# What is a Job Scheduler?

If you have ever worked on a busy, shared group server without a job scheduler, you quickly learned how troublesome it is to coordinate work with others. A batch job scheduler is a tool which manages coordination of work and provides a simple but powerful interface to use in developing workflows, pipelines and just submitting ad-hoc computational jobs. Submitted jobs, which amount to a request for a set of resources along with a description of the work to be performed, go into a queue where the scheduler applies access and priority policies to assign the work to the proper resources in the proper order. All while taking advantage of every opportunity to improve overall utilization of the available computational resources.

CZBiohub HPC computational resources use Slurm as the job scheduler. From the [SchedMD Slurm](http://slurm.schedmd.com/overview.html) Slurm overview:

The Simple Linux Utility for Resource Management (SLURM) is an open source, fault-tolerant, and highly scalable cluster management and job scheduling system for large and small Linux clusters. SLURM requires no kernel modifications for its operation and is relatively self-contained. As a cluster workload manager, SLURM has three key functions. First, it allocates exclusive and/or non-exclusive access to resources (compute nodes) to users for some duration of time so they can perform work. Second, it provides a framework for starting, executing, and monitoring work (normally a parallel job) on the set of allocated nodes. Finally, it arbitrates contention for resources by managing a queue of pending work.

All site cluster resources are managed by a Slurm instance with four main open partitions:

| Partition | Description |
| --- | --- |
| interactive | Provides quick access to workstation nodes. |
| cpu (default) | Nodes that provide CPU-only processing. |
| gpu | Nodes which contain GPU co-processers. |
| preempted | Scavenger partition allows use of all resources |

[ Standard partitions ]

Additional partitions may be present on some resources where necessary to support specific workloads. Resources are selected by using a combination of partition, node, core, memory, GRES and/or constraints (features).

This configuration is a starting point, from which the configuration is expected to evolve to meet the needs of the community as they arise. Please [Contact Us](/Contact_Us) if the scheduler config is not working for a given workflow or software as we are happy to make adjustments, create Quality of Service rules, reservations or help develop tools and workarounds.

# Resource Control and Fairshare

SLURM allows control of resources and access to resources based on a set of policies which are expected to evolve over time in response to the needs of the community and projects which use these resources. The primary way access is granted is through a mechanism known as fairshare. Fairshare attempts to adjust a user or group or projects job priority based on historical usage. When it works well it achieves two primary goals:

1. In the event there are one or just a few users it allows them access to as many resources as are available.
2. When there are many users and thus contention for resources, it ensures that users with few jobs or historically light usage have their jobs moved higher in the queue to maintain a fair distribution of resources.

Fairshare can be weighted, applied to groups and modified in many ways to ensure that individuals, groups and projects get the access they need. In addition, fairshare can be modified in the other direction, limiting individuals, groups and projects to a subset of the available resources. Ultimately the goal of a fairshare scheduler is two-fold:

1. Fairly distribute the available resources over time, maximizing overall utilization.
2. Ensure that all users are equally (un)happy.

If you find yourself more (un)happy than other users, please [Contact Us](/Contact_Us) and let us know so we can work with you to bring your (un)happiness back in line with the average level.

# Quality of Service

A QoS can be created to give a user, group or project priority access to some set of resources. A common use case for QoS is to provide fast turnaround for workloads initiated from web services or for data processing pipelines that ingest and precess data from instruments. When used for “normal” user submitted workloads it often becomes a restriction/throttle over time and most people who start with a QoS wind up preferring fairshare instead. However, in cases where workloads need to be throttled, this is a handy tool. For instance, the interactive partition uses a QoS to limit users to a set number of jobs and CPUs in order to ensure there are always at least some resources available for others to get fast turnaround on interactive sessions.

# Reservations

In cases where a project has a deadline or which requires quick turnaround and access to resources that are in shorter supply (nodes with large local scratch space, for example), a reservation can be created to set aside the resource for a limited time for the given project. In almost all cases this will reduce the overall utilization of the cluster for everyone, however, so it should be used sparingly and only when there is no other viable solution. Please [Contact Us](/Contact_Us) to get more information about what is possible or to request a reservation.