GPU SMITH
COMPUTE ENGINEERING

GPU SMITH / RESEARCH & PRACTICAL GUIDANCE

Slurm GPU Accounting: Billing Weights and Utilization Proof

> Plan and build private AI compute with GPU Smith. We connect workload requirements with GPU hardware, networking, deployment and operating decisions so your infrastructure fits the work it must perform.

Published by GPU Smith · 2026-11-09 · slurm gpu accounting · slurm tresbillingweights gpu · sacct gpu hours · sreport gpu utilization · accountingstoragetres gres gpu · slurm fairshare gpu · gpu chargeback showback · dcgm gpu utilization monitoring

Original article: https://gpusmith.com/articles/en/slurm-gpu-accounting-billing-weights-utilization



In this article

1. Executive Summary
2. Introduction and Background
3. The Four Questions and the Four Ledgers
4. Allocation Records: From sacct to GPU-Hours
5. Billing Weights, Units and Fairshare Usage
6. Observed Activity: Task Summaries, Sampling and MIG
7. Joining Telemetry to Jobs
8. Implementation Guidance: Reconciliation, Retention and Promotion
9. Data Analysis and Evidence
10. Acceptance Cases (Hypothetical Examples)
11. Implications and Future Directions
12. Frequently Asked Questions (FAQs)
13. Conclusion

Executive Summary

Slurm GPU accounting is four separate ledgers that are often read as one. The **allocation ledger** records what was reserved: `AllocTRES` is populated only after a job starts running and is blank for pending jobs (slurm.schedmd.com). The **duration ledger** records how long the reservation was held, from start and end timestamps, with a separate `Suspended` field. The **billing ledger** records what a site decides to charge: the billing TRES (Trackable RESource) is derived from partition `TRESBillingWeights`, and by default jobs are billed on allocated CPUs only, so a GPU adds nothing to the bill until a GPU weight is set (slurm.schedmd.com). The **activity ledger** records what a device was observed doing, and it is the weakest of the four.

The activity ledger is weak because Slurm's per-task GPU figures, `gres/gpumem` and `gres/gpuutil`, are high-water marks across the ranks of a step (slurm.schedmd.com). They are unavailable for NVIDIA Multi-Instance GPU (MIG) devices, because NVML (the NVIDIA Management Library) does not support utilization metrics for MIGs (slurm.schedmd.com). A university operator documents the consequence directly: missing fields do not prove zero usage (icarus.hpc.ntua.gr). A high-water mark therefore cannot be integrated into productive GPU time, and an absent field is an unknown, not a zero.

Published site policies point the same way for money. NERSC charges for resources made unavailable to other jobs (docs.nersc.gov), Yale bills allocated resources whether or not the job uses them (docs.ycrc.yale.edu), GWDG counts allocated GPUs for the entire runtime (docs.hpc.gwdg.de), and Harvard's fairshare formula multiplies runtime by allocated resources weighted by TRES (docs.rc.fas.harvard.edu). These weights are local policy, expressed in different units (Pitt's service units, Harvard's TRES weights normalized to a CPU core, Alliance's reference GPU units), and must not be copied as universal prices (crc-pages.pitt.edu) (docs.alliancecan.ca).

The arithmetic is simple and should be kept simple. For a constant allocation, allocated GPU-hours equal devices times elapsed seconds divided by 3,600: 4 GPUs held for 7,200 s is 8.0 GPU-hours. With the example weights `CPU=1.0,Mem=0.25G,GRES/gpu=8.0`, one 2-hour job bills 128 units per second under the default sum, 64 under `MAX_TRES`, and 96 under `MAX_TRES_GRES` (added in Slurm 25.05) (slurm.schedmd.com).

As of 9 November 2026 the stable line is Slurm 26.05, with 26.05.5 announced on 13 October 2026 (lists.schedmd.com), while 26.11.0rc1 is a release candidate not intended for production. The conclusion: allocation and billing records can support showback now; sampled activity supports quota, chargeback or procurement decisions only when device identity, coverage and sampling gaps are demonstrated.

Introduction and Background

"Slurm GPU accounting" is searched as if it were one feature, but a platform engineer, a capacity buyer and a showback analyst each want a different number from the same cluster: how many GPU-hours a project held, how many internal service units that hold costs under the site's policy, and how busy the devices were. Slurm can produce the first two from its accounting database. It can produce only a partial, sampled approximation of the third.

This is a scheduler-accounting asset: which records establish allocation, which establish billing, which establish activity, and when the evidence is too incomplete to move a quota, a charge or a purchase order. It is not a GPU-sharing comparison, a DCGM (Data Center GPU Manager) field dictionary or an argument for any utilization target. Examples are synthetic and labeled; no measurements from any real cluster are reported.

This report treats SchedMD's documentation, NVIDIA's documentation and published site policies as the evidence base, and leaves each site to supply its own rate, policy and installed plugin set.

Three framing facts anchor the rest:

- **No records without configuration:** Slurm keeps no accounting records unless an accounting storage type is configured, and `sreport` depends on a database.
- **Pin the release:** the documentation consulted here was rendered for the 26.05 line, so validate commands against the installed release with `sacct --helpformat`.
- **Showback first:** FinOps (financial operations) practice distinguishes showback from chargeback mainly by formality, meaning how formally expenses reach official accounting budgets (www.finops.org). Showback is required in any FinOps practice, while chargeback depends on organizational accounting policy (www.finops.org). This report is written for the showback decision.

The Four Questions and the Four Ledgers

What was reserved

Jobs receive no generic resources, including GPUs, unless they request them at submit time. The reservation is therefore an allocation fact. In the accounting database it appears as `AllocTRES`, which is blank for pending jobs. GPUs appear there only if `gres/gpu` is tracked in `AccountingStorageTRES`, because the default tracked

set is Billing, CPU, Energy, Memory, Node, FS/Disk, Pages and VMem (slurm.schedmd.com). Typed entries such as `gres/gpu:a100` track only jobs that request that type, so the generic type must be tracked too, otherwise a request with only the generic one is not accounted for (slurm.schedmd.com).

How long it was held

Duration comes from start and end timestamps and a raw elapsed-seconds field. Two details change the arithmetic:

- **Parent versus child:** `sacct` job records contain a primary entry for the job plus entries for steps, so summing parent and child durations double-counts.
- **Suspension:** a suspended job's generic resources do not become available to other jobs (slurm.schedmd.com), while its CPUs are released. Whether the installed billing policy treats suspended GPU time as billable is a site question, quantified with the `Suspended` field.

How the site bills it

Billing is policy. The billing TRES is calculated after allocation and visible in `sacct` output, and it feeds fairshare and any limit set on the billing TRES (slurm.schedmd.com). Discoverer+ states that Slurm bills on allocated resources (docs.discoverer.bg), and NERSC charges for the entirety of any reservation requested (docs.nersc.gov). Billing is not observed activity, and it is not currency.

What activity was observed

Activity is sampled telemetry, either Slurm's per-task summaries or an external collector. Neither is an allocation record, and neither converts into one without stated assumptions.

Table 1 below consolidates the ledgers into a field dictionary: what each field is, its scope and statistic, and what its absence means.

Field or record	Source and unit	Scope	Statistic	What missing means
AllocTRES	Accounting database; counts per TRES, for example `gres/gpu`; set after start	Job or step	Allocation at run time	Blank for pending jobs; a typed GPU is absent if the type is not tracked
Start, End, elapsed seconds	Accounting database; timestamps and seconds	Job or step	Wall duration	Window clipping needs `--truncate`
Suspended	Accounting database; duration formatted like Elapsed	Job or step	Time suspended	Zero is plausible; confirm treatment in the installed billing policy
Billing TRES	Partition `TRESBillingWeights`; integer, truncated (slurm.schedmd.com)	Per partition	Weighted sum or max	Default is allocated CPUs only
Raw usage	`sshare`; TRES-seconds charged to an account or user, decaying (slurm.schedmd.com)	Association	Decayed cumulative	Reset or purge can make it zero without meaning no use

Field or record	Source and unit	Scope	Statistic	What missing means
gres/gpuutil, gres/gpumem	`TRESUsageIn*` fields; per-task high-water marks	Step	Ave, Max, Min, Tot of per-task marks	Not collected, disabled, or MIG: unknown, not zero
sreport utilization	Rollup tables; time in hours or minutes (slurm.schedmd.com)	Cluster	Allocated, Down, Idle, Planned, Reported	Gaps in Reported time
NVML or DCGM samples	External collector; percent or fraction per sample period (docs.nvidia.com)	Device or MIG instance	Time kernels executing	Gap in samples is unknown, not zero

Table 1 should be read left to right as a chain of custody. The first three rows establish allocated GPU-hours, the next two establish the internal charge, and the last three describe activity. Only the first five rows are exact records; the rest are samples or summaries whose scope and statistic must be stated whenever a figure is quoted.

Allocation Records: From sacct to GPU-Hours

Pinning the installation

Before any query is trusted, record the Slurm release, the `SelectType` and accounting plugins, the GPU backend and the sharing mode. Slurm releases follow a six-month cycle: 24.05 was the first release on that cadence, and the support table lists November 2026 as the end-of-support point for 25.05. Run `sacct --helpformat` on the installation and keep the output with the report.

Field names also drift across third-party tutorials. A field such as `GPUFrac` is sometimes shown in examples, but it does not appear in the sacct field list on the 26.05 documentation page consulted here, so validate every field against the installed `--helpformat`.

A reproducible extraction

The following extraction is illustrative and must be validated on the installed release. It selects allocations only, clips jobs to the reporting window, and keeps raw seconds.

```
sacct --helpformat
sacct -a -X -P -T -S 2026-10-01 -E 2026-11-01 \
  --format=Cluster,JobIDRaw,Account,Partition,State,Start,End,ElapsedRaw,Suspended,AllocTRES%80
```

Other sites publish the same pattern: Pitt tells users to inspect billed TRES with `AllocTRES` (crc-pages.pitt.edu), and GWDG shows `alloctres` for past jobs (docs.hpc.gwdg.de). Each flag has a reason. `-X` restricts output to the job allocation and ignores steps, which avoids double counting but also means that, without steps, utilization statistics for the job allocation are reported as zero; those zeros must not be read as measured values. `-T` clips start and end to the window and is useful for exact run times within any given period; when it is used, derive the in-window duration from the clipped timestamps and compare with the raw

elapsed field on the installed release. `-a` shows all users only for root or when private data does not restrict it. By default `sacct` shows only jobs with Eligible time, and state filters behave in surprising ways: asking for RUNNING returns suspended jobs as well.

Deduplication checklist

- **Parent versus child:** use allocation rows (`-X`) for allocated GPU-hours, and step rows only for task summaries.
- **Requeues:** duplicate records can result from requeues, federation, resizes, or job-ID wraparound (slurm.schedmd.com), and the submit time is reset when a job is requeued. Include duplicate records deliberately and decide which attempts are billable.
- **Arrays:** key every row by cluster, array job ID, task ID and attempt, never by job ID alone.
- **Multi-cluster:** one accounting daemon can collect from several clusters, so include the cluster in every key.
- **Typed GRES:** report each GPU type or MIG profile as its own row before applying any weight.
- **Retroactive edits:** administrators can retroactively change a job's `AllocTRES` through `sacctmgr` as of 25.11, so retain a hash of exported rows and the export date.

Allocated GPU-hours

For a constant allocation, allocated GPU-hours = allocated devices times elapsed allocation seconds divided by 3,600. **Example (synthetic):** a job holding 4 GPUs for 7,200 s is 4 x 7,200 / 3,600 = **8.0 GPU-hours**. For an allocation that changes, sum device count times interval duration. **Example (synthetic):** 8 GPUs for the first 1,800 s and 4 GPUs for the next 5,400 s is (8 x 1,800 + 4 x 5,400) / 3,600 = (14,400 + 21,600) / 3,600 = **10.0 GPU-hours**. Where a resize creates separate records, each interval must be present exactly once. A GPU allocation hours report by account and month is the sum of these allocation rows, grouped by account, partition and GPU type.

Two cautions complete the method:

- **Allocation is not the final charge:** ICARUS shows the allocation-based pattern for CPUs, where raw CPU time is allocated CPUs multiplied by elapsed seconds (icarus.hpc.ntua.gr), and warns that CPU-hours alone do not establish the final charge (icarus.hpc.ntua.gr). The same logic applies to GPU-hours.
- **Units differ:** do not add whole-GPU hours to MIG slice hours without a documented weight, because the slice is a different unit.

Billing Weights, Units and Fairshare Usage

What a TRES billing weight does

Partition `TRESBillingWeights` assign a weight to each tracked TRES. The resulting usage amount is used when calculating fairshare and when enforcing the billing TRES limit on jobs. By default the billing value is the sum over TRES types of allocated amount times weight, and when the weights are configured a job is billed for each TRES used. Memory weights are applied to mebibytes unless a suffix is given: an unsuffixed `Mem=0.25`

on a job allocated 8 GiB (8,192 MiB) results in 2,048 billed units, while `Mem=0.25G` on the same job results in 2 billed units. The billing amount is stored as an integer and truncated if needed, so very small fractional rates can vanish.

SUM, MAX_TRES and MAX_TRES_GRES

Sites can instead bill the larger of the weighted per-node TRES, enabled with the `MAX_TRES` priority flag (slurm.schedmd.com). The `MAX_TRES_GRES` flag, added in Slurm 25.05, sums billable GRES such as GPUs while taking the maximum of the other node TRES. **Billing worksheet (synthetic example):** weights `CPU=1.0,Mem=0.25G,GRES/gpu=8.0`; a job with 4 GPUs, 32 CPUs and 256 GiB for 2 hours (7,200 s).

- **SUM (default):** 32 x 1.0 + 256 x 0.25 + 4 x 8.0 = 32 + 64 + 32 = **128 billing units per second**, or 921,600 billing TRES-seconds.
- **MAX_TRES:** MAX(32, 64, 32) = **64**, or 460,800 TRES-seconds.
- **MAX_TRES_GRES:** 4 x 8.0 + MAX(32, 64) = **96**, or 691,200 TRES-seconds.

The same allocated hardware yields three defensible bills, which is why the policy flag must be recorded next to every reported number. Pitt's published policy shows one site's choice: it bills on the maximum of the weighted resources (crc-pages.pitt.edu), with a service unit defined as roughly one core-hour (crc-pages.pitt.edu).

Typed GPUs and MIG profiles

If weights are defined only for a typed GPU, they are not applied to generic GPUs, so a partition should define both. For MIG the profile name is part of the type. SchedMD's MIG example tracks `gres/gpu`, a whole-GPU type and a slice type together. **Example (synthetic):** with `GRES/gpu:a100_3g.20gb=3.0`, a job holding 2 such instances for 5 hours is 2 x 5 = 10 instance-hours and 2 x 3.0 x 5 = **30.0 billing units**. CU Boulder's Alpine policy, as of November 2022, sets an acceleration factor for A100 and MI100-class GPUs (research-computing-user-tutorials.readthedocs.io). KU Leuven configures a typed per-GPU weight for A100 nodes (docs.vscentrum.be). Alliance's published scheme shows how a real site derives slice weights from the larger of compute and memory share: an H100-2g.20gb instance has its own RGU score (docs.alliancecan.ca), and the group is charged for the maximum of RGU, core and memory bundles used (docs.alliancecan.ca).

Fairshare usage is not a bill

Fairshare consumes the billing amount but is a scheduling factor, not an invoice. By default the computing resource is allocated CPU time unless billing weights are configured. The raw usage that `sshare` reports is TRES-seconds charged to an account or user, and it decays when a half-life is defined. The decay is configured by `PriorityDecayHalfLife`, which halves the original usage value at the stated age (slurm.schedmd.com), and usage clears only at `PriorityUsageResetPeriod`, whose default never clears historic usage. **Example (synthetic):** with a 7-day half-life, the 921,600 TRES-seconds above count as 460,800 after 7 days and 230,400 after 14 days. In the classic algorithm the half-life formula favors the most recent usage and effective usage is defined against the parent account; Fair Tree has been the default since Slurm 19.05. The fair-share factor orders queued jobs and is not a hard cutoff.

One trap follows: a QOS usage factor of 0 adds no timed usage from the job, so showback built on raw usage silently omits such jobs. Harvard counts failed and requeued usage in fairshare because users are expected to use allocated time effectively (docs.rc.fas.harvard.edu); that is a policy choice, not a Slurm default.

From units to money

An internal monetary charge equals policy-defined billable units times an approved, dated rate. The rate belongs to finance, not to the scheduler. Yale's published rate is described as derived to closely match the prorated cost of a similar dedicated node over its warrantied period (docs.ycrc.yale.edu), and ACCESS documentation states that resource providers can modify their exchange rates (access-ci.atlassian.net). No rate is proposed here, because none can be derived from Slurm records alone.

Observed Activity: Task Summaries, Sampling and MIG

What the Slurm GPU fields are

Adding `gres/gpu` to `AccountingStorageTRES` also sets `gres/gpumem` and `gres/gpuutil` (slurm.schedmd.com). They are collected by the job accounting gather plugin, which collects GPU data only where the backend supports it: documented for NVIDIA GPUs via the NVML autodetect backend and for AMD GPUs via the RSMI backend. SchedMD's example for two tasks on two GPUs shows `gpumem` carrying an M suffix and `gpuutil` as a bare number; the total across tasks is the sum of per-task marks, so a total of 200 for `gpuutil` is two tasks at 100, not 200 percent of anything.

The statistic matters more than the name. The Max, Min, Ave and Tot fields summarise per-task high-water marks, and the documentation describes the GPU values as high-water marks over all ranks in the step (slurm.schedmd.com). ICARUS puts the operational consequence plainly: Slurm reports GPU memory and utilization high-water marks across tasks (icarus.hpc.ntua.gr).

What a high-water mark cannot prove

A `gpuutil` average of 100 in `TRESUsageInAve` is compatible with a device that was busy for one sample and idle for the remaining hours, and with a device that was busy throughout. The figure cannot be multiplied by elapsed time to recover productive GPU time. **Example (synthetic):** two steps both report `gpuutil=100`; one ran for 60 s and one for 36,000 s on the same GPU count. Their reported values are identical, and the integral implied by multiplying them differs by a factor of 600 with no new evidence behind it. For that reason, derived "activity hours" must never come from `TRESUsageInAve` alone.

Unknown, disabled, unsupported and measured zero

Four states look the same in a spreadsheet and mean different things. Use all four in every report.

- **Missing:** the field is blank or absent for a row. Availability of metrics relies on the gather plugin in use, and the cause is undetermined until checked.
- **Unsupported:** the combination cannot measure it. MIG devices have no `gpumem` or `gpuutil` accounting because NVML does not support utilization metrics for MIGs; NVIDIA's MIG guide states that NVML does not support attribution of utilization metrics to MIG devices (docs.nvidia.com).

- **Disabled:** the site turned it off. `JobAcctGatherParams=DisableGPUAcct` skips accounting of GPU usage and any GPU driver library call (slurm.schedmd.com), and a task interval of 0 collects accounting only once, at job termination.
- **Measured zero:** the collector ran, covered the interval, and the device was idle. This is the only state that supports a claim of idleness.

ICARUS tells its users that missing fields do not prove zero usage (icarus.hpc.ntua.gr), and it also discloses that its documented commands have not been tested against its own system (icarus.hpc.ntua.gr). That honesty about test status is the standard this report applies to every command in it.

Backend and sharing support matrix

The support matrix below records, per backend and sharing mode, what SchedMD's GRES documentation says about allocation records and Slurm-side activity fields. It is not a market survey, and unsupported cells stay explicit.

- **NVIDIA whole GPU, NVML autodetect:** allocation is recorded through `gres/gpu` in `AccountingStorageTRES`; `gpumem` and `gpuutil` are documented as available, as high-water marks only.
- **AMD GPU, RSMI autodetect:** the same TRES configuration applies and the Slurm fields are documented as available; AMD's engine-activity API reports a percentage from 0 to 100 (rocm.docs.amd.com).
- **NVIDIA MIG:** typed tracking uses profile names and has been supported since 21.08; Slurm provides no `gpumem` or `gpuutil` accounting, and NVIDIA's MIG guide says NVML does not support attribution of utilization metrics to MIG devices (docs.nvidia.com). MIG devices must be pre-partitioned.
- **Sharding:** tracked only if `shard` is added to `AccountingStorageTRES`; shards do not fence processes on the GPU, so activity attribution is undefined.

Only whole-device NVIDIA or AMD with the matching backend has a documented Slurm-side activity summary; elsewhere the correct report cell is unknown.

Joining Telemetry to Jobs

What the collectors measure

NVML defines GPU utilization as the percent of time over the past sample period during which one or more kernels was executing (docs.nvidia.com), with a sample period between 1/6 s and 1 s depending on the product (docs.nvidia.com). It says nothing about how much of the device was in use during that time. DCGM offers finer fields: graphics-engine activity is the fraction of time any portion of the graphics or compute engines were active (docs.nvidia.com), and SM (streaming multiprocessor) activity is the fraction of time at least one warp was active on a multiprocessor, averaged over all multiprocessors (docs.nvidia.com). The legacy utilization field is documented in percent 0 to 100 and is marked deprecated in favor of a ratio field (docs.nvidia.com), so mixing percent and ratio fields silently scales results by 100.

Collection intervals decide what can be inferred. DCGM profiling metrics default to 1 Hz (docs.nvidia.com), while the `dcgm-exporter` collection interval option documents a 30,000 ms default (docs.nvidia.com). A 30 s sample says little about sub-second bursts, so carry the sampling period alongside every value. Profiling collection also conflicts with other NVIDIA developer tools (docs.nvidia.com), a coverage risk on shared nodes.

Stable identity: device, instance and job

A join is only as good as its keys. The exporter can read scheduler-created GPU-to-job mapping files (docs.nvidia.com), and repeated job IDs within each mapping file are deduplicated while mappings across files are preserved (docs.nvidia.com). The DCGM job-statistics workflow needs stats recording to be enabled before the target workload starts (docs.nvidia.com), and DCGM recommends running the query in epilogue scripts as part of job cleanup (docs.nvidia.com), which is the natural scheduler hook. For MIG, DCGM can provide metrics for workloads running on MIG devices (docs.nvidia.com), each MIG device gets its own UUID beginning with `MIG-` on current drivers (docs.nvidia.com), and a layout change while the exporter runs requires an exporter restart (docs.nvidia.com). Older CUDA releases let one process enumerate only a single MIG device (docs.nvidia.com).

Slurm's own side of the join has an equally important rule: `CUDA_VISIBLE_DEVICES` inside a cgroup-constrained job is relative, so index 0 in the job is not necessarily device 0 on the node. Key the join on device UUID and MIG UUID, never on index, and keep a node-level inventory table with effective dates. An open-source exporter shows a related caution: the CEA `slurm_exporter` requires GPU accounting to be enabled explicitly (github.com) and advises sites without GPUs or accounting to leave it off (github.com).

A reproducible activity integral

Where a collector provides timestamped samples with stable identity, an activity estimate can be built as a separate, labeled calculation. **Example (synthetic):** one GPU is allocated for 14,400 s (4.0 h). A collector with a 30 s period should produce 480 samples; 360 are present (75.0 percent coverage) with a mean kernel-active ratio of 0.50. Covered activity is 360 x 30 x 0.50 = 5,400 s = **1.5 h**. The 120 missing samples span 3,600 s, so activity lies between 1.5 h (missing samples all idle) and 2.5 h (missing samples all busy), that is, **37.5 to 62.5 percent of the allocation**. The honest result is a bounded range, not a point value. For shared devices, add a documented rule for splitting device activity between jobs, because a device-level ratio is not a per-job ratio.

Implementation Guidance: Reconciliation, Retention and Promotion

Reconcile three totals

Compute allocated GPU-hours from allocation rows, then compute the same period from `sreport`, then compare. `sreport` reads hourly, daily and monthly rollups, uses CPU by default unless `-T` is given (icarus.hpc.ntua.gr), accepts a comma-separated list of TRES names for most reports and reports in minutes by default. A GPU variant such as `sreport -T gres/gpu -t Hours cluster AccountUtilizationByUser Start=2026-10-01 End=2026-11-01` is illustrative; confirm the options on the installed release. Reports cannot be generated for a more specific time range than the hour.

Interpret the cluster-level rows with care. Allocated is time that nodes were in use with active jobs or an active reservation, which is an allocation notion. Idle is time that nodes were not Allocated, Down, PlannedDown or Planned. Reported is the total time for which records are available. A parent account's usage can be larger than the sum of its children, and with oversubscription and no preemption accounting can exceed actual utilization. ICARUS adds a practical note for account reports: the blank-login row is the account total, so it should not be added to per-user rows (icarus.hpc.ntua.gr).

Differences between `sacct` rows and `sreport` totals are expected from rollup timing, clipped windows and reservations; state tolerances first and record each difference with its cause.

Retention, privacy and raw records

Retention is explicit configuration. Job records are never purged unless a purge age is set (slurm.schedmd.com), units default to months, and the age is based on job end time. Usage records, the tables behind `sreport`, are purged separately, and suspend records have their own parameter, so they are retained by default but may be purged independently. Archived data is written to flat files that can later be reloaded. Retain raw exports of allocation rows and the weights file for every reporting period, because a purge or a retroactive edit changes what a later query returns.

Privacy follows from the same settings. Non-root users see only jobs launched under their own user by default, and `PrivateData=usage` prevents users from viewing the usage of any other user. Aggregate to account or project level unless the audience is authorized to see user and job detail.

Promotion gate for a showback report

A report moves from draft to showback only when every item below can be evidenced:

- **Version record:** Slurm release, plugins, GPU backend and sharing mode pinned, with `--helpformat` output retained.
- **Policy record:** partition weights, SUM or MAX flag, QOS usage factors and decay settings versioned in source control.
- **Rate owner:** a named finance owner and an approved, dated rate, or an explicit statement that the report is in service units only.
- **Raw record retention:** exports retained with row counts and hashes, outside the purge window.
- **Unknown state:** every activity cell carries one of missing, unsupported, disabled or measured zero.
- **Coverage statement:** telemetry coverage and sampling period stated for every activity figure.
- **Disagreement log:** every `sacct` versus `sreport` difference explained.

Data Analysis and Evidence

Published site policies

The only reliable way to see how much billing policy varies is to read it side by side. *Table 2* below lists published weights and rules from public sites, each labeled as that site's policy, dated by the page where a date is stated. None is a recommendation.

Site (policy owner)	What is billed	Published GPU treatment (site policy)
University of Pittsburgh CRC	Allocation; max of weighted resources (crc-pages.pitt.edu); pending jobs not charged (crc-pages.pitt.edu)	GPU weight 8 on its A100 partitions in the published weight table (crc-pages.pitt.edu)
Harvard FASRC	Runtime times allocated resources weighted by TRES; weights normalized to a Cascade Lake core of 1.0 (docs.rc.fas.harvard.edu)	A100 weight 836.5 (docs.rc.fas.harvard.edu); weights derived from theoretical peak performance normalized to FP64 (www.rc.fas.harvard.edu)
Yale YCRC (FY27)	Allocated resources billed whether or not used (docs.ycrc.yale.edu); actual runtime, not requested walltime (docs.ycrc.yale.edu)	A100 80 SU per GPU-hour, H100 150, H200 240, B200 300 (docs.ycrc.yale.edu)
KU Leuven (VSC)	Weights times walltime; cost tool assumes a worst case (docs.vscentrum.be)	Typed A100 weight 141.6666667 (docs.vscentrum.be)
NERSC	Resources made unavailable to other jobs (docs.nersc.gov); 100 nodes allocated and one used still charges 100 (docs.nersc.gov)	Perlmutter GPU charge factor 1 per node-hour; future systems may differ (docs.nersc.gov)
Alliance (Canada)	Maximum of RGU, core and memory bundles (docs.alliancecan.ca)	A100-40gb reference at 4.0 RGU (docs.alliancecan.ca); all GPUs treated as equivalent before RAC 2023 (docs.alliancecan.ca)
Discoverer+ (Bulgaria)	Slurm bills on allocated resources (docs.discoverer.bg)	GPU billing counter described as a fairness mechanism for host CPU and memory (docs.discoverer.bg)
GWDG (Germany)	Only allocated GPUs counted on GPU partitions (docs.hpc.gwdg.de); allocated GPUs count for the entire runtime (docs.hpc.gwdg.de)	150 core-hours per GPU on a shared partition in its worked example (docs.hpc.gwdg.de)
JHU Rockfish	GPUs times billed cores per GPU times walltime (docs.arch.jhu.edu)	2 A100 for 3 hours is 72 core-hours (docs.arch.jhu.edu); inspect with `AllocTRES` (docs.arch.jhu.edu)
TACC Vista	Node-based; charges resources used, not those requested (docs.tacc.utexas.edu)	1 SU per node-hour in the published queue table (docs.tacc.utexas.edu)

Table 2 supports three observations. First, most rows charge on allocation, and TACC states that its node-based system charges for resources used rather than requested, which for whole nodes still means the nodes the job holds while running. Second, GPU weights use incompatible units (SUs, core-hours, credits, RGUs, node-hours), so a "per GPU-hour" figure copied between sites is meaningless without its unit. Third, the sites document their rules in prose and tables; none of them equates a billing weight with observed activity. NCSA's Delta documentation bases charges on the resources reserved for the job (docs.ncsa.illinois.edu), and the San Diego Supercomputer Center's Expanse guide on the resources tied up by it (sdsc.edu). Harvard counts failed and requeued usage in fairshare because users are expected to use allocated time effectively (docs.rc.fas.harvard.edu), a policy choice rather than a Slurm default.

Why allocation, not activity, is the billing basis

The FinOps framework treats allocation of shared cost as a modeled split (www.finops.org), and the FOCUS (FinOps Open Cost and Usage Specification) v1.4 text supports allocating costs where resource usage might not match the units the resource is measured in (focus.finops.org). Showback is an allocation strategy producing reporting on consumption of IT services (www.finops.org).

Acceptance Cases (Hypothetical Examples)

The cases below are synthetic, and *Table 3* below summarizes them. Each was designed to exercise one failure mode of the reconciliation, using billing policy P: `GRES/gpu=8.0` only, other weights 0, default SUM, so the GPU billing component equals allocated GPU-hours times 8.0. None reflects a measured cluster.

Case (Hypothetical Example)	Allocation record	Allocated GPU-hours	GPU billing units (policy P)	Observed activity	Reconciliation status
1. Idle allocation	1 GPU, 7,200 s	2.0	16.0	Full coverage, all samples 0: **measured zero**	Reconciled; charge stands because the device was held
2. Active short job	1 GPU, 300 s	0.083	0.667	10 samples at 30 s, mean 0.90: 270 s	Reconciled; short window, activity within allocation
3. Multi-step job	4 GPUs, 3,600 s; step.0 uses 4 GPUs 3,000 s; step.1 uses 2 GPUs 500 s	4.0	32.0	Step summaries are high-water marks: **no integral**	Use allocation row; summing job and steps gives 7.61 GPU-hours, a double count
4. Failed then requeued	Attempt 1: 2 GPUs, 1,800 s, node failure; attempt 2: 2 GPUs, 3,600 s, completed	3.0 (attempt 1: 1.0; attempt 2: 2.0)	24.0	Not applicable	Default view shows the latest record only; **billing of attempt 1 is a policy decision**
5. Typed MIG	2 x `a100_3g.20gb`, 5 h	10.0 instance-hours (not whole-GPU hours)	30.0 at example weight 3.0	`gpuutil` **unsupported** for MIG	Allocation reconciled; activity unknown
6. Unknown telemetry	1 GPU, 14,400 s	4.0	32.0	75.0 percent coverage: **1.5 to 2.5 h**	Bounded range only; **insufficient for a procurement claim**

Table 3 shows how each decision depends on a different record. Cases 1 and 2 reconcile because the collector covered the interval. Case 3 turns on the instruction to use allocation rows: 4 x 3,600 + 4 x 3,000 + 2 x 500 = 27,400 GPU-seconds, or 7.61 GPU-hours, against the true allocation of 14,400 GPU-seconds. Case 4 depends on the duplicate-record rules, since default views show only the most recent record when several match and the original submit time needs the duplicates option. Cases 5 and 6 are where the explicit unknown state earns its place: one is unsupported and the other only bounded.

Implications and Future Directions

For capacity buyers, the evidence thresholds split in two. Allocated GPU-hours and billing units, reconciled between `sacct` and `sreport`, support showback and internal rate cards. They do not support a claim that GPUs were idle or saturated. A purchase argument that rests on activity needs sampled telemetry with demonstrated coverage, stable device identity and an attribution rule for shared devices; without these the defensible statement is the bounded range from the worked example, or "unknown".

For quota design, fairshare is the wrong place to encode a belief about productive use. It decays usage with a half-life and orders jobs rather than charging for them. A weight revision changes future accrual but not records already written, so quota changes should carry the policy version and effective date.

Release timing matters. As of 9 November 2026 the stable line is 26.05, and the 26.11 release candidate is explicitly not for production (lists.schedmd.com). The release notes show active movement here: `MAX_TRES_GRES` changed how GPUs enter a maximum-style bill. Each upgrade should re-run field-list and billing-flag validation. Rates remain an organizational decision, derived from node cost over warranty (docs.ycrc.yale.edu) or from relative performance (www.rc.fas.harvard.edu); a rate card without an owner and a date is not a rate card.

Frequently Asked Questions (FAQs)

How are GPU hours obtained from sacct?

Pin the fields with `--helpformat`, select allocations with `-X`, clip with `-T` for monthly windows, and multiply the `gres/gpu` count in `AllocTRES` by elapsed seconds divided by 3,600. Handle requeues and resizes with the duplicates option, and handle MIG profiles per type.

How do tresbillingweights apply to GPUs?

Weights are per partition and default to allocated CPUs only. A GPU contributes only when a `GRES/gpu` weight is set, and typed GPUs need their own weights.

What does sreport show for GPU utilization?

Time-based cluster states such as Allocated, Idle and Reported. That is allocation time, not device activity.

Which AccountingStorageTRES entries do GPUs need?

At minimum `gres/gpu`, which also sets `gres/gpumem` and `gres/gpuutil`. Add typed entries alongside the untyped one, and `shard` if shards must be tracked.

Can GPU hours drive chargeback, and can DCGM replace Slurm accounting?

Chargeback is units times an approved rate, and FinOps practice treats it as dependent on organizational accounting policy (www.finops.org). DCGM samples devices while Slurm records allocations; the join needs job mapping (docs.nvidia.com), and a missing sample is unknown, not idle.

Conclusion

Slurm GPU accounting is trustworthy where it records decisions the scheduler made: which devices a job held, for how long, and under which partition weights. It is weakest where decision makers most want certainty: how productively the devices were used. The records that establish allocated GPU-hours and internal service units are `AllocTRES`, the job timestamps and the partition billing weights, reconciled against `sreport` and retained as raw exports. The records that describe activity are task-level high-water marks and external samples, and neither supports an integral of productive time without stated coverage and attribution assumptions.

The practical standard is a four-ledger report with an explicit unknown state. Allocation, duration, billing and activity are reported separately, each with its source, scope and statistic. A cell that is missing, unsupported or disabled is shown as such, and a measured zero is claimed only with demonstrated coverage. A report that meets the promotion gate can serve showback. A procurement or quota change that depends on activity should wait until the coverage statement exists.

Source links

Links cited in this article, in order of first appearance. Full addresses are provided for readers using extracted text.

1. https://slurm.schedmd.com/sacct.html
2. https://slurm.schedmd.com/tres.html
3. https://slurm.schedmd.com/gres.html
4. https://icarus.hpc.ntua.gr/docs/job-accounting
5. https://docs.nersc.gov/jobs/policy
6. https://docs.ycrc.yale.edu/clusters-at-yale/job-scheduling/priority-tier
7. https://docs.hpc.gwdg.de/start_here/accounting/index.html
8. https://docs.rc.fas.harvard.edu/kb/fairshare
9. https://crc-pages.pitt.edu/user-manual/slurm/service-units
10. https://docs.alliancecan.ca/wiki/Allocations_and_compute_scheduling
11. https://slurm.schedmd.com/slurm.conf.html
12. https://lists.schedmd.com/mailman3/hyperkitty/list/slurm-announce@lists.schedmd.com/2026/10
13. https://www.finops.org/framework/capabilities/invoicing-chargeback
14. https://docs.discoverer.bg/slurm-gpu-billing-explainer.html
15. https://docs.nersc.gov/policies/resource-usage
16. https://slurm.schedmd.com/sshare.html
17. https://slurm.schedmd.com/sreport.html
18. https://docs.nvidia.com/deploy/nvml-api/api/structnvmlUtilization__t.html
19. https://slurm.schedmd.com/priority_multifactor.html
20. https://research-computing-user-tutorials.readthedocs.io/en/stable/clusters/alpine/condo-fairshare-and-resource-access.html
21. https://docs.vscentrum.be/leuven/credits.html
22. https://access-ci.atlassian.net/wiki/spaces/ACCESSdocumentation/pages/467111963/Use+credits

23. https://docs.nvidia.com/datacenter/tesla/mig-user-guide/latest/getting-started-with-mig.html
24. https://rocm.docs.amd.com/projects/amdsmi/en/docs-6.4.1/doxygen/docBin/html/group__tagGPUMonitor.html
25. https://docs.nvidia.com/datacenter/dcgm/latest/user-guide/feature-overview.html
26. https://docs.nvidia.com/datacenter/dcgm/latest/reference/field-identifiers.html
27. https://docs.nvidia.com/datacenter/dcgm/latest/reference/command-line-reference/dcgm-exporter.html
28. https://docs.nvidia.com/datacenter/dcgm/latest/release-notes/dcgm-exporter.html
29. https://docs.nvidia.com/datacenter/tesla/mig-user-guide/latest/mig-device-names.html
30. https://github.com/cea-hpc/slurm_exporter
31. https://slurm.schedmd.com/slurmdbd.conf.html
32. https://www.rc.fas.harvard.edu/blog/updated-tres-weighting-scheme
33. https://docs.arch.jhu.edu/en/latest/1_Clusters/Rockfish/3_Slurm/Usage.html
34. https://docs.tacc.utexas.edu/hpc/vista
35. https://docs.ncsa.illinois.edu/systems/delta/en/latest/user_guide/job_accounting.html
36. https://sdsc.edu/support/user_guides/expanse
37. https://www.finops.org/framework/capabilities/allocation
38. https://focus.finops.org/docs/specification/v1-4/attributes/data-generator-calculated-split-cost-allocation-handling
39. https://www.finops.org/resource/terminology
40. https://lists.schedmd.com/mailman3/hyperkitty/list/slurm-announce@lists.schedmd.com/2026/11

THE PUBLISHER

About GPU Smith

GPU Smith provides independent engineering and research for private AI infrastructure. We help technical buyers and operators reason about GPU workload sizing, hardware selection, procurement, deployment and inference operations, from the compute system to the networking, power and cooling requirements around it.

Start with the workload

Useful infrastructure decisions begin with the models, data, throughput, latency and operating constraints a team actually has. GPU Smith helps connect those requirements with system design and deployment choices. The goal is a privately controlled AI environment whose capacity and operational demands are understood before a purchasing decision.

Hardware and supplier research

Our public reference library covers GPUs, complete systems and networking components. The server-vendor directory and manufacturing research help teams investigate suppliers, compare options and follow sources behind technical claims. We distinguish manufacturer specifications from measured benchmarks and advertised capabilities from tested configurations. Reference pages are research resources, not a live inventory listing or a binding equipment quote.

Deployment and operations

GPU Smith's engineering scope includes infrastructure integration, networking, power, cooling and the practical operation of inference workloads. Our published articles explain the tradeoffs behind deployment, procurement and ongoing operations so teams can ask better questions and document decisions.

Work with GPU Smith

Explore the hardware library, GPU references, networking references and vendor research. Contact GPU Smith with your workloads and deployment constraints to discuss a project and confirm current scope, pricing and availability.

Inclusion of a manufacturer or product in our research does not imply a partnership, certification or endorsement.

Website: https://gpusmith.com

This article is educational. Verify consequential medical, legal, financial, regulatory and technical decisions with appropriate professionals and the cited primary sources. Company services are described separately from the article's research findings.