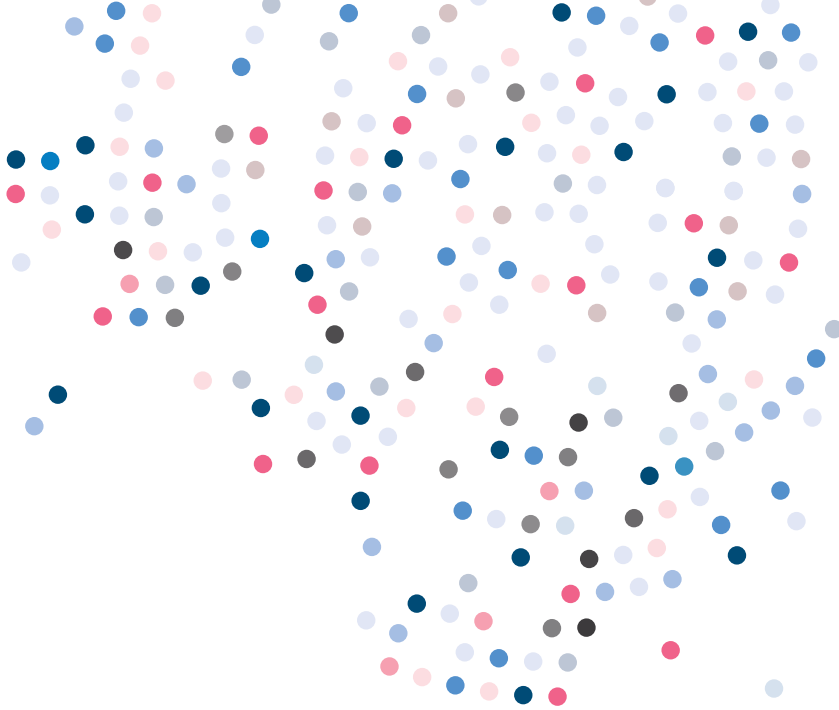
run:
ai



The Essential Guide:

Machine Scheduling for AI Workloads on GPUs

Abstract

Newly emerging AI technologies pervade every industry: game-changing products and services like voice-controlled devices, autonomous vehicles, and even cures for illness, are here or on the horizon. Organizations that are AI-driven are making their products more intelligent and optimizing processes like operations and decision-making. These capabilities are transforming industries and revolutionizing business.

Deep Learning (DL) is at the epicenter of this revolution. It is based on complex neural network models that mimic the human brain. The development of such DL models is extremely compute-intensive and has been enabled, in great measure, by new hardware accelerators that satisfy the need for massive processing power. Organizations are investing heavily in bringing AI accelerators into their data centers or using them on the public cloud, but continue to struggle with the cost-effective and efficient management of these critical resources.

This white paper addresses the challenges of expensive and limited compute resources and identifies solutions for optimization of resources, applying concepts from the world of virtualization, High-Performance Computing (HPC), and distributed computing to deep learning.

Contents

Deep Learning Meets the Enterprise

Which industries are actually implementing AI? Burgeoning AI in the enterprise is impacting almost every industry, from medicine to retail, automotive to financial. A recent 451 Voice of the Enterprise report ("AI and Machine Learning Use Cases 2020") found that 29% of enterprises have deployed machine learning in some capacity. "We expect to see organizations using machine learning to solve increasingly complex use cases as they move from applying AI to problems to which they have previously applied deterministic software approaches (that is, rules-based ones) to new use cases that were not previously solvable using software alone."

DATA SCIENCE BREAKTHROUGHS – THE STARS ALIGN

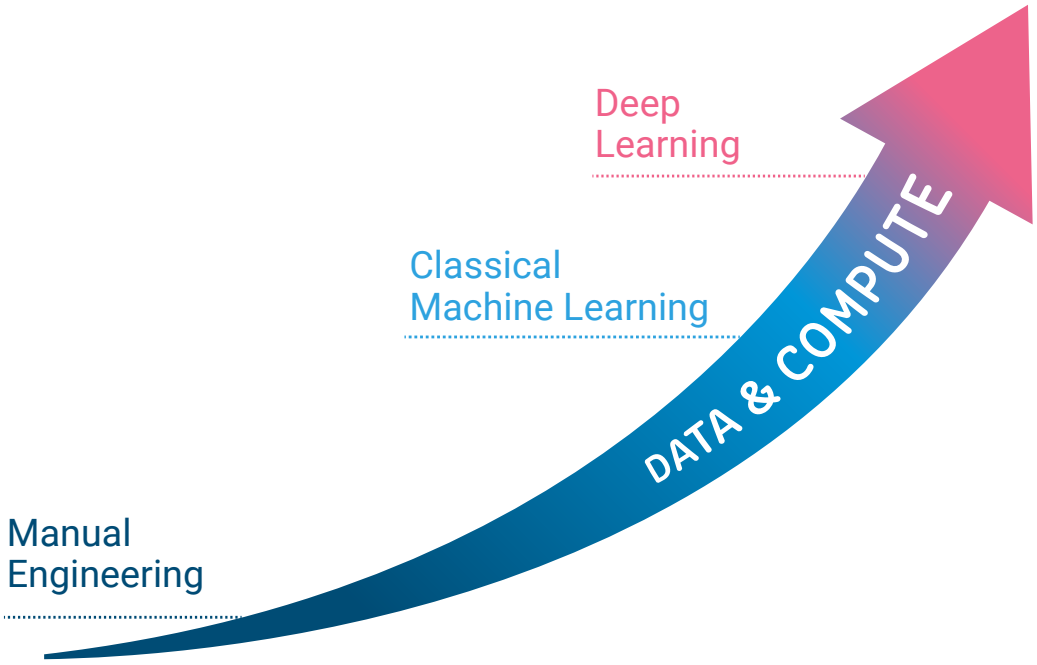


Figure 1: Exponential growth in data and computing needs to support Deep Learning

There are two primary factors fueling the acceleration of development in Deep Learning (DL):

1. A vast increase in the quantity of enterprise data. Proliferation of data such as text, images, and video, feeds computational DL models that enable advances in areas like Natural Language Processing and Computer Vision.

2. Exponential increases in the availability of raw computing power. Hardware accelerators, such as Nvidia's GPUs (Graphics Processing Units) and Google's DL-dedicated TPUs (Tensor Processing Units), are readily available, making AI accessible to every enterprise.

Because of the factors mentioned above, Deep Learning is becoming more commonplace in the enterprise. As more enterprises see the success of the applications of DL technology, they venture ahead to develop new technologies that apply to their specific markets and consumers. But, as interest in and adoption of DL in the enterprise emerges, there are challenges in the process of implementing data science initiatives that stop some enterprises from realizing success.

DEEP LEARNING ≠ SOFTWARE ENGINEERING

Enterprise IT has grown comfortable with the traditional cycles of software development, and in recent years DevOps has emerged to streamline development and create efficiencies and scale around deploying code. While DevOps automates the process of building and deploying code, data scientists build and deploy models.

The process of building an ML model is different than developing software. It includes prototyping, exploring different approaches for the problem they are solving for, preparing data, model training, and finally, deploying models into production. Many of these tasks will eventually be automated, but today a lot of manual work is involved. One of the primary differences between software engineering and deep learning relates to their different needs around computing infrastructure.

Deep Learning is different than traditional software development in that it is based on experimentation. Tuning to achieve optimal parameters is what these experiments are doing - essentially choosing the right model architecture with the optimal hyperparameters and combination of values and weights for the problem at hand. Sometimes this process takes weeks and many of these experiments run in parallel. Therefore, data centers are experiencing a new strain - many workloads which are all running simultaneously, all of them highly compute intensive.

U.S. DEEP LEARNING MARKET, BY SOLUTION, 2014-2025 (USD MILLION)

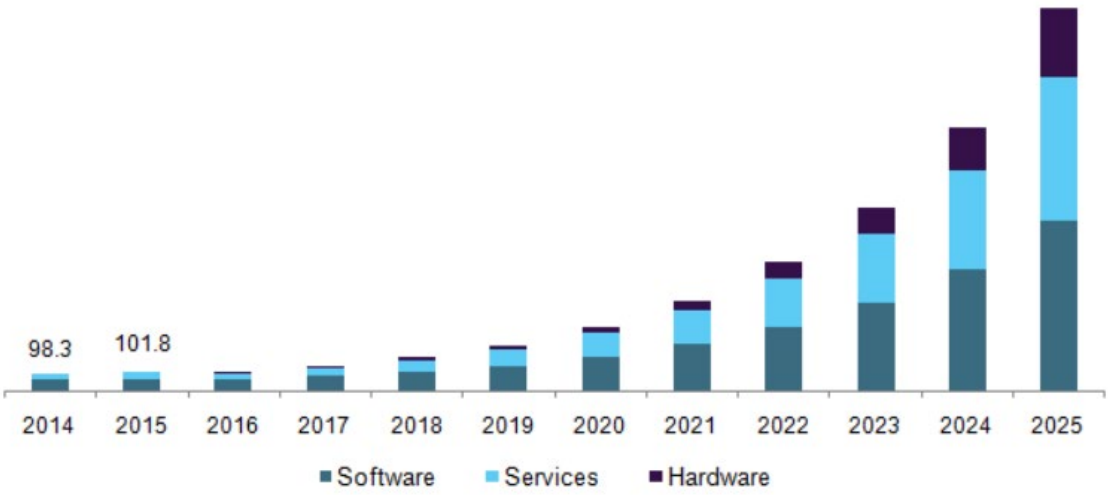


Figure 2: Increasing hardware needs to support burgeoning Deep Learning activity
Source: www.grandviewresearch.com/industry-analysis/deep-learning-market and https://www.grandviewresearch.com/industry-analysis/machine-learning-market

These compute-intensive workloads are run on specialized hardware that is growing in popularity. As the graph above shows, enterprises are investing in deep learning software, services and particularly in specialized AI hardware at unprecedented levels. According to GrandView Research, "The global machine learning market size was valued at USD 6.9 billion in 2018 and is anticipated to register a CAGR of 43.8% from 2019 to 2025."

Nvidia's GPU currently dominates the market for specialized AI hardware accelerators. New accelerators like Google's TPU are now available, and others are in development. Intel has invested billions of dollars in this area in the last years and many startups have entered the field as well. Some of them may eventually perform better than GPU, as they are built from the ground up for DL workloads. As more dedicated AI chips come to market, compute power should become less expensive and better able to meet demand.

Unfortunately, the cost of compute power is not the only challenge facing data science in the enterprise.

IT AND DATA SCIENCE ARE NOT YET ALIGNED

As machine learning initiatives grow, and businesses become increasingly interested in solving challenges with data science and experimentation, additional challenges are created for IT.

As companies buy additional AI accelerators like GPU, managing, maintaining and scaling the new hardware becomes difficult. GPUs are typically deployed in data centers as bare metal and are allocated statically to data scientists. Static allocations frustrate data scientists by slowing down the process of experimentation, which should be able to use more compute for heavy computations and less for lighter ones. Instead, bottlenecks are created when researchers try to run their models.

A recent 451 study highlighted this challenge, "Nearly half of enterprises indicate that their current AI infrastructure will not be able to meet demand."

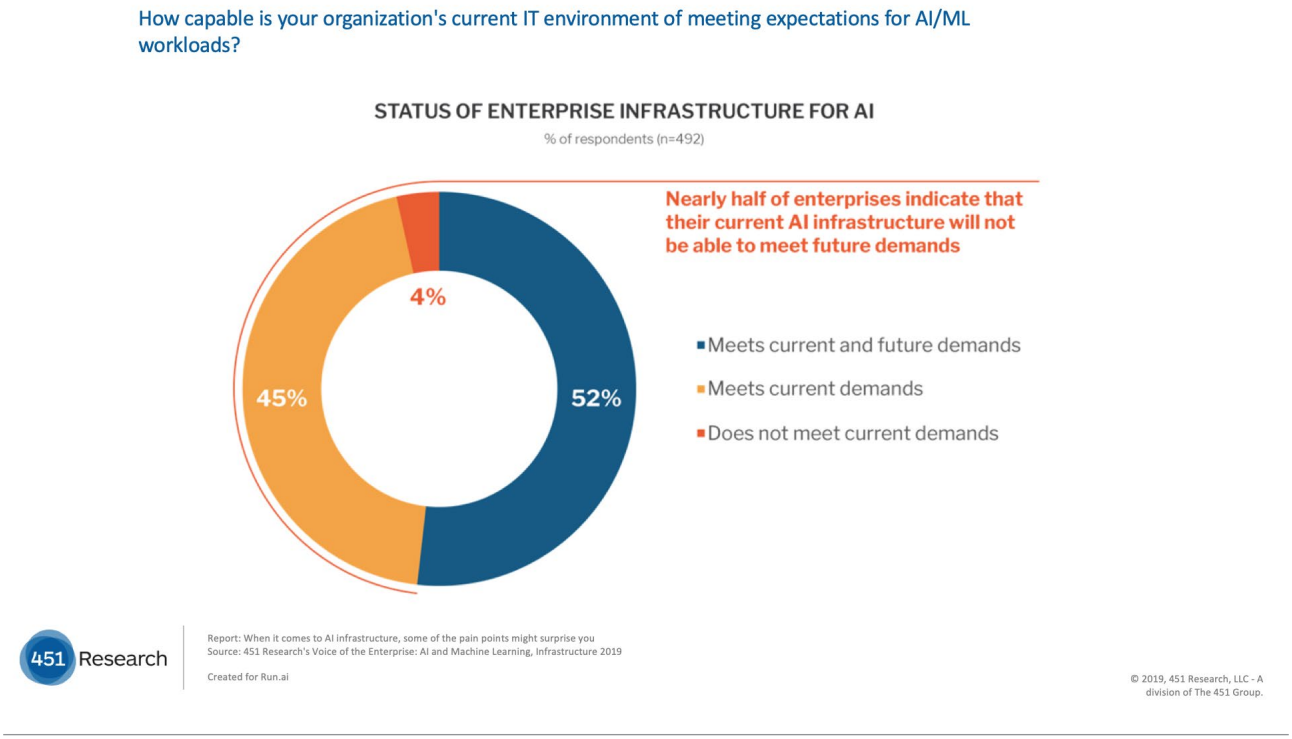


Figure 3: Findings in answer to the question: "How capable is your organization's current IT environment of meeting expectations for AI/ML workloads?"

In addition to needing a seemingly endless supply of compute, managing bare metal creates infrastructure hassles for IT teams. Visibility of infrastructure across teams and locations, maintenance of hardware, provisioning of new resources, and other critical tasks are cumbersome.
Organizations that are investing heavily in GPUs find it difficult to manage allocations and achieve efficient use of hardware. This leads to having clusters of AI accelerators with very low utilization, and frustrated users whose productivity is limited by inefficient infrastructure, manual processes, and long and expensive development cycles.

Run:AI's understanding of the causes of inefficient use of GPUs in data science experimentation is at the center of the Run:AI technological solution.
More about that below, in Data Scientists: Two Profiles.

WHAT ABOUT KUBERNETES?

Can't Kubernetes solve infrastructure bottlenecks and help to orchestrate workflows?

Containers play an important role in the world of data science. Containers make experimentation easier by creating an environment that incorporates all of the dependencies that a data scientist needs in order to perform a task.

Kubernetes, an orchestration layer for those containerized workloads, was built to help automate deployment, scale, and management of containers. Kubernetes has become the de-facto standard for container orchestration and is already implemented in many enterprise IT environments. Could a Kubernetes scheduler be used for AI workloads? Not really. Kubernetes does not address some of the complexities of running AI workloads and managing those workflows. Unfortunately, Kubernetes lacks the batch scheduling capabilities that are crucial to AI scaling, like fair scheduling with management of multiple scheduling queues and automatic queueing or dequeuing of workloads according to priorities and policies, efficient management of multi-node distributed workloads with gang scheduling, and more.

In addition, Kubernetes is unable to perform topology-aware scheduling that has a critical effect on the performance of DL workloads. Topology-aware scheduling refers to the ability of the scheduler to optimize job placement by taking into account the exact hardware setup of components like CPU sockets, Network Interface Cards (NICs), GPU interconnects, and more.

Kubernetes schedulers were not built for AI workloads and therefore are a long way from being the sole orchestration tool bridging AI computing infrastructure and AI workloads.

At Run:AI we built a solution to solve both of the challenges listed above – management and optimization of GPU, using Kubernetes to simplify the implementation. Let us explain in more detail in the next chapter.

The Run:AI Solution – Simplifying GPU Management and Machine Scheduling

Run:AI aims at abstracting hardware accelerators away from data scientists and DL engineers to fast track the development and deployment of DL projects. At the heart of the Run:AI vision is a Kubernetes-based platform that decouples data science workloads from hardware accelerators so that scientists can focus on building models, sending those models for processing over the hardware accelerators. Scientists can focus on how to set up and manage their infrastructure.

Run:AI seeks to provide the ideal data science computing infrastructure, including:

- Elastic compute resources that afford every data scientist as many resources as are needed,
- The ability to train and run large-scale distributed computing in one click,
- A tool that provides scalability and maximizes the utilization of expensive GPU resources,
- A powerful platform based on a simple Kubernetes plug-in, removing complexity for both data science and IT teams.

Run:AI's virtualization software comprises an automated, distributed computing technology, allowing IT administrators to manage resource allocations more efficiently, and ultimately reduce GPU idle time and increase cluster utilization. From the data scientist's point of view, virtualizing AI infrastructure allows them to consume more GPU compute power, to either run more experiments, which improves their productivity and the quality of their science, or to run distributed training using multiple GPUs, which shortens training times. These improvements translate into a 2x increase in hardware utilization and 2x faster model training.

DATA SCIENCE WORKFLOWS: TWO PROFILES

The Run:AI solution is based on a separation of two distinct data science workflows.

The two workflows are **train** and **build**. In many organizations, this is seen as a single workflow but from a computational standpoint, they are completely different:

- Training is generally associated with very long, compute-intensive workloads that go on for days and even weeks, while tuning parameters takes place. Thus, with this type of workload, optimizing utilization of a company's GPU resources is critical, as these are high memory and compute-intensive workloads.

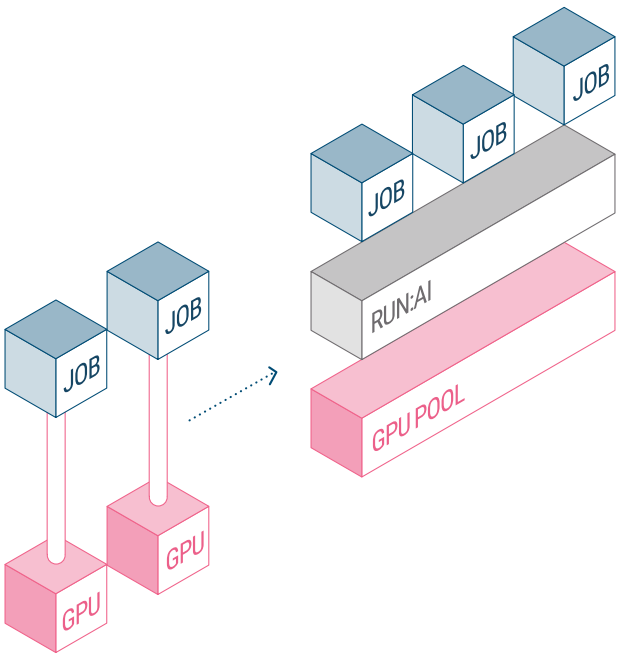
- Before the training process begins, however, data scientists typically build a training model. They code, they tweak, and they debug in an effort to connect the data to the code and launch a training session. Looking at the computational needs of this build phase, it can be characterized as an interactive process with short input cycles from both data scientists and GPU. Performance is far less important and GPU utilization is typically very low.

In summary:

Build	Train
Development and debugging	Training, hyperparameter optimization
Interactive execution	Unattended execution
Short cycles	Long runtime
Throughput is not that important	Throughput is highly important
Low GPU utilization	High GPU utilization

Now that the differences between the two workflows are clear, let's look at them in the context of the life cycle of developing an AI model with the Run:AI platform.

Run:AI's software works by pooling heterogeneous resources so they can be used within two logical environments, to natively support build and train compute characteristics and increase utilization. The virtual pool exists inside a Kubernetes cluster.



THE RUN:AI SCHEDULER

Run:AI's dedicated batch scheduler, running on Kubernetes, enables crucial features for the management of DL workloads. This includes an advanced multi-queue mechanism, fixed and guaranteed quotas, a manager for priorities and policies, automatic pause/resume, multi-node training, and more. It provides an elegant solution to simplify complex scheduling processes.

By pooling the resources and managing them using the Run:AI scheduler, administrators gain control - easily aligning resources with business goals, onboarding new users, maintaining and adding new hardware to the pool - and visibility, including a holistic view of GPU usage and utilization. In addition, data scientists can automatically provision resources without depending on IT.

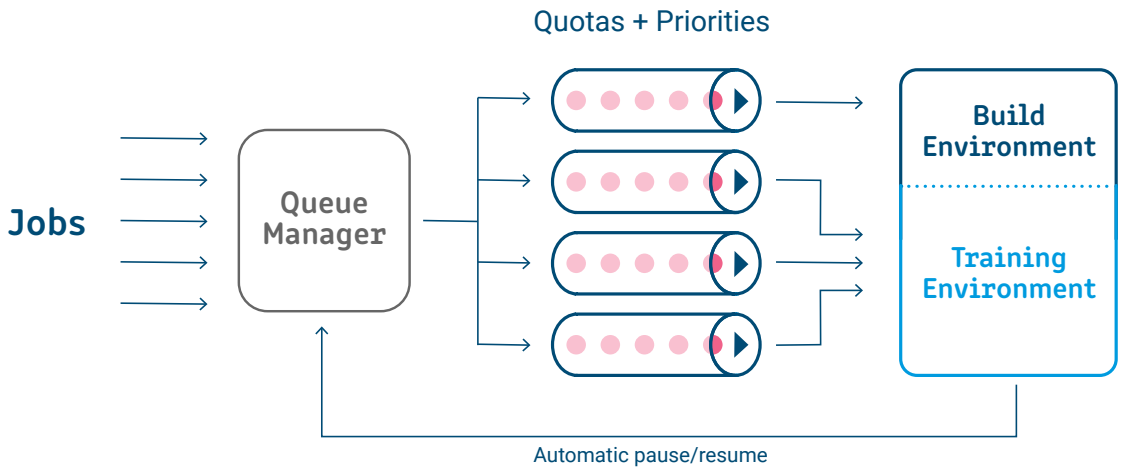


Figure 4: Run:AI Scheduler

The pool of two logical environments, for build and for training workloads, interacts with the Run:AI scheduler.

FIXED AND GUARANTEED QUOTAS

The status quo in most deep learning initiatives today is to use fixed quotas of resources. Researchers are assigned specific GPU for their projects. As opposed to projects with a fixed quota, projects with a guaranteed quota of GPUs can use more GPUs than their quota allows and minimize idle resource time. To do so, the system allocates available resources to a job, even if the job is assigned to a project which is over quota.

Guaranteed quotas essentially break the boundaries of fixed allocations and make data scientists more productive, freeing them from limitations of the number of concurrent experiments they can run, or the number of GPUs they can use for multi-GPU training sessions. This greatly increases utilization of the overall GPU cluster.

Fixed quotas	Guaranteed quotas
Fit most build workloads	Fit train workloads
GPUs are always available	Users can go over quota
	More concurrent experiments
	More multi-GPU training

Ideally, GPUs for build workflows should always be available for users. It's the data scientists' dev and debug environment, and they need ready access to it. Training workflows on the other hand are used for experimentation. It's more opportunistic. In an ideal scenario, they would have as many resources as their models need in the experimentation phase. When GPUs are idle, the system can allow data scientists that currently need to run more experiments to use those idle GPUs.

Projects translate into the Run:AI Scheduler as queues and depending on the organization's preference. They can be modeled as individuals, as teams of people, or as actual business activities, where each queue has its own priority and quota.

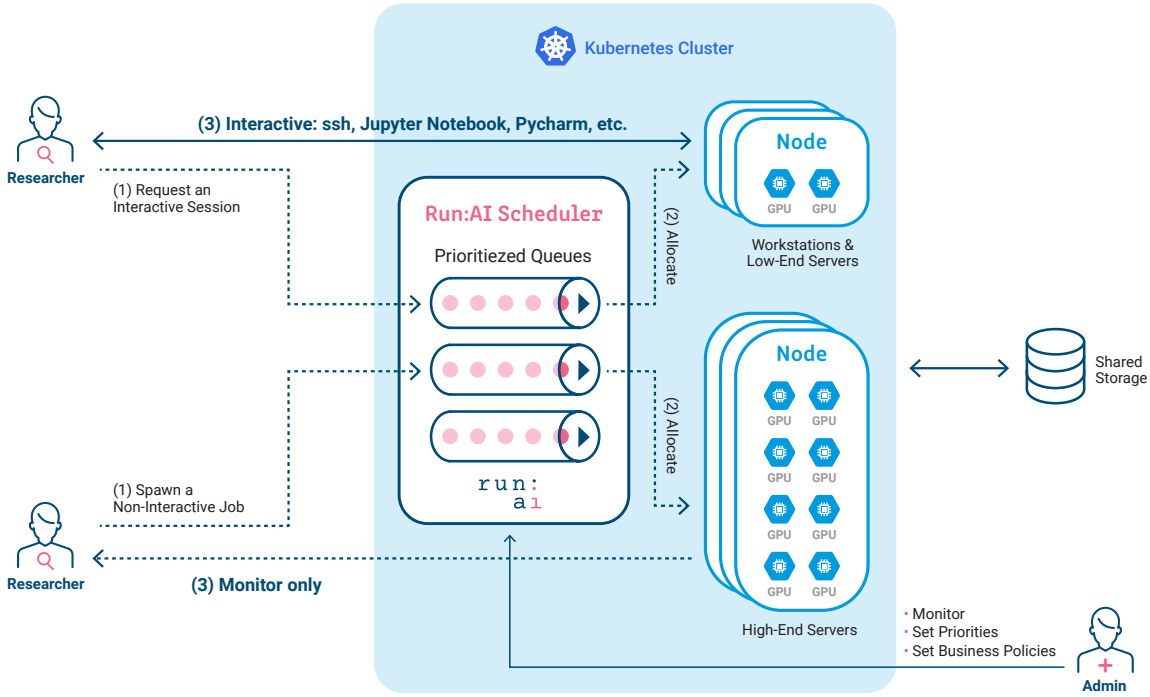


Figure 5: Assignment of GPUs to different researchers for both build and training environments

So how do the Run:AI Scheduler and Guaranteed Quotas help with optimal resource utilization? In the diagram above, this is explained. Two environments are shown, one is the build environment and the other is the training environment.

- For the build environment, fixed quotas of GPUs are assigned to users.
- For the larger, training environment, the greater proportion of the pool's resources is assigned, so that all the resources can be easily shared among all the users.
- Within the larger training environment, Run:AI uses guaranteed quotas to ensure that jobs with higher priority get the resources they need.

Projects with a guaranteed quota of GPUs, as opposed to projects with a fixed quota, can use more GPUs than their quota allows. So ultimately the system allocates available resources to a job submitted to a queue, even if the queue is over quota. In cases where a job is submitted to an under-quota queue and there are not enough available resources to launch it, the scheduler starts to become smarter and pause a job from a queue that is over quota, while taking priorities and fairness into account.

SUSAN NEEDS MORE COMPUTE

Let's put it in plain speak.

Each day Susan, the data scientist, gets a static amount of resources, for example, two GPUs that she can use. Those GPUs are always available, but only available for her. There is usually no problem to interact with the GPUs for short cycles and her workflow is smooth - at least for the build part - when her focus is to iteratively develop and debug a model before she trains it for the long run. But what happens when the train phase kicks in? Suddenly, her workloads demand additional GPU compute that is often unavailable, and resources are limited to fixed quotas that she cannot exceed, even in cases where her peers are not using their resources.

Typically, Susan will need more compute resources to do three things:

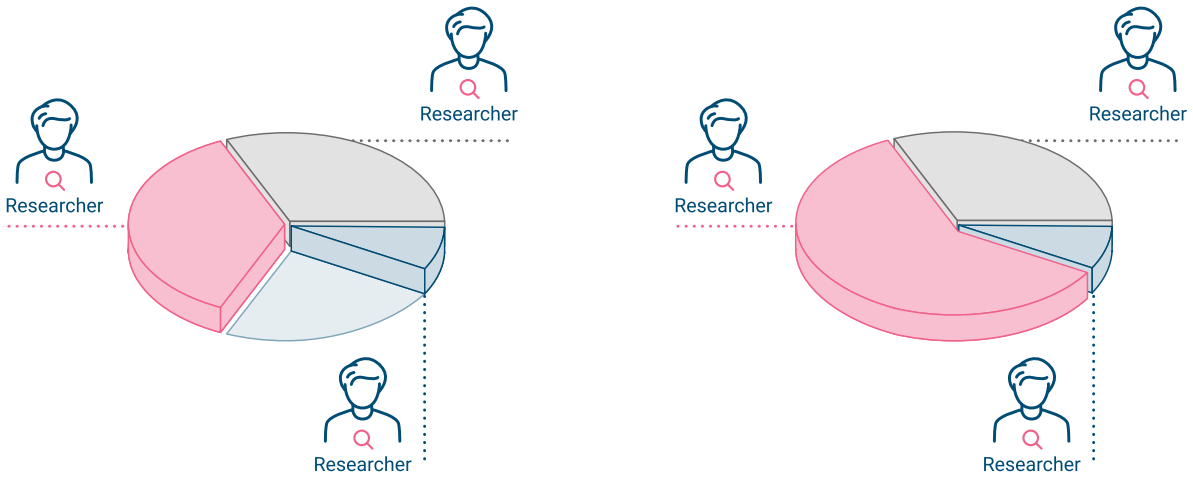
1. Run more experiments
2. Run multi-GPU training to speed up her training, and to enable her to experiment with larger batch sizes and achieve higher model accuracy
3. Build a new model while a training model runs on its own

With Run:AI, Susan is capable of all of these.

How? With guaranteed quotas. All the resources of Susan's organization are pooled together and she can have a much larger assignment of GPUs, providing her and every other user with flexible computational power, based on our guaranteed GPU quota.

As mentioned above, every data scientist on Susan's team has a fixed quota when they are building models. The system will provide a fixed quota for the build, so the scientists are always able to interact with GPUs. However, once they want to go over quota, to run more experiments, use multi GPU training, run larger jobs or multiple experiments in parallel, they just send jobs to be orchestrated in the larger pool that is shared in the training environment.

If Susan is not using the GPUs that are assigned to her, other people can use them. But if she needs them, they will always be made available to her. That's her 'guaranteed quota'. The preempted job will be automatically restarted once resources become available again.



WHAT HAPPENS WHEN OTHER RESEARCHERS ENTER THE PICTURE?

In the figure below Susan, Joe and Jemma are sharing an eight GPU cluster. In a world of static allocations or fixed quotas, each user gets two GPUs and no more. In a world of guaranteed quotas, in contrast, there is a virtual assignment of GPUs to each of the users; Susan can always access two GPUs but if she needs four GPUs and her workmates are idle, she can access four. If Joe or Jemma would need to use those GPUs again, the system would reclaim them and bring them back. Schematically, it looks like this.

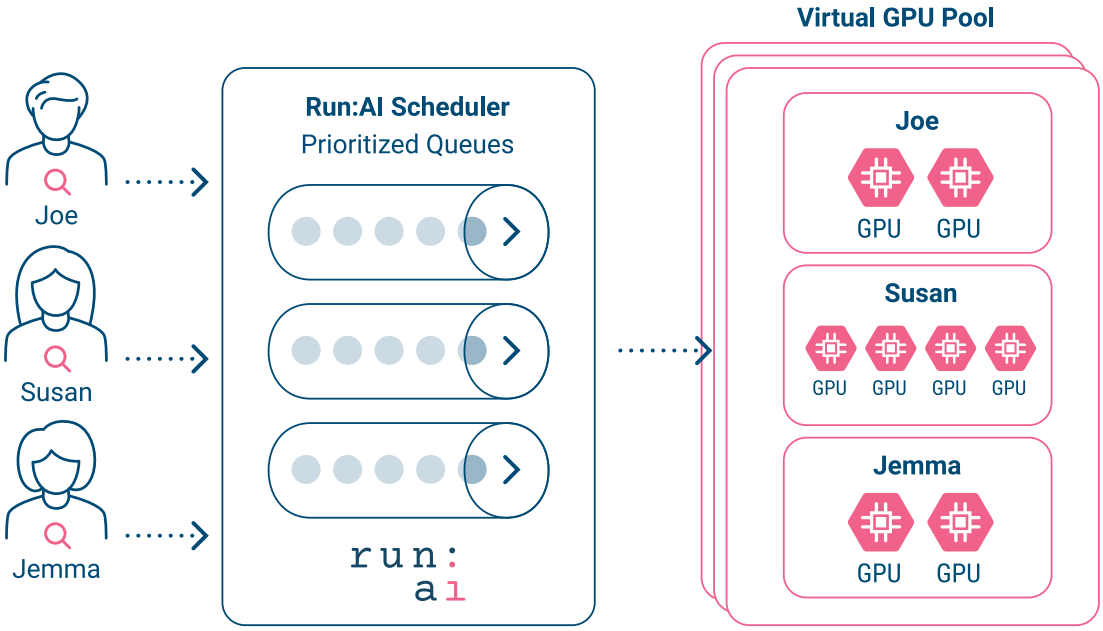


Figure 6: Sharing a pool of GPU for training.

In summary, with guaranteed quotas, if users are idle and there is free capacity, jobs will be allocated, even if they are associated to an over-quota project. When new jobs enter the system, the Run:AI Scheduler manages those queues according to business goals and provides a mechanism that stops and resumes workloads according to priorities and policy. This increases the utilization of the overall cluster and enhances the productivity of the data science teams.

Run:AI in Action – Case Study

Run:AI recently engaged with an organization that employs 30 Deep Learning engineers. When we first met, these engineers were working in a heterogenous hardware environment that included a mix of different types of GPUs -- from low-end GPU workstations to high-end DGXs and servers with Nvidia Tesla V100s. Before Run:AI's involvement, the organization had been assigning the hardware statically. It had started out with a small group of data scientists and engineers, each of whom was assigned dedicated resources in a manual, static way whereby different users received different quantities and different types of resources. When the group started to grow and DL initiatives scaled, IT bought more and more hardware units to keep pace.

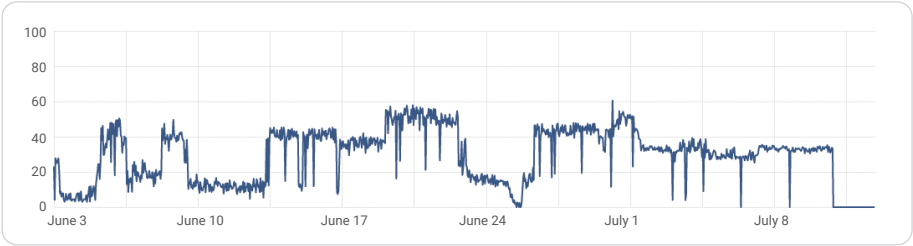
Instead of seeing the organization's productivity scale as they expected, they were met with growing frustration. New users that joined the organization received dedicated resources, but the productivity of the team remained relatively low. Concerned data scientists were frustrated with infrastructure bottlenecks despite mushrooming hardware resources. Overheads were skyrocketing, yet ROI remained elusive.

Run:AI's initial analysis of the utilization of the GPUs in the cluster quickly revealed the classical symptoms:

- Very low all-round GPU utilization of the cluster- in this case, only 28% (Figure 7)
- Relatively high, but fluctuating utilization for multi-GPU (train) jobs (Figure 8)
- Very low utilization on several interactive (build) jobs - the GPU is idle for long periods and even when jobs run, utilization is lower than 50%. (Figure 9)

LOW GPU UTILIZATION

Some peaks, but mostly inefficient and underutilized resources



DIFFERENT USAGE PROFILES

The top graph shows Training workloads were often taxing GPU capacity

The bottom shows mostly idle resources for Build workloads

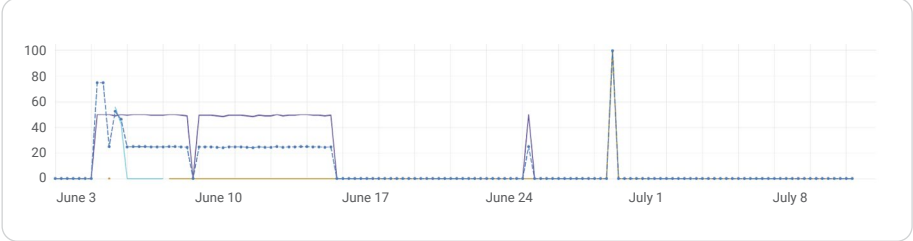


Figure 7,8,9: Metrics before integration with Run:AI

This was obviously a sub-optimal scenario -- some users were working with their GPUs intermittently (producing the peaks of utilization in graph 8 above), while most of the time, the GPU was idle. They were seeing an overall utilization of only 28%.

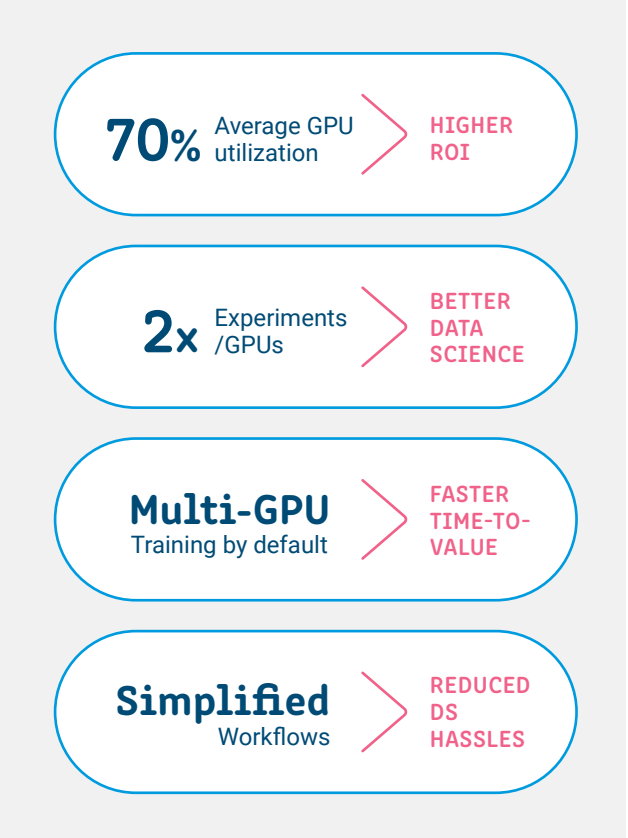
INCREASING UTILIZATION AND SPEEDING UP EXPERIMENTS WITH RUN:AI

Once the company chose Run:AI, their first task was to integrate the Run:AI system into the Kubernetes cluster and pool together all of the resources.

Then the GPUs were divided virtually into two groups:

1. **Build** with **fixed quotas**: in this group, each researcher got a workstation with a fixed quota of two low-end GPUs so that they could still interact with GPUs while building interactive models. There was no resource sharing in this group.
2. **Train** with **guaranteed quotas**: in this group, GPUs were assigned to researchers who were training models. The newer, high-end DGXs were used but all of the GPUs were pooled together to maximize computing power for workloads that crunch for days. This enabled them to see results faster. In this group, Run:AI assigned policies and implemented guaranteed quotas.

On top of all of this, the Run:AI management tool was installed to give the customer a holistic view of all the GPUs, how they were utilized and how admins could intervene to further optimize utilization.



Now…

- There are more experiments being run and results are achieved faster - productivity measured as experiments-per-GPU has doubled
- Average all-round utilization of their GPU cluster stands at more than 70%
- When looking at this in terms of cluster utilization, that translates as higher ROI on hardware expenditure
- Considering that data scientists are no longer concerned with infrastructure and tools but rather on increasing the number and quality of experiments, that represents simplified workflows and superior data science
- IT departments gain better visibility and control, easy onboarding of new users, a simplified way to integrate new hardware, automatic resource provisioning, and analytics to help with the planning of future resource spend

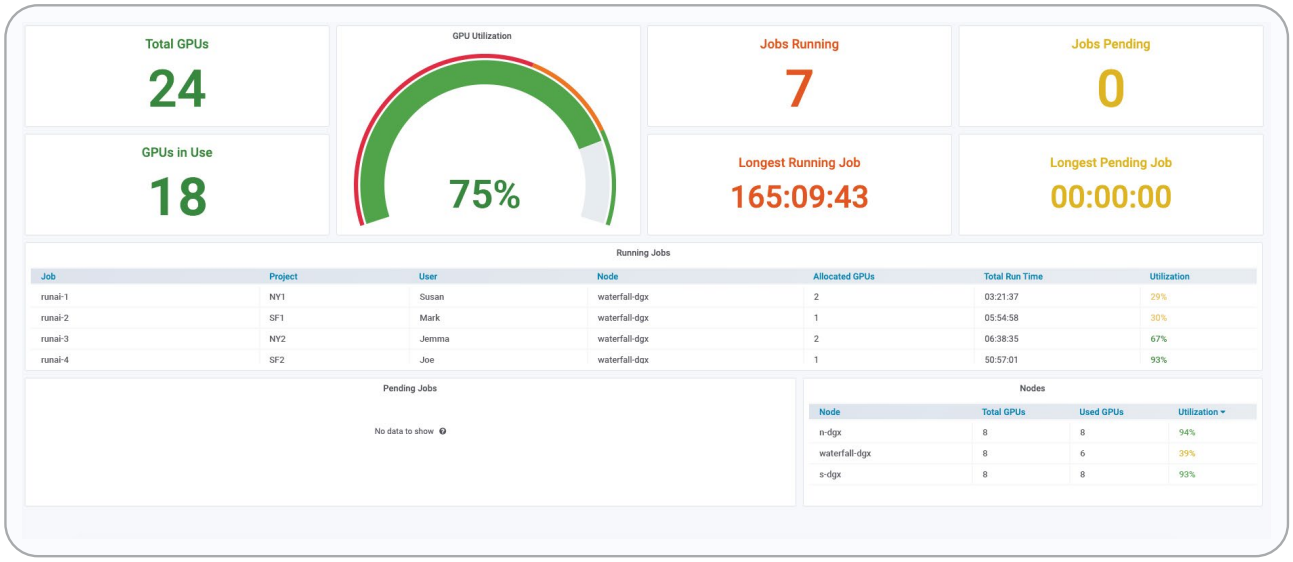


Figure 10: The Run:AI GUI shows 75% utilization of the company's infrastructure.

Conclusion

The steep increase in the application of Deep Learning to practically every vertical is bringing to bear a dramatic need for increased computing power. For enterprises with increasing data science initiatives, the mere purchase and deployment of additional hardware units is, alone, not enough to provide the solution. Without a strategy for a cost-effective and efficient utilization of these resources, lack of productivity and elusive ROI will continue to be the order of the day.

The Run:AI software solution decouples data science workloads from hardware by defining two task types according to two distinct workloads – a smaller build type and a larger train type, assigning limited GPU resources to the smaller type and guaranteeing unlimited resources to the larger.

By pooling resources and applying an advanced scheduling mechanism to data science workflows, with guaranteed quotas for priority jobs, Run:AI customers can increase the number of experiments they run, speed time to results, and ultimately meet the business goals of their AI initiatives.

For more information on virtualizing your AI infrastructure with Run:AI, contact us at info@run.ai.