AI-Enabling Workloads on Large-Scale GPU-Accelerated System: Characterization, Opportunities, and Implications

Baolin Li*, Rohin Arora*, Siddharth Samsi†, Tirthak Patel*,
William Arcand†, David Bestor†, Chansup Byun†, Rohan Basu Roy*, Bill Bergeron†,
John Holodnak†, Michael Houle†, Matthew Hubbell†, Michael Jones†, Jeremy Kepner†,
Anna Klein†, Peter Michaleas†, Joseph McDonald†, Lauren Milechin†, Julie Mullen†, Andrew Prout†,
Benjamin Price†, Albert Reuther†, Antonio Rosa†, Matthew Weiss†,
Charles Yee†, Daniel Edelman†, Allan Vanterpool‡,
Anson Cheng‡, Vijay Gadepally†, Devesh Tiwari*

*Northeastern University, †MIT Lincoln Laboratory, ‡US Air Force

***Abstract*—Production high-performance computing (HPC) systems are adopting and integrating GPUs into their design to accommodate artificial intelligence (AI), machine learning, and data visualization workloads. To aid with the design and operations of new and existing GPU-based large-scale systems, we provide a detailed characterization of system operations, job characteristics, user behavior, and trends on a contemporary GPU-accelerated production HPC system. Our insights indicate that the pre-mature phases in modern AI workflow take up significant GPU hours while underutilizing GPUs, which opens up the opportunity for a multi-tier system. Finally, we provide various potential recommendations and areas for future investment for system architects, operators, and users.**

I. Introduction

This paper shares the operational experience gained from managing and running the GPU-based MIT Supercloud supercomputer [41] that is primarily installed to serve AI needs for domain scientists and engineers. GPUs are not new to the systems and architecture community, or the cloud computing data centers. The systems and architecture community has more than a decade of experience operating GPU-based systems. Many HPC centers share and celebrate the success of running GPU-accelerated supercomputers [17], [49], [52]. However, as a community, we do not have a deep understanding of the basic characteristics of how our users use these GPU-accelerated large-scale systems. In particular, we have a limited understanding of the primary purposes of the GPU-accelerated systems, the resource consumption characteristics of AI workloads, the types of jobs that are executed, and how efficiently the users use these systems.

However, this lack of understanding is not fully unexpected. When a new kind of computing accelerator [42] emerges, it takes multiple years to port mission-critical applications. In the meantime, the hardware also matures, and early experiences often share the maturity and reliability of new hardware in the production system. As the hardware gets more mature and powerful, new applications emerge and take advantage of the new hardware. GPUs have followed a similar path over the last decade. GPUs are now at a point where their hardware reliability maturity is well-proven and has enjoyed wide adoption from a variety of computational science applications. Consequently, this is a timely point to understand how users use the GPU resources and the common characteristics.

Our deployment and operational experience with a GPU-accelerated supercomputer enable us to ask these questions about the user job and system characteristics. To the best of our knowledge, one of the most closely related works is a recent study from Microsoft [23]. While useful, this prior study lacks the level of detail needed in our community to design better architectural/system techniques and better operate GPU-accelerated systems. In particular, the previous study lists only high-level statistics (e.g., runtime and number of GPUs used), but it does not provide detailed information about the fine-grained resource utilization of GPUs (e.g., streaming multiprocessor utilization, GPU memory utilization, power consumption). The previous study does not provide details about how the characteristics of the jobs belonging to the same and different users vary over time. Our study bridges these understanding gaps and provides several new insights useful for the systems and architecture community running deep learning and machine learning workloads, the primary workloads of Supercloud.

Our study brings significant advances over the previous studies due to richer and more fine-grained information/GPU-level statistics available in our dataset and new trends/opportunities *We share multiple operational-related experiences, new opportunities, and lessons learned which, we hope, are useful to the community.* **The highlights of our characterization include the following.**

We find that most GPU-accelerated jobs tend to have low utilization of different resources such as SM, memory, and PCIe bandwidth. However, resource utilization can vary greatly during job execution, switching between idle and active phases, even reaching the maximum capacity at times. This property opens up the need to develop production-suitable dynamic techniques to share non-contending GPU resources among concurrent jobs to improve machine utilization and job throughput, without having a large impact on job performance [7], [18], [58], [61].

Interestingly, we find that users have a variety of job run times and GPU utilization characteristics, including "expert" users with many jobs and GPU hours. The system can be best served by identifying users' needs and providing efficient solutions. Our system employs one such solution by providing different interfaces for different job types for scheduling. On Supercloud, a large fraction of the jobs are multi-GPU and a majority of the users run at least one multi-GPU job. Characterizing these jobs is important because of their notable GPU hours footprint. *While 40% of the multi-GPU jobs have idle GPUs, the resource consumption tends to be similar among active GPUs.* This allows the same resource management policy to be deployed across all GPUs of multi-GPU jobs.

We present a new method of classifying HPC jobs: mature, exploratory, development, and IDE. We find that a large fraction of the jobs on our system is exploratory (e.g., hyper-parameter tuning). Users can benefit from a two-tier system design, with GPUs of different qualities being priced differently, so that they can run their lower-utility jobs on the slower GPUs. We discover a paradigm shift on HPC systems as they proceed toward catering GPU-accelerated artificial intelligence and machine learning workloads: a considerable portion of the users execute a high rate of low-utilization development and IDE jobs. These jobs can benefit from resource sharing and state-saving (model checkpointing).

Overall, to the best of our knowledge, this study is the first work to classify the deep learning jobs in mature and non-mature jobs, and is the first to show that jobs have irregular and unpredictable utilization phases. We discover that even jobs submitted by the same user vary widely in terms of their resource consumption characteristics.

To promote rapid advancement in this area, our relevant framework, data, and project updates are available publicly at `https://dcc.mit.edu/` Finally, we note that our findings are based on our experience with Supercloud and should be generalized or extended to other systems with caution and further quantitative validation.

II. System Specifications and Operational Lessons

In this section, we provide a description of our system specifications, its workloads and users, our dataset and methodology, and the operational details and lessons. The system studied in this work is the MIT Supercloud.

System Description. The Supercloud system is an HPC cluster designed for GPU-accelerated AI workloads, including the needs of the Air Force and government labs. It consists of 224 Intel Xeon Gold 6248 based nodes (two CPUs per node, each with 20 cores; 2-way hyperthreading per-core), each

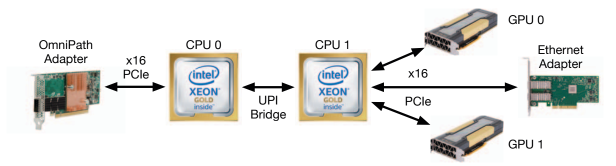


Fig. 1: Configuration of a node on the Supercloud system.

TABLE I: Specifications of the Supercloud system.

Node Specifications			
Number of Nodes	224	Node RAM	384 GB
Number of CPU Cores	8960 cores (two CPUs per node, each with 20 cores; 2-way hyperthreading per-core)	Processor	Intel Xeon Gold 6248
Interconnect	100 Gb/s Omnipath two-layer partial fat-tree	Network	25 Gb/s Ethernet CX-4
GPU Specifications			
Number of GPUs	448	Type	Nvidia Volta V100
GPUs per Node	2	RAM	32 GB
Storage Specifications			
Local	1 TB SSD & 3.8 TB HDD	Shared	873 TB SSD

with two Nvidia Volta V100 GPUs. There are 448 GPUs in total, each with a RAM of 32GB. The node configuration is shown in Fig. 1. Each node has a RAM of 384 GB, with a local storage capacity of 1TB SSD + 3.8 TB HDD and a total distributed solid-state storage capacity of 843 TB. The Supercloud system's specifications are summarized in Table I.

Types of Workloads and Users. Several different types of workloads run on the Supercloud system: deep learning, batch jobs, interactive jobs, map-reduce, data visualization and analysis, and general machine learning and artificial intelligence workloads. The Supercloud system is designed as an interactive system to be used by novice users who are using the system to actively design and prototype their AI solutions. Unlike other production HPC clusters, the Supercloud system is also designed for users to prototype, experiment, and deploy their workflows. A typical user's workflow is described in Fig. 2. A large subset of Supercloud users tend to use the system to develop, train and benchmark models in PyTorch [33] and Tensorflow [1]. Typical workloads include large language models [6] as well as published image classification models such as ResNet50 [21] and VGG16 [50] for specific tasks. The Supercloud cluster provides a comprehensive suite of open-source tools for serial and distributed training of neural networks, including tools such as PyTorch, PyTorch-Lightning [16], Tensorflow, Horovod [47], and many other software dependencies. The infrastructure also includes high-performance computing tools such as MPI and the Nvidia Collective Communications Library (NCCL) configured for GPUDirect and RDMA over ethernet for optimal performance. User requirements typically include bleeding-edge or recently published open-source implementations of deep learning libraries or publications, which can be easily installed in the userspace. Supercloud also hosts commonly used deep learning datasets in a shared space to avoid unnecessary duplication of data.

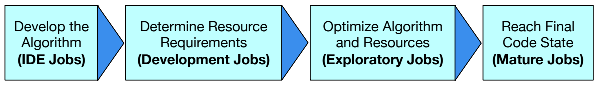

Develop the Algorithm (IDE Jobs)
Determine Resource Requirements (Development Jobs)
Optimize Algorithm and Resources (Exploratory Jobs)
Reach Final Code State (Mature Jobs)

Fig. 2: A typical user's interaction with the Supercloud system.

System Monitoring. The Supercloud system uses Slurm workload manager for job scheduling. Data collection for this study leveraged Slurm's built-in capabilities to enable transparent, lightweight, and automated system monitoring [60]. Specifically, Slurm provides job prolog programs to start time-series monitoring of the jobs running on the cluster. It also includes a number of plugins that can report detailed job data such as CPU usage, memory usage, and file I/O, which can be used to understand job behavior. We also used vendor-provided software for monitoring GPUs. The Nvidia System Management Interface (`nvidia-smi`) is a command-line utility that enables the management and monitoring of Nvidia GPU devices [10]. The Slurm prolog is used to start the collection of CPU-based time series data on every node assigned to a job running on the system. Additionally, if the job requests one or more GPUs, the prolog also launches the `nvidia-smi` utility on all nodes assigned to that job. Both time series are written to independent files on the local storage on each compute node as a way to avoid overloading the cluster-wide shared file system. The CPU time series data is collected at 10-second intervals and the GPU time series data is collected at an interval of 100ms. Both time intervals were empirically chosen as a compromise between data volume and usability. The collection of both time series is automatically stopped at the end of a job through the SLURM epilog, which runs at job termination. The epilog is also responsible for copying the collected data back to the central file system of Supercloud.

Dataset Description. Over the duration of our study of 125 days, 191 unique users executed 74,820 jobs in total on the Supercloud system. For GPU analysis, jobs running for less than 30 seconds are filtered out since no activity is observed for these very short jobs, and 47,120 jobs are considered. The dataset is anonymized and for all jobs, the minimum, mean, and maximum resource utilization of a variety of CPU and GPU metrics are collected. CPU and scheduler-related data is reported from Slurm logs, and GPU-related data is profiled using the `nvidia-smi` command. In the end, both datasets are combined using job Ids to create a single dataset. We also collected a detailed log of resource utilization of 2149 jobs at an interval of 100ms (time series dataset) to study variability during the run.

General Methodology. We study different job and user characteristics including run times, queue wait times, power consumption, PCIe bandwidths, and job functions. We also study the utilization of different GPU resources including the GPU streaming multiprocessors (referred to as SM utilization), the GPU memory bandwidth (referred to simply as memory utilization in keeping with the Nvidia terminology), the GPU memory amount (referred to as memory size utilization). For each of these metrics, the average over multiple GPUs was computed to get a single number for multi-GPU jobs. We generally use empirically-obtained cumulative distribution functions (CDFs), star charts, pie charts, box plots, and correlation bars to present our results. Detailed methodological details and figure explanations are provided in conjunction with the result presentations in the following sections. Most of the analysis was done using the SciPy stack (Pandas, Matplotlib, Numpy). A multi-core accelerated version of Pandas (Modin) was leveraged to analyze the large (42 GB) time series dataset.

System Operations Details. Supercloud's job queue is configured based on hardware type. At the time of this paper's research and submission, the system was homogeneous and there was a single job queue for all jobs, regardless of the job's function, number of CPU cores, requested wall times, number of GPUs, and CPU memory requirement. In the interim, new CPU-only hardware also has been added to the system and supports multiple queues based on hardware requests (GPU/CPU-only) [44]. Below, we summarize some of our major lessons learned from operating a GPU-accelerated supercomputer, primarily designed to serve AI workloads.

Summary of Operational Challenges and Experiences

Our biggest challenge has been maintaining the software stack updated as the hardware evolves. Although this work only describes experiences with only one system, our data center has multiple supercomputers and hence, the hardware is heterogeneous. Somewhat to our surprise, keeping the software stack updated and compatible has been most demanding. In particular, performing ML regression tests on multiple GPUs for scaling and efficiency testing is challenging and time-consuming. As new software versions roll out, regression testing is done on (new) hardware to ensure compatibility. This testing suite also checks the multi-GPU functionality of deep learning workloads. But, more automated tools and workflows are needed in this domain.

Related to previous learning, our users spend a significant portion of their effort in re-tuning their libraries and software stack as the hardware changes. A large share of our positive user experience is attributed to making this process smoother for our users. User requests often include missing packages and newer library versions. Therefore, twice every year the library versions (Anaconda, PyTorch, and TensorFlow) are frozen and released system-wide as modules. This ensures reproducibility and also saves the effort of user install. We have observed a high usage of PyTorch in the past year, compared to TensorFlow.

Scaling of deep learning workloads from single-node to multi-node settings is non-trivial for many users, whom we train to correctly configure SLURM and MPI and get them working with deep learning libraries. The usage of the system

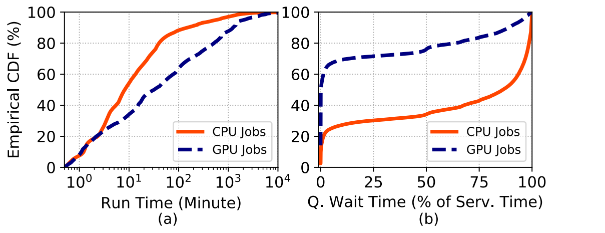


Fig. 3: (a) Distribution of runtimes of GPU and CPU jobs. (b) Queue wait time of GPU and CPU jobs as % of their total service time.

often increases closer to the deadlines of popular deep learning conferences like ICML and NeurIPS and there are requests for increased allocations. We account for this effect in our analysis.

Hardware reliability has been fairly stable over the recent few years and accounts for less than 0.5% job failures. Instead, we are observing that different deep learning jobs have different utility and hence, may be early terminated based on observed training loss and hence, future resource managers and schedulers need to plan for such shift in user behavior – discussed in more detail in Sec. VI.

Monitoring resource consumption is key to improving operational efficiency for any data center. Over time, we learned that while vendor-provided tools are useful in many contexts, especially in generic data center context, they need tuning and tweaking to be production-suitable for HPC centers. For example, the logging tools can easily overload the metadata server and shared file system. Also, they can interfere, creating load imbalance among the processes of the same job due to the potential malfunction of one of the nodes – especially harmful for parallel jobs that rely on tight synchronization. Therefore, HPC centers need to invest effort in making such tools (e.g., DCGM [11], nvidia-smi) suitable and ready for their systems – easy plug-and-play is not always possible.

III. Characterization of General Job Behavior and Resource Utilization

We begin our analysis by presenting the general characteristics of the workloads that run on the Supercloud system.

The most basics of these characteristics are the service time characteristics such as the run times and the queue wait times of the GPU-accelerated jobs (GPU jobs). Fig. 3(a) shows the empirical cumulative distribution function (CDF) of the queue wait times and run times of all GPU jobs. First, we look at the characteristics of the GPU jobs. While the median run time of GPU jobs is 30 minutes, GPU jobs have a diverse set of run times: the 25th percentile run time is 4 minutes and the 75th percentile is 300 minutes (note that the x-axis of run times has a logarithmic scale). In fact, the run time for some jobs can be as low as less than 1 minute, and for others, it can be as high as more than 20 hours.

Next, we look at the runtimes of CPU jobs (jobs without GPU request) for comparison. The figure shows that CPU jobs run for a much shorter duration than GPU jobs. While the median run time of GPU jobs is 30 minutes, the median run time of CPU jobs is only 8 minutes. This disparity exists because of the long-running, multi-threaded, GPU-intensive workloads, such as deep learning and recommendation model training, big data analysis and visualization, and scientific computation jobs that are run on the GPUs [9], [20], [48].

While CPU jobs tend to have shorter run times, they tend to have relatively longer queue wait times than GPU jobs. To demonstrate this, we plot an empirical CDF of the queue wait times as percentages of their respective service times (queue wait times + run times) in Fig. 3(b). More than 50% of the GPU jobs spend less than 2% of their service times waiting in the queue, while less than 20% of the CPU jobs do the same. The reason for this is not only because CPU jobs have shorter run times, but they also have longer queue wait times: 70% of the CPU jobs spend more than one minute in the queue; in contrast, 70% of the GPU jobs spend less than one minute in the queue. The reason for the long wait times of CPU jobs is that CPU jobs usually request all cores and full memory of the nodes (or a large fraction at least), as CPUs are their only computational resource. In the case of GPU jobs, users do not need all CPU cores and memory available on the node, and they request fewer CPU cores and memory. This results in the GPU jobs being scheduled quickly because the required resources are more likely to be available as they can be co-located on the same CPU node. Note that Supercloud does not co-locate jobs on the same GPU at this point. However, it allows CPU resources to be divided among jobs.

Takeaway: As a system designed for GPU-accelerated jobs, Supercloud is highly efficient at processing and serving time-critical GPU workloads, with 70% of the GPU jobs spending less than one minute in the queue. While HPC systems typically tend to have very long wait times because they only support exclusive node reservations [35], [49], [52], based on regular and active interactions with Supercloud's users, our system administrators have determined that GPU jobs do not tend to have high CPU resource requirements. Therefore, Supercloud achieves low wait times by investing in provisioning enough resources to meet the GPU demand and co-locating not-CPU-intensive GPU jobs on the same CPU node. While our operational experience suggests that such a strategy could be useful at Supercloud, and potentially worth exploring for some other similar HPC centers if GPU job characteristics are similar to Supercloud, there is further need and opportunity for effective techniques for co-locating network-bandwidth bounded GPU jobs. Mitigating network-bandwidth-caused interference at the production level requires more mature tools.

Next, we drive the discussion to other resource utilization characteristics of GPU jobs. We do not compare these characteristics to CPU jobs as the resources and their usages are

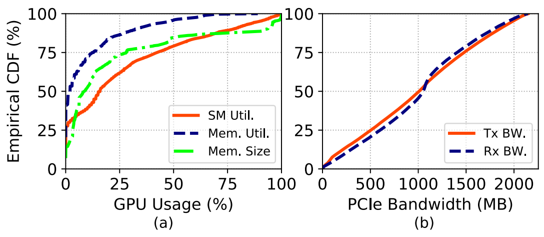


Fig. 4: (a) Distribution of GPU jobs in terms of their GPU resource utilization. (b) Distribution of GPU jobs in terms of their PCIe Tx and Rx bandwidth utilization.

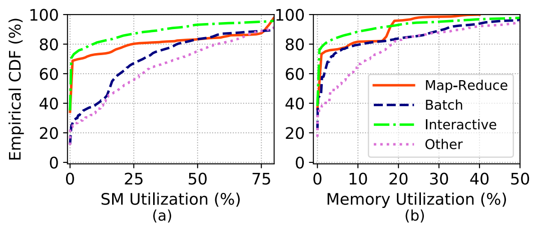


Fig. 5: SM (a) and Memory utilization (b) of different types of GPU jobs (map-reduce, batch, interactive, and other).

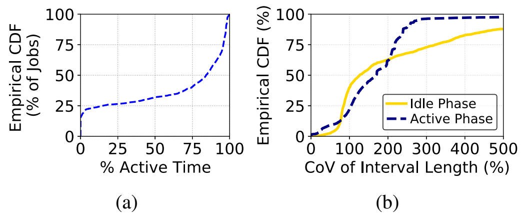


Fig. 6: (a) Empirical CDF of the total amount of time that jobs spend in active phases as a percentage of their run times. (b) Empirical CDF of the coefficient of variation (CoV) of the lengths of idle and active intervals during a run.

disjoint for CPUs and GPUs, and they cannot be directly compared. Hence, from here on, we only discuss GPU jobs (referred to simply as "jobs").

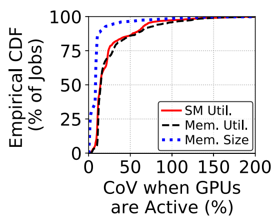
Fig. 4(a) shows the empirical CDFs of the average utilization of different GPU resources including SM (usage percentage of the GPU streaming multiprocessors), memory (percentage of the GPU memory bandwidth used), and memory size (percentage of the GPU memory amount used). In general, we observe that SM (median utilization is 16%) is more heavily utilized than memory bandwidth (median utilization is 2%) and size (median usage is 9%). This indicates that GPU workloads tend to be more compute-intensive, which is as expected as they are mostly deep learning and machine learning jobs. This trend is also in keeping with the workload behaviors observed on other contemporary systems [24], [49], [52]. In addition, Fig. 4(b) shows that different jobs achieve a wide range of PCIe bandwidths for both the transmission of packets (Tx bandwidth) and the receiving of packets (Rx bandwidth). The linearly increasing empirical CDF trend suggests a uniform distribution of bandwidths across all jobs. This indicates a large variety of data transmission patterns between the GPUs and the host CPUs, depending on the job requirements.

Interestingly, from Fig. 4, we observe that most jobs do not fully utilize GPU resources: only 20% of the jobs have more than 50% SM utilization, only 4% of the jobs have more than 50% memory utilization, and only 15% of the jobs have more than 50% memory size usage. This demonstrates the opportunity for space-sharing of GPU resources among multiple jobs. For example, a large portion of the jobs ($\approx 30\%$) have close to zero GPU SM utilization and 40% memory utilization, i.e., they are memory-intensive. These jobs can potentially share space on the GPU with SM-intensive jobs, without the two types of jobs contending with each other for resources.

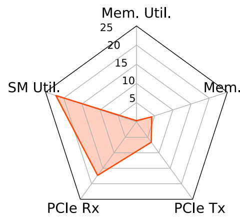
To further explore in this direction, in Fig. 5(a) and (b), we plot the SM utilization and memory utilization, respectively, of different types of jobs: map-reduce (1% of all jobs), batch (30%), interactive (4%), and other (65%). Note that we are able to identify map-reduce, batch, and interactive jobs as they are submitted using their individual interfaces. Other jobs (mostly deep learning jobs, and other artificial intelligence, machine learning) are submitted via the general Slurm interface. We find that these "other" jobs have the highest SM and memory utilization as they are GPU-intensive, followed by batch jobs. On the other hand, map-reduce and interactive jobs tend to have low SM and memory utilization as most of their time is spent in data movement and user-interactive debugging and development, respectively.

Takeaway: Most GPU-accelerated jobs tend to have low utilization of different resources such as SM, memory, memory size, and PCIe bandwidth ($< 50\%$). Different jobs have different levels of utilization of each resource. This property indicates the opportunity to share non-contending GPU resources among concurrent jobs to improve machine utilization and job throughput, without having a large impact on job performance [13], [18], [61]. However, interference during co-location is difficult to mitigate by prediction. Based on our experience, it is worth considering building a system having different GPU tiers (with different prices), giving users the flexibility to run their low-utilization jobs (e.g., interactive jobs) on slower/smaller GPUs, leaving higher-performance GPUs for jobs that need them. One can apply new research techniques on our data to demonstrate that the economic benefits of such an approach is a more user-friendly solution for many systems including Supercloud.

While the low average utilization opens up abundant opportunities for GPU resource-sharing, we note that the problem space is not straightforward. This is because the average utilization behavior of a job is not representative of the utilization behavior at all times during the run. To find out

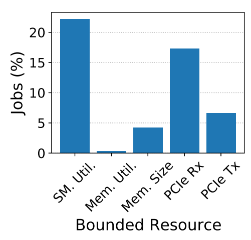


(a)

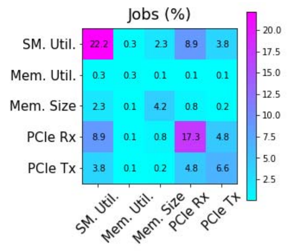


(b)

Fig. 7: (a) Empirical CDFs of the CoVs of different GPU resources (SM,memory, and memory size). (b) Percentage of jobs which have a particular resource as the bottleneck.



(a)



(b)

Fig. 8: Fraction of all GPU jobs experiencing resource bottlenecks: (a) single resource (b) two resources

if this is the case, we collected a detailed time-series log of resource utilization for a representative fraction of jobs at an interval of 100ms. Note that in the previous analysis, the minimum, mean, and maximum resource utilization during the run were reported at the end of the job execution – this was done to achieve low overhead during production, instead of fine-grained (e.g., 100ms) data collection for all jobs.

Our analysis of the logs reveals that the GPU jobs have "active phases" and "idle phases." GPU resources are used during the active phases and they remain unused during the idle phases (only the host CPUs are used). Fig. 6(a) shows an empirical CDF of the total amount of time that jobs spend in active phases as a percentage of their run times. The figure shows that the median job spends 84% of its time actively using the GPUs (i.e., the GPUs spend 16% of a median job's run time idling). The 75^{th} percentile active GPU time is 95%, and the 25^{th} percentile active GPU time is only 14%. Moreover, Fig. 6(b) shows the empirical CDFs of the coefficient of variation (CoV) of the lengths of idle and active intervals during a run. CoV is the standard deviation of the lengths of all idle or active intervals during a run as a percentage of the mean length. The figure shows that both idle (median 126%) and active (median 169%) phases have a high CoV in general, i.e., they do not occur at a fixed periodic interval. Thus, we found that GPUs can be idle during a significant portion of a median GPU job's runtime, and these idle phases occur at irregular intervals. *This suggests that while the GPUs can be shared among multiple jobs, this has to be achieved carefully, depending on their irregular idle phases, so as to minimize the adverse performance impact.*

In fact, even when the GPUs are actively being used, the utilization of different GPU resources may still vary. Fig. 7(a) shows the empirical CDFs of the CoVs of different GPU resources (SM, memory, and memory size). The figure shows that the median CoV of SM utilization is 14%, that of memory utilization is 14.6%, and of memory size utilization is 8.2%. Over 25% of all jobs have SM utilization CoV of 23% or higher during their active phases. This indicates a great degree of variability in resource utilization during the execution. Interestingly, during the active phases, resource usage can even reach as high as the maximum. Fig. 7(b) shows a radar plot where each edge represents the percentage of jobs that have the particular resource as a bottleneck. A job is considered to have a resource bottleneck if the maximum job usage of that resource reaches the limit at any point during the run. The figure shows that while the average SM utilization is low, 22% of the jobs have SM utilization as a bottleneck. On the other hand, $\approx$ 0% of the jobs have memory utilization as a bottleneck. This demonstrates that resource utilization can vary drastically during job execution, and therefore, resource sharing techniques should consider the temporal variations and bottlenecks in utilization.

Next, we dig deeper to understand the resource bottlenecks for various GPU jobs in more detail. First, in Fig. 8 (a) (barplot version of Fig. 7(b)), we show the fraction of jobs that have hit the maximum utilization possible for a given resource. For example, similar to the previous plot, The figure shows that 22% jobs used SM at 100% at some point during their execution. SM utilization is the primary resource bottleneck for the largest fraction of jobs. The next result (Fig. 8 (b)) shows the fraction of jobs utilizing two resources to the fullest during their runtime – although the maximum utilization may occur at different times. We observe that a small fraction of jobs experiences both PCIe Rx and PCI Tx bandwidth bottlenecks during the same run. Also, approx. 9% jobs experience both PCIe Rx and SM utilization bottlenecks during the same run. In general, jobs experiencing any two or more resource bottlenecks during the same run are less than 10%.

Takeaway: While the average resource utilization of a GPU job is low, it is not representative of the utilization behavior at all times during the job's run. Resource utilization can vary greatly during job execution, switching between idle and active phases, even reaching the maximum capacity at times. But, the co-location opportunity is present due to explicit time-spaced idle phases on GPUs. Thus, there is a need and opportunity for developing production-suitable GPU job co-location tools that take into account temporal variation and bottlenecks, such that the performances of co-located jobs are not impacted. However, this effort also needs to be augmented with online architectural tools that can predict

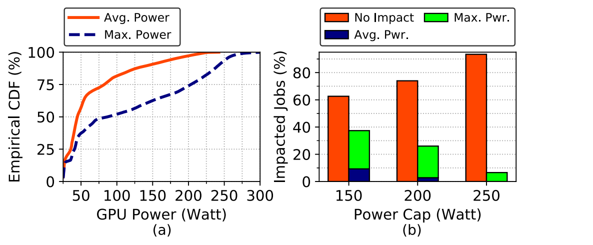


Fig. 9: (a) Empirical CDF of the average and maximum GPU power consumption of all jobs. (b) Percentage of jobs not impacted under different power cap regimes.

future idle GPU phases and resource-contention among multiple resources for more effective co-location.

The utilization of all of the above resources (SM, memory, memory size, and PCIe) directly impacts another GPU resource: power. One might expect that due to the low utilization of an average GPU's compute, memory, and network resources, its power consumption should be low. This is true for the most part as shown in Fig. 9(a). The figure shows the empirical CDF of the average and maximum (maximum recorded value during the run) GPU power consumption of all jobs. The median average power consumption is only 45W and the median maximum power consumption is only 87W, while the maximum possible power draw of the V100 GPU is 300W. This demonstrates that most jobs consume less than half or even a third of the available power on average. Previous works have demonstrated similar low-power-consumption behavior for both CPU-based and GPU-accelerated workloads [15], [28], [36], [38], [39].

However, the Supercloud system has enough power to support all GPUs at their maximum possible power, and most of this power goes unused. An effective way to use this power is to over-provision the system with more GPUs to improve job throughput and/or support a higher load. But, this would require capping the power consumption of the GPUs so as to prevent a power failure due to an unanticipated power surge. This leads us to ask how the jobs would be impacted if the GPUs were to be power-capped at different levels to support over-provisioning the system with the same amount of power. Fig. 9(b) shows the percentage of jobs that would not be impacted at all, would be impacted considering their maximum power consumption and would be impacted considering only their average power consumption for three power cap levels: 150W (50% of 300W, i.e., double the current number of GPUs can be supported), 200W, and 250W. Even with a power cap of 150W, over 60% of the jobs would remain un-impacted, and less than 10% of the jobs would be impacted based on their average power consumption. Our results identify that power-capping can be a promising method to conserve power and/or improve throughput by over-provisioning, without adversely affecting the performance of GPU jobs.

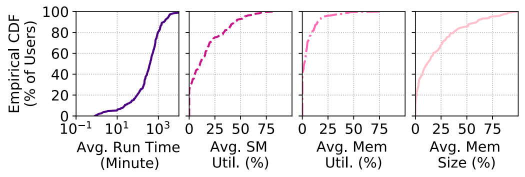


Fig. 10: Empirical CDFs of the average job run time and the average SM, memory,and memory size utilization of all jobs run by an user.

Takeaway: GPU workloads do not draw most of the power that is available to them, even at their maximum draw. This opens up the opportunity to power-cap the production AI workloads with minimal performance impact and potentially reduce the overall carbon footprint of deep learning serving systems. While promising as a research idea, it has not been evaluated for production workloads. Our evidences further support the design and representative evaluation of such methods.

However, we caution that over-provisioning a GPU-accelerated system is quite complex and should be considered carefully. Specifically, simply provisioning more GPUs may not be optimal for some systems; instead, existing GPUs could be supplied lower power. The electricity bill savings could be invested in buying more CPUs or faster storage systems instead of buying more GPUs and power-capping them. This decision requires understanding the user base and their resource requirements at a given HPC center.

IV. Analysis of User Behavior and Trend Characteristics

The analysis of general job and utilization characteristics shed light on some interesting trends. To further analyze these characteristics, we now investigate how they are affected by and differ based on the behavior of different users.

Over the duration of our study, 191 unique users executed 47,120 jobs in total on the Supercloud system. All users who executed GPU-accelerated jobs do not display the same average characteristics. To demonstrate this, in Fig. 10, we plot the empirical CDFs (across all users) of the average job run time and the average SM, memory, and memory size utilization of all jobs run by a user. User behaviors vary greatly, especially when it comes to the average job run times. The median average job run time of users is 392 minutes; 25% of the users have an average job run time of fewer than 135 minutes, while 25% of the users have an average job run time of more than 823 minutes. In fact, some users have an average job run time of as low as 49 seconds, while others have runtimes in the order of days. Similarly, users also have varying job resource utilization behaviors, with 50% of the users having an average job SM utilization of 10.75%, memory utilization of 1.8%, and memory size utilization of 11.2%. Most users have very low average job resource utilization. Only 32% and 5% of the users have an average SM and memory utilization of $> 20\%$, respectively.

While the behaviors vary from user to user, the job characteristics have lower variance when averaged by user than in the general scenario (compare Fig. 10 to Fig. 3(a) and Fig. 4(a)). The reason for this is that the top few users submit most of the jobs and their average behavior is reported. While a median user submits 36 jobs, top 5% of the users submit 44% of the jobs, and top 20% of the users submit 83.2% of the jobs. This Pareto Principle is as expected and has been observed on other production HPC systems as well [35], [43], [49]. This highlights the point that different users have different job requirements: some run only short-running and low-utilization batch jobs, some run long-running and high-utilization machine learning jobs, and others run long-running but low-utilization interactive jobs. Different users' needs must be considered for the most efficient scheduling and management of jobs. *One way that we achieved this is by providing different interfaces to submit map-reduce, batch, and interactive jobs (i.e., users classify which job class their job belongs to). This allows the scheduler to better understand their needs and provide effective resource management.* The system can be further improved by recognizing the needs of other types of jobs as well.

Takeaway: Our Supercloud users have a variety of average job run times and GPU resource utilization characteristics. Different users have different requirements and expectations from the system based on the types of jobs that they run. The system can be best served by identifying those needs and providing easy-to-use solutions to improve efficiency and throughput. In our operational experience, we implemented one such solution by providing different interfaces for different job types for scheduling. While effective, this comes at a greater caution-cost on Supercloud: the complexity of managing more interfaces and some users potentially over-burdening all interfaces. Such solutions are effective, but require greater investment in user-base management.

While the average job behaviors vary across different users, an interesting question to investigate is the degree to which the job behaviors vary among the jobs submitted by a user. For example, if all or a majority of the jobs of a user have similar behavior, then the resource demand of that user can be relatively easily predicted and met. On the other hand, if a user has a seemingly stochastic submission of multiple different types of job behaviors, then that user is more difficult to serve efficiently.

Fig. 11 shows the empirical CDFs (across all users) of the CoV of job run times and the CoV of SM, memory, and memory size utilization across all jobs run by a user. The figure shows that the behavior of different jobs submitted by a user varies greatly for an average user. As an instance, the median CoV of job run time of a user is 155%; 75% of the users have a job run time CoV of more than 86%, while 25% of the users have a job run time CoV of more than 227%. In fact, some users have a job run time CoV of over 1000%. Similarly, a median user also has a high degree of variance among the jobs in terms of their resource utilization behaviors: the median CoV of SM utilization is 121%, that of memory utilization is 182%, and of memory size utilization is 99%. *This finding reveals that individual users have a diverse set of job behaviors and their jobs cannot be treated as a monolith.*

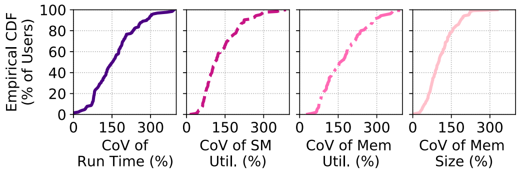


Fig. 11: Empirical CDFs of the variance in the job run times and the variance in SM, memory, and memory size utilization across all jobs run by the same user.

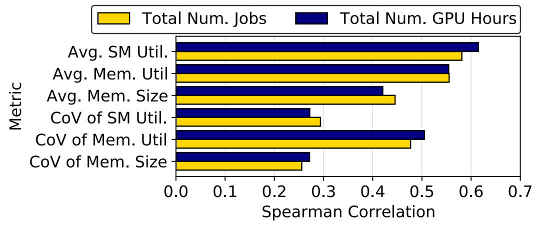


Fig. 12: The Spearman correlation of the number of jobs and GPU hours of an user with job characteristics.

Nonetheless, even if an average user does not have predictable behavior, if the users who tend to run the most number of jobs and consume the most number of GPU hours ("expert" users) have predictable behaviors, then learning about their behaviors could help make the system more efficient in general. To identify such trends, in Fig. 12 we correlate the number of jobs and the total number of GPU hours of a user with the average run time and utilization characteristics and their variances. The correlation used is Spearman correlation, which performs ranked linearity correlation and is useful for detecting monotonic relationships. A correlation of plus or minus one reveals a strong positive or negative relationship, respectively, while a correlation of zero reveals no monotonic relationship. The figure shows that a high positive correlation exists between the number of jobs / GPU hours of a user and the average SM/memory utilization across jobs (note that all correlations are statistically significant: p-value <0.05). This means that expert users are likely to use GPU resources more efficiently. However, the correlation between the number of jobs / GPU hours of a user and the CoV of SM/memory utilization across jobs is quite low (≤ 0.5), especially in the case of SM utilization. *The results show that an expert user is not likely to have more predictable job behavior just because they run more jobs and consume more GPU hours. The top users are just as likely as other users to have a high variance in job characteristics.*

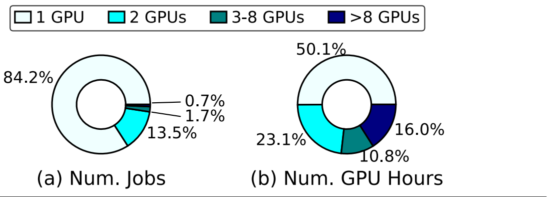


Fig. 13: (a) Fraction of jobs that run on different number of GPUs. (b) Fraction of the system's total GPU hours consumed by jobs of different sizes.

Takeaway: A typical Supercloud user submits multiple jobs with a diverse set of run time and resource utilization characteristics. While an "expert" user with many jobs and GPU hours is likely to use GPU resources more efficiently, their jobs still may have a wide variety of utilization characteristics. This makes it difficult to predict the behavior of individual users. This is an opportunity for designing new strategies to apply ML-based techniques to predict user behavior in a lightweight manner, suited for production AI-enabling supercomputers.

V. Examination of Jobs with Multi-GPU Usage Requirements

So far we have discussed the general characteristics of jobs of all sizes in terms of the number of GPUs that they run on. However, jobs with the requirement of multiple GPUs implicitly have a higher resource demand. Therefore, it is useful to separately study their behavior and GPU resource utilization characteristics.

We begin by first calculating the fraction of jobs that run on multiple GPUs. Fig. 13 shows that about 84% of all jobs run on a single GPU and about 16% run on two GPUs or more. This means that a significant number of jobs are multi-GPU (7.5k of the 47k jobs) and they consume a substantial percentage of all GPU hours consumed on the system (50%). However, only ≈2.4% of the jobs run on more than two GPUs and less than 1% of the jobs run on nine or more GPUs. This finding is consistent with what is observed on other GPU-based systems [23], [27], [61]. For example, on Microsoft's Philly clusters, 93% of the jobs are run on one GPU and only 2.5% of the jobs run on more than four GPUs [23].

While the percentage of jobs that run on multiple GPUs is 16%, when we break down by users, about 60% of the users have run at least one multi-GPU job, 13% of the users have run jobs with at least three GPUs, and 5.2% of the user users have run jobs with nine or more GPUs. Thus, the significance of multi-GPU jobs is appreciable and they should be characterized, as we do next.

Takeaway: A non-negligible fraction of the jobs on our Supercloud system are multi-GPU jobs. In addition, about 60% of the users run at least one multi-GPU job. Our results indicate that characterizing these jobs is important because of their notable footprint in terms of GPU hours consumed – more architectural research is needed to benchmark and characterize multi-GPU jobs.

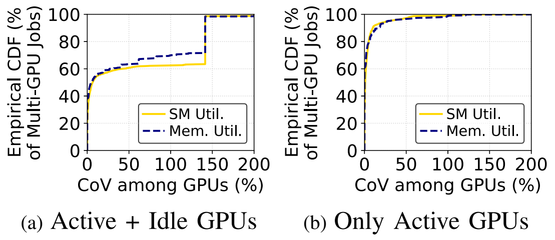


Fig. 14: (a) Variance in the utilization of SM and memory resources across all GPUs of multi-GPU jobs. (b) same as (a) except idle GPUs are removed from the variance calculation.

We now look at the characteristics of multi-GPU jobs. One may expect that requiring multiple GPUs potentially results in longer queue wait times as it may take longer for the resources to become available. However, our analysis shows that the median queue wait time of single-GPU jobs is 3 seconds, the median wait time of 2-GPU jobs is 1 second, the median wait time of 3-8 GPU jobs is 1 second, and the median wait time of >8-GPU jobs is 1 second (results not plotted). Thus, there is no significant difference. In most cases, multi-GPU jobs are scheduled quickly with a high priority and are placed as densely as possible, either on the same node or on neighboring nodes on the network interconnect. In terms of other job behavior properties, we again observe no significant difference between the run times and utilization characteristics of single-GPU jobs and multi-GPU jobs.

However, this does not necessarily mean that an individual multi-GPU job has the same utilization of SM and memory resources across its GPUs. Fig. 14(a) shows the empirical CDFs of the coefficient of variation of the SM utilization, memory utilization, and memory size utilization across all of the GPUs of a multi-GPU job (only multi-GPU jobs are included in the CDFs as single-GPU jobs would have a CoV of zero). The figure shows that about 50% of the jobs experience little to no variability among the GPUs in terms of the resource utilization of all three types of utilization. However, we observe that about 40% of the jobs experience very high CoV across all three resources. The reason for this is because these jobs have half or more of their GPUs idle, i.e., a sizable fraction of the GPUs have an average utilization of close to zero for all resources. However, if only the active GPUs of the job are considered for the CoV calculation, the CoV tends to be much lower (Fig. 14(b)). This demonstrates that the resource consumption behavior across GPUs remains uniform for multi-GPU jobs.

Takeaway: Multi-GPU jobs have similar run times and utilization characteristics as single jobs and do not experience an increase in wait times in proportion to their

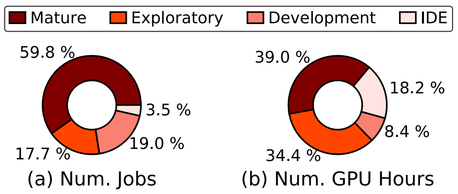


Fig. 15: (a) Distribution of mature, exploration, development, and IDE jobs. (b) System's total GPU hours consumed by different types of jobs.

sizes. While 40% of the multi-GPU jobs have idle GPUs, the resource consumption tends to be similar among active GPUs. This allows the same resource sharing technique to be deployed across all GPUs of multi-GPU jobs on Supercloud without requiring GPU-specific fine-tuning or resource utilization instrumentation but needs system-level techniques to avoid pathological cases. In our operational experience, educating and training users on using multi-GPU has been one of the key efforts at Supercloud. As the HPC community moves forward with more GPU-accelerated deep-learning workloads, user education and training for multi-GPU jobs will become one of the most critical aspects, akin to scaling CPU jobs to multiple nodes.

VI. Investigation of the Algorithm Development Life-Cycle

The last aspect of the Supercloud system that we analyze is its unique job classification in terms of different types of jobs submitted in the process of an algorithm's development. As discussed in Sec. II and shown in Fig. 2, Supercloud's users submit different types of jobs when designing their codes: IDE (design algorithm), development (determine resource requirement), exploratory (optimize algorithm / deep-learning model parameters), and mature jobs (finalize the code state). These jobs have different types of resource consumption characteristics, as we see next.

Fig. 15(a) shows a breakdown of different types of jobs along the development life-cycle. Around 60% of all jobs are mature jobs, i.e., these jobs are completed with a zero exit code. We found that 3.5% of the jobs are IDE jobs, i.e., these jobs tend to be interactive jobs that run for a long time and timeout. About 19% of the jobs are development jobs. These jobs are run while the algorithm is being developed and the code is being debugged. Lastly, about 18% of the jobs are exploratory jobs. These are the jobs that are terminated by the user before completion as they deem the jobs to be sub-optimal in the process of determining the optimal algorithm parameters (e.g., hyper-parameter tuning).

When we look at the breakdown in terms of GPU hours consumed (Fig. 15(b)), only 39% of the GPU hours are consumed by mature jobs, while 61% of all GPU hours consumed on the system are consumed by other types of jobs. About 34% of the system's GPU hours are consumed

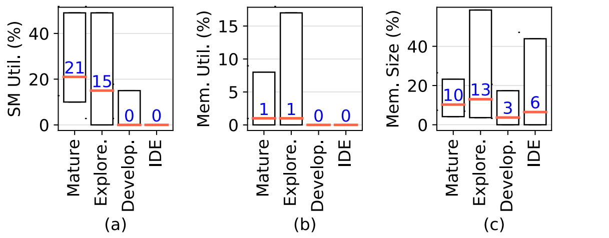


Fig. 16: Job characteristics comparison of different categories of jobs among different resources types.

by exploratory jobs, even though they only make up 18% of the jobs. This is because a median exploratory job (62 minutes) runs longer than a median mature job (36 minutes). The behavior is due to the fact that training deep learning models require many hyper-parameter-tuning jobs that get terminated by the user once they realize that the job hyper-parameters are not optimal. Users have a lower utility for these jobs as there are other hyper-parameters that enable faster/better training. This is a new trend on HPC systems that primarily run deep learning jobs.

In addition, development and IDE jobs consume ≈27% of all GPU hours. A disproportionate 18% of the GPU hours are consumed by IDE jobs (only 3% of all jobs). This is because IDE jobs run for a long time until their timeout limit (the timeout limit is 12 hours or 24 hours, depending on the requested amount) and they reserve the GPU resources during their entire run.

Takeaway: As the GPU-accelerated Supercloud system is geared toward deep learning jobs, a large portion of the jobs are exploratory as they are in the process of hyper-parameter tuning. For such workloads, users with similar characteristics like Supercloud can benefit from a multi-tier system design, with GPUs of different qualities being priced differently, so that they can run their lower-utility exploratory jobs on the slower GPUs. A considerable number of jobs on the Supercloud system are also development or IDE jobs that run until they encounter a failure or timeout. To ensure that these jobs do not lose their state, there is a growing need for architectural and system support for low-overhead checkpoint/restart mechanisms and support for fast persistent storage. More generally, architectural techniques that co-exploit hardware reliability, user-induced job termination, and model-accuracy curve are required to minimize the impact of less-reliable or fail-slow hardware.

Next, we consider the differences among the job characteristics of the different types of jobs. Understanding these differences can help system administrators identify and mitigate the effects of low resource utilization. Fig. 16 shows the box-plots (the center line shows the median and the top and bottom of the box show the 25th percentile and the 75th percentile, respectively) of mature jobs, exploratory jobs, development jobs, and IDE jobs for characteristics

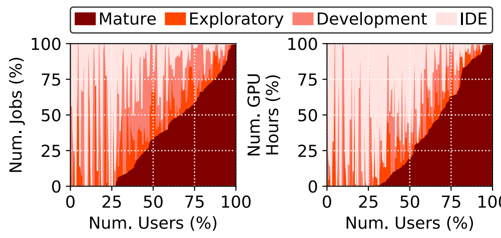


Fig. 17: (a) Distribution of users and their respective fraction of all jobs categorized by different job types. (b) Distribution of users and their respective total GPU hours consumed categorized by different job types.

such as (a) SM utilization, (b) memory utilization, and (c) memory size utilization. The figure shows that compared to mature and exploratory jobs, development and IDE jobs have a much lower resource utilization. As an instance, for SM utilization, the median SM utilization of mature jobs, exploratory jobs, development jobs, and IDE jobs is 21%, 15%, 0%, and 0%, respectively. In fact, even the 75th percentile SM utilization of IDE jobs (which make up 3.5% of all jobs, but consume 18.2% of all GPU hours) is 0%. This indicates that while development and IDE jobs run for a comparably long time and reserve considerable GPU resources, they do not use those resources effectively. *Detecting this lack of resource usage and co-scheduling other jobs that can use these resources has the potential of considerably reducing the resource-usage inefficiency caused by development and IDE jobs, even if they run for a long time,*

Takeaway: Compared to mature and exploratory jobs on Supercloud, development and IDE jobs do not efficiently consume the GPU resources that they run on. More dynamic runtime detection and reactive mitigation techniques are needed to identify such jobs while they are running. Other jobs that need the GPU resources can be co-run with them to improve system throughput and reduce the sub-optimal use of GPU resources.

Lastly, we investigate how user behavior is reflected in the breakdown of different types of jobs. While it is expected that a small fraction of the users has the most amount of resource usage corresponding to all four types of jobs, it is useful to observe if users experience high rates of exploratory, development, and IDE jobs in comparison to their rates of mature jobs. This can help identify emerging trends in HPC users, which traditionally mostly execute scientific applications and codes that have been mature for years, if not decades [24], [49]. Fig. 17(a) shows the division of the four types of jobs for different users. The x-axis shows the users normalized from 0% to 100%, sorted in the order of their fraction of all jobs that are mature. The y-axis shows the percentage of different types of jobs for each user. The figure shows that for more than 50% of the users, the percentage of all jobs that are mature is less than 40%.

Fig. 17(b) shows a similar classification of the four types of jobs for the percentage of total GPU hours consumed by the user. The x-axis shows the users normalized from 0% to 100%, sorted in the order of their fraction of GPU hours that are mature. The y-axis shows the percentage of GPU hours consumed by different types of jobs for each user. The figure shows that for more than 50% of the users, the percentage of GPU hours that are consumed by mature or exploratory jobs is less than 20%. The rest of the GPU hours are consumed by exploratory, development, or IDE jobs. In fact, for more than 25% of the users, exploratory, development, and IDE jobs constitute over 60% of all of their GPU hours. This trend reflects the changing algorithm development life-cycle employed by HPC users on machine-learning-oriented systems.

Takeaway: A large fraction of the users leverage the Supercloud system for exploratory, development, and IDE workloads. This trend reflects a paradigm shift on HPC systems as they move toward catering GPU-accelerated artificial intelligence and machine learning workloads, which are actively experimented with and evolve at a much faster rate than traditional HPC scientific codes and workloads.

VII. Related Work

In this section, we present the work related to our study. The work in this area can be classified into the following broad categories.

System Operations and Management. System operators and researchers using different systems have analyzed characteristics of their systems, including recent studies targeted toward deep learning workloads on GPU systems [3]–[5], [22], [35], [40], [49], [53], [56]. Our work adds to this collection by characterizing and demonstrating newer trends such as the observation that most GPU jobs are not likely to be adversely impacted, even under an aggressive power-cap.

Job Arrival and Scheduling. Characterizing job arrival patterns and scheduling them efficiently has been the focus of many HPC researchers [25], [31], [35], [43], [46], [49]. While HPC jobs typically have to wait in queue for long times [49], [52], we demonstrate that jobs on our system only have to wait for a few minutes as a result of our resource sharing policy.

HPC User Behavior. Some of the previous works have attempted to capture the behavior of an average HPC-system user [35], [45], [57]. For example, Schlagkamp et al. [45] observed that users often overestimate resource requirements in terms of the required number of nodes and run time. In contrast, we show that the users attempt to determine the resource requirements of their jobs actively during the development life-cycle.

Failure Characterization and Impact. Extensive research has been conducted about the characteristics and impact of CPU/GPU-failures on an HPC system [12], [14], [19], [26], [30], [32], [51], [54], [55], [59]. These studies focus on performing a long-term study of failures on a supercomputer with the goal of accurate prediction. While generally jobs that time out or have a runtime failure are considered failed jobs, our study is the first-of-its-kind to study non-failed, but non-matured jobs and demonstrate their impact on the resource consumption footprint of an HPC system.

Node Sharing and Co-Scheduling. Previous studies have explored the idea of sharing CPU or GPU nodes among multiple co-located workloads [2], [7], [8], [13], [18], [29], [37], [58], [61]. For example, Gandiva [58] and Gandiva-Fair [7] use the concept of time-sharing to swap applications in and out depending on their idle and active phases, while Gavel [29] and GSLICE [13] use space-sharing to run multiple applications simultaneously using Nvidia MPS. Our findings of active/idle GPU phases, uniform work across GPUs, varying resource utilization, and low utilization of many development/IDE jobs support the need for dynamic and reactive resource-sharing techniques to improve efficiency and throughput.

VIII. Discussion and Conclusion

In this study, we presented lessons learned from system operations and characterization and analysis of job and user behaviors on an active GPU-accelerated HPC system. In this study, we examined the resource consumption footprint of jobs using Supercloud's unique job classification and provided strategies to overcome resource under-utilization.

To the best of our knowledge, this study is the first work to classify the deep learning jobs in mature and non-mature jobs. Our study reveals that almost 60% of GPU hours are being spent on exploratory, developmental, and IDE jobs. This is the first study to show that jobs have irregular and unpredictable utilization phases. We discovered that even jobs submitted by the same user vary widely in terms of their resource consumption characteristics. This is, in contrast to, widely held beliefs in the traditional HPC world that jobs from the same user are often very similar. Hence, user-specific predictive resource management strategies may not remain effective. Our relevant framework, data, and project updates are available publicly at https://dcc.mit.edu/

Finally, we discuss various potential recommendations and areas for future investment for system architects, operators, and users.

I. System architects. Our analysis provides evidence that it is a productive strategy for system designers to focus on providing more hardware support for transparent and interference-free co-location, an already important consideration for CPUs. Second, our data suggest that GPU hardware reliability is currently not a concern (GPU reliability has improved drastically over different NVIDIA generations) — most AI-centric jobs fail or are killed by the users for other reasons (e.g., hyperparameter tuning).

High-performant GPUs, but offering different levels of reliability and price. *Our work makes a case that it might be economical for vendors to produce high performance, but potentially less resilience and error correction support, at a lower production cost and market price. Same-generation GPUs from the same vendor can have multiple versions at a different price range.*

Multi-Instance GPU (MIG) support in Nvidia GPUs is a useful step toward mitigating the low-utilization challenge via co-location. Our work provides further improvement directions. For example, resetting MIG configurations require GPUs to be idle and takes unto few seconds with user intervention, and determining the optimal configuration for a set of co-located workload requires multiple manual resetting trials and model checkpointing overhead. Hardware support for automatic re-partitioning without job interruption would be ideal, especially when all jobs belong to the same user/tenant (limited privacy concerns).

II. System operators. Our real-world data-based analysis encourages system operators to consider the benefit of multi-tier GPU-accelerated data centers. *Instead of buying only the latest-and-fastest GPUs, it might be more cost-effective to mix them with some less-expensive, less-powerful, or even less-reliable GPUs for exploratory and IDE jobs.* This recommendation is useful toward guiding our next system acquisition at our data center. This approach also increases the capacity of the data center under the same cost budget and reduces the job wait time. Secondly, the system operator can leverage the low-resource utilization based on the job category to incentivize users for co-location, using coupon-based incentives or other mechanisms [34].

III. Users. Users should recognize the unique resource-consumption aspects of deep learning jobs — developmental and exploratory jobs may not consume the same amount of GPU resources, or even benefit from high resource allocation (or faster GPUs).

Different GPU types for exploratory and matures jobs, and incentive for co-location. If there is an option, users can use cheaper and less powerful GPUs for exploratory jobs and employ faster and more expensive GPUs selectively for mature jobs. Similarly, based on our data, users have an incentive to opt for co-location for their exploratory jobs.

IX. Acknowledgement

Research was sponsored by the United States Air Force Research Laboratory and the United States Air Force Artificial Intelligence Accelerator and was accomplished under Cooperative Agreement Number FA8750-19-2-1000. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the United States Air Force or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, 2016, pp. 265–283.

[2] O. Alipourfard, H. H. Liu, J. Chen, S. Venkataraman, M. Yu, and M. Zhang, "Cherrypick: Adaptively Unearthing the Best Cloud Configurations for Big Data Analytics," in *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)*, 2017, pp. 469–482.

[3] W. Allcock, P. Rich, Y. Fan, and Z. Lan, "Experience and Practice of Batch Scheduling on Leadership Supercomputers at Argonne," in *Workshop on Job Scheduling Strategies for Parallel Processing*. Springer, 2017, pp. 1–24.

[4] G. P. R. Alvarez, P.-O. Östberg, E. Elmroth, K. Antypas, R. Gerber, and L. Ramakrishnan, "Towards Understanding Job Heterogeneity in HPC: A NERSC Case Study," in *2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*. IEEE, 2016, pp. 521–526.

[5] N. Attig, P. Gibbon, and T. Lippert, "Trends in Supercomputing: The European Path to Exascale," *Computer Physics Communications*, vol. 182, no. 9, pp. 2041–2046, 2011.

[6] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," 2020.

[7] S. Chaudhary, R. Ramjee, M. Sivathanu, N. Kwatra, and S. Viswanatha, "Balancing Efficiency and Fairness in Heterogeneous GPU Clusters for Deep Learning," in *Proceedings of the Fifteenth European Conference on Computer Systems*, 2020, pp. 1–16.

[8] S. Chen, C. Delimitrou, and J. F. Martínez, "Parties: Qos-Aware Resource Partitioning for Multiple Interactive Services," in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2019, pp. 107–120.

[9] J. Choquette, W. Gandhi, O. Giroux, N. Stam, and R. Krashinsky, "NVIDIA A100 Tensor Core GPU: Performance and Innovation," *IEEE Micro*, 2021.

[10] N. Corporation, "Nvidia-smi," 2016. [Online]. Available: http://developer.download.nvidia.com/compute/DCGM/docs/nvidia-smi-367.38.pdf

[11] N. Corporation, "Dcgm," 2021. [Online]. Available: https://developer.nvidia.com/dcgm

[12] D. Das, M. Schiewe, E. Brighton, M. Fuller, T. Cerny, M. Bures, K. Frajtak, D. Shin, and P. Tisnovsky, "Failure Prediction by Utilizing Log Analysis: A Systematic Mapping Study," in *Proceedings of the International Conference on Research in Adaptive and Convergent Systems*, 2020, pp. 188–195.

[13] A. Dhakal, S. G. Kulkarni, and K. Ramakrishnan, "GSLICE: Controlled Spatial Sharing of GPUs for a Scalable Inference Platform," in *Proceedings of the 11th ACM Symposium on Cloud Computing*, 2020, pp. 492–506.

[14] C. Di Martino, Z. Kalbarczyk, R. K. Iyer, F. Baccanico, J. Fullop, and W. Kramer, "Lessons Learned from the Analysis of System Failures at Petascale: The Case of Blue Waters," in *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. IEEE, 2014, pp. 610–621.

[15] B. Dutta, V. Adhinarayanan, and W.-c. Feng, "GPU Power Prediction via Ensemble Machine Learning for DVFS Space Exploration," in *Proceedings of the 15th ACM International Conference on Computing Frontiers*, 2018, pp. 240–243.

[16] W. Falcon and .al, "Pytorch lightning," *GitHub. Note: https://github.com/PyTorchLightning/pytorch-lightning*, vol. 3, 2019.

[17] M. Feldman. (2018) Top500. [Online]. Available: https://www.top500.org/news/new-gpu-accelerated-supercomputers-change-the-balance-of-power-on-the-top500/

[18] J. Gu, M. Chowdhury, K. G. Shin, Y. Zhu, M. Jeon, J. Qian, H. Liu, and C. Guo, "Tiresias: A {GPU} Cluster Manager for Distributed Deep Learning," in *16th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 19)*, 2019, pp. 485–500.

[19] S. Gupta, T. Patel, C. Engelmann, and D. Tiwari, "Failures in Large Scale Systems: Long-Term Measurement, Analysis, and Implications," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2017, pp. 1–12.

[20] J. Han, M. M. Rafique, L. Xu, A. R. Butt, S.-H. Lim, and S. S. Vazhkudai, "Marble: A Multi-GPU Aware Job Scheduler for Deep Learning on HPC Systems," in *2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*. IEEE, 2020, pp. 272–281.

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[22] Q. Hu, P. Sun, S. Yan, Y. Wen, and T. Zhang, "Characterization and prediction of deep learning workloads in large-scale gpu datacenters," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2021, pp. 1–15.

[23] M. Jeon, S. Venkataraman, A. Phanishayee, J. Qian, W. Xiao, and F. Yang, "Analysis of Large-Scale Multi-Tenant {GPU} Clusters for {DNN} Training Workloads," in *2019 {USENIX} Annual Technical Conference ({USENIX}{ATC} 19)*, 2019, pp. 947–960.

[24] M. D. Jones, J. P. White, M. Innus, R. L. DeLeon, N. Simakov, J. T. Palmer, S. M. Gallo, T. R. Furlani, M. Showerman, R. Brunner *et al.*, "Workload Analysis of Blue Waters," *arXiv preprint arXiv:1703.00924*, 2017.

[25] W. Joubert and S.-Q. Su, "An Analysis of Computational Workloads for the ORNL Jaguar System," in *Proceedings of the 26th ACM international conference on Supercomputing*. ACM, 2012, pp. 247–256.

[26] K. Lee, M. B. Sullivan, S. K. S. Hari, T. Tsai, S. W. Keckler, and M. Erez, "On the Trend of Resilience for GPU-Dense Systems," in *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks–Supplemental Volume (DSN-S)*. IEEE, 2019, pp. 29–34.

[27] Y. Luan, X. Chen, H. Zhao, Z. Yang, and Y. Dai, "SCHED^2: Scheduling Deep Learning Training via Deep Reinforcement Learning," in *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2019, pp. 1–7.

[28] F. Mantovani and E. Calore, "Performance and Power Analysis of HPC Workloads on Heterogeneous Multi-Node Clusters," *Journal of Low Power Electronics and Applications*, vol. 8, no. 2, p. 13, 2018.

[29] D. Narayanan, K. Santhanam, F. Kazhamiaka, A. Phanishayee, and M. Zaharia, "Heterogeneity-Aware Cluster Scheduling Policies for Deep Learning Workloads," in *14th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 20)*, 2020, pp. 481–498.

[30] B. Nie, J. Xue, S. Gupta, T. Patel, C. Engelmann, E. Smirni, and D. Tiwari, "Machine Learning Models for GPU Error Prediction in a Large Scale HPC System," in *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2018, pp. 95–106.

[31] D. Nurmi, J. Brevik, and R. Wolski, "QBETS: Queue Bounds Estimation from Time Series," in *Workshop on Job Scheduling Strategies for Parallel Processing*. Springer, 2007, pp. 76–101.

[32] G. Ostrouchov, D. Maxwell, R. Ashraf, C. Engelmann, M. Shankar, and J. Rogers, "GPU Lifetimes on Titan Supercomputer: Survival Analysis and Reliability," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC) 2020*, 2020, pp. 15–20.

[33] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[34] T. Patel, R. Garg, and D. Tiwari, "{GIFT}: A coupon based throttle-and-reward mechanism for fair and efficient i/o bandwidth management on parallel storage systems," in *18th {USENIX} Conference on File and Storage Technologies ({FAST} 20)*, 2020, pp. 103–119.

[35] T. Patel, Z. Liu, R. Kettimuthu, P. Rich, W. Allcock, and D. Tiwari, "Job Characteristics on Large-Scale Systems: Long-Term Analysis, Quantification and Implications," in *2020 SC20: International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*. IEEE Computer Society, 2020, pp. 1186–1202.

[36] T. Patel and D. Tiwari, "PERQ: Fair and Efficient Power Management of Power-Constrained Large-Scale Computing Systems," in *Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing*, 2019, pp. 171–182.

[37] T. Patel and D. Tiwari, "Clite: Efficient and QoS-Aware Co-Location of Multiple Latency-Critical Jobs for Warehouse Scale Computers," in *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2020, pp. 193–206.

[38] T. Patel, A. Wagenhäuser, C. Eibel, T. Hönig, T. Zeiser, and D. Tiwari, "What does Power Consumption Behavior of HPC Jobs Reveal?: Demystifying, Quantifying, and Predicting Power Consumption Characteristics," in *2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 2020, pp. 799–809.

[39] T. Patki, Z. Frye, H. Bhatia, F. Di Natale, J. Glosli, H. Ingolfsson, and B. Rountree, "Comparing GPU Power and Frequency Capping: A Case Study with the MuMMI Workflow," in *2019 IEEE/ACM Workflows in Support of Large-Scale Science (WORKS)*. IEEE, 2019, pp. 31–39.

[40] A. Pecchia, D. Cotroneo, Z. Kalbarczyk, and R. K. Iyer, "Improving Log-Based Field Failure Data Analysis of Multi-Node Computing Systems," in *2011 IEEE/IFIP 41st International Conference on Dependable Systems & Networks (DSN)*. IEEE, 2011, pp. 97–108.

[41] A. Reuther, J. Kepner, C. Byun, S. Samsi, W. Arcand, D. Bestor, B. Bergeron, V. Gadepally, M. Houle, M. Hubbell *et al.*, "Interactive supercomputing on 40,000 cores for machine learning and data analysis," in *2018 IEEE High Performance extreme Computing Conference (HPEC)*. IEEE, 2018, pp. 1–6.

[42] A. Reuther, P. Michaleas, M. Jones, V. Gadepally, S. Samsi, and J. Kepner, "Survey of machine learning accelerators," in *2020 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, 2020, pp. 1–12.

[43] G. P. Rodrigo, P.-O. Östberg, E. Elmroth, K. Antypas, R. Gerber, and L. Ramakrishnan, "Towards Understanding HPC Users and Systems: A NERSC Case Study," *Journal of Parallel and Distributed Computing*, vol. 111, pp. 206–221, 2018.

[44] S. Samsi, M. L. Weiss, D. Bestor, B. Li, M. Jones, A. Reuther, D. Edelman, W. Arcand, C. Byun, J. Holodnack *et al.*, "The mit supercloud dataset," in *2021 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, 2021, pp. 1–8.

[45] S. Schlagkamp, R. F. da Silva, J. Renker, and G. Rinkenauer, "Analyzing Users in Parallel Computing: A User-Oriented Study," in *2016 International Conference on High Performance Computing & Simulation (HPCS)*. IEEE, 2016, pp. 395–402.

[46] S. Schlagkamp, R. Ferreira da Silva, W. Allcock, E. Deelman, and U. Schwiegelshohn, "Consecutive Job Submission Behavior at Mira Supercomputer," in *Proceedings of the 25th ACM International Symposium on High-Performance Parallel and Distributed Computing*, 2016, pp. 93–96.

[47] A. Sergeev and M. D. Balso, "Horovod: fast and easy distributed deep learning in TensorFlow," *arXiv preprint arXiv:1802.05799*, 2018.

[48] S. Shams, R. Platania, K. Lee, and S.-J. Park, "Evaluation of Deep Learning Frameworks over Different HPC Architectures," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2017, pp. 1389–1396.

[49] N. A. Simakov, J. P. White, R. L. DeLeon, S. M. Gallo, M. D. Jones, J. T. Palmer, B. Plessinger, and T. R. Furlani, "A Workload Analysis of NSF's Innovative HPC Resources Using XDMoD," *arXiv preprint arXiv:1801.04306*, 2018.

[50] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.

[51] M. Snir, R. W. Wisniewski, J. A. Abraham, S. V. Adve, S. Bagchi, P. Balaji, J. Belak, P. Bose, F. Cappello, B. Carlson *et al.*, "Addressing Failures in Exascale Computing," *The International Journal of High Performance Computing Applications*, vol. 28, no. 2, pp. 129–173, 2014.

[52] D. Stanzione, J. West, R. T. Evans, T. Minyard, O. Ghattas, and D. K. Panda, "Frontera: The Evolution of Leadership Computing at the National Science Foundation," in *Practice and Experience in Advanced Research Computing*, 2020, pp. 106–111.

[53] V. V. Stegailov and H. E. Norman, "Challenges to the Supercomputer Development in Russia: a HPC User Perspective," *Program Systems: Theory and Applications*, vol. 5, no. 1, pp. 111–152, 2014.

[54] D. Tiwari, S. Gupta, J. Rogers, D. Maxwell, P. Rech, S. Vazhkudai, D. Oliveira, D. Londo, N. DeBardeleben, P. Navaux, L. Carro, and A. Bland, "Understanding GPU Errors on Large-Scale HPC Systems and the Implications for System Design and Operation," in *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*, 2015, pp. 331–342.

[55] D. Tiwari, S. Gupta, G. Gallarno, J. Rogers, and D. Maxwell, "Reliability Lessons Learned from GPU Experience with the Titan Supercomputer at Oak Ridge Leadership Computing Facility," in *SC'15: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2015, pp. 1–12.

[56] R. van Zon, M. Ponce, E. Spence, and D. Gruner, "Trends in Demand, Growth, and Breadth in Scientific Computing Training Delivered by a High-Performance Computing Center," *arXiv preprint arXiv:1901.05520*, 2019.

[57] N. Wolter, M. O. McCracken, A. Snavely, L. Hochstein, T. Nakamura, and V. Basili, "What's Working in HPC: Investigating HPC User Behavior and Productivity," *CTWatch Quarterly*, vol. 2, no. 4A, pp. 9–17, 2006.

[58] W. Xiao, R. Bhardwaj, R. Ramjee, M. Sivathanu, N. Kwatra, Z. Han, P. Patel, X. Peng, H. Zhao, Q. Zhang *et al.*, "Gandiva: Introspective Cluster Scheduling for Deep Learning," in *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)*, 2018, pp. 595–610.

[59] L. Yang, B. Nie, A. Jog, and E. Smirni, "Practical Resilience Analysis of GPGPU Applications in the Presence of Single-and Multi-bit Faults," *IEEE Transactions on Computers*, 2020.

[60] A. B. Yoo, M. A. Jette, and M. Grondona, "Slurm: Simple linux utility for resource management," in *Workshop on job scheduling strategies for parallel processing*. Springer, 2003, pp. 44–60.

[61] H. Zhao, Z. Han, Z. Yang, Q. Zhang, F. Yang, L. Zhou, M. Yang, F. C. Lau, Y. Wang, Y. Xiong *et al.*, "HiveD: Sharing a {GPU} Cluster for Deep Learning with Guarantees," in *14th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 20)*, 2020, pp. 515–532.