HPC@PoliTO

Job Scheduling Policy

Version: 1.0

Politecnico di Torino

www.hpc.polito.it

1 Overview

The Legion cluster uses **SLURM** (https://slurm.schedmd.com/documentation.html) as its workload manager. As typically more jobs are submitted than there are free resources, SLURM must decide which job to start next. This decision is made through a **numeric priority score**: the higher the score, the sooner your job runs.

Two main scheduling mechanisms work together:

- **Multifactor priority** — computes a score for every pending job based on several weighted components described in this document.
- **Backfill scheduling** — allows a lower-priority job to start early if doing so does not delay any higher-priority job's estimated start time. A small, short job may therefore run before a large one even with a lower score.

Practical implication: accurately specifying your job's wall time with `--time` helps the backfill scheduler place your job more aggressively. Overestimating runtime reduces your chances of early backfill placement.

2 Priority score

The priority score is computed as a weighted sum of several factors. Each factor is normalized by SLURM to a value between 0 and 1; the weights below are the configured values for this cluster.

2.1 Priority calculation

Given the weights configured on Legion, the full SLURM priority formula reduces to:

$$\textbf{Priority} = \mathbf{30\,000} \times \text{Partition} + \mathbf{3\,000} \times \text{Fairshare} + \mathbf{3\,000} \times \text{QOS} + \mathbf{2\,500} \times \text{JobSize} + \mathbf{1\,500} \times \text{Age} + \mathbf{1\,000} \times \text{GPU_TRES}$$

Rule of thumb: the partition tier is the dominant factor — it outweighs all other components combined at maximum values. This ensures that jobs scheduled on restricted partitions (see Manual, Chapter 1) are always assigned a much higher priority that those scheduled on open-access partitions. Within the same partition tier, **fairshare** and **QOS** are the next strongest levers available to users.

2.2 Weight table

Component	Weight	Meaning
Partition	30 000	Tier of the partition submitted to (see Section 3.1)
Fairshare	3 000	Remaining share of cluster resources for your group
QOS	3 000	Quality-of-Service level assigned to your account
Job size	2 500	Larger jobs (more CPUs or GPUs) score slightly higher
Age	1 500	Jobs waiting longer accumulate a small priority boost
GPU TRES	1 000	Per-GPU resource weight, added on top of job-size factor

All other SLURM priority components (e.g. nice, association) are zero-weighted on this cluster and have no effect on scheduling order.

3 Component details

3.1 Partition tier

Every partition has a configured `PriorityTier` and `PriorityJobFactor`. These multiply together to produce the effective partition weight. On this cluster there are two bands:

Class	Example partitions
Open-access (Tier 1)	`cpu_sapphire`, `gpu_a40`, `gpu_h200`, ...
Reserved (Tier 3)	`lining_cpu`, `fair_gpu`, `cgvg_gpu`, ...
Education (Tier 1)	`edu_sapphire`, `edu_a40`, `edu_v100`

Jobs subimtted to reserved partitions achieve the highest possible priority through `PriorityJobFactor=3` (effective partition weight = $30\,000 \times 3 = 90\,000$) rather than through preemption — a reserved partition job still queues behind already-running jobs on any given node.

3.2 Fairshare

Fairshare tracks how much compute each account has consumed over the past **30 days** (half-life of the decay function). The factor ranges from 0 to 1:

- **Factor** > 0.5: your group has used *less* than its proportional share. Priority is boosted — up to $+3\,000$ points.

- **Factor** < 0.5: your group has used *more* than its share. Priority is reduced.

Check your current fairshare factor with:

```
sshare -u $USER -a
```

The column `FairShare` shows your factor.

3.3 Age boost

The longer a job waits in the queue, the higher its age factor — growing linearly up to a maximum wait of **1 day**, after which it is capped at 1.0 (maximum contribution: +1 500 points). The age boost is designed to prevent indefinite starvation of small or low-priority jobs, but it is intentionally weak relative to partition and fairshare.

3.4 QOS

Quality-of-Service levels are currently assigned by administrators as the default QOS for all users.

3.5 Job size and GPU TRES

Larger jobs receive a slightly higher job-size factor (proportional to the fraction of cluster resources requested). Jobs requesting GPUs receive an additional TRES weight of 1 000 per GPU resource factor, reflecting the higher cost of GPU allocations. In practice, these two components are secondary — they rarely change scheduling order between jobs that differ significantly in partition tier or fairshare.

4 Worked examples

The following examples use realistic but illustrative normalised factor values. Run `sprio -u $USER` to see your actual live priority breakdown.

> **Example A** — Standard CPU job, fresh submission, average fairshare

Component	Factor	Points
Partition (`cpu_sapphire`, Tier 1)	1.0×1	30 000
Fairshare (used ≈50% of share)	0.50	1 500
QOS (default)	0.50	1 500
Job size (8 CPUs, factor ≈ 0.25)	0.25	625
Age (just submitted)	0.00	0
GPU TRES (no GPUs)	—	0
Total		**33 625**

Example B — Same job, 12 hours later in the queue

Component	Factor	Points
Partition (`cpu_sapphire`, Tier 1)	1.0	30 000
Fairshare	0.50	1 500
QOS	0.50	1 500
Job size	0.25	625
Age (12 h/24 h = 0.5)	0.50	**750**
Total		**34 375**

Age adds only +750 over 12 hours of waiting — partition and fairshare dominate.

Example C — GPU job, under-utilised group (high fairshare)

Component	Factor	Points
Partition (`gpu_a40`, Tier 1)	1.0	30 000
Fairshare (group used little of its share)	0.85	2 550
QOS	0.50	1 500
Job size (32 CPUs + 4 GPUs, factor ≈ 0.40)	0.40	1 000
Age (6 h waiting, factor = 0.25)	0.25	375
GPU TRES (4 GPUs, factor ≈ 0.50)	0.50	500
Total		**35 925**

Example D — Reserved partition, fresh GPU job

Component	Factor	Points
Partition (`fair_gpu`, Tier 3, JobFactor 3)	1.0×3	**90 000**
Fairshare	0.50	1 500
QOS	0.50	1 500
Job size (factor ≈ 0.30)	0.30	750
Age (just submitted)	0.00	0
GPU TRES (4 GPUs, factor ≈ 0.50)	0.50	500
Total		**94 250**

A reserved-partition job scores $\approx 2.7\times$ higher than an open-access GPU job (Example C), purely from the partition weight difference.

Key takeaway: the difference between Examples C and D (≈36 k vs ≈94 k) is entirely due to partition tier. Within the open-access tier, fairshare (Example C vs Example A: +1 425 points) is the primary lever users can influence through their submission behaviour.

5 Practical tips

Submit to the right partition. Choosing a partition with a longer `MaxTime` (e.g. `cpu_sapphire_ext` at 120 h instead of `cpu_sapphire` at 24 h) does *not* increase your priority — both are Tier 1. Use the partition whose wall-time limit you actually need; the scheduler cannot distinguish intent.

Request only what you need. The job-size factor slightly favours larger jobs, but over-requesting CPUs or memory makes your job harder to fit into available slots, increasing queue time in practice. Accurately specified requests almost always result in faster starts.

Monitor your fairshare. Run `sshare -u $USER -a` regularly. If your `FairShare` factor is below 0.3, your group is above its historical share and will be deprioritised. Spreading submissions over time helps maintain a healthy fairshare level.

Use wall-time estimates accurately. Accurate `--time` values allow backfill scheduling to slot your job into gaps between larger jobs. A job estimated at 2 h may start hours before a lower-priority job estimated at 24 h, even if both are in the same queue.

Chain jobs with dependencies. Use `--dependency=afterok:`*`JOBID`* to sequence work rather

than submitting large batches simultaneously. This keeps the queue readable and allows SLURM to plan resource reservations more effectively.

6 Useful commands

Command	What it shows
`squeue -u $USER`	Your pending and running jobs
`sshare -u $USER -a`	Fairshare factor and historical usage
`sprio -u $USER`	Priority component breakdown per job
`sinfo -p <partition>`	Node availability in a given partition
`sacctmgr show assoc u=$USER`	QOS and account associations
`scontrol show job <ID>`	Full details of a specific job