Manage the Workloads, not the Cluster: Designing a Control Plane for Large-Scale AI Clusters

Ruiqi Lai[1] Siyu Cao[1] Leqi Li[1] Luo Mai[2] Dmitrii Ustiugov[1]
[1]*NTU Singapore* [2]*University of Edinburgh*

Abstract

The rapid adoption of large language model (LLM) services, such as ChatGPT and DeepSeek, has created unprecedented demand for computational resources, particularly on accelerator-equipped clusters (e.g., GPUs, NPUs). These workloads present unique challenges due to their highly dynamic traffic patterns and multi-dimensional resource demands, including power, memory, and computing. Existing GPU cluster management systems fall short, as they treat accelerators as monolithic units and allocate resources once at the placement time, leading to imbalanced utilization of the above three resource types across the cluster. To address these issues, we propose redefining the LLM serving cluster management as a bin-packing problem, where the resource-specific budgets abstract away hardware resources. We introduce Shapeshifter, the cluster manager that dynamically adjusts the workload deployments to balance the utilization levels of all three resources in the GPUs across the cluster. Shapeshifter monitors resource demands of LLM workload, abstracts away hardware resources with multi-dimensional resource budgets and continuously re-balances resource utilization of LLM workload before allocation of hardware resources. ShapeShifter aims to increase GPU cluster utilization and deployment density while delivering high-quality LLM inference serving. Key future research directions include exploring multi-dimensional model placement, exploring rapid resource rebalancing mechanisms without service disruption, and efficient scheduler policy design.

ACM Reference Format:
Ruiqi Lai[1] Siyu Cao[1] Leqi Li[1] Luo Mai[2] Dmitrii Ustiugov[1] , [1]*NTU Singapore* [2]*University of Edinburgh* . 2025. Manage the Workloads, not the Cluster: Designing a Control Plane for Large-Scale AI Clusters. In *The 5th Workshop on Machine Learning and Systems (EuroMLSys '25), March 30-April 3, 2025, Rotterdam, Netherlands.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3721146.3721937

EuroMLSys '25, March 30-April 3, 2025, Rotterdam, Netherlands

ACM ISBN 979-8-4007-1538-9/2025/03
https://doi.org/10.1145/3721146.3721937

1 Introduction

Emerging LLM services have started to revolutionize the computing industry. Services, such as ChatGPT [16] and DeepSeek [5], have amassed hundreds of millions of users. This massive user base has generated substantial demand for computing resources. Large language models (LLMs) are typically deployed on accelerator-equipped clusters (GPUs, NPUs), which are renowned for their high costs and intensive resource consumption. Consequently, providing efficient and reliable services becomes a critical problem.

LLM online serving workloads pose unique challenges for data center architecture and cluster management. First, a large fraction of the application is user-facing, exhibiting highly dynamic traffic [12, 17]. Such a dynamic traffic pattern will lead to corresponding variations in resource usage for LLM workloads. Second, LLM workloads feature multi-dimensional resource demands. In addition to conventional memory and computational resources, a key characteristic of LLM workloads is their power-intensive nature [18, 24]. This implies that a successful cluster manager must account for all resource dimensions, including the power supply of the data center.

Our studies characterize the multi-dimensional resource usage of LLM workloads, including power, memory, and computing resources, under highly dynamic, volatile patterns. We reveal that these resources do not strongly correlate with one another. Moreover, we find that any of the above resources can become a bottleneck depending on the ever-changing characteristics of traffic.

Designing an efficient LLM-serving cluster management system is key to satisfying the growing demand for multi-dimensional resources by maximizing the utilization of available hardware. Prior work in GPU cluster management falls short of addressing this need for several reasons. First, existing systems treat GPUs (and other accelerators) as monolithic black boxes [9, 21, 26] and only consider their overall utilization [17, 23], ignoring the multi-dimensional resource demand of LLM workloads. Second, current systems lack

Ruiqi Lai[1] Siyu Cao[1] Leqi Li[1] Luo Mai[2] Dmitrii Ustiugov[1]
[1] NTU Singapore [2] University of Edinburgh

a mechanism that can dynamically adjust the demands of specific resource types [24], leading to imbalanced resource utilization in the cluster.

To address these challenges, we propose to model LLM online service management as a bin-packing problem. We abstract the hardware resources within the cluster, such as GPUs, as fundamental resource allocation units. For instance, a single GPU provides a certain amount of memory, computation, and power resources. We then periodically monitor the multi-dimensional resource demands across the cluster. By tracking metrics such as token count and request volume, estimating the current resource requirements becomes straightforward. The role of our cluster manager is to minimize the number of bins (GPUs) needed to satisfy these demands.

For concrete cluster management operations, we propose managing workload resource demands instead of managing cluster resource allocation. We introduce *Shapeshifter* that dynamically adjusts the resource demands of LLM workloads, by devising resource demands from the LLM workloads, projecting resource budgets based on the hardware characteristics, and continuous re-evaluation and re-balancing of the cluster resources utilization.

With Shapeshifter, we open a new perspective on LLM serving cluster management, which can increase GPU cluster resource utilization, improving deployment density and overall system performance. First, more research is necessary to find methods for exploring and navigating the multi-dimensional space of model instance deployment and placement atop of the heterogeneous cluster hardware. Second, we need to design efficient scheduling and placement policies able to solve the multi-dimensional bin-packing problem within the acceptable time ranges. Finally, we will design novel approaches and mechanisms for continuous re-balancing of the cluster resources without service downtime.

2 Background

2.1 LLM Inference Workloads

LLM inference processes user requests, where each request comprises multiple tokens—individual units of text, such as words, subwords, or characters, that the model uses to understand and generate responses. LLM inference generates tokens in an auto-regressive way, which means previous output tokens will be used as the input to generate the next token. Generating the first token is defined as the prefill phase while generating subsequent tokens is defined as the decode phase. Typically, the decode phase dominates a request's end-to-end latency. During this procedure, LLM inference needs to store the KV-cache to speed up processing, which contains the self-attention results from previous tokens. KV-cache has a one-to-one correspondence with the processing tokens, so the memory demand of LLM inference grows linearly with the processing token number. LLM inference's compute resource usage scales proportionally with the current batch size [4].

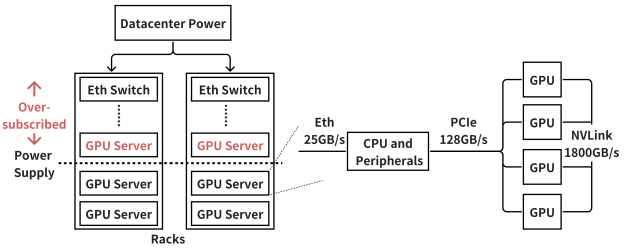


Figure 1. GPU hardware layout.

2.2 Hardware Layout of GPU Cluster

Current LLM inference workloads are deployed on GPU clusters with complicated interconnection topology and possible heterogeneous configuration.

Fig. 1 shows the typical hardware layout of GPU in the data center. GPU nodes reside in racks in data centers. Each node features multiple GPUs. Each GPU provides a complete set of resources as a unit. Datacenters use power distribution units (PDU) to power up racks. Although data centers can adjust the number of PDUs one rack has, modern data centers prefer to provide power in a uniform distribution. The data center should provide power to racks roughly similar to all other racks within the same data center. This indicates that the power supply to each rack is constant.

With a constant power supply on each rack, datacenters often oversubscribe rack power [3, 6, 19]. In Fig. 1, we mark those over-subscribed GPU servers in red. The oversubscription ratio is determined by the ratio of power density to the power supply.

2.3 Power Oversubscription

The industry trends reveal that the demand for power, memory and computing continues to grow exponentially [20, 22, 25]. As a result, the power density is increasing dramatically while the data center and rack-level power supply remain fixed [7, 8, 13]. This growing power density pushes the providers to over-subscribe the rack power, leading to the *dark-rack* effect, resulting in the impossibility of powering up all GPU nodes, or all GPUs, in a rack simultaneously,

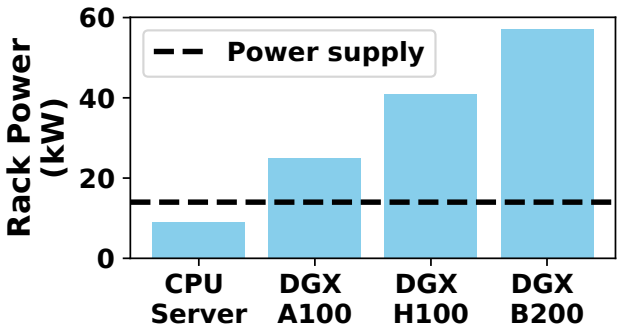


Figure 2. Rack power for different type of servers.

making power one of the key limiting factors for the LLM inference cluster operation.

Datacenters use power distribution units (PDU) to power up racks. Although data centers can adjust the number of PDUs one rack has, modern data centers prefer to provide power in a uniform distribution. The data center should provide power to racks roughly similar to all other racks within the same data center. This indicates that the power supply to each rack is constant.

The power oversubscription ratio is growing. Fig. 2 shows the power density of several recent GPU servers and CPU servers. We use the dashed line to indicate the power supply of a conventional, non-AI rack [2, 10]. Compared to traditional CPU servers, recent GPU servers' power demand is increasing from year to year. Therefore, we project the power oversubscription ratio will also grow in response to this rapidly growing power-demand trend in GPU servers.

2.4 Existing GPU Cluster Manager

GPU cluster manager typically comprises the request router, load balancer and GPU manager. When requests arrive, the cluster manager routes them to different GPUs. GPU manager periodically monitors the cluster status and adjusts the cluster size with GPU allocation, deallocation, and live migration.

However, existing GPU cluster managers allocate resources as an entire GPU, which tightly couples the resources; hence, compute, memory, and power resources cannot be adjusted separately, as in CPU schedulers [1]. Prior works typically focus on one dimension of resources. DynamoLLM [24] is the state-of-the-art GPU cluster manager that solely focuses on power consumption. Singularity [23] focuses only on memory usage. Finally, ServerlessLLM [9] focuses on queue occupancy but disregards the utilization of individual resources.

Therefore, existing cluster managers lack a holistic approach that considers all key dimensions, including memory, computing, and power.

3 Workload Characterization

In the following sections, we will study the correlation of the utilization profiles of the three key resources in GPU clusters: computing, memory capacity, and power, with a set of micro-benchmarks to demonstrate that the system need to account for all the above resource dimensions.

3.1 Methodology

We develop a load generator that can generate LLM inference traffic load. The load generator can up-sample or down-sample the original traces to simulate different traffic load. In the following experiments, we sweep the traffic load and send requests to LLM inference engine, after requests get processed, we record their end-to-end latency. To normalize the latency of requests with different lengths, we use slowdown instead of absolute end-to-end latency in Fig. 5. We calculate each request's latency slowdown with the unloaded latency of the same request. For each request, the unload latency is calculated on a single GPU of the same type to fairly evaluate performance degradation under load for requests with short and long prompts:

$$\text{Slowdown} = \frac{\text{Latency}_{\text{in a GPU cluster under load}}}{\text{Latency}_{\text{on a single GPU in isolation}}}$$

In the following experiments, we use a one-hour trace from Azure[12], and use vLLM[11] as the inference engine because it is the state-of-the-art LLM inference engine. We choose Meta's Llama3-8B model. In the following experiment, we use A100-40GB GPU if not otherwise stated. The setup is with a similar memory capacity ratio of the down-scaled Llama3- 70B (141GB) on an NVIDIA B200 (192GB).

3.2 LLM Resource Demand Correlation

Existing GPU clusters allocate cluster resources at the coarse granularity of entire GPUs and often monitor the utilization of a single resource type, e.g., memory [17, 23], computing [17] and power [24]. Instead, we advocate for a holistic approach accounting for all resource types. The questions we ask here are: First, do all resource types in the LLM workload strongly correlate? Second, what key resource types should the GPU cluster manager monitor in the LLM workload? To answer whether resources of LLM inference correlate with each other, we measure the compute, memory, and power usage when replaying the real LLM inference trace on a single NVIDIA A100-40GB GPU with Llama3-8B model. We measure power usage using `nvidia-smi` using the `Power Draw`

Ruiqi Lai[1] Siyu Cao[1] Leqi Li[1] Luo Mai[2] Dmitrii Ustiugov[1]
[1] NTU Singapore [2] University of Edinburgh

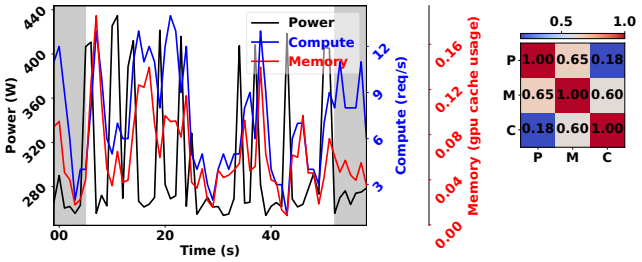


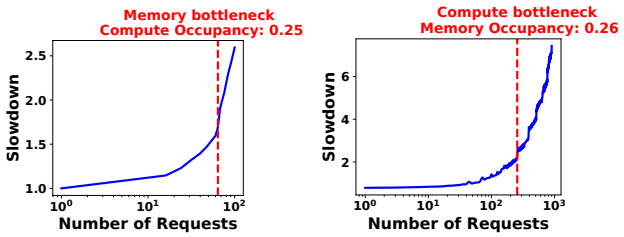
Figure 3. Various resources usage over a minute in the trace (left): The shadowed areas show strong correlation while others do not. Pearson correlation matrix for an hour-long trace (right): C, M, P stand for compute, memory, power, respectively.

counter. We profile GPU memory usage with the KV-cache usage metric, which is vLLM runtime's metric that shows the fraction of the total available memory space in GPU occupied by the KV-cache. We profile compute usage by measuring a proxy counter: we measure the number of requests processing concurrently reported by the vLLM runtime instead of directly measuring FLOPs[14, 15] because the latter approach introduces substantial slowdown.

The left part of Fig. 3 shows the memory, computing, and power resources usage over one minute taken during the replay of the full trace for illustration of the trends. The right part of the figure shows their Pearson correlation matrix collected during the replay of the whole, one-hour-long trace. Surprisingly, we observe that resources do not correlate strongly in a general case. For example, the three resources show a high correlation during the first shaded area (0 to 5 seconds), while at the end of this minute, the three resources show different trends in the second shaded area (55 to 60 seconds). The right subplot of Fig. 3 shows the correlation matrix captured over the entire period of the one-hour trace replaying. It indicates that power resource usage does not highly correlate with memory or computing resource usage. Although memory and computing resources show a high correlation over the entire one hour, it's clear that the correlation does not always hold within shorter periods. The poor correlation we identified between the three resources can be attributed to two reasons. First, a single inference comprises tens of GPU kernels with diverse resource requirements, and each GPU executes tens to hundreds of requests in batches. The requests can arrive and finish at any time due to the continuous batching [11], which might cause resource utilization variability. Second, we also observe that the vLLM runtime metrics can lag behind request execution by 0.5-1.5 seconds, the impact of which is difficult to determine with the existing measurement systems. The obtained results suggest that the desired GPU cluster control plane must rely on carefully measuring *each* of these dimensions to drive the placement and load balancing optimizations.

Next, we use two micro-benchmarks to showcase that any of these three resource types can become the bottleneck.

3.3 Memory and Compute as the Bottleneck



(a) Requests served in the long-request pool.

(b) Requests served in the short-request pool.

Figure 4. Latency slowdown of requests served in long-request pool and short-request pools vs. the number of requests in a batch.

LLM inference serving memory resource demand is proportional to the number of tokens, which is the sum of the input and output length of the served requests, while computing resource demand is proportional to the number of requests. However, the number of tokens can vary drastically among different requests and is unpredictable due to the unpredictability of output token length. As a result, computing and memory resources can become the bottleneck in different situations. For the long requests with many tokens, memory resources are likely to run out before computing resources. In contrast, for short requests, computing resources are more likely to become the bottleneck.

State-of-the-art cluster managers[24] separate requests by token length, serving long and short requests in distinct GPU pools to improve energy efficiency through tailored power configurations. To evaluate this approach, we analyze a system configured with two GPU pools, following the classification method of DynamoLLM, where "short" and "long" requests are defined as the 5^{th} and 95^{th} percentile token lengths in the Azure trace, respectively. Each pool employs a single A100-40GB GPU.

We measure latency slowdown by sweeping the number of requests processed in batches for both pools. Fig 4 illustrates the latency slowdown versus the number of requests.

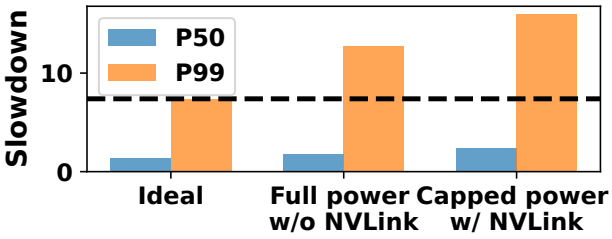


Figure 5. 50^{th} and 99^{th} percentile latency slowdown for requests served under three different configurations. 1. Ideal: with NVLink but without power capping. 2. The GPUs are in different racks, hence without NVLink. 3. The GPUs are in the same rack with power constraints.

For long requests (Fig. 4a), latency increases sharply once memory capacity is exhausted (marked by the dashed line), as queuing delays dominate. For short requests (Fig. 4b), latency increases when compute resources are saturated (dashed line), reflecting the limited processing throughput.

The results highlight distinct bottlenecks: long requests are constrained by memory capacity, while short requests face compute limitations. This imbalance underscores how LLM inference workloads exhibit divergent resource demands depending on request characteristics. Resource underutilization arises when compute or memory remains idle due to this mismatch, suggesting opportunities for better resource coordination.

3.4 Power as the Bottleneck

Existing GPU cluster managers try to allocate GPUs within a node to leverage intra-node GPU interconnections like NVLink. However, this layout increases the power density of GPU servers. We assume that racks are uniformly powered in datacenters, so increasing the power density of a single rack will disrupt such uniform power distribution and force the cluster manager to apply power capping to all the GPUs on that rack. Another choice is to allocate GPUs across nodes to decrease the power density of GPU servers, but this triggers higher communication overhead.

To analyze trade-offs in power-constrained environments, we model two real-world GPU deployment strategies under rack power oversubscription and evaluate their performance via a microbenchmark. First, we simulate co-located deployments where providers prioritize physical proximity and NVLink-enabled communication but reduce GPU frequency to comply with rack power limits. Second, we model scattered deployments where workloads are distributed across clusters without high-speed interconnects (e.g., NVLink) to avoid localized power bottlenecks.

We evaluate these scenarios using vLLM on two NVIDIA A5000 GPUs (chosen for administrative access), replaying an Azure trace under three configurations. First, we set the ideal case with full GPU power with NVLink enabled, representing unconstrained theoretical performance. The second deployment is scattered deployment: Full GPU power with NVLink disabled, simulating distributed workloads across clusters lacking interconnects due to partial rack activation. The third is co-located deployment: Capped GPU power (via frequency limits) with NVLink enabled, reflecting power-constrained but locality-optimized deployments. This setup quantifies how each strategy contends with distinct bottlenecks—scattered deployments sacrifice communication efficiency, while co-located deployments trade peak compute performance for power savings.

Fig. 5 shows the 50^{th} percentile and 99^{th} percentile slowdown under the three conditions. Compared with its theoretical performance, full power without NVLink triggers 1.6×99^{th} percentile latency while capped power with NVLink triggers 2.1×99^{th} percentile latency. We show that allocating GPU in either way can cause a performance degradation compared to the theoretical performance.

4 ShapeShifter: Multi-Dimensional LLM Cluster Manager

4.1 Holistic Cluster Resource Accounting

Micro-benchmarks in Sec 3 demonstrate that all the resource dimensions can become the performance bottleneck in the real online LLM serving system. Therefore, an efficient cluster manager should not overlook any of them.

We formulate LLM cluster management as a multi-dimensional bin-packing problem to address this challenge. Over a given time period, our objective is to minimize the number of GPUs (bins) required to meet the complete resource demands of the current LLM workload.

To solve the multi-dimensional bin-packing problem, we first need to estimate all resource demands across multiple dimensions. We project each GPU as a multi-dimensional resource bin. We can estimate memory resource demand by combining the number of processed tokens and model replicas deployed. We can derive the required compute resources based on the total number of incoming requests. Finally, we can evaluate the power demand using the aggregate power consumption readings from all GPUs in the cluster. For example, an NVIDIA A100 GPU provides 40GB of memory resources, computational resources for up to 100 requests (calculated based on the KV-cache footprint of the model)

Ruiqi Lai[1] Siyu Cao[1] Leqi Li[1] Luo Mai[2] Dmitrii Ustiugov[1]
[1] NTU Singapore [2] University of Edinburgh

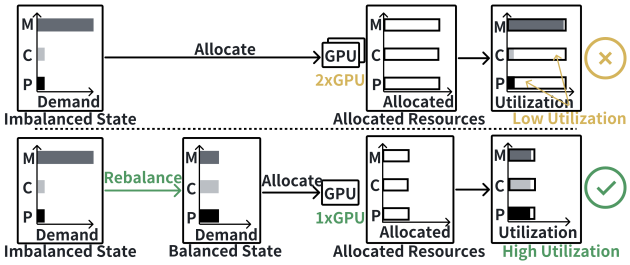


Figure 6. Top subfigure: resource allocation without reshaping. Bottom subfigure: resource allocation with reshaping. M, C, P stand for memory, compute, power, respectively.

and has 400W budget according to its TDP. By formulating the problem in this way, we can determine an optimal allocation strategy that minimizes the number of active GPUs while ensuring that all resource constraints are satisfied. This approach enables more efficient resource utilization, reducing energy consumption and operational costs while maintaining service performance.

However, under such modeling, the number of bins is determined solely by the resource dimension with the largest demand, i.e., the resource bottleneck. In § 3, we demonstrate that all resource dimensions can become potential bottlenecks in a real system and they do not always correlate. If we allocate GPUs based on the bottlenecked resource type, other dimensions of resources in allocated GPU will be under low utilization.

4.2 ShapeShifter Workload Manager

To solve the bin-packing problem, we propose to manage the workload deployments rather than GPU assignment to the models deployed in the cluster, as in the existing cluster managers [9, 24]. We find that the LLM workload can be conducted in multiple configurations with completely different resource demands. For example, configurations such as the parallelism strategy, parallelism levels, load balancing strategy, and GPU power capping settings can all significantly change the current resource demand in different dimensions. Therefore, we propose that the cluster manager should aim to re-balance the workload resource demands across multiple dimensions before making placement decisions. Fig. 6 shows the difference between resource allocation without re-balancing and with re-balancing. Since the resource demands of LLM workload are initially unbalanced, with memory resource demand greatly surpassing the other two types of resources, the cluster manager has to allocate two GPUs to satisfy the memory demands. Such an allocation causes low utilization of computing and power resources. On the contrary, Shapeshifter can adjust the LLM deployment configurations before proceeding to allocating resources. After re-balancing, in this example, the cluster manager only needs to allocate a single GPU. Such a re-balancing step significantly improves resource utilization and reduces the number of GPUs used to satisfy the resource demands of LLM workloads.

4.3 Challenges and Open Questions

Expanding Resource Monitoring and Modeling. While current modeling focuses on memory, computing, and power, other resource dimensions—such as locality, network bandwidth, and GPU heterogeneity, need deeper investigation. Locality (e.g., data placement across GPUs/nodes) impacts communication overhead, while network bandwidth constraints can throttle distributed LLM inference. Heterogeneous GPU clusters further complicate resource allocation, as varying hardware capabilities (e.g., a cluster with both NVIDIA A100 and H100, and AMD GPUs) require an abstract, unified model to characterize LLM workloads across all dimensions. Future research must define generalized metrics to quantify these factors and integrate them into the multi-dimensional bin-packing framework. For instance, modeling inter-GPU communication costs or dynamically adapting to heterogeneous hardware profiles could prevent underutilization and latency spikes.

Continuous Resource Re-Balancing Mechanism Without Service Disruption. Adjusting workload configurations to balance resource demands is critical, but existing methods often require shutting down and restarting LLM instances disrupting the inference serving service. Rapid traffic fluctuations require a mechanism to reconfigure parallelism strategies, power caps, or load-balancing policies without downtime. Achieving this necessitates low-overhead state migration techniques, such as migrating model weights and KV-cache across GPUs. However, LLMs' massive memory footprints and stateful inference contexts (e.g., KV-caches) pose significant technical challenges. Research into lightweight reconfiguration protocols and live migration frameworks could bridge this gap, enabling sub-second re-balancing to match dynamic cluster conditions.

Efficient Scheduler Policy Design. Solving the multi-dimensional bin-packing problem with reshaping introduces a new space of scheduling possibilities. Brute-force approaches are impractical for real-time decision-making, especially as cluster scales grow. Efficient policies are needed to approximate optimal allocations within milliseconds. Challenges

include balancing solution quality with computational overhead and integrating reshaping actions (e.g., adjusting parallelism levels) into the scheduling loop.

5 Related Work

Datacenter cluster management. In traditional CPU-centric workload management, finer-grained resource scheduling is becoming an emerging trend in cluster management. For example, Borg [26] introduced the idea of assigning different memory and compute resources to different tasks at an early stage, with a unified cluster manager responsible for resource allocation. More recent works, such as Quicksand [21], have attempted to achieve even finer-grained resource scheduling. They designed new user interaction models that decompose a function into smaller fungible functions, which are then executed by proclets—a fundamental resource scheduling unit capable of migrating within milliseconds—to improve cluster resource utilization.

However, these works still treat GPU resources as monolithic scheduling units, lacking finer-grained, multidimensional resource isolation. Simply applying traditional CPU cluster optimization strategies to GPU clusters is also challenging. Unlike CPUs, which have well-established mechanisms for compute and memory isolation, GPUs still lack mature virtualization solutions for effective resource isolation at the hardware level.

GPU cluster management. For GPU cluster management, there is currently no unified resource scheduling framework. Most existing works focus on a single dimension of resource allocation while neglecting the multidimensional resource demands of large language models (LLMs). For example, DynamoLLM [24] focuses on the power consumption of LLM workload, proposing separate GPU pools for different types of requests. Splitwise [17] focuses on the memory and computing demands of the prefill and decode phase in LLM inference and proposes splitting GPU pools solely for the prefill phase and the decode phase. ServerlessLLM [9] focuses on optimizing the cold start latency of an LLM inference system with scalability. However, it performs auto-scaling based on queue metrics, dismissing GPU hardware-related metrics like memory and compute utilization. Singularity [23] proposes by mechanism including preemption, live-migration, checkpoint saving and restoring to separate DNN workloads from hardware deployments. However, their method focuses on traditional DNN workloads, and will trigger large overhead when dealing with LLM workloads, also, they only focus on the utilization of GPU memory resources. Existing works on GPU cluster management for LLM inference consider only partial resource dimensions, leading to the underutilization of certain resources. These approaches fail to optimize overall resource efficiency without a unified scheduling framework that accounts for the multidimensional demands of LLM workloads.

6 Conclusion

In conclusion, managing LLM serving clusters presents unique challenges due to dynamic traffic patterns and multi-dimensional resource demands encompassing computing, memory, and power. Existing GPU cluster management systems fall short by treating accelerators as monolithic units and lacking dynamic adjustment mechanisms, leading to imbalanced resource utilization. We propose Shapeshifter, a cluster manager that addresses these issues by modeling LLM service management as a bin-packing problem, abstracting hardware resources, and dynamically adjusting workload deployments to balance resource utilization.

Shapeshifter offers a novel perspective on LLM serving cluster management, aiming to increase GPU cluster resource utilization, decrease its operational and embodied carbon footprint, and enhance the sustainability of LLM serving systems. Future research directions include exploring multi-dimensional model placement, developing rapid resource re-balancing mechanisms, and efficient scheduler policy design.

7 Acknowledgements

We express our sincere gratitude to Yao Fu from the University of Edinburgh for their valuable insights and contributions to this work. We also extend our appreciation to the members of the HyScale Lab for their feedback and fruitful discussions throughout this project. The NTU Seed Tier-1 grant supported this work.

References

[1] [n. d.]. Kubernetes. Available at https://kubernetes.io.

[2] NVIDIA [n. d.]. *NVIDIA DGX SuperPOD: Data Center Design Featuring NVIDIA DGX H100 Systems.* NVIDIA. https://docs.nvidia.com/dgx-superpod/design-guides/dgx-superpod-data-center-design-h100/latest/electrical.html Accessed: February 11, 2025.

[3] Data Canopy 2024. *4 Tips to Avoid Oversubscribing to Power in Your Data Center.* Data Canopy. https://datacanopy.com/4-tips-avoid-oversubscribing-power-data-center/

[4] Amey Agrawal, Nitin Kedia, Ashish Panwar, Jayashree Mohan, Nipun Kwatra, Bhargav S. Gulavani, Alexey Tumanov, and Ramachandran Ramjee. 2024. Taming Throughput-Latency Tradeoff in LLM Inference with Sarathi-Serve. In *Proceedings of the 18th Symposium on Operating System Design and Implementation (OSDI).* 117–134.

[5] DeepSeek AI. 2025. DeepSeek-R1. https://huggingface.co/deepseek-ai/DeepSeek-R1 Introduces DeepSeek-R1, a reasoning model comparable

to OpenAI-o1 in math, code, and reasoning tasks. It highlights the open-sourcing of DeepSeek-R1-Zero, DeepSeek-R1.

[6] Shoaib Akram, Joseph Izraelevitz, Christos Kozyrakis, Radhika Mittal, Jennifer Switzer, Rachee Singh, and Rebecca Isaacs. 2021. Prediction-Based Power Oversubscription in Cloud Platforms. In *2021 USENIX Annual Technical Conference*. https://www.microsoft.com/en-us/research/uploads/prod/2020/10/Per-VM-Capping-ATC21.pdf Accessed: February 11, 2025.

[7] CoreSite. 2025. Facing the Data Center Power Density Challenge. https://www.coresite.com/blog/facing-the-data-center-power-density-challenge Accessed 2025-02-10.

[8] Fierce Network. 2024. Cloud providers want to crank up rack power for AI. https://www.fierce-network.com/cloud/cloud-providers-want-crank-rack-power-10x-ai Accessed 2025-02-10.

[9] Yao Fu, Leyang Xue, Yeqi Huang, Andrei-Octavian Brabete, Dmitrii Ustiugov, Yuvraj Patel, and Luo Mai. 2024. ServerlessLLM: Low-Latency Serverless Inference for Large Language Models. In *Proceedings of the 18th Symposium on Operating System Design and Implementation (OSDI)*. 135–153.

[10] Steven Hambruch and Dennis O'Brien. 2024. *DGX SuperPOD Data Center Best Practices with DGX B200*. Technical Report. NVIDIA. https://docs.nvidia.com/nvidia-dgx-superpod-data-center-best-practices-with-dgx-b200.pdf Accessed: February 11, 2025.

[11] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the 29th ACM Symposium on Operating Systems Principles (SOSP)*. 611–626.

[12] Microsoft Azure. [n. d.]. Azure Public Dataset: Azure LLM Inference Trace 2023. Available at https://github.com/Azure/AzurePublicDataset/blob/master/AzureLLMInferenceDataset2023.md.

[13] Newmark. 2025. US data center power consumption to double by 2030. https://www.datacenterdynamics.com/en/news/us-data-center-power-consumption/ Accessed 2025-02-10.

[14] NVIDIA. 2025. NVIDIA Nsight Compute. https://developer.nvidia.com/nsight-compute Accessed: 2025.

[15] NVIDIA. 2025. NVIDIA Nsight Systems. https://developer.nvidia.com/nsight-systems Accessed: 2025.

[16] OpenAI. 2024. ChatGPT. https://chatgpt.com ChatGPT helps you get answers, find inspiration and be more productive. It is free to use and easy to try. Just ask and ChatGPT can help with writing [4]..

[17] Pratyush Patel, Esha Choukse, Chaojie Zhang, Aashaka Shah, Íñigo Goiri, Saeed Maleki, and Ricardo Bianchini. 2024. Splitwise: Efficient Generative LLM Inference Using Phase Splitting. In *Proceedings of the 51st International Symposium on Computer Architecture (ISCA)*. 118–132.

[18] Pratyush Patel, Esha Choukse, Chaojie Zhang, Íñigo Goiri, Brijesh Warrier, Nithish Mahalingam, and Ricardo Bianchini. 2024. Characterizing Power Management Opportunities for LLMs in the Cloud. In *ASPLOS (3)*. 207–222.

[19] T. Patki. 2023. *Towards Safe Power Oversubscription and Energy Efficiency of Data Centers*. Ph. D. Dissertation. University of South Florida. https://digitalcommons.usf.edu/cgi/viewcontent.cgi?article=10164&context=etd Accessed: February 11, 2025.

[20] Precedence Research. 2025. In-Memory Computing Market. https://www.precedenceresearch.com/in-memory-computing-market Accessed 2025-02-10.

[21] Zhenyuan Ruan, Shihang Li, Kaiyan Fan, Marcos K. Aguilera, Adam Belay, Seo Jin Park, and Malte Schwarzkopf. 2023. Unleashing True Utility Computing with Quicksand. In *Proceedings of The 19th Workshop on Hot Topics in Operating Systems (HotOS-XIX)*. 196–205.

[22] Tao Xu Hongyang Chen Schahram Dustdar Sylvain Gigan Deniz Gunduz Ekram Hossain Shiqiang Zhu, Ting Yu. 2024. Intelligent Computing: Concepts, Methodologies, and Applications. *Intelligent Computing* 2024, Article ID 0006 (2024), 1–XX. https://doi.org/10.34133/icomputing.0006 Accessed 2025-02-10.

[23] Dharma Shukla, Muthian Sivathanu, Srinidhi Viswanatha, Bhargav S. Gulavani, Rimma Nehme, Amey Agrawal, Chen Chen, Nipun Kwatra, Ramachandran Ramjee, Pankaj Sharma, Atul Katiyar, Vipul Modi, Vaibhav Sharma, Abhishek Singh, Shreshth Singhal, Kaustubh Welankar, Lu Xun, Ravi Anupindi, Karthik Elangovan, Hasibur Rahman, Zhou Lin, Rahul Seetharaman, Cheng Xu, Eddie Ailijiang, Suresh Krishnappa, and Mark Russinovich. 2022. Singularity: Planet-Scale, Preemptive and Elastic Scheduling of AI Workloads. *CoRR* abs/2202.07848 (2022).

[24] Jovan Stojkovic, Chaojie Zhang, Íñigo Goiri, Josep Torrellas, and Esha Choukse. 2024. DynamoLLM: Designing LLM Inference Clusters for Performance and Energy Efficiency. In *Proceedings of the 31st IEEE Symposium on High-Performance Computer Architecture (HPCA)*.

[25] TechInsights. 2025. Memory Market Outlook: AI Demand and Tight Supply Drive Resurgence. https://www.techinsights.com/blog/memory-market-outlook-ai-demand-and-tight-supply-drive-resurgence Accessed 2025-02-10.

[26] Abhishek Verma, Luis Pedrosa, Madhukar Korupolu, David Oppenheimer, Eric Tune, and John Wilkes. 2015. Large-scale cluster management at Google with Borg. In *Proceedings of the 2015 EuroSys Conference*. 18:1–18:17.