurmctld daemon will log events to the syslog file at the specified level of detail.	
SlurmctldTimeout	The interval, in seconds, that the backup controller waits for the primary controller to respond before assuming control.	
SlurmdDebug	The level of detail to provide slurmd daemon's logs. The default value is info.	
SlurmdLogFile	Fully qualified pathname of a file into which the slurmd daemon's logs are written. The default value is none (performs logging via syslog).	
SlurmdPidFile	Fully qualified pathname of a file into which the slurmd daemon may write its process id. This may be used for automated signal processing.	
SlurmdPort	The port number that the Slurm compute node daemon, slurmd, listens to for work.	
SlurmdSpoolDir	Fully qualified pathname of a directory into which the slurmd daemon's state information and batch job script information are written.	
SlurmdSyslogDebug	The slurmd daemon will log events to the syslog file at the specified level of detail.	
SlurmdTimeout	The interval, in seconds, that the Slurm controller waits for slurmd to respond before configuring that node's state to DOWN.	

Parameter	Description	fsched Version
SlurmSchedLogFile	Fully qualified pathname of the scheduling event logging file. The syntax of this parameter is the same as for SlurmctldLogFile.	
SlurmSchedLogLevel	The initial level of scheduling event logging, similar to the SlurmctldDebug parameter used to control the initial level of slurmctld logging.	
SrunPortRange	The srun creates a set of listening ports to communicate with the controller, the slurmstepd and to handle the application I/O.	
StateSaveLocation	Fully qualified pathname of a directory into which the Slurm controller, slurmctld, saves its state (e.g. ""/usr/local/slurm/checkpoint"").	
SuspendExcNodes	Specifies the nodes which are to not be placed in power save mode, even if the node remains idle for an extended period of time.	
SuspendExcParts	Specifies the partitions whose nodes are to not be placed in power save mode, even if the node remains idle for an extended period of time.	
SuspendProgram	SuspendProgram is the program that will be executed when a node remains idle for an extended period of time.	
SuspendTimeout	Maximum time permitted (in seconds) between when a node suspend request is issued and when the node is shutdown.	
TaskPlugin	Identifies the type of task launch plugin, typically used to provide resource management within a node	

Parameter	Description	fsched Version
TaskPluginParam	Optional parameters for the task plugin. Multiple options should be comma separated.	
TCPTimeout	Time permitted for TCP connection to be established. Default value is 2 seconds.	
TmpFS	Fully qualified pathname of the file system available to user jobs for temporary storage. This parameter is used in establishing a node's TmpDisk space. The default value is "/tmp".	
TopologyParam	Comma separated options identifying network topology options.	
TopologyPlugin	Identifies the plugin to be used for determining the network topology and optimizing job allocations to minimize network contention.	
TrackWCKey	Boolean yes or no. Used to set display and track of the Workload Characterization Key. Must be set to track correct wckey usage.	
TreeWidth	Slurmd daemons use a virtual tree network for communications. TreeWidth specifies the width of the tree (i.e. the fanout).	
UnkillableStepProgram	If the processes in a job step are determined to be unkillable for a period of time specified by the UnkillableStepTimeout variable, the pro-.	
UnkillableStepTimeout	The length of time, in seconds, that Slurm will wait before deciding that processes in a job step are unkillable after they have been signaled with.	
UsePAM	If set to 1, PAM (Pluggable Authentication Modules for Linux) will be enabled. PAM is used to establish	

Parameter	Description	fsched Version
	the upper bounds for resource limits.	
VSizeFactor	Memory specifications in job requests apply to real memory size (also known as resident set size).	
WaitTime	Specifies how many seconds the srun command should by default wait after the first task terminates before terminating all remaining tasks.	
X11Parameters	For use with Slurm's built-in X11 forwarding implementation.	
MaxServerThreads	Maximum parallel threads to service incoming RPCs	fsched-10.25 +
MaxAgentThreads	Maximum parallel threads to service outgoing RPCs	fsched-10.25 +
FschedSrunPingInterval	Time interval between srun pings	fsched-10.25 +
FschedSrunPingMaxFailures	Max failures of srun pings before srun is killed	fsched-10.25 +
FschedAgentConnectTimeout	Timeout for agent connection	fsched-10.25 +
FschedPrologRetryInterval	Time interval between prolog retries	fsched-10.25 +
MaxInprogressRpcCalls	Max number of inprogress RPC calls allowed	fsched-10.25 +
SelectType	Identifies the type of resource selection algorithm to be used. Acceptable values include: 'select/cons_tres' or 'select/cons_tres_ex'	fsched-10.61 +

Parameter	Description	fsched Version
PriorityWeightFairshareUsed	An integer value that sets the degree to which the fair-share-used component contributes to the job's priority	fsched-10.87 +

Cgroup Parameters

Parameter	Description	Version Requirement
CGroup_ConstrainCores	If set to 1, constrains CPU cores using cgroups.	
CGroup_TaskAffinity	If set to 1, enables task affinity using cgroups.	Obceleted since fsched-10.67 +

File Parameters

Parameter	Description	Version Requirement
CliFilterPluginsFile	A lua script file for CLI filter plugins.	
JobSubmitPluginsFile	A lua script file for job submission plugins.	

CAE CPU Pinning Recommendations

This document records recommendations about CPU pinning when using CAE software. It applies to MPI applications.

Introduction

CAE software usually uses many processes for parallel computation and uses MPI for inter-process communication. When building the MPI runtime environment, there are two common ways under the SLURM scheduler:

- Direct: start MPI processes directly with the `srun` command.
- Indirect: start MPI processes with `mpirun` or `mpiexec`.

Under these two approaches, the pinning method and results differ.

Direct

- Resources are allocated by the scheduler.
- The scheduler performs pinning automatically based on resource allocation.

Indirect

- Resources are allocated by the scheduler.
- The MPI runtime performs pinning automatically based on resource allocation.

> ⚠ CAUTION
>
> Note: In this mode, OpenMPI gets pinning information through the cgroup `cpuset`.

CPU Pinning Recommendations

Because OPENMPI obtains pinning information through the cgroup `cpuset`, the cluster must enable cgroup and use the following configuration:

```
TaskPlugin = task/affinity,task/cgroup
TaskPluginParam = autobind=threads
```

```
CGroup_ConstrainCores=Yes
CGroup_TaskAffinity=no # Not required for fsched 10.67+
```

Where:

- `task/affinity`: enables pinning.
- `task/cgroup`: enables cgroup.
- `autobind=threads`: automatically pins to threads.

Checkpoint and Restore

Overview

The Checkpoint feature can save the state of a running job to disk, and later restore it to continue execution. This operation captures process memory, open files, and execution state, and is complex to implement.

Important note: The Checkpoint feature is not guaranteed to work for all applications. Success depends on application characteristics, system configuration, and runtime conditions. This feature is currently experimental and must be thoroughly tested before production use.

Supported Versions

fsched supports this feature after version (TBD).

Potential Use Cases

If the Checkpoint feature is reliable and effective for your application, it can be used in the following scenarios:

- Node failure recovery: After a node failure, fsched can automatically restore jobs from the Checkpoint to avoid restarting from scratch (requires application compatibility and successful restore).
- Long-running jobs: Provides some interruption protection for compatible applications.
- Planned maintenance: Pause jobs before maintenance, but recovery is not guaranteed.
- Resource management: If supported by the application, jobs can be moved off busy nodes.

Important Considerations

The feasibility of these scenarios depends entirely on your specific application:

- Application compatibility: Most complex applications have limitations and may be completely unsupported.
- Reliability: Checkpoint or restore operations may fail for various reasons.
- Testing requirements: Each application and workload must be extensively tested.
- No guarantees: Even after testing, Checkpoint/restore can still fail in production.

Do not rely on Checkpoint as a primary fault-tolerance or job-management mechanism without extensive validation.

How It Works

This feature is based on CRIU (Checkpoint/Restore In Userspace). CRIU is a Linux tool that can freeze a running application and save its state to disk. We use a customized version optimized for HPC workloads.

The Checkpoint process includes two steps:

1. Freeze: pause the job (duration depends on job size and system load)
2. Dump: write the job state to disk

During restore, fsched attempts to reconstruct the job state, including:

- Memory contents
- Open file descriptors
- Network connections
- Process tree structure
- Environment variables

Note: Whether restore succeeds depends on many factors. Complex applications with external dependencies, network state, or hardware access may not restore correctly.

Key Limitations

> **WARNING**
>
> The Checkpoint feature is complex and prone to failure
>
> Many applications cannot use the Checkpoint feature correctly. Even if basic tests pass, failures may still occur in production.
>
> Known incompatibilities:
>
> - GPU and specialized hardware access (typically unsupported)
> - Most MPI applications (unreliable)
> - Applications with complex network state

- Real-time or timing-sensitive applications
- Applications that depend on external services
- Complex inter-process communication patterns

Even simple applications can fail:

- File I/O operations in progress during Checkpoint
- Network connections that time out or cannot be re-established
- Temporary files or system resources that change between Checkpoint and restore
- Kernel or library version differences

Performance impact

- Checkpoint interrupts the job (depending on memory size, it may last from seconds to minutes)
- Checkpoint creation generates heavy I/O load
- Storage requirements are equal to or greater than the job's memory size
- Failed Checkpoints waste time and resources

System requirements

- Linux kernel 4.x or later is strongly recommended
- A shared file system with high I/O performance is required
- Sufficient free disk space (recommend several times the job's memory size)

Mandatory testing: Never assume Checkpoint will work. You must test extensively under real conditions. Even then, failures can occur in production.

Getting Started

Cluster Administrators

To enable Checkpoint on the cluster, configure the following:

```
# Enable Checkpoint plugin
CheckpointType = checkpoint/criu
```

This is the minimum configuration. For fsched-specific advanced settings, see the Configuration Options section below.

> ⚠ WARNING
>
> System requirements: Checkpoint requires Linux kernel 3.11 or later. For full functionality and reliability, Linux kernel 4.x or later is strongly recommended. CentOS/RHEL 7 may not support all features.

Users

Submit Checkpoint-Enabled Jobs

Use the `--checkpoint` option when submitting jobs to enable automatic Checkpointing:

```
sbatch --checkpoint=30:00 my_job_script.sh
```

This command creates a Checkpoint every 30 minutes. Supported time formats:

- `minutes` (for example `30`)
- `minutes:seconds` (for example `30:00`)
- `hours:minutes:seconds` (for example `2:30:00`)
- `days-hours` (for example `1-0` for 1 day)
- `days-hours:minutes` (for example `1-0:30`)
- `days-hours:minutes:seconds` (for example `1-2:30:00`)

Suggested starting intervals (adjust based on testing):

- Testing: `--checkpoint=5:00` (every 5 minutes)
- After application compatibility is verified: `--checkpoint=30:00` to `--checkpoint=2:00:00`
- Balance Checkpoint overhead against potential lost work

Manual Checkpoint Operations

You can also manually control Checkpoints with `scontrol`:

Manually create a Checkpoint:

```
scontrol checkpoint create <job_id>
```

Creates a Checkpoint for a running job, which may fail depending on application state.

Vacate a job and save its state (release resources):

```
scontrol checkpoint vacate <job_id>
```

Checkpoints the job and terminates it. If Checkpoint fails, the job is not terminated.

Restart a Checkpointed job:

```
scontrol checkpoint restart <job_id>
```

Restores the job from the last Checkpoint; it may fail for various reasons.

Basic Workflow Example

```
# 1. Submit a Checkpoint-enabled test job
$ sbatch --checkpoint=1:00:00 test_job.sh
Submitted batch job 12345

# 2. Check job status
$ squeue -j 12345
  JOBID PARTITION NAME  USER  ST TIME  NODES
  12345 compute   test  alice R  2:15  1

# 3. Attempt vacate (may fail)
$ scontrol checkpoint vacate 12345

# 4. If successful, the job is checkpointed and can be restarted
$ scontrol checkpoint restart 12345

# Note: verify job output to ensure correct recovery
```

Configuration Options

You can customize Checkpoint behavior with the following options. All options are optional and have reasonable defaults.

fsched-Specific Features

fsched provides several advanced Checkpoint features not available in standard Slurm:

Automatic recovery on node failure - When a node failure causes a job to be requeued, fsched automatically attempts to restore from the last Checkpoint instead of restarting from scratch. This can provide fault tolerance for compatible applications, but success is not guaranteed. Additional timeout configuration is required (see below).

Load-aware Checkpoint - Skips Checkpoint when system load is too high to reduce impact on other workloads. If load remains high, Checkpoint may be skipped for a long time.

Incremental Checkpoint - Saves only memory changes since the last Checkpoint. This can save time and storage for applications with stable memory access patterns, but it introduces dependencies between Checkpoints and can complicate the restore process.

Pre-dump - Copies memory before the final freeze to reduce pause time. Effectiveness depends on the application's memory access pattern.

Node Failure Recovery Configuration

To enable automatic Checkpoint recovery on node failure, use the following additional configuration:

```
BatchStartTimeout = 10
SlurmdTimeout = 15
AuthInfo = cred_expire=180,ttl=180,requeue_timeout=30
```

WARNING

Do not use these settings in high-load environments.

These timeout settings are aggressive and are mainly used for fast failure detection and requeue. But under high node load, they can cause false positives and unnecessary job failures:

- Jobs may be incorrectly marked failed when nodes slow down temporarily.
- Normal jobs may be terminated and requeued unnecessarily.
- The system may become unstable at high utilization.

Use these settings only when:

- The cluster typically has ample idle capacity.
- Node failures are frequent and you need fast recovery.
- Occasional false positives are acceptable.

> For high-utilization clusters, do not enable automatic node-failure recovery. Use manual Checkpoint/vacate/restart instead.

Basic Options

`CRIU_MaxEpochs` (default: `1`)

- Number of Checkpoint snapshots to keep
- Old Checkpoints are deleted automatically
- Higher values provide more restore options but use more disk space
- Example: `CRIU_MaxEpochs = 3` keeps the last 3 Checkpoints

Performance Optimization

`CRIU_Incremental` (default: `NO`)

- When enabled, saves only changes since the last Checkpoint
- Can reduce Checkpoint time and storage for some applications
- Drawback: restore requires a complete incremental chain; any file corruption causes restore failure
- Adds system complexity and potential failure points

`CRIU_FullEvery` (default: `10`)

- When using incremental Checkpoint, perform a full Checkpoint every N Checkpoints
- Prevents very long incremental chains that slow restore
- Applies only when `CRIU_Incremental = YES`
- Example: with `CRIU_FullEvery = 10`, Checkpoints 1-9 are incremental, 10 is full, then repeat

`CRIU_PreDump` (default: `NO`)

- Pre-copies memory before the final freeze to reduce freeze time
- Effectiveness depends on application memory access patterns
- May not significantly reduce freeze time for applications with frequent memory changes

`CRIU_PreDumpDelay` (default: `5` seconds)

- Wait time between pre-dump and final dump
- Allows the application to stabilize after pre-dump

- Applies only when `CRIU_PreDump = YES`

Load-Aware Checkpoint

`CRIU_LoadAware` (default: `NO`)

- Skips scheduled Checkpoint when system load is too high
- If the system stays busy, Checkpoint may be skipped for a long time
- Checks CPU, I/O, and memory load against thresholds

`CRIU_LoadThresholdCPU` (default: `0.95`)

- Skip Checkpoint if CPU utilization exceeds this ratio (0.0 to 1.0)
- Example: `0.95` means skip if CPU is above 95% utilization

`CRIU_LoadThresholdIO` (default: `0.50`)

- Skip Checkpoint if I/O wait exceeds this ratio (0.0 to 1.0)
- Example: `0.50` means skip if I/O wait exceeds 50%

`CRIU_LoadThresholdMemory` (default: `0.85`)

- Skip Checkpoint if memory usage exceeds this ratio (0.0 to 1.0)
- Example: `0.85` means skip if memory usage exceeds 85%

Configuration Examples

Minimal configuration (start here):

```
CheckpointType = checkpoint/criu
```

With incremental (test thoroughly before use):

```
CheckpointType = checkpoint/criu
CRIU_Incremental = YES
CRIU_FullEvery = 5
CRIU_MaxEpochs = 2
```

With load-aware (Checkpoints may be skipped):

```
CheckpointType = checkpoint/criu
CRIU_LoadAware = YES
CRIU_LoadThresholdCPU = 0.90
CRIU_LoadThresholdIO = 0.60
CRIU_LoadThresholdMemory = 0.80
```

Note: Start with the minimal configuration. Add features only after verifying they work for your specific application.

Command Reference

User Commands

Command	Description	When to Use
`sbatch --checkpoint=<interval> script.sh`	Submit a job with automatic Checkpoint enabled	Start a new job that needs periodic Checkpointing
`scontrol checkpoint create <job_id>`	Create a Checkpoint immediately	Before risky operations or for manual backup
`scontrol checkpoint vacate <job_id>`	Checkpoint and vacate the job, releasing resources	Planned maintenance or resource rebalancing
`scontrol checkpoint restart <job_id>`	Restore and continue a vacated job	After maintenance or when resources are available
`scontrol show job <job_id>`	View job details including Checkpoint status	Check Checkpoint configuration and status

Example Commands

Check whether a job has Checkpoint enabled:

```
scontrol show job 12345 | grep Checkpoint
```

Attempt to create a Checkpoint:

```
scontrol checkpoint create 12345
# Check logs to verify success
```

Attempt to vacate multiple jobs:

```
for job in 12345 12346 12347; do
  scontrol checkpoint vacate $job
  # Verify each Checkpoint before relying on it
done
```

Note: Before assuming a job is recoverable, verify Checkpoint success by checking logs and testing restores.

Best Practices

Choosing a Checkpoint Interval

Checkpoint interval selection depends on many factors and requires repeated testing:

Considerations:

- Checkpoint overhead increases with job memory size
- Storage I/O performance affects Checkpoint duration
- More frequent Checkpoints increase overhead but reduce work lost on failure
- Applications vary in how sensitive they are to Checkpoint interruptions

Testing approach:

1. Start with a longer interval (1-2 hours)
2. Measure Checkpoint duration and impact on the application
3. Adjust based on observed overhead and acceptable risk
4. Monitor Checkpoint failures and adjust accordingly

There is no universal "best" interval; determine it through repeated testing for your application and workload.

Storage Management

- Monitor Checkpoint storage usage: `du -sh /path/to/checkpoint/dir/<job_id>`

- Checkpoints are not automatically cleaned up after jobs finish
- For long-running jobs, consider using `CRIU_MaxEpochs` to limit storage
- Ensure sufficient free space (recommend 2x the job's memory size)

Application Compatibility

> **WARNING**
>
> You must test before production use.
>
> Checkpoint works at the process level and not all applications are compatible. Applications with complex I/O, network connections, hardware devices, or timing sensitivity may not run reliably.
>
> Required testing:
>
> 1. Test with the real application and workload
> 2. Verify output correctness after restore
> 3. Test on the actual hardware and configuration used in production
> 4. Record application-specific limitations or workarounds
>
> Do not assume Checkpoint works without thorough testing.

Test Checklist

Before using Checkpoint for important workloads, complete comprehensive testing:

1. Basic functionality test:

```
sbatch --checkpoint=5:00 test_job.sh
# Wait for the first Checkpoint to complete
scontrol checkpoint vacate <job_id>
scontrol checkpoint restart <job_id>
# Verify the job actually restarted (may fail)
```

2. Output verification:
    - Compare output files byte-for-byte with normal runs
    - Check for corrupted, duplicated, or missing data
    - Verify application state consistency after restore

3. Measure overhead:
    - Measure Checkpoint duration (may be much longer than expected)
    - Measure impact on total job runtime
    - Check storage usage
4. Test multiple scenarios:
    - Checkpoint at different stages of job execution
    - Test restore on different nodes (may fail)
    - Use real workloads, not simple examples
    - Test Checkpoint failure scenarios
5. Long-term reliability:
    - Run multiple Checkpoint/restore cycles
    - Monitor for performance degradation or data corruption
    - Test under system load

Important: Passing basic tests does not guarantee production reliability. Continuous monitoring is required after deployment.

FAQ

Q: Can GPU jobs use Checkpoint? A: GPU Checkpointing has severe limitations and is not recommended. CRIU's GPU state support is incomplete, and most CUDA/GPU applications cannot Checkpoint or restore properly. If you must use it, test thoroughly and expect issues.

Q: What happens to Checkpoint files after a job finishes? A: Checkpoint files remain on disk and must be cleaned up manually to free space. Monitor Checkpoint storage usage and delete old Checkpoint directories that are no longer needed.

Q: Can MPI jobs use Checkpoint? A: MPI Checkpointing is complex and usually unreliable. It depends heavily on MPI implementation, network architecture, and process communication patterns. Many MPI jobs cannot be Checkpointed successfully, and even with extensive testing, reliability may be limited.

Q: How much disk space does Checkpoint require? A: Each Checkpoint is roughly equal to the job's memory usage. With `CRIU_MaxEpochs = 3`, you need at least 3x the memory size. Incremental Checkpoints can save some space but still require significant storage. Use high-performance storage and ensure adequate space.

Q: Can I move Checkpoints to another cluster? A: No. Checkpoints are tightly coupled to a specific system configuration (including kernel version, libraries, file paths, and hardware) and can only be restored on the same cluster where they were created.

Q: Will Checkpoint affect job performance? A: Yes. Jobs are paused during Checkpoint creation (freeze time), which may last from milliseconds to seconds depending on memory size. Writing Checkpoint data also creates I/O overhead. Pre-dump can mitigate but not eliminate the impact. There is no performance impact between Checkpoints.

Q: What happens if Checkpoint fails? A: The job continues to run normally. Failed Checkpoints are recorded but do not terminate the job. However, you lose that restore point; if the job fails before the next successful Checkpoint, you must restart from an earlier Checkpoint or from scratch.

Q: Can I trigger Checkpoint from within a job script? A: No. Checkpoint must be controlled via fsched commands (`scontrol`). Job scripts cannot request Checkpoint directly. If needed, implement an application-level Checkpoint mechanism separately.

Q: How do I confirm Checkpoint success? A: Check fsched logs for Checkpoint success/failure messages; log location depends on configuration. Note: the absence of errors does not mean the Checkpoint is valid—only a successful restore test confirms it.

Q: Can I change the Checkpoint interval after submission? A: No. The Checkpoint interval is fixed at job submission time and cannot be changed while the job is running.

Q: Is Checkpoint the same as fsched job requeue? A: No. Job requeue (`--requeue`) terminates the job and starts it over. Checkpoint saves process state and continues from the interruption point. They serve different purposes.

Advanced fsched Configuration in FCP

FCP is a graphical cluster management platform that provides advanced configuration and management for FSCHED clusters. Through FCP, administrators can easily configure various advanced fsched options to optimize resource allocation and scheduling policies.

For the complete FSCHED advanced configuration documentation, see:

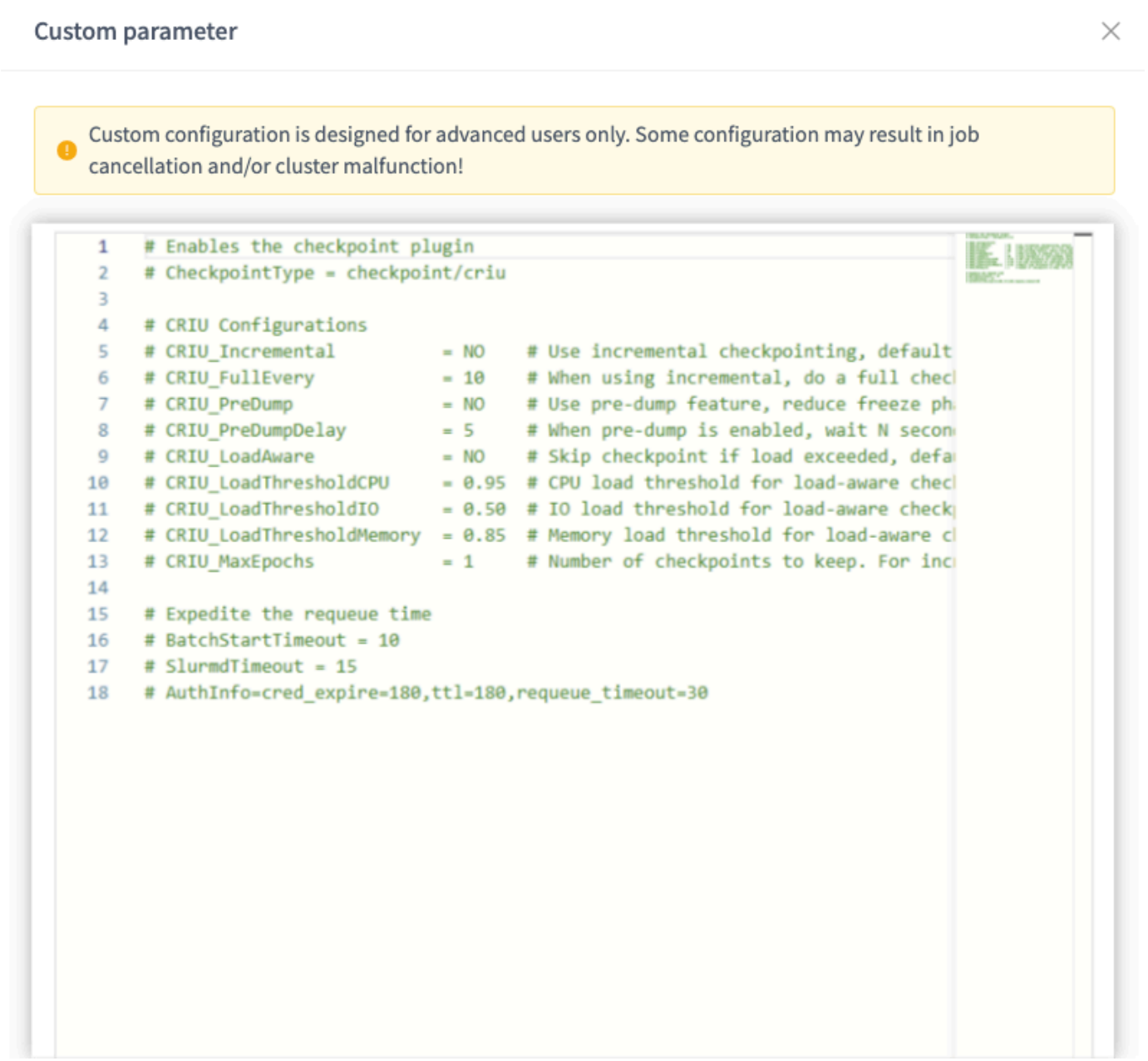
- Cluster Custom Parameters
- Partition Custom Parameters

Cluster Advanced Configuration

If you need advanced configuration for the entire cluster, use the cluster settings page in FCP. Steps:

1. Select the target cluster and open the cluster details page.
2. On the "Cluster Overview" page, click "Edit" for "Custom Parameters".

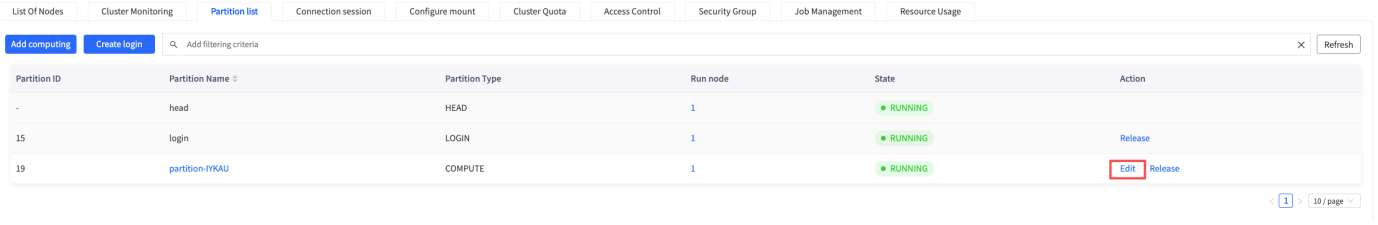
3. In the dialog, add or modify the required advanced configuration parameters.



Partition Advanced Configuration

If you need advanced configuration for a specific partition, use the partition settings page in FCP. Steps:

1. Select the target cluster and open the cluster details page.
2. On the "Partition List" tab, select the partition to configure and click "Edit".



3. In the dialog, add or modify the required advanced configuration parameters.

Edit computing partition

Modifying the configuration is only effective for later nodes. Nodes with auto scaling enabled will be released immediately after auto scaling is turned off.

Custom configuration is designed for advanced users only. Some configuration may result in job cancellation and/or cluster malfunction!

Common configuration

Default partition: ON

Advanced configuration

Allow groups: All groups ×

Allow users: All users ×

* The longest run of the job: 0 Hour No Limit

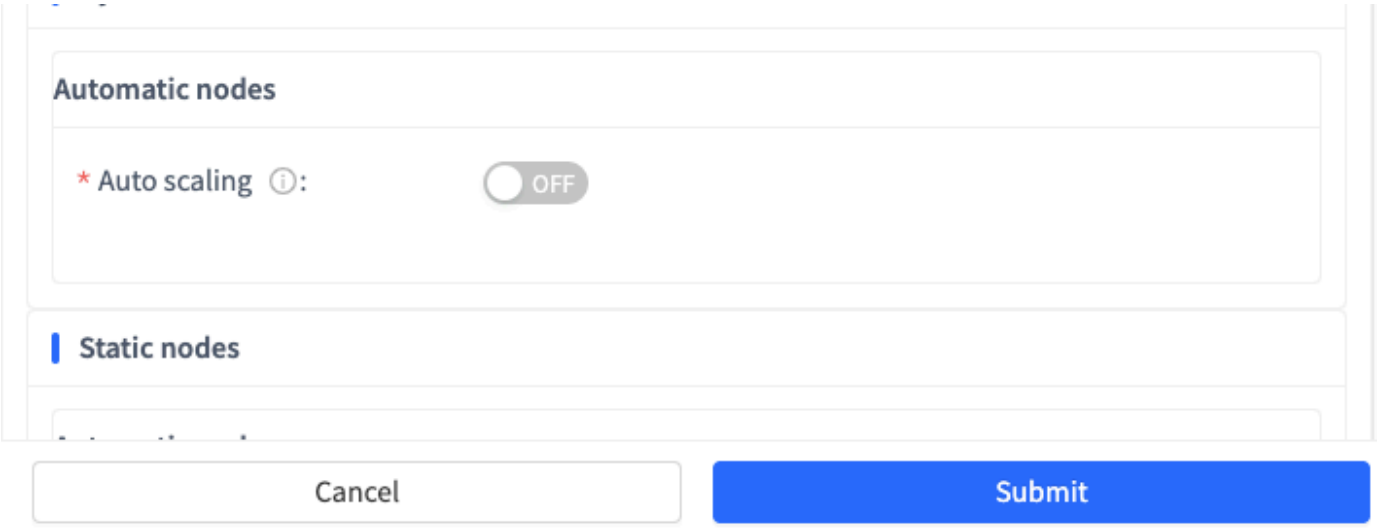
Maximum number of cpus used: 0 No Limit

* Load threshold: OFF

Custom parameter:

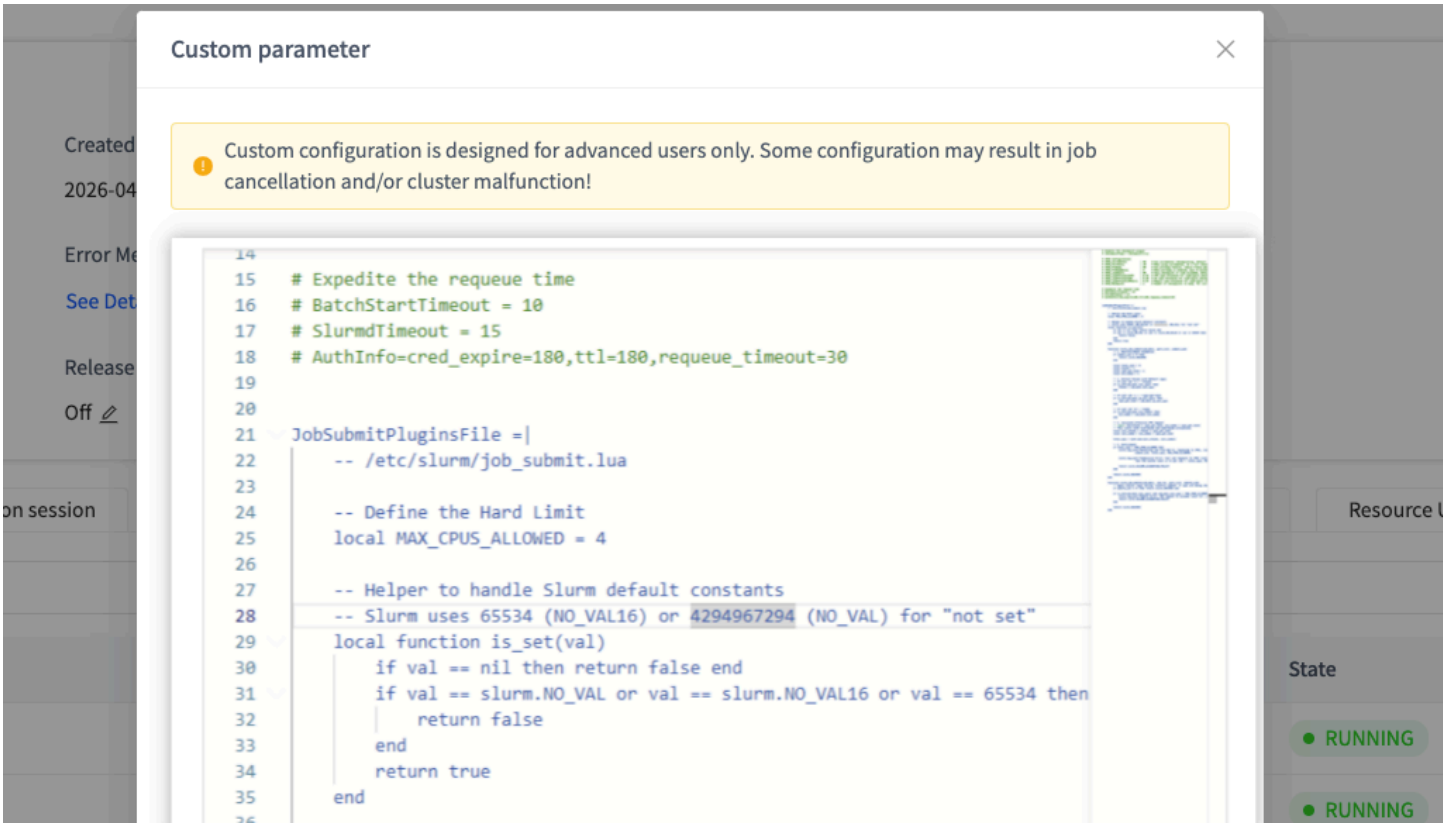
```
1
```

Dynamic nodes



Special Parameters

For file-type parameters, you can use multiline syntax. For example:



Advanced Configuration Syntax

Parameters in the advanced configuration box support a YAML-like syntax. The rules are:

- Each parameter is on its own line, in the format `parameter_name = parameter_value`.
- You may have spaces between the parameter name and the equals sign, but the parameter name cannot contain spaces.
- Parameter values can be strings, numbers, or booleans (yes/no).

- Comment lines start with `#`; comments are not parsed.
- Multiline parameter values use the `|` symbol, and subsequent lines must be indented.

QoS Rejection Policy Flags

fsched supports QoS (Quality of Service) rejection policy flags to control behavior when jobs exceed resource limits. With these flags, administrators can control scheduling policies more granularly.

> TIP
>
> This feature applies to fsched 10.101 and later.

Flag Details

DenyOnMaxPerJob

When `DenyOnMaxPerJob` is set, if a job requests resources that exceed the QoS per-job maximum resource limit (`MaxTRESPerJob`), the job is rejected immediately instead of entering the queue.

Use cases:

- Strictly limit per-job resource usage.
- Avoid users submitting oversized jobs that occupy the queue for a long time.

DenyOnMaxPerUser

When `DenyOnMaxPerUser` is set, if a job would cause the user's resource usage to exceed the QoS per-user maximum resource limit (`MaxTRESPerUser`), the job is rejected immediately instead of entering the queue.

Use cases:

- Strictly limit per-user resource usage.
- Prevent a single user from consuming too many cluster resources.

DenyOnGrp

When `DenyOnGrp` is set, if a job would cause the group's resource usage to exceed the QoS group maximum resource limit (`GrpTRES`), the job is rejected immediately instead of entering the queue.

Use cases:

- Strictly limit total resource usage for a group.

- Avoid a group's jobs over-consuming cluster resources.

How to Use

View QoS Flags

```
sacctmgr show qos format=name,flags
```

Set Rejection Policy Flags

Use `sacctmgr` to set QoS flags:

```
# Set a single flag
sacctmgr modify qos <qos_name> set flags=DenyOnMaxPerJob

# Set multiple flags
sacctmgr modify qos <qos_name> set flags=DenyOnMaxPerJob,DenyOnMaxPerUser

# Add a flag to existing flags
sacctmgr modify qos <qos_name> set flags+=DenyOnGrp
```

Remove Rejection Policy Flags

```
# Remove a specific flag
sacctmgr modify qos <qos_name> set flags-=DenyOnMaxPerJob
```

Configuration Examples

Example 1: Limit Per-Job Resources and Reject Oversized Jobs Immediately

1. Create a QoS and set the per-job maximum CPU count to 16.

```
sacctmgr add qos limited_job
sacctmgr modify qos limited_job set MaxTRESPerJob=cpu=16
```

2. Set the `DenyOnMaxPerJob` flag.

```
sacctmgr modify qos limited_job set flags=DenyOnMaxPerJob
```

3. Associate the QoS with a user.

```
sacctmgr modify user alice account=myaccount set qos=limited_job
```

4. Test: when user `alice` submits a job requesting more than 16 CPUs, it is rejected immediately.

```
alice@head:~$ sbatch -n 20 job.sh
sbatch: error: Batch job submission failed: Job violates accounting/QOS
policy
```

Example 2: Limit Total User Resources and Reject Oversized Jobs Immediately

1. Create a QoS and set the per-user maximum CPU count to 32.

```
sacctmgr add qos limited_user
sacctmgr modify qos limited_user set MaxTRESPerUser=cpu=32
```

2. Set the `DenyOnMaxPerUser` flag.

```
sacctmgr modify qos limited_user set flags=DenyOnMaxPerUser
```

3. Associate the QoS with a user.

```
sacctmgr modify user bob account=myaccount set qos=limited_user
```

4. Test: when user `bob` already has jobs using 24 CPUs, submitting another job requesting 16 CPUs will be rejected immediately.

```
bob@head:~$ squeue -u bob
JOBID PARTITION NAME USER ST TIME NODES NODELIST(REASON)
  101 partition job1  bob  R 0:30     3 compute[1-3]

bob@head:~$ sbatch -n 16 job2.sh
sbatch: error: Batch job submission failed: Job violates accounting/QOS
policy
```

Example 3: Combine Multiple Flags

1. Create a QoS and set multiple limits.

```
sacctmgr add qos strict_qos
sacctmgr modify qos strict_qos set MaxTRESPerJob=cpu=16
sacctmgr modify qos strict_qos set MaxTRESPerUser=cpu=32
sacctmgr modify qos strict_qos set GrpTRES=cpu=64
```

2. Set all three rejection policy flags at the same time.

```
sacctmgr modify qos strict_qos set
flags=DenyOnMaxPerJob,DenyOnMaxPerUser,DenyOnGrp
```

With this configuration:

- A single job requesting more than 16 CPUs will be rejected.
- A new job will be rejected when the user's total usage exceeds 32 CPUs.
- A new job will be rejected when the group's total usage exceeds 64 CPUs.

Behavior Comparison

Without Rejection Policy Flags (Default Behavior)

When a job exceeds resource limits, it enters the PENDING state and waits for resources to be released before it can run.

```
bob@head:~$ squeue
JOBID PARTITION NAME USER ST TIME NODES NODELIST(REASON)
  102 partition job2  bob PD 0:00     1 (MaxCpuPerUser)
  101 partition job1  bob  R 0:30     3 compute[1-3]
```

With Rejection Policy Flags Enabled

When a job exceeds resource limits, it is rejected immediately and the submission fails.

```
bob@head:~$ sbatch job2.sh
sbatch: error: Batch job submission failed: Job violates accounting/QOS
```

```
policy
```

Notes

> ⚠ WARNING
>
> 1. Rejection policy flags cause job submission to fail instead of waiting. Choose whether to enable them based on actual needs.
> 2. These flags affect only newly submitted jobs that exceed limits, and do not affect jobs that are already running.
> 3. Before enabling rejection policies, it is recommended to test resource limits without flags first to ensure they are reasonable.
> 4. Rejection policy flags are different from the `DenyOnLimit` flag: `DenyOnLimit` applies to all limit types, while these three flags can target different limit types separately.

Related Documents

- QOS New Policy Parameters
- sacctmgr Command Reference

QoS New Policy Parameters

fsched (fsched-10.61+) supports new QoS policy parameters:

1. `MaxTRESRunMinsPerAccount`
   - For all accounts associated with this QoS, the maximum total resource minutes that all running jobs of each account can use.
2. `MaxTRESRunMinsPerUser`
   - For all users associated with this QoS, the maximum total resource minutes that all running jobs of each user can use.

> ⚠ WARNING
>
> - Manually modifying slurm accounting associations and QoS settings conflicts with the UI cluster quota settings, so do not use them together with the UI cluster quota feature.
> - In addition to updating fsched, you must also update the slurmdbd container image. Otherwise, there will be a bug when setting these two new parameters (impact: if the value to set for MaxTRESRunMinsPerAccount equals the current value of MaxTRESRunMinsPerUser, the setting will fail with no error).
> - If you submit jobs without using `-t` to specify `time_limit`, the default `time_limit` is unlimited. In that case, if `MaxTRESRunMinsPerAccount` or `MaxTRESRunMinsPerUser` is set, the job will always be pending because the resource minutes exceed the maximum limit.

Example

1. Assume users `alice` and `bob` exist, both with primary group `jjtest`.

```
root@head1:~# id alice
uid=2006(alice) gid=2013(jjtest)
groups=2001(fsadmin),2003(defaultGroup),2013(jjtest)
root@head1:~# id bob
uid=2007(bob) gid=2013(jjtest)
groups=2001(fsadmin),2003(defaultGroup),2013(jjtest)
```

2. Create associations for the current cluster and users `alice` and `bob`.

```
sacctmgr add account jjtest cluster=fastone-18
sacctmgr add user alice account=jjtest cluster=fastone-18
```

```
sacctmgr add user bob account=jjtest cluster=fastone-18
```

3. Create a QoS and set `MaxTRESRunMinsPerAccount` and `MaxTRESRunMinsPerUser`.

```
sacctmgr add qos test_tres
sacctmgr modify qos test_tres set MaxTRESRunMinsPerAccount=cpu=3
sacctmgr modify qos test_tres set MaxTRESRunMinsPerUser=cpu=2
```

4. Associate the QoS with the associations of users `alice` and `bob`.

```
sacctmgr modify user alice account=jjtest cluster=fastone-18 set
qos=test_tres
sacctmgr modify user bob account=jjtest cluster=fastone-18 set
qos=test_tres
```

5. User `alice` submits a job with `time_limit` of 2 minutes, then submits another job with `time_limit` of 1 minute. The later job will be pending due to `MaxCpuRunMinsPerUser`.

```
alice@head1:~$ srun -t 2 sleep 100&
alice@head1:~$ srun -t 1 sleep 100&
alice@head1:~$ squeue
         JOBID PARTITION     NAME     USER ST       TIME  NODES
NODELIST(REASON)
            40 partition    sleep    alice PD       0:00      1
(MaxCpuRunMinsPerUser)
            39 partition    sleep    alice  R       0:07      1
compute1
```

6. Before the jobs in step 5 finish, user `bob` submits a job with `time_limit` of 2 minutes. The job will be pending due to `MaxCpuRunMinsPerAccount`.

```
bob@head1:~$ srun -t 2 sleep 600&
bob@head1:~$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES
NODELIST(REASON)
                42 partition    sleep    alice PD       0:00      1
(MaxCpuRunMinsPerUser)
                43 partition    sleep      bob PD       0:00      1
(MaxCpuRunMinsPerAccount)
```

```
                41 partition    sleep    alice  R       0:21      1
compute1
```

Using cgroup v2

fsched (fsched-10.87+) adds support for systems that use cgroup v2. On Ubuntu 22 hosts, in some cases you must manually configure systemd to ensure the system uses cgroup v2.

> ⚠ WARNING
>
> - On Ubuntu 22 hosts, first check `/proc/1/cgroup`. If there is only one line and it is `0::/init.scope`, then the system is already using cgroup v2 and no extra configuration is needed. Otherwise, configure systemd as described below.
> - After modifying systemd, the host can no longer mount cgroup v1. Therefore, if SLURM is configured to use cgroup, you cannot install older fsched versions (before fsched-10.70). Otherwise, `slurmd` cannot start and the node will become unknown.

Configure systemd to ensure the system uses cgroup v2

On Ubuntu 22 hosts, if `/proc/1/cgroup` has multiple lines or the first line starts with a non-zero value, configure systemd as follows to ensure the system uses cgroup v2.

1. Check `/proc/1/cgroup` and confirm systemd must be configured.

```
root@jjtest-ubuntu22-3:~# cat /proc/1/cgroup
2:cpuacct:/
1:freezer:/
0::/init.scope
```

2. Modify grub to add the parameters `systemd.unified_cgroup_hierarchy=1 systemd.legacy_systemd_cgroup_controller=0 cgroup_no_v1=all`.

   For example, if `GRUB_CMDLINE_LINUX` is currently:

```
root@jjtest-ubuntu22-3:~# grep -w GRUB_CMDLINE_LINUX /etc/default/grub
GRUB_CMDLINE_LINUX="quiet splash net.ifnames=0 biosdevname=0"
```

   Then comment out the original `GRUB_CMDLINE_LINUX` line and set the new `GRUB_CMDLINE_LINUX` to append the cgroup parameters to the existing options:

```
root@jjtest-ubuntu22-3:~# sed -e
's@^GRUB_CMDLINE_LINUX=@#GRUB_CMDLINE_LINUX=@' -i /etc/default/grub
echo 'GRUB_CMDLINE_LINUX="quiet splash net.ifnames=0 biosdevname=0
systemd.unified_cgroup_hierarchy=1
systemd.legacy_systemd_cgroup_controller=0 cgroup_no_v1=all"' >>
/etc/default/grub
```

3. Update grub.

```
root@jjtest-ubuntu22-3:~# update-grub
```

4. Reboot the host to apply the new kernel command line options.

5. Check `/proc/1/cgroup`. If the result is as follows, the configuration succeeded.

```
root@jjtest-ubuntu22-3:~# cat /proc/1/cgroup
0::/init.scope
```

Fairshare Used Factor That Considers Actual Resource Usage

Slurm uses the `priority/multifactor` plugin for multifactor job scheduling. The `Fairshare Factor` considers the difference between committed resources and actually allocated resources. fsched (fsched-10.87) adds the new `FairshareUsed Factor`, which considers the difference between committed resources and actually consumed resources.

Purpose

When calculating a user's job priority, this factor considers the actual resources consumed by the user's completed jobs. More actual consumption lowers the priority associated with `FAIRSHAREUSED`, thereby reducing the overall priority.

> **WARNING**
>
> - If you upgrade to this fsched version and later downgrade to an older version, the `slurmctld` service will fail to start due to an incompatible `state` file version. In the `slurmctld` log you will see `F: Can not recover assoc_usage state, incompatible version, got 2561 need >= 8192 <= 8704`. In this case, manually delete the `assoc_usage` file under the cluster `state` directory, then restart `slurmctld`.

> **TIP**
>
> - To use the `FairshareUsed Factor`, you must configure the corresponding `association` for `cluster-account-user` in `accounting` first.
> - Currently, `FairshareUsed Factor` supports only the default `Fair Tree` fairshare algorithm and does not support the `classic` fairshare algorithm.
> - If you set a relatively large weight for `FairshareUsed Factor`, then the more resources are actually consumed, the smaller the pending jobs' priority values for that user, and the lower the priority.

Cluster Configuration

`PriorityWeightFairshareUsed`: a cluster custom parameter of integer type that represents the weight of `FairshareUsed Factor`.

Inspection Tools

sshare

`sshare` is used to list resource shares for `association` in the cluster. fsched (fsched-10.87) adds a new `sshare` option `--ext` that includes information related to `FairshareUsed Factor`.

sprio

`sprio` is used to view the factors that make up job scheduling priority. fsched (fsched-10.87) adds a new `sprio` option `--ext` that includes information related to `FairshareUsed Factor`.

Usage Example

1. Confirm that the corresponding `association` for `cluster-account-user` is configured in `accounting`.

```
root@ubuntu22-4c-3:~# sacctmgr list assoc cluster=fastone-5
Cluster    Account       User  Partition     Share   Priority GrpJobs
GrpTRES GrpSubmit     GrpWall   GrpTRESMins MaxJobs       MaxTRES
MaxTRESPerNode MaxSubmit     MaxWall   MaxTRESMins                  QOS
Def QOS GrpTRESRunMin
---------- ---------- ---------- ---------- --------- ---------- ------
- ------------- --------- ----------- ------------- ------- -----------
-- -------------- --------- ----------- ------------- ----------------
--- --------- -------------
fastone-5       root                                1
normal
fastone-5       root       root                     1
normal
fastone-5 _fsched_a+                                1
normal
fastone-5         jj                                1
normal
fastone-5         jj         jj partition+          1
user-fastone-5-jj-p+
fastone-5      test1                                1
normal
fastone-5      test1      alice partition+          1
fastone-5-test1-par+
fastone-5      test1        bob partition+          1
fastone-5-test1-par+
fastone-5      test2                                1
```

```
normal
fastone-5      test2    charlie partition+         1
fastone-5-test2-par+
```

2. Modify the cluster custom parameters, set the priority plugin to `priority/multifactor`, and set the desired `PriorityWeightFairshareUsed`. Here, to make actual resource consumption have a larger impact on priority, a larger `PriorityWeightFairshareUsed` is configured.

```
PriorityType=priority/multifactor
PriorityWeightFairshare=30
PriorityWeightFairshareUsed=3000
```

3. Submit jobs that consume more resources with some users, and submit jobs that allocate resources but consume fewer resources with other users, to test the impact of the new factor.

```
alice@ubuntu22-4c-1:~$ srun -n4 sleep 100&
[1] 2512319
alice@ubuntu22-4c-1:~$ srun -n4 sleep 100&
[2] 2512372
```

```
charlie@ubuntu22-4c-2:~$ srun stress-ng --cpu 1 --cpu-load 100 -t 100s
&
[1] 164380
charlie@ubuntu22-4c-2:~$ srun stress-ng --cpu 1 --cpu-load 100 -t 100s
&
[2] 164387
```

4. Before and after the resource-intensive jobs complete, use `sshare --ext` to view resource shares and compare the changes.

```
root@ubuntu22-4c-3:~# sshare -a --ext
             Account       User  RawShares  NormShares    RawUsage
EffectvUsage  FairShare RawUsageUse EffectvUsageU FairShareU
-------------------- ---------- ---------- ----------- ----------- ----
--------- ---------- ----------- ------------- ----------
root                                          0.000000         517
1.000000                      14      0.000000
root                        root           1    0.200000           0
0.000000   0.800000           0      0.000815   0.600000
_fsched_all                               1    0.200000           0
0.000000                      0      0.000000
```

```
jj                                      1    0.200000           0
0.000000                     0      0.000000
jj                         jj          1    1.000000           0
0.000000   1.000000          0      0.000000   1.000000
test1                                   1    0.200000         394
0.763174                    14      0.999185
test1                   alice           1    0.500000         394
1.000000   0.200000         14      1.000000   0.200000
test1                     bob           1    0.500000           0
0.000000   0.400000          0      0.000000   0.400000
test2                                   1    0.200000         122
0.236825                     0      0.000000
test2                 charlie           1    1.000000         122
1.000000   0.600000          0      0.000000   1.000000
```

```
root@ubuntu22-4c-3:~# sshare -a --ext
            Account       User  RawShares  NormShares    RawUsage
EffectvUsage  FairShare RawUsageUse EffectvUsageU FairShareU
-------------------- ---------- ---------- ----------- ----------- ----
--------- ---------- ----------- ------------ ----------
root                                          0.000000         898
1.000000                177817      0.000000
root                     root           1    0.200000           0
0.000000   0.800000          0      0.000000   0.800000
_fsched_all                             1    0.200000           0
0.000000                     0      0.000000
jj                                      1    0.200000           0
0.000000                     0      0.000000
jj                         jj           1    1.000000           0
0.000000   1.000000          0      0.000000   1.000000
test1                                   1    0.200000         705
0.784619                    14      0.000081
test1                   alice           1    0.500000         705
1.000000   0.200000         14      1.000000   0.400000
test1                     bob           1    0.500000           0
0.000000   0.400000          0      0.000000   0.600000
test2                                   1    0.200000         193
0.215381                177803      0.999919
test2                 charlie           1    1.000000         193
1.000000   0.600000      177803      1.000000   0.200000
```

5. Before and after the resource-intensive jobs complete, use `sprio --ext` to view the priority changes of users' pending jobs.

```
root@ubuntu22-4c-3:~# sprio --ext -l
        JOBID PARTITION     USER   PRIORITY       SITE        AGE
ASSOC  FAIRSHARE    JOBSIZE  PARTITION        QOS        NICE
TRES FAIRSHAREU
            674 partition    alice        606          0          0
0          6          0          0          0          0
600
            675 partition  charlie       3018          0          0
0         18          0          0          0          0
3000
```

```
root@ubuntu22-4c-3:~# sprio --ext -l
        JOBID PARTITION     USER   PRIORITY       SITE        AGE
ASSOC  FAIRSHARE    JOBSIZE  PARTITION        QOS        NICE
TRES FAIRSHAREU
            675 partition  charlie        618          0          0
0         18          0          0          0          0
600
            676 partition    alice       1206          0          0
0          6          0          0          0          0
1200
            677 partition  charlie        618          0          0
0         18          0          0          0          0
600
```

License Scheduling

Overview

fsched provides license server monitoring and license allocation management for the current cluster. When a license query script exists, fsched automatically monitors license usage changes and adjusts the number of licenses available to jobs in real time, ensuring licenses are allocated correctly during scheduling.

> ⚠ WARNING
>
> - Enabling license scheduling affects job scheduling and must be configured carefully by administrators.

> 💡 TIP
>
> - Ensure network connectivity between the cluster head node and the license server.
> - The license query script is uploaded to the cluster head node. The fixed path and file name must be `/etc/fastone/license_query`, and the script output format must follow the specified format. See >>> License Query Script Output Format.
> - Because fsched converts license names and server names to lowercase, users must submit license names and server names in lowercase when submitting jobs.
> - If the license query script uses the license server client tool `lmstat`, you must add the license server domain name to `/etc/hosts` on the head node first.

Configuration

License scheduling provides two configuration tools for administrators:

1. License Allocation management: manage current cluster license allocations to limit the maximum number of licenses the cluster can use, using the `sacctmgr` resource command.
2. License Sync management: manage synchronization of license usage.

License Allocation Management

Use the `sacctmgr` resource command to manage license allocation for the current cluster.

Get the current cluster name

```
scontrol show config | grep ClusterName
```

Add license allocation

1. Check whether the license already exists.

```
sacctmgr show resource name=test8 server=server3
```

2. If it does not exist, you can add it in one step:

```
sacctmgr add resource name=test8 server=server3 servertype=flexlm
type=license count=60 cluster=fastone-1 percentallowed=70
```

3. If it already exists:

If the license already exists, you cannot set a new `count`. If you need to modify it, use the `sacctmgr modify resource` command.

3.1 Check whether there is an allocation for the current cluster:

```
sacctmgr show resource withclusters name=test8 server=server3
cluster=fastone-1
```

3.2 If there is no allocation for the current cluster, you can set `percentallowed` for the current cluster:

```
sacctmgr add resource name=test8 server=server3 cluster=fastone-1
percentallowed=70
```

3.3 If there is already an allocation for the current cluster, you cannot set it again. If you need to modify it, use the `sacctmgr modify resource` command.

Parameter descriptions:

- `name`: license name
- `cluster`: fsched cluster name
- `count`: total number of licenses, including the total available inside and outside the cluster
- `percentallowed`: percentage allowed for the current cluster

- `server`: license server name
- `servertype`: license server type
- `type`: resource name, should be `license` in this workflow

Modify license allocation

- To modify the total license count:

```
sacctmgr modify resource name=matlab server=server2 set count=200
```

- To modify `percentallowed` for the current cluster:

```
sacctmgr modify resource name=matlab server=server2 cluster=fastone-1 set
percentallowed=30
```

Parameter descriptions:

- `name`: license name
- `server`: server name
- `cluster`: fsched cluster name
- `count`: new total license count
- `percentallowed`: new percentage

Delete license allocation

- If there is only one cluster, or if you want to delete allocations for all clusters:

```
sacctmgr delete resource where name=matlab server=server2
```

- If there are multiple clusters and you want to delete the allocation for the current cluster:

```
sacctmgr delete resource where name=matlab server=server2 cluster=fastone-1
```

Parameter descriptions:

- `name`: license name
- `server`: server name

- `cluster`: fsched cluster name

Query license allocation

- Query allocations for the current cluster:

```
# Query all license allocations
sacctmgr show resource withclusters cluster=fastone-1

# Filter by name
sacctmgr show resource withclusters cluster=fastone-1 name=nastran

# Filter by server
sacctmgr show resource withclusters cluster=fastone-1 server=flex_host

# Exact query
sacctmgr show resource withclusters cluster=fastone-1 name=matlab
server=rlm_host
```

- Query all license allocations:

```
sacctmgr show resource
```

Parameter descriptions:

- `name`: filter by license name
- `server`: filter by server name
- `cluster`: fsched cluster name

License Sync Management

Use the output of the license query script to automatically sync license usage changes to the current cluster.

Set the license query script

License Query Script Output Format

The script must print license information to standard output (stdout), one line per entry, in the following format:

```

```

```
name@server_name:consumed
```

Field	Description
`name`	license name
`server_name`	license server name
`consumed`	current number of licenses in use

Example output:

```
ansys@flexlm_server:5
meba@flexlm_server:20
```

- The license query script takes no input parameters.
- Script examples and documentation can be found on the head node at `/opt/fsched/wrappers/statesvc/license_tools/`.

License query script deployment

Copy the script to the fixed query path `/etc/fastone/license_query`. For example, with the script `license_query_flexlm.sh`:

```
cp license_query_flexlm.sh /etc/fastone/license_query
chmod +x /etc/fastone/license_query
```

Stop license sync

Delete the license query script `/etc/fastone/license_query` to stop synchronization automatically.

View license sync status

```
/opt/fsched/wrappers/statesvc/bin/stateclient lic-sync-status
```

This shows the current license sync status, including sync status, last update time and content, and error messages.

Configuration Example

The following commands are run on the cluster head node, using a single-cluster scenario as an example.

1. Check whether the license to be managed has already been allocated to the current cluster; if not, allocate it.

   - View the current cluster name:

```
[root@centos7-16c-1 ~]# scontrol show config | grep ClusterName
ClusterName             = fastone-2
```

   - View allocated licenses:

```
[root@centos7-16c-1 ~]# sacctmgr show resource
      Name     Server     Type  Count % Allocated ServerType
---------- ---------- -------- ------ ----------- ----------
```

   - The total license count can be confirmed via `Total` in `lmstat` output:

```
[root@centos7-16c-1 ~]# ./lmstat -f ansys -c 1055@10.106.1.111
lmstat - Copyright (c) 1989-2016 Flexera Software LLC. All Rights Reserved
Flexible License Manager status on Fri 3/13/2026 18:39

License server status: 1055@ansys-lic
    License file(s) on ansys-lic:
/fastone/softwares/ansys/ansys/2024R1/ansys_inc/shared_files/licensing/lic

ansys-lic: license server UP (MASTER) v11.19.4

Vendor daemon status (on ansys-lic):

  ansyslmd: UP v11.19.4
Feature usage info:

Users of ansys:  (Total of 100 licenses issued;  Total of 0 licenses in us

[root@centos7-16c-1 ~]# ./lmstat -f meba -c 1055@10.106.1.153
lmstat - Copyright (c) 1989-2016 Flexera Software LLC. All Rights Reserved
Flexible License Manager status on Fri 3/13/2026 18:43

License server status: 1055@ansys-lic
    License file(s) on ansys-lic:
/fastone/softwares/ansys/ansys/2024R1/ansys_inc/shared_files/licensing/lic
```

```
ansys-lic: license server UP (MASTER) v11.19.4

Vendor daemon status (on ansys-lic):

  ansyslmd: UP v11.19.4
Feature usage info:

Users of meba:  (Total of 100 licenses issued;  Total of 0 licenses in use
```

- If this `Name,Server` combination does not exist, allocate the maximum available licenses for the current cluster. Use a unique server name defined by the administrator that corresponds to the license server:

```
[root@centos7-16c-1 ~]# sacctmgr add resource name=ansys
server=flexlm_server servertype=flexlm_server type=license count=100
cluster=fastone-2 percentallowed=100
Adding Resource(s)
  ansys@flexlm_server
  Cluster - fastone-2  100%
Settings
  Name           = ansys
  Server         = flexlm_server
  Description    = ansys
  ServerType     = flexlm_server
  Count          = 100
  Type           = License
Would you like to commit changes? (You have 30 seconds to decide)
(N/y): y
[root@centos7-16c-1 ~]# sacctmgr add resource name=meba
server=flexlm_server servertype=flexlm_server type=license count=100
cluster=fastone-2 percentallowed=100
Adding Resource(s)
  meba@flexlm_server
  Cluster - fastone-2  100%
Settings
  Name           = meba
  Server         = flexlm_server
  Description    = meba
  ServerType     = flexlm_server
  Count          = 100
  Type           = License
Would you like to commit changes? (You have 30 seconds to decide)
(N/y): y
```

- View the configuration results:

```
[root@centos7-16c-1 ~]# sacctmgr show resource -p
Name|Server|Type|Count|% Allocated|ServerType|
ansys|flexlm_server|License|100|100|flexlm_server|
meba|flexlm_server|License|100|100|flexlm_server|
```

2. Set license synchronization.

   - Copy the script to the fixed query path `/etc/fastone/license_query`, using `license_query_flexlm.sh` as an example:

```
cp license_query_flexlm.sh /etc/fastone/license_query
chmod +x /etc/fastone/license_query
```

   - View license sync status:

```
[root@centos7-16c-1 ~]# /opt/fsched/wrappers/statesvc/bin/stateclient
lic-sync-status
Sending requests to server localhost:20051
Status: OK
Last Update: 2026-03-13T11:08:24Z
Licenses (2):
  ansys@flexlm_server: 30
  meba@flexlm_server: 20
```

Cluster Usage Example

The following commands can be run on any node in the cluster.

1. View the licenses available to the current cluster.

The output in the example shows available licenses (`license_name@server_name`). `LastConsumed` shows total consumption for the license (including inside and outside the cluster), and `Free` shows remaining available licenses for the current cluster.

```
[root@centos7-16c-1 ~]# scontrol show lic --ext
LicenseName=meba@flexlm_server
LicenseName=meba@flexlm_server
    Total=100 Used=0 Free=80 Reserved=0 Remote=yes
    LastConsumed=20 LastDeficit=20 LastUpdate=Unknown
LicenseName=ansys@flexlm_server
```

```
    Total=100 Used=0 Free=70 Reserved=0 Remote=yes
    LastConsumed=30 LastDeficit=30 LastUpdate=Unknown
```

2. Specify the requested license count when submitting a job.

The job below requests more licenses than the current `Free` value for that license in the cluster, so the job is pending due to `Licenses`.

```
[root@centos7-16c-1 ~]# srun -L ansys@flexlm_server:90 sleep 600&
[1] 5586
[root@centos7-16c-1 ~]# srun: job 7 queued and waiting for resources

[root@centos7-16c-1 ~]# squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES
NODELIST(REASON)
                 7 partition    sleep     root PD       0:00      1
(Licenses)
```

After some time, if external license usage is released, fsched will automatically update the monitored license consumption to the cluster and the job can run.

```
[root@centos7-16c-1 ~]# squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES
NODELIST(REASON)
                 7 partition    sleep     root  R       0:05      1 jj-
centos7-2
[root@centos7-16c-1 ~]# scontrol show lic --ext
LicenseName=meba@flexlm_server
    Total=100 Used=0 Free=80 Reserved=0 Remote=yes
    LastConsumed=20 LastDeficit=20 LastUpdate=Unknown
LicenseName=ansys@flexlm_server
    Total=100 Used=90 Free=10 Reserved=0 Remote=yes
    LastConsumed=5 LastDeficit=0 LastUpdate=Unknown
```

License Query

Overview

fsched supports real-time queries of cluster license usage status and user license allocations.

> ⚠ WARNING
>
> - A large number of concurrent queries may impact system performance. Since user license allocations are updated every 30 seconds, periodic queries should be at least 30 seconds apart.
> - `list-user-lic-allocs` consumes significant memory when there are many users and license types. Unfiltered queries can consume a large amount of memory in large clusters. Ensure the fsched server has enough memory, and reserve at least 2x memory for large-scale query operations. Memory usage reference by cluster size:
>   - 1,000 users × 300 licenses = 300,000 records → about 121MB memory (recommend reserving 242MB)
>   - 1,000 users × 500 licenses = 500,000 records → about 167MB memory (recommend reserving 334MB)
>   - 1,000 users × 1,000 licenses = 1,000,000 records → about 335MB memory (recommend reserving 670MB)
>   - 2,000 users × 1,000 licenses = 2,000,000 records → about 669MB memory (recommend reserving 1.3GB)

> 💡 TIP
>
> - All query commands support `--help` for detailed usage.
> - License names are normalized to lowercase in fsched.
> - Query results are based on real-time job data and the SLURM native API to ensure accuracy.
> - Data from different servers for the same license type is automatically aggregated.

User License Allocation Query

Function Description

The `list-user-lic-allocs` command queries user license allocation summaries and shows license requests for each user's running jobs. This feature is computed from real-time job data and

reflects the current distribution of license usage among users in the cluster.

Key features:

- Real-time calculation: based on currently running job data
- Automatic aggregation: usage from different servers for the same license type is aggregated
- Active filter: only records with license_count > 0 are shown
- Cross-server aggregation: no client-side data merging needed

Query All User License Allocations

Query license allocations for all users in the current cluster:

```
./stateclient list-user-lic-allocs
```

Sample output:

```
Sending requests to server localhost:20051
User License Allocations (4):
========================================
User: alice
  License: matlab
  Count: 8
----------------------------------------
User: alice
  License: ansys
  Count: 3
----------------------------------------
User: bob
  License: matlab
  Count: 2
----------------------------------------
User: charlie
  License: nastran
  Count: 5
----------------------------------------
```

Parameters:

- `--user`: filter by a specific username
- `--license`: filter by a specific license type name

Cluster License Summary Query

Function Description

The `list-cluster-lic-summaries` command queries cluster license resource summary statistics and shows allocation summaries for each license type at the cluster level. This feature is based on SLURM's `fsched_list_licenses` API and provides authoritative license statistics.

Key features:

- Authoritative data: obtained from the SLURM native API
- Automatic aggregation: statistics from different servers for the same license type are aggregated
- Global view: provides a complete view of license usage inside and outside the cluster
- Resource monitoring: supports monitoring license quotas and actual usage

Query All License Summaries

Query summary statistics for all license types in the current cluster:

```
./stateclient list-cluster-lic-summaries
```

Sample output:

```
Sending requests to server localhost:20051
Cluster License Summaries (3):
========================================
License: matlab
  Used: 125
  Allowed: 180
  Total Consumed: 770
----------------------------------------
License: ansys
  Used: 30
  Allowed: 50
  Total Consumed: 45
----------------------------------------
License: nastran
  Used: 15
  Allowed: 100
  Total Consumed: 85
----------------------------------------
```

Parameters:

- `--license`: filter by a specific license type name

Field Descriptions

Used (Current Cluster Usage):

- The number of licenses allocated to running jobs within the current cluster
- Reflects actual license usage within the cluster

Allowed (Cluster Quota):

- Total license quota allocated to the current cluster
- Upper limit of licenses that cluster jobs can use

Total Consumed (Global Consumption):

- Total actual license consumption including the current cluster and outside the cluster
- Reflects actual usage on the license server

Cluster Quota Wildcard User Feature

The cluster quota feature is the management UI in the fsched platform for configuring various user quotas for a cluster. Slurm management is user-centric, and fsched introduces the concept of a wildcard user on top of this. A wildcard user is a special user in Slurm that matches all users, making it easy to manage quotas for multiple users.

Automatic Rules and Manual Rules

There are two categories of rules in cluster quotas: `automatic rules` and `manual rules`.

- Automatic rules: rules generated automatically by the system based on user quota information. By default, the system adds one automatic rule for each partition without a manual rule, allowing all users to submit jobs in that partition.
- Manual rules: rules configured manually by users, which can manage quotas for one or more users.

> TIP
>
> - Automatic rules are added only when no manual rules exist.
> - Once a user adds a manual rule to a partition, the automatic rule is removed.
> - When the manual rule is deleted, the automatic rule is re-added.

Quota Priority

Rules are divided into `wildcard rules` and `non-wildcard rules`:

- Wildcard rules: represented by `all users`, matching all users. Automatic rules only add wildcard rules.
- Non-wildcard rules: represented by specific usernames, matching specific users.

When a user submits a job, the system matches in the following priority order:

1. Non-wildcard rules
2. Wildcard rules

> TIP
>
> - Users can configure both wildcard and non-wildcard rules in cluster quotas.

- When users submit jobs, the system matches according to the priority above.

> ⚠ WARNING
>
> - If a job submitted by a user already matches a non-wildcard rule, the system will not check wildcard rules.

Example

Assume there are the following users and partitions:

- Users: User A, User B
- Partitions: Partition 1, Partition 2, Partition 3

Assume the rules are:

- User A has a manual rule (non-wildcard) in Partition 1.
- User B has a manual rule (non-wildcard) in Partition 2.
- Partition 2 has a manual wildcard rule.
- Partition 3 has an automatic wildcard rule.

In this case:

- User A
  - When submitting jobs in Partition 1, the system matches User A's manual non-wildcard rule first.
  - When submitting jobs in other partitions, the system uses wildcard rules.
- User B
  - Cannot submit jobs in Partition 1 because there is no matching rule.
  - When submitting jobs in Partition 2, the system matches User B's manual non-wildcard rule first.
  - When submitting jobs in Partition 3, the system matches wildcard rules.

Job Submission Lua Plugin

The Job Submit Plugin runs on the head node. It executes after a user's job request is submitted to the head node and before the job is actually queued.

Purpose

The Job Submit Plugin can do the following:

- Modify job submission information, such as the job's partition, cores, and memory amount.
- Check job information to determine whether it is allowed to run, and so on.

Installation and Configuration

Add the following to `slurm.conf`:

```
JobSubmitPlugins=job_submit/lua
```

Save the Lua script to `/etc/slurm/job_submit.lua`.

Supported Functions

```
-- Executed on job submission
function slurm_job_submit(job_desc, part_list, submit_uid)
end

-- Executed on job modification
function slurm_job_modify(job_desc, job_rec, part_list, modify_uid)
end
```

Parameters:

- `job_desc`: job description information
- `part_list`: all partition information
- `submit_uid`: UID of the submitting user
- `job_rec`: current job record (job_modify only)
- `modify_uid`: UID of the user performing the modification (job_modify only)

Data Structures

Job Descriptor (job_desc)

Field	Type	Description
account	string	Job account name
admin_comment	string	Admin comment
array_task_cnt	number	Number of array tasks
batch_features	string	Job batch features
burst_buffer	string	Job burst buffer
comment	string	Job comment
cpus_per_tres	string	CPUs per TRES (resource manager)
delay_boot	number	Delay boot time
direct_set_prio	number	Directly set priority
features	string	Job features
gres	string	Requested GRES (generic resources)
group_id	number	Job group ID
job_id	number	Job ID
job_state	number	Job state
licenses	string	Job licenses
max_cpus	number	Maximum CPUs available to the job
max_nodes	number	Maximum nodes available to the job

Field	Type	Description
mem_per_tres	string	Memory allocated per TRES
min_cpus	number	Minimum CPUs requested by the job
min_mem_per_node	number	Minimum memory requirement per node
min_mem_per_cpu	number	Minimum memory requirement per CPU
min_nodes	number	Minimum nodes requested by the job
name	string	Job name
nice	number	Job priority offset (scheduling)
pack_job_id	number	Pack job ID
pack_job_id_set	string	Pack job ID set
pack_job_offset	number	Pack job offset
partition	string	Job partition; note this is the requested value and may be empty
pn_min_cpus	number	Minimum CPUs per node
pn_min_memory	number	Minimum memory per node (64-bit integer)
priority	number	Job priority
qos	string	Job Quality of Service (QoS)
reboot	number	Job reboot setting
req_switch	number	Job requested switch information
site_factor	number	Site factor, usually for scheduling policies

Field	Type	Description
spank_job_env	table	SPANK (Slurm plugin) job environment variables
spank_job_env_size	number	Size of SPANK job environment variables
time_limit	number	Job time limit
time_min	number	Minimum job time limit
tres_bind	string	TRES binding information
tres_freq	string	TRES frequency information
tres_per_job	string	TRES per job
tres_per_node	string	TRES per node
tres_per_socket	string	TRES per socket
tres_per_task	string	TRES per task
user_id	number	User ID
user_name	string	User name
wait4switch	number	Job wait-for-switch status
wckey	string	Job workload key (workload key)

Partition (part_list)

Field	Type	Description
allow_accounts	string	Allowed account names
allow_alloc_nodes	string	Allowed node names

Field	Type	Description
allow_groups	string	Allowed group names
allow_qos	string	Allowed QoS settings
alternate	string	Alternate partition name
billing_weights_str	string	Partition billing weight string
default_time	number	Default time limit (minutes)
def_mem_per_cpu	number	Default memory per CPU (if allocated by CPU)
def_mem_per_node	number	Default memory per node
deny_accounts	string	Denied account names
deny_qos	string	Denied QoS settings
flag_default	number	Whether it is the default partition (0: no, 1: yes)
flags	number	Partition flags field (e.g., enabled)
max_cpus_per_node	number	Maximum CPUs allowed per node
max_mem_per_cpu	number	Maximum memory per CPU
max_mem_per_node	number	Maximum memory per node
max_nodes	number	Maximum nodes
max_nodes_orig	number	Original maximum nodes
max_share	number	Maximum sharing ratio
max_time	number	Maximum time limit (minutes)
min_nodes	number	Minimum nodes

Field	Type	Description
min_nodes_orig	number	Original minimum nodes
name	string	Partition name
nodes	string	Node list
priority_job_factor	number	Job priority factor
priority_tier	number	Priority tier
qos	string	QoS settings
state_up	number	Whether the partition is active (0: inactive, 1: active)

Global Object slurm

Log Functions

Field/Function	Description
log_error	Log error messages
log_info	Log informational messages (level 0)
log_verbose	Log verbose messages (level 1)
log_debug	Log debug messages (level 2)
log_debug2	Log more detailed debug messages (level 3)
log_debug3	Log even more detailed debug messages (level 4)
log_debug4	Log the most detailed debug messages (level 5)
log_user	Log user messages

Error Codes

Field/Function	Description
ERROR	Indicates a general error
FAILURE	Indicates operation failure
SUCCESS	Indicates operation success
ESLURM_ACCESS_DENIED	Access denied
ESLURM_ACCOUNTING_POLICY	Accounting policy error
ESLURM_INVALID_ACCOUNT	Invalid account
ESLURM_INVALID_LICENSES	Invalid licenses
ESLURM_INVALID_NODE_COUNT	Invalid node count
ESLURM_INVALID_TIME_LIMIT	Invalid time limit
ESLURM_JOB_MISSING_SIZE_SPECIFICATION	Job missing size specification
ESLURM_MISSING_TIME_LIMIT	Missing time limit

Other Definitions

Field/Function	Description
ALLOC_SID_ADMIN_HOLD	Allocation ID reserved by admin
ALLOC_SID_USER_HOLD	Allocation ID reserved by user
INFINITE	Infinite value
INFINITE64	64-bit infinite value
MAIL_JOB_BEGIN	Send mail when job begins

Field/Function	Description
MAIL_JOB_END	Send mail when job ends
MAIL_JOB_FAIL	Send mail when job fails
MAIL_JOB_REQUEUE	Send mail when job requeues
MAIL_JOB_TIME100	Send mail when job reaches 100% time
MAIL_JOB_TIME90	Send mail when job reaches 90% time
MAIL_JOB_TIME80	Send mail when job reaches 80% time
MAIL_JOB_TIME50	Send mail when job reaches 50% time
MAIL_JOB_STAGE_OUT	Send mail when job stage ends
MEM_PER_CPU	Memory per CPU
NICE_OFFSET	Priority offset
JOB_SHARED_NONE	Job not shared
JOB_SHARED_OK	Job shared
JOB_SHARED_USER	User shared job
JOB_SHARED_MCS	MCS shared job
NO_VAL64	64-bit invalid value
NO_VAL	Invalid value
NO_VAL16	16-bit invalid value
NO_VAL8	8-bit invalid value
SHARED_FORCE	Force sharing

Job Descriptor Bit Flags

Field/Function	Description
GRES_DISABLE_BIND	Disable resource binding
GRES_ENFORCE_BIND	Enforce resource binding
KILL_INV_DEP	Kill invalid dependencies
NO_KILL_INV_DEP	Do not kill invalid dependencies
SPREAD_JOB	Spread job
USE_MIN_NODES	Use minimum nodes

Job Time Limits

fsched allows users to specify the maximum runtime when submitting jobs. After the time limit is exceeded, the scheduler will forcibly terminate the job.

Partition Default Time Limit

Administrators can set a default job time limit on a partition. For example, you can set the default time limit for all jobs in a partition to 1 hour. Then all jobs submitted to that partition are subject to this limit. The partition default time limit uses the partition configuration parameter `DefaultTime` when the user does not specify a time limit. Supported formats:

- Minutes
- Minutes:Seconds
- Hours:Minutes:Seconds
- Days-Hours
- Days-Hours:Minutes
- Days-Hours:Minutes:Seconds
- UNLIMITED

Note: When specifying seconds, the actual limit may exceed the value, depending on controller load, possibly by several minutes.

Partition Maximum Time Limit

A partition can specify a maximum time limit. Users cannot exceed this limit when submitting or modifying job time limits. This parameter is `MaxTime`.

Supported formats:

- Minutes
- Minutes:Seconds
- Hours:Minutes:Seconds
- Days-Hours
- Days-Hours:Minutes
- Days-Hours:Minutes:Seconds
- UNLIMITED (default)

User-Defined Time Limit

If users want to adjust a job's time limit, they can use `scontrol update job` to modify the job's `TimeLimit` field, with a minimum unit of minutes. For example, to set the time limit to 2 hours:

```
scontrol update job <jobid> timelimit=02:00:00
```

By default, users can only shorten the time limit and cannot extend it. If users want to extend the time limit, set `SlurmctldParameters` to `allow_user_incr_time` in the global configuration. Then users can extend the `TimeLimit` field via `scontrol update job`, but still cannot exceed the partition maximum time limit.

Note: If fsched is managed by FCP / FCCE, set:
SlurmctldParameters=cloud_dns,nohold_on_prolog_fail,allow_user_incr_time

timelimit supports the following formats:

- Minutes
- Minutes:Seconds
- Hours:Minutes:Seconds
- Days-Hours
- Days-Hours:Minutes
- Days-Hours:Minutes:Seconds

Fairshare Scheduling Policy

Overview

Slurm uses the priority/multifactor plugin to implement multifactor job scheduling. The Fairshare policy is the core mechanism to ensure fair resource allocation. With proper configuration and use of Fairshare, you can significantly improve the fairness and efficiency of cluster resource utilization, ensuring that all users receive cluster resources in line with their shares. The Fairshare policy includes two key factors:

Fairshare Factor (resource allocation fairness factor)

- Purpose: considers the difference between committed resources and actually allocated resources for users/accounts.
- Principle: based on the resource allocation share (Share) for users/accounts and the actual allocated resources they receive.
- Effect: users who receive allocations beyond their committed share get lower priority.

FairshareUsed Factor (resource usage fairness factor)

- Purpose: considers the difference between committed resources and actually consumed resources for users/accounts.
- Principle: based on the resource allocation share (Share) for users/accounts and their actual resource consumption.
- Effect: users who consume more than their committed share get lower priority.
- Supported version: FairshareUsed Factor is supported starting from fsched-10.87.

Functional Details

Fairshare Factor mechanism

- Tracks resource allocation for users/accounts.
- Calculates a fairness index for resource allocation.
- Affects scheduling priority for new jobs.
- Ensures that, over the long term, each user/account receives resources consistent with their share.

FairshareUsed Factor mechanism:

- Tracks actual resource consumption for users/accounts (CPU hours, memory usage, etc.).
- Calculates a fairness index for resource usage.
- Affects scheduling priority for new jobs.
- Prevents inefficient resource usage.

Configuration Guide

Configure Association

- Configure the cluster-account-user associations in the accounting system.
- For the Fsched SE version: enable the "per-user resource limit" feature in the cluster management UI and set cluster resource quotas for each user.

Core Parameter Configuration

Add the following parameters to the `slurm.conf` configuration file:

```
# Enable the multifactor priority plugin
PriorityType=priority/multifactor

# Set fair scheduling weights (example values)
PriorityWeightFairshare=30      # Resource allocation fairness factor
weight
PriorityWeightFairshareUsed=3000  # Resource usage fairness factor weight
```

Parameter Notes:

In this example, `PriorityWeightFairshareUsed` is set to a higher value (3000), which makes actual resource consumption have a greater impact on job priority. Adjust the actual value based on your cluster's needs.

Weight Configuration Recommendations

- Adjust weights based on cluster characteristics. In production, it is recommended to establish a periodic review mechanism, analyze resource allocation/usage fairness monthly, and adjust share allocations based on usage.
- In resource-constrained environments, increase the FairshareUsed weight.
- If you want to prioritize allocation fairness, increase the Fairshare weight.

Monitoring Tools

Use sshare to Monitor Resource Shares

View detailed resource share information:

```
sshare -a --ext
```

Field descriptions:

- `RawShares`: raw share value
- `NormShares`: normalized share
- `RawUsage`: raw resource usage
- `EffectvUsage`: effective resource usage
- `FairShare`: Fairshare Factor value
- `FairShareU`: FairshareUsed Factor value

Use sprio to Analyze Job Priority

View job priority composition:

```
sprio --ext -l
```

Field descriptions:

- `PRIORITY`: total priority
- `FAIRSHARE`: Fairshare Factor contribution
- `FAIRSHAREU`: FairshareUsed Factor contribution
- Other standard priority factors

Usage Examples

Basic Environment Check

1. Confirm association configuration:

```
sacctmgr list assoc cluster=<cluster_name>
```

2. Check the current scheduling configuration:

```
scontrol show config | grep Priority
```

Fairness Tests

Test scenario 1: verify Fairshare Factor

- User alice submits multiple jobs that consume a lot of resources:

```
alice@ubuntu22-4c-1:~$ for i in `seq 100`;srun -c1 --exclusive sleep
3600 &;done
```

- Observe priority changes:

  Before and after the jobs complete, use `sshare --ext` and `sprio --ext` to view resource share and priority changes.

Test scenario 2: verify FairshareUsed Factor

- User alice submits jobs with low resource consumption:

```
alice@ubuntu22-4c-1:~$ srun -n4 sleep 100&
[1] 2512319
alice@ubuntu22-4c-1:~$ srun -n4 sleep 100&
[2] 2512372
```

- User charlie submits jobs with high resource consumption:

```
charlie@ubuntu22-4c-2:~$ srun stress-ng --cpu 1 --cpu-load 100 -t 100s
&
[1] 164380
charlie@ubuntu22-4c-2:~$ srun stress-ng --cpu 1 --cpu-load 100 -t 100s
&
[2] 164387
```

- Observe priority changes:

Before and after the jobs complete, use `sshare --ext` and `sprio --ext` to view resource share and priority changes.

Notes

- Version compatibility: If you downgrade from this version to an older fsched, the `slurmctld` service will fail to start due to an incompatible state file version. Example error: `Can not recover assoc_usage state, incompatible version`. Solution: manually delete the `assoc_usage` file under the cluster state directory, then restart `slurmctld`.

- Algorithm limitation: FairshareUsed supports only the Fair Tree fairshare algorithm, and does not support the classic fairshare algorithm.

- Weight impact: A very high FairshareUsed weight may give overly high priority to users with low resource usage. Determine the optimal weight ratio through testing.

- Data accuracy: Ensure accounting data collection is working properly, and periodically verify the accuracy of resource usage statistics.

Partition AllowUsers

fsched (fsched-dev(FIXME)+) supports setting allowed users by partition.

Purpose

After configuring allowed users by partition, only the allowed users can run jobs in that partition.

> TIP
>
> For a partition configured with `AllowUsers=user1`:
>
> - Only `user1` can submit new jobs.
> - Jobs already running by other users before this parameter was configured are not affected.
> - Pending jobs from other users submitted before this parameter was configured will remain pending. Because of the user restriction (`REASON` is `PartitionConfig`), they cannot run even if resources are available.

Partition Configuration

In `partitions.conf`, add the configuration item on the line that starts with `PartitionName` for the partition you want to configure.

Parameter	Description	Value Type and Range	Default
`AllowUsers`	Only allowed users can run jobs in the partition	Comma-separated list of usernames	Default is `ALL`, meaning all users can submit jobs to the partition

Example

1. Modify the partition configuration that needs allowed users and add:

```
AllowUsers=test
```

2. After the cluster reconfiguration takes effect, run jobs in the partition with different users. The allowed user can run, and other users cannot.

```
[test@centos7-16c-1 ~]$ srun hostname
srun: job 33 queued and waiting for resources
srun: job 33 has been allocated resources
centos7-64c-6

[jj@centos7-16c-1 ~]$ srun hostname
srun: E: slurm_allocate_resources_blocking: RESPONSE_SLURM_RC:
Access/permission denied
srun: E: Unable to allocate resources: Access/permission denied
```

Partition OverMemoryKill

fsched supports a per-partition policy: kill a job when it exceeds the requested memory. Configure it on the line that starts with `PartitionName` for the partition in `partitions.conf`. The options include:

1. Kill jobs on memory overuse: `OverMemoryKill`

- If the partition where the job runs is configured with `OverMemoryKill=YES`, the job will be killed when the memory it uses exceeds the memory it requested.

Notes:

1. Killing jobs on memory overuse can be delayed, and it does not guarantee the system will never OOM.
2. If the task plugin is configured as cgroup, cgroup is used to kill jobs on memory overuse; otherwise jobacct is used.
3. cgroup currently does not work on Ubuntu 22.
4. If no memory request is specified when submitting a job, the default memory is used as the requested memory.

Example

```
root@head1:~# cat /etc/slurm/partitions.conf

#
# PARTITION partition-8PSBV
PartitionName=partition-8PSBV Nodes=compute1,compute2 Default=YES
OverMemoryKill=YES
#   DUMMY

#   NODES
NodeName=compute1 CPUs=16 RealMemory=13926  Weight=1 State=CLOUD
NodeName=compute2 CPUs=16 RealMemory=13926  Weight=1 State=CLOUD


root@head1:~# srun --mem=3 hostname
srun: Exceeded job memory limit
Jul 17 15:43:46.625179 206113 slurmstepd   0x14becbec1b80: E: Step 463.0
exceeded memory limit (7354368 > 3145728), being killed
compute1
```

Partition Administrators

By default, administrative operations on the cluster can only be performed by the super administrator. In large environments, the super administrator may not be able to handle all administrative operations. Therefore, we introduce the concept of partition administrators. A partition administrator is a special user who can perform administrative operations on a specific partition in the cluster, but cannot administer the entire cluster. Currently, partition administrators can perform the following operations on a partition:

- Cancel jobs in the partition.
- Set whether the partition can accept new jobs (DRAIN/UP).

Supported Versions

10.61 and later

Usage

- Partition administration is configured with the partition `Admins` parameter. This parameter supports multiple partition administrators. The format is Linux usernames, separated by commas (,).
- Users with partition administrator privileges can use `scancel` to cancel other users' jobs in the corresponding partition.
- Users with partition administrator privileges can use `bkill` and `qdel` in the wrappers to cancel other users' jobs in the corresponding partition.
- Users with partition administrator privileges can use `scontrol update partition=<partition_name> state=<UP/DRAIN>` to set whether the partition can accept new jobs.

Example

Assume there is an admin user `admin`. We can configure them as the administrator of partition `compute` as follows:

```
PartitionName=compute Nodes=compute[1-3] ... Admins=admin
```

> ⚠ WARNING

- Partition administrators can only perform management operations on their partitions, not on the entire cluster.
- If a partition administrator is configured incorrectly or the username cannot be resolved, the cluster controller will fail to start.
- When using `scontrol`, avoid updating other partition options at the same time; otherwise the state update will fail.

Overview

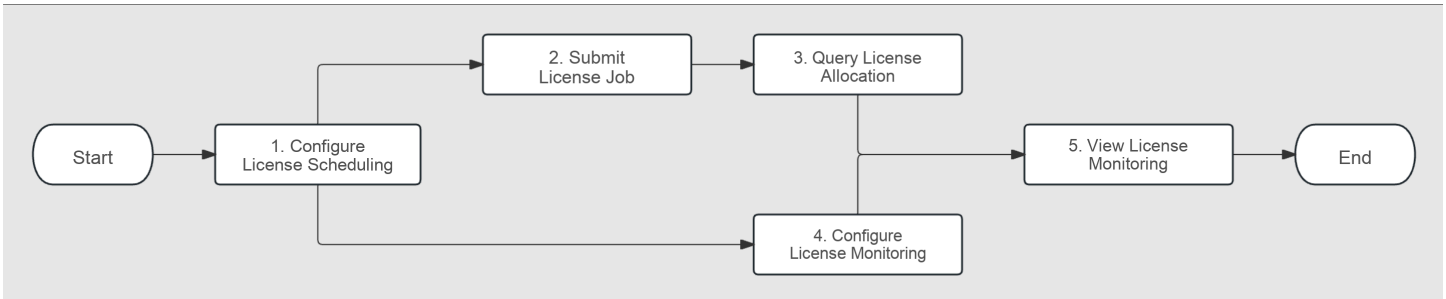
What Is Application License Scheduling?

The Fsched scheduler treats application software licenses as shared cluster resources for unified scheduling and management. By monitoring license usage in real time, it automatically adjusts the number of licenses available to jobs, ensuring licenses are correctly allocated during scheduling.

Core Value

- Resource optimization: avoid idle or overused licenses
- Automatic management: real-time monitoring and automatic adjustment of license allocation
- Job assurance: ensure jobs receive the required licenses during scheduling

Typical Workflow



Configure License Scheduling

Prerequisites

Network connectivity requirements:

- Network connectivity between the cluster head node and the license server must be normal.
- Ensure the firewall allows communication on relevant ports (for example, FlexLM default port 27000).

Permission requirements:

- □ Important: Administrator privileges are required.
- Changes to license scheduling configuration directly affect job scheduling. Operate with caution.

Management Tools

Fsched provides two core configuration tools:

Tool	Function Description	Comparable Tool
License Server Management	Configure and monitor external license servers	Similar to system monitoring tools
License Allocation Management	Manage license allocation policies within the cluster	Directly use the `sacctmgr` resource command

Note: The license server client query tool is uploaded to the cluster head node by the administrator and is set via the `--command-path` parameter when adding a license server.

License Server Management

Function Description

Used to configure and monitor external license servers. When external license servers change, the cluster's available license count is automatically adjusted. Supports major license systems such as FlexLM and RLM.

Add a License Server

Command example:

```
./stateclient add-lic-server \
  --addr flex_host:27000 \
  --command-path /usr/local/bin/lmstat \
  --tracked-lics matlab,nastran,ansys \
  --type flexlm \
  --name matlab_server
```

Parameter details:

Parameter	Description	Required	Default
`--addr`	License server address, format: `hostname:port`	□ Required	None
`--command-path`	Full path to the license query command (such as lmstat)	□ Required	None
`--tracked-lics`	List of license names to monitor, comma-separated	□ Required	None
`--type`	License server type	□ Optional	"flexlm"
`--name`	Server identifier name for management	□ Optional	"host_port" format

Notes:

- The license server name must be unique and correspond one-to-one with the server address `host:port`. The default is `host_port`.
- Because fsched converts license names and server names to lowercase, users must submit license names and server names in lowercase when submitting jobs.

Practical use cases:

- Add a new license server to the monitoring system.
- Expand the list of monitored license features.

Remove a License Server

Command example:

```
./stateclient remove-lic-server --addr flex_host:27000
```

Parameter details:

Parameter	Description	Required
`--addr`	Server address to remove	□ Required

Notes:

- Removing a server also removes monitoring for all licenses on that server.
- Ensure no jobs are using licenses from that server.

View License Server Status

Command example:

```
./stateclient list-lic-servers
```

Output includes:

- Server address and name
- Server type and status
- Monitored license list
- Connection status (OK/ERROR)

License Allocation Management

Function Description

Manage license allocation policies within the cluster, controlling the number and percentage of licenses available to each cluster.

Get the Current Cluster Name

```
scontrol show config | grep ClusterName
```

Add License Allocation

1. Check whether the license already exists.

```
sacctmgr show resource name=test8 server=server3
```

2. If it does not exist, you can add it in one step:

```
sacctmgr add resource name=test8 server=server3 servertype=flexlm
type=license count=60 cluster=fastone-1 percentallowed=70
```

3. If it already exists:

If the license already exists, you cannot set a new `count`. If you need to modify it, use the `sacctmgr modify resource` command.

3.1 Check whether there is an allocation for the current cluster:

```
sacctmgr show resource withclusters name=test8 server=server3
cluster=fastone-1
```

3.2 If there is no allocation for the current cluster, you can set `percentallowed` for the current cluster:

```
sacctmgr add resource name=test8 server=server3 cluster=fastone-1
percentallowed=70
```

3.3 If there is already an allocation for the current cluster, you cannot set it again. If you need to modify it, use the `sacctmgr modify resource` command.

Parameter descriptions:

- `name`: license name
- `cluster`: fsched cluster name
- `count`: total number of licenses, including inside and outside the cluster
- `percentallowed`: percentage allowed for the current cluster

- `server`: license server name
- `servertype`: license server type
- `type`: resource name, should be `license` in this workflow

Modify License Allocation

- To modify the total license count:

```
sacctmgr modify resource name=matlab server=server2 set count=200
```

- To modify `percentallowed` for the current cluster:

```
sacctmgr modify resource name=matlab server=server2 cluster=fastone-1 set
percentallowed=30
```

Parameter descriptions:

- `name`: license name
- `server`: server name
- `cluster`: fsched cluster name
- `count`: new total license count
- `percentallowed`: new percentage

Delete License Allocation

- If there is only one cluster, or you want to delete allocations for all clusters:

```
sacctmgr delete resource where name=matlab server=server2
```

- If there are multiple clusters and you want to delete the allocation for the current cluster:

```
sacctmgr delete resource where name=matlab server=server2 cluster=fastone-1
```

Parameter descriptions:

- `name`: license name
- `server`: server name

- `cluster`: fsched cluster name

Query License Allocation

- Query allocations for the current cluster:

```
# Query all license allocations
sacctmgr show resource withclusters cluster=fastone-1

# Filter by name
sacctmgr show resource withclusters cluster=fastone-1 name=nastran

# Filter by server
sacctmgr show resource withclusters cluster=fastone-1 server=flex_host

# Exact query
sacctmgr show resource withclusters cluster=fastone-1 name=matlab
server=rlm_host
```

- Query all license allocations:

```
sacctmgr show resource
```

Parameter descriptions:

- `name`: filter by license name
- `server`: filter by server name
- `cluster`: fsched cluster name

Complete Configuration Example

Scenario Description

Configure monitoring and allocation for Synopsys APEX20K and primesim licenses.

Step 1: Check Existing Configuration

- View the current cluster name:

```
[root@centos7-16c-1 ~]# scontrol show config | grep ClusterName
ClusterName             = fastone-1
```

- View allocated licenses:

```
[root@centos7-16c-1 ~]# sacctmgr show resource
     Name     Server     Type  Count % Allocated ServerType
---------- ---------- -------- ------ ----------- ----------
```

The output shows that there are no license allocations configured yet.

Step 2: Determine Total License Count

Use the lmstat command to view the actual license count:

```
[root@centos7-16c-1 ~]# ./lmstat -f APEX20K -c 27000@10.106.32.69
lmstat - Copyright (c) 1989-2016 Flexera Software LLC. All Rights Reserved.
Flexible License Manager status on Mon 9/8/2025 11:43

License server status: 27000@centos7-64c-9
    License file(s) on centos7-64c-9: /fastone/software/lic/Synopsys.dat:

centos7-64c-9: license server UP (MASTER) v11.14.0

Vendor daemon status (on centos7-64c-9):

   snpslmd: UP v11.14.0
Feature usage info:

Users of APEX20K:  (Total of 999 licenses issued;  Total of 0 licenses in
use)
```

Key information: `Total of 999 licenses issued` *means there are 999 total licenses.*

Step 3: Configure License Allocation

Allocate license usage permissions for the cluster:

```
[root@centos7-16c-1 ~]# sacctmgr add resource name=APEX20K
server=10.106.32.69_27000 servertype=flexlm type=license count=999
cluster=fastone-1 percentallowed=100
```

```
Adding Resource(s)
  apex20k@10.106.32.69_27000
  Cluster - fastone-1  100%
Settings
  Name           = apex20k
  Server         = 10.106.32.69_27000
  Description    = apex20k
  ServerType     = flexlm
  Count          = 999
  Type           = License
Would you like to commit changes? (You have 30 seconds to decide)
(N/y): y
[root@centos7-16c-1 ~]# sacctmgr add resource name=primesim
server=10.106.32.69_27000 servertype=flexlm type=license count=999
cluster=fastone-1 percentallowed=100
Adding Resource(s)
  primesim@10.106.32.69_27000
  Cluster - fastone-1  100%
Settings
  Name           = primesim
  Server         = 10.106.32.69_27000
  Description    = primesim
  ServerType     = flexlm
  Count          = 999
  Type           = License
Would you like to commit changes? (You have 30 seconds to decide)
(N/y): y
```

View allocation results:

```
[root@centos7-16c-1 ~]# sacctmgr show resource
      Name     Server     Type  Count % Allocated ServerType
---------- ---------- -------- ------ ----------- ----------
  apex20k 10.106.32+  License    999         100     flexlm
  primesim 10.106.32+  License    999         100     flexlm
```

Step 4: Configure License Monitoring

Check existing monitoring configuration:

```
[root@centos7-16c-1 ~]# /opt/fsched/wrappers/statesvc/bin/stateclient list-
lic-servers
Sending requests to server localhost:20051
```

```
License Servers (0):
========================================
```

Add license server monitoring:

```
[root@centos7-16c-1 ~]# /opt/fsched/wrappers/statesvc/bin/stateclient add-lic-server --addr 10.106.32.69:27000 --command-path /root/lmstat --tracked-lics APEX20K,primesim
Sending requests to server localhost:20051
Successfully added license server: 10.106.32.69:27000 (auto-generated-name) with 2 tracked licenses
```

Verify monitoring configuration:

```
[root@centos7-16c-1 ~]# /opt/fsched/wrappers/statesvc/bin/stateclient list-lic-servers
Sending requests to server localhost:20051
License Servers (1):
========================================
Server: 10.106.32.69:27000
  Name: 10.106.32.69_27000
  Type: flexlm
  Command Path: /root/lmstat
  Tracked Licenses (2): apex20k, primesim
  Status: OK
----------------------------------------
```

Troubleshooting Guide

Q: Connection fails when adding a server

- Check network connectivity: `ping license_server`
- Verify port access: `telnet license_server 27000`
- Confirm the lmstat command path is correct

Q: License monitoring status is abnormal

- Check whether the license server is running normally
- Verify that the tracked-lics names match actual license feature names
- Confirm sufficient permissions to execute the monitoring command

Q: Jobs cannot obtain licenses

- Check whether license allocation configuration exists
- Verify that the license counts are sufficient
- Check whether license monitoring status is normal

Best Practices

1. Naming conventions: use meaningful names for license servers
2. Capacity planning: set `percentallowed` appropriately to avoid resource contention
3. Monitoring coverage: ensure all critical license features are within the monitoring scope
4. Regular checks: regularly verify consistency between license allocation and actual usage

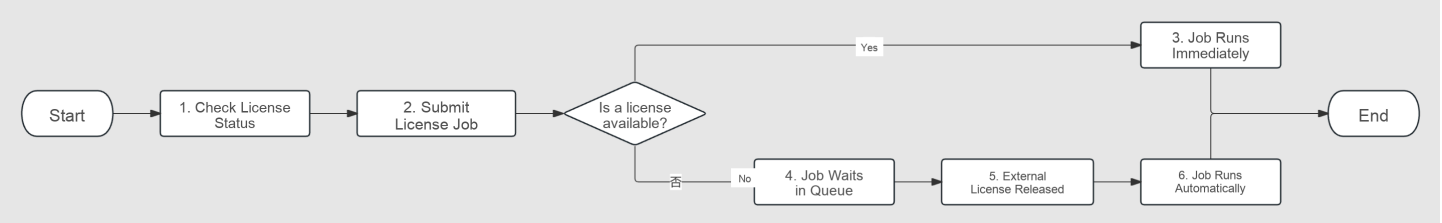
Submit License Jobs

A license job is a compute task that requires a specific software license to run. The Fsched scheduler ensures jobs run only after the required license resources are obtained through intelligent license management.

Core features:

- Resource awareness: real-time monitoring of license availability
- Smart queuing: automatically waits when licenses are insufficient
- Dynamic allocation: automatically obtains licenses after external licenses are released
- Flexible configuration: supports both CLI and template submission methods

License job workflow:



Prerequisites

- License resources have been configured. See Configure License Scheduling

CLI Submission Method

Check License Availability

Command syntax:

```
scontrol show lic --ext
```

Sample output and interpretation:

```
[root@centos7-16c-1 ~]# scontrol show lic --ext
LicenseName=primesim@10.106.32.69_27000
    Total=999 Used=0 Free=999 Reserved=0 Remote=yes
    LastConsumed=0 LastDeficit=0 LastUpdate=Unknown
```

```
LicenseName=apex20k@10.106.32.69_27000
    Total=999 Used=0 Free=99 Reserved=0 Remote=yes
    LastConsumed=900 LastDeficit=900 LastUpdate=Unknown
```

Field details:

Field	Description	Example Value Analysis
`LicenseName`	License identifier: `license_name@server_name`	`apex20k@10.106.32.69_27000`
`Total`	Total number of licenses	999
`Used`	Used within the current cluster	0 (no usage in the cluster)
`Free`	Remaining available in the cluster	99 (key decision indicator)
`Reserved`	Reserved license count	0
`Remote`	Whether it is a remote license	yes (from external server)
`LastConsumed`	Total consumption (inside + outside cluster)	900 (900 used externally)
`LastDeficit`	Most recent shortage amount	900 (short by 900)

Key insights:

- `Free=99` indicates the cluster currently has 99 apex20k licenses available
- `LastConsumed=900` indicates external systems are using 900 licenses
- Total licenses: 999; available to the cluster: 99; external usage: 900

Submit a License Job

Basic command syntax:

```
srun -L <license_name>@<server_name>:<count> <command>
```

Example:

```
# Request 100 apex20k licenses to run a task
[root@centos7-16c-1 ~]# srun -L apex20k@10.106.32.69_27000:100 sleep 600&
[1] 7070
```

Job Status Monitoring

Status when licenses are insufficient:

```
[root@centos7-16c-1 ~]# squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES
NODELIST(REASON)
                80 partition    sleep     root PD       0:00      1
(Licenses)
```

Status explanation:

- `ST=PD`: pending state
- `REASON=(Licenses)`: waiting due to insufficient license resources
- Job ID 80 is waiting for 100 licenses, but only 99 are currently available

License Release and Job Execution

Status after external licenses are released:

```
# Job starts running
[root@centos7-16c-1 ~]# squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES
NODELIST(REASON)
                80 partition    sleep     root  R       5:21      1
centos7-64c-6

# License status update
[root@centos7-16c-1 ~]# scontrol show lic --ext
LicenseName=apex20k@10.106.32.69_27000
    Total=999 Used=100 Free=899 Reserved=0 Remote=yes
    LastConsumed=0 LastDeficit=0 LastUpdate=Unknown
```

Status change analysis:

- `Used=100`: the job is using 100 licenses

- `Free=899`: available count decreases to 899
- `LastConsumed=0`: external usage cleared (possibly released)
- Job status changes from `PD` to `R` (Running)

Template Submission Method

Applicable Scenarios

Advantages of job templates:

- □ Automated calculation: dynamically determine license requirements based on job size
- □ Unified policy: ensure consistency of license allocation
- □ Simplified operation: users do not need to manually calculate license counts
- □ Resource optimization: avoid over- or under-requesting licenses

Template Configuration Example

Ansys HPC Pack license template:

```
#SBATCH -L ansys@se-lic:{% set cpus = (np | toInt) * (node_num | toInt) %}
{% if cpus <= 4 %}0{% elseif cpus <= 12 %}1{% elseif cpus <= 36 %}2{%
elseif cpus <= 132 %}3{% elseif cpus <= 516 %}4{% else %}0{% endif %}
```

Template logic breakdown:

CPU core range	Licenses consumed	Applicable scenario
≤ 4 cores	0	Small compute tasks
5-12 cores	1	Medium compute tasks
13-36 cores	2	Large compute tasks
37-132 cores	3	Extra-large compute tasks
133-516 cores	4	Extreme scale compute
>516 cores	0	Special handling

Calculation logic:

- `cpus = np × node_num`: total CPU cores
- Determine license count based on core range
- This example applies to the core-based licensing mode of Ansys HPC Pack

Template Configuration UI

```
parameters:
  - name:  node_num
    option:
      enum:
        - value: 3
        - value: 4
          display: 4
outputs:
  - path: taskout.log
    type: TEXT
script: |-
    #!/bin/bash
    #SBATCH -L ansys@se-lic:{% set cpus = (np | toInt) * (node_num | toInt) %}{% if cpus <= 4 %}0{% elseif cpus <= 12 %}1{% elseif cpus <= 36 %}2{% el
    set -ex

    # 设置license环境变量
    export ANSYSLMD_LICENSE_FILE=1055@10.105.1.9

    # 应用command的绝对路径
    NODELIST=/fastone/softwares/nodelist
    LOGFILE=taskout.log

    case {{ version }} in

        2020)
            AppFolder=""
            ;;
        2021)
            AppFolder="/fastone/softwares/ansys/ansys/2021/ansys_inc/v211/fluent"
            ;;
        2022)
```

UI steps:

1. Select or create a job template
2. Add license configuration parameters in the Script section
3. Save the template configuration

Submit Jobs Using Templates

Workflow:

1. Log in to the job submission UI
2. Select the configured license template
3. Set job parameters (node count, process count, etc.)
4. Submit the job; the system automatically calculates the required license count

Benefits:

- Users do not need to understand the specific license calculation rules
- The system automatically allocates an appropriate number of licenses based on job size
- Ensures optimized license usage

Best Practices Guide

License Request Strategy

Reasonable estimation:

```
# Recommended: request based on actual need
srun -L apex20k@server:50 simulation_job

# Avoid: over-requesting
srun -L apex20k@server:200 simulation_job  # may waste resources

# Avoid: under-requesting
srun -L apex20k@server:5 simulation_job   # may impact performance
```

Job Queue Optimization

Monitoring and adjustment:

- Check `scontrol show lic --ext` regularly to understand license status
- Use `squeue` to monitor job wait status
- Adjust submission time or job size based on queue conditions

Template Design Recommendations

Customized template rules:

```
# Design templates based on software characteristics
#SBATCH -L <software>@<server>:{% if task_type == "small" %}1{% elseif
task_type == "medium" %}2{% else %}4{% endif %}
```

Troubleshooting

Issue 1: Jobs wait a long time for licenses

```
# Check current license status
scontrol show lic --ext

# Check external license usage
./lmstat -f <feature> -c <server>
```

Issue 2: Incorrect license count calculation

- Verify template logic is correct
- Check node and process count settings
- Confirm license calculation rules match actual software requirements

Issue 3: License server connection failure

- Verify network connectivity
- Check license server status
- Confirm server address and port are correct

Advanced Features

Jobs with Multiple Licenses

Complex job scenario:

```
# Request multiple licenses at the same time
srun -L ansys@server1:2,matlab@server2:1 complex_simulation
```

Query License Allocation

Fsched provides complete license query capabilities, supporting real-time monitoring of license usage from both user allocation and cluster summary perspectives. This helps administrators and users fully understand license resource status.

Notes

Important performance reminder

Query frequency limits:

- License allocation data is updated every 30 seconds; the recommended query interval is no less than 30 seconds
- Frequent queries may impact system performance; query as needed

Memory consumption warning:

The `list-user-lic-allocs` command may consume significant memory when there are many users and license types. Ensure sufficient memory is reserved:

Cluster Size	Record Count	Estimated Memory	Recommended Reserved Memory
Small/Medium	1,000 users × 300 licenses	~121MB	242MB
Medium	1,000 users × 500 licenses	~167MB	334MB
Large	1,000 users × 1,000 licenses	~335MB	670MB
Extra Large	2,000 users × 1,000 licenses	~669MB	1.3GB

Usage tips

- All commands support `--help` for detailed instructions

- Use filters to reduce data volume
- In production, set up query monitoring and alerts

Prerequisites

- License resources have been configured. See Configure License Scheduling

User License Allocation Query

Function Description

Command: `list-user-lic-allocs`

Core value:

- Real-time monitoring: dynamically calculated from running job data
- Automatic aggregation: same license types across servers are automatically aggregated
- Precise filtering: only shows licenses actually in use
- Simplified processing: no client-side data merging required

Data sources:

- Real-time job runtime data
- Active license allocations in the current cluster
- Automatic filtering of zero-usage records

Query All User Allocations

Basic command:

```
./stateclient list-user-lic-allocs
```

Sample output and analysis:

```
Sending requests to server localhost:20051
User License Allocations (4):
========================================
User: alice
  License: matlab
```

```
  Count: 8
----------------------------------------
User: alice
  License: ansys
  Count: 3
----------------------------------------
User: bob
  License: matlab
  Count: 2
----------------------------------------
User: charlie
  License: nastran
  Count: 5
----------------------------------------
```

Interpretation:

- User alice: 8 MATLAB licenses + 3 ANSYS licenses
- User bob: 2 MATLAB licenses
- User charlie: 5 NASTRAN licenses
- Total: 4 active allocation records

Filtered Queries

Filter by user:

```
# Query license allocations for a specific user
./stateclient list-user-lic-allocs --user alice
```

Sample output:

```
User: alice
  License: matlab
  Count: 8
----------------------------------------
User: alice
  License: ansys
  Count: 3
----------------------------------------
```

Filter by license type:

```
# Query usage for a specific license type
./stateclient list-user-lic-allocs --license matlab
```

Sample output:

```
User: alice
  License: matlab
  Count: 8
----------------------------------------
User: bob
  License: matlab
  Count: 2
----------------------------------------
```

Combined filters:

```
# Query a specific license used by a specific user
./stateclient list-user-lic-allocs --user alice --license ansys
```

Cluster License Summary Query

Function Description

Command: `list-cluster-lic-summaries`

Core value:

- Authoritative data: based on SLURM native `fsched_list_licenses` API
- Global view: provides complete license usage inside and outside the cluster
- Resource monitoring: supports quota and utilization monitoring
- Automatic aggregation: license data across servers is automatically merged

Data sources:

- SLURM core license management API
- Real-time license server status
- Usage statistics inside and outside the cluster

Query Cluster Summary

Basic command:

```
./stateclient list-cluster-lic-summaries
```

Sample output and deep analysis:

```
Sending requests to server localhost:20051
Cluster License Summaries (3):
=======================================
License: matlab
  Used: 125
  Allowed: 180
  Total Consumed: 770
---------------------------------------
License: ansys
  Used: 30
  Allowed: 50
  Total Consumed: 45
---------------------------------------
License: nastran
  Used: 15
  Allowed: 100
  Total Consumed: 85
---------------------------------------
```

Field Details

Used (Current Cluster Usage)

- Definition: number of licenses allocated to running jobs within the current cluster
- Calculation: ∑ (licenses requested by all running jobs)
- Monitoring focus: actual usage inside the cluster
- Example: MATLAB has 125 licenses in use by cluster jobs

Allowed (Cluster Quota)

- Definition: total license quota allocated to the current cluster
- Configuration source: License Allocation management settings
- Business meaning: maximum number of licenses cluster jobs can use

- Example: MATLAB quota is 180; ANSYS quota is 50

Total Consumed (Global Consumption)

- Definition: total actual license consumption including the current cluster and outside the cluster
- Data source: real-time monitoring of license servers
- Business meaning: reflects true pressure on license resources
- Example: MATLAB total consumption is 770, meaning external systems used 645

Filtered Queries

Filter by license type:

```
# Query summary information for a specific license type
./stateclient list-cluster-lic-summaries --license matlab
```

Sample output:

```
License: matlab
  Used: 125
  Allowed: 180
  Total Consumed: 770
----------------------------------------
```

Use Cases and Best Practices

Daily Monitoring

Resource utilization monitoring:

```
# Daily resource usage check
./stateclient list-cluster-lic-summaries

# Calculate utilization: Used / Allowed × 100%
# MATLAB: 125/180 = 69.4% utilization
# ANSYS: 30/50 = 60% utilization
```

User behavior analysis:

```
# Monitor high-consumption users
./stateclient list-user-lic-allocs | sort -k6 -nr
```

Capacity Planning

Quota adjustment decisions:

```
# Analyze quota usage
./stateclient list-cluster-lic-summaries --license ansys
```

Decision basis:

- Sustained utilization >80% → consider increasing quota
- Total Consumed close to total licenses → consider purchasing more licenses

Troubleshooting

Resource contention analysis:

```
# Check which users are consuming many resources
./stateclient list-user-lic-allocs --license matlab

# Verify cluster quota settings
./stateclient list-cluster-lic-summaries --license matlab
```

View License Monitoring

The license monitoring feature provides a graphical interface to monitor license usage, helping users and administrators understand application license usage status and the health of management services in real time. It supports both online viewing and offline analysis.

Prerequisites

- The application license monitoring feature requires enabling the application license management service monitoring in advance. For details, see FCP -> Deployment Guide -> Environment Configuration -> Monitoring Configuration.
- Access permissions for the Cluster Analysis module and the License Monitoring Analysis page
- Login permissions for the FCP web management system

Steps

1. In the FCP web management system, click Operations -> Analysis & Reports -> Cluster Analysis.

2. Select the "License Monitoring Analysis" tab to enter the application license monitoring page.

3. On the License Monitoring Analysis page, view information about application licenses and the application license management service.

   The system supports online and offline viewing of monitoring data. For offline viewing, click "Export to CSV" or "Export to Image" in the top-right corner to export monitoring data locally.

Per-Partition CPU Pinning

`fsched` supports CPU pinning by partition. When a job is submitted to the corresponding partition, pinning is applied automatically. The approach is to disable global CPU pinning, but enable thread pinning for specific partitions. Configure it as follows.

Notes

- This configuration requires restarting `slurmd`. It does not affect running jobs and only applies to new jobs.
- It is recommended to drain all machines before applying the change so that no jobs are scheduled during the modification.

Global Configuration

Use the following configuration to enable `task/affinity` and disable global CPU pinning.

```
TaskPlugin = task/affinity
TaskPluginParam = None
```

Partition Configuration

For the partitions where you want pinning, set partition-level pinning.

```
CpuBind = Thread
```

Apply

Restart `slurmd` and `slurmctld`.

Change Job Resource Requests

fsched (fsched-10.61+) supports changing resource requests for running jobs. Currently supported parameters include:

1. `MinMemoryNode`
   - Change the minimum memory required per node for a job (in MB).
2. `MinCPUsNode`
   - Change the minimum CPU count required per node for a job.

Purpose

By changing resource requests for running jobs, you can reduce over-allocated resources during execution so that other jobs pending due to insufficient resources can run.

> ⚠ WARNING
>
> - Reducing the requested resources while a job is running does not reduce the job's actual resource usage, so other jobs may cause node resource load to become too high.

> 💡 TIP
>
> - The root user can modify resource requests for any user's jobs.
> - Regular users can modify resource requests only for their own jobs.
> - Currently `MinCPUsNode` only supports decreasing the value; increasing it will produce an error (the dev version will add support for increasing it).

Cluster Configuration

Modify the following configuration to use the plugin `select/cons_tres_ex`, which supports changing resource requests for running jobs.

```
SelectType=select/cons_tres_ex
```

Example 1: How to Modify

1. Submit a job, wait for it to run, and modify `MinMemoryNode`.

```
[root@head-1 ~]# srun -w compute-1 --mem 8000 sleep 120&
[1] 17033
[root@head-1 ~]# squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES
NODELIST(REASON)
             289 partition    sleep     root  R       0:02      1
compute-1
```

```
[root@head-1 ~]# scontrol update job 289 MinMemoryNode=3000
```

2. View the modified `MinMemoryNode`.

```
[root@head-1 ~]# scontrol show job 289
JobId=289 JobName=sleep
UserId=root(0) GroupId=root(0) MCS_label=N/A
Priority=4294901751 Nice=0 Account=root QOS=normal WCKey=*
JobState=RUNNING Reason=None Dependency=(null)
Requeue=1 Restarts=0 BatchFlag=0 Reboot=0 ExitCode=0:0
RunTime=00:00:24 TimeLimit=UNLIMITED TimeMin=N/A
SubmitTime=2024-12-05T14:01:30 EligibleTime=2024-12-05T14:01:30
AccrueTime=Unknown
StartTime=2024-12-05T14:01:30 EndTime=Unknown Deadline=N/A
SuspendTime=None SecsPreSuspend=0 LastSchedEval=2024-12-05T14:01:30
Partition=partition-9C3RA AllocNode:Sid=head-1:14475
ReqNodeList=compute-1 ExcNodeList=(null)
NodeList=compute-1
BatchHost=compute-1
NumNodes=1 NumCPUs=1 NumTasks=1 CPUs/Task=1 ReqB:S:C:T=0:0:*:*
TRES=cpu=1,mem=3000M,node=1,billing=1
Socks/Node=* NtasksPerN:B:S:C=0:0:*:* CoreSpec=*
MinCPUsNode=1 MinMemoryNode=3000M MinTmpDiskNode=0
Features=(null) DelayBoot=00:00:00
OverSubscribe=OK Contiguous=0 Licenses=(null) Network=(null)
Command=sleep 120
WorkDir=/root
Power=
```

3. Submit a job, wait for it to run, and modify `MinCPUsNode`.

```
[root@head-1 ~]# srun -w compute-1 --mincpus=3 sleep 120&
[1] 18209
[root@head-1 ~]# squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES
NODELIST(REASON)
               290 partition    sleep     root  R       0:02      1
compute-1
```

```
[root@head-1 ~]# scontrol update job 290 MinCPUsNode=2
```

4. View the modified `MinCPUsNode`.

```
[root@head-1 ~]# scontrol show job 290
JobId=290 JobName=sleep
UserId=root(0) GroupId=root(0) MCS_label=N/A
Priority=4294901750 Nice=0 Account=root QOS=normal WCKey=*
JobState=RUNNING Reason=None Dependency=(null)
Requeue=1 Restarts=0 BatchFlag=0 Reboot=0 ExitCode=0:0
RunTime=00:00:28 TimeLimit=UNLIMITED TimeMin=N/A
SubmitTime=2024-12-05T14:09:32 EligibleTime=2024-12-05T14:09:32
AccrueTime=Unknown
StartTime=2024-12-05T14:09:32 EndTime=Unknown Deadline=N/A
SuspendTime=None SecsPreSuspend=0 LastSchedEval=2024-12-05T14:09:32
Partition=partition-9C3RA AllocNode:Sid=head-1:14475
ReqNodeList=compute-1 ExcNodeList=(null)
NodeList=compute-1
BatchHost=compute-1
NumNodes=1 NumCPUs=2 NumTasks=1 CPUs/Task=1 ReqB:S:C:T=0:0:*:*
TRES=cpu=2,mem=1M,node=1,billing=2
Socks/Node=* NtasksPerN:B:S:C=0:0:*:* CoreSpec=*
MinCPUsNode=2 MinMemoryNode=1M MinTmpDiskNode=0
Features=(null) DelayBoot=00:00:00
OverSubscribe=OK Contiguous=0 Licenses=(null) Network=(null)
Command=sleep 120
WorkDir=/root
Power=
```

Example 2: Allow Other Jobs to Run After Modification

1. Modify `MinMemoryNode`.

```
[root@head-1 ~]# sinfo -Nel
Thu Dec  5 14:35:37 2024
NODELIST   NODES        PARTITION       STATE CPUS    S:C:T MEMORY
TMP_DISK WEIGHT AVAIL_FE REASON
compute-1      1 partition-9C3RA*   allocated    4    4:1:1  13926
0      1   (null) none
compute-2      1 partition-9C3RA*        idle    4    4:1:1  13926
0      1   (null) none
```

```
[root@head-1 ~]# srun -w compute-1 --mem 8000 sleep 120&
[3] 22415
[2]   Done                    srun -w compute-1 --mincpus=3 sleep 120
[root@head-1 ~]# srun -w compute-1 --mem 8000 sleep 120&
[4] 22428
[root@head-1 ~]# srun: job 300 queued and waiting for resources

[root@head-1 ~]# squeue
            JOBID PARTITION     NAME     USER ST       TIME  NODES
NODELIST(REASON)
            300 partition    sleep     root PD       0:00      1
(Resources)
            299 partition    sleep     root  R       0:04      1
compute-1
```

```
[root@head-1 ~]# scontrol update job 299 MinMemoryNode=2000
[root@head-1 ~]# srun: job 300 has been allocated resources

[root@head-1 ~]# squeue
            JOBID PARTITION     NAME     USER ST       TIME  NODES
NODELIST(REASON)
            299 partition    sleep     root  R       0:31      1
compute-1
            300 partition    sleep     root  R       0:03      1
compute-1
```

2. Modify `MinCPUsNode`.

```
[root@head-1 ~]# sinfo -Nel
Thu Dec  5 14:35:37 2024
NODELIST   NODES        PARTITION       STATE CPUS    S:C:T MEMORY
TMP_DISK WEIGHT AVAIL_FE REASON
compute-1      1 partition-9C3RA*   allocated    4    4:1:1  13926
```

```
0      1   (null) none
compute-2      1 partition-9C3RA*        idle    4    4:1:1  13926
0      1   (null) none
```

```
[root@head-1 ~]# srun -w compute-1 --mincpus=3 sleep 120&
[1] 21997
[root@head-1 ~]# srun -w compute-1 --mincpus=3 sleep 120&
[2] 22014
[root@head-1 ~]# srun: job 298 queued and waiting for resources

[root@head-1 ~]# squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES
NODELIST(REASON)
             298 partition    sleep     root PD       0:00      1
(Resources)
             297 partition    sleep     root  R       0:03      1
compute-1
```

```
[root@head-1 ~]# scontrol update job 297 MinCPUsNode=1
[root@head-1 ~]# srun: job 298 has been allocated resources

[root@head-1 ~]# squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES
NODELIST(REASON)
             297 partition    sleep     root  R       0:28      1
compute-1
             298 partition    sleep     root  R       0:04      1
compute-1
```

Adaptive Scheduling

fsched (fsched-dev(FIXME)+) supports adaptive scheduling.

Purpose

After setting the relevant parameters, the system periodically checks the resource usage of each running job and dynamically modifies the job's resource requests based on the results.

> ⚠ WARNING
>
> - Adjusting the requested CPU count of running jobs may conflict with CPU pinning.

> 💡 TIP
>
> - Only single-node jobs are supported.
> - Adaptive scheduling takes effect only in partitions where the relevant parameters are configured.
> - Operations that adjust job resource requests are recorded in the OS logs on the head node.
> - The current memory average uses the instantaneous total memory value of the job.

Cluster Configuration

Modify the following configuration to use the plugin `select/cons_tres_ex`, which supports changing resource requests for running jobs.

```
SelectType=select/cons_tres_ex
```

Partition Configuration

The following partition parameters are used for adaptive scheduling.

Parameter	Description	Value Type and Range	Default
`AdaptSchedInterval`	Check interval	Integer, default unit is minutes, range 1 to 180	Default is invalid, meaning no periodic

Parameter	Description	Value Type and Range	Default
		minutes	checks will run
`AdaptMinJobElapsed`	Minimum job runtime	Integer, default unit is seconds, minimum 1 second	Default is invalid, meaning no minimum job runtime limit
`AdaptCpuMode`	CPU adjustment mode	Options include `INC_DEC` (increase or decrease), `INC` (increase only), `DEC` (decrease only)	Default 0, meaning no CPU adjustment
`AdaptMemMode`	Memory adjustment mode	Options include `INC_DEC` (increase or decrease), `INC` (increase only), `DEC` (decrease only)	Default 0, meaning no memory adjustment
`AdaptMemBasis`	Memory adjustment basis	Options include `MAX` (based on maximum), `AVE` (based on average)	Default `MAX`, meaning adjust memory based on the maximum

Example

1. Modify cluster configuration.

```
SelectType=select/cons_tres_ex
```

2. Modify the partition configuration for the partition where adaptive scheduling is enabled.

```
AdaptSchedInterval=10
AdaptMinJobElapsed=60
AdaptCpuMode=INC_DEC
AdaptMemMode=INC_DEC
AdaptMemBasis=MAX
```

3. After the cluster reconfiguration succeeds, run a job in a partition configured with adaptive scheduling and view the job's current `MinCPUsNode` and `MinMemoryNode`.

```
[jj@centos7-16c-1 ~]$ srun -p partition-9BWXR -w centos7-16c-2 stress -
-cpu 3 --vm 1 --vm-bytes 30M --vm-keep -t 1200s&
```

```
[jj@centos7-16c-1 ~]$ squeue
         JOBID PARTITION     NAME     USER ST       TIME  NODES
NODELIST(REASON)
            22 partition   stress       jj  R       0:05      1
centos7-16c-2
[jj@centos7-16c-1 ~]$ scontrol show job 22
JobId=22 JobName=stress
UserId=jj(2001) GroupId=jj(2004) MCS_label=N/A
Priority=4294901759 Nice=0 Account=_fsched_all QOS=fastone-1-
_fsched_all-partition-9bwxr WCKey=*
JobState=RUNNING Reason=None Dependency=(null)
Requeue=1 Restarts=0 BatchFlag=0 Reboot=0 ExitCode=0:0
RunTime=00:00:14 TimeLimit=UNLIMITED TimeMin=N/A
SubmitTime=2025-07-22T16:22:18 EligibleTime=2025-07-22T16:22:18
AccrueTime=Unknown
StartTime=2025-07-22T16:22:18 EndTime=Unknown Deadline=N/A
SuspendTime=None SecsPreSuspend=0 LastSchedEval=2025-07-22T16:22:18
Partition=partition-9BWXR AllocNode:Sid=centos7-16c-1:15590
ReqNodeList=centos7-16c-2 ExcNodeList=(null)
NodeList=centos7-16c-2
BatchHost=centos7-16c-2
NumNodes=1 NumCPUs=1 NumTasks=1 CPUs/Task=1 ReqB:S:C:T=0:0:*:*
TRES=cpu=1,mem=1M,node=1,billing=1
Socks/Node=* NtasksPerN:B:S:C=0:0:*:* CoreSpec=*
MinCPUsNode=1 MinMemoryNode=1M MinTmpDiskNode=0
Features=(null) DelayBoot=00:00:00
OverSubscribe=OK Contiguous=0 Licenses=(null) Network=(null)
Command=stress --cpu 3 --vm 1 --vm-bytes 30M --vm-keep -t 1200s
WorkDir=/fastone/users/jj
Power=
```

4. After the job runs for a while (longer than `AdaptMinJobElapsed` and after the next adjustment interval), the job's requested resources will be adjusted automatically. You can view `MinCPUsNode` and `MinMemoryNode` again.

```
[jj@centos7-16c-1 ~]$ scontrol show job 22
JobId=22 JobName=stress
```

```
UserId=jj(2001) GroupId=jj(2004) MCS_label=N/A
Priority=4294901759 Nice=0 Account=_fsched_all QOS=fastone-1-
_fsched_all-partition-9bwxr WCKey=*
JobState=RUNNING Reason=None Dependency=(null)
Requeue=1 Restarts=0 BatchFlag=0 Reboot=0 ExitCode=0:0
RunTime=00:07:22 TimeLimit=UNLIMITED TimeMin=N/A
SubmitTime=2025-07-22T16:22:18 EligibleTime=2025-07-22T16:22:18
AccrueTime=Unknown
StartTime=2025-07-22T16:22:18 EndTime=Unknown Deadline=N/A
SuspendTime=None SecsPreSuspend=0 LastSchedEval=2025-07-22T16:22:18
Partition=partition-9BWXR AllocNode:Sid=centos7-16c-1:15590
ReqNodeList=centos7-16c-2 ExcNodeList=(null)
NodeList=centos7-16c-2
BatchHost=centos7-16c-2
NumNodes=1 NumCPUs=4 NumTasks=1 CPUs/Task=1 ReqB:S:C:T=0:0:*:*
TRES=cpu=4,mem=30M,node=1,billing=4
Socks/Node=* NtasksPerN:B:S:C=0:0:*:* CoreSpec=*
MinCPUsNode=4 MinMemoryNode=30M MinTmpDiskNode=0
Features=(null) DelayBoot=00:00:00
OverSubscribe=OK Contiguous=0 Licenses=(null) Network=(null)
Command=stress --cpu 3 --vm 1 --vm-bytes 30M --vm-keep -t 1200s
WorkDir=/fastone/users/jj
Power=
```

5. You can view the related logs in the OS logs on the head node.

```
[root@jjtest-head ~]# grep "resource adjust" /var/log/messages
Nov  6 18:24:17 jjtest-head statesvc[1006657]: [2025-11-06
18:24:17.142] [statesvc] [info] start update and resource adjustment
threads
Nov  6 18:24:17 jjtest-head statesvc[1006657]: [2025-11-06
18:24:17.143] [statesvc] [info] resource adjustment thread started
Nov  6 18:26:17 jjtest-head statesvc[1006657]: [2025-11-06
18:26:17.166] [statesvc] [info] _modify_job_resources: job 3 (user:
root) resource adjusted: cpu=1->7, mem=1MB->30MB
```

Load Thresholds

fsched supports per-partition load thresholds for memory (in MiB), CPU utilization (in %), and average CPU run-queue length (decimal, no unit). Configure them on the line that starts with `PartitionName` for the partition in `partitions.conf`. The options include:

1. Scheduling memory threshold `LoadSchedMem`
2. Stop memory threshold `LoadStopMem`
3. Scheduling CPU utilization threshold `LoadSchedUt`
4. Stop CPU utilization threshold `LoadStopUt`
5. Scheduling 15-second average CPU run-queue length threshold `LoadSchedR15s`
6. Stop 15-second average CPU run-queue length threshold `LoadStopR15s`
7. Scheduling 1-minute average CPU run-queue length threshold `LoadSchedR1m`
8. Stop 1-minute average CPU run-queue length threshold `LoadStopR1m`
9. Scheduling 15-minute average CPU run-queue length threshold `LoadSchedR15m`
10. Stop 15-minute average CPU run-queue length threshold `LoadStopR15m`

TIP

For fsched scheduler configuration, all options above are numeric and must not include units.

Option	Description
LoadSchedMem/LoadStopMem	Integer, default unit is MiB
LoadSchedUt/LoadStopUt	Integer, default unit is %
LoadSchedR15s/LoadStopR15s	Decimal or integer, no unit
LoadSchedR1m/LoadStopR1m	Decimal or integer, no unit
LoadSchedR15m/LoadStopR15m	Decimal or integer, no unit

TIP

- The current load is checked every 30 seconds. The load metrics are obtained as follows:

Metric	Data Source	Specific /proc/stat Field	Corresponding top Output	Description
mem	/proc/meminfo	MemAvailable line or MemFree + Buffers + Cached	avail Mem field in the top memory line (MB)	Available memory size, updated every 5 seconds
ut	/proc/stat	Mainly uses the idle field; all CPU time fields are used to compute total time	EMA-smoothed value of 100% - id%	CPU utilization, EMA smoothing over a 15-second window, updated every 5 seconds
r15s	/proc/stat	procs_running line, procs_blocked line	No corresponding output (running count can be seen in the Tasks line)	15-second EMA-smoothed run-queue length
r1m	/proc/stat	procs_running line, procs_blocked line	No corresponding output (running count can be seen in the Tasks line)	1-minute EMA-smoothed run-queue length
r15m	/proc/stat	procs_running line, procs_blocked line	No corresponding output (running count can be seen in the Tasks line)	15-minute EMA-smoothed run-queue length

- Expected time for load changes to trigger loadsched/loadstop actions (estimated; actual values may differ):

Metric	Expected Time
mem	35 seconds
ut	45 seconds
r15s	45 seconds

Metric	Expected Time
r1m	1 minute 30 seconds
r15m	15 minutes 30 seconds

- If the partition where the node resides is configured with scheduling thresholds `LoadSched[XXX]`:
  - The node is drained and stops accepting new jobs if any of the following conditions are met:
    - Remaining node memory is below `LoadSchedMem`.
    - Node CPU utilization is above `LoadSchedUt`.
    - Node 15-second average CPU run-queue length is greater than `LoadSchedR15s`.
    - Node 1-minute average CPU run-queue length is greater than `LoadSchedR1m`.
    - Node 15-minute average CPU run-queue length is greater than `LoadSchedR15m`.
  - The node is undrained and can accept new jobs if all of the following conditions are met:
    - Remaining node memory is above `LoadSchedMem`.
    - Node CPU utilization is below `LoadSchedUt`.
    - Node 15-second average CPU run-queue length is less than `LoadSchedR15s`.
    - Node 1-minute average CPU run-queue length is less than `LoadSchedR1m`.
    - Node 15-minute average CPU run-queue length is less than `LoadSchedR15m`.
- If the partition where the node resides is configured with stop thresholds `LoadStop[XXX]` and scheduling thresholds `LoadSched[XXX]`:
  - If any of the following conditions are met, jobs on the node are STOPped in priority order (for the same priority, by start time). STOPped jobs do not release memory. Lower-priority jobs are stopped first (for the same priority, the later-started job), until only the last job remains:
    - Remaining node memory is below `LoadStopMem`.
    - Node CPU utilization is above `LoadStopUt`.
    - Node 15-second average CPU run-queue length is greater than `LoadStopR15s`.
    - Node 1-minute average CPU run-queue length is greater than `LoadStopR1m`.
    - Node 15-minute average CPU run-queue length is greater than `LoadStopR15m`.
  - If all of the following conditions are met, the node is undrained and STOPped jobs are CONTINUEd in priority order. Higher-priority jobs are continued first (for the same priority, the earlier-started job):
    - Remaining node memory is above `LoadSchedMem`.
    - Node CPU utilization is below `LoadSchedUt`.

- Node 15-second average CPU run-queue length is less than `LoadSchedR15s`.
- Node 1-minute average CPU run-queue length is less than `LoadSchedR1m`.
- Node 15-minute average CPU run-queue length is less than `LoadSchedR15m`.

Notes:

1. You can set only one type of resource `LoadSched[XXX]` or `LoadSched[XXX]`.
2. If you want to set `LoadStop[XXX]` for a resource, you must also set the corresponding `LoadSched[XXX]`, and the following conditions must be met by resource type. Otherwise it is an invalid configuration; reconfiguration will fail, and `slurmctld` or `slurmd` restart will fail:
   - The value of `LoadStopMem` is less than `LoadSchedMem`.
   - The value of `LoadStopUt` is greater than `LoadSchedUt`.
   - The value of `LoadStopR15s` is greater than `LoadSchedR15s`.
   - The value of `LoadStopR1m` is greater than `LoadSchedR1m`.
   - The value of `LoadStopR15m` is greater than `LoadSchedR15m`.
3. If a node belongs to multiple partitions, the configuration of the last partition takes effect.
4. Because threshold checks are periodic polling, it cannot guarantee that node resources will never exceed the load thresholds.

Example

```
root@compute1:~# cat /etc/slurm/partitions.conf

#
# PARTITION partition-U7KH5
PartitionName=partition-U7KH5 Nodes=compute1 Default=YES LoadSchedMem=1300
LoadStopMem=1200  LoadSchedUt=80 LoadStopUt=90
#   DUMMY

#   NODES
NodeName=compute1 CPUs=16 RealMemory=13926  Weight=1 State=CLOUD
```

Skip Node Completing

To prevent node resource fragmentation, standard Fsched delays scheduling new jobs when a node releases resources until all releasing jobs on the node complete. This state is called "Completing". Nodes in Completing do not accept new job scheduling. However, this behavior reduces scheduling efficiency for EDA-type jobs that require only one CPU/slot. To address this, we provide the ability to skip node Completing.

Supported Versions

10.69 and later

Usage

- Add the `fsched_no_node_completing` parameter to `SlurmctldParameters`.

Overview

Fsched is a deeply optimized derivative of the open-source Slurm scheduler (19.05 branch), designed for high-performance computing scenarios. This guide provides a quick reference for Fsched core commands to help users use cluster resources efficiently.

Intended Audience

- Basic users: quickly master common commands with this guide
- Advanced users: use advanced features together with the Slurm official documentation

Common Command Quick Reference

Below are some of the most commonly used commands:

- **`sbatch`**: Submit a job script to run. The script can also include one or more `srun` commands to start parallel tasks.
- `srun`: Run parallel jobs interactively in real time, usually for short tests, or combined with `salloc` and `sbatch`.
- `salloc`: Allocate resources for jobs that need real-time handling. A typical scenario is to allocate resources and start a shell, then use that shell to run `srun` commands to execute parallel tasks.
- **`sinfo`**: Show partition or node status with many filtering, sorting, and formatting options.
- **`squeue`**: Show jobs and job-step status in the queue, with many filtering, sorting, and formatting options.
- `scancel`: Cancel pending or running jobs or job steps, and can also send arbitrary signals to all processes in running jobs or job steps.
- `sacct`: Show accounting information for active or completed jobs or job steps (corresponding to billable compute time).
- `scontrol`: Display or set the status of Slurm jobs, partitions, nodes, etc.
- `sacctmgr`: Command-line tool for managing accounting data

sbatch

Overview

`sbatch` is the command in Slurm for submitting batch jobs:

- Returns to the command line immediately after submission (runs in the background)
- Runs automatically on compute nodes when resources are available (not on login nodes)
- Use cases: any task that does not require interaction

Common Options

Option	Description	Example
`-p, --partition=partition`	Specify partition	`-p gpu`
`-J, --job-name=jobname`	Set job name	`-J my_job`
`-o, --output=out`	Output log path	`-o /path/output.log`
`-e, --error=err`	Error log path	`-e /path/error.log`
`-N, --nodes=N`	Number of nodes	`-N 2` (2 nodes)
`-n, --ntasks=ntasks`	Total CPU cores	`-n 8` (8 cores)
`-c, --cpus-per-task=ncpus`	Cores per task	`-c 4` (4 cores per task)
`-t, --time=minutes`	Time limit	`-t 1:30:00` (1h 30m)
`-w, --nodelist=hosts...`	Specify nodes	`-w node[1-3]`
`-x, --exclude=hosts...`	Exclude nodes	`-x node5`

Option	Description	Example
`-W, --wait`	Wait for job completion	`sbatch -W ...`
`--wckey=wckey`	Specify wckey	`--wckey="project1"`
`--wrap[=command string]`	Wrap command string in an sh script and submit	`--wrap "hostname"`
`-h, --help`	Show full help	`sbatch -h`

Examples

Run a single command

```
sbatch -p compute -o make.log --wrap "make test"
```

Effect: Run `make test` in the compute partition and save logs to `make.log`

 Note: Add `-W` to wait for the job to finish (useful for chaining tasks in scripts)

Submit a job via script

Script example (`myjob.sh`):

```
#!/bin/bash
echo "Task started at $(date)"
hostname
sleep 10
echo "Task finished at $(date)"
```

The user's sbatch command is:

```
sbatch -p compute -o job.log myjob.sh
```

This submission command submits the `mybash.sh` script to the compute partition and prints the standard output and error of the `myjob.sh` script to `job.log`.

Put common options in the script header

 Tip: It's recommended to put common options in the script header (example below for mybash.sh):

```
#!/bin/bash
#SBATCH -J my_job
#SBATCH -p compute
#SBATCH -N 2
#SBATCH -o %j.out
#SBATCH -e %j.err
srun your_program
```

> %j is a filename pattern substitution; see the section below

The user's sbatch command is:

```
sbatch mybash.sh
```

Filename pattern substitution

The following substitution tokens can be used in output filenames (must be used inside double quotes):

Common tokens

Token	Description	Example
`%j`	Job ID	`job-%j.out` → `job-12345.out`
`%x`	Job name	`%x-%j.out` → `myjob-12345.out`
`%u`	Username	`%u-%j.out` → `user1-12345.out`
`%a`	Job array index	`output-%a.out` → `output-1.out`
`%N`	Short hostname	Generate a separate file per node

Zero-padding examples

```
# Generate job0123.out (4-digit padding)
sbatch -o "job%4j.out" script.sh

# Generate job123-01.out (2-digit task ID padding)
sbatch -o "job%j-%2t.out" script.sh
```

> 💡 Tip: For more complex pattern substitution, see the Slurm official documentation or `man sbatch`

Common environment variables

Input environment variables

Some options can be set via environment variables. Command-line options take precedence and will override environment variable settings. The environment variables and their corresponding options are:

Variable	Option	Example
`SBATCH_PARTITION`	`-p`	`export SBATCH_PARTITION=gpu`
`SBATCH_JOB_NAME`	`-J`	`export SBATCH_JOB_NAME=myjob`
`SBATCH_TIME`	`-t`	`export SBATCH_TIME=1:00:00`
`SBATCH_OUTPUT`	`-o`	`export SBATCH_OUTPUT=job-%j.out`

Output variables

Slurm outputs the following variables in job scripts, and job scripts can use these variables:

Variable	Description
`SLURM_JOB_ID`	Current job ID
`SLURM_JOB_NAME`	Job name

Variable	Description
`SLURM_JOB_NODELIST`	Allocated node list
`SLURM_SUBMIT_DIR`	Job submission directory
`SLURM_CPUS_PER_TASK`	CPUs per task
`SLURM_GPUS`	Allocated GPU count

sinfo

Overview

`sinfo` is the command in Slurm for viewing cluster partitions and node status:

- Display partition information
- View node resource status
- Monitor system load

Common Options

Option	Description	Example
`-a, --all`	Show all partitions	`sinfo -a`
`-l, --long`	Show detailed information	`sinfo -l`
`-N, --Node`	Show by node	`sinfo -N`
`-p, --partition=PARTITION`	Specify partition	`sinfo -p gpu`
`-t, --states=node_state`	Filter by state	`sinfo -t idle`
`-o, --format=format`	Custom output format	`sinfo -o "%P %a %D %T"`
`-S, --sort=fields`	Sort output	`sinfo -S +P,-m`
`-i, --iterate=seconds`	Refresh interval	`sinfo -i 5` (refresh every 5 seconds)

Examples

View partition status

```
# sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
compute*     up   infinite      1  down* ip-10-10-2-109
compute*     up   infinite      2    mix ip-10-10-2-[70,80]
```

Here, PARTITION indicates the partition, NODES indicates the number of nodes, NODELIST is the node list, and STATE indicates the node state. idle means the node is idle, and allocated means the node has one or more jobs allocated.

View detailed partition information

sinfo -l shows more information:

```
# sinfo -l
PARTITION AVAIL TIMELIMIT JOB_SIZE   ROOT OVERSUBS GROUPS NODES STATE
NODELIST
compute*  up    infinite  1-infinite no   NO       all    2     mixed
compute[1-2]
```

View detailed node information

sinfo -Nl shows one node per line, i.e. details for each node:

```
# sinfo -Nl
NODELIST NODES PARTITION STATE CPUS S:C:T MEMORY TMP_DISK WEIGHT
node1    1     compute*  idle  16  2:8:1 64000   0        1
node2    1     compute*  alloc 16  2:8:1 64000   0        1
```

Output Field Details

- `AVAIL`: up means available, down means unavailable.
- `CPUS`: Number of CPUs on each node.
- `S:C:T`: Number of CPU sockets (S), CPU cores (C), and threads (T) on each node. One CPU socket can contain multiple CPU cores, and so on.
- `SOCKETS`: Number of CPU sockets on each node.

- `CORES`: Number of CPU cores on each node.
- `THREADS`: Number of threads on each node.
- `GROUPS`: User groups that can use the partition; all means all groups can use it.
- `JOB_SIZE`: Minimum and maximum number of nodes available for a user job. If only one value is shown, min and max are the same. infinite means no limit.
- `TIMELIMIT`: Job walltime limit (walltime refers to actual elapsed time measured by a clock). infinite means no limit. If limited, the format is "days-hours:minutes:seconds".
- `MEMORY`: Physical memory size in MB.
- `NODELIST`: Node name list, formatted like node[1-10,11,13-28].
- `NODES`: Node count.
- `NODES(A/I)`: Node count, with state format "available/idle".
- `NODES(A/I/O/T)`: Node count, with state format "available/idle/other/total".
- `PARTITION`: Partition name; a trailing * means it is the default partition.
- `ROOT`: Whether resources are restricted to the root account.
- `OVERSUBSCRIBE`: Whether job allocations can exceed compute resources (such as CPU count):

  > - no: oversubscription not allowed.
  > - exclusive: exclusive; only these jobs can use the resources (equivalent to `srun --exclusive`).
  > - force: resources are always oversubscribed.
  > - yes: resources can be oversubscribed.

- `STATE`: Node state. Possible values include:

  > - allocated, alloc: allocated.
  > - completing, comp: completing.
  > - down: down.
  > - drained, drain: drained.
  > - draining, drng: draining.
  > - fail: failed.

    - failing, failg: failing.
    - future, futr: future (available later).
    - idle: idle and can accept new jobs.
    - maint: maintenance.
    - mixed: mixed; the node is running jobs but has some idle CPU cores and can accept new jobs.
    - perfctrs, npc: unavailable due to network performance counters in use.
    - power_down, pow_dn: powered down.
    - power_up, pow_up: powering up.
    - reserved, resv: reserved.
    - unknown, unk: unknown.

  Note: if the state has a suffix *, the node is not responding.

- `TMP_DISK`: Size of the partition where `/tmp` resides, in MB.

srun

Overview

`srun` is the command in Slurm for interactive submission of parallel jobs: it returns to the terminal only after the task completes.

Use cases:

- Applications that require a GUI
- Programs that require terminal interaction (such as `bash`, `dc_shell`, `pt_shell`)

Common Options

Common srun options are similar to sbatch. See the sbatch documentation; for more options use the `srun --help` command.

Examples

Interactive terminal task

```
$ srun -c 8 --mem 10240 --pty pt_shell
```

 Effect: Allocate 8 CPU cores and 10GB memory and run `pt_shell` in interactive mode

Basic parallel task

```
srun -p compute -c 2 hostname
```

 Effect: Allocate 2 CPU cores in the compute partition to run the `hostname` command

salloc

Overview

`salloc` is the command in Slurm for dynamically allocating resources and keeping a session:

- First reserves resources and keeps the allocation
- Then runs the specified command (or drops into an interactive shell if none is specified)
- Automatically releases resources when the command/session ends

Common Options

Resource allocation options for `salloc` are largely the same as `sbatch`. See the `sbatch` options section in the documentation; for more options, use the `salloc --help` command.

Examples

Allocate 48 cores across three machines, reserve 716800M (700G) memory per machine, and run the my_job.sh script.

```
salloc -n 48 -N 3 --mem 716800 my_job.sh
```

In the my_job.sh script, nested `srun` is supported. For applications that require a GUI or pseudo-terminal, you can run them with `srun` inside my_job.sh. Or you can run `salloc [args] srun [args] application` directly.

squeue

Overview

`squeue` is the command in Slurm for viewing job queue status:

- Display all pending and running jobs
- Support multiple filtering, sorting, and formatting options
- View detailed job information and state reasons

Common Options

Option	Description	Example
`-a, --all`	Show all jobs in queues	`squeue -a`
`-j, --job=job(s)`	Specify job ID	`squeue -j 123,124`
`-u, --user=user_name(s)`	Filter by user	`squeue -u user1`
`-p, --partition=partition(s)`	Filter by partition	`squeue -p gpu`
`-t, --states=states`	Filter by state	`squeue -t RUNNING`
`-i, --iterate=seconds`	Refresh interval	`squeue -i 5` (refresh every 5 seconds)
`-o, --format=format`	Custom output	`squeue -o "%i %P %j %u %t %M %D %R"`
`-S, --sort=fields`	Sort output	`squeue -S "-t"` (sort by time desc)
`--start`	Show expected start time	`squeue --start`

Examples

Use squeue to view job status.

```
# squeue -a
JOBID PARTITION     NAME     USER ST       TIME            NODES
NODELIST(REASON)
8     compute       sleep    root  R       8-02:07:58      2     ip-10-10-
2-[70,80]
```

Here JOBID is the job ID, NAME is the job name, USER is the user, TIME is elapsed runtime, NODES is the number of nodes in use, and NODELIST is the list of nodes running the job. `-a` means show all jobs in all queues.

squeue supports showing multiple fields. For example, in addition to the default output, show the submit time and working directory:

```
squeue -a -o "%.18i %.9P %.8j %.8u %.2t %20V %.10M %.6D %R %Z"
```

Output Field Details

- `JOBID`: Job ID.
- `PARTITION`: Partition name.
- `NAME`: Job name.
- `USER`: Username.
- `ST`: State.

  - PD: pending, PENDING.
  - R: running, RUNNING.
  - CA: cancelled, CANCELLED.
  - CF: configuring, CONFIGURING.
  - CG: completing, COMPLETING.
  - CD: completed, COMPLETED.
  - F: failed, FAILED.
  - TO: timeout, TIMEOUT.

    - NF: node failure, NODE FAILURE.
    - SE: special exit state, SPECIAL EXIT STATE.

- `TIME`: Elapsed runtime.
- `NODELIST(REASON)`: Allocated node list (reason):

    - AssociationCpuLimit: The association's specified CPUs are in use; the job will run eventually.
    - AssociationMaxJobsLimit: The association's max jobs limit has been reached; the job will run eventually.
    - AssociationNodeLimit: The association's specified nodes are in use; the job will run eventually.
    - AssociationJobLimit: The job has reached its maximum allowed job count limit.
    - AssociationResourceLimit: The job has reached its maximum allowed resource limit.
    - AssociationTimeLimit: The job has reached its time limit.
    - BadConstraints: The job has constraints that cannot be satisfied.
    - BeginTime: The job's earliest start time has not been reached.
    - Cleaning: The job was requeued and is still performing cleanup from the previous run.
    - Dependency: The job is waiting for a dependent job to finish.
    - FrontEndDown: No front-end node is available to run this job.
    - InactiveLimit: The job has reached the system inactive limit.
    - InvalidAccount: The job user account is invalid; cancel and resubmit with the correct account.
    - InvalidQOS: The job QOS is invalid; cancel and resubmit with the correct QoS.
    - JobHeldAdmin: The job is held by the system administrator.
    - JobHeldUser: The job is held by the user.
    - JobLaunchFailure: The job could not be launched, possibly due to filesystem failures or invalid program names.
    - Licenses: The job is waiting for the required licenses.
    - NodeDown: The required nodes are down.
    - NonZeroExitCode: The job ended with a non-zero exit code.
    - PartitionDown: The required partition is in DOWN state.
    - PartitionInactive: The required partition is in Inactive state.
    - PartitionNodeLimit: The job's required nodes exceed the current partition limit.
    - PartitionTimeLimit: The job's required partition reached its time limit.

- PartitionCpuLimit: The CPUs for the job's partition are already in use; the job will run eventually.
- PartitionMaxJobsLimit: The partition's max jobs limit has been reached; the job will run eventually.
- PartitionNodeLimit: The specified nodes in the job's partition are already in use; the job will run eventually.
- Priority: The required partition has higher priority jobs or reservations.
- Prolog: The job's PrologSlurmctld prolog is still running.
- QOSJobLimit: The job's QOS has reached its max jobs limit.
- QOSResourceLimit: The job's QOS has reached its max resource limit.
- QOSGrpCpuLimit: All CPUs for the job's QoS group are in use; the job will run eventually.
- QOSGrpMaxJobsLimit: The job's QoS group max jobs limit has been reached; the job will run eventually.
- QOSGrpNodeLimit: All nodes for the job's QoS group are in use; the job will run eventually.
- QOSTimeLimit: The job's QOS has reached its time limit.
- QOSUsageThreshold: The required QOS usage threshold was violated.
- ReqNodeNotAvail: The required nodes are not available, such as when nodes are down.
- Reservation: The job is waiting for its reserved resources to become available.
- Resources: The job will wait until the required resources are available.
- SystemFailure: Slurm system failure, such as filesystem or network failure.
- TimeLimit: The job exceeded its time limit.
- QOSUsageThreshold: The required QoS usage threshold was violated.
- WaitingForScheduling: Waiting to be scheduled.

scancel

Overview

`scancel` is the command tool in Slurm for cancelling submitted jobs:

- Can cancel pending (PENDING) or running (RUNNING) jobs
- Supports bulk job cancellation
- Can cancel regular jobs, array jobs, and job steps
- Requires job owner or administrator privileges

Common Options

Option	Description	Example
`-u, --user=user_name`	Cancel jobs for a specific user	`scancel -u user1`
`-n, --name=job_name`	Cancel by job name	`scancel -n jobname`
`-p, --partition=partition`	Cancel by partition	`scancel -p gpu`
`-t, --state=states`	Cancel by state	`scancel -t PENDING`
`-i, --interactive`	Interactive confirmation	`scancel -i 12345`
`-v, --verbose`	Verbose mode	`scancel -v 12345`
`-Q, --quiet`	Quiet mode	`scancel -Q 12345`
`-w, --nodelist`	Cancel by node	`scancel -w node1`
`-b, --batch`	Cancel batch jobs	`scancel --batch 12345`
`-s, --signal=name`	Send a specific signal; default is SIGKILL	`scancel --signal=USR1 12345`

sacct

Overview

`sacct` is the command in Slurm for viewing job accounting data:

- Display detailed information for completed jobs
- Provide more comprehensive historical job data than `squeue`
- Support complex filtering and formatted output

Common Options

Option	Description	Example
`-j`	Filter by job ID	`sacct -j 12345,12346`
`-u`	Filter by user	`sacct -u user1,user2`
`-S`、`-E`	Specify start and end time	`sacct -S2022-02-18-11:40 -E2022-03-03-12:00`
`-T`	Handle jobs spanning time ranges	`sacct -T`
`-p`	Parsable output (delimiter |)	`sacct -p`
`-X`	Show job-level information only	`sacct -X`
`-o`	Custom output fields	`sacct -o "JobID,JobName,Partition,Elapsed"`
`--format`	Same as `-o`	`sacct --format=JobID%15,State%10`
`-n`	Hide headers	`sacct -n`

Option	Description	Example
`-P`	Parsable output (delimiter |)	`sacct -P`
--page-jobs	Set number of jobs per page	sacct --page-jobs 100
--page-last-jobid	Set the last jobid from previous query	sacct --page-last-jobid 123

Examples

View details for a specific job

```
sacct -j 12345
```

View jobs for the current user

```
sacct -u $USER
```

Complex time filtering

```
sacct -T -S2022-02-18-11:40 -E2022-03-03-12:00
```

Filter by state

```
sacct -s COMPLETED,FAILED
```

Generate CSV output

```
sacct -P -o "JobID,JobName,Partition,State,Elapsed" > jobs.csv
```

View job resource usage

```
sacct -o "JobID,JobName,AllocCPUS,ReqMem,Elapsed" -X
```

Output Field Details

Field	Description
JobID	Job ID
JobName	Job name
Partition	Partition name
Account	Account name
User	Username
State	Job state
Submit	Submit time
Start	Start time
End	End time
Elapsed	Elapsed time
Eligible	Time eligible to run
ReqCPUS	Requested CPU count
AllocCPUS	Allocated CPU count
ReqMem	Requested memory
AllocNodes	Allocated node count

scontrol

Overview

`scontrol` is the command tool in Slurm for system control and configuration management:

- View and modify Slurm configuration
- Manage jobs, nodes, partitions, and other resources
- Most operations require administrator privileges
- Provides real-time system status viewing

Common Options

Feature	Command Example	Description
Job Management		
View jobs	`scontrol show job <jobid>`	Show job details
Job extended info	`scontrol show job_ex <jobid>`	Show extra job info
Modify job	`scontrol update jobid=123 ...`	Modify job parameters
Cancel job	`scontrol kill <jobid>`	Terminate a running job
Node Management		
View node	`scontrol show node <node>`	Show node details
Modify node	`scontrol update nodename=node01 ...`	Update node config
Drain node	`scontrol update nodename=node01 state=DRAIN`	Set node to maintenance
Partition Management		

Feature	Command Example	Description
View partition	`scontrol show partition <name>`	Show partition config
Modify partition	`scontrol update partition=debug ...`	Update partition parameters
Other Features		
View config	`scontrol show config`	Show system config
View licenses	`scontrol show lic`	Show license status

Examples

Job management

View extended job info

```
scontrol show job_ex 12345
```

Extended field details

Field	Description
RespHost	Host name for interactive job requests
Port	Allocated response port (alloc_resp_port)
OtherPort	Other ports (used for notifications such as `SRUN_PING`)
LastActivity	Last active time for job allocation (job `last_time_active`)

Modify job parameters

```
scontrol update jobid=12345 TimeLimit=1-12:00:00
```

Modify job priority

```
scontrol update jobid=12345 Priority=1000
```

Terminate a job

```
scontrol kill 12345 "Maintenance required"
```

Node control

Set node to maintenance

```
scontrol update nodename=node01 state=DRAIN reason="Hardware upgrade"
```

Resume node

```
scontrol update nodename=node01 state=RESUME
```

Partition control

Change partition state

```
scontrol update partition=debug state=UP
```

Set partition node weight

```
scontrol update partition=debug weight=100
```

State Management

Node state types

State	Description
IDLE	Node is idle/available
ALLOC	Node is allocated
MIXED	Node is partially allocated
DRAIN	Node is in maintenance
FAIL	Node failure
DOWN	Node is down

Partition state types

State	Description
UP	Partition available
DOWN	Partition unavailable
DRAIN	Partition in maintenance

Notes

- Most modification operations require administrator privileges
- Configuration changes may affect system operation; use a maintenance window
- Node state changes may take time to take effect
- Some changes may require updating the slurm.conf configuration file
- Test before production operations

 Best practices:

- Check current status before important operations: `scontrol show <entity>`
- Verify after changes: `scontrol reconfigure`
- Record reasons for maintenance operations: `scontrol update ... reason="<details>"`

- Use node range expressions for batch operations: `node[01-08,12]`

sacctmgr

Overview

`sacctmgr` is the command tool in Slurm for managing accounting data:

- Manage Accounts, Users, resource limits (QOS), etc.
- Configure associations and permissions
- Requires administrator privileges

Common Options

Option	Description	Example
`add`	Add entity	`sacctmgr add account`
`modify`	Modify entity	`sacctmgr modify user`
`delete`	Delete entity	`sacctmgr delete qos`
`show`	Show info	`sacctmgr show cluster`
`-i`, `--immediate`	Take effect immediately	`sacctmgr -i add ...`
`-p`, `--parsable`	Parsable output	`sacctmgr -p show ...`
`-Q`, `--quiet`	Quiet mode	`sacctmgr -Q delete ...`

Examples

Manage accounts

```
# Add account
sacctmgr add account myaccount Description="test account" Organization=CS

# Modify account
sacctmgr modify account myaccount set GrpTRES=cpu=100
```

```
# Delete account
sacctmgr delete account myaccount
```

Manage users

```
# Add user
sacctmgr add user user1 Account=myaccount

# Set default account for user
sacctmgr modify user user1 set DefaultAccount=myaccount

# Delete user
sacctmgr delete user user1
```

Manage QOS

```
# Add QoS
sacctmgr add qos high Priority=100

# Add qos: limit high to 10 CPUs, 10 RTX2080 GPUs, and 4 V100 GPUs
sacctmgr add qos high set
GrpTRES=cpu=10,gres/gpu:rtx2080=10,gres/gpu:v100=4

# Limit high to 10 CPUs, 10 RTX2080 GPUs, and 4 V100 GPUs
sacctmgr modify qos high set
maxtresperuser=cpu=1,gres/gpu:rtx2080=1,gres/gpu:v100=1

# Modify QoS limits
sacctmgr modify qos high set MaxTRESPerJob=cpu=32

# Clear limits; -1 means unlimited
sacctmgr modify qos high set
GrpTRES=cpu=-1,gres/gpu:rtx2080=-1,gres/gpu:v100=-1

# Modify group limits
sacctmgr modify qos high set
GrpTRES=cpu=-1,gres/gpu:rtx2080=-1,gres/gpu:v100=-1

# Delete QoS
sacctmgr delete qos high
```

```
# Add user to qos
sacctmgr modify user <user_name> set qos=<qos_name>
```

View associations

```
sacctmgr show Association
format=Cluster,Account,User,Partition,QOS,GrpSubmit,GrpWall,GrpTRESMins,MaxJo
format=GrpTRES%60
```

View user details

```
sacctmgr show user user1 format=User,DefaultAccount,AdminLevel
```

View QOS configuration

```
sacctmgr show qos format=Name,Priority,MaxTRESPerJob
```

Key Parameter Details

Resource Types (TRES)

Type	Description	Example
`cpu`	CPU cores	`cpu=100`
`mem`	Memory (MB)	`mem=100G`
`gres/gpu`	GPU count	`gres/gpu=2`
`billing`	Billing weight	`billing=30`

Common limits

Limit parameter	Description
`GrpTRES`	Group total resource limit
`MaxTRESPerUser`	Per-user resource limit
`MaxTRESPerJob`	Per-job resource limit
`MaxJobs`	Max jobs limit

Notes

- All modifications take effect immediately and are irreversible
- Deleting a parent account will delete all child accounts
- Resource limits are based on real-time calculation and may affect queued jobs
- For complex configurations, run simulation tests first
- Back up configuration before production operations:

```
sacctmgr -p dump <cluster_name> file=./<file_name>.cfg
```

> Tip: For more advanced features (such as federated cluster configuration), see `man sacctmgr` or visit Slurm official documentation

bacct

Overview

`bacct` is used to display accounting statistics for completed jobs.

Options

Option	Description	Key Differences
`-h`	Print command usage and exit	Provides more detailed help information
`-V`	Print version	No difference
`-l`	Display detailed job information in long format	No difference
`-b`	Display job information in brief format	No difference
`-d`	Display only successfully completed (DONE) jobs	No difference
`-e`	Display only exited (EXIT) jobs	No difference
`-u`	Display jobs for a specified user or all users	No difference
`-q`	Display jobs submitted to a specified queue	No difference
`-m`	Display jobs executed on a specified host	No difference
`-P`	Display jobs belonging to a specified project	project maps to wckey
`-C`	Display jobs completed within a specified time interval	No difference
`-S`	Display jobs submitted within a specified time interval	No difference

Option	Description	Key Differences
`-D`	Display jobs dispatched within a specified time interval	No difference

Output Format

SUMMARY format (default)

Displays aggregated job statistics, including:

Field	Description
`Total number of done jobs`	Total number of successfully completed jobs
`Total number of exited jobs`	Total number of failed/exited jobs
`Total CPU time consumed`	Total CPU time consumed by all jobs
`Average CPU time consumed`	Average CPU time
`Maximum/Minimum CPU time of a job`	Max/min CPU time for a single job
`Total wait time in queues`	Total wait time in queues
`Average/Maximum/Minimum wait time in queue`	Average/max/min wait time
`Average/Maximum/Minimum turnaround time`	Average/max/min turnaround time (submission to completion)
`Average/Maximum/Minimum hog factor`	Average/max/min CPU utilization (CPU time/turnaround time)
`Average/Maximum/Minimum expansion factor`	Average/max/min expansion factor (turnaround time/run time)

Field	Description
`Total/Average/Maximum/Minimum Run time`	Total/average/max/min run time
`Total throughput`	Total throughput (jobs/hour)
`Beginning/Ending time`	Start/end time of the time range

BRIEF format (`-b` option)

Brief output includes the following fields:

Field	Description	Key Differences
`U/UID`	Username	No difference
`QUEUE`	Queue name	No difference
`SUBMIT_TIME`	Submission time	No difference
`CPU_T`	CPU time	No difference
`WAIT`	Wait time	No difference
`TURNAROUND`	Turnaround time	No difference
`FROM`	Submission host	Always shown as N/A
`EXEC_ON`	Execution host	No difference
`JOB_NAME`	Job name	No difference

LONG format (`-l` option)

Long format outputs full details for each job, including job ID, user, queue, submission time, start time, completion time, CPU time, wait time, turnaround time, execution host, job name, exit status, and more.

Examples

1. Show statistics for all completed jobs (default SUMMARY format)

```
bacct
```

2. Show job statistics for a specific user

```
bacct -u username
```

3. Show all jobs in brief format

```
bacct -b
```

4. Show detailed job information in long format

```
bacct -l
```

5. Show only successfully completed jobs

```
bacct -d
```

6. Show only failed/exited jobs

```
bacct -e
```

7. Show job statistics for a specific queue

```
bacct -q queuename
```

8. Show jobs executed on a specific host

```
bacct -m hostname
```

9. Show jobs for a specific project

```
bacct -P project_name
```

10. Query jobs by time range

```
# Query jobs completed between December 1, 2025 and January 31, 2026
bacct -C 2025/12/1,2026/1/31

# Query jobs completed on May 6
bacct -C 5/6

# Query jobs submitted in the last 3 days
bacct -S .-3,
```

Time Format

Format	Meaning	Example
`MM/DD`	Month/day	`5/1` means all day on May 1
`MM/DD,MM/DD`	Time range	`5/1,5/8` means May 1 to May 8
`YYYY/MM/DD`	Full date	`2024/12/1`
`.-N,`	Relative time	`.-7,` means the last 7 days

Notes

- SUMMARY format is shown by default
- `-b` and `-l` are mutually exclusive; only one can be used
- If no time option is specified, the default query is jobs completed in the last 7 days

bbot

Overview

`bbot` moves pending jobs to the bottom of the queue to affect scheduling order.

Parameters

Option	Description	Key Differences
`-h`	Print command usage and exit	Provides more detailed help information
`-V`	Print version	No difference

`job_id` Filters

Filter	Description	Key Differences
job_ID	Move the job with the specified ID to the bottom of the queue	No difference
job_ID position	Move the job with the specified ID to the bottom of the queue. The `position` parameter specifies the position counted from the bottom (1=last, 2=second to last, ...)	No difference

Examples

1. Move a specific job to the bottom of the queue

```
bbot 10023
```

2. Move a job to the third-to-last position

```

```

```
bbot 10023 3
```

Notes

- Regular users can move only their own jobs, and the adjustment applies within the same user's jobs (not across partitions).
- Admin users can move any user's jobs; changes apply within the global queue (not across partitions).
- For array jobs, only whole-array moves are supported via `job_ID`.

bhist

> INFO
>
> This command is supported starting from `fsched-10.87`.

Overview

`bhist` displays job history information.

Parameters

Option	Description	Key Differences
`-a`	Show completed and unfinished job information	No difference
`-b`	Brief format	No difference
`-C`	Show jobs completed/exited within the specified time interval	No difference
`-d`	Show completed job information	No difference
`-D`	Show jobs dispatched within the specified time interval	No difference
`-e`	Show exited job information	No difference
`-h`	Print help information	Provides more detailed help information
`-J`	Show jobs with the specified job name	No difference
`-Jd`	Filter by job description	No difference
`-l`	Long format with detailed job information	Details listed in later sections

Option	Description	Key Differences
`-m`	Show jobs dispatched to the specified host	No difference
`-p`	Show pending job information	No difference
`-P`	Show jobs belonging to the specified project	No difference, `project` maps to `wckey`
`-q`	Show jobs submitted to the specified queue	No difference
`-r`	Show running job information	No difference
`-s`	Show suspended job information	No difference
`-S`	Show jobs submitted within the specified time interval	No difference
`-u`	Show jobs submitted by the specified user	No difference
`-V`	Print version information	No difference
`-w`	Wide format	No difference

Default Output Fields

Field	Description	Key Differences
`Time Summary`	Summary of time spent in different job states	
- `PEND`	Total wait time for the job (excluding user suspend time)	No difference
- `PSUSP`	Total user suspend time while pending	Pending job suspension not supported
- `RUN`	Total run time for the job	No difference

Field	Description	Key Differences
- `USUSP`	Total user suspend time after dispatch	Shows suspend time without distinguishing user vs system suspend
- `SSUSP`	Total system suspend time after dispatch	Dummy value is 0
- `UNKWN`	Total time in unknown state	Dummy value is 0
- `TOTAL`	Total time across all states	Shows `PEND+RUN+USUSP`

Output Fields for `-l`

Field	Description	Key Differences
`Project`	Project of the submitted job	No difference, `project` maps to `wckey`
`Command`	Job command	`Command` is unavailable after the job has been completed for some time
`Requested Resources`	Full resource requirement string from `bsub`	Only `rusage[mem=xxx]` is shown
`Execution CWD`	Actual working directory during job execution	No difference
`Effective RES_REQ`	Effective resource requirements after scheduler parsing or construction	`select` and `order` are dummy values; `rusage` shows the `mem` value
`Terminated jobs`	Exit reason for terminated jobs	Exit reason is a dummy value
`Interactive jobs`	Special information for interactive jobs	Output also includes `CWD`, `home`

Field	Description	Key Differences
`Dispatched Tasks`	Allocated task count and hosts	No difference
`Allocated Slots`	Allocated slot count and hosts when `LSB_ENABLE_HPC_ALLOCATION=Y`	No difference
`Pid`	Job process ID	`Pid` is unavailable after the job has been completed for some time
`Suspend event time`	Time when the job was suspended	Suspend time is not available; estimated as a single suspend event
`Memory usage information`	Peak and average memory usage	No difference
`Submit host`	Submission host	`Submit host` is unavailable after the job has been completed for some time
`Queue`	Queue name	No difference
`Job Description`	Job description	No difference

`job_id` Filters

Filter	Description	Key Differences
None (default)	Show all historical jobs	To avoid database pressure from too many jobs, the default query is limited to the past week. You can set the range via `-C`. `-D` and `-S` still query the past week and filter within the results.

Filter	Description	Key Differences
job_ID...	Restore one or more jobs with the specified `id`.	No difference
job_ID[index_list]...	Restore one or more jobs with the specified `id` and `index_list`.	No difference

Examples

1. Show a specific job

```
bhist 11301
```

2. Filter jobs by user

```
bhist -u test
```

3. Show jobs from the past day

```
bhist -C .-1, -a
```

Notes

- To avoid database pressure from too many jobs, the wrapper limits the default query to the past week. You can set the range via `-C`.
- If there are many jobs in the query range, database pressure is still possible.

bhosts

Overview

`bhosts` is the FSCHED replacement for LSF `bhosts`, showing cluster node status. It supports multiple output formats and host filtering.

Parameters

Supported Options

Option	Description	Key Differences
`-a`	Show all hosts, including resource providers (such as EGO or OpenStack).	Resource providers (such as EGO or OpenStack) are not supported.
`-aff`	Show host topology information for CPU and memory affinity scheduling.	Detailed topology after enabling affinity is not implemented.
`-alloc`	Show slot counts; exclusive and non-exclusive jobs display different slot counts.	For exclusive jobs, `-alloc` or not both show all CPUs on the node.
`-e`	Show resources exported to other clusters.	Multi-cluster is not implemented, so it always shows no exports.
`-l`	Show host information in long format (multi-line output).	Details listed in later sections.
`-w`	Show host information in wide format (no truncation).	`STATUS` implements `closed_Busy`, `closed_Full`, and `closed_Excl`. Other states (such as `closed_Adm`, `closed_Cu_Excl`) are not implemented.

Option	Description	Key Differences
`-noheader`	Remove column headers from output.	No difference.
`-R`	Show only hosts that satisfy a resource requirement expression.	Selection via `mem`, `slots`, `status` is implemented. Other `select` options are not.
`-V`	Print version.	No difference.
`-x`	Show hosts with high job exit rate.	Exit-rate settings and statistics are not implemented, so it always shows none.
`-X`	Show uncompressed host group and compute unit output.	Host groups and compute units are not implemented, so output is uncompressed with or without `-X`.
`-h`	Show command usage.	Provides more detailed help information.

Default Output Fields

The following fields are for nodes only, without considering host groups or compute units.

Field	Description	Key Differences
`HOST_NAME`	Host name.	Host groups and `lost_and_found` nodes are not shown.
`STATUS`	Host status and `sbatchd` daemon status.	Supports `ok`, `unavail`, `unreach`, and `closed`. `closed_Cu_excl` is not implemented. `sbatchd` daemon status is not supported.
`JL/U`	Maximum number of slots per user on the host. `-` means unlimited.	Always shown as `-`.

Field	Description	Key Differences
`MAX`	Maximum available job slots on the host. `-` means unlimited.	The `MAX` field is implemented.
`NJOBS`	Number of tasks for all scheduled jobs on the host.	No difference.
`RUN`	Number of tasks for all running jobs on the host.	No difference.
`SSUSP`	Number of tasks for all system-suspended jobs on the host.	Always 0.
`USUSP`	Number of tasks for all user-suspended jobs on the host.	Suspended job count is implemented, but system vs user suspend is not distinguished.
`RSV`	Task count for pending jobs with reserved slots on the host.	LSF requires `SLOT_RESERVE` in `lsb.queues`. The current implementation uses reservations created by `scontrol create reservation`, which is different.

Output Fields for `-l`

Field	Description	Key Differences
`STATUS`	Shows reasons for host closure, including `closed_Adm`, `closed_Busy`, `closed_Full`, `closed_Excl`, etc.	Implements `closed_Busy`, `closed_Full`, and `closed_Excl`. Other states (such as `closed_Adm`, `closed_Cu_Excl`) are not implemented.
`CURRENT LOAD`	Shows total and reserved host load.	Output is based on total node resources and does not aggregate allocated and reserved resources across all jobs.

Field	Description	Key Differences
`LOAD THRESHOLD`	Shows scheduling thresholds (`loadSched`) and suspend thresholds (`loadStop`) at node level.	Node-level load thresholds are not implemented, so this shows `-`.
`CONFIGURED AFFINITY CPU LIST`	Shows configured CPU affinity information for the host.	Only outputs `AFFINITY: Disabled`, no detailed CPU list.

`host_name` Filters

Filter	Description	Key Differences
None (default)	Show all hosts	No difference
host_name ...	Show information for selected hosts	No difference
cluster_name	Show nodes for the selected cluster.	Multi-cluster is not supported; only the current cluster is shown.

Examples

Example 1: Default Output

```
$ bhosts
HOST_NAME          STATUS     JL/U MAX NJOBS RUN SSUSP USUSP RSV
host01             ok         -    4    2    2   0     0     0
host02             closed     -    8    3    1   2     0     0
```

Note: The default format shows core information like host name, status (such as `ok`), and running job counts.

Example 2: Long Format for a Specific Host

```
$ bhosts -l host01
HOST  host01
STATUS            CPUF JL/U MAX NJOBS RUN SSUSP USUSP RSV DISPATCH_WINDOW
ok                1.0   -    4     2   2      0     0     -

CURRENT LOAD USED FOR SCHEDULING:
r15s r1m r15m ut% pg io ls it tmp swp mem slots
Total        ... (detailed load data)

LOAD THRESHOLD USED FOR SCHEDULING:
... (threshold information)
```

Note: `-l` provides a multi-line detailed report including CPU load and scheduling thresholds.

Example 3: Wide Format Without Headers

```
$ bhosts -w -noheader
host01 ok         -    4   2    2   0     0     0
host02 closed     -    8   3    1   2     0     0
```

Notes

1. Option Conflicts

- `-l`, `-w`, and custom format (not implemented yet) cannot be used together. For example:

  ```
  $ bhosts -l -w
  Error: conflicting output formats
  ```

2. Feature Differences vs LSF

- Unsupported LSF options: `-o` (custom output), `-a`, and `-m` are not implemented in the current version. Contact customer support if needed.
- LSF content not available: user quotas have no one-to-one support in FSCHED and are not shown. All `SUSPEND` states are treated as `USER SUSPEND`.

3. Other Notes

- Host filtering: append host names to limit output (for example `bhosts host01 host02`).
- Field meanings: `STATUS` is the node state identifier. `NJOBS/RUN/SSUSP/USUSP/RSV` represent total jobs, running jobs, system-suspended jobs, and so on.

bjobs

Overview

`bjobs` is a job status query tool for FSCHED, compatible with LSF `bjobs` syntax. It lets users filter jobs and customize output format.

Parameters

Supported Options

Option	Description	Key Differences
`-A`	Show job array summary information	No difference
`-a`	Show jobs in all states (including recently completed jobs)	Default cleanup time for completed jobs differs: LSF 1 hour, FSCHED 900 seconds
`-d`	Show recently completed jobs	Default cleanup time for completed jobs differs: LSF 1 hour, FSCHED 900 seconds
`-G`	Show jobs for the specified user group	No difference
`-J`	Show jobs or job arrays with the specified name	No difference
`-Jd`	Show jobs with the specified description	No difference
`-l`	Long format with detailed job information (multi-line output)	Details listed in later sections

Option	Description	Key Differences
`-m`	Show jobs dispatched to the specified host	No difference
`-noheader`	Remove column headers from output	No difference
`-o`	Set a custom output format	Details listed in later sections
`-P`	Show jobs for the specified project	No difference, `project` maps to `wckey`
`-p`	Show pending jobs and reasons they are not dispatched	No difference
`-q`	Show jobs in the specified queue	No difference
`-r`	Show running jobs	No difference
`-s`	Show suspended jobs and suspension reasons	Can filter suspended jobs; suspension reason may show `None`
`-UF`	Show unformatted detailed job information	No difference
`-u`	Show jobs submitted by a specified user or all users	Supports specific users and `all`, not user groups
`-w`	Wide format (do not truncate fields)	No difference
`-json`	Show output in JSON format	No difference
`job_id`	Specify a job or job array to display	Default cleanup time for completed jobs differs: LSF 1 hour, FSCHED 900 seconds
`-h`	Show help information	Provides more detailed help information
`-V`	Print LSF version	No difference

Default Output Fields

Field	Description	Key Differences
`JOBID`	Job ID	No difference
`USER`	User who submitted the job	No difference
`STAT`	Job status	USUSP and SSUSP distinction may differ; no UNKWN, WAIT, ZOMBI states
`QUEUE`	Queue name	No difference
`FROM_HOST`	Submit host	No difference
`EXEC_HOST`	Execution host	No difference
`JOB_NAME`	Job name	No difference
`SUBMIT_TIME`	Submit time	No difference

Output Fields for `-l`

Field	Description	Key Differences
Job	Job ID assigned by LSF	No difference
User	User who submitted the job	No difference
Project	Project for the job	No difference, `project` maps to `wckey`
Command	Command executed by the job	No difference
CWD	Working directory on the submission	No difference

Field	Description	Key Differences
	host	
Execution CWD	Actual working directory during job execution	No difference
Migration threshold	Migration threshold specified via `bsub -mig`	No difference
PENDING REASONS	Reasons the job is in PEND or PSUSP state	Shows simplified reasons
SUSPENDING REASONS	Reasons the job is in USUSP or SSUSP state	Shows simplified reasons
loadSched	Job load scheduling threshold	No difference
loadStop	Job load suspension threshold	No difference
JOB STATUS	Job status	USUSP and SSUSP distinction may differ; no UNKWN, WAIT, ZOMBI states
RESOURCE USAGE	Resource usage	CPU time, MEM, NTHREAD, PGID, PIDS implemented; SWAP not implemented
Requested resources	Resource requirement string specified in `bsub`	`rusage[mem=xxx]` implemented; others not implemented
JOB_DESCRIPTION	User-provided job description	No difference

Field	Description	Key Differences
MEMORY USAGE	Memory usage details	`MAX MEM` and `AVG MEM` implemented. `MAX MEM` is the max across nodes, `AVG MEM` averages node memory and task counts
RESOURCE REQUIREMENT DETAILS	Resource request details	`rusage[mem=xxx]` implemented; shows dummy `select[type == local] order[r15s:pg]`; others not implemented
SCHEDULING PARAMETERS	Load threshold information	Shows only queue thresholds, not host thresholds. Implements `r15s`, `r1m`, `r15m`, `ut`, `mem`; not implemented: `pg`, `io`, `ls`, `it`, `tmp`, `swp`

Output Fields for `-A`

Field	Description	Key Differences
JOBID	Job ID for the job array	No difference
ARRAY_SPEC	Job array description in name[index] format	No difference
OWNER	Job array owner	No difference
NJOBS	Total number of jobs in the array	No difference
PEND	Number of jobs in PEND state in the array	No difference
RUN	Number of jobs in RUN state in the array	No difference
DONE	Number of jobs successfully completed in the array	No difference
EXIT	Number of jobs that exited abnormally in the array	No difference
SSUSP	Number of system-suspended jobs in the array	Dummy value is 0
USUSP	Number of user-suspended jobs in the array	Counts all suspended jobs

Field	Description	Key Differences
PSUSP	Number of paused jobs in the array	Dummy value is 0

Custom Format (`-o` option)

With `-o`, you can customize fields, alignment, width, and delimiter. Syntax example:

Syntax:

```
[field_name[:[-]width][:unit_prefix]] [delimiter="delimiter"]
```

Examples:

1. Basic format:

   ```
   bjobs -o "jobid job_name stat"
   ```

2. Width and alignment:

   ```
   # Right alignment (default), left alignment (with `-`), and fixed width
   bjobs -o "jobid:10 stat:-5 user:8"
   ```

3. Custom delimiter:

   ```
   bjobs -o "jobid description delimiter=, submit_time"
   ```

Supported Fields and Aliases:

Field	Alias	Description
`jobid`	id	Unique job identifier.
`job_description`	description	Job description text (optional).
`stat`		Current state (such as `RUN`, `PEND`, `EXIT`).
`user`		Submitting user name.

Field	Alias	Description
`exec_host`		Allocated compute node list (suspended jobs show `-`).
`job_name`		Job name.
`queue`		Queue name.
`from_host`		Submission workstation host name.
`command`		Launched command (optional).
`submit_time`		Submit time (format: `YYYY-MM-DD HH:MM`).
`start_time`		Job start time (format: `YYYY-MM-DD HH:MM`).
`finish_time`		Job finish time (format: `YYYY-MM-DD HH:MM`).
`run_time`		Elapsed run time (seconds; completed jobs show `0`).
`nalloc_slot`		Allocated CPU cores.
`cpu_exec_host`		Host and allocated core format (such as `2*hostA,1*hostB`).
`account`		Job account information.
`requeue`		Whether the job can be requeued.
`tmp_disk`		Temporary disk space requirement (MB).
`min_nodes`		Minimum node count.
`max_nodes`		Maximum node count.
`ntasks_per_node`		Tasks per node.

Notes:

- Field name sensitivity: Field names are case-insensitive but must match the list.

- Wildcard handling: Only `-J` and `-Jd` support the `*` wildcard (automatically converted to regex).
- Unit prefix: Automatic unit conversion (such as KB/MB) is not implemented; values are shown as-is.

Unsupported LSF Features

The following parameters are not available in the FSCHED version:

Parameter	Reason
-N	Normalized CPU time cannot be calculated.
-W, -WF, -WL, -WP	Resource usage and progress estimation are not supported.

Examples

1. Show all jobs (including completed)

```
bjobs -a
```

2. Filter jobs by user group

```
bjobs -G developers
```

3. Custom output format (pipe delimiter)

```
# Show ID, name, status, and submit time, separated by `|`:
$ bjobs -o "jobid job_name stat submit_time delimiter='|'"
```

4. Filter by user and suppress headers

```
bjobs -u alice -noheader
```

5. Show suspended jobs and reasons (wide format)

```
bjobs -p -w
```

Notes

1. Parameter conflict rules: Format options `-l`, `-o`, `-UF`, and `-w` are mutually exclusive and cannot be used together. User filters `-G` and `-u` cannot be combined.

2. User and group validation: The specified user or group name must exist (except for `-u all`). Invalid names will result in an error.

3. Wildcard rules: In `-J` or `-Jd`, entering `*test` matches any job name/description containing `test`.

4. Alternatives for unimplemented features: For resource statistics, use FSCHED built-in commands (such as `squeue --Format=...`).

bkill

Overview

`bkill` is a tool for terminating FSCHED jobs and provides an LSF-like command-line interface. It supports selectively terminating jobs using multiple filters (such as user, queue, host), and lets you specify the signal to send.

Available Options

Option	Requires Value	Description	Notes
`-l`	No	List all supported signal names and exit.	Only enumerates signals; does not terminate jobs.
`-b`	No	Force fast termination (LSF-compatible behavior).	In FSCHED, this forces `SIGKILL` instead of the default `SIGTERM`.
`-C "reason"`	Yes	Add a termination reason note. (Currently not implemented)	The argument is silently ignored; FSCHED does not support this feature.
`-J <job_name>`	Yes	Terminate jobs by exact job name match.	Only exact matches are supported; no wildcards or fuzzy search.
`-m <host_name>`	Yes	Filter jobs scheduled to the specified host/host group.	Host name must match FSCHED configuration; otherwise the command exits with an error.
`-q <queue_name>`	Yes	Filter jobs in the specified queue.	Queue names are case-sensitive and must exist.

Option	Requires Value	Description	Notes
`-s <signal_name>`	Yes	Specify the termination signal (for example `TERM`, `KILL`).	Only values defined in the `Signals` array are supported (for example `TERM, KILL, INT`); numeric or custom signals are not accepted.
`-u <user_name>`	Yes	Filter jobs by user or user group.	Use `all` to target all users; invalid users will cause the command to exit with an error.
`-V`	No	Show version information and exit.	No other side effects.

Common Examples

```
# Terminate all jobs for user alice (requires permission)
bkill -u alice

# Force-terminate jobs in queue gpu_queue
bkill -q gpu_queue -s KILL

# Terminate by exact job name "model_train"
bkill -J model_train
```

Notes

FSCHED vs LSF Differences

1. **`-C`** is not supported: The LSF feature for recording notes is removed in FSCHED; input for this option is ignored.
2. Signal behavior differs:
    - In LSF, `-b` may only send `SIGTERM`, but in FSCHED it forces `SIGKILL` (equivalent to `kill -9`).

- The default signal is always `SIGTERM` (you can explicitly specify it with `-s TERM`).

Parameter Validation Rules

- Filter combination: Multiple filters (for example `-u alice -q gpu`) are combined with logical AND; only jobs that match all conditions are terminated.
- Permission checks: If `-u` is not specified and you do not have admin permissions, you can only terminate jobs owned by the current user.

Error Handling

- If a parameter value is invalid (for example a non-existent queue name or signal name), the command fails and exits without performing any operation.
- If the filters match zero jobs, a warning is printed but the exit code is still success.

blaunch

Overview

`blaunch` is an FSCHED CLI tool that replaces LSF, allowing you to run tasks or commands on specified hosts and providing LSF-like behavior.

Parameters

Options

Option	Value Required	Purpose	Notes
-n	No	Set standard input (STDIN) to `/dev/null`.	Flag option, no value required.
`-u <host_file>`	Yes	Read host list from a file and execute on all listed hosts.	Conflicts with `-z`; error if file is missing or invalid.
`-z <hostname>`	Yes	Specify a single host to run on.	Conflicts with `-u`.
--use-login-shell	No	Use the user's login shell (such as Bash or Zsh) to execute commands.	Flag option; behavior is undefined if used with `--no-shell`.
--no-shell	No	Run the command directly without any intermediate shell.	Flag option; behavior is undefined if used with `--use-login-shell`.

Detailed Option Notes

1. **`-n`**

- Purpose: Set job standard input to `/dev/null` to avoid reading from the terminal.

- Example:

```
blaunch -n my_command arg1 arg2
```

2. **`-u <host_file>`**

- Value requirements: `<host_file>` is a text file containing host names, one per line (blank lines are ignored).
- Example:

```
blaunch -u hosts.txt my_script.sh
```

- Notes: If the file does not exist or the format is invalid, the command fails and exits.

3. **`-z <hostname>`**

- Value requirements: `<hostname>` is the target host name (for example `host1.example.com`).
- Example:

```
blaunch -z host2 compute_task
```

4. **`--use-login-shell`**

- Purpose: Execute commands via the user's login shell to ensure environment variables and configuration are loaded.
- Example:

```
blaunch --use-login-shell ./my_script.sh
```

- Notes: Behavior is undefined if used with `--no-shell`.

5. **`--no-shell`**

- Purpose: Execute commands directly without any intermediate shell.
- Example:

```
blaunch --no-shell my_binary
```

- Notes: Behavior is undefined if used with `--use-login-shell`.

Examples

Example 1: Basic Task Execution

```
blaunch echo "Hello World"
```

- Result: prints `Hello World` on the default host.

Example 2: Specify a Single Host

```
blaunch -z host3 python script.py
```

- Runs `python script.py` on `host3`.

Example 3: Load Host List from File

Assume `hosts.txt` contains:

```
node1
node2
```

Run:

```
blaunch -u hosts.txt ./run.sh
```

- Runs `./run.sh` on `node1` and `node2`.

Example 4: Conflict Error Handling

Try using `-u` and `-z` together:

```
blaunch -u hosts.txt -z host4 cmd  # Error!
```

- Error message: `FATAL: conflicting with -z`

Notes

1. Command structure: The command format is `blaunch [options] [host] command [arguments]`. If no host is specified (such as with `-u/-z`), the first remaining argument is treated as the target host and the rest as the command and arguments.

2. Conflict checks: Using `-u` and `-z` together, or `--use-login-shell` with `--no-shell`, results in errors or undefined behavior.

3. FSCHED/LSF differences: Some LSF-specific features are not implemented:

    - Job queue management (`-q`).
    - Resource quota control (such as memory, GPU allocation).
    - Dependency definitions for distributed tasks.

4. Error handling: If option parsing fails, a detailed error message is printed and execution stops.

bmod

Overview

`bmod` modifies parameters of submitted jobs.

> ⚠ WARNING
>
> - Reducing requested resources while a job is running does not reduce actual usage, so other jobs may cause node load to become too high.

Parameters

Option	Description	Key Differences
`-help`	Print command usage and exit	Provides more detailed help information
`-V`	Print version	No difference
`-version`	Print version information	No difference
`-G`	Modify user group name	Maps to the `account` field
`-Gn`	Clear user group name	No difference
`-P`	Modify project name	Maps to the `wckey` field
`-Pn`	Clear project name	No difference
`-r`	Enable automatic requeue	No difference
`-rn`	Disable automatic requeue	No difference
`-w`	Modify job dependencies	Supports `done(id)`, `ended(id)`, `exit(id,code)`

Option	Description	Key Differences
`-wn`	Clear job dependencies	No difference
`-cwd`	Modify job working directory	No difference
`-cwdn`	Clear working directory	No difference
`-o`	Modify standard output file (append mode)	No difference
`-on`	Clear standard output file	No difference
`-oo`	Modify standard output file (overwrite mode)	No difference
`-sp`	Modify job priority	No difference
`-spn`	Clear job priority	No difference
`-J`	Modify job name	No difference
`-Jd`	Modify job description	No difference
`-Jdn`	Clear job description	No difference
`-Jn`	Clear job name	No difference
`-R`	Replace resource requirements	Supports only `rusage[mem=X]` (memory MB), `rusage[tmp=X]` (temp disk MB), `span[ptile=X]` (tasks per node), `span[hosts=X]` (node count)
`-Rn`	Clear resource requirements	No difference
`-x`	Set exclusive node allocation	No difference
`-xn`	Clear exclusive node allocation	No difference

Option	Description	Key Differences
`-m`	Modify execution host list	No difference
`-mn`	Clear execution host list	No difference
`-n`	Modify task count (min_tasks[,max_tasks])	No difference
`-nn`	Clear task count	No difference
`-q`	Modify queue name	No difference
`-qn`	Clear queue name	No difference
`-U`	Modify reservation name	No difference
`-Un`	Clear reservation	No difference
`-b`	Modify job start time	No difference
`-bn`	Clear job start time	No difference
`-t`	Modify job termination deadline	No difference
`-tn`	Clear termination deadline	No difference
`-W`	Modify run time limit ([hour:]minute)	No difference
`-We`	Set estimated run time ([hour:]minute)	No difference
`-We+`	Increase run time ([hour:]minute)	Increases based on current TimeLimit
`-Wen`	Clear estimated run time	No difference

Option	Description	Key Differences
`-Wep`	Extend run time by percentage	Extends based on a percentage of current TimeLimit
`-Wn`	Clear run time limit	No difference

`job_id` Parameters

Format	Description	Key Differences
`job_ID`	Modify the job with the specified ID	No difference
`job_ID[index]`	Modify a specific task in a job array	No difference
`job_ID[start-end]`	Modify a range of tasks in a job array	No difference

Cluster Configuration

Modify the following configuration to use the plugin `select/cons_tres_ex` for changing resource requests of running tasks.

```
SelectType=select/cons_tres_ex
```

Limits for Modifying Running Jobs

For running jobs (RUNNING state), `bmod` only supports modifying the following parameters:

Category	Supported Options	Notes
Memory resources	`-R "rusage[mem=X]"`	Modify memory requirements
Time limits	`-W`, `-We`, `-We+`, `-Wep`, `-Wn`, `-Wen`	Modify run time limits
Requeue	`-r`, `-rn`	Enable/disable automatic requeue

For pending jobs (PENDING state), all supported parameters can be modified.

For completed jobs, no parameters can be modified.

Examples

1. Modify memory requirements for a running job

```
[root@head-1 ~]# bsub -R "rusage[mem=8000]" sleep 300
Job <303> is submitted to default queue.
[root@head-1 ~]# bsub -R "rusage[mem=8000]" sleep 300
Job <304> is submitted to default queue.
[root@head-1 ~]# bjobs
JOBID   USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME
SUBMIT_TIME
303     root    RUN   partition- head-1      compute-1   sleep 300  Dec
5 14:52
304     root    PEND  partition- head-1                  sleep 300  Dec
5 14:52
```

```
[root@head-1 ~]# bmod -R "rusage[mem=1000]" 303
Parameters of job <303> are being changed
[root@head-1 ~]# bjobs -l

Job <303>, User <root>, Project <*>, Status <RUN>, Queue <partition-
9C3RA>, Command <sleep
              300>
Dec 5 14:52: Submitted from host <head-1>, CWD </root>, Output File
</dev/null>, Error File
              </dev/null>, Requested Resources <rusage[mem=1000]>;
Dec 5 14:52: Started 1 Task(s) on Host(s) <compute-1>, Allocated 1
Slot(s) on Host(s)
              <compute-1>, Execution Home </root>, Execution CWD
</root>
Dec 5 14:55: Resource usage collected.
              MEM: 0 Mbytes; NTHREAD: 3
              PGID: 16165; PIDS: 16165
              PGID: 16172; PIDS: 16172 16174

 MEMORY USAGE:
 MAX MEM: 0 Mbytes
------------------------------------------------------------------------
-------
```

```
Job <304>, User <root>, Project <*>, Status <RUN>, Queue <partition-
9C3RA>, Command <sleep
              300>
Dec 5 14:52: Submitted from host <head-1>, CWD </root>, Output File
</dev/null>, Error File
              </dev/null>, Requested Resources <rusage[mem=8000]>;
Dec 5 14:53: Started 1 Task(s) on Host(s) <compute-1>, Allocated 1
Slot(s) on Host(s)
              <compute-1>, Execution Home </root>, Execution CWD
</root>
Dec 5 14:55: Resource usage collected.
              MEM: 0 Mbytes; NTHREAD: 3
              PGID: 16192; PIDS: 16192
              PGID: 16199; PIDS: 16199 16201

 MEMORY USAGE:
 MAX MEM: 0 Mbytes
```

2. Modify job name

```
bmod -J "new_name" 12345
```

3. Modify job queue

```
bmod -q normal 12345
```

4. Modify temporary disk requirements

```
bmod -R "rusage[tmp=10240]" 12345
```

5. Modify job dependencies

```
bmod -w "done(12344)" 12345
```

6. Enable automatic requeue

```
bmod -r 12345
```

7. Modify run time limit

```
bmod -W 120 12345
```

bpeek

> INFO
>
> This command is supported starting from `fsched-TBD`.

Overview

`bpeek` displays the standard output of unfinished batch jobs. It lets users monitor job execution in real time, similar to viewing a log file.

Parameters

Option	Description	Key Differences
`-f`	Follow output continuously (like `tail -f`) until the job completes.	No difference.
`-q`	Show the most recently submitted job in the specified queue.	No difference.
`-m`	Show the most recently submitted job on the specified host.	No difference.
`-J`	Show the most recently submitted job with the specified name (supports wildcard `*`).	No difference.
`-V`	Print version and exit.	No difference.

`job_id` Parameters

Format	Description	Key Differences
None (default)	Show the most recently submitted job for the current user.	No difference.
job_ID	Show the job with the specified `id`.	No difference.

Format	Description	Key Differences
job_ID[index_list]	Show the job array with the specified `id` and `index_list`.	When `-J` matches multiple array tasks, you must specify the index explicitly.

> ⚠ MUTUALLY EXCLUSIVE OPTIONS
>
> `-q`, `-m`, `-J`, and the `job_ID` parameter are mutually exclusive. Use only one at a time.

Examples

1. View output for a specific job

```
bpeek 10023
```

2. Follow job output continuously

```
bpeek -f 10023
```

3. View the most recent job output in a queue

```
bpeek -q normal
```

4. View the most recent job output on a host

```
bpeek -m node01
```

5. View the most recent job output by name

```
bpeek -J my_simulation
```

6. View output for a specific task in a job array

```
bpeek "10023[5]"
```

Notes

- Only batch jobs are supported; interactive jobs are not supported.
- The job must have started (not in pending state), otherwise it reports "Job `<job_id>` has not been dispatched yet".
- The job cannot be completed (completed/failed/cancelled/timeout), otherwise it reports "Job `<job_id>` already finished".
- With `-f`, the command keeps streaming output until the job completes or the user interrupts (Ctrl+C).
- When using `-J` and multiple tasks in the same array match, you must specify the task index explicitly, for example `bpeek "job_id[index]"`.

bqueues

Overview

`bqueues` queries queue information in an FSCHED cluster, replacing LSF `bqueues`. It provides multiple formats and filters to help users quickly understand queue status, resource allocation, and scheduling policies.

Parameters

Available Options

Option	Description	Key Differences
`-alloc`	Show job slot statistics; exclusive and non-exclusive jobs show different slot counts.	No difference.
`-h`	Print command usage and exit.	Provides more detailed help information.
`-l`	Show queue information in long format, including description, features, scheduling parameters, and more.	Details listed in later sections.
`-m`	Show queue information for specified hosts, host groups, clusters, or all hosts.	Host group display is not supported.
`-noheader`	Remove column headers from output.	No difference.
`-u`	Show queue information for specified users, user groups, or all users.	User group display is not supported.
`-V`	Print LSF version information and exit.	No difference.

Option	Description	Key Differences
`-w`	Wide format output (no truncation).	`RSV` output not implemented; not all states are supported (such as `Inact_Win`, `Inact_Adm`).

Default Output Fields

Field	Description	Key Differences
`QUEUE_NAME`	Queue name.	No difference.
`PRIO`	Queue priority (higher value means higher priority).	No difference.
`STATUS`	Queue status, including `Open`, `Closed`, `Active`, `Inactive`.	No difference.
`MAX`	Maximum job slots available in the queue; `-` means unlimited.	No difference.
`JL/U`	Maximum job slots per user in the queue; `-` means unlimited.	No difference.
`JL/P`	Maximum job slots per processor in the queue; `-` means unlimited.	Per-processor max job slots configuration and display are not implemented.
`JL/H`	Maximum job slots per host in the queue; `-` means unlimited.	No difference.
`NJOBS`	Total job slots for all jobs in the queue, including pending, running, and suspended jobs.	No difference.
`PEND`	Total task count of pending jobs in the queue.	No difference.

Field	Description	Key Differences
`RUN`	Total task count of running jobs in the queue.	No difference.
`SUSP`	Total task count of suspended jobs in the queue.	No difference.

Output Fields for `-l`

Field	Description	Key Differences
`Description`	Typical usage description for the queue.	Description contains only the queue name, no additional details.
`Default queue indication`	Default queue indicator.	No difference.
`PARAMETERS/STATISTICS`	Nice value, status, max job count, per-user job limits, and more.	`NICE` not implemented; `STATUS`, `JL/P`, `SSUSP`, `RSV` for `Inact_Win` and `Inact_Adm` not implemented.
`Schedule delay for a new job`	Scheduling delay after a new job is submitted.	Fixed at 0 seconds; `NEW_JOB_SCHED_DELAY` not implemented.
`Interval for a host to accept two jobs`	Interval for a host to accept two jobs.	Fixed at 0 seconds; `JOB_ACCEPT_INTERVAL` not implemented.
`SCHEDULING PARAMETERS`	Scheduling parameters including `r15s`, `r1m`, `r15m`, `ut`, `mem` load metrics.	Implements `r15s`, `r1m`, `r15m`, `ut`, `mem`; does not implement `pg`, `io`, `ls`, `it`, `tmp`, `swp`.

Field	Description	Key Differences
`SCHEDULING POLICIES`	Scheduling policies such as `FAIRSHARE`.	Fixed output `FAIRSHARE`; policy configuration not implemented.
`USER_SHARES`	User share allocation.	Fixed output `[default, 1]`; fairshare configuration not implemented.
`USERS`	List of users allowed to use the queue.	No difference.
`HOSTS`	List of hosts where the queue can schedule jobs.	No difference.

Examples

1. Basic Queue List

```
bqueues
```

By default, shows basic information for all queues in short format, including name, priority, status, and job counts.

2. Wide Format

```
bqueues -w
```

Extends output width for more fields (such as resource limits and load thresholds).

3. Filter Queues by Host

```
bqueues -m compute-cluster01
```

Shows only queues that can run jobs on the cluster named `compute-cluster01`.

4. Show Queues Allowed for a User

```
bqueues -u developer_team
```

Lists queues that allow user `developer_team` to submit jobs. If you use `-u all`, it shows queues open to all users.

5. Long Format Details

```
bqueues -l
```

Shows detailed parameters for each queue (such as scheduling policy, load thresholds, and associated hosts).

6. Combined Options

```
bqueues -m all -l --noheader
```

Filters queues available on all nodes and shows detailed information in long format without headers.

Notes

1. LSF compatibility differences: The original LSF `-o` option (custom output format) is not supported yet. Contact customer support if needed.

2. Parameter validation: Invalid host, cluster, or user names cause an error (for example `Bad host name` or `User cannot use the queue`). Parameter values are case-sensitive and must match cluster configuration.

3. Special handling: Using `-m all` shows only queues whose node count equals the total cluster node count. If no queue name is specified and the filter does not match (for example the host does not exist), the command returns exit code `255` and prints the reason.

bresume

> INFO
>
> This command is supported starting from `fsched-10.87`.

Overview

`bresume` is used to resume suspended jobs.

Options

Option	Description	Main Differences
`-J`	Resume only jobs with the specified name.	No difference.
`-m`	Resume only jobs allocated to a specified host or host group.	Host filtering is supported; host groups are not supported.
`-q`	Resume only jobs in the specified queue.	No difference.
`-u`	Resume jobs owned by the specified user, user group, or all users.	Supports specific users and all users; user-group filtering is not supported.
`-h`	Print command usage and exit.	Provides more detailed help information.
`-V`	Print version and exit.	No difference.

`job_id` Filters

Filter	Description	Main Differences
Default (none)	Resume one job: the most recently submitted job, or the most recently submitted job that matches other specified options.	No difference.
0	Resume multiple jobs: all jobs that match other specified options.	No difference.
job_ID...	Resume one or more jobs with the specified id(s).	No difference.
job_ID[index_list]...	Resume one or more jobs with the specified id(s) and index_list.	More accurate resume results output.

Examples

1. Resume a specific job

```
bresume 10023
```

2. Resume the most recently submitted job that matches the filter

```
bresume -u test
```

3. Resume all jobs that match the filter

```
bresume -u test 0
```

Notes

- In LSF, normal users cannot resume jobs suspended by administrators. The wrapper has no such restriction.

bstop

> INFO
>
> This command is supported starting from `fsched-10.87`.

Overview

`bstop` is used to suspend unfinished jobs.

Options

Option	Description	Main Differences
`-a`	Suspend all jobs.	No difference.
`-J`	Suspend only jobs with the specified name.	No difference.
`-m`	Suspend only jobs allocated to a specified host or host group.	Host filtering is supported; host groups are not supported.
`-q`	Suspend only jobs in the specified queue.	No difference.
`-u`	Suspend jobs owned by the specified user, user group, or all users.	Supports specific users and all users; user-group filtering is not supported.
`-h`	Print command usage and exit.	Provides more detailed help information.
`-V`	Print version and exit.	No difference.

`job_id` Filters

Filter	Description	Main Differences
Default (none)	Suspend one job: the most recently submitted job, or the most recently submitted job that matches other specified options.	No difference.
0	Suspend multiple jobs: all jobs that match other specified options.	No difference.
job_ID...	Suspend one or more jobs with the specified id(s).	No difference.
job_ID[index_list]...	Suspend one or more jobs with the specified id(s) and index_list.	More accurate suspension results output.

Examples

1. Suspend a specific job

```
bstop 10023
```

2. Suspend the most recently submitted job that matches the filter

```
bstop -u test
```

3. Suspend all jobs that match the filter

```
bstop -u test 0
```

Notes

- In LSF, pending jobs can also be suspended; in the wrapper, pending jobs cannot be suspended.

bsub

Overview

`bsub` is used to submit jobs to the FSCHED cluster and is compatible with the LSF command-line interface. It supports both batch and interactive job submission and provides rich resource management and scheduling options.

Options

I/O and Working Directory

Option	Requires Value	Format	Description
`-cwd`	Yes	Path string	Specify the job working directory. Supports relative and absolute paths.
`-i`	Yes	File path	Read job standard input from the specified file.
`-o`	Yes	File path (supports `%J`, `%I` placeholders)	Append standard output to the specified file.
`-oo`	Yes	File path (supports `%J`, `%I` placeholders)	Overwrite standard output to the specified file.
`-e`	Yes	File path (supports `%J`, `%I` placeholders)	Append standard error to the specified file.
`-eo`	Yes	File path (supports `%J`, `%I` placeholders)	Overwrite standard error to the specified file.

PLACEHOLDER NOTES

- `%J` - Job ID (automatically converted to `%A` in FSCHED)

- `%I` - Array index (automatically converted to `%a` in FSCHED)

Resource Allocation

Option	Requires Value	Format	Description
`-n`	Yes	`min_tasks[,max_tasks]`	Specify the number of tasks for the job. You can provide a minimum or a min/max range.
`-m`	Yes	Hostname list (space-separated)	Specify the host nodes on which the job runs.
`-R`	Yes	Resource requirement expression	Specify resource requirements; supports `span`, `rusage[mem=...]`, `rusage[tmp=...]`.
`-x`	No	None	Exclusive mode; the job exclusively uses the allocated nodes.
`-gpu`	Yes	`number[/mode][/j_exclusive][/mps]`	Specify GPU resource requirements.

RESOURCE EXPRESSION NOTES

The `-R` option supports the following formats:

- `span[hosts=N]` - Allocate across N hosts
- `span[ptile=N]` - Allocate N tasks per host
- `rusage[mem=SIZE]` - Memory requirement (for example `rusage[mem=4G]`)
- `rusage[tmp=SIZE]` - Temporary space requirement (for example `rusage[tmp=10G]`)

Process Resource Limits (ulimit)

Option	Requires Value	Format	Description
`-M`	Yes	Memory size (MB)	Set per-process memory limit (ulimit -m).
`-C`	Yes	Size or `unlimited`	Set core file size limit (ulimit -c).
`-c`	Yes	Time (milliseconds)	Set CPU time limit (ulimit -t).
`-D`	Yes	Size or `unlimited`	Set data segment size limit (ulimit -d).
`-F`	Yes	Size or `unlimited`	Set file size limit (ulimit -f).
`-S`	Yes	Size or `unlimited`	Set stack size limit (ulimit -s).
`-v`	Yes	Size or `unlimited`	Set virtual memory limit (ulimit -v).
`-p`	Yes	Count or `unlimited`	Set process count limit (ulimit -u).
`-T`	Yes	Count	Set thread count limit.
`-ul`	No	None	Inherit resource limits from the current shell.

Time and Scheduling

Option	Requires Value	Format	Description
`-W`	Yes	`[hour:]minute`	Set job run time limit. For example `2:30` means 2 hours 30 minutes.
`-b`	Yes	Date-time string	Set the earliest start time for the job.
`-t`	Yes	Date-time string	Set the termination deadline for the job.
`-w`	Yes	Dependency expression	Set job dependencies. Supports `done()`, `ended()`, `exit()`, `started()`.

Option	Requires Value	Format	Description
`-H`	No	None	Put the job into suspended state after submission.
`-ti`	No	None	Enable automatic orphan job termination.

DEPENDENCY EXPRESSION EXAMPLES

- `done(12345)` - Wait for job 12345 to finish successfully
- `ended(12345)` - Wait for job 12345 to finish (success or failure)
- `exit(12345,0)` - Wait for job 12345 to exit with code 0
- `started(12345)` - Wait for job 12345 to start running

Job Attributes

Option	Requires Value	Format	Description
`-J`	Yes	`jobname[index_list]`	Specify the job name. Supports arrays like `myjob[1-10]`.
`-Jd`	Yes	Description text	Specify job description.
`-q`	Yes	Queue name	Specify the submission queue.
`-P`	Yes	Project name	Specify the project the job belongs to.
`-G`	Yes	User group name	Specify the fair-share group.
`-sp`	Yes	Priority (0-100)	Set the user-specified job priority; higher value means higher priority.
`-r`	No	None	Enable automatic retry on job failure.
`-rn`	No	None	Disable automatic retry on job failure.

Interactive Jobs

Option	Requires Value	Format	Description
`-I`	No	None	Submit an interactive job with direct input/output forwarding.
`-Is`	No	None	Submit an interactive job with a pseudo-terminal and shell.
`-Ip`	No	None	Submit an interactive job with a pseudo-terminal, supports terminal control sequences.
`-IX`	No	None	Submit an interactive job with X11 GUI.
`-L`	Yes	Shell path (absolute path)	Specify the login shell.
`-tty`	No	None	Display job output and error messages on the screen.

> ⚠ MUTUALLY EXCLUSIVE OPTIONS
>
> `-I`, `-Is`, `-Ip`, and `-IX` are mutually exclusive; only one can be used.

Execution Commands

Option	Requires Value	Format	Description
`-E`	Yes	Command string	Pre-execution command run before the job starts.
`-Ep`	Yes	Command string	Post-execution command run after the job ends.

Environment Variable Control

Option	Requires Value	Format	Description
`-env`	Yes	Environment variable spec	Control environment variable propagation. Supports `all`, `none`, selective propagation, etc.

ENVIRONMENT VARIABLE SPEC FORMAT

- `all` - Propagate all environment variables (default)
- `none` - Do not propagate any environment variables
- `VAR1,VAR2` - Propagate only specified variables
- `~VAR1,~VAR2` - Exclude specified variables
- `VAR=value` - Set variable values
- Combinations are allowed, for example `all,~TMPDIR,MYVAR=newvalue`

Notifications

Option	Requires Value	Format	Description
`-B`	No	None	Send an email notification when the job starts.
`-N`	No	None	Send an email notification when the job ends.
`-Ne`	No	None	Send an email notification when the job exits.
`-u`	Yes	Email address	Specify the email recipient address (not fully implemented yet).

Other

Option	Requires Value	Format	Description
`-K`	No	None	Wait for the job to complete and return the exit code. Batch jobs only.

Option	Requires Value	Format	Description
`-V`	No	None	Print version information and exit.

> ⚠ OPTION LIMITATIONS
>
> The `-K` option applies only to batch jobs and is mutually exclusive with interactive options (`-I`, `-Is`, `-Ip`, `-IX`).

Examples

Basic Submission

```
# Submit a simple batch job
bsub ./myjob.sh

# Submit a job with a name
bsub -J myjob ./run.sh

# Submit an array job
bsub -J "arrayjob[1-10]" ./process.sh
```

Resource Allocation

```
# Specify task count and run time
bsub -n 8 -W 2:30 ./parallel_job.sh

# Specify memory and temporary space requirements
bsub -R "rusage[mem=4G,tmp=10G]" ./data_process.sh

# Allocate across multiple nodes
bsub -n 16 -R "span[hosts=4]" ./mpi_job.sh

# Specify GPU resources
bsub -gpu 2 ./gpu_task.sh
```

Process Limits

```
# Set memory and CPU time limits
bsub -M 2048 -c 3600000 ./limited_job.sh

# Set multiple resource limits
bsub -M 4096 -S 102400 -F 1048576 ./constrained_job.sh

# Inherit current shell resource limits
bsub -ul ./inherit_limits.sh
```

I/O Redirection

```
# Redirect output to files
bsub -o output.%J.log -e error.%J.log ./task.sh

# Overwrite output files
bsub -oo output.log -eo error.log ./task.sh

# Use array task placeholders
bsub -J "job[1-5]" -o "out.%J.%I.log" ./array_task.sh
```

Scheduling Control

```
# Delay job start
bsub -b "2025-01-01:09:00" ./morning_job.sh

# Set dependencies
bsub -w "done(12345)" ./dependent_job.sh

# Suspend job after submission
bsub -H ./paused_job.sh

# Set priority
bsub -sp 80 ./high_priority.sh
```

Interactive Jobs

```
# Basic interactive job
bsub -I bash

# Interactive job with pseudo-terminal
bsub -Ip -n 4 python
```

```
# X11 GUI job
bsub -IX matlab

# Use a login shell
bsub -Is -L /bin/bash
```

Environment Variable Control

```
# Do not propagate any environment variables
bsub -env "none" ./clean_env.sh

# Propagate only specified variables
bsub -env "PATH,HOME,USER" ./selective.sh

# Exclude specific variables
bsub -env "all,~TMPDIR,~DISPLAY" ./filtered.sh

# Set variable values
bsub -env "DEBUG=1,WORKERS=8" ./configured.sh
```

Notifications and Waiting

```
# Send emails on start and end
bsub -B -N ./notify_job.sh

# Wait for job completion
bsub -K ./wait_job.sh
echo "Job completed, exit code: $?"
```

Combined Usage

```
# Full job configuration example
bsub \
  -J "production[1-100]" \
  -q normal \
  -n 8 \
  -W 4:00 \
  -R "span[hosts=1] rusage[mem=8G]" \
  -M 8192 \
  -o "logs/output.%J.%I.log" \
  -e "logs/error.%J.%I.log" \
```

```
-P myproject \
-sp 50 \
-B -N \
./production_task.sh
```

Notes

Limitations

1. Resource expression support
   - The `-R` option supports only `span[hosts=N]`, `span[ptile=N]`, `rusage[mem=SIZE]`, and `rusage[tmp=SIZE]`.
   - Advanced LSF resource syntax such as `select`, `cu`, and `affinity` is not supported.
2. Email notifications
   - The `-u` option is not fully implemented yet; email notifications may be limited by system configuration.
3. GPU resources
   - The `-gpu` option supports parameter parsing; full functionality depends on FSCHED GPU scheduling configuration.
4. Priority
   - The `-sp` option accepts priorities 0-100; actual scheduling behavior depends on system configuration.

Placeholder Conversion

- LSF `%J` (job ID) is automatically converted to FSCHED `%A`.
- LSF `%I` (array index) is automatically converted to FSCHED `%a`.
- Placeholders are valid only for `-o`, `-oo`, `-e`, and `-eo`.

Option Conflicts

- Interactive options (`-I`, `-Is`, `-Ip`, `-IX`) are mutually exclusive.
- `-K` cannot be used together with interactive options.

- `-o` conflicts with `-oo`, and `-e` conflicts with `-eo` (the latter overwrites the former).

Default Behavior

- If `-o` or `-e` is not specified, standard output and error are discarded by default (sent to `/dev/null`).
- If `-cwd` is not specified, the current working directory at submission time is used.
- Automatic job retry is enabled by default (`-r`), and can be disabled with `-rn`.

Unsupported Options

The following LSF options are not supported in the current version. Using them will show a warning but will not affect job submission:

- CSM-related options (for example `-alloc_flags`, `-cn_cu`)
- Advanced LSF features (for example `-app`, `-sla`, `-clusters`)
- File staging (for example `-f`, `-stage`)
- Checkpointing and migration (for example `-k`, `-mig`)
- Alternative submission formats (for example `-json`, `-yaml`)

For the complete list of unsupported options, see the FSCHED technical documentation.

bswitch

> INFO
>
> This command is supported starting from `fsched-10.87`.

Introduction

`bswitch` is used to switch an unfinished job from one queue to another.

Parameters

Option	Description	Main Differences
`-J`	Switch only jobs with the specified job name.	No difference.
`-q`	Switch only jobs in the specified queue.	No difference.
`-u`	Switch only jobs submitted by the specified user.	Supports specified user and all users, but does not support filtering by user group.
`-V`	Print the version number.	No difference.
`-h`	Print command usage and exit.	Provides more detailed help information.

job_id Filters

Filter	Description	Main Differences
Default (none)	Switch one job: the most recently submitted job, or the most recently submitted job that matches other specified options.	No difference

Filter	Description	Main Differences
0	Switch multiple jobs: all jobs that match other specified options.	No difference
job_ID...	Switch one or more jobs with the specified id(s).	No difference
job_ID[index_list]...	Switch one or more jobs with the specified id(s) and index_list.	More accurate switch results output

Examples

1. Switch the specified job to the **`compute`** queue

```
bswitch compute 11307
```

2. Switch the most recently submitted job that matches the filter to the **`compute`** queue

```
bswitch -u test compute
```

3. Switch all jobs that match the filter to the **`compute`** queue

```
bswitch -u test compute 0
```

Notes

- Only jobs in the `pending` state can be switched
- For jobs in the `running` state, the system will output an error message indicating the operation is not supported

btop

Overview

`btop` moves pending jobs to the top of the queue to affect scheduling order.

Options

Option	Description	Main Differences
`-h`	Print command usage and exit	Provides more detailed help information
`-V`	Print version	No difference

`job_id` Filters

Filter	Description	Main Differences
job_ID	Move the specified job ID to the top of the queue	No difference
job_ID position	Move the specified job ID to the top of the queue; the `position` argument specifies the position from the top (1 = first, 2 = second, etc.)	No difference

Examples

1. Move a job to the top of the queue

```
btop 10023
```

2. Move a job to the third position

```
btop 10023 3
```

Notes

- Normal users can only move their own jobs, and the order change applies only within that user's jobs (not across partitions).
- Administrators can move any user's jobs; the change applies to the global queue (not across partitions).
- For array jobs, only whole-job moves by `job_ID` are supported.

esub

When users submit jobs, you can use an `esub` script to control submissions. The `esub` script can validate job parameters and choose to accept, modify, or reject the job. The `esub` script runs on the submission host.

> ⚠ WARNING
>
> `esub` currently checks in the following order and uses the first file found:
>
> - Fixed location: `/etc/wrappers/esub`
> - `esub` under the `LSF_SERVERDIR` directory

How It Works

When users submit with `bsub`, it checks for an `esub` script at the fixed locations. If present, `bsub` executes the script and passes environment variables to it. The `esub` script uses these variables to decide whether to accept, modify, or reject the job. Based on the `esub` exit code, `bsub` decides what to do.

- If `esub` returns 0, the job is accepted and parameter updates are applied.
- If `esub` returns `LSF_SUB_ABORT_VALUE`, the job is rejected.
- If `esub` returns any other value, the job is submitted with the original parameters.

Deployment

1. Install `lsf-wrapper` and load environment variables.
2. Place the `esub` script at `$LSF_SERVERDIR/esub` or `/etc/wrappers/esub`.
3. Subsequent `bsub` submissions will be controlled by the `esub` script.

Environment Variables Used by **esub**

- `LSF_INVOKE_CMD`: The command name that invokes `esub`; currently only `bsub` is supported.
- `LSB_SUB_ABORT_VALUE`: To reject the job, `esub` should `exit $LSB_SUB_ABORT_VALUE`.
- `LSB_SUB_PARM_FILE`: `bsub` parameters are stored in this file. `esub` reads it during submission and can accept/modify/reject the job based on parameters listed in the next section.

- `LSB_SUB_MODIFY_FILE`: If you want to modify job parameters, write new option values to this file. Supported parameters are listed in the next section.
- `LSB_SUB_MODIFY_ENVFILE`: If you want to modify user environment variables, write the new variables to this file.

Parameters Supported by **`esub`**

Currently supported parameters:

Option	esub Variable	Purpose
-e, -eo	LSB_SUB_ERR_FILE	Standard error file path
-o, -oo	LSB_SUB_OUT_FILE	Standard output file path
-i	LSB_SUB_IN_FILE	Standard input file path
-W	LSB_SUB_RLIMIT_RUN	Run time limit
-n	LSB_SUB_NUM_PROCESSORS	Task count
-R	LSB_SUB_RES_REQ	Resource request (`-R`)
-b	LSB_SUB_BEGIN_TIME	Job start time
-t	LSB_SUB_TERM_TIME	Job forced termination time
-x	LSB_SUB_EXCLUSIVE	Exclusive nodes
-w	LSB_SUB_DEPEND_COND	Dependency condition
-E	LSB_SUB_PRE_EXEC	Prelog script path
-Ep	LSB_SUB_POST_EXEC	Postlog script path
-J	LSB_SUB_JOB_NAME	Job name
-P	LSB_SUB_PROJECT_NAME	Project name

Option	esub Variable	Purpose
-G	LSB_SUB_USER_GROUP	Fairshare user group name
-q	LSB_SUB_QUEUE	Queue/partition name
-Is	LSB_SUB_PTY_SHELL	Request PTY shell
-Ip	LSB_SUB_PTY	Request PTY
-L	LSB_SUB_LOGIN_SHELL	Use login shell
-Jd	LSB_SUB_JOB_DESCRIPTION	Job description

`esub` Script Examples

Require Project (`-P`), Reject if Missing

```
#!/bin/bash

source ${LSB_SUB_PARM_FILE}

# abort the job if mandatory parameter is missing
if [ -z "${LSB_SUB_PROJECT_NAME}" ]; then
    echo "Error: Mandatory parameter '-P <project_name>' is missing."
    echo "Please specify a project name using '-P' option when submitting
jobs."
    exit ${LSB_SUB_ABORT_VALUE}
fi

# continue the job
exit 0
```

Require Fairshare Group (`-G`), Reject if Missing

```
#!/bin/bash

source ${LSB_SUB_PARM_FILE}

# abort the job if mandatory parameter is missing
```

```
if [ -z "${LSB_SUB_USER_GROUP}" ]; then
    echo "Error: Mandatory parameter '-G <account_name>' is missing."
    echo "Please specify an account name using '-G' option when submitting
jobs."
    exit ${LSB_SUB_ABORT_VALUE}
fi

# continue the job
exit 0
```

lshosts

Overview

`lshosts` displays static resource information for hosts in the cluster, including host names, architecture types, CPU information, and memory configuration.

Parameters

Option	Description	Key Differences
`-w`	Wide format (no field truncation)	No difference
`-l`	Long format (multi-line output) with additional fields and topology info	See long format section
`-T`	Show CPU topology information for hosts	NUMA hierarchy information is omitted
`-R`	Show only hosts that satisfy a resource requirement expression	Supports `select` clauses (`ncpus`/`mem`/`type`, supports `defined()`, supports `&&`/`
`-s`	Show static shared resource information	Always outputs "No shared resources"
`-a`	Show all hosts	Default is already all hosts; this option is a no-op
`-cname`	Include cluster name in output	Outputs "The -cname option is only supported on LSF/XL submission clusters"
`-V`	Print version	No difference
`-h`	Print command usage	Provides more detailed help information

Default Output Fields

Field	Description	Key Differences
`HOST_NAME`	Host name	No difference
`type`	Host type (architecture)	Architecture is converted to uppercase (for example `X86_64`)
`model`	Host model (CPU model)	Hardcoded to "Intel"
`cpuf`	Relative CPU performance factor	Hardcoded to "2.5"
`ncpus`	Number of processors on the host	No difference
`maxmem`	Maximum physical memory available to user processes (with unit, such as `15.6G`)	No difference
`maxswp`	Total swap space	Hardcoded to "-" (not configured)
`server`	Whether the host is a server host (`Yes`/`No`)	Hardcoded to "Yes"
`RESOURCES`	Boolean resources and external static resource values defined for the host	Hardcoded to "()" (no resources)

Output Fields for `-l`

Field	Description	Key Differences
`ndisks`	Number of local disk drives directly attached to the host	Hardcoded to 1
`maxtmp`	Maximum configured `/tmp` space (MB)	No difference
`rexpri`	Execution priority for remote jobs run by RES (-20 to 20)	Hardcoded to 0
`nprocs`	Number of physical processors (sockets)	No difference

Field	Description	Key Differences
`ncores`	Cores per processor	No difference
`nthreads`	Threads per core	No difference
`RUN_WINDOWS`	Time window when LIM considers the host available for remote jobs	Hardcoded to "(always open)"
`LOAD_THRESHOLDS`	Thresholds for scheduling interactive jobs	All fields hardcoded to "-" (not configured)
`RESOURCES`	Host static resources (long format)	Shows "Not defined"

Filters

Filter	Description	Key Differences
None (default)	Show all host information	No difference
host_name...	Show information for specified hosts; multiple host names are space-separated	No difference
cluster_name	Show hosts for the specified cluster	Multi-cluster is not supported; only the current cluster is shown

---

Examples

1. Show basic information for all hosts (default format)

```
lshosts
```

2. Show all hosts in wide format

```
lshosts -w
```

3. Show detailed information for all hosts in long format

```
lshosts -l
```

4. Show CPU topology information for hosts

```
lshosts -T
```

5. Show information for specific hosts

```
lshosts compute1
lshosts compute1 compute2 compute3
```

6. Filter hosts with CPU count >= 16

```
lshosts -R 'select[ncpus>=16]'
```

7. Filter hosts with memory > 64GB (unit is MB)

```
lshosts -R 'select[mem>65536]'
```

8. Filter hosts by architecture

```
lshosts -R 'select[type==X86_64]'
```

9. Filter with combined conditions

```
lshosts -R 'select[ncpus>=16 && mem>32768]'
```

lsinfo

> INFO
>
> This command is supported starting from fsched 10.79.

Overview

The `lsinfo` command displays cluster resource configuration, host models, and host types. It supports filtering by category and querying detailed configuration for specific resources.

Options

Option	Description	Conflicts With
`-l`	Show detailed information in multi-line format	Cannot be used with `-w`
`-w`	Show information in wide horizontal format	Cannot be used with `-l`
`-m`	Show current cluster host models (including CPU factor and architecture)	Cannot be used with `-M`
`-M`	Show all supported host models (including historical models and special configs)	Cannot be used with `-m`
`-r`	Show only resource configuration (such as CPU, memory, storage)	No conflicts
`-t`	Show only host types (such as OS architecture and platform type)	No conflicts
`[resource_name...]`	Query one or more resources by name, listed at the end of the command (for example `ncpus mem`)	No conflicts

---

Examples

1. Show all resource configuration

```
lsinfo -r
```

2. Query GPU and memory configuration

```
lsinfo gpu mem
```

3. Compare host models

```
lsinfo -m   # current model
lsinfo -M   # all models
```

4. Combine format and type query

```
lsinfo -l -t  # long format host types
```

Notes

1. Default behavior When no options are specified, the command shows:
   - All resource configuration (equivalent to `-r`)
   - All host types (equivalent to `-t`)
   - Current host model (equivalent to `-m`) and historical models (equivalent to `-M`)
2. Configuration file precedence The following admin-configured files override the default display (not visible to normal users):
   - `lsinfo_rsrc.conf`: custom resource descriptions
   - `lsinfo_model.conf`: extended host models
   - `lsinfo_type.conf`: additional host types
3. Resource aliases Some resources have equivalent aliases, for example:
   - `r1m` → `cpu`

- `it` → `idle`
- `swp` → `swap`

lsload

Overview

`lsload` displays load information for cluster nodes. It is similar to the LSF `lsload` command. It queries the FSCHED scheduler for host resource usage and supports custom output formats and filters.

Options

Available Options

Option	Description	Main Differences
`-a`	Show all hosts, including Dynamic Cluster VM hosts.	Dynamic Cluster VM hosts are not supported; only standard LSF hosts are shown.
`-I`	Show only specified load indices.	No difference.
`-l`	Long format output with extra fields for I/O and external load indices.	`it` outputs dummy value `0`; external load indices are not included.
`-w`	Wide format output without truncation.	`it` outputs `0`.
`-N`	Show normalized CPU run queue length index.	Uses dummy `cpu_factor` value `2.5` in calculation.

Option	Description	Main Differences
`-E`	Show effective CPU run queue length index.	No difference.
`-n`	Show load information for the specified number of hosts.	No difference.
`-R`	Show load information only for hosts that satisfy resource requirements.	Supports `select` with any of `r15s,r1m,r15m,ut,pg,io,ls,it,tmp,swp,status`; logical and arithmetic operators are not supported; other `res_req` are not supported.
`-V`	Print version.	No difference.
`-h`	Show command usage.	Provides more detailed help information.

Default Output Fields and `-l` Output Fields

Field	Description	Main Differences
`HOST_NAME`	Standard host name, typically a two-component internet domain name.	No difference.
`status`	Host status, possible values include `ok`, `-ok`, `busy`, `lockW`, `lockU`, `unavail`.	Implements `ok`, `unavail`, `busy`, `lockU`; `-ok` and `lockW` are not implemented.
`r15s`	15-second exponential average CPU run queue length.	No difference.
`r1m`	1-minute exponential average CPU run queue length.	No difference.

Field	Description	Main Differences
`r15m`	15-minute exponential average CPU run queue length.	No difference.
`ut`	1-minute exponential average CPU utilization, range 0–1.	No difference.
`pg`	1-minute exponential average memory paging rate (pages/sec).	No difference.
`ls`	Current logged-in user count.	No difference.
`it`	Host idle time (UNIX: keyboard idle; Windows: screen saver active time).	Outputs dummy value `0`.
`tmp`	Available space in `/tmp`.	No difference.
`swp`	Available swap space.	No difference.
`mem`	Available memory.	No difference.
`io`	Disk I/O rate exponential average, hidden by default; shown with `-l`.	No difference.

`hostname` Filters

Filter	Description	Main Differences
Default (none)	Show all hosts.	No difference.
host_name ...	Show the selected hosts.	No difference.
cluster_name	Show hosts in the specified cluster.	Multi-cluster is not supported; only the current cluster is shown.

Examples

Example 1: Show default load information for all hosts

```
lsload
```

Field notes: includes host name, status (`ok/busy/unavail/lockU`), CPU load indices (r15s/r1m, etc.), memory usage, and more.

Example 2: Limit to top 3 hosts

```
lsload -n 3
```

Shows only the top 3 hosts by load.

Example 3: Filter by resource conditions

```
# Show hosts with memory > 10000MB
lsload -R "select[mem>10000]"
```

Example 4: Use wide format

```
lsload -w
```

Adjusts field layout for a wider display.

Notes

Compatibility Differences (LSF vs FSCHED)

- Unsupported feature: the `-o` option for custom output is not supported; using it will error.

Parameter Conflicts

- CPU run queue options are mutually exclusive: `-N` and `-E` cannot be used together.
- Format option precedence: if `-l` and `-w` are both used, the last one takes effect.

Supported `-R` Syntax

`-R` uses an LSF-style resource string, for example:

```
"select[mem>10000]"
```

qacct

Overview

`qacct` queries accounting information for completed jobs. It is similar to the Sun Grid Engine (SGE) `qacct` command. You can filter and view job records for specific conditions and aggregate by job, user, queue, project, host, and more.

Options

Supported Options

Option	Requires Value	Description	Main Differences
`-j`	Optional	Filter by job ID/job name/pattern. With a value: show details only (no summary). Without a value: show details + system summary.	No difference.
`-o`	Optional	Filter or group by submitter; supports multiple users (e.g. `user1,user2`).	No difference.
`-u`	Optional	Same as `-o`, filter/group by submitter.	No difference.
`-b`	Yes	Show jobs that started after the specified time. Format: `[[CC]YY]MMDDhhmm[.SS]` (e.g. `202601150800`, `01150800`, `2601150800`, `01150800.30`).	No difference.
`-e`	Yes	Show jobs that started before the specified time. Format: `[[CC]YY]MMDDhhmm[.SS]`.	No difference.

Option	Requires Value	Description	Main Differences
`-d`	Yes	Show jobs that started within the last N days.	No difference.
`-E`	No	Use job end time instead of start time for time filtering (use with `-b`/`-e`/`-d`).	No difference.
`-h`	Optional	Filter/group by host name.	No difference.
`-t`	Yes	Filter array job tasks by task ID; must be used with `-j`.	No difference.
`-A`	Optional	Filter/group by account.	No difference.
`-P`	Optional	Filter/group by project.	`project` maps to `wckey`.
`-q`	Optional	Filter/group by queue.	No difference.
`-g`	Optional	Filter/group by user group.	No difference.
`-pe`	Optional	Filter/group by parallel environment (PE).	Extracted from job comment field (`PE:name` or `PE:name:slots`).
`-slots`	Optional	Filter/group by slots (CPU count).	No difference.
`-D`	Optional	Filter/group by department.	Always shows `defaultdepartment` (no department concept).
`-ar`	Optional	Filter/group by advanced reservation (AR).	Always shows `0` (no AR concept).
`-l`	Yes	Filter by resource requirements (e.g. `mem_free=1G,arch=x86_64`).	Supports `hostname`, `arch`, `num_proc`, `mem_free`, `mem_total`, `virtual_free`,

Option	Requires Value	Description	Main Differences
			`virtual_total`. Only jobs that actually ran (`start_time != 0`).
`-m`	No	Show master tasks only.	Placeholder; behaves the same as without this option.
`-f`	Yes	Specify a custom accounting file path.	Not supported; errors and suggests using the database instead.

Output Modes

Mode	Trigger	Output
Detail mode	`-j <job_id|job_name|pattern>`	Detailed accounting for matched jobs (no summary).
Summary mode	No `-j`, or `-j` with no value	Grouped summary statistics (WALLCLOCK, UTIME, STIME, CPU, MEMORY, etc.).
Mixed mode	`-j` (without value)	All job details + Total System Usage summary.

Summary Fields

Field	Description	Main Differences
`WALLCLOCK`	Total wall-clock time (seconds).	No difference.
`UTIME`	User CPU time (seconds).	No difference.
`STIME`	System CPU time (seconds).	No difference.
`CPU`	Total CPU time (UTIME + STIME, seconds).	No difference.

Field	Description	Main Differences
`MEMORY`	Memory usage (bytes).	Placeholder; shows 0.000.
`IO`	I/O bytes.	Placeholder; shows 0.000.
`IOW`	I/O wait time (seconds).	Placeholder; shows 0.000.

Detail Mode Fields

Field	Description	Main Differences
`qname`	Queue name (partition).	No difference.
`hostname`	Execution host.	No difference.
`group`	User group.	No difference.
`owner`	User name.	No difference.
`project`	Project name (wckey).	`project` maps to `wckey`.
`department`	Department name.	Always `defaultdepartment`.
`jobname`	Job name.	No difference.
`jobnumber`	Job ID.	Array jobs show `array_job_id`.
`taskid`	Task ID (array jobs).	Non-array jobs show `undefined`.
`account`	Account name.	Defaults to `sge`.
`priority`	Job priority.	Always `0`.
`qsub_time`	Submit time.	No difference.
`start_time`	Start time.	No difference.

Field	Description	Main Differences
`end_time`	End time.	No difference.
`granted_pe`	Granted parallel environment.	Always `NONE`.
`slots`	Allocated slots (CPU count).	Extracted from TRES data.
`failed`	Failure code.	Mapped from job state (26/37/100, etc.).
`exit_status`	Job exit status.	Failed jobs add 128 (ABNORMAL_EXIT_STATE_BASE).
`ru_wallclock`	Wall-clock time.	No difference.
`ru_utime`	User CPU time.	No difference.
`ru_stime`	System CPU time.	No difference.
`ru_maxrss`	Max resident set size.	Extracted from job step TRES data.
`cpu`	Total CPU time.	No difference.
`mem`	Memory usage (with `s` suffix).	Extracted from job step TRES data.
`maxvmem`	Max virtual memory.	Extracted from job step TRES data.

Examples

Example 1: Query accounting for a specific job (detail mode)

```
$ qacct -j 5678
==============================================================
qname        normal
```

```
hostname     compute01
group        users
owner        user1
project      project1
department   defaultdepartment
jobname      myjob
jobnumber    5678
taskid       undefined
account      sge
priority     0
qsub_time    Mon Jan 15 10:30:00 2026
start_time   Mon Jan 15 10:31:00 2026
end_time     Mon Jan 15 11:30:00 2026
...
```

Note: Shows full accounting details for job `5678`, without summary totals.

Example 2: Query by job name pattern

```
$ qacct -j "test_*"
==============================================================
qname        normal
hostname     compute01
...
==============================================================
qname        normal
hostname     compute02
...
```

Note: Supports `*` and `?` wildcards for job name matching.

Example 3: Summary by user and time range

```
$ qacct -o user1 -b "202601010000"
OWNER        WALLCLOCK         UTIME         STIME           CPU             
================================================================================
user1             3600         100.000        50.000        150.000
```

Note: Shows aggregated usage for user `user1` for jobs started after `2026-01-01 00:00`.

Example 4: Summary grouped by queue

```
$ qacct -q
CLUSTER QUEUE     WALLCLOCK         UTIME         STIME           CPU
================================================================================
normal              36000        1000.000       500.000       1500.000
highmem             72000        2000.000      1000.000       3000.000
```

Note: Shows summary grouped by queue.

Example 5: Summary grouped by host

```
$ qacct -h
HOST         WALLCLOCK         UTIME         STIME           CPU             M
================================================================================
compute01         2317         0.000         0.000         0.000             0
compute02          111         0.000         0.000         0.000             0
```

Note: Shows summary grouped by execution host.

Example 6: Query a specific array task

```
$ qacct -j 1234 -t 10
==============================================================
qname        normal
hostname     compute01
...
jobnumber    1234
taskid       10
...
```

Note: `-t` must be used with `-j` to query an array task.

Example 7: Filter by resource requirements

```
$ qacct -l "mem_free=1G,arch=x86_64"
HOST         WALLCLOCK         UTIME         STIME           CPU             M
```

```
==============================================================================
compute01        2317         0.000         0.000         0.000
```

Note: Shows only jobs that ran on nodes meeting the resource constraints.

Example 8: Filter by end time

```
$ qacct -E -b "202601010000" -e "202601312359"
```

Note: `-E` makes time filters apply to job end time instead of start time.

Example 9: Show total system summary (default mode)

```
$ qacct
Total System Usage
    WALLCLOCK         UTIME          STIME            CPU             MEMORY
==============================================================================
    108000        3000.000       1500.000       4500.000              0.000
```

Note: With no parameters, shows total system usage.

Example 10: Mixed mode (details + summary)

```
$ qacct -j
==============================================================
qname        normal
...
==============================================================
qname        normal
...

Total System Usage
    WALLCLOCK         UTIME          STIME            CPU             MEMORY
==============================================================================
    108000        3000.000       1500.000       4500.000              0.000
```

Note: When `-j` is used without a value, all job details are shown followed by the system summary.

Notes

1. Option combination rules

- Detail mode: `-j` with a value (`-j <job_id\|name\|pattern>`) shows detailed info only.
- Summary mode: no `-j`, or `-j` without a value, shows aggregated statistics.
- Filter combinations: multiple filters (`-j`, `-o`, `-h`, `-q`, `-l`, etc.) use AND logic.
- Special combinations:
  - `-l` and `-h` together show the intersection (jobs on hosts matching `-h` and meeting `-l` resource constraints).
  - `-t` must be used with `-j`, and `-j` must include a value.

2. Time filtering behavior

- Default range: without time parameters, defaults to jobs started within the last 30 days (paging size adjustable via `FSCHED_ACCT_PAGE_SIZE`).
- Time basis:
  - Without `-E`: filters by job start time (`start_time`).
  - With `-E`: filters by job end time (`end_time`).
- Time format: `[[CC]YY]MMDDhhmm[.SS]` (e.g. `202601150800`, `01150800`, `2601150800`, `01150800.30`).

3. Query performance

- Paging: non-specific queries (no `-j` or `-j` without a value) use paging to avoid memory spikes.
- Environment variable: `FSCHED_ACCT_PAGE_SIZE` controls page size (default 100).

qconf

Overview

`qconf` is a tool for querying Sun Grid Engine (SGE) configuration information. This implementation provides a compatibility interface for the FSCHED environment. The command lets users view queues, parallel environments, and other system configuration parameters.

Parameters

Supported Options

Queue Management

Option	Value Required	Value Range	Purpose	Key Differences
`-sql`	No	None	List all queue names	
`-sq`	Yes	Queue name	Show detailed configuration for a specific queue	Some fields are dummy values

Host Management

Option	Value Required	Value Range	Purpose	Key Differences
`-sel`	No	None	List all execution hosts	
`-se`	Yes	Hostname	Show detailed configuration for a specific host	Some fields are dummy values
`-sh`	No	None	List all administration hosts	Returns the control node

Option	Value Required	Value Range	Purpose	Key Differences
`-ss`	No	None	List all submit hosts	Returns the control node

Host Group Management

Option	Value Required	Value Range	Purpose	Key Differences
`-shgrpl`	No	None	List all host groups	Read from config file; returns `@allhosts` if not configured
`-shgrp`	Yes	Host group name	Show host group details	Read from config file
`-shgrp_tree`	Yes	Host group name	Show host groups in tree form	Displays host group hierarchy
`-shgrp_resolved`	Yes	Host group name	Show resolved host list	Expands all hostnames

Parallel Environments

Option	Value Required	Value Range	Purpose	Key Differences
`-spl`	No	None	List all parallel environments	Returns `smp` and `mpi` by default
`-sp`	Yes	Parallel environment name	Show parallel environment details	Read from config file

Users and Projects

Option	Value Required	Value Range	Purpose	Key Differences
`-suserl`	No	None	List all users	From config file or database
`-suser`	Yes	User name	Show user details	From config file or database; some fields are dummy values
`-sprjl`	No	None	List all projects	From config file or database
`-sprj`	Yes	Project name	Show project details	From config file or database

Access Control

Option	Value Required	Value Range	Purpose	Key Differences
`-sul`	No	None	List all user sets/ACLs	Read from config file
`-su`	Yes	ACL name	Show user set details	Read from config file
`-sm`	No	None	List all managers	Currently fixed to `root`
`-so`	No	None	List all operators	Fixed to `NONE`

Configuration Management

Option	Value Required	Value Range	Purpose	Key Differences
`-sconf`	Optional	`global` or hostname	Show global or host configuration	From system configuration

Option	Value Required	Value Range	Purpose	Key Differences
`-ssconf`	No	None	Show scheduler configuration	From system configuration
`-sconfl`	No	None	List all configuration hosts	Returns the control node

Complex Resources

Option	Value Required	Value Range	Purpose	Key Differences
`-sc`	No	None	Show all complex resource attribute definitions	Returns a predefined resource attribute list

Resource Quotas

Option	Value Required	Value Range	Purpose	Key Differences
`-srqsl`	No	None	List all resource quota sets	From the database
`-srqs`	No	Quota set name (optional)	Show quota set details; show all if no name is specified	Only name, description, enable status, and job/runtime/CPU limits are supported; complex filter rules are not supported

Scheduler Status

Option	Value Required	Value Range	Purpose	Key Differences
`-sss`	No	None	Show scheduler status	Returns control node information

Other Options

Option	Value Required	Value Range	Purpose	Key Differences
`-help`	No	None	Show help	Shows all supported options
`-version`	No	None	Show version	Shows wrapper version and build information

Advanced Queries

Option	Value Required	Value Range	Purpose	Key Differences
`-sobjl`	Yes	object_type attribute value	List objects matching the criteria	Partially supported

---

Queue Configuration Output (`-sq`)

Field	Description	Key Differences
`qname`	Queue name	From queue configuration
`hostlist`	Host list	From queue configuration
`seq_no`	Sequence number	Dummy value `0`
`load_thresholds`	Load thresholds	Fixed value `"np_load_avg=1.75"`
`suspend_thresholds`	Suspend thresholds	Dummy value `"NONE"`
`nsuspend`	Number of suspended slots	Dummy value `1`
`suspend_interval`	Suspend interval	Dummy value `"00:05:00"`

Field	Description	Key Differences
`priority`	Priority	From queue configuration
`min_cpu_interval`	Minimum CPU interval	Dummy value `"00:05:00"`
`processors`	Processor limit	Dummy value `"UNDEFINED"`
`qtype`	Queue type	Dummy value `"BATCH INTERACTIVE"`
`ckpt_list`	Checkpoint interface list	Dummy value `"NONE"`
`pe_list`	Parallel environment list	Fixed value `"make smp mpi"`
`rerun`	Rerun flag	Dummy value `"FALSE"`
`slots`	Total slots	From queue configuration
`tmpdir`	Temporary directory	Fixed value `"/tmp"`
`shell`	Shell path	Fixed value `"/bin/sh"`
`prolog`	Prolog script	From system configuration or `"NONE"`
`epilog`	Epilog script	From system configuration or `"NONE"`
`shell_start_mode`	Shell start mode	Fixed value `"posix_compliant"`
`starter_method`	Starter method	Dummy value `"NONE"`
`suspend_method`	Suspend method	Dummy value `"NONE"`
`resume_method`	Resume method	Dummy value `"NONE"`
`terminate_method`	Terminate method	Dummy value `"NONE"`
`initial_state`	Initial state	From queue configuration

Field	Description	Key Differences
`s_rt`	Soft real-time limit	From queue configuration or `"INFINITY"`
`h_rt`	Hard real-time limit	From queue configuration or `"INFINITY"`
`s_cpu`	Soft CPU limit	Dummy value `"INFINITY"`
`h_cpu`	Hard CPU limit	Dummy value `"INFINITY"`
`s_fsize`	Soft file size limit	Dummy value `"INFINITY"`
`h_fsize`	Hard file size limit	Dummy value `"INFINITY"`
`s_data`	Soft data limit	Dummy value `"INFINITY"`
`h_data`	Hard data limit	Dummy value `"INFINITY"`
`s_stack`	Soft stack limit	Dummy value `"INFINITY"`
`h_stack`	Hard stack limit	Dummy value `"INFINITY"`
`s_core`	Soft core file limit	Dummy value `"INFINITY"`
`h_core`	Hard core file limit	Dummy value `"INFINITY"`
`s_rss`	Soft RSS limit	Dummy value `"INFINITY"`
`h_rss`	Hard RSS limit	Dummy value `"INFINITY"`
`s_vmem`	Soft virtual memory limit	Dummy value `"INFINITY"`
`h_vmem`	Hard virtual memory limit	Dummy value `"INFINITY"`

Host Configuration Output (`-se`)

Field	Description	Key Differences
`hostname`	Hostname	From host configuration
`load_scaling`	Load scaling	Dummy value `"NONE"`
`complex_values`	Complex resource values	Dummy value `"NONE"` (load data is in the `load_values` field)
`load_values`	Load values	Real-time load from system information
`processors`	Processor count	From host configuration
`user_lists`	User list	Dummy value `"NONE"`
`xuser_lists`	Excluded user list	Dummy value `"NONE"`
`projects`	Project list	Dummy value `"NONE"`
`xprojects`	Excluded project list	Dummy value `"NONE"`
`usage_scaling`	Usage scaling	Dummy value `"NONE"`
`report_variables`	Report variables	Dummy value `"NONE"`

Parallel Environment Configuration Output (`-sp`)

Field	Description	Key Differences
`pe_name`	Parallel environment name	Fixed to `smp` or `mpi`
`slots`	Slots	smp=999, mpi=99999
`allocation_rule`	Allocation rule	smp="$pe_slots", mpi="$fill_up"
`accounting_summary`	Accounting summary	smp=TRUE, mpi=FALSE
`control_slaves`	Control slaves	Fixed value TRUE

Field	Description	Key Differences
`job_is_first_task`	Job is first task	Fixed value TRUE

User Information Output (`-suser`)

Field	Description	Key Differences
`name`	User name	From config file or database
`oticket`	Override tickets	Dummy value `0`
`fshare`	Functional shares	Dummy value `0`
`delete_time`	Delete time	Dummy value
`default_project`	Default project	From config file or database

Examples

Example 1: List all queues

```
qconf -sql
```

Sample output:

```
compute
test
```

Example 2: Show queue details

```
qconf -sq compute
```

Sample output:

```
qname                 compute
hostlist              compute[1-11]
seq_no                0
...
```

Example 3: List all execution hosts

```
qconf -sel
```

Sample output:

```
compute1
compute2
test1
```

Example 4: Show host details

```
qconf -se compute1
```

Sample output:

```
hostname              compute1
load_scaling          NONE
complex_values        NONE
...
```

Example 5: Show global configuration

```
qconf -sconf global
```

Sample output:

```
#global:
execd_spool_dir              /opt/fastone/wrappers/sge/spool
mailer                       /bin/mail
...
```

Example 6: List all host groups

```
qconf -shgrpl
```

Sample output:

```
@allhosts
```

Example 7: List all parallel environments

```
qconf -spl
```

Sample output:

```
smp
mpi
```

Example 8: List all users

```
qconf -suserl
```

Sample output:

```
root
```

Example 9: Show project details

```
qconf -sprj project_a
```

Sample output:

```
name project_a
oticket 0
fshare 0
```

```
acl NONE
xacl NONE
```

Example 10: List resource quota sets

```
qconf -srqsl
```

Sample output:

```
normal
high
medium
low
```

Notes

Unsupported Features

- All modification options (`-a*`, `-m*`, etc.): No configuration changes are supported
- Administrative options (`-k*`, `-clearusage`, etc.): No administrative operations are supported

Parameter Limitations

- `-sp` only supports predefined parallel environments (`smp`, `mpi`)
- `-sconf` only supports `global` configuration

qdel

Overview

`qdel` deletes jobs in the FSCHED system. It replaces the same command in SGE (Sun Grid Engine) and allows users to terminate specific jobs or all jobs for specified users.

Options

Optional Parameters

Option	Requires Value	Purpose	Notes
`-f`	No	SGE-compatible force-delete flag. In practice, deletions are always forced, so this flag has no effect.	Kept for compatibility; not required.
`-u`	Yes	Comma-separated list of users whose jobs should be deleted, or `"*"` for all users.	Non-existent users cause an error; use `"*"` with caution.

Positional Parameters

Parameter	Required	Purpose
`<job_task_list>`	No	Comma-separated list of FSCHED job IDs to delete, e.g. `12345,67890`.
		If **`-u`** is not used, this parameter is required or the command fails.

Examples

Example 1: Delete specific jobs

```
qdel 12345,67890
```

Terminates jobs with IDs `12345` and `67890`.

Example 2: Delete jobs for specific users

```
qdel -u user1,user2
```

Deletes all running jobs for users `user1` and `user2`.

Example 3: Delete jobs for all users (use with caution)

```
qdel -u "*"
```

Terminates all jobs in the system. This action is irreversible; use only with appropriate permissions.

Notes

1. Required parameters: Provide at least one of `<job_task_list>` or `-u`. If both are provided, both are applied.
2. User validation: If a user specified with `-u` does not exist and is not `"*"`, the command exits with an error.
3. Permissions: You can only delete your own jobs unless you have administrator privileges.

Unsupported Features

- SGE notification options (such as `-notify`) are not implemented.

qhold

Overview

`qhold` pauses (holds) specified jobs or tasks. Targets can be selected by job identifiers, user names, or hold types.

Options

Optional Parameters

Option	Required	Type	Purpose
`-h <hold_list>`	No	String	Specify hold types (such as `u`, `o`, `s`).
`-u <user_list>`	No	String	Filter jobs by user. Use `"*"` for all users, or a specific user (e.g., `userA`).

Positional Parameters

Parameter	Required	Type	Purpose
`job_task_list`	Yes (if `-u` is not used)	List	List of job/task identifiers to hold, e.g. `123`, `task456`, `userA.789`.

Examples

Example 1: Hold by job ID

```
qhold 123 task456
```

Effect: Holds job `123` and task `task456`.

Example 2: Hold by user

```
qhold -u userA
```

Effect: Holds all jobs for user `userA`.

Example 3: Hold by type (check compatibility)

```
qhold -h u job101
```

Effect: Attempts to hold job `job101` with type `"u"`. This may be limited by FSCHED/SGE differences.

Notes

Parameter Constraints

1. Mutual exclusion:
   - `-u <user_list>` and `job_task_list` cannot be used together.
     ```
     qhold -u userA job101  # Error: conflicting parameters
     ```
2. Required condition:
   - You must specify either `-u` or `job_task_list`.
3. User filtering limits:
   - If the user does not exist and `"*"` is not used, the command errors as if the job does not exist.

Feature Differences

SGE Feature	Current Status	Notes
`-h <hold_list>`	Partial support (WIP)	Mapping may be incomplete; refer to SGE docs
Notification options	Not supported	Email notifications and related features are not implemented

---

Additional Notes

- Job identifier format: `job_task_list` must follow SGE syntax (such as wildcards or range expressions).
- For detailed hold type definitions, refer to the official SGE documentation.

qhost

View host status, resources, and load information in the cluster.

Options

Supported Options

Option	Requires Value	Range/Value	Purpose	Main Differences
(default)	No	None	Show basic status and resource info for all hosts.	No difference
`-F`	Optional	Resource list (comma-separated)	Show all or selected resource attributes (25 total).	`m_topology` and `m_topology_inuse` are dummy "NONE".
`-h`	Yes	Host list (comma-separated)	Show only specified hosts; supports `*` wildcard.	No difference
`-j`	No	None	Show job info running on hosts.	Only running jobs; `master` column is dummy "MASTER".
`-l`	Yes	Resource expression	Show hosts that satisfy resource requirements.	Supports `arch=`, `num_proc=`, `mem_total=`, `mem_free=`.
`-cb`	No	None	Show core binding columns (default behavior).	NSOC/NCOR/NTHR columns shown by default.
`-ncb`	No	None	Suppress core binding columns	No difference

Option	Requires Value	Range/Value	Purpose	Main Differences
			(NSOC, NCOR, NTHR).	
`-q`	No	None	Show queue instances on hosts.	Queue type is dummy "BIP"; reserved slots are dummy 0.
`-u`	Yes	User name	Show only hosts running jobs for the user.	No difference
`-xml`	No	None	Output in XML.	XML schema differs slightly from native SGE.
`-help`	No	None	Show help.	Detailed help implemented.

Default Output Fields

Field	Description	Main Differences
`HOSTNAME`	Host name	First line is `global` and all fields are `-`.
`ARCH`	Host architecture (e.g. lx-amd64)	Offline nodes show `-`.
`NCPU`	CPU count	Offline nodes show `-`.
`NSOC`	Socket count	Offline nodes show `-` (hidden with `-ncb`).
`NCOR`	Cores per socket	Offline nodes show `-` (hidden with `-ncb`).
`NTHR`	Threads per core	Offline nodes show `-` (hidden with `-ncb`).
`LOAD`	Average load	Two decimals; offline nodes show `-`.
`MEMTOT`	Total memory	With unit (G/M); offline nodes show `-`.

Field	Description	Main Differences
`MEMUSE`	Used memory	With unit (G/M); offline nodes show `-`.
`SWAPTO`	Total swap	No unit; offline nodes show `-`.
`SWAPUS`	Used swap	No unit; offline nodes show `-`.

Resource Attributes for `-F`

Supports 25 resource attributes with `hl:` (host level) prefix:

Resource	Description	Format	Main Differences
`arch`	Host architecture	String	None
`num_proc`	CPU count	Float (6 decimals)	None
`mem_total`	Total memory	With unit `G`	None
`swap_total`	Total swap	No unit	None
`virtual_total`	Total virtual memory (mem + swap)	With unit `G`	None
`m_topology`	CPU topology string	String	Dummy "NONE"
`m_socket`	Socket count	Float (6 decimals)	None
`m_core`	Core count	Float (6 decimals)	None
`m_thread`	Thread count	Float (6 decimals)	None

Resource	Description	Format	Main Differences
`load_avg`	Average load	Float (6 decimals)	None
`load_short`	Short load (15s)	Float (6 decimals)	From fsched r15s
`load_medium`	Medium load (1m)	Float (6 decimals)	From fsched r1m
`load_long`	Long load (15m)	Float (6 decimals)	From fsched r15m
`mem_free`	Free memory	With unit `G`	None
`swap_free`	Free swap	No unit	None
`virtual_free`	Free virtual memory	With unit `G`	None
`mem_used`	Used memory	With unit `G`	None
`swap_used`	Used swap	No unit	None
`virtual_used`	Used virtual memory	With unit `G`	None
`cpu`	CPU utilization (%)	Float (6 decimals)	Calculated from load/ncpu
`m_topology_inuse`	CPU topology in use	String	Dummy "NONE"
`np_load_avg`	Normalized average load	Float (6 decimals)	load/ncpu
`np_load_short`	Normalized short load	Float (6 decimals)	r15s/ncpu
`np_load_medium`	Normalized medium load	Float (6 decimals)	r1m/ncpu

Resource	Description	Format	Main Differences
`np_load_long`	Normalized long load	Float (6 decimals)	r15m/ncpu

Output for `-j`

Shows jobs running on each host under the host line:

Field	Description	Main Differences
`job-ID`	Job ID	None
`prior`	Job priority (0–1)	None
`name`	Job name	None
`user`	Job owner	None
`state`	Job state	Only running jobs (`r`)
`submit/start at`	Submit/Start time	Formatted output
`queue`	Queue@host	Format: `queue@host`
`master`	Master/Slave indicator	Dummy "MASTER"
`ja-task-ID`	Array task ID	Empty for non-array jobs

Output for `-q`

Shows queue instances under each host line:

Field	Description	Main Differences
`queuename`	Queue name	None
`qtype`	Queue type	Dummy "BIP"

Field	Description	Main Differences
`used/reserved/total`	Used/Reserved/Total slots	`reserved` is dummy 0
`states`	Queue state code	Mapped from node states

Queue state codes:

- (empty) - normal
- `a` - alarm (node DOWN/DRAIN/FAIL)
- `u` - unavailable (node unavailable)
- `d` - disabled (node DRAIN)
- `E` - error (node error state)

Examples

Example 1: Show basic info for all hosts

```
qhost
```

Sample output:

```
HOSTNAME                ARCH         NCPU NSOC NCOR NTHR  LOAD  MEMTOT
MEMUSE  SWAPTO  SWAPUS
----------------------------------------------------------------------------
------------------
global                  -               -    -    -    -     -       -
-       -       -
node01                  lx-amd64       16    2    8    1  2.35   31.3G
12.5G   8192    1024
node02                  lx-amd64       32    2   16    1  4.20   62.5G
28.3G  16384    2048
```

Example 2: Show resource attributes for a host

```
qhost -h node01 -F mem_total,num_proc,load_avg
```

Sample output:

```
HOSTNAME                ARCH         NCPU NSOC NCOR NTHR  LOAD  MEMTOT
MEMUSE  SWAPTO  SWAPUS
-------------------------------------------------------------------------------
-------------------
global                  -               -    -    -    -     -       -
-       -       -
node01                  lx-amd64       16    2    8    1  2.35   31.3G
12.5G   8192    1024
  hl:mem_total=31.300000G
  hl:num_proc=16.000000
  hl:load_avg=2.350000
```

Example 3: Show jobs running on hosts

```
qhost -j
```

Sample output:

```
HOSTNAME                ARCH         NCPU NSOC NCOR NTHR  LOAD  MEMTOT
MEMUSE  SWAPTO  SWAPUS
-------------------------------------------------------------------------------
-------------------
global                  -               -    -    -    -     -       -
-       -       -
node01                  lx-amd64       16    2    8    1  2.35   31.3G
12.5G   8192    1024
  12345  0.50 test_job  alice  r  01/15/2024 10:30:00  compute@node01
MASTER
  12346  0.45 analysis  bob    r  01/15/2024 11:00:00  compute@node01
MASTER
```

Example 4: Filter hosts by resources

```
qhost -l arch=lx-amd64,num_proc=16
```

Effect: Show hosts with architecture lx-amd64 and 16 CPUs.

Example 5: Show queue instances

```
qhost -q
```

Sample output:

```
HOSTNAME                ARCH         NCPU NSOC NCOR NTHR  LOAD  MEMTOT
MEMUSE  SWAPTO  SWAPUS
-------------------------------------------------------------------------------
-------------------
global                  -               -    -    -    -     -       -
-       -       -
node01                  lx-amd64       16    2    8    1  2.35   31.3G
12.5G   8192    1024
  compute      BIP   4/0/16
```

Example 6: Suppress core binding columns

```
qhost -ncb
```

Sample output:

```
HOSTNAME                ARCH         NCPU  LOAD  MEMTOT  MEMUSE  SWAPTO
SWAPUS
-------------------------------------------------------------------------------
-------
global                  -               -     -       -       -       -
-
node01                  lx-amd64       16  2.35   31.3G   12.5G   8192
1024
```

Example 7: Filter hosts by user jobs

```
qhost -u alice -j
```

Effect: Show only hosts running jobs for user alice, with job details.

Example 8: Filter hosts by wildcard

```

```

```
qhost -h "node*"
```

Effect: Show all hosts that start with `node`.

Example 9: XML output

```
qhost -xml
```

Effect: Output host info in XML for programmatic parsing.

qmod

修改 FSCHED 中队列和作业的状态。

参数介绍

队列操作

选项	是否需要值	目标类型	功能描述	主要差异
`-d`	是	队列名	禁用队列，阻止后续作业派发到该队列	无差异
`-e`	是	队列名	启用队列，允许后续作业派发到该队列	无差异
`-sq`	是	队列名	挂起队列，并停住该队列中正在运行的作业	原生队列显示 `s`，wrapper 显示 `d`；原生作业显示 `S`，wrapper 显示 `s`
`-usq`	是	队列名	恢复队列，并继续该队列中被停住的作业	不区分 suspend 来源，可能恢复 `-sj` 作业
`-cq`	是	队列名	清除队列 error 状态	dummy 实现；队列存在时仅输出 no error
`-rq`	是	队列名	重新调度队列中已派发的作业	重新入队后原生显示 `Rq`，wrapper 显示 `qw`

作业操作

选项	是否需要值	目标类型	功能描述	主要差异
`-sj`	是	作业 ID	挂起作业	无差异

选项	是否需要值	目标类型	功能描述	主要差异
`-usj`	是	作业 ID	恢复被挂起的作业	无差异
`-cj`	是	作业 ID	清除作业 error 状态	dummy 实现；作业存在时仅输出 not in error state
`-rj`	是	作业 ID	重新调度作业	重新入队后原生显示 `Rq`，wrapper 显示 `qw`

自动判断目标类型的兼容选项

选项	是否需要值	目标类型	功能描述	主要差异
`-s`	是	队列名或作业 ID	自动判断目标后执行挂起操作	队列目标：原生队列显示 `s`，wrapper 显示 `d`；原生作业显示 `S`，wrapper 显示 `s`
`-us`	是	队列名或作业 ID	自动判断目标后执行恢复操作	队列目标不区分 suspend 来源
`-c`	是	队列名或作业 ID	自动判断目标后清除 error 状态	dummy 实现；目标存在时仅输出无 error 状态
`-r`	是	队列名或作业 ID	自动判断目标后执行重新调度操作	重新入队后原生显示 `Rq`，wrapper 显示 `qw`

通用选项

选项	是否需要值	功能描述	主要差异
`-f`	否	强制执行状态修改	部分支持；支持队列操作和作业挂起/恢复
`-help`	否	显示帮助信息	无差异

选项	是否需要值	功能描述	主要差异
`-version`	否	显示版本信息	wrapper 扩展，原生不支持

目标与多目标规则

目标格式

目标	说明
`queue`	队列名
`job_id`	数字作业 ID
`queue@host`	SGE queue instance 语法；当前不支持按单个主机上的 queue instance 操作，通常会报 invalid queue

多目标

同一个选项可以接收逗号分隔或空格分隔的多个目标：

```
qmod -d main,debug
qmod -sj 12345 12346
```

处理规则：

- 目标按命令行顺序逐个处理。
- 有效目标会继续执行，无效目标会输出对应错误。
- 只要任一目标失败，最终退出码为 1。
- 多个操作会按出现顺序执行，例如 `qmod -d main -e main` 会先 disable 后 enable。

使用示例

示例1：禁用队列

```
qmod -d main
```

效果：禁用 `main` 队列，阻止后续作业派发到该队列。

示例2：启用队列

```
qmod -e main
```

效果：启用 `main` 队列，允许后续作业派发到该队列。

示例3：挂起队列

```
qmod -sq main
```

效果：挂起 `main` 队列，并停住该队列中运行中的作业。

示例4：恢复队列

```
qmod -usq main
```

效果：恢复 `main` 队列，并继续该队列中处于 suspended 状态的作业。

示例5：挂起作业

```
qmod -sj 12345
```

效果：停住作业 `12345`。挂起期间作业资源不会释放。

示例6：恢复作业

```
qmod -usj 12345
```

效果：继续运行被挂起的作业 `12345`。

示例7：重新调度可重新运行作业

```
qmod -rj 12345
```

效果：若作业允许重新调度，则将其重新入队；否则输出 not restartable。

示例8：重新调度队列中的作业

```
qmod -rq main
```

效果：枚举 `main` 队列中已派发的作业，并逐个尝试重新调度。

示例9：多目标按顺序处理

```
qmod -d main,debug -e main
```

效果：先禁用 `main` 和 `debug`，再启用 `main`。

qrls

qrls

Release the hold state of jobs or tasks. This command mimics SGE (Sun Grid Engine) `qrls` behavior but is implemented on FSCHED.

Parameters

Parameter	Type	Required	Description and Constraints
`-h`	String	No	Specify hold types to release (such as `"s"`). Must be used with another valid parameter (such as `-u` or `job_task_list`); cannot be used alone.
`-u`	Username string or `*`	No	Specify user list (such as `"user1,user2"`). Mutually exclusive with `job_task_list`; at least one of them is required.
`job_task_list`	Positional parameter	Yes (if `-u` is not used)	Format `jobID.task_range` (for example `job123.0-5`). Mutually exclusive with `-u` and must follow FSCHED task identifier rules.

Examples

Scenario 1: Release all held tasks for a user

```
qrls -u user1
```

- Purpose: Release all held jobs/tasks for user `user1`.

Scenario 2: Release holds for specific tasks

```
qrls job123.0-5
```

- Purpose: Release the hold state for tasks `0` to `5` of job `job123`.

Scenario 3: Use **`-h`** with a user

```
qrls -h s -u user2
```

- Purpose: Release holds of a specific type (such as `"s"`) for user `user2`.

Notes

1. Mutual exclusion:
   - `-u` and `job_task_list` cannot be used together. If `-u` is not specified, `job_task_list` is required.
   - Example of invalid usage:

     ```
     qrls -u user3 job456.0  # Error: -u and job list cannot be used
     together.
     ```

2. **`-h`** dependency:
   - `-h` must be used with at least one valid parameter (`-u` or `job_task_list`); it cannot be used alone.
3. User validation:
   - If a username in `-u` is invalid and `"*"` is not used, the command errors indicating the task does not exist.
4. Format requirements:
   - `job_task_list` must strictly follow the job identifier format (such as `jobID.task_range`).

Unsupported SGE Features

The following SGE features are not implemented due to compatibility differences with FSCHED:

- Notification options (such as email notifications).

- Some advanced hold types may not map to FSCHED equivalents.

qrsh

Overview

`qrsh` is the FSCHED replacement for the SGE interactive job launcher. It submits parallel or resource-constrained tasks and supports core options for environment, resources, and job behavior.

Options

Supported Parameters

Option	Type/Range	Description
`-A`	Account name (string)	Specify the account for the job.
`-b`	`y/n` (case-insensitive)	Control whether to execute a binary directly; default is `yes`.
`-cwd`	None	Use the current working directory as the job working path.
`-hard`	None	Make subsequent `-l/-q` options hard constraints (must be satisfied).
`-M`	Email address (string)	Set the email address for notifications.
`-m`	`b/e/a` (characters)	Email trigger: `b` (begin), `e` (end), `a` (always). Use with `-M`.
`-N`	Job name (string)	Set the job name; no spaces or special characters.
`-now`	Enabled by default	Start the job immediately; no need to specify explicitly.
`-pe`	`mpi/smp` + slots	Define parallel environment and slots (e.g., `-pe smp 4`). Only `mpi` and `smp` are supported.

Option	Type/Range	Description
`-soft`	None	Make subsequent `-l/-q` options soft constraints (allow adjustment).
`-v`	`"VAR=value"` format	Set environment variables (one per option). Example: `-v "PATH=/new/path"`.
`-wd`	Path string	Set the job working directory, overriding `-cwd`.

Unsupported or Limited Options

Invalid Options

The following options are ignored in FSCHED:

- **`-notify`**, **`-noshell`**, **`-nostdin`**: ignored by the system.

Limited Options

Option	Limitation
`-h/-hold_jid/-pty`	Job hold and dependency features are not implemented.
`-l`	Supports `num_proc`, `mem_free`, `hostname`, but allocation logic may be incomplete.
`-q`	Only a single queue name is supported; scheduling behavior may differ from SGE.

Examples

Example 1: Set parallel environment and email notification

```
qrsh -pe smp 4 -M user@example.com -m e my_script.sh
```

Starts a job with the `smp` environment and emails on completion.

Example 2: Enforce hard resource constraints

```
qrsh -hard -l "num_proc=8,mem_free=16G" my_program --arg1
```

Requires 8 CPU cores and 16GB memory (must be satisfied).

Example 3: Set environment variables

```
qrsh -v "DEBUG=1" python my_script.py
```

Enables the `DEBUG` environment variable before running the script.

Notes

1. Defaults:
   - `-b` defaults to `yes`, running binaries directly.
   - `-now` triggers immediate start by default.
2. Resource limits:
   - When using `-l/-q`, confirm parameter compatibility with FSCHED configuration.
3. Environment export:
   - All environment variables are inherited by default (no `-V` required).

qsh

Overview

`qsh` submits jobs and supports specifying resources, accounts, and execution environments. It is a compatible implementation of the SGE `qsub` command, adapted for FSCHED.

Options

Parameter List

Option	Type	Purpose	Limits/Notes
`-A <account>`	Requires value	Specify the account for the job.	Must provide a valid account name.
`-cwd`	No value	Use the current directory as the job working directory.	Not enabled by default; must be specified explicitly.
`-hard`	No value	Subsequent `-l/-q` options are hard constraints (must be satisfied).	Default may be hard; if neither `-hard` nor `-soft` is specified, behavior is system-dependent.
`-l <resources>`	Requires value	Request resources, e.g. `mem_free=2G,hostname=node01,num_proc=4`.	Only some resources are

Option	Type	Purpose	Limits/Notes
		Supported: memory (`mem_free`), host (`hostname`), processors (`num_proc`).	available; multiple entries are comma-separated.
`-M <email>`	Requires value	Set the email address for notifications.	Must be used with `-m` to define triggers.
`-m <options>`	Requires value	Email trigger events (e.g., `b` start, `e` end).	Ensure mail is configured in the system.
`-N <name>`	Requires value	Set the job name.	Defaults to a random name; provide a meaningful one.
`-now`	No value	Run immediately (enabled by default).	Explicit use improves clarity.
`-p <priority>`	Requires value	Set priority (range: -1023 ~ 1024).	Admins only; ignored for normal users.
`-pe <type> <slots>`	Requires 2 values	Specify parallel type and slots (e.g., `mpi 4` or `smp 8`). Valid types: `mpi`, `smp`.	Both type and slots are required.

Option	Type	Purpose	Limits/Notes
`-q <queue>`	Requires value	Bind the job to a specific queue (single queue only).	May be limited or unstable.
`-soft`	No value	Subsequent `-l/-q` options are soft constraints (try to satisfy, can relax).	Default may be hard; mutually exclusive with `-hard`.
`-S <interpreter>`	Requires value	Specify script interpreter path (e.g., `/bin/bash`).	Path must be valid and support shebang.
`-V`	No value	Export all environment variables (enabled by default; no need to specify).	Disabling has no effect.
`-w <level>`	Requires value	Set job verification level.	May be unavailable or unstable.
`-wd <dir>`	Requires value	Specify the job working directory path.	Mutually exclusive with `-cwd`.

Examples

1. Basic submission

```
qsh -N "my_analysis" analysis_script.sh
```

2. Resource request and parallel job

```
# Request 8 cores, 16GB memory, and use MPI:
qsh -l "num_proc=8,mem_free=16G" -pe mpi 4 parallel_job.py
```

3. Queue binding and immediate execution

```
# Bind to batch queue and run immediately:
qsh -q batch -now my_script.sh
```

4. Email notifications

```
# Send an email to user@example.com on completion:
qsh -M user@example.com -m e report_generation.sh
```

Notes

Functional Limits

- `-p`: only administrators can set priority; normal users are ignored.
- `-l`: only `mem_free` (memory), `hostname` (host), and `num_proc` (processors) are supported.
- `-q`: only a single queue name is supported; behavior may be unstable.

Additional Notes

- Environment export: `-V` is enabled by default; no need to add it.
- Verification (`-w`) and notification options: may be unstable or not implemented; do not rely on them.

Incompatible Features

Option	Limitation
`-p`	Normal users cannot set priority.
`-q`	Only a single queue name is supported.

Option	Limitation
`-w`	Not reliably implemented; avoid using.
`-l`	Resource options are limited to `mem_free`, `hostname`, and `num_proc`.

qstat

View job queue status and details, with filters for users, states, and output formats.

Options

Supported Options

Option	Requires Value	Range/Value	Purpose	Main Differences
`-u`	Yes	User list or `*` (comma-separated)	Filter jobs by user (supports wildcards).	No difference
`-f`	No	None	Show full-format queue info with queue summary and job list.	No difference
`-j`	Yes	Job ID	Show detailed status for a specific job ID.	No difference
`-xml`	No	None	Output in XML format.	No difference
`-g`	Yes	`c/d/t`	Group results: cluster (`c`), job arrays (`d`), or parallel tasks (`t`).	`-g t`: all tasks show the same queue; the first is MASTER, others SLAVE.
`-s`	Optional	State character list	Filter by job state; if omitted, show all states.	No difference
`-q`	Yes	Queue name	Filter by queue.	No difference

Option	Requires Value	Range/Value	Purpose	Main Differences
`-t`	No	None	Show array job tasks (same as `-g d`).	No difference
`-r`	No	None	Show job resource requests.	Soft Resources always empty; Binding fixed to "NONE".
`-pri`	No	None	Show job priority info.	nurg/npprior/ntckts/ppri are dummy values (0.0).
`-urg`	No	None	Show job urgency info.	All fields are dummy values (0.0).
`-ext`	No	None	Show extended job attributes.	department is dummy "defaultdep"; tickets/share are dummy values.
`-ncb`	No	None	Suppress binding parameters (use with `-r`).	No difference
`-explain`	Yes	State character (a/c/A/E)	Explain queue states.	No difference
`-F`	Optional	Resource attribute list (comma-separated)	Show queue resource attributes.	Supported: mem_total, num_proc, qname, slots.
`-U`	Yes	User name	Show queues accessible to the user.	Simplified implementation.
`-ne`	No	None	Hide empty queues.	No difference

Option	Requires Value	Range/Value	Purpose	Main Differences
`-qs`	Yes	Queue state characters	Filter queues by state.	No difference
`-l`	Yes	Resource expression	Filter queues/jobs by resource.	Supports `arch=`, `num_proc=`, `mem_total=`.

Output Fields

Default Output

Field	Description	Main Differences
`job-ID`	Job ID	No difference
`prior`	Job priority (0–1)	No difference
`name`	Job name	No difference
`user`	Job owner	No difference
`state`	Job state	No difference
`submit/start at`	Submit/Start time	No difference
`queue`	Queue@host	No difference
`slots`	Job slots	No difference
`ja-task-ID`	Array task ID	No difference

Output for `-r`

Shows job resource requests under each job line:

Field	Description	Main Differences
`Full jobname`	Full job name	No difference
`Requested PE`	Requested parallel environment	Extracted from comment field
`Hard Resources`	Hard resource requirements	Resource priority is dummy value (0.000000)
`Soft Resources`	Soft resource requirements	Always empty (dummy value)
`Binding`	Core binding info	Fixed to dummy value "NONE"

Additional Columns for `-pri`

Field	Description	Main Differences
`nurg`	Normalized urgency	Dummy value 0.0
`npprior`	Normalized priority	Dummy value 0.0
`ntckts`	Normalized tickets	Dummy value 0.0
`ppri`	POSIX priority	Dummy value 0

Additional Columns for `-urg`

Field	Description	Main Differences
`nurg`	Normalized urgency	Dummy value 0.0
`urg`	Urgency	Dummy value 0.0
`rrcontr`	Resource reservation contribution	Dummy value 0.0

Field	Description	Main Differences
`wtcontr`	Waiting time contribution	Dummy value 0.0
`dlcontr`	Deadline contribution	Dummy value 0.0
`deadline`	Job deadline	Dummy value (empty string)

Additional Columns for `-ext`

Field	Description	Main Differences
`ntckts`	Normalized tickets	Dummy value 0.0
`project`	Project name	No difference
`department`	Department	Dummy value "defaultdep"
`cpu`	CPU usage	Empty for running jobs
`mem`	Memory usage	Empty for running jobs
`io`	IO usage	Empty for running jobs
`tckts`	Total tickets	Dummy value 0
`ovrts`	Override tickets	Dummy value 0
`otckt`	Other tickets	Dummy value 0
`ftckt`	Functional tickets	Dummy value 0
`stckt`	Share tree tickets	Dummy value 0
`share`	Share tree value	Dummy value 0.00

Output for `-f`

Shows full queue information with queue summary and job list:

Field	Description	Main Differences
`queuename`	Queue name	No difference
`qtype`	Queue type	Dummy value "BIP"
`resv/used/tot`	Reserved/Used/Total slots	`resv` is dummy value 0
`load_avg`	Average load	No difference
`arch`	Architecture	No difference
`states`	Queue state	No difference

Output for `-g c`

Field	Description	Main Differences
`CLUSTER QUEUE`	Cluster queue name	No difference
`CQLOAD`	Queue average load	No difference
`USED`	Used slots	No difference
`RES`	Reserved slots	Dummy value 0
`AVAIL`	Available slots	No difference
`TOTAL`	Total slots	No difference
`aoACDS`	Queue instance state summary	No difference
`cdsuE`	Queue state details	No difference

Output for `-j`

Shows job details, including:

Basic info: `Job Number`, `Job Name`, `Owner`, `State`, `Submission Time`, `Start Time`, `Queue`, `Slots`

Resource info (running/completed jobs only): `Requested Resources`, `Granted Resources`

Other info:

- `Parallel Environment` - extracted from comment, or "NONE"
- `Project` - from wckey
- `Department` - dummy value "defaultdep"

Output for `-g t`

Shows parallel task info:

Field	Description	Main Differences
`master`	MASTER/SLAVE indicator	All tasks show the same queue; first is MASTER, others are SLAVE

Examples

Example 1: Show detailed info for specific users

```
qstat -u userA,userB -f
```

- Effect: Shows full-format job info for users `userA` and `userB`.

Example 2: Filter running jobs

```
qstat -s r
```

- Effect: Lists only jobs in running state.

Example 3: XML output for a specific job

```
qstat -j 12345 -xml
```

- Effect: Displays job `12345` details in XML.

Example 4: Group array tasks and filter hold states

```
qstat -g d -s huo
```

- Effect: Groups by array jobs and filters jobs in user/admin hold states.

Example 5: Show job resource requests

```
qstat -r
```

- Effect: Shows full job name, PE, and resource requests (Hard Resources, Soft Resources, Binding).

Example 6: Show extended job attributes

```
qstat -ext
```

- Effect: Shows project, department, CPU/memory/IO usage, and ticket info.

Example 7: Show cluster queue summary

```
qstat -g c
```

- Effect: Shows queue summary including load, slot usage, and status.

Example 8: Filter by queue and show resource attributes

```
qstat -F -q compute
```

- Effect: Shows resource attributes (memory, CPU count, slots) for the specified queue.

qsub

Submit jobs to the FSCHED scheduler.

Options

Basic Options

Option	Description	Main Differences
`-b`	Run the command as a binary executable (y/n)	No difference
`-N`	Job name	No difference
`-cwd`	Use the current working directory	No difference
`-verify`	Verification mode; do not actually submit the job	No difference

Accounts and Projects

Option	Description	Main Differences
`-A`	Account name	No difference
`-P`	Project name	No difference

Scheduling and Queues

Option	Description	Main Differences
`-q`	Queue name	No difference
`-a`	Job start time (YYMMDDhhmm format)	No difference
`-h`	Hold the job after submission	No difference

Resource Requests

Option	Description	Main Differences
`-l`	Resource requirement list	Supports `mem_free`, `h_rt`, `num_proc`, `hostname`
`-pe`	Parallel environment and slot count	No difference
`-masterq`	Master queue (for parallel jobs); requires `-pe`	No difference
`-binding`	CPU binding policy; supports `linear`, `striding`, `explicit`, etc.	No difference

Priority

Option	Description	Main Differences
`-p`	Job priority (-1023 to 1024)	No difference
`-js`	Job share (larger SGE js means higher priority)	No difference

Input/Output

Option	Description	Main Differences
`-o`	Standard output path	No difference
`-e`	Standard error path	No difference
`-i`	Standard input path	No difference
`-j`	Merge stdout and stderr (y/n)	No difference

Email Notifications

Option	Description	Main Differences
`-m`	Mail options (b/e/a/s/n)	No difference
`-M`	Mail recipient address	No difference

Job Dependencies

Option	Description	Main Differences
`-hold_jid`	Job dependency list	No difference
`-hold_jid_ad`	Array job dependency list	No difference

Array Jobs

Option	Description	Main Differences
`-t`	Array task ID range; supports `start-end:step`	No difference
`-tc`	Limit concurrent array tasks	Extension option (not in SGE), maps to SLURM `--array=start-end%max`

Array format notes:

- `start-end` - task ID range
- `:step` - optional step (default 1)

Job Restart

Option	Description	Main Differences
`-r`	Job restartable (y/n)	No difference

Advanced Options

Option	Description	Main Differences
`-C`	Script directive prefix (default `#$`)	Prefix must start with `#` or error (SGE allows with warning)
`-@`	Read options from file	No difference
`-c`	Checkpoint options	Only supports `-c n` (disable checkpoint)
`-S`	Script interpreter	Validates shebang to match interpreter
`-shell`	Enable/disable shell wrapper (y/n)	Accepted but shell wrapper is always used
`-sync`	Wait for job completion (y/n)	No difference
`-w`	Verification options (e/w/n/v/p)	`v` is equivalent to `-verify`
`-notify`	Notify job before termination/suspension	Accepted but has no effect

Environment and Context

Option	Description	Main Differences
`-v`	Export specific environment variables (comma-separated)	No difference
`-V`	Export all environment variables	All environment variables are passed by default (equivalent to default `-V`)
`-ac`	Add context variables	Stored in job comment as `CONTEXT:var=value` (does not affect scheduling)
`-dc`	Delete context variables	Stored in job comment
`-sc`	Set context variables (replace all)	Stored in job comment

Resource Request Control

Option	Description	Main Differences
`-hard`	Hard resource request; subsequent `-l` and `-q` are parsed	Default is hard mode
`-soft`	Soft resource request; subsequent `-l` and `-q` are ignored	Soft mode ignores subsequent `-l` and `-q`

Job Execution Control

Option	Description	Main Differences
`-now`	Run immediately or fail (y/n)	No difference
`-wd`	Working directory; supports relative and absolute paths	No difference

---

Examples

Example 1: Basic submission

```
qsub -N myjob -o out.txt script.sh
```

- Effect: Submits the job and sets the output file name.

Example 2: Specify resource requirements

```
qsub -l num_proc=4 job.sh
```

- Effect: Requests 4 CPU cores.

Example 3: Set job dependency

```
qsub -hold_jid 123 analysis.py
```

- Effect: Runs after job ID 123 completes.

Example 4: Submit a parallel job

```
qsub -pe mpi 8 code.py
```

- Effect: Starts an MPI parallel environment with 8 slots.

Example 5: Submit an array job

```
qsub -t 1-5:2 array.sh
```

- Effect: Creates an array with step 2 (IDs: 1, 3, 5).

Example 6: Set start time and priority

```
qsub -a 202601101430 -p 500 delay_job.sh
```

- Effect: Sets the start time and priority.

Example 7: Set environment variables

```
qsub -v DEBUG=1,WORKERS=4 -cwd test.sh
```

- Effect: Exports variables and uses the current directory.

Example 8: Synchronous wait

```
qsub -sync y wait_job.sh
```

- Effect: Submits the job and waits for completion.

Example 9: Verification mode

```
qsub -verify myscript.sh
```

- Effect: Validates the job configuration without submitting.

Example 10: Read options from a file

```
qsub -@ options.txt myscript.sh
```

- Effect: Reads option configuration from a file.

Example 11: Binary command execution

```
qsub -b y hostname
```

- Effect: Runs the command as a binary executable.

Example 12: Custom script directive prefix

```
qsub -C '#@' custom_prefix.sh
```

- Effect: Uses `#@` as the script directive prefix (instead of default `#$`).

fscgdet

fscgdet

`fscgdet` is a tool that lets users remove processes managed by Fsched from the corresponding cgroup.

Usage

```
fscgdet <command>
```

DANGER

- This tool only works for Fsched jobs.
- This tool only works in proctrak/cgroup mode.
- This tool must have `setuid` permission; check after installation.

fseff

`fseff` is a port of Slurm's `seff` tool for printing a job's resource utilization efficiency. Because `seff` is written in Perl and depends on Slurm's Perl modules, and those modules are only available on CentOS 6, it was ported.

Usage

```
fseff <job id>
```

Output

Typical output:

```
Job ID:                 10003
Cluster:                fastone
User/Group:             root/root
State:                  RUNNING
CPU:                    1
CPU Utilized:           00:00:00
CPU Efficiency:         0.00% of 01:10:15 cpu-walltime
Job Wall-clock time:    01:10:15
Memory Utilized:        0KB (estimated maximum)
Memory Efficiency:      0.00% of (1000KB/node)
WARNING: Efficiency statistics may be misleading for running jobs.
```

Explanation:

- `Job ID` Job ID
- `Cluster` Cluster where the job ran
- `User/Group` User and group that own the job
- `CPU` Number of CPUs used by the job
- `CPU Utilized` CPU time used by the job, format `HH:MM:SS`
- `CPU Efficiency` Job CPU efficiency, ratio of actual used to requested
- `Job Wall-clock time` Total job runtime
- `Memory Utilized` Amount of memory consumed by the job
- `Memory Efficiency` Job memory efficiency, ratio of actual used to requested

fsjobs

fsjobs

Description

`fsjobs` is used to query and display job information in the cluster. Compared with `squeue`, fsjobs provides more job resource details, such as resource usage and job state. It also provides more filtering options, such as by job state, user, name, and so on.

`fsjobs` supports custom output formats; use `-o` to specify output fields.

`fsjobs` is included in `fsched` 10.26 and later.

Notes:

- If the command is used in `fsched` versions earlier than 10.26, some information cannot be obtained and some fields are unavailable.
- Information is collected once every 30s, so there may be some delay.

Options

```
Usage: fsjobs [OPTIONS]

Options:
  -h,--help                   Print this help message and exit
  -j,--job TEXT               Job ID to query
  -p,--partition TEXT         Partition to query
  -o,--format TEXT (Env:FSJOB_FORMAT)
                              Output format
  -s,--state UINT:value in {cancelled->4,complete->3,failed->5,pending-
>0,running->1} OR {4,3,5,0,1} [1]  ...
                              State to query
  -u,--user TEXT              User to query
  -n,--name TEXT              Name to filter
  -H,--no-header              Do not print header
  --list-fields               List available fields and exit
```

- `-j,--job TEXT`: Specify the job ID to query. Supported formats are "jobid" or "jobid_arrayid".
- `-p,--partition TEXT`: Specify the partition to query.

- `-o,--format TEXT`: Specify output fields. Also supports configuration via environment variable (FSJOB_FORMAT).
- `-s,--state UINT`: Specify job states to query. Supported states: `cancelled`, `complete`, `failed`, `pending`, `running`.
- `-u,--user TEXT`: Specify the user to query.
- `-n,--name TEXT`: Specify a job name filter. Wildcards are supported. For example, `fast-job-*` matches jobs starting with `fast-job-`.
- `-H,--no-header`: Do not print the header.
- `--list-fields`: List available fields and exit.

Custom Format

`fsjobs` supports custom output formats; use `-o` to specify output fields.

- Fields to output are separated by spaces, for example `-o jobid name state`.
- Each field supports the following syntax:

```
<field>[:[-]width[.precision][!]:[unit]] ... [delimiter="separator"]
```

Where:

- Field: the field to display. See the supported list below.
- `-`: right align. Default is left align.
- Width: field width. If the field length is less than the width, it is padded with spaces.
- Precision: precision for floating numbers. Default is 1. This option only applies to floating numbers.
- `!`: force truncation. If the field length is greater than the width, it is truncated.
- Unit: field unit. Supported: `K`, `M`, `G`, `T`, `P`, `E`, `Z`, `Y`.
- delimiter: separator between fields. Default is ` `.

Supported Fields

Field	Description
alloccpu	Allocated CPU count
allocmem	Allocated memory

Field	Description
allocnode	Allocated node count
command	Command executed
cpu	Requested CPU count
cpu_eff	Job CPU efficiency (ratio of actual used to requested)
group	Submitting user's group name
groupid	User's group ID
jobid	Job ID
mem	Requested memory
mem_eff	Job memory efficiency (ratio of actual used to requested)
name	Job name
node	Requested node count
nodelist	Nodes allocated to the job
partition	Partition where the job runs
rawid	For array jobs, the ID of the individual job
state	Job state
taskid	Task ID in an array job
time	Job runtime
timelimit	Job time limit
used_cpu	Total CPU time actually used (cumulative CPU time, user + kernel)

Field	Description
used_mem	Total current memory used
max_mem	Maximum memory used
user	Submitting username
userid	Submitting user ID

Example

For example, the default format is: `jobid:-12 name:12! state:10 partition:12! user:12! time:-10 timelimit:-10 req_cpu:-8 cpu_used:-8 cpu_eff:-8 req_mem:-8 mem_used:-8 mem_eff:-8 nodelist:16 delimiter=' '`

```
$ fsjobs
       JOBID NAME         STATE      PARTITION    USER               TIME
TIMELIMIT      CPU USED_CPU  CPU_EFF      MEM USED_MEM  MEM_EFF NODELIST
  5948342_12 wl_sub_17224 RUNNING    compute      admin            1:08:03
UNLIMITED        1     0:50       1%       1M       1M     100% compute-012
  5948342_29 wl_sub_17224 RUNNING    compute      admin            1:08:03
UNLIMITED        1     0:30       0%       1M       1M     100% compute-029
     5994579 wlsub_172241 RUNNING    compute      admin              39:56
UNLIMITED        1     1:09       2%       1M      66M    6600% compute-085
     5995201 wlsub_172241 RUNNING    compute      admin              39:56
UNLIMITED        1     0:00       0%       1M       0M       0% compute-064
     5995204 wlsub_172241 RUNNING    compute      admin              39:56
UNLIMITED        1     0:00       0%       1M       0M       0% compute-065
```

- Here `NAME` is forcibly truncated. In the example above, the `NAME` field width is 12, but `wlsub_172241` exceeds it, so it is truncated to `wlsub_17224`.

fsloads

fsloads

Description

`fsloads` is used to query and display node load information in the cluster. Compared with `sinfo`, it provides more detailed node load information.

`fsloads` supports custom output formats; use `-o` to specify output fields.

`fsloads` is included in `fsched` 10.26 and later.

Notes:

- If the command is used in `fsched` versions earlier than 10.26, some information cannot be obtained and will show as 0.
- Information is collected once every 15s, so there may be some delay.

Options

```
Usage: fsloads [OPTIONS]

Options:
  -h,--help                   Print this help message and exit
  -p,--partition TEXT         Partition to query
  -o,--format TEXT (Env:FSLOADS_FORMAT)
                              Output format
  -H,--no-header              Do not print header
  --list-fields               List available fields and exit
```

- `-p, --partition`: Specify the partition to query.
- `-o, --format`: Specify output fields. See the format below.
- `-H, --no-header`: Do not print the header.
- `--list-fields`: List available fields and exit.

Custom Format

`fsloads` supports custom output formats; use `-o` to specify output fields.

- Fields to output are separated by spaces, for example `-o name state`.

- Each field supports the following syntax:

```
<field>[:[-]width[.precision][!]:[unit]] ... [delimiter="separator"]
```

Where:

- Field: the field to display. See the supported list below.
- `-`: right align. Default is left align.
- Width: field width. If the field length is less than the width, it is padded with spaces.
- Precision: precision for floating numbers. Default is 1. This option only applies to floating numbers.
- `!`: force truncation. If the field length is greater than the width, it is truncated.
- Unit: field unit. Supported: `K`, `M`, `G`, `T`, `P`, `E`, `Z`, `Y`.
- delimiter: separator between fields. Default is `|`.

Supported Fields

Field	Description
alloc_cpu	Allocated CPU
alloc_mem	Allocated memory
buffers	Memory used for buffers in the system
cached	Memory used for cache in the system
cfg_mem	Total memory configured in FSCHED
cpu	CPU allocation and capacity (allocated / total)
cpu_idle	Percentage of CPU idle time
cpu_iowait	Percentage of CPU time waiting for I/O
cpu_irq	Percentage of CPU time handling IRQ
cpu_spec	CPU spec: boards/sockets/cores/threads

Field	Description
cpu_steal	Percentage of CPU time stolen by hypervisor
cpu_sys	CPU utilization in system time
cpu_user	CPU utilization in user time
idle_cpu	Unallocated CPUs in FSCHED
io	I/O rate
load	Overall CPU utilization
mem	Available memory
mem_avail	Total available memory in the system
mem_free	Free memory in the system
mem_total	Total memory in the system
node	Node name
partitions	Partitions the node belongs to
pg	Page I/O rate (pages/sec)
pg_faults	Page fault rate (pages/sec)
pg_in	Page-in rate (pages/sec)
pg_out	Page-out rate (pages/sec)
r15m	15-minute load average
r15s	15-second load average
r1m	1-minute load average

Field	Description
r_rate	Disk read I/O rate
rq15m	15-minute run-queue load average
rq1m	1-minute run-queue load average
rq5m	5-minute run-queue load average
sessions	Number of active login sessions
state	Node state
swap_free	Free swap space in the system
swap_total	Total swap space in the system
swp	Available swap space
tmp	Available space in /tmp
tmp_free	Free space in /tmp
tmp_total	Total space in /tmp
total_cpu	Total CPUs configured in FSCHED
users	Number of logged-in users in the system
ut	Total CPU utilization
w_rate	Disk write I/O rate

State Field

State	Description
DRNG	Node is in DRAIN state and is cleaning completed jobs

State	Description
DRNG*	Node is in DRAIN state and is cleaning completed jobs; network unreachable
DRAIN	Node is in DRAIN state
DRAIN*	Node is in DRAIN state; network unreachable
FAILG	Node is in FAILED state and is cleaning completed jobs
FAILG*	Node is in FAILED state and is cleaning completed jobs; network unreachable
FAIL	Node is in FAILED state
FAIL*	Node is in FAILED state; network unreachable
DOWN	Node is fully down
ALLOC	Node has running jobs
ALLOC*	Node has running jobs, but network unreachable
ALLOC+	Node has running jobs and is also cleaning completed jobs
COMP	Node has no running jobs and is cleaning completed jobs
COMP*	Node has no running jobs and is cleaning completed jobs; network unreachable
IDLE	Node is idle
IDLE*	Node is idle, but network unreachable

Example

For example, the default format is: `node:-11! state:7! partitions:20! cpu:-7 r15s:-4 r1m:-4 r15m:-4 load:-4 mem_avail:-9:M pg:-5 swp:-5:G tmp:-5:G users:-5 sessions:-8 delimiter=' '`

```
$ fsloads
           NODE STATE   PARTITIONS              CPU R15S  R1M R15M LOAD
```

```
MEM_AVAIL    PG   SWP   TMP USERS SESSIONS
     centos-001 IDLE    centos                0/16  0.0  0.0  0.1   1%
13915M   0.0    3G   38G     0        0
     centos-002 IDLE    centos                0/16  0.0  0.0  0.1   1%
14158M   0.0    3G   38G     0        0
    centos6-001 IDLE    centos6               0/16  0.0  0.0  0.0   0%
0M   0.0    0G    0G     0        0
    centos6-002 IDLE    centos6               0/16  0.0  0.1  0.1   1%
15423M   0.0    0G    2G     0        0
    compute-001 ALLOC   compute              80/80  0.1  0.1  0.5  31%
1632M   0.1    3G   37G     0        0
    compute-002 ALLOC+  compute              11/80  0.1  0.4  0.8   5%
2682M   0.1    3G   38G     0        0
    compute-003 ALLOC+  compute              31/80  0.0  0.3  1.5   4%
1639M   0.1    3G   38G     0        0
    compute-004 ALLOC   compute              80/80  0.0  0.7  2.2  39%
463M   0.1    3G   36G     0        0
    compute-005 ALLOC+  compute              21/80  0.6  0.7  1.1  25%
1941M   0.1    3G   37G     0        0
```

- In the default format, `NODE` is forcibly truncated. In the example above, the `NODE` field width is 20 and right-aligned, but the hostname length exceeds it, so it is truncated.
- In the example, `centos6-001` is an older node, so it does not support data collection and shows 0 values.

fsquota

fsquota

Description

`fsquota` is used to view user resource quota information (limits and usage) in the cluster. It displays account-associated limits, QoS limits, and current resource usage in a readable format.

Why Use fsquota

Before submitting jobs, use `fsquota` to check:

- How many more jobs you can submit
- How much CPU, GPU, and memory are still available
- Whether limits differ across partitions

This helps avoid job submission failures and plan work effectively.

Options

Option	Description
`-a, --account <account>`	Show limits for a specific account
`-q, --qos <qos_name>`	Show limits for a specific QoS
`-f, --format <format>`	Output format: `text` or `json`
`-w, --wide`	Show full output, no truncation
`--no-color`	Disable colored output

Quick Start

```
# View current limits
fsquota

# Check limits before submitting to the GPU partition
fsquota | grep -A 10 "gpu"
```

```
# See how many jobs you can still submit
fsquota | grep "Jobs"

# View available CPU and GPU
fsquota | grep -E "CPU|GPU"
```

Understanding the Output

Shown Resources

Resource	Description
Running Jobs	Maximum number of running jobs
Jobs	Maximum number of submitted jobs (running + pending)
Accruing Jobs	Maximum number of jobs that can accrue priority
CPU	Maximum CPU count
Memory(GB)	Maximum memory in GB
Nodes	Maximum number of compute nodes
GPU	Maximum GPU count

Partition-Specific Limits

The same resource may appear multiple times:

- General limits (no partition): limits applied when no partition is specified
- Partition-specific limits: limits applied when submitting to that partition (for example, `gpu-partition`)

Be sure to check the limits for the target partition where you will submit jobs.

Field Meanings

- LIMIT: Maximum usable amount
- USED: Amount currently in use

- AVAIL: Amount still available (LIMIT - USED)

When You See `-` (No Limit)

A dash indicates no limit is set for that resource. You can use all available resources of that type in the cluster.

When AVAIL Is 0

This means the limit has been reached. You must wait for running jobs to complete before submitting more jobs.

Account: (ALL)

This indicates default limits. If you have not configured specific limits, the cluster provides these defaults to ensure fair resource sharing.

Examples

Example 1: Basic Usage

```
$ fsquota
USER LIMITS (jxin)

Account: test    QoS: normal
──────────────────────────────────────────────────────────────────────────────
RESOURCE         LIMIT      USED      AVAIL  SOURCE
──────────────────────────────────────────────────────────────────────────────
Running Jobs         100         5         95  Assoc
Jobs                 200        10        190  Assoc
Accruing Jobs          -         -          -  (no limit)
CPU                 1000        80        920  Assoc
Memory(GB)           500        40        460  Assoc
Nodes                 20         2         18  Assoc
GPU                   10         1          9  Assoc

Account: test    QoS: normal    Partition: gpu-partition
──────────────────────────────────────────────────────────────────────────────
RESOURCE         LIMIT      USED      AVAIL  SOURCE
──────────────────────────────────────────────────────────────────────────────
Running Jobs          20         3         17  Assoc
Jobs                  40         5         35  Assoc
Accruing Jobs          -         -          -  (no limit)
CPU                  400        60        340  Assoc
Memory(GB)           200        30        170  Assoc
```

```
Nodes                    5         1          4  Assoc
GPU                     20         4         16  Assoc
```

This means:

- Without specifying a partition: you can submit 190 more jobs (200 - 10)
- On `gpu-partition`: you can submit 35 more jobs (40 - 5)
- The partition limits are stricter

Example 2: Check Before Submitting Jobs

```
# Prepare to submit 50 jobs to the GPU partition
$ fsquota | grep -A 5 "gpu-partition"
Account: test    QoS: normal    Partition: gpu-partition
─────────────────────────────────────────────────────────────────────────────
RESOURCE         LIMIT      USED      AVAIL  SOURCE
─────────────────────────────────────────────────────────────────────────────
Running Jobs          20         3         17  Assoc
Jobs                  40         5         35  Assoc

# You can only submit 35 more jobs, not 50. Adjust the plan accordingly.
```

Common Workflows

Before Submitting Jobs

```
# 1. View limits
fsquota

# 2. View a specific partition (if needed)
fsquota | grep -A 10 "partition name"

# 3. Confirm there are enough jobs/CPU/GPU available in the AVAIL column
```

When Job Submission Fails

```
# Check whether you hit the limit
fsquota | grep "Jobs"

# If AVAIL is 0, wait for running jobs to complete
# Or cancel some jobs to free quota
```

For Administrators

Report Mode

Use `--report` to view resource usage for all users. This helps identify users near limits and plan resource allocation.

```
# View all users
fsquota --report

# Filter by account
fsquota --report -a project1

# Filter by QoS
fsquota --report -q high

# Wide mode (no truncation)
fsquota --report -w

# JSON output for scripting
fsquota --report -f json
```

Report Output Example

```
USER         ACCOUNT      QOS        PARTITION            RUN_JOBS
SUBMIT_JOBS  ACCRUE_JOBS          CPU       MEM(GB)    NODES      GPU
------------------------------------------------------------------------------
------------------------------------------------------------
alice        project1     normal     -                       5/100
10/200           -      80/1000      40/500     2/20     1/10
alice        project1     normal     gpu-partition           3/20
5/40             -       60/400      30/200      1/5     4/20
bob          project2     high       -                       2/50
3/100            -       40/500      20/250     1/10     0/5
charlie      project1     normal     -                     95/100
190/200          -      920/1000     460/500    18/20    9/10
```

Interpreting the Report:

- Each row shows resources in the `used/limit` format
- Users are ordered by: user -> account -> QoS -> partition

- Charlie is close to limits (95/100 jobs, 18/20 nodes)
- Bob has plenty of available resources

Common Admin Tasks

Identify Users Near Limits:

```
# View sorted output for all users
fsquota --report | less

# Find users with high used/limit ratios
# Users near limits may need:
#   - Increased quota allocation
#   - Guidance on resource usage
#   - Investigation of stuck jobs
```

Monitor Account Usage:

```
# View all users under a specific account
fsquota --report -a project1

# Compare usage across accounts
fsquota --report -a project1 > project1.txt
fsquota --report -a project2 > project2.txt
```

Generate Usage Reports:

```
# JSON output for automation
fsquota --report -f json > usage_report.json

# Parse with jq or other tools
fsquota --report -f json | jq '.[] |
select(.qos_associations[].resources.cpu.available < 100)'
```

fsstat

fsstat

`fsstat` is based on the `sdiag` API and is used to evaluate and debug the status and rates of various RPCs in Fsched health.

Parameters

Parameter	Purpose	Default
-i, --interval	Query interval	1s
-o, --output	Output (jsonl) format	

Output

Typical output:

```
Mon Mar  4 04:55:16 2024

  Latency:        0ms
  Server:         Mon Mar  4 04:55:16 2024

  Server threads: 3
  Agent queue:    0
  Agent count:    0
  DBD agent queue:0

  Jobs:
    Submitted:           5 (   0       0)
    Started:             5 (   0       0)
    Completed:           0 (   0       0)
    Canceled:            0 (   0       0)
    Failed:              0 (   0       0)
    Pending:             0 (   0       0)
    Running:             5 (   0       0)

  RPCs:
            MESSAGE_NODE_REGISTRATION_STATUS( 1002):       25(   0       0)
0.000( 0.0   0.000)
                        REQUEST_BUILD_INFO( 2001):       12(   0       0)
```

```
0.000( 0.0   0.000)
                          REQUEST_JOB_INFO( 2003):        1(   0       0)
0.000( 0.0   0.000)
                         REQUEST_NODE_INFO( 2007):        1(   0       0)
0.000( 0.0   0.000)
                       REQUEST_PARTITION_INFO( 2009):     2(   0       0)
0.000( 0.0   0.000)
                           REQUEST_STATS_INFO( 2035):    12(   1       1)
0.000(-0.0  -0.000)
                     REQUEST_SUBMIT_BATCH_JOB( 4003):     5(   0       0)
0.000( 0.0   0.000)
                      REQUEST_COMPLETE_PROLOG( 6018):     5(   0       0)
0.000( 0.0   0.000)
                     ACCOUNTING_REGISTER_CTLD(10003):     1(   0       0)
0.000( 0.0   0.000)
```

Explanation:

- `Mon Mar 4 04:55:16 2024` indicates the timestamp for this sample point
- `Latency` is the time spent by the RPC that retrieves the data; under high slurmctld load this value increases (when there is no server thread)
- `Server` is the server-side time for this data
- `Server threads` is the current number of server threads; SLURM code limits this to 256
- `Agent queue` is the current length of outgoing messages
- `Agent count` is the number of messages being processed (one message requires two server threads)
- `DBD agent queue` is the number of messages sent to slurmdbd
- `Jobs` contains job-related information. Note that this section is not updated in real time. Each entry format is as follows (the meaning inside `<>` is not included in output):

  ```
  <Category>: <Total count> ( <Change from previous sample; negative
  means decrease> <Rate / s> )
  ```

- `RPCs` contains information about RPC calls. The format is:

  ```
  <RPC name>(<code>): <Total count>( <Change from previous sample;
  negative means decrease> <Rate / s> ) <Average execution time in
  seconds>( <Change from previous sample; negative means decrease> <Rate
  / s> )
  ```

FAQ

Cluster Configuration FAQ

1. Conditions that trigger reconfiguration

When head node HA is not configured for the cluster, reconfiguration is triggered under the following conditions:

- The user has changed the cluster, for example by adding or removing nodes or modifying the configuration.
- The user has manually used the "Reconfigure" feature.
- The management stack cannot obtain information from required services and uses reconfiguration to retrieve critical information again. These services include:
  - `fs-scale`: used to obtain cluster job information and maintain the DRAIN state of cluster nodes.
  - `fs-statesvc`: used to collect job information and analyze job states.

If head node HA is configured for the cluster, automatic reconfiguration is no longer performed.

Scheduler Command FAQ

1. How is priority determined? How can priority be adjusted?

View job priority

```
squeue # View job information. The larger the PRIORITY value, the higher
the priority.
JOBID PARTITION     NAME       USER       ST       TIME  NODES NODELIST
CPUS REASON          PRIORITY TIMELIMIT ACCOUNT
12345   compute    my_job     user1      R        02:15  1     node01
4    None            1000     01:00:00   myaccount
12346   compute    another_job user2     PD       00:10  1     (None)
4    Resources       900      02:00:00   otheraccount
12347   compute    test_job   user1      CG       00:05  1     node02
4    None            950      01:30:00   myaccount

squeue -j <job_id> # Query a specific job ID, including its priority.

squeue -u <username> # Query a specific user, including job priorities.
```

```
scontrol show job <job_id> # Query detailed information for a specific job
ID, including its priority.
```

Adjust priority

```
# Set priority when submitting a job
sbatch --priority=10000 my_script.sh # 1000 is the priority value. The
larger the value, the higher the priority.

# An administrator adjusts the priority of an already submitted job
scontrol update jobid=12345 priority=2000 # Change the priority of job ID
12345 to 2000.
```

Load Threshold FAQ

1. How do stop and resume work?

At the scheduler level, a `SIGTSTP` signal is sent first and can be trapped. Two seconds later, a `SIGSTOP` signal is sent and cannot be trapped.

2. Can all jobs be resumed?

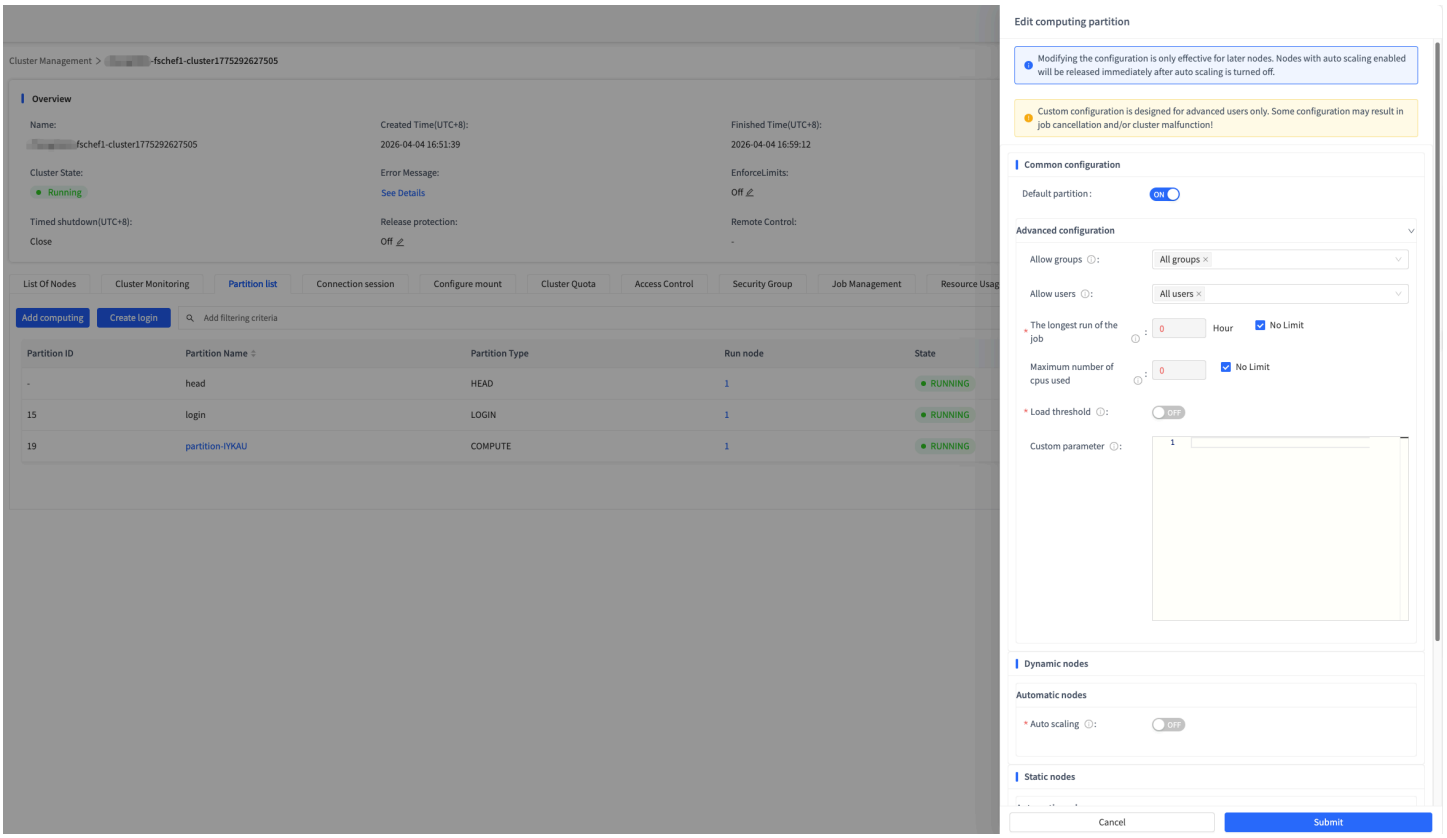
Jobs stopped because of the `LoadstopMem` or `LoadstopUt` load-threshold parameters can automatically resume and continue running after the load threshold is adjusted.

3. How do I configure load thresholds in FCP?

Load thresholds support custom parameters by partition, and they can be set when creating or editing a partition. The custom parameter configuration entry points are as follows:

- When creating a cluster, set them under Compute Partition Advanced Configuration > Custom Parameters.
- After the cluster is created, set them when creating a compute partition under Advanced Configuration > Custom Parameters.
- After the cluster or compute partition has been created successfully, set them from Partition List > Advanced Configuration > Edit Custom Parameters.

Example configuration:



For the supported parameters and detailed descriptions, see the Load Threshold Introduction.

4. Do stopped jobs release compute resources at the scheduler level? What about at the actual system level?

Memory is not released, but CPU resources are released.

5. What is the difference between load-stopped jobs, the `scontrol suspend` command, and `scancel -s SIGSTOP/SIGTSTP`?

- `loadstop`: This is a feature that automatically stops jobs under specific load conditions. It is typically used for dynamic job management to prevent cluster overload. Stopped jobs enter the `ST` state, memory remains allocated, and CPU resources are released.
- `scontrol suspend`: This is a manual command used to suspend a specified job. After it is executed, the job is suspended. Node resources are released, and users can resume the job later with `scontrol resume`.
- `scancel -s SIGSTOP/SIGTSTP`: This command sends signals and can force a specified job to pause by sending `SIGSTOP` or `SIGTSTP` through `scancel`. `SIGSTOP` cannot be caught, so the job stops immediately. `SIGTSTP` can be caught, so the job may choose how to handle it. It cannot stop jobs submitted by the `batch` command. Stopped jobs enter the `ST` state, memory remains allocated, and CPU resources are released.

6. What is the stopping order policy for jobs?

The stopping order policy follows priority, stopping lower-priority jobs first. If a job priority is changed manually, the automatic stopping order used by load thresholds does not change and will still stop jobs according to the old priority.

7. Can load threshold configuration be modified online?

Yes. In the web UI, go to Cluster Management > Partition List > Advanced Configuration > Edit Custom Parameters. See the screenshot in question 3 for reference.

8. Can a similar stop effect to `loadstop` be achieved manually, where resources are not released?

No.

9. Will manual user actions conflict with the automatic actions in load thresholds?

No.

10. What are the polling intervals for load checks and job/node stop-start actions?

- Load polling interval: Load monitoring, such as automatic job stop/start actions, polls at the configured interval. The default is 30 seconds.
- Node stop/start interval: The interval between node stop/start operations, such as draining a node, is set to 300 seconds. In other words, 300 seconds must pass between two drain operations on the same node.

Other FAQ

1. Child process handling: the difference between `cgroup` and `pgid`

`cgroup` mainly focuses on resource control and monitoring, while `pgid` focuses on process organization and management. The differences are as follows:

Feature	cgroup	pgid
Definition	Control group used for resource limits and monitoring	Process group ID used to identify a set of related processes

Feature	cgroup	pgid
Function	Resource limiting, usage monitoring, priority management	Signal management, job control
Use cases	Container management, multi-user resource control	Terminal job management, process relationship management
Scope	Resource management and limits	Process management and job control

Core Node Post-Restore Procedure

Assume we back up the Core node on a regular basis. When a Core node fails, we restore the backup data to a new Core node. However, a backup is only a point-in-time snapshot. After the backup, the cluster may have changed in ways that are not included in the backup data. Therefore, after restoring the data, additional handling is required to ensure normal cluster operation. Note: Handling differences between the backup and the live nodes is risky. Avoid doing this while there are jobs running on the affected nodes.

This document describes how to handle changes that occurred after the backup point:

Pre-Restore Preparation

1. If the Core node can be temporarily disconnected from the network, disconnect it before the restore to avoid data conflicts.
2. Because access to the management console is still required during the restore steps, keep the minimum network access needed to reach the console.

Nodes Removed After the Backup

After the Core node is restored from backup, if some nodes have been removed from the cluster, those removed nodes will still be considered part of the cluster. If such a node has been repurposed, the cluster configuration process will re-manage that node, which may impact workloads already running on it. Therefore, before restoring network connectivity, remove those nodes from the restore target cluster (the cluster being restored).

- Select these nodes in the UI and remove them.
- Because the network is disconnected and cleanup cannot complete, choose "Force Remove."
- Wait for node removal to complete.
- After finishing the backup restore and resolving cluster node differences, you can restore the Core node network.

Nodes Added After the Backup

After the Core node is restored from backup, if nodes were added to the cluster, those new nodes will disappear from the management platform after the restore. If you reconfigure the cluster at this time, the new nodes will be removed, which can impact workloads. Therefore, after the restore, re-add the new nodes to the restore target cluster (the cluster being restored):

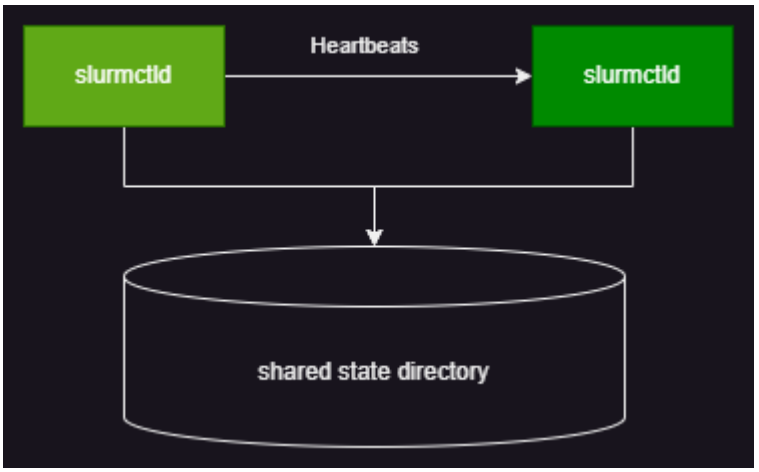
- Use the following command to set newly added nodes to the `DRAIN` state. The `reason` must be consistent; otherwise it will be reset to `IDLE`.

  ```
  scontrol update nodename=<node_name> state=DRAIN reason=MAINT:RECOVER
  ```

- After jobs finish, restore the Core node network and re-add the nodes to the cluster.

Head Node High Availability

Fsched supports high availability for head nodes in an active/standby setup. By configuring head node HA, you can ensure head node availability. When a head node fails, Fsched automatically switches to a standby head node to maintain service continuity. Multiple head nodes share state information through a shared file system to keep state consistent.



Terms

Head node: the node that hosts the fsched controller. Primary node: the head node that handles jobs under normal conditions. Standby node: the head node that provides hot standby.

Prerequisites

Head node HA uses a file system to transfer state information, so a shared file system is required. This is typically provided by a reliable NFS system. To ensure HA reliability, the NFS system must be reliable, and it should provide at least the following performance (because this shared storage is used by all clusters, the values below are per cluster):

- Read performance: > 10 reads/s and 300 KB/s
- Write performance: > 100 writes/s and 10000 KB/s
- Capacity: > 10 GB

This shared file system must be specified in the platform configuration using the following setting:

Configure Head Node HA

After the shared state directory is configured, head node HA is automatically enabled when the cluster has more than one head node.

How Head Node HA Works

- Fsched sorts head nodes by hostname. The first is the primary node; the others are standby nodes.
- The primary node periodically (~15s) sends heartbeat signals to standby nodes and uses the shared directory to notify them of head node status.
- When the primary node fails, the heartbeat exceeds the configured timeout (~30s). A standby node begins taking over the primary node's work.
- Other client components in Fsched check head node status and switch to a standby node by polling according to the configuration.

Handling Head Node HA Failures

Cluster Status After Head Node Failure

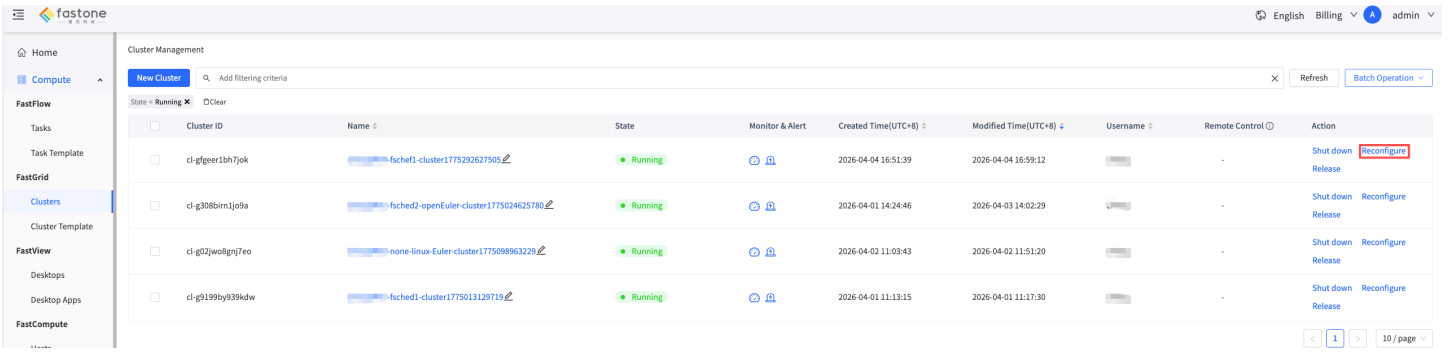
Because of the head node's special role, the platform does not make assumptions about head nodes, especially in the local system where the node state cannot be known with certainty. The platform does not automatically evict head nodes. Therefore, *after any head node failure, the platform cannot push configuration to the cluster normally*.

Head Node Recovery

After a head node failure, a standby node takes over head node operations. If the failure has been resolved, restore the head node using the steps below.

Node Recovery

After the head node issue is resolved, click "Reconfigure" in the UI to push configuration and restore the cluster.



Head Node Replacement

If the head node issue cannot be resolved, replace the head node by following these steps:

- Delete the failed head node in the UI.
- Add a new head node.

Temporary Handling During Head Node Failure

When the primary node fails, all cluster commands and cluster nodes try each controller in configuration order, and each node has a timeout. If it is clear that the primary node needs time to recover and business latency needs to be optimized, you can temporarily remove the failed node. The platform will reconfigure the cluster and select a new node as the primary node.

If you need to make cluster changes at this time, you should first restore the head node.

Head Node Primary/Standby Relationship

All slurm components try head nodes in configuration order. At any time, only one head node is the primary node, and the other head nodes are standby nodes. The primary/standby relationship is assigned as follows:

- Head nodes are assigned by hostname order. The first head node is the primary node, and the others are standby nodes. For example, `head-1` is the primary node and `head-2` is the standby node, and so on.

FAQ

- During job submission, the message "Job *jobid* is missing from controller" appears and the submission fails.

To ensure controller responsiveness and scheduling performance, all job state information is stored on the shared disk via asynchronous IO. We try to start scheduling and writing the job immediately after submission. However, writes may still be incomplete due to cache and other reasons. If the controller fails before job state is written, the backup controller loses information about the submitted job and reports that the job state does not exist. The user must resubmit the job.

- After the primary node fails, `scontrol ping` shows that the head node has not switched.

  The Fsched primary/standby label does not change with failover. When the primary node fails, the "primary" and "standby" roles do not change. After the standby node takes over cluster operations, it becomes a fully functional controller, but its role in the cluster is still "standby." If you want to reassign the primary/standby relationship, refer to Temporary Handling During Head Node Failure and adjust accordingly.

- Job runtime is longer than expected when a node is abnormal.

  Fsched records job runtime via the head node. If the head node is abnormal, the end of jobs cannot be recognized or recorded during the failover. This causes the job end time to be longer than expected.

- When a head node is abnormal, platform reconfiguration does not complete and stays in "configuring."

  Because of the head node's special role, the platform does not automatically evict head nodes. When any head node is abnormal, to avoid configuration differences across head nodes, the platform does not push configuration to the cluster. The cluster status remains "configuring" until the user intervenes, removes the failed node, and the platform reselects a primary node and pushes configuration.

Release Notes

Fsched 10.106

Updated: 2025.12.31

- slurm:
  - Added license server monitoring and license allocation management for the current cluster
  - Added adaptive scheduling
  - Added the `AllowUsers` parameter for partitions
  - Enhanced `fsched_list_job()` to support more pagination options
  - Added QoS denial behavior for "by job" and "by user"
  - Limited `wckey` input length to 42 characters
  - Added the `checkpoint/criu` plugin for job checkpoint and restore operations, supporting incremental checkpoints, pre-dump, and load-aware delays
  - Enhanced `cli_filter` to support wrappers and the custom field API
  - Fixed issues:
    - Fixed a problem where, if a user submitted a job without using `-c` to specify CPU cores, then after `slurmctld` restarted the core count was incorrectly stored as `0xfffe`, causing abnormal `CPUsPerTask` display and incorrect `bjobs` output
    - Fixed crashes caused by `sview`, `cpus_per_task` persistence, `gres_detail_str`, and related issues
- wrapper:
  - Added support for dynamic license accounting
  - Enhanced `fslsproc` with tree display and conflict detection
  - Added `btop` and `bbot` commands to change job order
  - Added `bpeek` to view the stdout/stderr of running batch jobs
  - Added `fsopt`, supporting `bsub`, `qsub`, `sbatch`, and `srun`; supports both interactive and batch commands
  - Enhanced `lshosts` to support the `-l`, `-T`, `-a`, and `-R` options, with filtering by host or cluster
  - Added support for the `fsquota` command to display resource quotas and limits
    - Displays accounting association limits and QoS policies
    - Displays current resource usage for jobs, CPU, memory, nodes, and GPU
    - Supports filtering by user, account, and QoS
    - Provides JSON output for programmatic access

- Added a new `cli_filter` adapter compatible with LSF and SGE
  - Supports the `bsub`, `qsub`, `qsh`, and `qrsh` commands
  - Adds custom fields to distinguish wrapper jobs from native SLURM commands
  - Adds comprehensive documentation including user guide, design, and custom fields
- Added the `-json` option to support JSON output
  - Added 6 custom output fields: `account`, `requeue`, `tmp_disk`, `min_nodes`, `max_nodes`, and `ntasks_per_node`
  - Expanded custom field support to a total of 86 field names, including 71 standard fields and 15 aliases
  - Improved field formatting and compatibility with LSF
  - Refactored the internal implementation to improve maintainability
- Enhanced the `statesvc` service `ListJobs` API with a force-refresh option
- `bsub`
  - Added the `-env` option, supporting the full LSF syntax: `all`, `none`, `selective`, `exclusion`, and `assignment`
  - Added support for `-H` (suspend job), `-Ne` (exit notification), and `-ti` (orphan process termination)
  - Added `ulimit` support, including `-M`, `-C`, `-c`, `-D`, `-F`, `-S`, `-v`, `-p`, `-T`, and `-ul`
  - Added support for `fsiod`, a native x11/stdio forwarding system. Experimental. Advantages: small footprint, fully asynchronous behavior, LSF-compatible behavior, and about a 10% performance improvement for `srun -x11`
  - `bsub -w` supports using `JOBNAME` as the job condition, allowing scripts to query job state directly by `JOBNAME`
    - Supports the `done(job_name)`, `ended(job_name)`, `exit(job_name)`, and `started(job_name)` syntax
- Fixed issues:
  - Fixed the lack of server-side user filtering in `qacct`, where each request fetched the full dataset and wasted bandwidth and memory
  - Fixed an issue where `qacct` without a specified job ID only fetched data and did not print results
  - Fixed pagination when `qacct` queried server data
  - Fixed an issue where `fsjobs` displayed only the current user's jobs by default

Fsched 10.96

Updated: 2025.09.25

- slurm:
  - `fsched ping`: added checks for jobs in the pending state
  - `fsched list jobs` API: added filter conditions for `comment`, `wckey`, `group_id`, and `node_name`
  - Removed erroneous logs for the `CgroupAutomount` configuration option
  - Added the `job_submit/intelliparams` job submission plugin
  - Changed `CR_LLN "load"` to use a ratio rather than available CPU count
  - Added the `FairshareUsed` factor, calculated from resources already consumed
  - Added the `--ext` option to `sshare` to include the `FairshareUsed` field
  - Added the `--ext` option to `sprio` to include the `FairshareUsed` field
  - Fixed issues:
    - Removed proactive loading while loading job information to avoid `slurmd` hangs
    - Used the generic `_get_avail_map` for batch job binding to fix node ordering during terminate job requests
    - Used connection, send, and receive timeout settings when fetching job details to fix `statesvc` hangs
    - Removed the `_access` check to fix permission denied errors for `prolog` and `epilog` tasks when using `root_squash`
    - Stopped sorting node names in `_job_test` to fix CPU binding issues
    - Fixed a race condition that caused `slurmctld` to crash during automatic scaling
    - Adjusted the log level for task cgroup errors
- wrapper:
  - Added the `bswitch` command to switch pending jobs to another queue
  - Added the `bstop` command to stop running jobs
  - Added the `bresume` command to resume stopped jobs
  - Added the `bhist` command to display job history
  - Added the `lsinfo` command
  - `bhosts`: added the `-a`, `-aff`, `-alloc`, `-e`, `-x`, `-X`, and `-R` options; added filtering by `cluster_name`; fixed status display for the `-l` and `-m` options
  - `lsload`: added the `-I`, `-w`, `-l`, `-N`, `-E`, `-R`, and `-a` options, and added filtering by host or cluster
  - `statesvc`: added expanded node lists (`expanded nodelists`)
  - Added support for mapping `bsub -G` to the Slurm account
  - Added support for `fscgdet` on both cgroup v1 and v2

- `statesvc`: added job extra information for `intelliparams`
- `bjobs`:
  - Switched to using the Fsched API with server-side filtering to load job information
  - Added the `start time` and `finish time` fields
- `bqueues`:
  - Added the `-m cluster_name` option
  - Added `loadSched` / `loadStop` information to the `-l` output
  - Added the `-alloc` option
  - Added the `-u user,all` option
  - Added `JL/U` and `JL/H` output
- Fixed issues:
  - Fixed an issue where `bsub -I` did not correctly forward command arguments, and fixed the permission issue with `bsub -Ep`
  - `bjobs`: fixed `-A` and `-UF`, fixed display by `array_job_id` list, fixed memory usage display, and added job descriptions, scheduling parameters, and resource requirement details to the `-l` output
  - `bqueues`:
    - Fixed scheduling parameter display in the `-l` output
    - Fixed multi-task jobs
    - Fixed `-m all`
    - Fixed error messages and error codes when a partition or host could not be found
    - Fixed `Users` in the `-l` output

Fsched 10.77

Updated: 2025.03.14

- Added support for systems using cgroup v2
- Added `LoadStop` and `LoadSched` parameter settings based on CPU load
- Added support for `bjobs -l` to display task information submitted by other users
- Fixed a number of known issues

Fsched 10.62

Updated: 2024.12.13

- Added support for configuring multiple partition administrators, authorizing them to cancel any job in the partition and control whether the partition accepts jobs (enable/stop)

- Added support for setting the maximum available CPU count at the partition level
- Added QoS policy support for setting the maximum resource minutes that all running jobs can use for each account or user; when a job exceeds the configured limit, it remains pending
- Allowed updating job memory, requiring the `select/cons_tres_ex` plugin when the job is running or pending
- Allowed updating job CPU allocation, only for single-node jobs and requiring the `select/cons_tres_ex` plugin
- Added support for querying job usage information, node load information, and user usage information for completed jobs
- Added parsing for parts of the `qsub` and `sqtat` command parameters in the SGE wrapper
- Fixed a number of known issues

Fsched 10.37

Updated: 2024.09.15

- Allowed users to increase the time limit of already submitted jobs
- Allowed configuring a partition-level option to kill jobs that exceed their requested memory
- Added `loadStop` and `loadSched` settings based on CPU and memory utilization
- Avoided potential job failures during scheduling when the authentication system, such as LDAP or NIS, becomes unavailable
- Added statistics such as resource usage for running jobs
- Improved response speed when canceling `srun` jobs
- Added node load and job load collection mechanisms, and used them to improve the output of the `lsload` and `bjobs` commands in the LSF wrapper
- Improved the failover mechanism in HA scenarios to shorten switchover time
- Improved stability under high load