CASE STUDY

Orchestrating AI Infrastructure Resources at Scale for EPFL

Learn how one of the largest research data centers in Europe uses Run:ai to launch their GPU as a Service and increase the efficiency of their compute resources with GPU virtualization.

Stats at a Glance

- Pooled more than 350 GPUs on a single Kubernetes-based cluster
- Reduce idle GPUs by more than 50%
- Hundreds of researchers with self-service access to GPUs

Customer

The École Polytechnique Fédérale de Lausanne (EPFL) is a research institute and university in Lausanne, Switzerland that specializes in natural sciences and engineering. EPFL manages one of the largest clusters of NVIDIA GPUs, with a mission to innovate in the fields of education, research, and technology transfer.

The Challenge

With a large number of research labs, hundreds of researchers, and one of the largest data centers in Europe with over 350 NVIDIA GPUs, the IT team at EPFL needed a better way to centralize and manage their AI Compute infrastructure. At the time, they had 8 separate IT support teams, no common networks, lacked services, and did not have tools to manage nodes between the labs.

Users were frustrated that there weren't enough compute resources at peak times, and research labs could not dedicate resources for larger experiments.

Initially, the IT team wanted to scale their infrastructure by creating a private GPU cloud with Kubernetes. However, they quickly realized they needed outside help. They were running Kubernetes but could not support sharing GPUs on an as-needed basis during the AI build process. They faced challenges managing multiple experiments and setting priorities between jobs and models. They also experienced a shortage in GPU resources.

In addition, the IT team had to coordinate GPU allocations manually using email and Excel sheets while researchers were leaving their allocated GPUs idle, which created more overhead.

Solution

EPFL turned to Run:ai to help create an efficient mass self-service solution, provide common provisioning tools, gain real-time visibility on GPU performance, optimize the scheduling of GPUs, and improve GPU utilization.

Creating "GPU as a Service"

With Run:ai, the IT team was able to orchestrate the usage, performance, and allocation of their GPUs to ensure that hundreds of researchers across different labs would get the right resources at the right time. Run:ai helped create a "GPU as a service" that was dynamic, easy to use, and improved access to resources. It freed up the IT team from manually coordinating and provisioning GPU allocations via email and spreadsheets. It made it easy for researchers to request and receive resources via a self-service platform, and to make sure their AI jobs get enough AI Compute power when pooled resources are freed.

Improving the Power and Efficiency of Existing Infrastructure

Beyond creating a self-service platform, Run:ai helped EPFL make use of fractional GPUs, integer GPUs, and multiple nodes of GPUs for distributed training on Kubernetes. This enabled the IT team to get more out of their GPUs and allocate to each workload the amount of compute power it needs.

With Run:ai's GPU virtualization technology, the IT team was also quickly able to see which users are underutilizing their GPUs and adjust quotas dynamically. As a result, the number of idle GPUs decreased, and they were able to service more researchers and larger workloads with the same hardware. With Run:ai, they transformed their data center to handle AI workloads based on needs rather than capacity.

"Run:ai perfectly fits our needs, and we were pleasantly surprised to find that many of the Run:ai features solve our exact use case - hundreds of GPUs at scale managed as one very large pooled cluster. Their support and customer success teams have been very open, working with us to tackle any custom requirements based on our needs and the needs of other large-scale GPU data centers."

For more information on how Run:ai can help you automate GPU scheduling, please visit www.run.ai

Run:ai's Kubernetes– based scheduler

- ✓ Plugs into Kubernetes clusters - simple to install and use
- ✓ Guaranteed quota and over-quota scheduling ensures access to provisioned resources as well as opportunistic access to idle GPUs
- ✓ Works with any K8s "flavor" such as Red Hat OpenShift and Cloud managed K8s platforms like EKS, AKS and GKE
- ✓ Gang scheduling synchronizes containers to share and communicate information for distributed workloads
- ✓ Manages batch jobs using multiple queues allowing admins to define different policies, priorities and requirements for each queue
- ✓ Topology-awareness means that networking links & infrastructure topology are taken into account when scheduling workloads