SPAD: Specialized Prefill and Decode Hardware for Disaggregated LLM Inference

Hengrui Zhang
hengrui.zhang@princeton.edu
Princeton University

Pratyush Patel
patelp1@cs.washington.edu
University of Washington

August Ning
aning@princeton.edu
Princeton University

David Wentzlaff
wentzlaf@princeton.edu
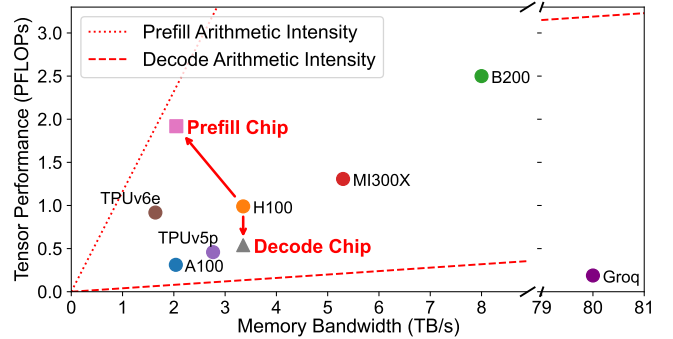Princeton University

Abstract

Large Language Models (LLMs) have gained popularity in recent years, driving up the demand for inference. LLM inference is composed of two phases with distinct characteristics: a compute-bound prefill phase followed by a memory-bound decode phase. To efficiently serve LLMs, prior work proposes prefill-decode disaggregation to run each phase on separate hardware. However, existing hardware poorly matches the different requirements of each phase. Current datacenter GPUs and TPUs follow a *more-is-better* design philosophy that maximizes compute and memory resources, causing memory bandwidth underutilization in the prefill phase and compute underutilization in the decode phase. Such underutilization directly translates into increased serving costs.

This paper proposes SPAD (Specialized Prefill and Decode hardware), adopting a *less-is-more* methodology to design specialized chips tailored to the distinct characteristics of prefill and decode phases. The proposed Prefill Chips have larger systolic arrays and use cost-effective GDDR memory, whereas the proposed Decode Chips retain high memory bandwidth but reduce compute capacity. Compared to modeled H100s, simulations show that the proposed Prefill Chips deliver 8% higher prefill performance on average at 52% lower hardware cost, while the proposed Decode Chips achieve 97% of the decode performance with 28% lower TDP.

End-to-end simulations on production traces show that SPAD reduces hardware cost by 19%-41% and TDP by 2%-17% compared to modeled baseline clusters while offering the same performance. Even when models and workloads change, SPAD can reallocate either type of chip to run either phase and still achieve 11%-43% lower hardware costs, demonstrating the longevity of the SPAD design.

1 Introduction

Large Language Models (LLMs) have become immensely popular due to their advanced capabilities and are being widely adopted in various applications spanning chatbots and code generation tools. However, serving LLMs incurs significant hardware costs. In early 2023, only three months after the introduction of GPT-3.5, it was estimated that ChatGPT cost nearly $700,000 per day to serve with around 30,000 NVIDIA

Figure 1. Comparison of LLM Serving Hardware. We estimate prefill and decode arithmetic intensities for BLOOM-176B (FP16, sequence length 1024) with batch sizes of 2 and 64 respectively (shown as the dashed lines). Only the FLOPs and memory accesses related to matrix multiplications are included. The tensor performance and memory bandwidth numbers are theoretical values reported by their specifications [8, 15, 16, 26, 42–44].

A100 GPUs [49]. The demand for LLMs has increased rapidly since then. In March 2025, NVIDIA announced that they had received orders for 3.6 million of their newest flagship Blackwell GPUs from cloud providers [38], of which a significant fraction will likely be used to serve LLMs.

Serving LLMs is expensive due to tight latency constraints coupled with high hardware requirements. LLM inference executes in two phases with different computational properties. In the compute-bound *prefill* phase, all tokens in the input prompt are processed in parallel to generate the KV cache and the first output token. In the memory-bound *decode* phase, subsequent output tokens are generated sequentially, where each new token depends on the KV cache state of all previous tokens. This dual-phase nature poses a challenge for existing hardware since each phase effectively utilizes only a subset of the hardware resources.

To improve efficiency, prior work has proposed two broad scheduling techniques. **Co-location**-based schedulings batch the prefill and decode phases of different requests together to improve hardware utilization, leveraging the fact that both

phases share the same model weights [4, 69]. However, this approach incurs large tail latencies due to prefill-decode interference [74], leading to a less responsive user experience. **Disaggregation**-based scheduling separates the execution of the prefill and decode phases onto different hardware by using interconnects to transfer the KV cache [51, 74]. Although this approach incurs transfer overheads, it improves overall performance by enabling phase-specific resource management and using hardware that better matches the computational characteristics of each phase.

Despite these software-level optimizations, the hardware efficiency of serving LLM inference is still fundamentally limited by the mismatch between the workload requirements (i.e., TFLOPS and memory accesses) and hardware resources (i.e., compute capacity and memory bandwidth). Current datacenter GPU/TPU design philosophy tends to fit as much compute capacity and cache as possible onto a reticle-sized die and pair it with High Bandwidth Memory (HBM) using the advanced CoWoS (Chip-on-Wafer-on-Substrate) packaging technology [28]. The resulting enormous TFLOPs and memory bandwidths make them the most popular hardware platform for serving LLMs. However, this *more-is-better* design philosophy drives up costs for disaggregation-based LLM inference: the high arithmetic intensity of the prefill phase leaves the expensive HBM underutilized, and the low arithmetic intensity of the decode phase leaves the compute capacity underutilized. Our simulations show that reducing the memory bandwidth of a modeled NVIDIA H100 by 40% would only increase prefill latency by 17%. Likewise, the simulated decode latency only increases by 22% if we reduce the compute capacity by half.

We propose SPAD (Specialized Prefill and Decode hardware), tailoring specialized hardware to the distinct characteristics of prefill and decode phases to improve disaggregation-based LLM serving efficiency. In contrast to the *more-is-better* design philosophy of GPUs, SPAD embraces a *less-is-more* design philosophy that rightsizes hardware for each phase, while still retaining the ability to run the other phase. For the compute-heavy prefill phase, we propose a specialized Prefill Chip with larger systolic arrays and a cost-effective GDDR-based memory system. For the low-arithmetic-intensity decode phase, we propose an area- and TDP-efficient Decode Chip with smaller systolic arrays and caches. Simulations with LLMCompass [71] show that compared to modeled H100s, our proposed Prefill Chips deliver 8% higher prefill performance on average at 52% lower hardware cost, while our proposed Decode Chips achieve 97% of the decode performance with 28% lower TDP.

We evaluate SPAD by provisioning cost-optimized heterogeneous clusters. End-to-end simulations on production traces for chatbot and code generation applications show that SPAD clusters reduce hardware cost by 19%-41% and TDP by 2%-17% compared to modeled baseline clusters while maintaining the same performance. Even as the models and

workloads change, SPAD can reallocate either type of chip to run either phase and still achieve an 11%-43% lower hardware cost, demonstrating the longevity of our design.

In summary, our contributions are as follows:

- Identify the inherent hardware inefficiency of modern GPUs for disaggregated LLM serving. (Section 3)
- Propose SPAD, a heterogeneous system that adopts a *less-is-more* philosophy to design specialized Prefill and Decode Chips to efficiently serve the corresponding phases of LLM inference. (Sections 4 and 5)
- Conduct extensive end-to-end cluster-level simulations demonstrating the cost-effectiveness and longevity of SPAD under various workloads and model architectures. (Sections 6 and 7)

2 Background and Related Work

We start by providing an overview of LLM architecture, hardware choices, and software-based serving techniques.

2.1 Generative LLMs

Transformers. Most modern LLMs like GPT-4 [47], DeepSeek-V3 [20], Llama-3 [25], and Grok-3 [66] are based on the decoder-only transformer architecture [62]. Each transformer block consists of two key components: a self-attention mechanism and a feed-forward neural network. Self-attention enables each token to directly compute relationships with all prior tokens in the sequence. Feed-forward networks (FFNs) process the attention-weighted tensors through linear and non-linear transformations. To support larger model sizes with cheaper inference, a sparse Mixture-of-Experts (MoE) architecture [20] has been adopted, which uses multiple FFNs, called experts, of which only a subset is dynamically activated through a communication-intensive routing mechanism [14].

Inference Phases. Generative LLMs operate in two distinct computational phases with different resource requirements. The prefill phase processes the input prompts provided by the user in one forward pass to generate the first output token and the key-value (KV) cache, which facilitates further token generation. Prefill computation is parallelized across all input tokens and has high compute utilization. The decode phase generates subsequent tokens one at a time by running forward passes with the previously generated token along with the KV cache of all prior tokens. This phase is memory-bound because generating each new token requires loading the entire model weights along with the growing KV cache.

Performance SLOs. LLM serving usually has tight latency requirements expressed as Service-Level Objectives (SLO). From a user perspective, the prefill *time to first token* (TTFT) measures the latency to receive an initial response, while the *time between tokens* (TBT) measures how quickly the decode

phase generates the rest of the response. Both TTFT and TBT are important to ensure an interactive user experience.

2.2 Hardware for LLMs

Today, LLMs are mainly served with GPUs and TPUs. To meet high resource demands, such hardware tends to maximize memory and compute capacities as much as possible.

Memory. LLM inference has high memory bandwidth and capacity requirements due to the large model sizes and KV cache sizes involved. High-end GPUs and TPUs incorporate high-bandwidth memory (HBM) [8, 15, 44] to meet these needs, and Groq uses SRAMs for even better performance [26], despite needing a large number of chips to meet capacity requirements. LPDDR and CXL memory have also been explored to reduce cost and power usage [48, 71].

Compute. Tensor operations like matrix multiplications dominate LLM execution, which has led hardware to adopt specialized components to accelerate their computation. For example, NVIDIA H100s incorporate 528 Tensor Cores to deliver nearly 1000 TFLOPs of FP16 dense matrix multiplication performance [44] and Google TPUs use large systolic arrays specifically designed for matrix computations [30–32, 40]. Non-tensor operations, such as activation or normalization functions, are usually mapped to a more general-purpose SIMD (Single Instruction Multiple Data) or vector units with lower peak performance. These tensor units and vector units can take a significant amount of die area, driving up the hardware manufacturing cost and TDP.

2.3 Efficient Serving Techniques

The computational differences between prefill and decode phases cause efficiency issues, inspiring software solutions.

Co-location. Traditional serving systems like Orca [69] run requests end-to-end on the same hardware [9, 69], batching them at request or iteration granularity. Recent systems like Sarathi [4], POD-Attention [33], and Nanoflow [75] chunk prefill phases to match hardware compute capacity and batch them with decode phases of different requests to better utilize memory bandwidth. This approach improves hardware utilization and can support very high throughput, but it incurs resource contention between prefill and decode computations [74], causing large tail TTFT and TBT latencies that can violate SLOs.

Disaggregation. Splitwise [51] and DistServe [74] disaggregate inference phases across different hardware clusters and use fast interconnects like Infiniband or NVLink to efficiently transfer KV caches between them. This approach eliminates cross-phase interference and enables phase-specific resource management and hardware choices. Prefills can optimize TTFTs by matching hardware compute capabilities, while decodes can optimize throughput by batching more requests

Table 1. LLM Serving Cluster Design Space

	Spec.	Hetero.	Disagg.	Latency	Throughput	Cost
Orca [69]	✗	✗	✗	Variable	Low	High
Sarathi [4]	✗	✗	✗	Variable	Very High	High
Groq [26]	✔	✗	?	Very Low	?	?
DistServe [74]	✗	✗	✔	Low	High	Med
Splitwise [51]	✗	✔	✔	Low	High	Med
SPAD	✔	✔	✔	Low	High	Low

Spec: specialized chip, **Hetero**: heterogeneous, **Disagg**: disaggregation-based scheduling, **Cost**: Cost per goodput. ?: Groq likely runs with small batches due to low memory capacity and has lower throughput.

to improve performance under SLOs. Due to its effectiveness, this idea has been adopted in production systems like NVIDIA Dynamo [45], Mooncake [53], and DeepSeek [20] alongside other optimizations. Some disaggregated systems also adopt co-location-based techniques, such as chunked prefills to better match hardware capacity [45, 73] and mixed batching to handle workload changes and provide better hardware utilization [51].
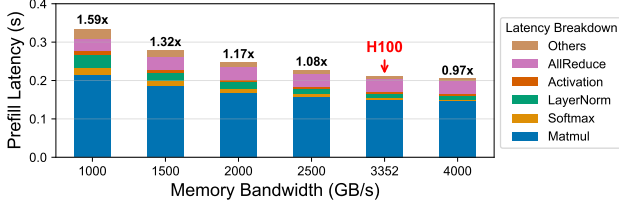
Other Techniques. Recent works improve serving efficiency using a variety of software-based techniques including efficient scheduling [52, 55, 58], memory management [24, 34, 70], kernel optimizations [18, 65, 68], quantization [22, 27, 72], power management [50], etc. These techniques are orthogonal to our work, so we do not discuss them here.

2.4 Cluster Designs and Trade-offs

LLM serving clusters can be characterized across three dimensions: hardware specialization, hardware homogeneity, and scheduling. Cluster operators usually choose the design based on hardware availability and workload requirements.

Prior work has explored different points within this design space, as shown in Table 1. Sarathi uses general-purpose GPUs, homogeneous chips, and co-location-based scheduling to enable high utilization with lower management complexity [4]. Clouds built using Google TPUs and Groq leverage specialized hardware to reduce latency and TCO, but probably use the same chips for the entire execution [2, 30]. DistServe disaggregates inference phases on homogeneous GPUs to improve throughput under SLOs [74]. Splitwise further shows that phase-specific hardware, such as H100s for prefills and A100s for decodes, can reduce overall TCO and power consumption [51]. ThunderServe further shows effective disaggregation using diverse cloud GPUs [29].

Our work explores a new point in the design space by specializing hardware for each phase in a disaggregation-based scheduling. We show that this approach substantially reduces costs while providing the same performance as existing cluster designs.

Figure 2. Simulated Prefill Latency Under Varying Memory Bandwidths. Hardware specifications are set according to a modeled H100 except for memory bandwidth. Simulated using LLMCompass [71] for an FP16 BLOOM-176B configuration with batch size 2, sequence length 1024, and tensor parallelism 8.
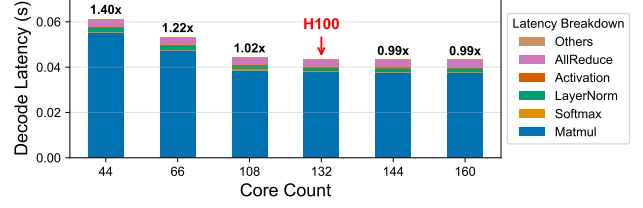

Figure 3. Simulated Decode Latency Under Varying Core Counts (SM count in NVIDIA GPUs). Hardware specifications are set according to a modeled H100 except the core count. Simulated by LLMCompass [71] with BLOOM-176B, batch size 64, sequence length 1024, and tensor parallelism 8.

3 Motivations for Phase-Specialized Hardware

Today's GPUs (also referred to as GPGPUs, General-Purpose Graphics Processing Units) are designed with a *more-is-better* philosophy to cater to the need of various workloads. This approach provides flexibility, but using the latest and greatest technologies also drives up costs. Given the dual-phase nature of LLM serving, prefill-decode disaggregation is commonly used to meet stringent latency SLOs and ensure a good user experience. In this section, we quantitatively show how GPUs are inefficient when running disaggregated prefill and decode phases. We choose BLOOM-176B [54] as a representative dense LLM because it fits on a typical 8-H100 machine, and we simulate it with FP16 precision as it can provide high production-class accuracy.

Prefills Underutilize Memory Bandwidth. Figure 2 shows how prefill phases underutilize the memory bandwidth on modeled H100 GPUs. We use LLMCompass [71] to simulate how the prefill phase latency changes as a function of the available memory bandwidth. Prefills are simulated with a batch size of 2 and a sequence length of 1024. We simulate an H100 GPU (3.35TB/s) as the baseline and sweep its memory bandwidth from 1TB/s to 4TB/s while keeping the rest of the hardware specifications the same.

Our results show that computation-intensive matrix multiplications dominate prefill time. Crucially, the prefill latency does not scale in proportion to the memory bandwidth. Even when the memory bandwidth is reduced to 2500 GB/s (about 0.75× that of H100), the latency only increases by 8%. This trend indicates that prefill phases do not require the large memory bandwidth provisioned on the H100 chip.

Decodes Underutilize Compute Capacity. Decode machines underutilize GPU compute cores when deployed using prefill-decode disaggregation due to their low arithmetic intensity. Figure 3 shows this by plotting the decode latency breakdown of BLOOM-176B while sweeping the core count (*i.e.*, the Streaming Multiprocessor count) on a simulated

H100 with LLMCompass. We use FP16, a batch size of 64, a sequence length of 1024, and a tensor parallelism of 8. We vary the number of cores from 44 to 160 while setting the other hardware specifications according to a modeled H100. We find that decode performance scales sub-linearly with an increased core count. Specifically, despite using nearly 20% fewer cores (108) than an H100 (132), the decode latency only increases by about 2%. This indicates that decode phases do not require the large compute capacity provisioned on the H100 chip for efficient execution.
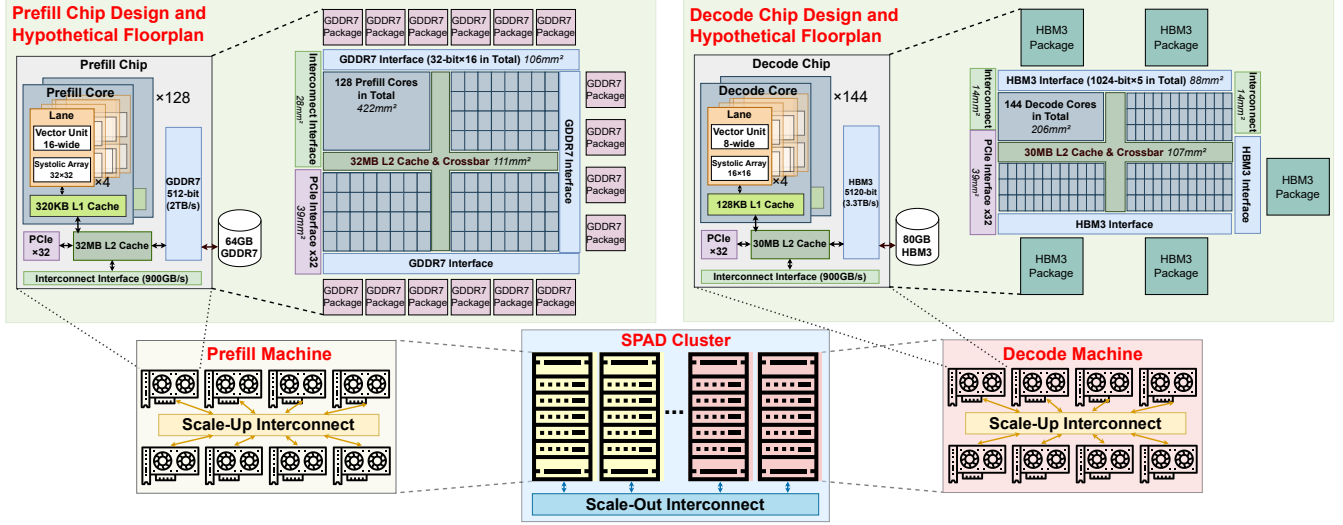
The prefill/decode bottleneck shifting under various conditions is further explored in Section B.1.

Takeaway. Underutilized hardware directly translates into increased costs for disaggregated LLM serving. These costs can be reduced by "right-sizing" regular GPU designs into separate chips to run the prefill and decode phases, respectively. The evolving models and workloads require that the hardware specialized for one phase should efficiently run the other phase to ensure flexibility [51]. In the remainder of this paper, we address these challenges and show how to tailor an existing hardware such as H100 into phase-specific designs to lower LLM serving costs at cluster scale.

4 SPAD: Overview

SPAD (**S**pecialized **P**refill **a**nd **D**ecode hardware) is a heterogeneous system incorporating specialized hardware for each inference phase to reduce disaggregated LLM serving costs at scale. In this section, we start by describing how SPAD clusters are organized and managed. In Section 5, we will describe the design methodology of our proposed chips.

Cluster Organization. Figure 4 shows an overview of a SPAD cluster. The design is similar to existing GPU-based disaggregated serving clusters but with one key difference: instead of homogeneous GPU machines, SPAD clusters consist of heterogeneous Prefill and Decode Machines optimized to run prefill and decode phases, respectively. Each proposed

Figure 4. Proposed SPAD Cluster and Chips Overview. Die area is estimated and will be further explained in Section 6.1.

Prefill/Decode Machine contains 8 Prefill/Decode Chips tailored to match the computational properties of the corresponding phase. Specifically, Prefill Chips optimize compute capabilities, whereas Decode Chips optimize memory bandwidth. Chips within a machine are connected to each other with a high bandwidth scale-up interconnect (*e.g.*, NVLink). Chips across machines are connected using a lower bandwidth scale-out interconnect (*e.g.*, Infiniband). We assume the same scale-up (900 GB/s total bandwidth per chip) and scale-out (50 GB/s per chip) interconnect as NVIDIA H100s [44].

Disaggregated Serving. LLM replicas run separately on Prefill and Decode Machines. Incoming requests are first scheduled on Prefill Machines, which run the prefill phase and transfer the computed KV caches to Decode Machines over scale-out interconnects to finish the request [51, 74].

Workload-Driven Provisioning. SPAD clusters are provisioned for a target workload by deciding the number of phase-specific chips to deploy. Due to the heterogeneity of the cluster, it is necessary to carefully select the ratio of Prefill and Decode Chips to ensure optimal performance and efficiency. Given a target model and the workload distribution, cluster operators can estimate the ideal number of machines required for each phase by sweeping the cluster design space [51]. Furthermore, operators can also use existing workload estimation techniques to allocate sufficient capacity to accommodate future workload demands [11, 13].

Adaptive Reallocation. Models and workloads can change during the multi-year lifespan of the cluster. In such cases, the provisioned ratio of Prefill and Decode Machines may not perfectly match the new requirements, leading to suboptimal performance. Since hardware is difficult to change once deployed, SPAD clusters retain efficiency by logically reallocating Prefill and Decode Machines as needed to run either phase. This consideration is fundamentally incorporated into our hardware design methodology, which enables each chip to also run the other phase cost-effectively. Techniques to further improve adaptability are discussed in Section B.2.

5 SPAD: Chip Design

In this section, we describe our *less-is-more* methodology to design SPAD chips. Using the H100 GPU as a reference design, we conduct a cost-aware architectural design space exploration to tailor specialized Prefill and Decode chips for a target workload. Crucially, we design our proposed chips with the flexibility to handle either phase, enabling them to be reallocated as workload profiles evolve. Later in Section 7.2, we show the longevity of our design through adaptive reallocation.

5.1 *Less-is-More* Design Methodology

Our goal is to design Prefill/Decode Chips that align with the characteristics of prefill/decode phases. Current mainstream LLM serving hardware, such as GPUs, fails to meet this goal due to their *more-is-better* design methodology: they tend to fit as much compute capacity as is possible onto reticle-limited dies paired with high-end HBMs to provide substantial memory bandwidth and capacity. NVIDIA H100 has a die area of 814 mm^2 with 80 GB of HBM3 and 3.35 TB/s memory bandwidth [44]. Chiplet technology has been adopted to further increase the die area to fit more compute capacity. AMD MI300X has 8 compute chiplets with 192 GB of HBM3 and 256 MB of LLC (Last-Level Cache) [7, 8]. NVIDIA B200s are reported to have two recticle-limited dies with 186 GB of HBM3E and 8 TB/s of memory bandwidth [43]. On the other hand, Groq uses SRAM instead of HBMs, achieving a memory bandwidth of up to 80 TB/s [26].
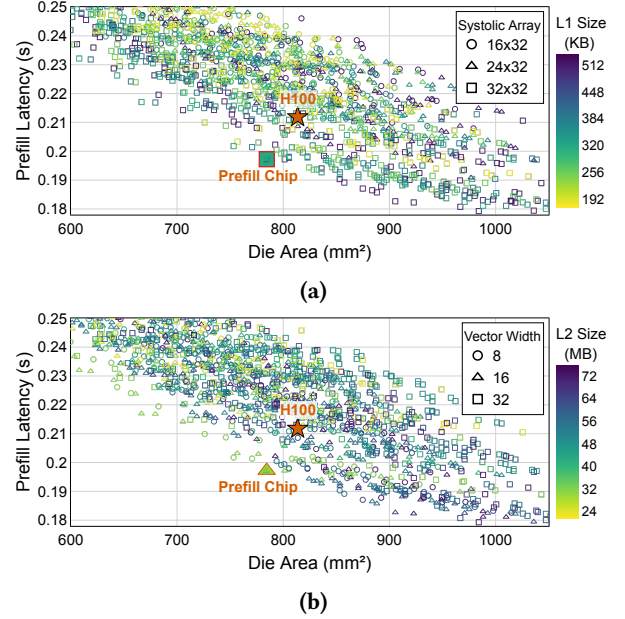
This *more-is-better* design methodology is not cost-effective for disaggregation-based LLM inference. As Section 3 shows, the prefill latency of a modeled H100 only increases by 17% if we reduce the memory bandwidth by 40%, and the decode latency only increases by 22% if we cut the compute capacity by 50%. These imply that prefills do not fully utilize the memory bandwidth offered by expensive HBMs, and decodes underutilize the compute capacity of the enormous dies. Thus, we reconsider whether *more-is-better* offers a favorable trade-off between performance and cost.

We adopt a *less-is-more* design methodology, treating cost as a first-class citizen. Our goal is to achieve the throughput and latency requirements of LLM serving at the lowest possible cost. A large die or high-end memory can increase the manufacturing cost and TDP, and a larger TDP can lead to higher power delivery and cooling equipment costs. Thus, we assess the cost-performance trade-offs with design space explorations. If an architectural component does not significantly impact performance, we consider cutting it to save cost. However, we cannot be too aggressive with cutting components since a phase-specialized chip needs to be able to run the other phase after adaptive reallocation. By carefully choosing the memory technology and using cost-aware architectural designs, our proposed Prefill/Decode Chips could achieve similar performance with lower hardware cost and TDP than H100s since their hardware characteristics align better with the arithmetic intensity of prefill/decode phases. We call this the *less-is-more* methodology, which reduces the cost/TDP per chip while allowing more chips to be deployed in clusters under the same cost/TDP budgets to achieve higher overall performance.

5.2 Prefill Chip Design

5.2.1 Memory. Figure 2 shows that even if the memory bandwidth of a modeled H100 is reduced by 40% to 2 TB/s , its simulated prefill latency will only be 17% higher. However, further reducing the bandwidth to 1.5 TB/s increases the latency by 32%. Latency breakdowns show that the latency increase is mainly caused by memory-bound non-tensor operations, such as Layer Normalization or Softmax, whose performance scales almost linearly with memory bandwidth. On the other hand, Matmul latency only increases by 16%, even when decreasing memory bandwidth from 4 TB/s to 2 TB/s. Therefore, we conclude that reduced memory bandwidth can be manageable as long as it is not decreased too much, and the increase in non-tensor operation latency can be compensated for by increasing Matmul performance. Also, since Prefill Machines only temporarily store the KV cache before transferring it to the Decode Machines, their memory capacity requirement is lower than that of the decode phase.

Following our *less-is-more* design philosophy, we propose to replace HBMs with GDDR memory as a cheaper alternative for prefill, which is commonly used in gaming GPUs [46] and desktop workstation GPUs [41].

(a)

(b)

Figure 5. Prefill Chip Design Space Exploration. Latencies of our chips and H100 are all simulated with LLMCompass [71]. Die areas of our chips are estimated and will be further explained in Section 6.1. H100 die area is reported by NVIDIA [44]. We use FP16 BLOOM-176B with tensor parallelism 8, sequence length 1024, and batch size 2. Larger systolic arrays significantly boost prefill performance. Smaller vector units have minimal performance impact.
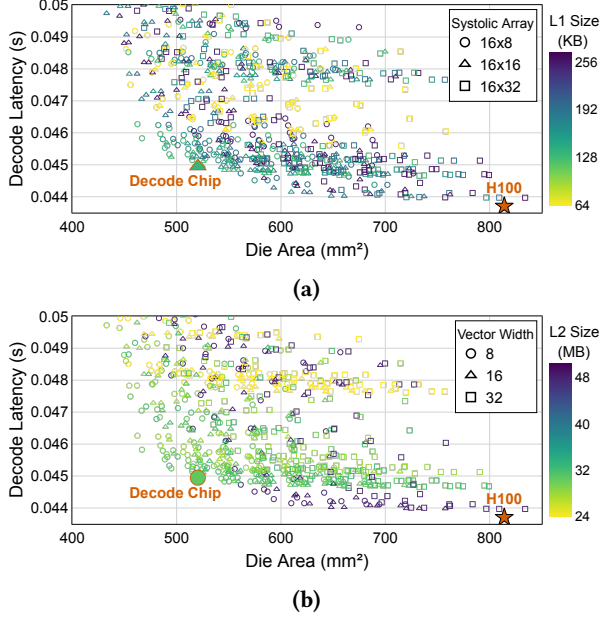
Table 2. Comparison of Memory Technologies

	Source Processor	Processor Bandwidth	Bandwidth/ Beachfront(PHY)♦	Estimated Cost♦
LPDDR5X	Apple M4 (3 nm)	120 GB/s	8 GB/s/mm	?
GDDR7	RTX 5090 (4 nm)	1792 GB/s	22 GB/s/mm	$3/GB
HBM3	H100 (4 nm)	3352 GB/s	68 GB/s/mm	$9/GB

♦ Bandwidth per beachfront of the processor PHYs, estimated based on specifications and annotated die photos [36, 44, 46, 56, 63].
♦ Cost modeling is explained in Section 6.1.

We do not choose other memory technologies like LPDDR since they have a lower bandwidth under the chip beachfront limits and do not meet the requirements of the prefill phase, as shown in Table 2. In contrast, as Table 3 shows, GDDR7 can provide 2 TB/s of bandwidth and 64 GB of capacity with a 512-bit bus and 16 packages, which meets prefill requirements. We estimate substituting HBM with GDDR could reduce memory cost by 3×, which we further explain in Section 6.1. Note that our proposed Prefill Chip has a smaller memory capacity (64 GB) compared to H100s (80 GB), and later in Section 7 our end-to-end simulations show that it is not a bottleneck for prefill phases since the KV cache is only temporarily stored.

Figure 6. Decode Chip Design Space Exploration. Latencies of our chips and H100 are all simulated with LLMCompass [71]. Die areas are estimated and will be explained in Section 6.1. H100 die area is reported by NVIDIA [44]. We use FP16 BLOOM-176B with tensor parallelism 8, sequence length 1024, and batch size 64. Our design strikes a desirable balance between performance and die area.

Table 3. SPAD Chips Compared with NVIDIA H100

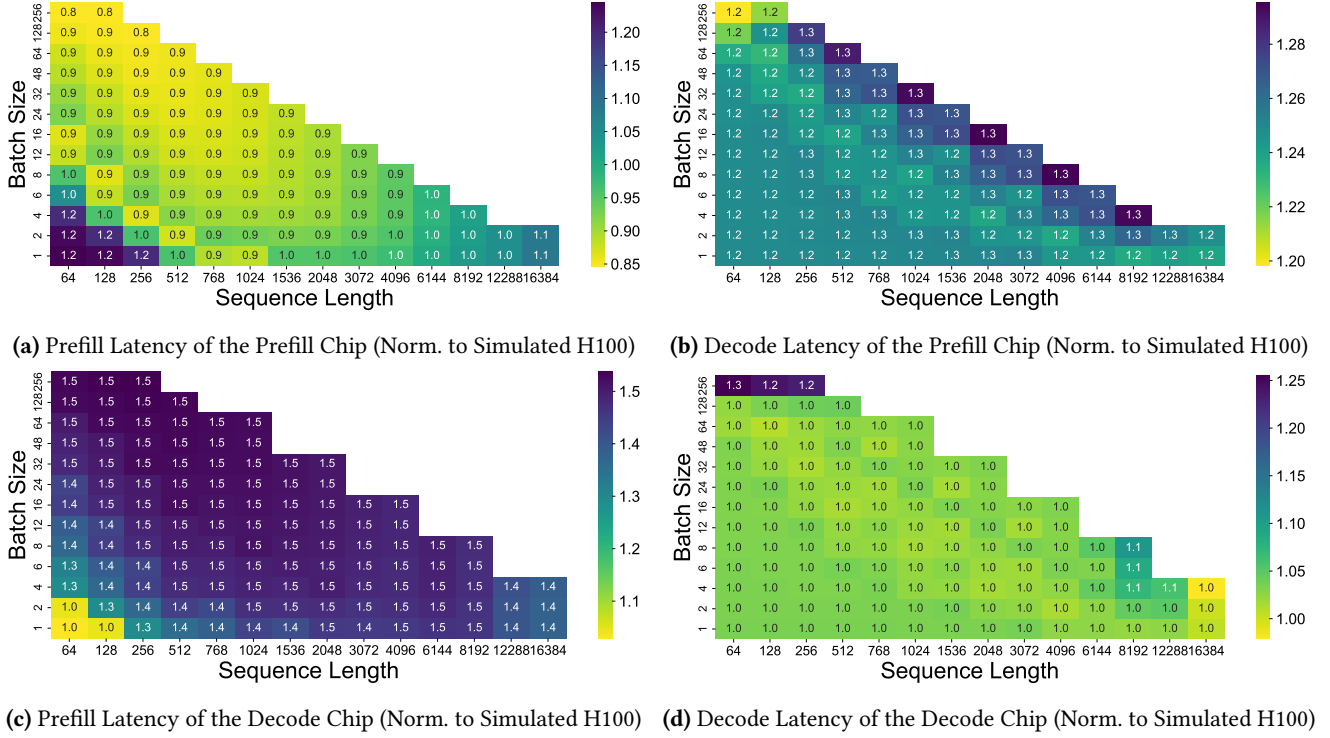| Specifications	Prefill Chip	Decode Chip	H100 [44]
Core Count	128	144	132
Lane per Core	4	4	4
Vector Width	16	8	Eq. to 32
Systolic Array	32×32	16×16	Eq. to 16×32
L1 Cache per Core	320 KB	128 KB	256 KB
L2 Cache	32 MB	30 MB	50 MB
Memory Protocol	GDDR7	HBM3	HBM3
Memory Bus Width	512-bit	5120-bit	5120-bit
Pin Speed	32 Gb/s	5.2 Gb/s	5.2 Gb/s
Memory Package Count	16	5	5
Capacity per Package	4 GB	16 GB	16 GB
Clock (Tensor)	1.83 GHz	1.83 GHz	1.83 GHz
Clock (Non-Tensor)	1.98 GHz	1.98 GHz	1.98 GHz
FP16/BF16 Tensor PFLOPs	1.92	0.54	0.99
FP32 Non-Tensor TFLOPs	32.4	18.2	66.9
Total L1 & L2 Cache Size	73 MB	48 MB	84 MB
Memory Configuration	64 GB GDDR7	80 GB HBM3	80 GB HBM3
Memory Bandwidth	2048 GB/s	3352 GB/s	3352 GB/s
Est. Die Area (@4nm)♦	784 mm^2	520 mm^2	814 mm^2
Est. Die Cost♦	$301	$187	$315
Est. Memory Cost♦	$192	$720	$720
Est. Norm. Total HW Cost♦	0.48	0.88	1
Est. TDP♦	596 W	507 W	700 W
Norm. Prefill Perf.✿	1.08	0.69	1
Norm. Decode Perf.✿	0.80	0.97	1

♦ The die area for our Prefill/Decode Chip is estimated. H100 die area is reported by NVIDIA [44]. Cost and TDP modeling is explained in Section 6.1.
✿ Performance numbers are from Fig. 7, simulated with LLMCompass [71].

5.3 Decode Chip Design

5.3.1 Memory. According to Figure 1, the decode phase is heavily memory bandwidth bound. While Prefill Machines only temporarily retain KV caches before transferring them, Decode Machines retain and use the KV cache for the rest of the request processing. Specifically, multiple requests are often batched together in decode phases to improve model weight reuse, and we need to store the KV cache for all of them. Also, since every newly generated token contributes to the KV cache, the KV cache size grows continuously until the request is complete. Therefore, the decode phase has higher memory capacity requirements to store these KV caches. Based on these observations, we choose to use HBM3 due to its high bandwidth and large capacity. Unlike Groq [26], we do not consider on-chip SRAM due to the high cost it would require to meet the memory capacity requirement.

5.3.2 Compute. Figure 3 shows that the simulated decode latency only increases by 22% even if we reduce the core count of a modeled H100 by half. Due to the *more-is-better* design methodology of H100s, the compute capacity remains underutilized for decodes. To eliminate this inefficiency, we follow our *less-is-more* philosophy and conduct a design space exploration to sweep different architectural configurations, as shown in Figure 6, identifying which hardware

5.2.2 Compute. Figure 1 shows the compute-intensive nature of the prefill phase. We observe that tensor operations, such as Matmuls, largely contribute to the compute intensity of prefill phases, and are commonly mapped to systolic arrays (or Tensor Cores in NVIDIA GPUs). On the other hand, memory-bound non-tensor operations, such as Layer Normalization, are mapped to general-purpose vector units (or CUDA Cores in NVIDIA GPUs).

Consequently, we propose to increase the tensor compute capacity to accelerate compute-bound tensor operations and reduce the non-tensor compute capacity since those non-tensor operations are memory-bound anyway. As shown in Figure 5, we use LLMCompass [71] to conduct a design space exploration for different combinations of core counts, vector widths, systolic array sizes, and cache sizes. We find that increasing the size of the systolic arrays significantly increases prefill performance while decreasing the size of the vector units has minimal performance impact. The L1 cache size is increased to accommodate the larger systolic arrays. We also decrease the L2 cache size as we find that approximately 30 MB of L2 is enough for LLM inference, and there is a diminishing return in further increasing L2 sizes.

(a) Prefill Latency of the Prefill Chip (Norm. to Simulated H100)

(b) Decode Latency of the Prefill Chip (Norm. to Simulated H100)

(c) Prefill Latency of the Decode Chip (Norm. to Simulated H100)

(d) Decode Latency of the Decode Chip (Norm. to Simulated H100)

Figure 7. Chip Performance Under Various Batch Sizes and Sequence Lengths. Lower is better. Latencies of our proposed chips and H100 are all simulated with LLMCompass [71] modeling FP16 BLOOM-176B with tensor parallelism 8. Only the combinations that fit the memory capacity are shown. A sensitivity study on parallelism strategies is further shown in Fig. 11 in Section A.

resources and to what extent can be cut without impacting performance.

We find that smaller systolic arrays and vector units are more efficient than larger ones for decode. Due to the low arithmetic intensity and memory-bound nature, large systolic arrays and vector units bring very marginal performance gains since decode phases cannot fully utilize them. Therefore, our proposed Decode Chip adopts a systolic array size of 16×16 and a vector width of 8. We did not further reduce tensor performance because the area savings were outweighed by the significant slowdown of running prefill, which can affect flexibility after reallocation.

We find that smaller caches are sufficient for decode. Large caches improve performance through better memory reuse. However, since decode phases are memory-bound in reading model weights and KV caches, larger caches do not help much for these streaming memory accesses. Compared to a modeled H100, we cut the L1 size by 50% and the L2 size by 40%.

5.4 Summary

A comparison of our proposed Prefill/Decode Chips with H100s is summarized in Table 3. The extra complexity arising from heterogeneous chips is discussed in Section B.3.

Prefill Chip. Compared to a modeled H100, our proposed Prefill Chip roughly doubles the tensor performance while maintaining a similar die area by reducing the unessential non-tensor performance by half and cutting down L2 cache size. A hypothetical floorplan is shown in Figure 4.

Figures 7a and 7b show the performance of the proposed Prefill Chip simulated with LLMCompass [71]. Compared to a modeled H100, it is 8% faster for prefills on average: the tensor operations are faster due to larger systolic arrays, but the non-tensor operations are slower due to reduced memory bandwidth. The hardware cost is reduced by 52%, mainly from substituting HBMs with cheaper GDDR7 memory.

Our proposed Prefill Chip can be slightly slower than modeled H100s on very few total batched tokens (≤ 256 tokens) or very long input prompts (≥ 12288 tokens). Very short input sequences have little weights reuse and low arithmetic intensity. For very long input sequences, the Softmax operation inside the attention mechanism becomes more dominant due to its quadratic complexity with respect to sequence length. Since our Prefill Chip has less memory bandwidth and non-tensor compute capability, Softmax becomes the new bottleneck. This bottleneck could be alleviated by chunking long prefills [4] or using sequence parallelism [64].

Decode Chip. The key specifications of our proposed Decode Chip are summarized in Table 3, and a hypothetical floorplan is shown in Figure 4. Compared to a modeled H100, our proposed Decode Chip reduces the die area by 36% and lower the TDP by 28%, primarily due to its lower compute capacity and smaller caches. Figure 7d shows that it still achieves 97% of the decode performance of a modeled H100 on average. Our Decode Chip can be slower for very large batch sizes (≥ 256) due to increased arithmetic intensities. However, such large batch sizes can be rare in production, especially for latency-sensitive workloads, due to the HBM capacity limit and the diminishing return of batching.

6 Evaluation Methodology

6.1 Cost and TDP Modeling

Total Hardware Cost. We account for the combined manufacturing costs of the die and memory. We exclude other costs such as masking, packaging, and design, since they are not commonly disclosed and estimates vary widely. Additionally, when manufacturing at scale, one-time mask and design costs can be amortized across all dies.

We modify LLMCompass' area model to model the die areas of our proposed Prefill Chip and Decode Chip guided by annotated H100 die photos [36]. For our proposed designs, we assume a 10% area overhead to account for white space and disabled defective components and a TSMC 4nm process node. We assume 4nm wafer costs of $20,000 per 300mm wafer, which aligns with estimates for modern process nodes [39, 61, 67]. To find die costs, we calculate the number of dies that can fit a single wafer.

For device memory costs, we estimate $3 per GB for GDDR7 based on current GDDR6 spot prices [60]. HBM pricing is less transparent, and estimates vary between $10 to $35 per GB [10, 21]. For our cost model, we assume that HBM costs are between 2×-4× the cost of GDDR based on publicly disclosed industry estimates [23]. In Section A Table 9, we further explore different HBM3 cost assumptions. Note that even the highest 1:4 ratio of $12 per GB is on the lower end of cost estimates for HBM3.

Table 3 details die area estimates for the proposed Prefill/Decode Chips. Based on these, we calculate die costs, memory costs (assuming a 1:3 GDDR7:HBM3 cost ratio), and total hardware costs for the three devices.

TDP Modeling for our Prefill/Decode Chip. The H100 has a TDP of 700 W [44], and we assume a 10% TDP overhead to account for VRM conversion loss and other peripherals [5, 35]. We assume each HBM package has a power consumption of 30 W [57]. Based on these, we assume the H100 die itself excluding HBMs has a TDP of $700 \times 90\% - 30 \times 5 = 480W$, and we assume our Prefill/Decode Chip has the same power density as an H100 die. GDDR7 power consumption is estimated by the reported 4.5 pJ/bit from Micron [37].
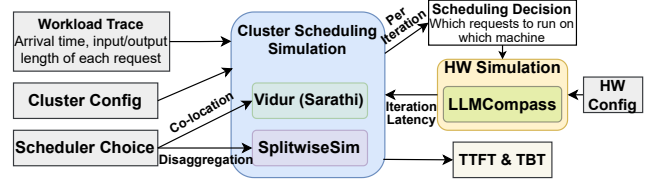

Figure 8. End-to-End Simulation Setup

6.2 End-to-end Simulations

We perform end-to-end evaluations that include both hardware architectural simulation and cluster-level scheduling simulation with workload traces to estimate how our designs translate into performance and cost improvements at scale. An overview of our simulation setup is shown in Figure 8.

Given a cluster configuration and a workload trace, the scheduler will dispatch each request to a machine within the cluster. We explore two such scheduling approaches: SplitwiseSim [51] as an implementation for disaggregated scheduling (Splitwise), and Vidur [3] as an implementation for co-location-based scheduling (Sarathi [4]). The fidelity of these scheduler implementations has been explored in their corresponding publications [3, 51].

At each iteration, cluster simulators make scheduling decisions to assign requests to machines, and these iteration-level request batches are fed into LLMCompass [71] to estimate how long it will take each machine to finish one iteration for the request batch scheduled upon it.

We extended LLMCompass to support H100 modeling and new models with a similar error rate as in the original paper [71]. We also extended SpitwiseSim and Vidur to use LLMCompass as their performance model. With LLMCompass serving as the unified architectural performance model across different schedulers and hardware, fair comparisons are achieved. **All results in this paper are simulated, not measured on real hardware.** In other words, we model the execution rather than performing the computation with actual parameter values.

6.3 Experimental Setup

Models. We evaluated three open-source models with different sizes, model architectures, and deployment strategies: ① BLOOM-176B [54] uses Multi-Head Attention [62] and we deployed it with FP16 and a tensor parallelism of 8 (TP=8). ② Llama3-70B [25] uses Grouped-Query Attention [6] with smaller KV cache footprints, and we deployed it with FP16 and TP=4. ③ DeepSeek-V2-236B [19] uses DeepSeekMoE [17] with Multi-head Latent Attention to compress the KV cache, and we deployed it with FP8 and expert parallelism 8 (EP=8).

Workloads. We use open-source request traces from Microsoft [12], representing two common LLM applications: *coding* (code completion) and *conversation* (chatbot). The

Table 4. Provisioning Results Summary

	Coding (70 req/s)			Conversation (70 req/s)		
	HW Requirement✤	Norm. HW Cost◆	Norm. TDP◆	HW Requirement✤	Norm. HW Cost◆	Norm. TDP◆
Sarathi	36 H100	36	36	34 H100	34	34
Splitwise-homo	25 H100	25	25	23 H100	23	23
Splitwise-hetero♠	21 H100 + 9 A100	25.5	25.5	13 H100 + 32 A100	29	29
Splitwise-pcap	21 H100 + 4 450W H100	25	23.6	6 H100 + 21 450W H100	27	19.5
SPAD (P+D)	18 Prefill + 7 Decode	**14.7**	**20.4**	8 Prefill + 17 Decode	**18.7**	**19.1**

✤ Minimum number of modeled 8-chip machines to meet the SLOs with BLOOM-176B. The unit is 8-chip machines, e.g., 36 H100 refers to 36 modeled 8-H100 machines and 18 Prefill refers to 18 8-Prefill-Chip machines. ◆ Normalized to the HW cost/TDP of a modeled 8-H100 machine. ♠ Assumes that A100s have half the hardware cost and TDP of H100s.

Table 5. Latency SLOs. Defined as the slowdown relative to running the request on modeled H100s without batching.

SLOs✤	P90 TBT	P90 TTFT	P99 TBT	P99 TTFT
Loose/**Normal**/Tight	2.5×/**2**/1.5×	4×/**3**/2×	6×/**5**/3×	8×/**6**/4×

✤ Normal SLOs are used unless otherwise specified.

coding workload has long input prompts (median: 1500 tokens) and short output sequences (median: 13 tokens), while the conversation workload has shorter input prompts (median: 1020 tokens) and longer output sequences (median: 129 tokens).

SLOs. We evaluate the maximum throughput that can be supported under normalized P90 and P99 TTFT and TBT SLOs, shown in Table 5. Similar to prior work [51], SLOs are defined relative to the execution latencies of the same request without any batching or contention on modeled H100s.

Baselines. We use Splitwise [51] and Sarathi [4] as baseline GPU-driven LLM serving cluster systems. Splitwise is a disaggregation-based system, and we compare SPAD with three of its variants: Splitwise-homo with H100s, Splitwise-pcap that uses H100s for prefill and hypothetical power-capped H100s (450W TDP)[1] for decode, and Splitwise-hetero with H100s for prefill and A100s for decode. Sarathi is a co-location-based system, and we configure it with modeled H100s. All baselines are evaluated as described in Section 6.2.

7 Results

7.1 Cluster Provisioning

We start by evaluating the efficacy of SPAD clusters when provisioned for a specific workload. Table 4 summarizes the

---

[1]In order to have a baseline that optimizes for TDP, we assume a hypothetical power-capped 450 W TDP H100 for decode with 76% of the peak FP16 tensor TFLOPs while retaining the same memory and interconnect specifications as the original 700 W H100. Since we do not have access to H100 power configurations and its dynamic voltage and frequency scaling (DVFS) implementation, our LLMCompass simulation is based on the hardware specification of the 350 W NVIDIA H100 PCIe [44]. We substitute its low-end memory and NVLink interconnect with those of the original 700W H100, which we assume adds 100 W TDP.
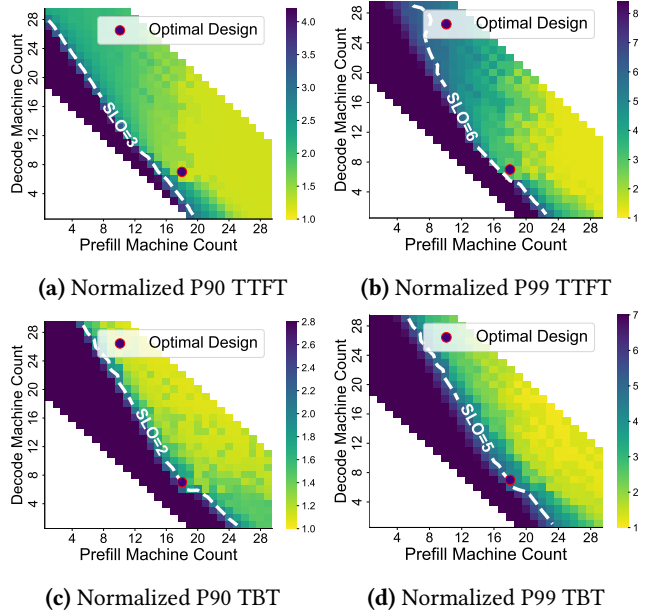

(a) Normalized P90 TTFT **(b)** Normalized P99 TTFT

(c) Normalized P90 TBT **(d)** Normalized P99 TBT

Figure 9. Provisioning Results with Coding Trace for SPAD. The optimal design has 18 prefill and 7 decode machines.

provisioning results for BLOOM-176B using the coding and conversation workloads with a target request rate of 70 req/s.

Coding. Compared to the best baseline, SPAD saves the hardware cost by **41%** and TDP by **13%**. Sarathi needs at least 36 modeled 8-H100 machines to meet all SLOs, while Splitwise needs at least 25 modeled 8-H100 machines. The minimal hardware requirement is derived by sweeping machine count as shown in Figures 12 and 13a in Section A. Due to prefill-decode interference, Sarathi can be unsuitable for low-latency workloads compared to disaggregated serving. Splitwise-hetero needs at least 21 modeled 8-H100 and 9 modeled 8-A100 machines, which does not improve cost-effectiveness because although the TFLOPS-to-memory-bandwidth ratio of A100 is closer to the theoretical arithmetic intensity of decode, the absolute bandwidth and TFLOPS are significantly lower, making it harder to meet strict latency SLOs. Figure 9 shows that SPAD needs the same amount of

Table 6. Provisioning Results under Various SLOs.

Workloads SLOs	Coding (70 req/s)			Conversation (70 req/s)		
	Loose	Normal	Tight	Loose	Normal	Tight
Sarathi (H100)♦	33	36	45	31	34	40
Splitwise (H100)♦	24	25	27	<u>22</u>	<u>23</u>	<u>27</u>
Splitwise (H100+A100)♦	20+9	21+9	27+0	13+20	13+32	27+0
Splitwise (H100+pcap)♦	<u>19+5</u>	<u>21+4</u>	<u>23+4</u>	3+23	6+21	11+23
SPAD (P+D)♦	18+6	18+7	21+7	8+17	8+17	13+14
Hardware Saving❖	42%	41%	40%	15% | 28%	19% | 31%	32% | 46%
TDP Saving❖	11%	13%	10%	13% | -8%	17% | 2%	21% | 18%

♦ The unit is modeled 8-chip machines.
❖ Hardware/TDP saving of SPAD compared to pareto-optimal baselines (<u>underlined</u>). There can be more than one Pareto-optimal baseline: Splitwise (H100) has lower hardware cost but higher TDP than Splitwise (H100+pcap).

Table 7. SPAD Reallocation After Changing Workload (Model Remains Unchanged: BLOOM-176B)

Provisioned Cluster (P+D)❖	Reallocated Workload	Reallocated Throughput	Min. HW❖♦ for Splitwise	(HW, TDP) Saving
18P+7D♠	Conversation	55 req/s	19 H100	(23%, -7%)
8P+17D✿	Coding	60 req/s	21 H100	(11%, 9%)

❖ The unit here is modeled 8-chip machines.
♦ The minimum hardware required to achieve the reallocated throughput. Sarathi performs consistently worse than Splitwise and is not shown.
♠ Initially provisioned for Coding (70 req/s).
✿ Initially provisioned for Conversation (70 req/s).

machines as Splitwise-homo, demonstrating the effectiveness of our *less-is-more* design methodology.

Conversation. Compared to the two Pareto-optimal baselines, SPAD saves hardware cost and TDP by **(19%, 17%)** relative to Splitwise-homo, and by **(31%, 2%)** relative to Splitwise-pcap. Splitwise-pcap saves TDP, but not hardware cost. Figure 14 in Section A shows the detailed provisioning results for SPAD. Compared to the coding workload, the conversation workload requires more Decode Machines due to its longer output sequences.

Changing SLOs. Table 6 shows the minimum hardware requirements to sustain 70 req/s under three sets of SLOs from loose to tight (as defined in Table 5), demonstrating SPAD's consistent performance under various SLOs. The Normal SLOs are the ones used in previous experiments.

7.2 Cluster Reallocation

Next, we evaluate how well an already provisioned SPAD cluster performs after reallocation when the workloads and models change. In this section, we compare with the modeled 700W TDP H100 as the baseline hardware due to its balanced performance across various workloads and SLO settings in the provisioning experiments.

Changing Workloads. Table 7 and Fig. 10a show that the cluster that was initially provisioned for the coding workload at 70 req/s can be repurposed for the conversation workload

Table 8. SPAD Reallocation After Changing the Model (Workload Remains Unchanged)
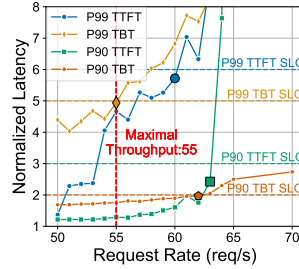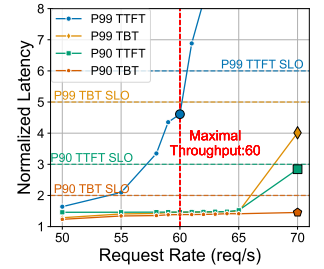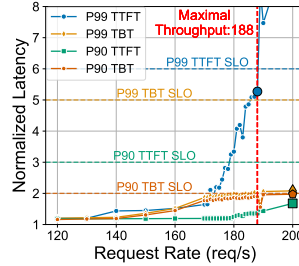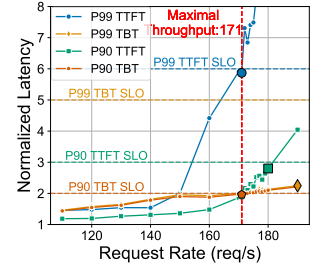
Provisioned Cluster (P+D)❖	Reallocated Model	Reallocated Throughput	Min. HW❖♦ for Splitwise	(HW, TDP) Saving
18P+7D♠	Llama3-70B	188 req/s	26 H100	(43%, 22%)
8P+17D✿	Llama3-70B	171 req/s	27 H100	(31%, 29%)
18P+7D♠	DeepSeek-V2	103 req/s	23 H100	(36%, 11%)
8P+17D✿	DeepSeek-V2	183 req/s	24 H100	(22%, 20%)

❖ The unit here is modeled 8-chip machines.
♦ The minimum hardware required to achieve the reallocated throughput.
♠ Initially provisioned for Coding (70 req/s) with BLOOM-176B.
✿ Initially provisioned for Conversation (70 req/s) with BLOOM-176B.

(a) Coding-Opt Cluster Running Conversation (BLOOM-176B)

(b) Conversation-Opt. Cluster Running Coding (BLOOM-176B)

(c) BLOOM-Opt. Cluster Running Llama3-70B (Coding)

(d) BLOOM-Opt. Cluster Running Llama3-70B (Conversation)

Figure 10. SPAD Clusters After Reallocation. Markers indicate the highest feasible request rate under each SLO; their minimum is the maximum supported cluster throughput.

at 55 req/s after reallocation, where 8 Prefill Machines are reallocated to run decode. The baseline needs at least 19 modeled 8-H100 machines to achieve the same throughput, so SPAD can still save the hardware cost by **23%** at the cost of **7%** larger TDP. Although the Prefill Chip runs decode at a reduced hardware efficiency, its hardware cost saving by using GDDR instead of HBM is still significant.

Table 7 and Fig. 10b show that the cluster initially provisioned for the conservation workload at 70 req/s can be reallocated to support 60 req/s for the coding workload, where 14 Decode Machines are reallocated for prefill. The baseline needs at least 21 modeled 8-H100 machines to achieve the same throughput, so SPAD still reduces the hardware cost by **11%** and TDP by **9%**. We attribute this benefit to the fact that

our Decode Chip is designed to run prefill phases reasonably well, so it does not sacrifice the performance much.

Changing Models. Table 8 shows that when the model evolves from Multi-Head Attention (MHA) to Grouped-Query Attention (GQA) and Multi-head Latent Attention (MLA) and MoE (DeepSeek-V2), the cluster initially provisioned for BLOOM-176B can also serve Llama3-70B and DeepSeek-V2 efficiently, achieving **22%-43%** hardware cost saving and **11%-29%** TDP saving compared to the modeled H100 baseline.

For Llama3-70B, the cost savings tend to be greater than for running BLOOM-176B, mainly because GQA enables Key/Value sharing inside each group, which increases the arithmetic intensity and favors our proposed Prefill Chips. For DeepSeek-V2, the cost savings are smaller mainly because it is a sparse MoE model and has smaller arithmetic intensity. In DeepSeek-V2, tokens are dispatched to 160 different routed experts. Since each token only activates a subset of total weights, the per-expert weight reuse across different tokens is smaller compared to dense models.
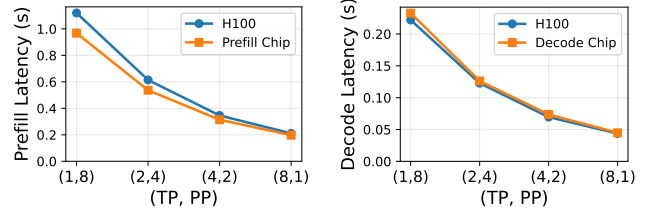
8 Conclusion

This work introduces SPAD, a heterogeneous system to accelerate disaggregation-based LLM serving. Leveraging the dual-phase nature of LLM inference, we adopt a *less-is-more* philosophy to design cost-effective Prefill and Decode Chips tailored to their distinct computational characteristics. Compared to modeled H100s, our proposed Prefill Chips deliver 8% higher prefill performance on average at 52% lower hardware cost, while our proposed Decode Chips achieve 97% of the decode performance with 28% lower TDP. End-to-end simulations show that SPAD reduces hardware costs by 19%-41% and TDP by 2%-17% compared to modeled baseline clusters while maintaining the same performance. As models and workloads change, SPAD can perform an adaptive chip reallocation and still achieve 11%-43% lower hardware costs, demonstrating the longevity of our design.

Acknowledgments
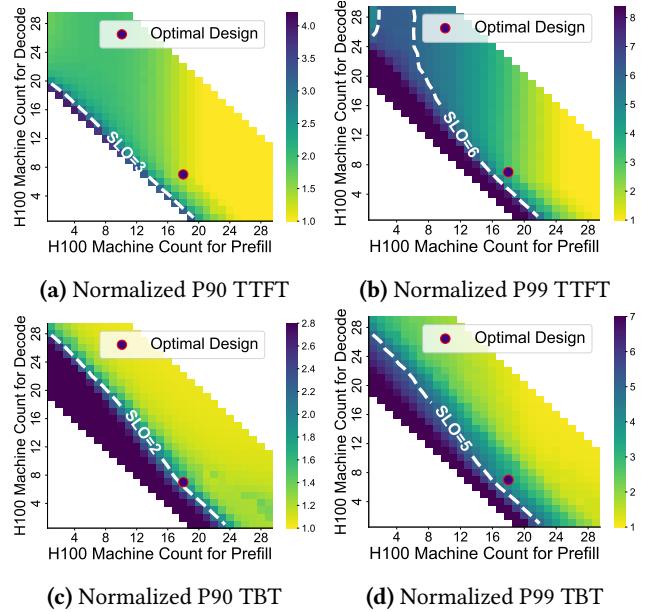
A Supplementary Results

Figure 11. Chip Performance Under Various Tensor (TP) and Pipeline Parallelism (PP). Latencies of our chips and H100 are all simulated with LLMCompass [71] and FP16 BLOOM-176B with sequence length 1024 and batch size 2 and 64 for prefill and decode respectively. Our proposed chips perform consistently under various model parallelisms.

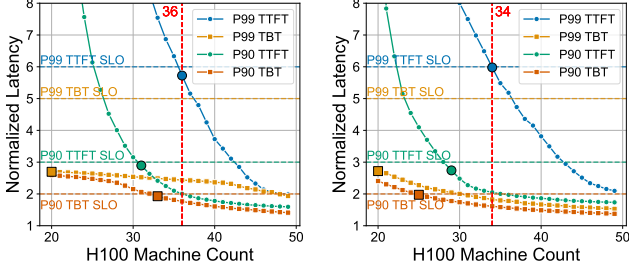**Table 9.** Chip Cost under Various HBM Cost Assumptions

HBM Cost Assumptions	$6/GB	$9/GB✤	$12/GB
Estimated HBM Cost	$480	$720	$960
Estimated Decode Chip Cost	$667	$907	$1147
Estimated H100 Cost	$795	$1035	$1275

✤ We use $9/GB for HBM cost in the the paper.

(a) Normalized P90 TTFT   **(b)** Normalized P99 TTFT

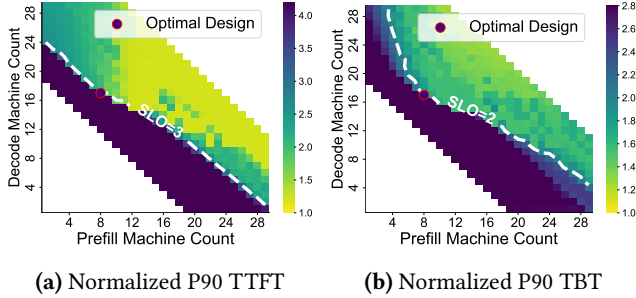**(c)** Normalized P90 TBT   **(d)** Normalized P99 TBT

Figure 12. Provisioning Results with Coding Trace (70 req/s) and BLOOM-176B for Splitwise-homo. At least 25 modeled 8-H100 machines are required to meet all the SLOs. Markers indicate one of the Pareto-optimal designs with 18 modeled 8-H100 machines for prefill and 7 for decode. All results here are simulated as explained in Section 6.2.

(a) Coding (70 req/s): 36 H100 machines to meet all SLOs **(b)** Conversation (70 req/s): 34 H100 machines to meet all SLOs

Figure 13. Provisioning Results with Sarathi (BLOOM-176B). Markers indicate the minimum modeled 8-H100 machine count to meet each SLO. All results here are simulated as explained in Section 6.2.

(a) Normalized P90 TTFT **(b)** Normalized P90 TBT

Figure 14. Provisioning Results with Conversation Trace for SPAD. The optimal design has 8 prefill and 17 decode machines. P99 TTFT/TBT figures are similar and not shown.
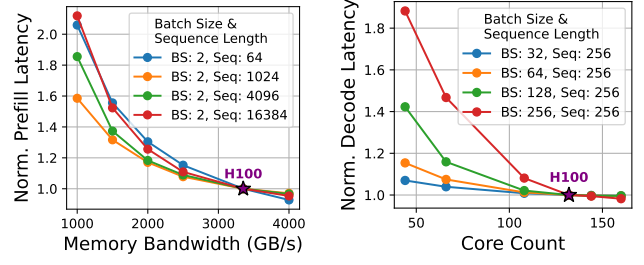
B Extended Discussion

B.1 Prefill/Decode Bottleneck Shifting

In Section 3, we show that prefill is compute-bound and decode is memory-bandwidth-bound under a common batch size and sequence length setting. However, these bottlenecks can dynamically shift under various conditions:

Prefill with very short sequences can shift towards memory-bandwidth-bound due to limited data reuse. Fig. 15a shows that the prefill latency is more sensitive to memory bandwidth when the sequence length is very small (e.g., 64). As a result, on the bottom left corner in Fig. 7a, our proposed Prefill Chip can be slower than the modeled H100s when the batched token size is small.

Prefill with long sequences can shift towards memory-bound. The quadratic complexity of the attention puts more pressure on memory. Fig. 15a shows that the prefill latency is more sensitive to memory bandwidth with long sequences. On the bottom right in Fig. 7a, the performance improvement of our proposed Prefill Chip diminishes and eventually reverses. Memory capacity becomes another bottleneck with

(a) Normalized Prefill Latency. Hardware specifications are set according to a modeled H100 except for memory bandwidth. **(b)** Normalized Decode Latency. Hardware specifications are set according to a modeled H100 except the core count.

Figure 15. Prefill/Decode Latency Under Various Settings. Simulated using LLMCompass [71] for an FP16 BLOOM-176B with tensor parallelism 8. All results are normalized to simulated H100s. (a) Prefill shifts towards memory-bandwidth-bound under very long or short sequences. (b) Decode shifts towards compute-bound under large batch sizes.

long sequences due to increasing KV cache size: For FP16 BLOOM-176B, assuming 90% of the memory capacity reserved for model weights and KV cache, 8 of our proposed Prefill Chips (64GB each) can store roughly 35K tokens, while 8 modeled H100s (80GB each) can store around 66K tokens.

Decode with large batch sizes can shift towards compute-bound. Fig. 15b shows that decode can be more sensitive to compute capacity with large batch sizes due to increased arithmetic intensity. On the top left corner of Fig. 7d, our proposed Decode Chips are slower than modeled H100s under batch size 256. However, this condition can be rare due to KV cache sizes and latency constraints.

B.2 Adaptability to Highly Variable Workloads

As shown in Section 7.2 and Tables 7 and 8, we rely on cluster reallocation to repurpose our proposed chips when the Prefill-to-Decode ratio of the workload changes dramatically. To further improve adaptability and robustness, we provide two recommendations:

Buffer pool. SPAD can be combined with a buffer pool composed of existing balanced hardware such as NVIDIA H100s. When the prefill and decode demands change, different portions of this pool can be allocated to prefill and decode according to the changing demands dynamically. We envision most of the workload still served by our proposed chips for the hardware cost and TDP benefits, with the buffer pool mainly to account for future workload variability.

Load predictor. At the orchestrator level, SPAD can be further combined with a runtime load predictor such as ARIMA (Autoregressive Integrated Moving Average) or Meta

Prophet [59], which has been incorporated in industry frameworks such as NVIDIA Dynamo Planner [1, 45]. At each time interval, the load predictor estimates the prefill loads and decode loads, which can be used to guide SPAD cluster reallocation under highly variable workloads.

B.3 Extra Complexity of Heterogeneous Chips

The *less-is-more* design methodology illustrates how to take an existing LLM serving hardware and tailor it into two specialized chips for different phases. In this design process, the baseline design and the derived prefill/decode chips share architectural similarity, minimizing the compatibility issue with existing software stacks and alleviating the extra NRE and software implementation costs. For example, the different systolic array sizes between the prefill and decode chips may require tuning tiling parameters, rather than developing two entirely different software implementations. There is no fundamental difficulty in supporting existing software frameworks and inference-time optimizations like quantization with a minimal amount of engineering effort involved. Moreover, the increasing LLM inference demand can amortize these costs through massive production and deployment of these chips.

References

[1] 2025. feat SLA-based Planner. https://github.com/ai-dynamo/dynamo/pull/1420.

[2] Dennis Abts, Garrin Kimmell, Andrew Ling, John Kim, Matt Boyd, Andrew Bitar, Sahil Parmar, Ibrahim Ahmed, Roberto DiCecco, David Han, John Thompson, Michael Bye, Jennifer Hwang, Jeremy Fowers, Peter Lillian, Ashwin Murthy, Elyas Mehtabuddin, Chetan Tekur, Thomas Sohmers, Kris Kang, Stephen Maresh, and Jonathan Ross. 2022. A software-defined tensor streaming multiprocessor for large-scale machine learning. In *Proceedings of the 49th Annual International Symposium on Computer Architecture* (New York, New York) *(ISCA '22)*. Association for Computing Machinery, New York, NY, USA, 567–580. doi:10.1145/3470496.3527405

[3] Amey Agrawal, Nitin Kedia, Jayashree Mohan, Ashish Panwar, Nipun Kwatra, Bhargav S. Gulavani, Ramachandran Ramjee, and Alexey Tumanov. 2024. Vidur: A Large-scale Simulation Framework for LLM Inference. In *Proceedings of Machine Learning and Systems*, P. Gibbons, G. Pekhimenko, and C. De Sa (Eds.), Vol. 6. 351–366. https://proceedings.mlsys.org/paper_files/paper/2024/file/b74a8de47d2b3c928360e0a011f48351-Paper-Conference.pdf

[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 USENIX Conference on Operating Systems Design and Implementation* (Santa Clara, CA, USA) *(OSDI'24)*. USENIX Association, USA, Article 7, 18 pages. https://dl.acm.org/doi/10.5555/3691938.3691945

[5] Mohamed Ahmed, Chao Fei, Fred C. Lee, and Qiang Li. 2016. High efficiency two-stage 48V VRM with PCB winding matrix transformer. In *2016 IEEE Energy Conversion Congress and Exposition (ECCE)*. 1–8. doi:10.1109/ECCE.2016.7855150

[6] Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. 2023. GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints. arXiv:2305.13245 [cs.CL] https://arxiv.org/abs/2305.13245

[7] AMD. 2023. AMD Instinct™ MI300X Accelerator Data Sheet. https://www.amd.com/content/dam/amd/en/documents/instinct-tech-docs/data-sheets/amd-instinct-mi300x-data-sheet.pdf Accessed: 2025-04-10.

[8] AMD. 2024. AMD Instinct MI300X Accelerator. https://www.amd.com/en/products/accelerators/instinct/mi300/mi300x.html. Accessed: 2025-03-27.

[9] Reza Yazdani Aminabadi, Samyam Rajbhandari, Ammar Ahmad Awan, Cheng Li, Du Li, Elton Zheng, Olatunji Ruwase, Shaden Smith, Minjia Zhang, Jeff Rasley, and Yuxiong He. 2022. DeepSpeed-inference: enabling efficient inference of transformer models at unprecedented scale. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis* (Dallas, Texas) *(SC '22)*. IEEE Press, Article 46, 15 pages. https://dl.acm.org/doi/abs/10.5555/3571885.3571946

[10] Astute Analytica. 2025. High Bandwidth Memory Market to Worth Over US$ 5,810.5 Million By 2033. https://www.globenewswire.com/news-release/2025/01/31/3018789/0/en/High-Bandwidth-Memory-Market-to-Worth-Over-US-5-810-5-Million-By-2033-Astute-Analytica.html

[11] Georgios Andreadis, Fabian Mastenbroek, Vincent van Beek, and Alexandru Iosup. 2022. Capelin: Data-Driven Compute Capacity Procurement for Cloud Datacenters Using Portfolios of Scenarios. *IEEE Transactions on Parallel and Distributed Systems* 33, 1 (2022), 26–39. doi:10.1109/TPDS.2021.3084816

[12] Azure. 2024. *Azure Public Dataset: Azure LLM Inference Trace 2023*. https://github.com/Azure/AzurePublicDataset/blob/master/AzureLLMInferenceDataset2023.md

[13] Mat Brown. 2024. Sizing and Capacity planning for Nutanix Cloud Infrastructure. https://www.nutanix.com/tech-center/blog/hybrid-cloud-sizing-and-capacity-planning

[14] Weilin Cai, Juyong Jiang, Fan Wang, Jing Tang, Sunghun Kim, and Jiayi Huang. 2025. A Survey on Mixture of Experts in Large Language Models. *IEEE Transactions on Knowledge and Data Engineering* 37, 7 (2025), 3896–3915. doi:10.1109/TKDE.2025.3554028

[15] Google Cloud. 2024. Cloud TPU v5p. https://cloud.google.com/tpu/docs/v5p. Accessed: 2025-03-27.

[16] Google Cloud. 2024. Cloud TPU v6e. https://cloud.google.com/tpu/docs/v6e. Accessed: 2025-03-27.

[17] Damai Dai, Chengqi Deng, Chenggang Zhao, R. X. Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y. Wu, Zhenda Xie, Y. K. Li, Panpan Huang, Fuli Luo, Chong Ruan, Zhifang Sui, and Wenfeng Liang. 2024. DeepSeekMoE: Towards Ultimate Expert Specialization in Mixture-of-Experts Language Models. arXiv:2401.06066 [cs.CL] https://arxiv.org/abs/2401.06066

[18] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. FLASHATTENTION: fast and memory-efficient exact attention with IO-awareness. In *Proceedings of the 36th International Conference on Neural Information Processing Systems* (New Orleans, LA, USA) *(NIPS '22)*. Curran Associates Inc., Red Hook, NY, USA, Article 1189, 16 pages.

[19] DeepSeek-AI, Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Hao Yang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jin Chen, Jingyang Yuan, Junjie Qiu, Junxiao Song, Kai Dong, Kaige Gao, Kang Guan, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruizhe Pan, Runxin Xu, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye,

Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Size Zheng, T. Wang, Tian Pei, Tian Yuan, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Liu, Xin Xie, Xingkai Yu, Xinnan Song, Xinyi Zhou, Xinyu Yang, Xuan Lu, Xuecheng Su, Y. Wu, Y. K. Li, Y. X. Wei, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Zheng, Yichao Zhang, Yiliang Xiong, Yilong Zhao, Ying He, Ying Tang, Yishi Piao, Yixin Dong, Yixuan Tan, Yiyuan Liu, Yongji Wang, Yongqiang Guo, Yuchen Zhu, Yuduan Wang, Yuheng Zou, Yukun Zha, Yunxian Ma, Yuting Yan, Yuxiang You, Yuxuan Liu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhewen Hao, Zhihong Shao, Zhiniu Wen, Zhipeng Xu, Zhongyu Zhang, Zhuoshu Li, Zihan Wang, Zihui Gu, Zilin Li, and Ziwei Xie. 2024. DeepSeek-V2: A Strong, Economical, and Efficient Mixture-of-Experts Language Model. arXiv:2405.04434 [cs.CL] https://arxiv.org/abs/2405.04434

[20] DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jiawei Wang, Jin Chen, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Litong Wang, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runxin Xu, Ruoyu Zhang, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Shuting Pan, T. Wang, Tao Yun, Tian Pei, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wanjia Zhao, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaokang Zhang, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xinnan Song, Xinxia Shan, Xinyi Zhou, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. X. Zhu, Yang Zhang, Yanhong Xu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Yu, Yi Zheng, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Ying Tang, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yu Wu, Yuan Ou, Yuchen Zhu, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yukun Zha, Yunfan Xiong, Yunxian Ma, Yuting Yan, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhipeng Xu, Zhiyu Wu, Zhongyu Zhang, Zhuoshu Li, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Ziyi Gao, and Zizheng Pan. 2025. DeepSeek-V3 Technical Report. arXiv:2412.19437 [cs.CL] https://arxiv.org/abs/2412.19437

[21] Depend. 2024. HBM Market Insight. https://depend-ele.com/hbm-market-insight-2/

[22] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale. arXiv:2208.07339 [cs.LG] https://arxiv.org/abs/2208.07339

[23] Jonathon Evans. 2022. Nvidia Grace. In *2022 IEEE Hot Chips 34 Symposium (HCS)*. 1–20. doi:10.1109/HCS55958.2022.9895599

[24] Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. 2024. Model Tells You What to Discard: Adaptive KV Cache Compression for LLMs. arXiv:2310.01801 [cs.CL] https://arxiv.org/abs/2310.01801

[25] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew

Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. 2024. The Llama 3 Herd of Models. arXiv:2407.21783 [cs.AI] https://arxiv.org/abs/2407.21783

[26] Groq. 2024. GroqCard Accelerator. https://groq.com/groqcard-accelerator/. Accessed: 2025-03-27.

[27] Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W Mahoney, Sophia Shao, Kurt Keutzer, and Amir Gholami. 2024. KVQuant: Towards 10 Million Context Length LLM Inference with KV Cache Quantization. In *Advances in Neural Information Processing Systems (NeurIPS)*. https://nips.cc/virtual/2024/poster/96936

[28] Yu-Chen Hu, Yu-Min Liang, Hsieh-Pin Hu, Chia-Yen Tan, Chih-Ta Shen, Chien-Hsun Lee, and S. Y. Hou. 2023. CoWoS Architecture Evolution for Next Generation HPC on 2.5D System in Package. In *2023 IEEE 73rd Electronic Components and Technology Conference (ECTC)*. 1022–1026. doi:10.1109/ECTC51909.2023.00174

[29] Youhe Jiang, Fangcheng Fu, Xiaozhe Yao, Taiyi Wang, Bin Cui, Ana Klimovic, and Eiko Yoneki. 2025. ThunderServe: High-performance and Cost-efficient LLM Serving in Cloud Environments. arXiv:2502.09334 [cs.DC] https://arxiv.org/abs/2502.09334

[30] Norm Jouppi, George Kurian, Sheng Li, Peter Ma, Rahul Nagarajan, Lifeng Nai, Nishant Patil, Suvinay Subramanian, Andy Swing, Brian Towles, Clifford Young, Xiang Zhou, Zongwei Zhou, and David A Patterson. 2023. TPU v4: An Optically Reconfigurable Supercomputer for Machine Learning with Hardware Support for Embeddings. In *Proceedings of the 50th Annual International Symposium on Computer Architecture* (Orlando, FL, USA) *(ISCA '23)*. Association for Computing Machinery, New York, NY, USA, Article 82, 14 pages. doi:10.1145/3579371.3589350

[31] Norman P. Jouppi, Doe Hyun Yoon, Matthew Ashcraft, Mark Gottscho, Thomas B. Jablin, George Kurian, James Laudon, Sheng Li, Peter Ma, Xiaoyu Ma, Thomas Norrie, Nishant Patil, Sushma Prasad, Cliff Young, Zongwei Zhou, and David Patterson. 2021. Ten Lessons From Three Generations Shaped Google's TPUv4i : Industrial Product. In *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*. 1–14. doi:10.1109/ISCA52012.2021.00010

[32] Norman P. Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, Rick Boyle, Pierre-luc Cantin, Clifford Chao, Chris Clark, Jeremy Coriell, Mike Daley, Matt Dau, Jeffrey Dean, Ben Gelb, Tara Vazir Ghaemmaghami, Rajendra Gottipati, William Gulland, Robert Hagmann, C. Richard Ho, Doug Hogberg, John Hu, Robert Hundt, Dan Hurt, Julian Ibarz, Aaron Jaffey, Alek Jaworski, Alexander Kaplan, Harshit Khaitan, Daniel Killebrew, Andy Koch, Naveen Kumar, Steve Lacy, James Laudon, James Law, Diemthu Le, Chris Leary, Zhuyuan Liu, Kyle Lucke, Alan Lundin, Gordon MacKean, Adriana Maggiore, Maire Mahony, Kieran Miller, Rahul Nagarajan, Ravi Narayanaswami, Ray Ni, Kathy Nix, Thomas Norrie, Mark Omernick, Narayana Penukonda, Andy Phelps, Jonathan Ross, Matt Ross, Amir Salek, Emad Samadiani, Chris Severn, Gregory Sizikov, Matthew Snelham, Jed Souter, Dan Steinberg, Andy Swing, Mercedes Tan, Gregory Thorson, Bo Tian, Horia Toma, Erick Tuttle, Vijay Vasudevan, Richard Walter, Walter Wang, Eric Wilcox, and Doe Hyun Yoon. 2017. In-Datacenter Performance Analysis of a Tensor Processing Unit. In *Proceedings of the 44th Annual International Symposium on Computer Architecture* (Toronto, ON, Canada) *(ISCA '17)*. Association for Computing Machinery, New York, NY, USA, 1–12. doi:10.1145/3079856.3080246

[33] Aditya K. Kamath, Ramya Prabhu, Jayashree Mohan, Simon Peter, Ramachandran Ramjee, and Ashish Panwar. 2025. POD-Attention:

Unlocking Full Prefill-Decode Overlap for Faster LLM Inference. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2* (Rotterdam, Netherlands) *(ASPLOS '25)*. Association for Computing Machinery, New York, NY, USA, 897–912. doi:10.1145/3676641.3715996

[34] 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 Symposium on Operating Systems Principles* (Koblenz, Germany) *(SOSP '23)*. Association for Computing Machinery, New York, NY, USA, 611–626. doi:10.1145/3600006.3613165

[35] Ya Liu, Annabelle Pratt, Pavan Kumar, Ming Xu, and Fred C. Lee. 2007. 390V Input VRM for High Efficiency Server Power Architecture. In *APEC 07 - Twenty-Second Annual IEEE Applied Power Electronics Conference and Exposition*. 1619–1624. doi:10.1109/APEX.2007.357734

[36] Locuza. 2022. Nvidia's AD102 officially revealed, how close were the previous estimates? https://locuza.substack.com/p/nvidias-ad102-officially-revealed

[37] Micron. 2024. Micron GDDR7 Memory Product Brief. https://www.micron.com/content/dam/micron/global/public/products/product-flyer/gddr7-product-brief.pdf Accessed: 2025-04-07.

[38] Stephen Nellis and Max A. Cherney. 2025. Nvidia CEO says orders for 3.6 million Blackwell GPUs exclude Meta. https://finance.yahoo.com/news/nvidia-ceo-says-orders-3-171501205.html Accessed: 2025-03-24.

[39] August Ning, Georgios Tziantzioulis, and David Wentzlaff. 2023. Supply Chain Aware Computer Architecture. In *Proceedings of the 50th Annual International Symposium on Computer Architecture* (Orlando, FL, USA) *(ISCA '23)*. Association for Computing Machinery, New York, NY, USA, Article 17, 15 pages. doi:10.1145/3579371.3589052

[40] Thomas Norrie, Nishant Patil, Doe Hyun Yoon, George Kurian, Sheng Li, James Laudon, Cliff Young, Norman Jouppi, and David Patterson. 2021. The Design Process for Google's Training Chips: TPUv2 and TPUv3. *IEEE Micro* 41, 2 (2021), 56–63. doi:10.1109/MM.2021.3058217

[41] NVIDIA. 2023. NVIDIA RTX 6000 Ada Generation Datasheet. https://www.nvidia.com/content/dam/en-zz/Solutions/design-visualization/rtx-6000/proviz-print-rtx6000-datasheet-web-2504660.pdf. Accessed: 2025-04-03.

[42] NVIDIA. 2024. NVIDIA A100 Tensor Core GPU. https://www.nvidia.com/en-us/data-center/a100/. Accessed: 2025-03-27.

[43] NVIDIA. 2024. NVIDIA Blackwell Datasheet. https://resources.nvidia.com/en-us-blackwell-architecture/datasheet. Accessed: 2025-03-27.

[44] NVIDIA. 2024. NVIDIA H100 Tensor Core GPU Architecture Overview. https://resources.nvidia.com/en-us-hopper-architecture/nvidia-h100-tensor-c. Accessed: 2025-03-27.

[45] NVIDIA. 2025. *Dynamo: A Datacenter Scale Distributed Inference Serving Framework*. https://github.com/ai-dynamo/dynamo Accessed: 2025-03-26.

[46] NVIDIA. 2025. NVIDIA GeForce RTX 5090 Graphics Card. https://www.nvidia.com/en-us/geforce/graphics-cards/50-series/rtx-5090/ Accessed: 2025-04-03.

[47] OpenAI. 2024. GPT-4 Technical Report. arXiv:2303.08774 [cs.CL] https://arxiv.org/abs/2303.08774

[48] Sang-Soo Park, KyungSoo Kim, Jinin So, Jin Jung, Jonggeon Lee, Kyoungwan Woo, Nayeon Kim, Younghyun Lee, Hyungyo Kim, Yongsuk Kwon, Jinhyun Kim, Jieun Lee, YeonGon Cho, Yongmin Tai, Jeonghyeon Cho, Hoyoung Song, Jung Ho Ahn, and Nam Sung Kim. 2024. An LPDDR-based CXL-PNM Platform for TCO-efficient Inference of Transformer-based Large Language Models. In *2024 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. 970–982. doi:10.1109/HPCA57654.2024.00078

[49] Dylan Patel and Afzal Ahmad. 2023. The Inference Cost Of Search Disruption – Large Language Model Cost Analysis. https://semianalysis.com/2023/02/09/the-inference-cost-of-search-disruption/. Accessed: 2025-03-24.

[50] 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 *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3* (La Jolla, CA, USA) *(ASPLOS '24)*. Association for Computing Machinery, New York, NY, USA, 207–222. doi:10.1145/3620666.3651329

[51] 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 *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*. 118–132. doi:10.1109/ISCA59077.2024.00019

[52] Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. 2023. Efficiently Scaling Transformer Inference. In *Proceedings of Machine Learning and Systems*, D. Song, M. Carbin, and T. Chen (Eds.), Vol. 5. Curan, 606–624. https://proceedings.mlsys.org/paper_files/paper/2023/file/c4be71ab8d24cdfb45e3d06dbfca2780-Paper-mlsys2023.pdf

[53] Ruoyu Qin, Zheming Li, Weiran He, Jialei Cui, Feng Ren, Mingxing Zhang, Yongwei Wu, Weimin Zheng, and Xinran Xu. 2025. Mooncake: Trading More Storage for Less Computation — A KVCache-centric Architecture for Serving LLM Chatbot. In *23rd USENIX Conference on File and Storage Technologies (FAST 25)*. USENIX Association, Santa Clara, CA, 155–170. https://www.usenix.org/conference/fast25/presentation/qin

[54] Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, Jonathan Tow, Alexander M. Rush, Stella Biderman, Albert Webson, Pawan Sasanka Ammanamanchi, Thomas Wang, Benoît Sagot, Niklas Muennighoff, Albert Villanova del Moral, Olatunji Ruwase, Rachel Bawden, Stas Bekman, Angelina McMillan-Major, Iz Beltagy, Huu Nguyen, Lucile Saulnier, Samson Tan, Pedro Ortiz Suarez, Victor Sanh, Hugo Laurençon, Yacine Jernite, Julien Launay, Margaret Mitchell, Colin Raffel, Aaron Gokaslan, Adi Simhi, Aitor Soroa, Alham Fikri Aji, Amit Alfassy, Anna Rogers, Ariel Kreisberg Nitzav, Canwen Xu, Chenghao Mou, Chris Emezue, Christopher Klamm, Colin Leong, Daniel van Strien, David Ifeoluwa Adelani, Dragomir Radev, Eduardo González Ponferrada, Efrat Levkovizh, Ethan Kim, Eyal Bar Natan, Francesco De Toni, Gérard Dupont, Germán Kruszewski, Giada Pistilli, Hady Elsahar, Hamza Benyamina, Hieu Tran, Ian Yu, Idris Abdulmumin, Isaac Johnson, Itziar Gonzalez-Dios, Javier de la Rosa, Jenny Chim, Jesse Dodge, Jian Zhu, Jonathan Chang, Jörg Frohberg, Joseph Tobing, Joydeep Bhattacharjee, Khalid Almubarak, Kimbo Chen, Kyle Lo, Leandro Von Werra, Leon Weber, Long Phan, Loubna Ben allal, Ludovic Tanguy, Manan Dey, Manuel Romero Muñoz, Maraim Masoud, María Grandury, Mario Šaško, Max Huang, Maximin Coavoux, Mayank Singh, Mike Tian-Jian Jiang, Minh Chien Vu, Mohammad A. Jauhar, Mustafa Ghaleb, Nishant Subramani, Nora Kassner, Nurulaqilla Khamis, Olivier Nguyen, Omar Espejel, Ona de Gibert, Paulo Villegas, Peter Henderson, Pierre Colombo, Priscilla Amuok, Quentin Lhoest, Rheza Harliman, Rishi Bommasani, Roberto Luis López, Rui Ribeiro, Salomey Osei, Sampo Pyysalo, Sebastian Nagel, Shamik Bose, Shamsuddeen Hassan Muhammad, Shanya Sharma, Shayne Longpre, Somaieh Nikpoor, Stanislav Silberberg, Suhas Pai, Sydney Zink, Tiago Timponi Torrent, Timo Schick, Tristan Thrush, Valentin Danchev, Vassilina Nikoulina, Veronika Laippala, Violette Lepercq, Vrinda Prabhu, Zaid Alyafeai, Zeerak Talat,

Arun Raja, Benjamin Heinzerling, Chenglei Si, Davut Emre Taşar, Elizabeth Salesky, Sabrina J. Mielke, Wilson Y. Lee, Abheesht Sharma, Andrea Santilli, Antoine Chaffin, Arnaud Stiegler, Debajyoti Datta, Eliza Szczechla, Gunjan Chhablani, Han Wang, Harshit Pandey, Hendrik Strobelt, Jason Alan Fries, Jos Rozen, Leo Gao, Lintang Sutawika, M Saiful Bari, Maged S. Al-shaibani, Matteo Manica, Nihal Nayak, Ryan Teehan, Samuel Albanie, Sheng Shen, Srulik Ben-David, Stephen H. Bach, Taewoon Kim, Tali Bers, Thibault Fevry, Trishala Neeraj, Urmish Thakker, Vikas Raunak, Xiangru Tang, Zheng-Xin Yong, Zhiqing Sun, Shaked Brody, Yallow Uri, Hadar Tojarieh, Adam Roberts, Hyung Won Chung, Jaesung Tae, Jason Phang, Ofir Press, Conglong Li, Deepak Narayanan, Hatim Bourfoune, Jared Casper, Jeff Rasley, Max Ryabinin, Mayank Mishra, Minjia Zhang, Mohammad Shoeybi, Myriam Peyrounette, Nicolas Patry, Nouamane Tazi, Omar Sanseviero, Patrick von Platen, Pierre Cornette, Pierre François Lavallée, Rémi Lacroix, Samyam Rajbhandari, Sanchit Gandhi, Shaden Smith, Stéphane Requena, Suraj Patil, Tim Dettmers, Ahmed Baruwa, Amanpreet Singh, Anastasia Cheveleva, Anne-Laure Ligozat, Arjun Subramonian, Aurélie Névéol, Charles Lovering, Dan Garrette, Deepak Tunuguntla, Ehud Reiter, Ekaterina Taktasheva, Ekaterina Voloshina, Eli Bogdanov, Genta Indra Winata, Hailey Schoelkopf, Jan-Christoph Kalo, Jekaterina Novikova, Jessica Zosa Forde, Jordan Clive, Jungo Kasai, Ken Kawamura, Liam Hazan, Marine Carpuat, Miruna Clinciu, Najoung Kim, Newton Cheng, Oleg Serikov, Omer Antverg, Oskar van der Wal, Rui Zhang, Ruochen Zhang, Sebastian Gehrmann, Shachar Mirkin, Shani Pais, Tatiana Shavrina, Thomas Scialom, Tian Yun, Tomasz Limisiewicz, Verena Rieser, Vitaly Protasov, Vladislav Mikhailov, Yada Pruksachatkun, Yonatan Belinkov, Zachary Bamberger, Zdeněk Kasner, Alice Rueda, Amanda Pestana, Amir Feizpour, Ammar Khan, Amy Faranak, Ana Santos, Anthony Hevia, Antigona Unldreaj, Arash Aghagol, Arezoo Abdollahi, Aycha Tammour, Azadeh HajiHosseini, Bahareh Behroozi, Benjamin Ajibade, Bharat Saxena, Carlos Muñoz Ferrandis, Daniel McDuff, Danish Contractor, David Lansky, Davis David, Douwe Kiela, Duong A. Nguyen, Edward Tan, Emi Baylor, Ezinwanne Ozoani, Fatima Mirza, Frankline Ononiwu, Habib Rezanejad, Hessie Jones, Indrani Bhattacharya, Irene Solaiman, Irina Sedenko, Isar Nejadgholi, Jesse Passmore, Josh Seltzer, Julio Bonis Sanz, Livia Dutra, Mairon Samagaio, Maraim Elbadri, Margot Mieskes, Marissa Gerchick, Martha Akinlolu, Michael McKenna, Mike Qiu, Muhammed Ghauri, Mykola Burynok, Nafis Abrar, Nazneen Rajani, Nour Elkott, Nour Fahmy, Olanrewaju Samuel, Ran An, Rasmus Kromann, Ryan Hao, Samira Alizadeh, Sarmad Shubber, Silas Wang, Sourav Roy, Sylvain Viguier, Thanh Le, Tobi Oyebade, Trieu Le, Yoyo Yang, Zach Nguyen, Abhinav Ramesh Kashyap, Alfredo Palasciano, Alison Callahan, Anima Shukla, Antonio Miranda-Escalada, Ayush Singh, Benjamin Beilharz, Bo Wang, Caio Brito, Chenxi Zhou, Chirag Jain, Chuxin Xu, Clémentine Fourrier, Daniel León Periñán, Daniel Molano, Dian Yu, Enrique Manjavacas, Fabio Barth, Florian Fuhrimann, Gabriel Altay, Giyaseddin Bayrak, Gully Burns, Helena U. Vrabec, Imane Bello, Ishani Dash, Jihyun Kang, John Giorgi, Jonas Golde, Jose David Posada, Karthik Rangasai Sivaraman, Lokesh Bulchandani, Lu Liu, Luisa Shinzato, Madeleine Hahn de Bykhovetz, Maiko Takeuchi, Marc Pàmies, Maria A Castillo, Marianna Nezhurina, Mario Sänger, Matthias Samwald, Michael Cullan, Michael Weinberg, Michiel De Wolf, Mina Mihaljcic, Minna Liu, Moritz Freidank, Myungsun Kang, Natasha Seelam, Nathan Dahlberg, Nicholas Michio Broad, Nikolaus Muellner, Pascale Fung, Patrick Haller, Ramya Chandrasekhar, Renata Eisenberg, Robert Martin, Rodrigo Canalli, Rosaline Su, Ruisi Su, Samuel Cahyawijaya, Samuele Garda, Shlok S Deshmukh, Shubhanshu Mishra, Sid Kiblawi, Simon Ott, Sinee Sang-aroonsiri, Srishti Kumar, Stefan Schweter, Sushil Bharati, Tanmay Laud, Théo Gigant, Tomoya Kainuma, Wojciech Kusa, Yanis Labrak, Yash Shailesh Bajaj, Yash Venkatraman, Yifan Xu, Yingxin Xu, Yu Xu, Zhe Tan, Zhongli Xie, Zifan Ye, Mathilde Bras, Younes Belkada, and Thomas Wolf. 2023. BLOOM: A 176B-Parameter Open-Access Multilingual Language Model. arXiv:2211.05100 [cs.CL] https://arxiv.org/abs/2211.05100

[55] Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Beidi Chen, Percy Liang, Christopher Ré, Ion Stoica, and Ce Zhang. 2023. FlexGen: high-throughput generative inference of large language models with a single GPU. In *Proceedings of the 40th International Conference on Machine Learning* (Honolulu, Hawaii, USA) *(ICML'23)*. JMLR.org, Article 1288, 23 pages. https://dl.acm.org/doi/abs/10.5555/3618408.3619696

[56] Omar Sohail. 2024. Snapdragon X Elite Die Shot Shows A Core Area Of 169.6mm², With The Entire CPU Cluster 78 Percent Larger Than Apple's M4. https://wccftech.com/snapdragon-x-elite-die-shot-compared-with-apple-m4/ Accessed: 2025-04-08.

[57] Keeyoung Son, Joonsang Park, Seongguk Kim, Boogyo Sim, Keunwoo Kim, Seonguk Choi, Hyunsik Kim, and Joungho Kim. 2023. Thermal Analysis of High Bandwidth Memory (HBM)-GPU Module considering Power Consumption. In *2023 IEEE Electrical Design of Advanced Packaging and Systems (EDAPS)*. 1–3. doi:10.1109/EDAPS58880.2023.10468315

[58] Vikranth Srivatsa, Zijian He, Reyna Abhyankar, Dongming Li, and Yiying Zhang. 2024. Preble: Efficient Distributed Prompt Scheduling for LLM Serving. arXiv:2407.00023 [cs.DC] https://arxiv.org/abs/2407.00023

[59] Sean J. Taylor and Benjamin Letham. 2018. Forecasting at Scale. *The American Statistician* 72, 1 (2018), 37–45. doi:10.1080/00031305.2017.1380080

[60] TrendForce. [n. d.]. DRAMeXchange. https://www.dramexchange.com/ Date Accessed: 07 April 2025.

[61] TrendForce. 2024. TSMC's 2nm Wafers Reportedly Set to Double in Price, Benefitting IP/ Material Companies. https://www.trendforce.com/news/2024/10/04/news-tsmcs-2nm-wafers-reportedly-set-to-double-in-price-benefitting-ip-material-companies/

[62] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) *(NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 6000–6010. https://dl.acm.org/doi/10.5555/3295222.3295349

[63] VideoCardz. 2025. NVIDIA GB202 Blackwell 760mm² GPU Die Shot Revealed: 24756 Cores and 512-bit Bus. https://videocardz.com/newz/nvidia-gb202-blackwell-760mm%C2%B2-gpu-die-shot-revealed-24756-cores-and-512-bit-bus Accessed: 2025-04-08.

[64] Bingyang Wu, Shengyu Liu, Yinmin Zhong, Peng Sun, Xuanzhe Liu, and Xin Jin. 2024. LoongServe: Efficiently Serving Long-Context Large Language Models with Elastic Sequence Parallelism. In *Proceedings of the ACM SIGOPS 30th Symposium on Operating Systems Principles* (Austin, TX, USA) *(SOSP '24)*. Association for Computing Machinery, New York, NY, USA, 640–654. doi:10.1145/3694715.3695948

[65] Mengdi Wu, Xinhao Cheng, Shengyu Liu, Chunan Shi, Jianan Ji, Kit Ao, Praveen Velliengiri, Xupeng Miao, Oded Padon, and Zhihao Jia. 2025. Mirage: A Multi-Level Superoptimizer for Tensor Programs. arXiv:2405.05751 [cs.LG] https://arxiv.org/abs/2405.05751

[66] xAI. 2025. *Grok 3 Beta — The Age of Reasoning Agents*. https://x.ai/blog/grok-3 Accessed: 2025-03-24.

[67] Jerry Yang and Levi Li. 2025. TSMC's price hikes send Apple A-series wafer costs soaring to US$18,000 per wafer. https://www.digitimes.com/news/a20250107PD217/apple-tsmc-3nm-chips-wafer.html

[68] Zihao Ye, Lequn Chen, Ruihang Lai, Wuwei Lin, Yineng Zhang, Stephanie Wang, Tianqi Chen, Baris Kasikci, Vinod Grover, Arvind Krishnamurthy, and Luis Ceze. 2025. FlashInfer: Efficient and Customizable Attention Engine for LLM Inference Serving. arXiv:2501.01005 [cs.DC] https://arxiv.org/abs/2501.01005

[69] Gyeong-In Yu, Joo Seong Jeong, Geon-Woo Kim, Soojeong Kim, and Byung-Gon Chun. 2022. Orca: A Distributed Serving System for Transformer-Based Generative Models. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*. USENIX Association, Carlsbad, CA, 521–538. https://www.usenix.org/conference/osdi22/presentation/yu

[70] Chen Zhang, Kuntai Du, Shu Liu, Woosuk Kwon, Xiangxi Mo, Yufeng Wang, Xiaoxuan Liu, Kaichao You, Zhuohan Li, Mingsheng Long, Jidong Zhai, Joseph Gonzalez, and Ion Stoica. 2025. Jenga: Effective Memory Management for Serving LLM with Heterogeneity. arXiv:2503.18292 [cs.DC] https://arxiv.org/abs/2503.18292

[71] Hengrui Zhang, August Ning, Rohan Baskar Prabhakar, and David Wentzlaff. 2024. LLMCompass: Enabling Efficient Hardware Design for Large Language Model Inference. In *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*. 1080–1096. doi:10.1109/ISCA59077.2024.00082

[72] Yilong Zhao, Chien-Yu Lin, Kan Zhu, Zihao Ye, Lequn Chen, Size Zheng, Luis Ceze, Arvind Krishnamurthy, Tianqi Chen, and Baris Kasikci. 2024. Atom: Low-bit Quantization for Efficient and Accurate LLM Serving. https://mlsys.org/virtual/2024/poster/2655

[73] Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. 2025. SGLang: efficient execution of structured language model programs. In *Proceedings of the 38th International Conference on Neural Information Processing Systems* (Vancouver, BC, Canada) *(NIPS '24)*. Curran Associates Inc., Red Hook, NY, USA, Article 2000, 27 pages. https://dl.acm.org/doi/10.5555/3737916.3739916

[74] Yinmin Zhong, Shengyu Liu, Junda Chen, Jianbo Hu, Yibo Zhu, Xuanzhe Liu, Xin Jin, and Hao Zhang. 2024. DistServe: disaggregating prefill and decoding for goodput-optimized large language model serving. In *Proceedings of the 18th USENIX Conference on Operating Systems Design and Implementation* (Santa Clara, CA, USA) *(OSDI'24)*. USENIX Association, USA, Article 11, 18 pages. https://dl.acm.org/doi/10.5555/3691938.3691949

[75] Kan Zhu, Yufei Gao, Yilong Zhao, Liangyu Zhao, Gefei Zuo, Yile Gu, Dedong Xie, Tian Tang, Qinyu Xu, Zihao Ye, Keisuke Kamahori, Chien-Yu Lin, Ziren Wang, Stephanie Wang, Arvind Krishnamurthy, and Baris Kasikci. 2025. NanoFlow: Towards Optimal Large Language Model Serving Throughput. arXiv:2408.12757 [cs.DC] https://arxiv.org/abs/2408.12757