SHAREing Performance and Utilisation Monitoring of Accelerated Compute

Report from the workshop held 2026-02-25

Ed Bennett, March 2026

Introduction

Accelerated compute provides a colossal amount of computing power to researchers, available at the touch of a button. The cost of providing this is not insubstantial: both financial, in terms of the associated embedded and ongoing carbon emissions, and in the human time needed to procure, install, and maintain the facilities. However, software does not automatically use all of the power available on a system—it is easy for users to make simple mistakes that result in only a single-digit percentage of the available capacity of a node being used, and even well-written and carefully-executed software can have significant room for efficiency gains in utilising latest-generation hardware.

As such it is extremely valuable to have data on how efficiently a machine's resources are being utilised, that tie back to the individual jobs, users, and projects on the facility. These can be directly exposed to users or their project leaders, and made visible to system administrators. The latter case can help understand if there are systemic problems with particular hardware units or the overall system configuration, as well as allowing support staff to engage low-efficiency users in discussions around how the resources may be more effectively used.

Elementary tools for this for CPU-based machines have long been build into schedulers such as Slurm. However, for accelerated compute making use of GPUs or similar accelerators, SHAREing WP2 identified that there was not only a lack of training material on how to deliver this kind of monitoring, but also not a clear consensus on what techniques or tools were available or could be deployed.

As a result, a workshop was held at Swansea University and online, on 2026-02-25, bringing together research technical professionals from HPC centres, and vendors of hardware and software tooling that supports this space, to identify current practice and future opportunities. What follows is a summary of the discussions at this workshop.

I'm grateful to all participants for their contributions to the discussions, and to those who took the time to review the draft of this document prior to its wider distribution. All mistakes in it remain my responsibility.

Current Practice

A significant number of admins present had no continuous performance/utilisation monitoring in place. The majority of these provided some tooling to users to monitor their own utilisation, such as using the command-line tool `seff`[1] to interrogate the efficiency of individual jobs, using the various `–smi` tools (NVIDIA, ROCm, and Intel) to check accelerator utilisation[2], and using lightweight profilers such as LIQWID and MAQAO In many cases, admins also perform *ad-hoc* spot checks on individual nodes or jobs, using similar tools.

Another substantial group of admins present used a set of home-grown scripts to collate and present information. A disadvantage discussed of this is that, lacking the development team and testing behind a more widely deployed product, such tools can be prone to unexpected failure modes—for example, a shell script invoking `ls` expecting results in a particular order, and creating an infinite loop when it received a different one.

A point of discussion was the need for delicacy when presenting information to users. Some products put ratings against job efficiency using ☺ or ☹ symbols; other groups had chosen not to do this (or to automate the process of contacting low-efficiency users), as they had not been able to eliminate the false positives (where low efficiency on one efficiency metric is due to being constrained by a different characteristic of the node). It was widely agreed that where there are concerns around job efficiency, discussing directly with the user is the most productive course of action, but that contact needs to be phrased as an offer of assistance rather than an admonishment of poor performance. Framing discussions as “how to reduce your queue time” or “how to get more done with less time budget” can be good starting points—researchers are largely not incentivised to care about efficiency, and prefer to get their computation done correctly at all costs.

Tools explicitly aiming to address performance monitoring that were in use by those present included Open XDMoD (with the Job Performance module), OKA Core (a proprietary product), Netdata, CEEMS, and ClusterCockpit. Of these, CEEMs and ClusterCockpit will be discussed further below. For both some of the above tools and home-grown script solutions, Grafana is used for presenting data to users of the tool (both administrators and system users), and Prometheus is used as a time-series database and/or for much of the systems monitoring.

It was explicitly noted that performance counters of the kind that may be tracked for CPU jobs are more challenging to obtain for GPUs. In some cases, centres resort to proxies such as GPU power draw instead. In particular, performance counters are harder to track for containerised or virtualised workloads. It was highlighted that using up-to-date tools is essential; for

[1] In fact, multiple `seff` tools exist, but all have the same aim of presenting efficiency information for a given job.
[2] It was highlighted in multiple discussions that `nvidia–smi` can give misleading results—utilisation of only a single streaming multiprocessor on a GPU can cause the tool to report 100% “GPU utilisation”; instead, it is “SM utilisation” that should be considered.

example, earlier versions of NVIDIA's Datacentre GPU Monitor (DCGM) did not tie back to Slurm account logs effectively; this is now done.

Ambitions

A key requirement of any tooling in this space is providing dashboards. Minimally, there should be control of visibility of dashboards based on role—some dashboards are only of interest to the systems team, while others are invaluable to PIs or individual users. Where data are presented visually, they should also be easily available in an accessible format for, for example, users of screen readers, and access to raw underlying metrics should be simple enough for additional tooling to be built on top of the solution. Specific dashboard functionality sought included having quick filters, for example, for users requesting significantly more resources than their jobs were able to make use of.

Beyond this, automated reporting was also considered as desirable by many. This includes alerts to users whose jobs are of low efficiency, and to administrators where users may need to be contacted directly, or where particular nodes or devices are underperforming. Ideally, user-facing reports would also give suggestions for improvement, particularly where the underperformance is due to a single option in a submission script. They may also identify where users could switch to using different resources for a better service (e.g. from tier 2 to a national service, or vice versa).

Tooling should monitor a wide range of metrics. For compute, this should include CPU and GPU utilisation, and as many performance counters as practical. Beyond compute, metrics for the network fabric (Infiniband, Omnipath) and I/O (both local and GPFS/Spectrum Scale, Lustre) would be valued; there exists commercial tooling for some of these, but having an open-source solution, and one that integrates with the monitoring solution rather than needing to be run explicitly, would be valuable. Beyond this, other target metrics included power utilisation, the effect of jobs on infrastructure (e.g. if one user's I/O were having a detrimental effect across the entire cluster), and application-specific metrics (by providing hooks for applications to report to).

Almost all those present worked with Slurm clusters; a smaller but significant number additionally worked with Kubernetes, particularly for AI workloads. A tool that worked with both would be better than one that worked with neither.

Being able to provide increased visibility of metrics to users is valuable. Integrating dashboards with existing user portals may be more valuable than providing a separate dashboard page that users need to explicitly navigate to. A traffic light system can give an at-a-glance summary to users who may not immediately understand the significance of a line graph of percentage efficiency.

Similarly, having proactive hooks within Slurm to reject jobs with an inefficient configuration before they start could drive engagement more than printing a message in the job epilogue. However, such hooks have been found to be difficult to make fully robust against edge case, with each false positive introducing additional manual workload for the systems team to ensure that valid jobs are not rejected.

Where users are disinclined to engage with efficiency discussions, a mandate from senior leadership to increase efficiency as part of a green certification strategy can be another motivator. It may also be possible for tooling to automatically manage the energy efficiency of a workload; for example, by downclocking the CPU for workloads where it is known to not affect the throughput. Another option would be systematically allocating "bonus" resources where an allocation was used particularly efficiently.

There may be scope for machine learning-based tools to reduce the human time required for some monitoring. Possible examples are triaging user jobs to select those of low efficiency that may particularly benefit from intervention; linking monitoring to debugging; and translating automated metrics-heavy alerts into a format that users can easily understand.

Tools

A number of tools were presented by their developers at the workshop. Slides of the full talks[3] are available at the workshop's Indico page; presentations are very briefly summarised here. Other tools that were mentioned by their users in discussion groups are not presented in detail, but may be alternatives for some of the below.

ClusterCockpit (https://clustercockpit.org)

ClusterCockpit is an all-in-one performance and utilisation monitoring solution, developed at the Friedrich-Alexander-Universität Erlangen-Nürnberg. It was presented by Jan Eitzinger, one of its developers. It is written in Go, and released under the MIT license.

ClusterCockpit comprises: a node agent that runs on each cluster node; a Slurm adapter that integrates data from the scheduler; a SQLite database for current data; a Job Archive for historic data (with multiple formats available); and a web UI component. These are tied together by a single backend component. It prioritises providing a first-class web UI, with user roles for users, project managers, support personnel, and administrators.

It provides both job- and node-focused views, via a purpose-built dashboard interface, rather than relying on Grafana. Metrics are sampled, so can be viewed as a time series over the life of a job in addition to per-job averages; this sampling is lightweight enough not to measurably

[3] With the exception of NVIDIA, for which the slides could not be shared and reference material was provided instead, and Linaro, who needed to redact one slide.

affect the performance of the HPL benchmark. It attempts to tag jobs by workload type by inspecting executable names and modules; this however sometimes gives false results.

It has experimental support for automatic energy optimisation, adjusting node settings to balance throughput and energy-to-solution; however, this does not yet support GPU jobs, and is limited to node-exclusive workloads.

CEEMS (https://ceems-dev.github.io/ceems/)

The Compute Energy & Emissions Monitoring Stack (CEEMS) started as a proof-of-concept internal project at CNRS in France; is has now grown to a widely-deployable set of tools with a scope similar to ClusterCockpit. It was presented by Mahendra Paipuri, its lead developer. It is written in C and Go, and is released under the GPLv3.

Unlike ClusterCockpit, CEEMS aims to leverage off-the-shelf open-source projects where available, to minimise the maintenance burden and maximise the benefits from the wider community. In particular, it uses Prometheus for its time series database and Grafana for visualisation, dashboards, and access control; a CLI client is also available. Performance monitoring is based on `cgroups`, the `perf` subsystem, and eBPF. Supported metrics include energy, compute performance on CPU and NVIDIA and AMD GPUs, and I/O (measured in a filesystem-agnostic way).

Similarly to ClusterCockpit, a low-overhead exporter runs on each node; the overhead is less than 0.5% in CPU requirement, and less than 100MiB in memory footprint. In addition to recording metrics in Prometheus, CEEMS can also perform continuous profiling, exporting to Grafana Pyroscope.

ClusterCockpit and CEEMS have comparable feature sets; the main difference is the approach to maintainability.

Linaro Forge (https://www.linaroforge.com)

Linaro Forge is a proprietary set of debugging and profiling tools for high-performance computing. It was presented by Rudy Shand of Linaro. Licensing is priced based on the number of MPI ranks on which the tool will operate.

The primary audience of Linaro Forge is application developers and users. There are two profiling modes available: Performance Reports, which gives a very high-level view of application performance, and MAP, which is a sampling profiler designed for hotspot analysis. The latter has an adaptive sampling rate; it will discard samples to achieve around 1,000 samples per process, avoiding very large sample files for longer-running applications. It is cross-platform, able to run on x86 and Arm, and with NVIDIA, AMD, or Intel GPUs.

Additionally, MAP has a thread affinity advisor, which will provide thread binding directives optimised for an application's memory access patterns.

Unlike ClusterCockpit and CEEMS, Performance Reports and MAP must be invoked explicitly by the user as part of a job when they wish to understand that job's performance; the tools can't be used to understand jobs after the fact in which they were not run. Extending Linaro's suite to cover the more passive performance monitoring space is on Linaro's agenda; however, their first priority is developer-focused tooling, and there is no commitment to a delivery date for the proposed performance monitoring features.

NVIDIA Run:ai

Run:ai is a set of tooling acquired by NVIDIA in 2024, of which some elements were open-sourced in 2025. While currently its implementation focuses on AI workloads deployed via Kubernetes, it is hoped that NVIDIA's acquisition of SchedMD will allow them to extend it to the HPC space. Run:ai was presented by Lee Davis, from NVIDIA's Run:ai team.

The primary problem Run:ai seeks to solve is making effective use of distributed, heterogenous resources. The use case is in organisations that have silos of compute capability, many of which are undersubscribed. When users submit workloads to a resource that is oversubscribed or underpowered, Run:ai may migrate it to run on a larger, less-occupied resource that would otherwise be going to waste, and thus get results more quickly without more investment of resource. (Should the undersubscribed resource become needed by its primary uses, Run:ai will migrate the guest workload back to its original target, and the performance will decrease to that originally expected by the developer.) Clusters can be on-premises, in the cloud, even air gapped.

Where workloads can only utillise a fraction of a GPU's capability, Run:ai is able to dynamically partition it between multiple jobs, allowing more jobs to run concurrently than would be possible for exclusive allocations.

As part of this, it includes tools to give visibility of the utilisation of such resources. This is expanded by Mission Control, which also brings in power consumption data.

Since the current implementation is tightly wed to Kubernetes and the presentation to AI terminology, the utility for HPC is currently limited; however, should NVIDIA follow through on extending the tooling to HPC, then it could lower the barrier to entry and enable more effective utilisation of heterogenous clusters by diverse job mixes.

AMD Omnistat

Omnistat is a scale-out cluster telemetry tool targeting AMD GPUs. It was presented by Jordà Polo, one of its developers. It is written primarily in Python, and is MIT licensed.

Omnistat collects data from the system management interface, including GPU and HBM utilisation; GPU power, clock speed, and thermals; and RAS error counts and throttling events.

It can also sample application-specific figures of merit and network traffic. The sampler is low overhead, with no measurable decrease in performance with a 10-second sampling interval, rising to only 1% with a 0.01s interval.

The tool may be run in two modes: system mode, aimed at characterising jobs across an entire cluster, and user mode, for individual users' jobs. The latter requires no setup by the administrator, running entirely within a user's permissions. For the former, similarly to CEEMS, Omnistat uses Prometheus as its time series database, and Grafana for dashboards.

In addition to collecting raw hardware counters, which can be difficult to interpret, Omnistat provides high-level aggregate metrics from them. For multi-GPU jobs, where there are limitations on the number of concurrent hardware counters that may be monitored, Omnistat can multiplex across GPUs to track a larger number of counters than one GPU allows.

While currently focused on AMD products, Jordà indicated that the team would not be averse to pull requests adding support for competitors' products.

Next steps

Following the above presentations, participants discussed their takeaways and next steps, including both what tools appeared most promising, and what barriers there may be to deployment.

Both ClusterCockpit and CEEMS will out of the box meet the goals of the event, and cover many if not all of the wish list developed in the initial discussions. For clusters exclusively using AMD accelerators, Omnistat would also be a viable choice. Some participants plan to investigate replacing their current in-house tooling with one of these three; many participants are excited at the prospect of being able to get better data to their users.

Linaro Forge and Run:ai both have interesting and useful capabilities, but in their current forms are less aligned with the needs discussed at this event.

There were some perceived barriers to deployment of these solutions. As is frequently the response to questions around barriers, the first point discussed was a lack of time: even with the benefit of finding out about the tooling at this event, actually attempting the deployment requires a larger time investment. It is hoped that the hackathon that the ACIT hub is running later this year will help cluster administrators set aside a little time to get a prototype deployment, that can then be extended to a full cluster.

For participants who don't themselves administer clusters, a related challenge is convincing their sysadmins that it is worth setting up this tooling. User-space tools such as Linaro Forge and Omnistat in user mode may be good stop-gaps in that case. The availability of more detailed training commissioned by SHAREing may also help in raising awareness of the importance of this class of tooling, and of how to deploy it easily.

Some organisations have zealous but underresourced cybersecurity teams who seek to audit all software being deployed on digital infrastructure. A review from a trusted central organisation such as JISC or NCSC may short circuit that requirement, reducing the number of steps in the workflow before deployment can occur.

There is also an open question as to what capabilities users will find useful, in defining what dashboards to make available. This is likely to vary on a cluster-by-cluster basis, as each has its own unique user community. It may be beneficial to build up a repository of dashboard templates and recommendations for what users do and do not find useful.

Conclusions

The majority of participants left the event significantly more informed about the availability of tools for performance and utilisation monitoring of accelerated compute than they arrived. Continuing to develop experience and subsequently training material on best practices of deploying these tools will significantly improve the monitoring for accelerated compute in the UK, and allow for more effective utilisation of these resources, increasing the impact and value for money of the resources.