Dynamic resource sharing and diverse compute platforms

Supporting AI innovation in modern HPC

William Tripp, Scott Davidson, Stig Telfer (StackHPC)

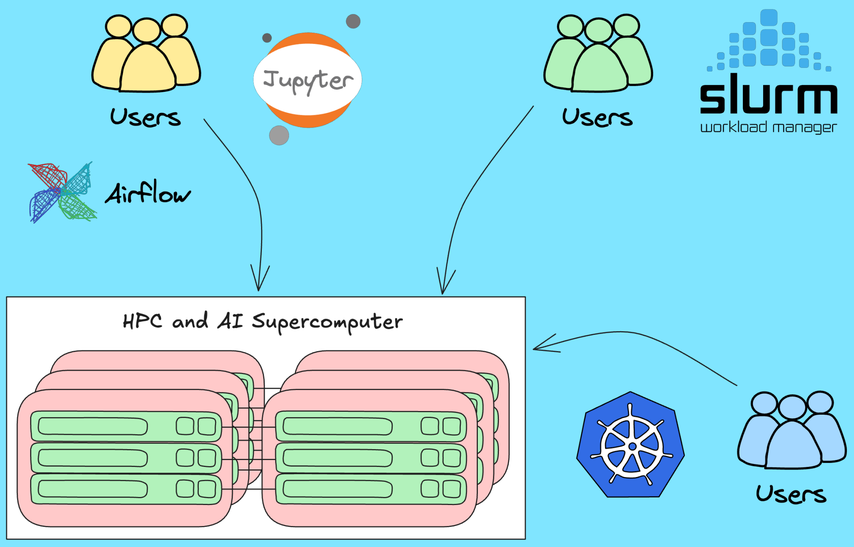
StackHPC

Abstract

Developments in recent years have demonstrated the need for flexibility in the provisioning of High-Performance Computing (HPC) services; capabilities in their infancy two years ago are now critical considerations for future procurements. The revolutionary effect on software workflows cannot always be accommodated by conventional HPC systems. Finite budgets and the high cost of current AI-capable infrastructure reinforce the need to maximise both efficiency and utilisation of high-value resources. Innovation requires a rethink of the status quo.

An optimal solution must provide compute platform agility by combining familiar HPC batch-scheduling interfaces, such as Slurm, with both interactive data analysis workflows, such as Jupyter notebooks, and emerging cloud-native research computing tools, such as MLFlow - all couched in a common shared infrastructure. At StackHPC we have been prototyping solutions based on combining Slurm and Kubernetes to provide a path to efficient and dynamic resource sharing in modern supercomputing facilities.

The Case for Resource Sharing in HPC and AI Research



A Case Study: As part of the AIRRFED project on the Dawn AI supercomputer at Cambridge University, the team at StackHPC have been developing infrastructure tooling to support federated, multi-tenant access to powerful GPUs and other AI resources within a shared HPC environment. Modern research use cases are highly diverse; ranging from single-user interactive Jupyter notebooks to production-ready Kubernetes clusters deployed within Trusted Research Environments (TREs).

To meet these demands, compute resources must be partitioned across multiple backends: isolated Kubernetes clusters for TREs, a Slurm cluster for traditional HPC workloads, and one or more scalable Kubernetes clusters for self-serviced interactive applications. This model offers flexibility but introduces challenges around utilisation. Reserved Kubernetes nodes may sit idle while Slurm users face long queue times.

To address this imbalance, we have been exploring the possibilities of dynamically reclaiming idle resources while preserving interactivity, security and performance of applications for self-service users.

One of the key challenges in addressing this resource utilisation problem arises from the fact that Kubernetes and Slurm operate under fundamentally different scheduling models, making it difficult to seamlessly share and reallocate nodes between them.

Bridging this gap requires new approaches to dynamic backfilling and pre-emption that respect the performance and isolation guarantees of both environments.

Recent developments in the open-source software ecosystem have made encouraging progress toward bridging this gap.

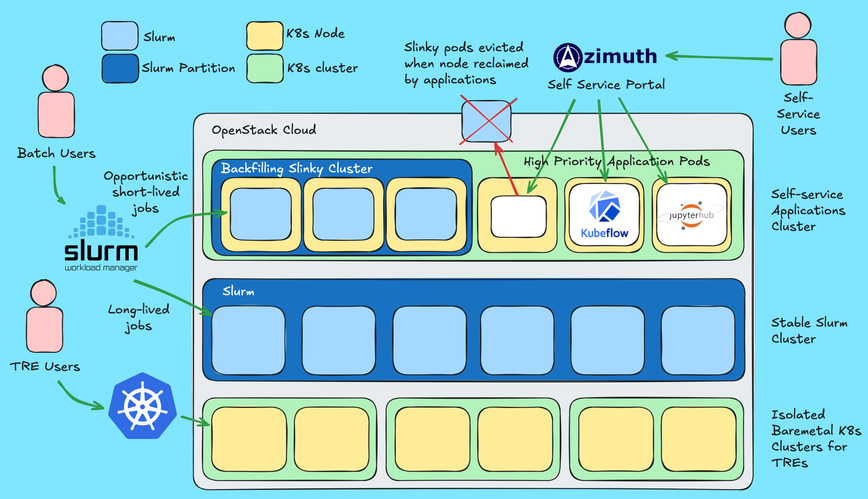
A path forward: Dynamic resource sharing using Slurm and Kubernetes

What is Slurm?

A batch computing queueing system, providing:

- Advanced scheduling
- Configurable fair-sharing policies
- Per-job resource limits
- Broad support for high-performance networking and compute hardware for specialised workloads (e.g. MPI jobs, NUMA awareness, etc.)

Slurm excels at running finite-time workloads requiring high-performance and which rely on highly-optimised software stacks built for large-scale supercomputers.



A maximally-utilised supercomputer, running a mix of cloud-native and HPC workloads with dynamic resource usage and workload prioritisation.

What is Kubernetes?

A cloud-native workload orchestrator, providing:

- Highly-available dynamic workload scheduling
- Reproducibility via containerisation
- Native cloud computing integrations (auto-healing, cluster auto-scaling, dynamic storage provisioning and more)

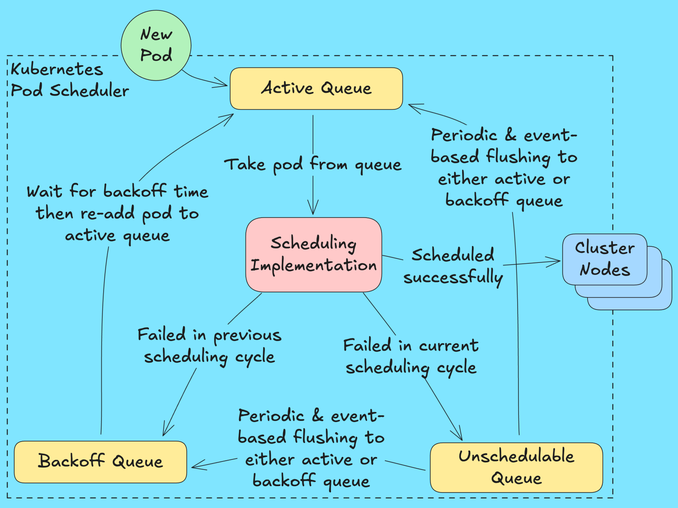
Kubernetes excels at orchestrating long-lived workloads (e.g. AI inference services) and catering for dynamic research tasks including interactive data analysis, event-based ETL pipelines or distributed ML model training.

The best of both worlds?

The team at StackHPC have recently been exploring a proof-of-concept solution for dynamic resource sharing which leverages SchedMD's Slinky project to provide containerised, auto-scaling Slurm clusters within Kubernetes. The goal of the initial phase was to demonstrate how idle compute resources could be backfilled into a shared Slurm pool and then automatically returned to interactive workloads as needed using Kubernetes pod pre-emption functionality.

A scheduling clash

The Kubernetes scheduler is optimised for efficient scheduling in large, churn heavy-clusters rather than strict priority-based ordering and pre-emption.



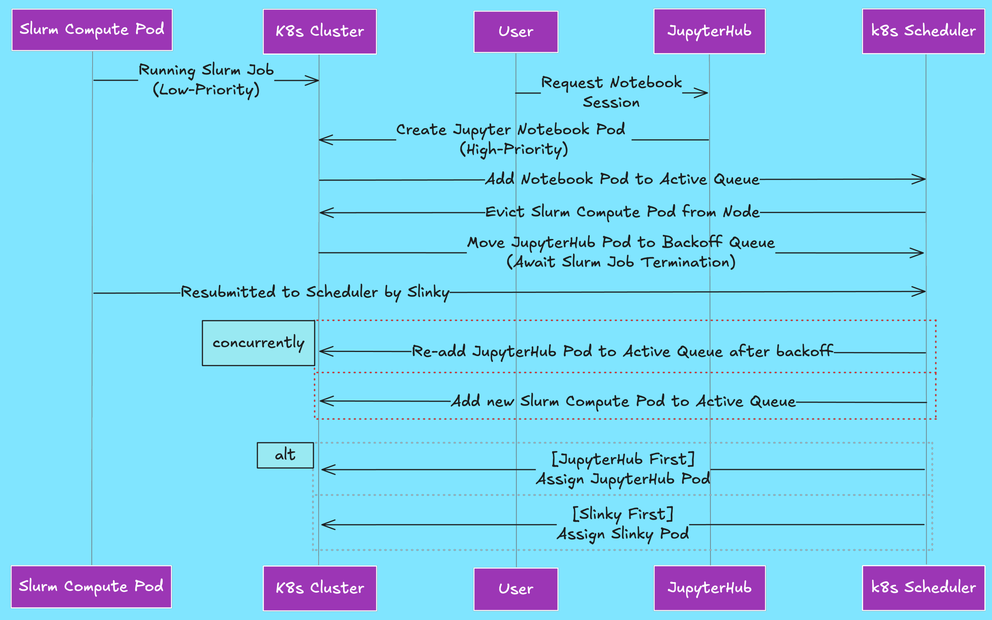
The default Kubernetes scheduler implementation.

The scheduler leverages three internal pod queues (left), with various processes which move pending workload pods between queues.

This design can lead to race conditions during scheduling which make pod-priority-based scheduling ineffective.

For dynamic resource sharing between batch and interactive workloads, the scheduler fails to respect priority ordering of Kubernetes pods. This leads to infinite scheduling loops and other undesirable scheduling issues (right).

As a result, simple pod-based pre-emption is not enough for dynamic resource sharing in HPC and AI supercomputers.



What's Next?

On-going developments in the open-source Slinky project suggest that it remains a promising pathway toward more efficient, fair and adaptive HPC & AI infrastructure. Furthermore, SchedMD's recently released Slurm Bridge project, which offers the exciting prospect of using the native Slurm scheduler as a drop-in replacement for the default Kubernetes scheduler, also advertises built-in support for mixed Kubernetes and Slurm environments. This, alongside the ability to use Slurm's own priority and pre-emption mechanisms, would avoid the inherent limitations of the default Kubernetes scheduler by placing all workloads in a single Slurm queue. The StackHPC team are excited to explore this avenue further and combine Slurm Bridge with our existing solutions for cloud-native HPC and AI research computing infrastructure.

Could your institution benefit from dynamic resource shaing in a cloud-native supercomputer? If so, come and talk to us!

Check out our blog post for more technical details.