

PROBLEM AWARE

# Slurm & Deep Learning

–  
Deep-Learning  
Workloads



## Table of contents

<b>4 Reasons Slurm Underperforms when Tackling Deep-Learning Workloads</b>	2
<b>What Is Slurm Used For in Deep Learning?</b>	2
<b>Why Slurm Falls Short for Deep Learning</b>	3-4
1. Slurm's Static Allocation Model Doesn't Fit the Data Science Paradigm	3
2. Slurm Is Complex and Difficult to Learn	3
3. DL and ML Are Increasingly Coupled with the Cloud-Native Ecosystem	4
4. Slurm Was Not Built for Inference	4
<b>Run:AI – A Solution Built for Deep Learning</b>	5

## 4 Reasons Slurm Underperforms when Tackling Deep-Learning Workloads

Thanks to the rise of advanced computing capabilities and the lower price of compute power, more and more businesses and organizations are leveraging AI to aid in what were formerly manual processes—or to initiate new processes that can't be accomplished in any other way.

Given its wide variety of use cases—from image recognition for quality assurance to real-time data analytics—deep learning (DL) offers greater versatility than standard machine learning (ML). Given enough compute power, it can derive insights and solutions from raw data with far less human intervention.

Many organizations, especially those with a background in the high-performance computing (HPC) world, first think of HPC tools when they're implementing DL. These organizations typically adopt Slurm (Simple Linux Utility for Resource Management), a leading HPC scheduling tool, to orchestrate the massive workloads associated with DL.

In some ways, this seems like a reasonable choice, since Slurm is a known quantity within the Linux world. It's also scalable, flexible, and very widely used. So it may come as a surprise that there are better options. In fact, if you're using Slurm for DL, it might be holding you back from achieving the full performance you need.

In this post, we'll look at some of the unique challenges posed by DL workloads. We'll then examine four reasons Slurm underperforms when tackling those workloads. We'll also explore how you can simplify all your DL workloads to get better results quicker, while taking advantage of all the resources available.


---

## What Is Slurm Used For in Deep Learning?

Slurm is very good at what it's designed to do: serve as an open-source and highly scalable [HPC workload manager and job scheduler](#) that works with most Linux distributions. For this reason, it seems to many like a logical choice at first. And since AI and ML are often viewed as a subset of HPC, many users have worked hard to implement Slurm as their HPC solution for DL workloads.

Scalability is a major plus of Slurm: It was designed to handle 100,000+ jobs, both in queues and on compute nodes. It also supports both synchronously parallel jobs and job arrays, which is essential, since every ML and data-science application relies on the power of interactive, massively parallel computations. Slurm also helps schedule the rapid dispatch of high multiples of tasks in parallel, allowing you to scale ML frameworks to tens of thousands of cores.

However, although these requirements are essential, they're not the only demands when it comes to DL workloads.



## Why Slurm Falls Short for Deep Learning

Slurm may be the most widely accepted framework for AI applications, both in enterprise and academic use, though other schedulers are available (such as [LSF and Kubernetes kube-scheduler](#)). But not all HPC tasks are created equal, and since Slurm was not expressly designed for DL, it can cause frustration and create bottlenecks.

Here are four reasons Slurm isn't the best choice for handling your DL tasks.

### 1 Slurm's Static Allocation Model Doesn't Fit the Data Science Paradigm

Slurm schedulers have a [few inherent drawbacks](#) when it comes to applications in data science. These include long wait times to start jobs and the inability to set limits for partitions by default (these can only be set by the user, meaning you can't maximize the number of users).

Data science, and especially DL, also demand long sessions, which Slurm doesn't handle well. And Slurm's visibility into GPU utilization and management isn't great. This means that you don't know where your resources are allocated, making it very difficult to define and set policies for GPU compute with the granularity you need.

Also, when users can't reasonably estimate the memory and time required, they often hoard resources:

- Users over-request resources, which leads to inefficiencies (idle or underutilized resources that could be better used for other jobs).
- When resources are tied up in this way, wait times increase even further, creating bottlenecks.

On top of this, the majority of HPC and ML users ["come from other disciplines than Computer Science."](#) Because of this, as well as the inadequacies already inherent in Slurm, they lack the knowledge and proficiency to allocate resources effectively and efficiently.

### 2 Slurm Is Complex and Difficult to Learn

While Slurm may be a favorite of experienced Linux system administrators, it can be overly complicated for people in a range of disciplines who are looking for the most straightforward way to orchestrate their DL workloads.

The ability to pause and resume, which is essential for long-running DL applications, can be especially difficult to program into Slurm jobs. Also, if your organization is using multiple HPC systems, it's important to know that Slurm code may not be portable.

It's certainly possible for experienced users to create powerful, fault-tolerant, and useful applications with Slurm. However, many people running DL jobs aren't experienced, nor are they interested in spending the time it takes to become Slurm experts. They just want results, and they're looking for a tool that can get those results as quickly and easily as possible.

### 3 DL and ML Are Increasingly Coupled with the Cloud-Native Ecosystem

Cloud-native technologies, built to run in any environment (but with cloud and hybrid-cloud infrastructures in mind), are enabling developers to create, deploy, and scale their services more easily. Kubernetes is a leading orchestration tool created to simplify management of these technologies, especially microservices.

As data science becomes more prevalent, today's leading ML tools are increasingly being built based on cloud-native technologies and Kubernetes, instead of Slurm. Kubeflow, MLflow, NVIDIA GPU Cloud (NGC), and others all have tight integration with Kubernetes. NGC, for example, provides pre-made data-science containers and tools for Kubernetes and is widely used by data scientists and MLOps engineers.

Kubernetes offers some advantages that Slurm doesn't. Most significantly, it's simpler to use. It also democratizes access to resources, meaning users can get the compute they need when they need it, without having to request infrastructure provisioning.

Additionally, because Kubernetes is becoming standard in areas like IT and operations, more and more people are also adopting it for DL. And as a native cloud-based tool, Kubernetes provides greater inherent scalability and portability compared to other environments.

However, Kubernetes doesn't natively offer some of the more flexible scheduling options that Slurm does, which is one barrier to widespread Kubernetes adoption for DL. Its default scheduler, [kubescheduler](#), is missing some of the batch system capabilities that Slurm has, which creates inefficiencies. And even more inefficiencies arise when resources are left idle.

### 4 Slurm Was Not Built for Inference

Most organizations train ML models in order to move them to production. While Slurm helps in the training stage, it was not built for running services and applications—Kubernetes was built for that. Load balancing, auto-scaling, and additional capabilities are crucial for services and model inference servers.

Along with the growth of ML, a new type of team has emerged: MLOps, which is situated between data science, dev, and IT. MLOps teams are critically important in the AI/DL space. That's because the ability to run models in production, create automatic pipelines that retrain models, and then move them to production is crucial. [According to Google Cloud](#), MLOps aims to unify the development (dev) and operation (ops) of ML systems, introducing automation and monitoring at all stages along the way.

All of this brings us back to the overarching benefit of a single orchestration platform that can run training workloads in both the research/development stage and the production stage. Having a single solution that can build, train, and deploy can be crucial for your organization's agility and ability to move quickly from research to production. That means finding a solution that works with today's cloud environments and can handle the specific inference needs of ML.



## Run:AI – A Solution Built for Deep Learning

Run:AI is an orchestration platform designed specifically for the intensive demands of DL. It gives you all of the tight, granular scheduling flexibility that Slurm offers, but also works hand in hand with Kubernetes so you can handle your AI and DL workloads faster—and with greater flexibility and efficiency.

You'll get all the transparency you need with Run:AI's flexible visual dashboard that shows real-time resource utilization. This lets you take full advantage of compute, scheduling, and orchestration—and eliminates the risk of jobs being killed.

Run:AI also offers:

- ✓ More efficient use of resources with pooling and dynamic allocation.
- ✓ A simplified way to set policies and orchestrate jobs with advanced scheduling algorithms.
- ✓ Full IT control of GPU utilization, with efficient access for data scientists as needed.

In addition, for DL to work best, you need a system that automatically distributes workloads across as many resources as necessary—without interruption or job killing. Neither Slurm nor Kubernetes alone gives you this capability. However, since Run:AI was designed for DL from the ground up, this feature is built right in.

Run:AI also lets you move AI models into production faster. Its dynamic and completely virtual model gives you the freedom to compute what you need, when you need it, and exactly how you need it to run.

If you want infinite compute with guaranteed availability, [sign up for a free trial today](#).

## About Run:ai

Run:ai is an AI management platform for MLOps, Data Science, and DevOps teams. In addition to helping these teams access and utilize their GPU resources more effectively, it also has a powerful set of features that can abstract infrastructure complexities and simplify the process of training and deploying models. With or without a GPU shortage, Run:ai enables data scientists to focus on innovation without having to worry about resource limitations.

Read more about how Run:ai supports  
data scientists here

[www.run.ai/runai-for-data-science](https://www.run.ai/runai-for-data-science)