GPU Utilization & Resource Allocation in Heterogeneous Clusters

Matthew Harper
University of Houston Victoria

Abstract

As big data processing systems and deep learning models have become increasingly significant and applied to many real-world applications, graphical processing unit (GPU) clusters with in-memory processing have become the primary processor choice for powerful high performance computing systems featuring in-memory capabilities for deep networks [1,13]. While these in-memory processing capabilities of GPUs exhibit powerful processing advantages when applied to deep model processing including speeds of up to 100x faster than batch approaches, they present challenges in resource management as a result of their lack of intra-processor sharing ability and associated specifications for resource management and schedulers. These challenges leave many cloud computing providers with distributed GPU cluster architectures which exhibit high rates of under-utilization and discretized periods of start and stop processing [1,3,12,13]. Common approaches to enhance GPU utilization and processing performance such as fairness and over-subscribing among others to counter GPU cluster hardware resource interference and the contribution to reduced performance and resulting under-utilization. However, these resource allocation optimization approaches each present less than ideal optimized solutions for generalized heterogenous GPU clusters [1, 12]. Moreover, for specific applications one or more of these pragmatic approaches may work well for the intended application. However, in today's modern generalized data centers and cloud computing distributed clusters, a more generalized approach is often the more highly desired outcome for holistic performance perspectives [1]. This work presents an overview of previous works and presents a case for use of predictive algorithms in dynamic resource allocation in heterogeneous GPU clusters to achieve enhanced utilization and discussed known issues relatable to resource allocation infrastructures within GPU clusters and proposes a multilayered deep network predictive algorithm for use in dynamic resource allocation in heterogeneous GPU clusters to achieve enhanced utilization.

1 Introduction

According to IBM, 90% of the world's data has been created in the last two years and evidence suggests data creation is not slowing but increasing exponentially. Moreover, the increase in data creation and big data processing systems requirements of high performance in-memory computing, has led to the development and use of advanced distributed computing techniques such as distributed-parallel computing, cloud computing, clustering, and high-performance computing integrated with heterogeneous computing infrastructures consisting of layered virtualization over multiple variations of concurrent CPU and GPU environments [33].

In order to process big data at the rates and volumes mentioned, not only have new techniques been developed such as Spark, Hadoop, and MapReduce, but advances in distributed cloud computing and clustering have led to a hyperbolic expansion of data centers around the globe. Combined with parallel computing architectures of GPUs, cloud computing infrastructures have contributed to accelerated training and inference often expected of deep model networks. This robust parallelization for tensor processing and neural networks presents GPU clusters as a preferred processing solution for many supervised and unsupervised AI/ML tasks. A common framework for distributed clusters consists of varying models of GPUs and CPUs to form heterogeneous GPU clusters. This offers flexible allocation and utilization of a diverse range of tasks and offers flexible system performance and scalability of existing systems and infrastructure. However, this new challenges are presented in heterogeneous GPU clusters such as scheduling resources, workload balancing, and under-utilization leading to decreased performance, suboptimal efficiency, and reduced realized availability with long queuing latency. Moreover, an increase in power usage for GPU clusters is realized adding unnecessary cost for both the cloud service provider and customer [12].

Inefficient GPU utilization in cloud clusters is becoming a significant concern and is leading to dangerous levels of excessively high energy consumption [6]. Despite an exponential demand for cloud computing service data centers typically operate large percentages of CPU and GPU processors with low utilization. Moreover, with utilization rates of 30% typical of GPU clusters, a significant ratio of workload assigned GPU cluster nodes remain in perpetual states of idleness and accounting for roughly 70% of the peak power demand energy consumption [6].

GPUs have gained notoriety and wide acceptance as a general-purpose processor due to their highly dense parallel computing capabilities largely a result of thread level parallelism for stream multi-processing (SM) [9]. Multiple tasks are then concurrently processed within GPUs to improve resource utilization. However, this presents challenges for resource allocation when the computing resources are intra-connected to heterogeneous clusters with varying hardware architectures and resource competition between concurrent work loading. Defining the dynamic computing resource requirements for each tasks can be a non-trivial exercise particularly for memory-intensive and compute-intensive resource requirements.

In addition to GPU cluster under-utilization attributed to multiprocessing of concurrent tasks, GPU utilization can be reduced as a result of distribution across servers referred to as intra-server interference which adds to overall latency in computational latency with extended queuing times.

This work presents an overview of known issues relatable to resource allocation infrastructures within GPU clusters and proposes a multilayered deep network predictive algorithm for use in dynamic resource allocation in heterogeneous GPU clusters to achieve enhanced utilization.

While there are numerous works on the phenomenon of oversubscription of cloud resources based on memory, CPU, network bandwidth, and power related issues, this work analyzes current trends in work loading and resource allocation associated with GPU utilization and the need for machine learning based dynamic subscription and allocation versus the under an over subscription as a result of complex scheduling of virtualized GPUs in the cloud.

2 Background

Demand for deep learning capable racks is driving growth with applications involving GPUs creating highly dense distributed clusters increasing rack power requirements from 10-14kW for existing data centers to 40-60kW for racks featuring GPU processors. This translates into data centers being an energy-intensive building types, consuming 10 to 50 times the energy per floor space of a typical commercial office building. Moreover, data centers can account for approximately 2% of the total U.S. electricity use. Not only has electrical power requirements increased but so has cloud computing demand. As this demand has increased, availability of existing resource allocation is extremely low as the table below provides some of the existing capacities of popular data centers by region [11].

Multi-tenant Distributed Cloud Availability		
Region	In-Use	[1]Availability
Bay Area, Ca	99.5%	.5%
Pheonix, Az	96.2%	3.8%
Ft. Worth, Tx	98.1%	1.9%
Northern Virginia	99.8%	.2%

1. Availability is assumed due to over subscription of GPU allocation.

To meet this increased demand, two methods have been utilized which include the integration of existing GPU clusters with new GPU clusters to form heterogeneous clusters and a resource allocation technique of over-subscribing based on trended load demands.

Moreover, increases in data creation and big data processing systems requirements of high performance in-memory computing, has led to the development and use of advanced distributed computing techniques such as distributed-parallel computing, cloud computing, clustering, and high-performance computing integrated with computing infrastructures consisting of layered virtualization over multiple variations of concurrent CPU and GPU environments known as heterogeneous clusters.

Heterogeneous GPU clusters can offer advantages of flexible allocation to meet user demand for different GPU types. However, heterogeneous clustered infrastructures create challenges associated with this scheduling resources in over diverse processor clusters, with the most notable being lower GPU utilization among others [12].

Previous research has analyzed low GPU utilization and presented a solution of fine-grained GPU sharing techniques [12]. However, the use of GPU sharing faces two problems, on the one hand some tasks require a QoS with complete availability of unrivaled access to a full GPU processing without multi-tenant concurrent computation [12]. Another issue with this approach is that common resource allocation containers such as Kubernetes are designed for coarse-grained GPU allocation and do not provide independent support for fine-grained allocation [12, 23, 24, 25, 26].

In addition, highly distributed clusters increase the risk of poor availability as a result of cluster interference attributed to resource allocation sharing in terms of number of GPUs assigned to each virtual cluster. GPUs assigned to train models for deep neural networks are required to have high availability for simultaneous in-memory processing required of modern deep learning frameworks [9, 17]. This quality of service requires the resource allocation scheduler to perform gang scheduling with efficient use of available resources within an RDMA domain. As a result, a significant loss in resource allocation efficiency is experienced particularly with dynamic workloads that change required in-memory compute requirements often at the expense of increased queuing time due to available virtualized resources [17].

GPU utilization across multiple jobs is typically low due to distribution across dedicated servers and intra-server interference. Dedicated servers exhibit a high level of standby processing time particularly within distributed architectures. Moreover, there is a noticeable trade-off in performance with locality and queuing delay. Scheduling within distributed architectures increase intra-server locality issues while offering high communication efficiency due to local processing, this adds to increased computational latency with long queuing times.

However, highly distributed clusters are able to counter the effects of intra-server locality and reduce computational latency with associated peripheral delays in queueing time but with the added contention of lowered network efficiency. Availability and throughput issues with intra-server communication can be enhanced with the use of advanced instar-server interconnects including the use of peripheral component interconnect express (PCIe) or with an NVLink as examples.

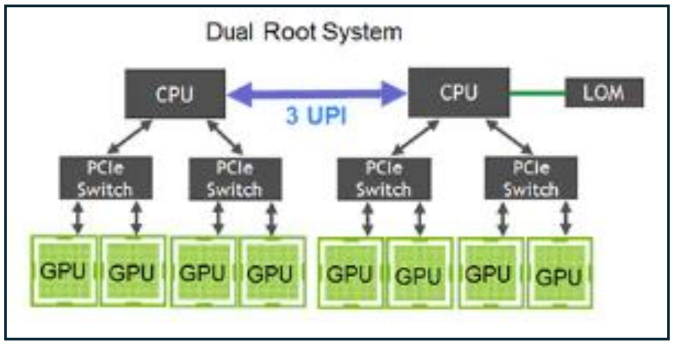


Figure 1: Distributed GPU cluster interconnected PCIe [20].

PCIe connections are common high-speed serial computer intra-connections based on the expansion bus standard with a maximum speed of 242 GB/s. NVLink is a new protocol that reduces communication latency between GPUs within a server illustrated in figure 2.

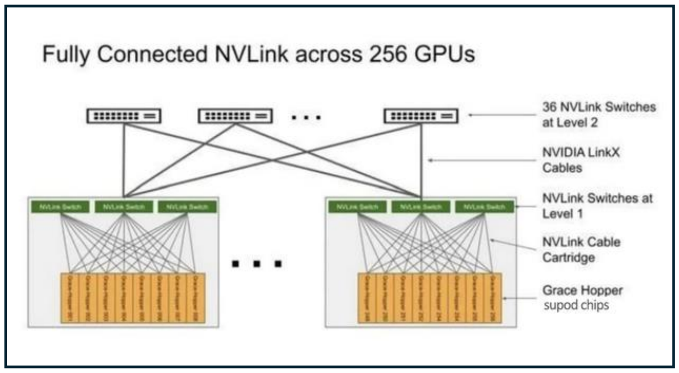

Fully Connected NVLink across 256 GPUs
36 NVLink Switches at Level 2
NVIDIA LinkX Cables
NVLink Switches at Level 1
NVLink Cable Cartridge
Grace Hopper supod chips

Figure 2: Distributed GPU cluster interconnected NVLink [21].

However, intra-communications within highly distributed clusters can be improved with the use of high-bandwidth network links such as remote direct memory access (RDMA) interconnects which enable the direct exchange of data in main memory without relying on the processor. *RDMA* techniques such as NVLink are becoming widely used in high performance computing (HPC) systems and offer kernel free intra-server high-bandwidth and low latency messaging [19].

GPUs have largely gained wide adoption as a general-purpose processor due to their ability for tensor computation with thread level parallelism for stream multi-processing (SM) [9]. Moreover, unique ability makes GPUs well-suited for deep learning computation due to their ability to perform iterative matrix multiplication, utilizing thousands of processing units, where a group of processing units is referred to as a streaming multi-processor (SM) [9]. With streaming multiprocessing, each kernel defines the number of threads required, which are called thread blocks. GPU clusters can consist of multiple streaming multi-processors (SM), memory caches, and interconnection networks including PCIe and NVLink mentioned above. However, one of the most common methods for multi-tasking is spatial multi-tasking (SPK) which divides GPUs into a multi-processor grids for kernel mapping [5].

Where CPUs share kernel states for processing, GPUs are able to perform processing of tensors without the need to share kernel states. As a result, (SPK) is one of the simplest forms of GPU resource allocation techniques for multi-tasking. Though this method offers improved performance gain it exhibits under-utilization as the figure below illustrates a common issue with GPU processor resource allocation [1, 3, 5, 9].

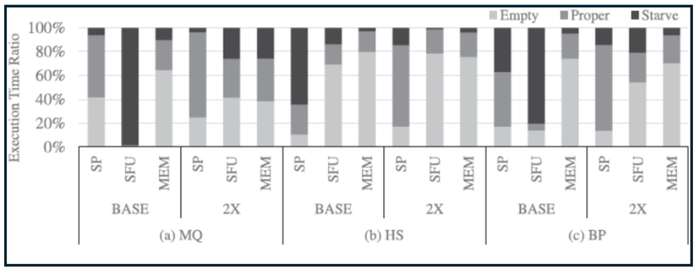


Figure 3: GPU execution stages of empty, proper, and starve [5].

3 Resource Allocation

Due to interconnection latency, concurrent work loading and over subscribing, and issues associated with under-utilization in GPU clusters illustrated in figure 3, resource allocation for virtual machines within cloud based virtualized systems is of paramount importance for efficient use of computational resources. Virtualized system provide many scheduling algorithms for resource allocation such as simple and minimal time scheduling algorithms, and the credit scheduling algorithm in XEN virtualized systems [8]. Previous works have also presented allocation algorithms based on dominant share, fairness, and adaptive characteristics which are discussed in section 4. In XEN virtualization management systems, when a physical CPU is on non-idle state, the scheduler looks in the workload que for tasks waiting to be executed. This approach minimizes realized CPU idle time with an objective of system load balancing on multi-processors and achieving high utilization of system resources [8].

The figure below illustrates an example architecture of resource allocation system, which consists of virtual machine, resource sensor, request rate sensor, resource mapping table and QoS goal.

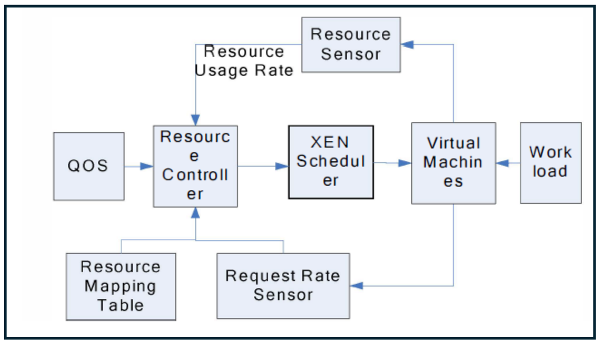


Figure 4: Resource allocation model example [8].

Resource utilization of the virtual machine changed because of time-varying workload in the virtual machine. After receiving the current resource utilization and request rate, the controller will search the target utilization interval by the mapping table. Then the controller will determine whether to reallocate resource in the next control interval. If the utilization is outside the target interval, the resources will be reallocated for the virtual machine. Then the controller will calculate the amount of required resource allocation for individual virtual machine clusters by the utilization control model [8].

However, heterogeneous cluster infrastructure integrated with both CPU and GPU processors complicates the resource allocation process. In fact, to calculate the required GPU utilization and resource allocation is typically considered non-trivial pursuit, particularly for dynamic adaptive allocation over distributed interconnected processors which often have both PCIe and NVLink protocols of varying network bandwidth and associated QoS. To assist with the allocation process, it is desirable to maximize the throughput of batch tasks while meeting the QoS of latency-sensitive (LS) tasks [7]. For additional categorization of resource allocation priority, GPU tasks can also be labeled as memory intensive (MI) tasks or compute intensive (CI) tasks based on the performance and latency specifications. However, the computing characteristics of the tasks are not only related to the hardware architecture, but also the resource contention between co-running tasks. As a result, runtime systems are required for dynamically determining resource allocation ratios of tasks [7].

Task consolidation on GPUs has received wide attention from both industry and academia. As a result, orchestration tools known as containers serve as abstraction architectures that supports concurrent execution of multiple kernels on a single or multiple GPUs with multiple independent queues.

Kubernetes is an orchestration tool to easily deploy
Containers such as Kubernetes are deployed on various nodes within a distributed GPU cluster. Originally developed by Google and eventually made open source, Kubernetes, assist resource allocation by creating forecasts used to determine how many tasks are attempting to reserve resource nodes and the required processing needed to meet that demand. Reserving too many nodes will result in oversubscribing of tasks and increased latency, while reserving fewer than the required nodes decreases GPU utilization seen in figure 3. Kubernetes can automatically spin up nodes and deploy applications to meet allocation demands. Once the demand drops, the idle containers are torn down to free up resources for other tasks workloads [31].

As previously stated, Kubernetes is designed for coarse-grained GPU allocation and does not provide independent support for fine-grained allocation [12, 23, 24, 25, 26]. As a result, Slurm is utilized as a fine-grained workload manager that supports distributed jobs and is the most popular scheduler for batch oriented HPC workloads [31].

Workload jobs are submitted to the Slurm manager, which automatically cues and allocates resources once available. Slurm offers enhanced scalability and can scale from small clusters of a few nodes to massive clusters with hundreds or more systems [31]. By allowing users to submit their jobs to the cluster manager, multiple different projects applications can be run on a cluster concurrently for maximum efficiency making this a great application for resource allocation of virtualize machines on distributed heterogeneous GPU clusters [3, 9, 31].

By complimenting Kubernetes with Slurm, systems can manage and allocate resources for multiple concurrent workloads. Moreover, Slurm can manage the allocation of computing nodes and GPUs, ensuring that these resources are available while Kubernetes can manage the deployment and scaling of machine learning workloads, ensuring that these workloads are highly available and can handle large amounts SM traffic across GPU clusters [22, 31].

4 Previous Works

There are many existing cloud systems which profile workload to improve resource allocation efficiency and GPU utilization by obtaining metrics such as training progress [9, 58], workload communication distribution [9, 15, 16] and workload execution time [8, 9]. An additional method for resource allocation optimization is to conduct online workload profiling on isolated machines for co-location strategies to harvest under-utilized resources [9]. However, heterogeneous GPU clusters have proven challenging for these methods which have increased GPU utilization overall but fail to account for diverse quality of service requirements in multi-tenant GPU processing which often present significant non-trivial obstacles.

As processing workloads continue to their migration into heterogeneous cloud cluster processing resources utilizing both CPUs and GPUs and often present different resource requirements for tasks requesting a variety of resources for execution for generalized cloud clusters.

Recent works have presented the idea of fairness in resource allocation based on demand profiling on the idea of *Dominant Share* which refers to the ratio of required resource to account for the maximum proportion of resources for a given workload requests. This can be represented as a ratio of the share of resources Yi = Ri/Ci if Ci is the amount of available resource for allocation and Ri is the amount of resource allocation requested by a workload task [3]. Formally this can be expressed in equation 1 as [3],

$$\mu = \max\left\{\frac{R_i}{C_i}\right\}$$

A fairness algorithm for resource allocation can then be formally presented and derived based on dominant shares equation as [3],

$$f_{\beta,\lambda}^{FDS} = sgn(1-\beta)\left(\sum_{j=1}^{n}\left(\frac{\mu_j x_j}{\sum_{k=1}^{n}\mu_k x_k}\right)^{1-\beta}\right)^{\frac{1}{\beta}}\left(\sum_{j=1}^{n}\mu_j x_j\right)^{\lambda},$$

However, difficulties can emerge from allocating resources for applications presenting heterogeneous resource requirements, typically CPU and GPU computing resources. One difficulty is how to quantify the resource usage of every workload and the fairness and efficiency of an allocation scheme of heterogeneous resources [3].

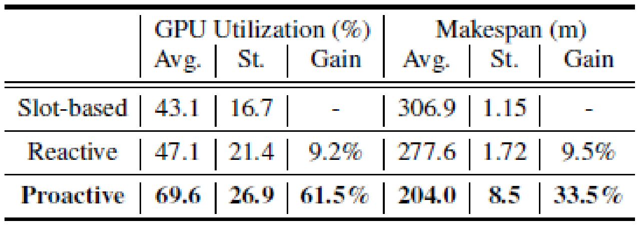
Another issue that presents with heterogeneous clusters and workload resource allocation requirements is that generalized cloud clusters are required to process workloads which are dissimilar, non-interchangeable, and often times are not proportional.

Oversubscribing is a common method utilized by cloud service providers (CSPs) typically which offer cloud systems architectures based on virtualization technologies which are widely dependent on resource allocation, scheduling, and sharing of processing computation. Most often, data centers consisting of heterogeneous clusters of resources including CPUs, GPUs, disk and memory usage, and cloud services based on the use of virtual machines. As a result of many user subscribers with minimal loading of available cluster GPUs, CSPs were accustomed to oversubscribing allocated resources based on minimal loading.

Initially, this worked well enough for most application, however, now there appears to be an increasing trend in not only the under-utilized usage in GPU processing, but a rapid escalation in the overall demand for cloud computing and GPU processing resources in general which have and continue to be integrated into many commercial applications.

This increase in demand is now a primary motivation of research into methods and techniques to enhance overall GPU utilization, reduce or slow the need for data center expansion by utilizing existing resources, and the prevention of oversubscription as cloud service providers.

Another notable work by Yeung et.al. [9] presents promising results for the use of machine learning based prediction for GPU resource allocation to increase utilization without the use of online profiling. Existing resource efficient decisions for DL in the cloud made by providers and consumers are performed by understanding and exploiting workload metrics. However, they can only do so by performing online profiling.



	GPU Utilization (%)			Makespan (m)		
	Avg.	St.	Gain	Avg.	St.	Gain
Slot-based	43.1	16.7	-	306.9	1.15	-
Reactive	47.1	21.4	9.2%	277.6	1.72	9.5%
Proactive	**69.6**	**26.9**	**61.5%**	**204.0**	**8.5**	**33.5%**

Figure 5. *Performance comparison of Deep Learning schedulers* [9].

Based on the tested results in figure 5, Yeung et.al. [9] provided a prediction algorithm which was successfully used to achieve an overall cluster GPU utilization of 69.6% in comparison to slot-based (43.1%) and conventional reactive (47.1%) approaches using online profiling. We propose a similar method but utilize a deep network based on a multi-layered feedback modeled deep network [9].

4 Dynamic Predictive Resource Allocation

Heterogeneous GPU clusters have proven challenging for traditional resource allocation methods which have increased GPU utilization overall but fail to consistently account for diverse quality of service requirements in multi-tenant GPU processing which often present significant non-trivial obstacles. Here we present a prediction-based method for the dynamic resource allocation of heterogenous clusters based on feedforward multilayered deep models.

Multi-layered deep networks, also known as deep feedforward networks, are considered basic fundamental examples of deep networks. These networks are neural networks that often contain numerous layers of artificial neurons known as input nodes or hidden nodes which process tensors as their inputs. The characteristic nature of theses network elements remain similar throughout artificial neural networks variations including the use of affine transformations, weight functions, and transfer and activation functions [3].

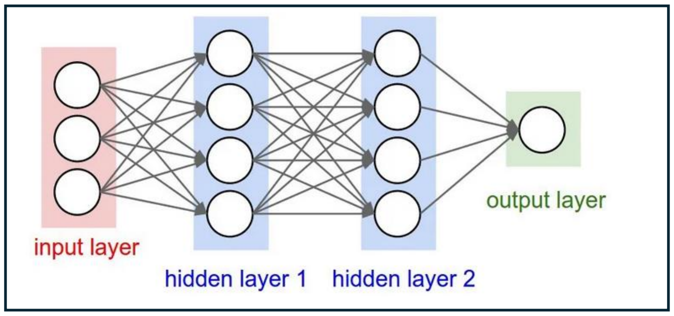


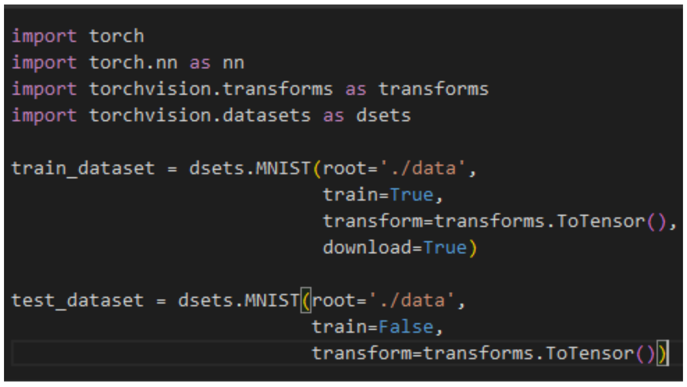
Figure6. *Multi-layered feedback deep network with 2 hidden layers* [31].

These models are often called feedforward networks based on their use and flow of information that goes through the network models. However, models that utilized feedback connections in which data flows back to the model are called recurrent neural networks or RNNs.

Weights and biases are utilized in the hidden layers to categorize cloud requested job tasks. In addition, prior to model input, the data set and memory dimension is evaluated as it iterates through the model and calculates the floating operations per second (FLOPs) for each operation based on inputs, output shape, and parameters [9, 30]. An example of this for a 2-dimensional matrix would be as follows,

Input Shape x Output Shape x Batch Size [9]

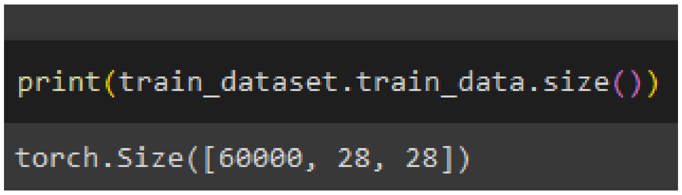
An example on how to perform this with PyTorch is as follows,



```
import torch
import torch.nn as nn
import torchvision.transforms as transforms
import torchvision.datasets as dsets

train_dataset = dsets.MNIST(root='./data',
                            train=True,
                            transform=transforms.ToTensor(),
                            download=True)

test_dataset = dsets.MNIST(root='./data',
                           train=False,
                           transform=transforms.ToTensor())
```



```
print(train_dataset.train_data.size())

torch.Size([60000, 28, 28])
```

where FLOPs = 28 x 28 x 60000 ~ 47 MB

The resulting resource manager resembles the figure below which integrates a deep model to evaluate and categorize the request job task. Once evaluated and prioritized is sent to the workload queue containerized with Kubernetes and finally scheduled and assigned workload with the Slurm workload manager which the workload queue submits jobs to the Slurm manager, which automatically cues and allocates resources once available [32].

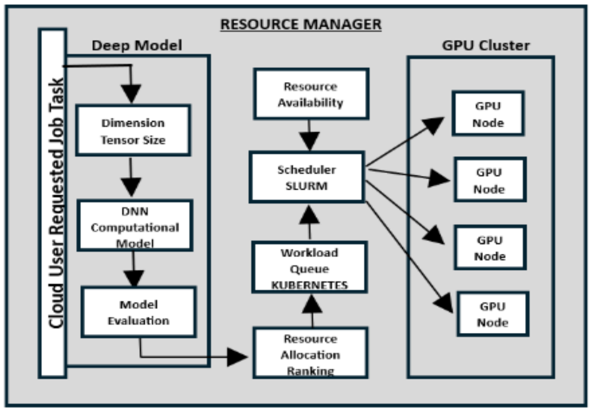
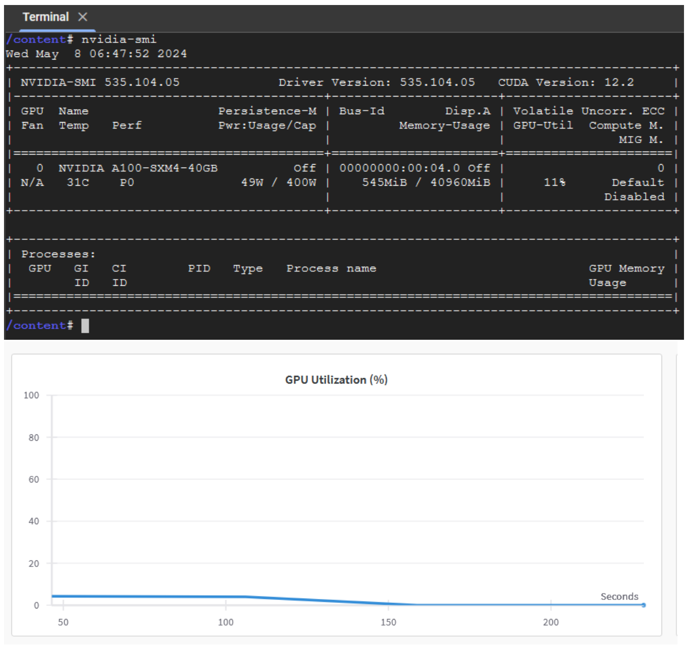


***Figure7**. Multi-layered deep model with predictive resource manager.*

Monitoring GPU utilization is known to be non-trivial to calculate [9, 30]. However, online real-time monitoring can be useful in observing how a GPU is utilized during kernel execution, in which users can obtain relevant metrics from an NVIDIA System Management Interface (nvidia-smi), and through this observe the GPU and Memory utilization. GPU utilization is defined as the percentage of time over the past sample period that one or more kernels are executed on the GPU. The figure below displays an NVIDIA A100-SXM4-40GB GPU with 11% utilization.


```
/content# nvidia-smi
Wed May  8 06:47:52 2024
+---------------------------------------------------------------------------------------+
| NVIDIA-SMI 535.104.05             Driver Version: 535.104.05   CUDA Version: 12.2     |
|-----------------------------------------+----------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |         Memory-Usage | GPU-Util  Compute M. |
|                                         |                      |               MIG M. |
|=========================================+======================+======================|
|   0  NVIDIA A100-SXM4-40GB          Off | 00000000:00:04.0 Off |                    0 |
| N/A   31C    P0              49W / 400W |    545MiB / 40960MiB |     11%      Default |
|                                         |                      |             Disabled |
+-----------------------------------------+----------------------+----------------------+

+---------------------------------------------------------------------------------------+
| Processes:                                                                            |
|  GPU   GI   CI        PID   Type   Process name                            GPU Memory |
|        ID   ID                                                             Usage      |
|=======================================================================================|
+---------------------------------------------------------------------------------------+
/content# 
```

***Figure8**. Example of GPU utilization on NVIDIA A100-SXM4-40GB.*

Profiling GPUs can also be performed to categorize quality of service required by executing workload on a dedicated GPU located in an isolated machine [14, 18, 24], with various device metrics obtainable by profiling tools6. Profiling can be categorized as one of two types: Coarse grained profiling obtains the number of kernels, kernel configuration, GPU/memory utilization, and kernel execution time.

5 Conclusion

Heterogeneous GPU clusters have proven challenging for traditional resource allocation methods which have increased GPU utilization overall but fail to consistently account for diverse quality of service requirements in multi-tenant GPU processing which often present significant non-trivial obstacles.

Moreover, the increase in data creation and big data processing systems requirements of high performance in-memory computing, has led to the development and use of advanced distributed computing techniques such as distributed-parallel computing, cloud computing, clustering, and high-performance computing integrated with heterogeneous computing infrastructures consisting of layered virtualization over multiple variations of concurrent CPU and GPU environments.

This has presented challenges inherent in resource allocation in GPU clusters including inefficient electrical power use, interconnection latency, concurrent work loading and over subscribing, and issues associated with under-utilization in GPU clusters illustrated in figure 3. As a result, resource allocation for virtual machines within cloud based virtualized systems is of paramount importance for efficient use of computational resources.

Here we presented a prediction-based method semi-based on previous works of Yeung et.al. [9] for the dynamic resource allocation of heterogenous clusters based on feedforward multilayered deep models. We demonstrated that there is potential applications in multi-layered deep networks, also known as deep feedforward networks, and have displayed significant enhancement and promising results in increased GPU utilization of heterogenous clusters. Moreover, with the use of Slurm and Kubernetes containers for abstracted management of resource allocation, workload scheduling can be performed with coarse-grained or fine-grained profiling when required for higher accuracy and enhanced performance.

References

[1] G. Yeung, D. Borowiec, R. Yang, A. Friday, R. Harper and P. Garraghan, "Horus: Interference-Aware and Prediction-Based Scheduling in Deep Learning Systems," in IEEE Transactions on Parallel and Distributed Systems, vol. 33, no. 1, pp. 88-100, 1 Jan. 2022, doi: 10.1109/TPDS.2021.3079202.
[2] J. Yao, Q. Lu, R. Tian, K. Li and H. Guan, "An Economy-Oriented GPU Virtualization With Dynamic and Adaptive Oversubscription," in IEEE Transactions on Computers, vol. 72, no. 5, pp. 1371-1383, 1 May 2023, doi: 10.1109/TC.2022.3199998.
[3] Q. Lu, J. Yao, Z. Qi, B. He and H. Guan, "Fairness-Efficiency Allocation of CPU-GPU Heterogeneous Resources," in IEEE

Transactions on Services Computing, vol. 12, no. 3, pp. 474-488, 1 May-June 2019, doi: 10.1109/TSC.2016.2597141.
[4] Q. Lu, J. Yao, H. Guan and P. Gao, "gQoS: A QoS-Oriented GPU Virtualization with Adaptive Capacity Sharing," in IEEE Transactions on Parallel and Distributed Systems, vol. 31, no. 4, pp. 843-855, 1 April 2020, doi: 10.1109/TPDS.2019.2948753.
[5] J. Kim, J. Cha, J. J. K. Park, D. Jeon and Y. Park, "Improving GPU Multitasking Efficiency Using Dynamic Resource Sharing," in IEEE Computer Architecture Letters, vol. 18, no. 1, pp. 1-5, 1 Jan.-June 2019, doi: 10.1109/LCA.2018.2889042.
[6] Zhang, H., School of Information, Shanxi College of Finance and Taxation, Taiyuan, 030024, China, & School of Finance and Economics, Taiyuan University of Technology, Taiyuan 030024, China. (2023). Virtual Machine Allocation in Cloud Computing Environments using Giant Trevally Optimizer. IJACSA International Journal of Advanced Computer Science and Applications, Vol. 14(No. 9), 1104–1105.
[7]Q. Sun, Y. Liu, H. Yang, Z. Luan and D. Qian, "SMQoS: Improving Utilization and Energy Efficiency with QoS Awareness on GPUs," 2019 IEEE International Conference on Cluster Computing (CLUSTER), Albuquerque, NM, USA, 2019, pp. 1-5, doi: 10.1109/CLUSTER.2019.8891047.
[8] Bide Mei and Ritai Yu, "Virtual Machine Resource Allocation Strategy Based on the Resource Utilization," IEEE Conference Anthology, China, 2013, pp.1-5, doi: 10.1109/ANTHOLOGY.2013. 6785010.
[9] Yeung, G., Borowiec, D., Friday, A., Harper, R., Garraghan, P., & Lancaster University. (n.d.). Towards GPU Utilization Prediction for Cloud Deep Learning. *Usenix Conference (2020) presentation.*
[10] Mo, Z., Xu, H., & Lau, W. C. (2024, March 27). Optimal Resource Efficiency with Fairness in Heterogeneous GPU Clusters. arXiv.org. https://arxiv.org/abs/2403.18545
[11] Gooding, Matthew. "US Data Center Power Consumption to Double by 2030". *Newmark*. (2024, January 15).
[12] Wang, S., Chen, S., & Shi, Y. (2024). GPARS: Graph predictive algorithm for efficient resource scheduling in heterogeneous GPU clusters. Future Generation Computer Systems, 152, 127–137.
[13] Mishra, S. Understanding GPU Clusters: Components and Uses (2024, January 19). *Medium*.
[14] Optimal Cluster and GPU Utilization with Run:AI. (2023, December 13). *NettApp*.
[15] GPU Computing. (n.d.). GPU Computing Knowledge Base. *Princeton Research Computing.*
[16] Data centers and servers. (n.d.-b). Department of Energy. Energy.gov. https://www.energy.gov/eere/buildings/data-centers-and-servers
[17] Jeon, M., Venkataraman, S., Phanishayee, A., Qian, J., Xiao, W., & Microsoft Research. (2019). Analysis of Large-Scale Multi-Tenant GPU clusters for DNN training workloads. In Proceedings of the 2019 USENIX Annual Technical Conference.
[18] Yousefzadeh-Asl-Miandoab, E. (2023, March 21). "Analysis of Large-Scale Multi-Tenant GPU Clusters for DNN Training Workloads" Summary. *Medium.*
[19] Woodall, T. S., Shipman, G., Bosilca, G., Graham, R. L., & Maccabe, A. (2006). High performance RDMA Protocols in HPC. In Lecture notes in computer science (pp. 76–85).
[20] 4U GPU System - PCIe Root Architectures. (n.d.). https://www.supermicro.org.cn/products/system/4U/4029/PCIe-Root-Architecture.cfm
[21] Brandon, et.al. (n.d.). NVLink, InfiniBand, and ROCE in AI GPU Interconnect Technologies - NADDOD blog.
[22] Singh, K. (2023, March 25). Integrating Slurm with Kubernetes for Scalable Machine Learning Workloads - *Collabnix*.
[23] Burwell, T., "Kubernetes vs. Slurm: Which Container Orchestration Tool is Right for You?", (2023, August 25 LinkedIn
[24] Wickberg, T. (n.d.). "HPC Workload Managers - Slurm and/or/vs Kubernetes". *SchedMD*.
[25] Pratt, N., Feldman, J.(n.d.), Introducing SUNK: A Slurm on Kubernetes Implementation for HPC and Large Scale AI — *CoreWeave* (KubeCon 2023).
[26] SUNK (Slurm on Kubernetes) | CoreWeave. (n.d.). *CoreWeave.* https://docs.coreweave.com/coreweave-machine-learning-and-ai/training/sunk
[27] Mohan, J., Phanishayee, A., Kulkarni, J., & Chidambaram, V. (2021, October 12). Synergy: Resource sensitive DNN scheduling in Multi-Tenant clusters. arXiv.org. https://arxiv.org/abs/2110.06073.
[28] Tang, X., & Fu, Z. (2020). CPU–GPU Utilization Aware Energy-Efficient Scheduling Algorithm on Heterogeneous Computing Systems. IEEE Access, 8, 58948–58958.
[29] Research computing GPU resources. (n.d.). https://help.rc.unc.edu/gpumonitor/
[30] Yousefzadeh-Asl-Miandoab, E. "Toward GPU Utilization Prediction for Cloud Deep Learning" Summary. *Medium.*
[31] Terry-Jack, M. (2021, December 9). Deep Learning: Feed Forward Neural Networks (FFNNs). Medium.
[32] Introduction to AI in the data center. (n.d.). NVIDIA GPU software [Video]. Coursera.
[33] Introduction to Concurrent Programming with GPUs Johns Hopkins University. (n.d.). Coursera Johns Hopkins University.