Check for updates

ACM DIGITAL LIBRARY | Association for Computing Machinery | acm open

Latest updates: https://dl.acm.org/doi/10.1145/3718958.3750521

RESEARCH-ARTICLE

# Astral: A Datacenter Infrastructure for Large Language Model Training at Scale

**QINGKAI MENG**, Nanjing University, Nanjing, Jiangsu, China

**HAO ZHENG**, Nanjing University, Nanjing, Jiangsu, China

**ZHENHUI ZHANG**, Nanjing University, Nanjing, Jiangsu, China

**CHONLAM LAO**, Harvard University, Cambridge, MA, United States

**CHENGYUAN HUANG**, Nanjing University, Nanjing, Jiangsu, China

**BAOJIA LI**, Tencent, Shenzhen, Guangdong, China

**View all**

**Open Access Support** provided by:

**Tencent**

**Nanjing University**

**Harvard University**

**Politecnico di Milano**

**PDF Download**
**3718958.3750521.pdf**
**21 January 2026**
**Total Citations:** 1
**Total Downloads:** 3041

**Published:** 08 September 2025

**Citation in BibTeX format**

SIGCOMM '25: ACM SIGCOMM 2025 Conference
*September 8 - 11, 2025*
*Coimbra, Portugal*

**Conference Sponsors:**
SIGCOMM

# Astral: A Datacenter Infrastructure for Large Language Model Training at Scale

Qingkai Meng[★], Hao Zheng[★], Zhenhui Zhang[★], ChonLam Lao[♣], Chengyuan Huang[★], Baojia Li[♡], Ziyuan Zhu[♡], Hao Lu[♡], Weizhen Dang[♡], Zitong Lin[♡], Weifeng Zhang[♡], Lingfeng Liu[♡], Yuanyuan Gong[♡], Chunzhi He[♡], Xiaoyuan Hu[♡], Yinben Xia[♡], Xiang Li[♡], Zekun He[♡], Yachen Wang[♡], Xianneng Zou[♡], Kun Yang[★], Gianni Antichi[♠], Guihai Chen[★], Chen Tian[★]

[★] *Nanjing University* [♣] *Harvard University* [♡] *Tencent*

[♠] *Politecnico di Milano and Queen Mary University of London*

## Abstract

The flourishing of Large Language Models (LLMs) calls for increasingly ultra-scale training. In this paper, we share our experience in designing, deploying, and operating our novel Astral datacenter infrastructure, along with operational lessons and evolutionary insights gained from its production use. Astral has three important innovations: *(i)* a same-rail interconnection network architecture on tier-2, which enables the scaling of LLM training. To physically deploy this high-density infrastructure, we introduce a distributed high-voltage direct current power system and a new air-liquid integrated cooling system. *(ii)* a full-stack monitoring system featuring cross-host and hierarchical logging correlation, which diagnoses failures at scale and precisely localizes root causes. *(iii)* an operator-granular forecasting component Seer that efficiently generates operator execution timelines with acceptable accuracy, aiding in fault diagnosis, model tuning, and network architecture upgrading. Astral infrastructure has been gradually deployed over 18 months, supporting LLM training and inference for multiple customers.

## CCS Concepts

• **Networks** → **Network architectures**; **Network monitoring**; **Network performance evaluation**.

## Keywords

Network Infrastructure; Large Language Model; Network Architecture; Network Monitoring; Network Simulations

**ACM Reference Format:**

Qingkai Meng, Hao Zheng, Zhenhui Zhang, ChonLam Lao, Chengyuan Huang, Baojia Li, Ziyuan Zhu, Hao Lu, Weizhen Dang, Zitong Lin, Weifeng Zhang, Lingfeng Liu, Yuanyuan Gong, Chunzhi He, Xiaoyuan Hu, Yinben Xia, Xiang Li, Zekun He, Yachen Wang, Xianneng Zou, Kun Yang, Gianni Antichi, Guihai Chen, Chen Tian. 2025. Astral: A Datacenter Infrastructure for Large Language Model Training at Scale. In *ACM SIGCOMM 2025 Conference (SIGCOMM '25), September 8–11, 2025, Coimbra, Portugal*. ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3718958.3750521

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

*SIGCOMM '25, September 8–11, 2025, Coimbra, Portugal*

© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 979-8-4007-1524-2/2025/09
https://doi.org/10.1145/3718958.3750521

## 1 Introduction

In recent years, Large Language Models (LLMs) have gained popularity for a wide range of tasks, including text summarization [13, 37, 44], multimedia content creation [10, 25, 28], code optimization [3], and personalized recommendation [34, 36]. This success has led LLMs to flourish, and in particular, we have witnessed two specific trends indicative of the exponential scaling of LLMs: *(i) model size has grown exponentially from billions to trillions, e.g.*, 1.5B in GPT-2, 175B in GPT-3, and 1T in GPT-4 [13, 37]; *(ii) training datasets have dramatically increased from billions of tokens to trillions of tokens, e.g.*, ~40 billion tokens in GPT-2 to more than 10 trillion tokens in GPT-4 and LLaMA 3 [22, 37, 40].

In this paper, we share our experience in building and running Astral, our new datacenter infrastructure natively built for LLM training and inference that can scale to half a million GPUs. Many companies have already shared similar experiences regarding tens of thousands of GPUs [20, 27, 39], but we feel this is the right time to provide ours as well. The reason is motivated by three important factors: *(i) recently, the growth in GPU's Floating Point Operations Per Second (FLOPS) has failed to keep pace with the FLOPS required for training models, e.g.*, a 4× increase from V100 in 2017 to H100 in 2022, while a 1500× increase from BERT-base [17] to GPT-3-175B [13]. *(ii) GPUs that are supplied to China are specially characterized by a much lower FLOPS compared to their more advanced products. (iii) our customers and internal teams take ultra-scale GPUs to develop models:* our company is dedicated to serving both large content providers requiring more than 10K GPUs for deploying as well as upgrading their models, and our internal teams exploring the extremes of LLM's capabilities.

Training state-of-the-art models with low-tier GPUs is possible, but it might take too much time, even if we are able to match the network and interconnection settings of modern deployments. This was not an option for us as we did not want this to affect the quality of our services alongside our economic benefits. The only way for us to be competitive was to build a datacenter network that scales to at least half a million low-tier GPUs, so that we could serve our 1.4 billion active users with our in-production LLM that exceeds one trillion parameters and has been trained on trillions of tokens. Building and running a datacenter at such a scale brings up several challenges that span multiple dimensions: from the need to deal with cooling, excessive wiring, and enormous power consumption to running Remote Direct Memory Access (RDMA) at scale alongside providing mechanisms for model tuning and operations so that we could provide the best possible performance and reliability.

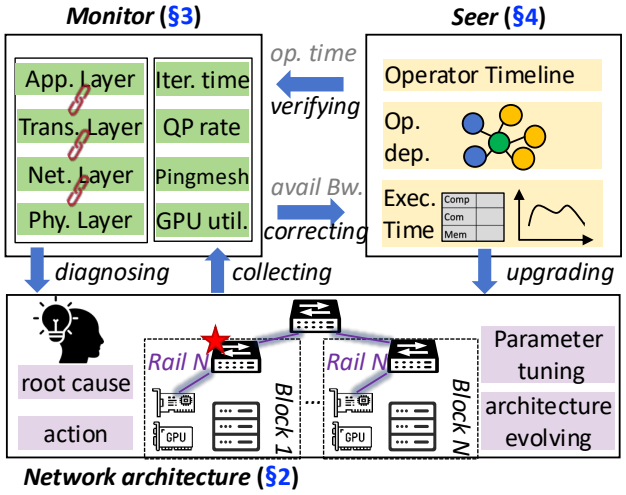



**Figure 1: Astral infrastructure for LLM.**

Our Astral infrastructure is overviewed in Figure 1. The Astral network architecture accommodating large-scale GPUs serves as the foundation for LLM workloads. Built atop this, a full-stack monitoring system is developed to enable comprehensive data collection and fault diagnosis; meanwhile, an operator-granular forecasting component Seer is integrated, offering valuable insights into fault diagnosis, model parameter tuning, and architecture upgrades by calibrating its predictions using monitoring data. The key contributions are as follows:

• Astral network architecture adopts a novel Aggregation switch and ToR switch interconnection design, *i.e.*, aggregating same-rail ToR switches to maximize Pod scale while achieving identical aggregated bandwidth across three tiers for uniform network abstraction, which improves scalability and enables flexible deployment while ensuring high training efficiency. To physically deploy our high-density infrastructure, we have built a distributed High-Voltage Direct Current (HVDC) power system for high reliability and introduced a new air-liquid integrated cooling system for high energy efficiency (§2).

• Our network is supported by a novel full-stack Astral monitoring system distinguished by its ability to correlate each layer from the application layer to the physical layer in line with the top-down system principle so that it is easier to spot RDMA inefficiencies or consistency issues. Based on the monitoring system, we build a cross-host and hierarchical correlation analyzer, which effectively handles comprehensive failure manifestations at a large scale, *e.g.*, fail-hang and fail-slow, and precisely localizes root causes (§3).

• We introduce a new forecasting component, Astral Seer, that effectively generates operator-granular timeline forecasts. Moreover, Seer leverages realistic monitoring data to calibrate operator execution time atop basic LLM modelling for high accuracy. Seer aids in diagnosing in-production failures with an expected timeline for reliability and providing model parameter setting recommendations, model framework evolution, and network architecture upgrades for high training efficiency (§4).

Astral infrastructure has been gradually deployed and used in our production for more than 18 months. Our experience shows that, by physically deploying Astral infrastructure, LLM training efficiency is scaled near-linearly (*e.g.*, 0.6% efficiency loss in our production statistics), and the average Power Usage Effectiveness (PUE) is improved by 16.34%. Additionally, with the gradual deployment of Astral monitoring system, the mean time to locate a failure has been reduced from days to minutes, *i.e.*, by up to 25× reductions. Moreover, Astral Seer forecasts the performance of LLM training and inference within seconds while achieving acceptable accuracy (*e.g.*, 0.3% deviation in Hunyuan models and across other dense models) when compared with production results.

This work does not raise any ethical issues.

## 2 Astral Network

We build a new network architecture for LLM training, with the following key attributes:

**Scalability:** To expedite model development cycles and harness the potential of LLMs, scaling LLM training efficiency near-linearly to millions of GPUs is crucial. To achieve this, the network architecture should *(i)* be able to connect GPUs as scalably as possible given today's hardware limitations, and *(ii)* enable high training efficiency at scale by minimizing impacts of GPU-to-GPU communication, including identical bandwidth at each tier and minimal network hops from an architecture perspective. Identical bandwidth reduces the likelihood of bottleneck bandwidth in communications. While reducing the number of hops decreases the frequency of Equal-Cost Multi-Path (ECMP) hashing and then minimizes hash polarization [5, 50]. Both help avoid congestion from a network architecture perspective, thereby reducing the impact of communications.

**Flexibility:** As an LLM infrastructure service provider, we should achieve flexible deployment of LLM tasks to accommodate customers' fluctuating demand for GPUs. This can be achieved by allocating GPUs within the same block/Pod whenever possible to reduce the impact of communication overhead. Hence, it is essential to expand the size of the block (at tier 1) and Pod (at tier 2) and allocate GPU resources within one block/Pod as much as possible. However, due to customers' varying demand for GPU expansion and contraction, fragmented deployment across Pods often occurs in production. Thus, identical bandwidth at each tier also aids in minimizing the impact of cross-Pod communication.

**Reliability:** Failures are an unwelcome yet common occurrence, resulting from the complexities of network topology and hardware instability. Particularly, as LLM training scales, failures become increasingly disruptive, slowing down the entire job, possibly involving tens of thousands of GPUs. In production, one critical risk that can lead to failure is optical module damage, whose impact can be mitigated at the network architecture level.

**High energy efficiency:** Since data centers are major energy consumers, energy efficiency, *i.e.*, saving non-renewable energy and reducing carbon emissions, is a key consideration when physically deploying our network architecture in real-world environments.

### 2.1 Astral Network Architecture

To meet the above key attributes, we design and build Astral network architecture for LLM training, which follows three key principles:

**P1: Aggregation of same-rail ToR switches maximizes Pod size.** Figure 2 compares the performance of allocating 1K GPU

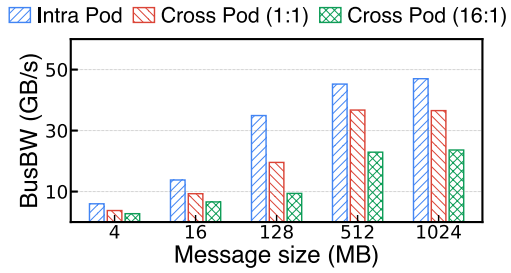



**Figure 2: All-to-all communication throughput.**

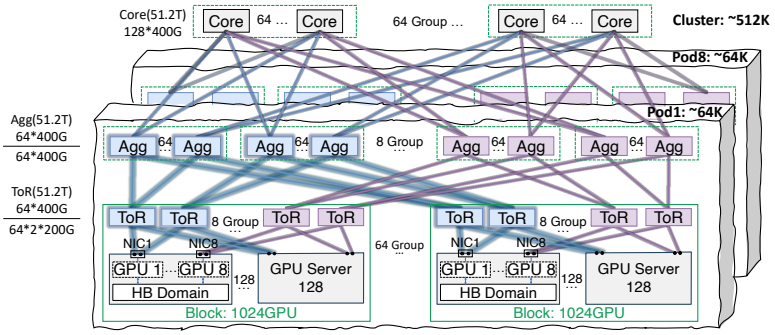



**Figure 3: Astral network architecture.**

for all-to-all collective communication within a single Pod versus across 32 Pods. As it indicates, fragmented deployment across multiple Pods decreases the all-to-all collective communication performance by 19%~37%. This is because even for all-to-all traffic, same-rail traffic accounts for a large proportion after enabling NVLink-optimized network communication [2, 46]. However, cross-Pod communication fails to fully utilize same-rail switches with fewer hops, causing more frequent ECMP hash polarization and severe congestion. In our observations, even with our optimized ECMP load balancing[1], the queue depth under cross-Pod communication exceeds the ECN threshold. As a result, maximizing the Pod size can avoid cross-Pod fragmented deployment and facilitate more same-rail communication, improving flexibility and scalability.

**P2: Identical aggregated bandwidth across all tiers enables uniform network abstraction.** Most LLM infrastructure providers, drawing on their cloud computing services experience, adopt bandwidth oversubscription in the Aggregation-Core layer (*i.e.*, tier 3) [20, 39]. However, oversubscription at any tier undermines network abstraction by exposing LLM training efficiency to the risk of communication bottlenecks. This impedes the linear scaling of GPU computational performance and limits flexible job deployment, which contradicts our design rationale. Figure 2 shows the impact of bandwidth oversubscription at tier 3 on communication throughput. Specifically, bandwidth oversubscription degrades all-to-all collective communication performance and model training performance by up to 52% and 3%, respectively. This performance gap arises because only ~15% of communication time remains after overlapping with computation. Intriguingly, the performance impact varies by LLM architecture. Dense transformer models mainly transfer AllReduce collective communication traffic from Data Parallelism (DP) and point-to-point traffic from Pipeline Parallelism (PP) between same-rail switches, thereby minimally traversing Core switches and thus tolerating a certain degree of bandwidth oversubscription at tier 3. In contrast, MoE-based transformer models rely on all-to-all collective communication due to Expert Parallelism (EP) and are more sensitive to bandwidth oversubscription. As MoE-based models gain popularity and uniform network abstractions emerge as a key requirement for tenants, identical aggregated bandwidth across all tiers becomes essential for our architecture.

**P3: Each port of a NIC is connected to a different ToR switch.** To achieve reliability, we employ network redundancy in the infrastructure. Specifically, our design aligns with recent experience that each port of a NIC is connected to a different ToR switch to avoid common optical modules and links damage, which is also adopted in the network architecture of IBM [21] and Alibaba [39].

Figure 3 overviews the Astral network architecture:

***(i) Host side:*** Given the maximal GPU limits within the intra-host network during deployment, each host is equipped with 8 GPUs. GPUs within the same host can communicate with each other via this intra-host network, *e.g.*, NVLink [6], with 400GBps-900GBps (bidirectional) bandwidth. To offer the maximum network capacity available today, we equip each host with 8 NICs, each with 2×200Gbps. Each of these 8 NICs serves a dedicated GPU (named rail), and thus each GPU has a dedicated 400Gbps of RDMA network, resulting in a total bandwidth of 3.2Tbps.

***(ii) Block side:*** Following dual-ToR designs [21, 39] for high reliability, two ToR switches are connected to two ports of the NIC bound by the same-rank GPUs respectively, forming same-rail links. Due to 8 NICs on one server, there are 16 switches with 51.2Tbps capacity in tier 1, which connect 1024 GPUs within a block.

***(iii) Pod side:*** To achieve the maximum number of GPUs on the same rail, two same-rail ToR switches in each block are connected to two groups of 64 Aggregation (Agg) switches in tier 2, respectively. Eventually, one Pod can support 64K GPUs for same-rail communication. Note that the link capacity between ToR and Agg switches is set to 400 Gbps, which is a balanced choice factoring in wiring complexity, optical module availability, and the logical port limits of switching ASICs. To our knowledge, Astral network architecture so far supports the largest scale of same-rank GPU-to-GPU communication within a Pod.

***(iv) Cluster side:*** Considering non-oversubscribed link bandwidth in tier 3, all Agg switches of the same rank in each group connect to 64 core switches for cross-rail communication. Finally, Astral network architecture interconnects 512K GPUs for LLM training and inference in the data center. Also, we share our experience

[1]The optimized ECMP scheme consists of two steps. First, the UDP source port number of each flow is selected to evenly distribute flows across equal paths from the perspective of a source-destination pair. This selection leverages hash linearity properties used by most commodity switching ASICs [51], making a best-effort attempt to minimize load imbalance. Second, if congestion is still detected based on ECN counters on switches (collected every five seconds), the switches report the congestion to a centralized controller. The controller then runs a hash simulator using the same hash algorithm implemented in production switches to reassign appropriate UDP source ports for the congested flows, enabling accurate path selection across all flows from different source-destination pairs. Such UDP source port reassignment takes effect in the next round of collective communications. The effectiveness of reassigning the UDP source port is shown in Figure 17 in the Appendix, which indicates that the ECN counters decrease and eventually stabilize after multiple reassignments. The choice of ECMP as the load balancing mechanism was motivated by operational simplicity, hardware compatibility, and minimal failure impact, as discussed in the Appendix A.

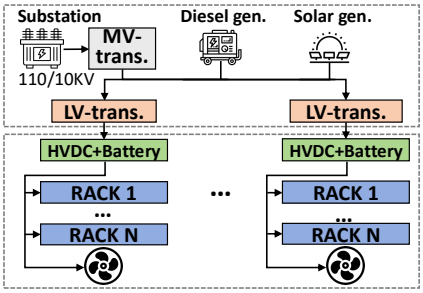
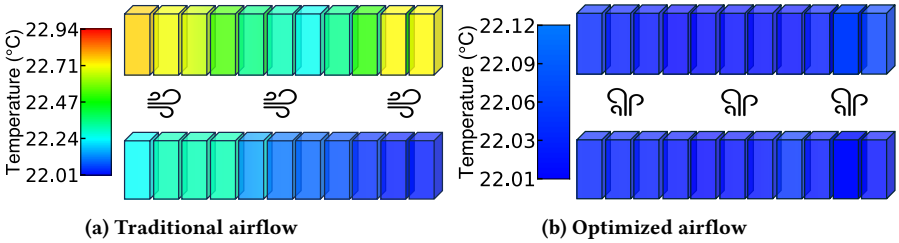



**Figure 4: Hierarchy of the distributed HVDC power system.**




(a) Traditional airflow (b) Optimized airflow

**Figure 5: Temperature distribution with air cooling.**

in extending Astral network architecture to connect multiple LLM data centers separated by hundreds of kilometers in Appendix B, indicating that the performance degradation could be negligible as long as the cross-datacenter bandwidth oversubscription ratio is within a certain range.

**Advantages over other production-ready network architectures.** Astral network architecture is distinguished by the following two key aspects: *(i) Significantly enhancing same-rail communication efficiency.* Meta [20] and ByteDance [27] follow the 3-tier CLOS-like network architecture, without dedicated optimization for same-rail communication. Alibaba [39] adopts a rail-optimized network architecture, enabling same-rail communication on ToR switches, while nevertheless achieving full interconnection on Agg switches. Meta [46] also provides a rail-only network architecture to improve same-rail communication efficiency. However, due to the absence of cross-rail connectivity in the rail-only network, cross-rail communication must traverse intra-host high-bandwidth interconnects, which incurs additional overhead and limits scalability, particularly for all-to-all communication patterns in MoE-based models. Compared to these network architectures, our Astral network architecture strives to maximize same-rail communication, *i.e.*, currently supporting up to 8K GPUs within a single rail, while also enabling cross-rail communication via Core switches. *(ii) Enabling identical bandwidth across all tiers facilitates better performance scaling for jobs deployed in a fragmented manner.* Meta [20], ByteDance [27] and Alibaba [39] opt for bandwidth oversubscription between Agg and Core switches, motivated by their observations of limited bandwidth capacity demands at tier-3 and a trade-off to scale GPU interconnects. However, these network architectures fail to support flexible deployment of LLM tasks and are less likely to achieve performance scaling, as bandwidth oversubscription across tiers increases the likelihood of communication becoming the bottleneck. In contrast to these production-ready network architectures, Astral network architecture preserves identical bandwidth across all tiers and supports significantly larger-scale GPUs with a more promising potential for near-linear performance scaling.

## 2.2 Astral Network Deployment

The physical deployment of new network architecture needs to address practical issues. We encountered and overcame new challenges in power supply and heat dissipation when deploying the Astral network.

**Energy-efficient and stable power management.** Since GPU-heavy workloads in LLM datacenters dramatically increase power consumption, *e.g.*, increased by 8× from 1 kWh without GPUs to 8 kWh with GPUs for one server, the power supply could become a critical bottleneck. To provide an energy-efficient and stable power supply, we develop a distributed HVDC power system and incorporate green energy as a supplemental source.

• *Hybrid-energy HVDC power system.* Traditional Alternating Current (AC) suffers from energy inefficiencies due to energy losses in multiple current conversions with Uninterruptible Power Supply (UPS) batteries. Moreover, UPS battery capacity fluctuates by 20%∼30% under LLM training, which leads to an unstable power supply. By contrast, HVDC power brings three main benefits: *(i)* improved energy efficiency by directly charging the battery; *(ii)* enhanced stability by naturally compensating for battery capacity fluctuations due to its finer power supply granularity; *(iii)* high compatibility with renewable energy sources such as solar and wind. Therefore, we develop a hybrid-energy HVDC power system for Astral infrastructure. Figure 4 illustrates the hierarchy of our distributed HVDC power system, where each unit delivers power to a row of racks and a cooling system. The distributed HVDC power supply for shared racks remains constant (approximately their Thermal Design Power (TDP)). In contrast, a single rack can elastically obtain up to 30% (empirical value) additional power above its TDP since the peak power can exceed TDP.

• *Green energy as a supplemental source.* We build roof-mounted solar power stations and flatland wind power stations to collect and store solar energy and wind energy as a supplement to electricity. According to our 2024 reports, the proportion of renewable energy is 22%, which reduces 778 thousand tons of carbon emissions.

**Energy-efficient yet efficacious cooling system.** Increasing transistor counts and the end of Dennard scaling result in chips with TDP that exceed the capabilities of suboptimally designed air cooling, especially for CPUs/GPUs, reaching thermal limits and performance degradation. To this end, we optimize the cooling system from two aspects when gradually deploying Astral infrastructure:

• *Optimization #1: Airflow optimization in the air cooling system can enable sufficient contact with the heat source.* Astral employs high-density racks to accommodate a block of server and switch equipment, which poses challenges to sufficient contact between the cool airflow and the cooling components. Figure 5a illustrates an issue where unintentionally designed airflow, characterized by air intake from both sides of the server rack, induces uneven rack temperature distribution, with inter-rack variation reaching 1°C. This is because excessively high air velocity at the air outlet reduces the amount of cool air drawn into the nearby racks, resulting in

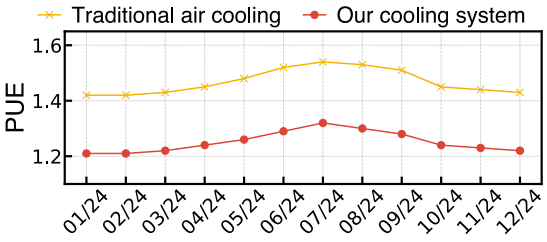



**Figure 6: Evolution of PUE in production.**

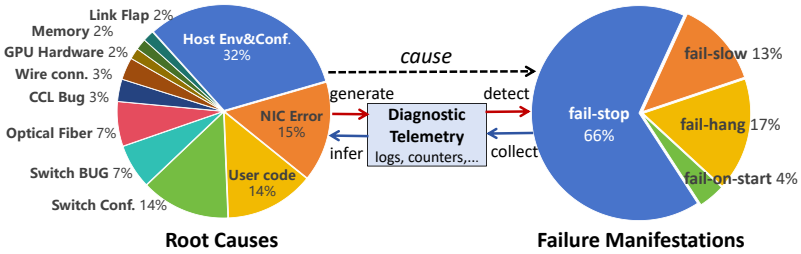



**Figure 7: Anomalies identified in Astral network.**

insufficient contact with the heat sources and uneven temperature distribution. In light of the principle that air velocity is inversely proportional to cross-sectional area when airflow capacity is constant in fluid dynamics, expanding the cross-sectional area can effectively alleviate this issue. To this end, we substitute the horizontal side airflow with the vertical bottom-up airflow as the larger cross-sectional area at the bottom allows for a moderate air velocity. Figure 5b shows that airflow optimization results in a low overall rack temperature, with a temperature variation of only 0.11°C across all racks.

• *Optimization #2: Air-liquid integrated cooling system can facilitate heat dissipation for high-power components.* Given that power-hungry components still experience high temperatures even with well-organized airflow, incorporating a liquid cooling solution, such as cold plates and immersion cooling, becomes essential. However, immersion cooling presents practical challenges such as material compatibility, corrosion, and toxicity. Moreover, we found that a liquid cold plate is sufficient and sustainable in production. Accordingly, air cooling is utilized for overall heat dissipation, whereas cold plates are employed towards localized high-power components. Based on our observations, different workloads (*e.g.*, GPU-intensive or CPU-intensive workloads) require varying power ratios between liquid and air cooling. To accommodate these varying power ratios, we integrate independent air and liquid cooling systems into a unified cooling system, where both share a primary cold source that provides 100% cooling capacity; otherwise, the cooling system cannot adapt to different workload patterns.

Figure 6 compares the PUE of the Astral infrastructure with that of a traditional data center infrastructure. With our cooling systems and power management, the average PUE of Astral infrastructure is reduced by up to 16.34%.

## 3 Astral Monitor and Diagnosis

### 3.1 Challenges and Key Design Rationales

The tightly coupled nature of distributed training systems makes them vulnerable to failure propagation, where anomalies in individual components can rapidly cascade to interconnected nodes and subsystems. Figure 7 presents a structured taxonomy with a statistical distribution of anomalies observed in our production environments, organized through three analytical dimensions: *failure manifestations*, *root causes*, and *diagnostic telemetry*.

The *failure manifestation* refers to the observable symptoms of training process degradation. We identify four distinct failure manifestations: Fail-on-start (4% prevalence), where training jobs abort during initialization; fail-stop (66%), characterized by abrupt job termination after partial execution; fail-slow (13%), manifesting as degraded iteration throughput; And fail-hang (17%), exhibiting complete progress stagnation without termination. Fail-on-start and fail-stop typically generate explicit error logs upon occurrence. In contrast, fail-slow and fail-hang maintain an unhealthy execution state without throwing diagnostic messages.

The *root cause* refers to the fundamental reason behind the *failure manifestations*, encompassing both software failures (such as NCCL bugs and misconfigurations) and hardware issues (including problems with CPUs, GPUs, switches, and links). These underlying issues trigger component-specific *diagnostic telemetry* comprising alerts, counters, and Key Performance Indicator (KPI) metrics, which serve as crucial evidence for root cause analysis.

When a failure occurs, operators must promptly identify the root cause and implement appropriate corrective actions to resolve the issue. However, accurately inferring the root cause from telemetry logs can be challenging due to several factors. During a failure event, an overwhelming volume of logs is typically generated, complicating the analysis process. Moreover, there is no longer a one-to-one relationship between telemetry logs and root causes. For instance, multiple root-cause errors (*e.g.*, GPU failures and network issues) can trigger the same NCCL timeout messages. Conversely, specific failure manifestations, such as fail-hang or fail-slow scenarios, may not generate explicit telemetry logs at all. If the root cause cannot be accurately identified and addressed, it may lead to repeated training failures or inefficient utilization of computational resources.

To cope with the aforementioned challenges, we summarize three key design rationales for our monitoring system:

**Comprehensive full-stack monitoring.** The LLM training platform is a complex distributed software stack that spans the underlying computing and networking resources to the upper-layer user interfaces. Full-stack monitoring helps in rapid response to anomalies and understanding anomalies. In particular, monitoring the application layer (*i.e.*, CUDA and NCCL library) can more directly observe failure manifestations in the training progress, while monitoring the underlying components can directly sense root-cause errors. Intermediate layers, such as network-wide connectivity, can help understand the propagation process from the root cause to the user's perception of the anomaly.

**Cross-hosts and hierarchical correlation analysis.** Detecting and understanding anomalies from unorganized heterogeneous logs is challenging. Traditional detection methods based on threshold-based alerts for individual metrics (*e.g.*, communication time) are not flexible for varying training scenarios and cannot reveal the

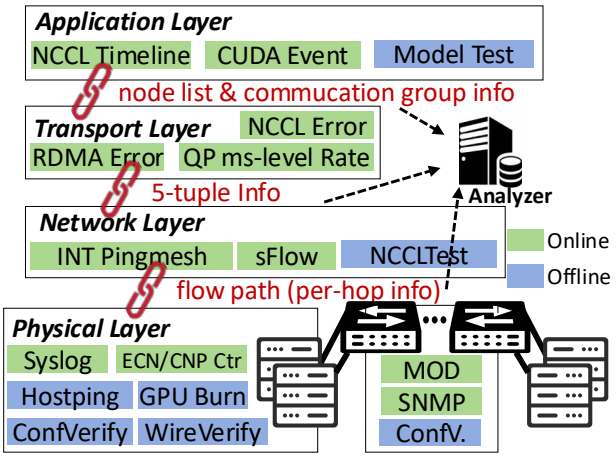



**Figure 8: Astral Monitoring System.**

complete failure stack trace. To this end, cross-host analysis enables threshold-agnostic anomaly detection through horizontal comparison of the metrics across multiple nodes, where similarity-based checking helps identify outlier nodes that deviate from the majority regular pattern. Furthermore, hierarchical correlation delves deeper into root cause analysis by linking application-layer failure manifestations with underlayer metrics through associated information between all the stacked monitors. In this way, we can reveal a clear propagation path from the root cause to the failure manifestation.

**Offline testing before delivery and after unhandled failure.** Offline testing is essential alongside online monitoring. Given that a large proportion of fail-on-start and fail-stop failures stem from pre-existing anomalies (*e.g.*, 32% of failures are caused by the host environment and configurations), conducting offline testing before delivery is vital for minimizing these issues. Furthermore, considering the continuous iteration of software and hardware, it is difficult to identify and isolate all unknown failures by online monitoring alone. Systematic offline toolsets are needed to provide a robust fallback strategy by enabling the reproduction and in-depth analysis of such failures.

Prior efforts have developed specialized monitoring tools tailored for training clusters [11, 16, 27, 48] (see §6 for details). However, they lack an integrated full-stack framework to structurally associate all-layer telemetry logs from multiple heterogeneous entities (*e.g.*, hosts, links, NICs, and switches). In particular, abnormalities at the network layer may spread and cause abnormal indicators of several host monitors and network monitors. However, the inherent differences between network and host logs limit existing comparison-based methods, requiring a systematic approach to correlate them for precise root-cause diagnosis. Simply isolating the hosts could lead to both anomaly recurrence and resource underutilization. Aegis [11] introduces correlation analysis, such as using 'connection reset by peer' logs to identify faulty devices and comparing NCCL instrumentation logs across devices to find the slowest-performing nodes. While insightful, this approach primarily identifies symptoms rather than root causes, and automatic drill-down into the underlying causes of detected anomalies remains unexplored.

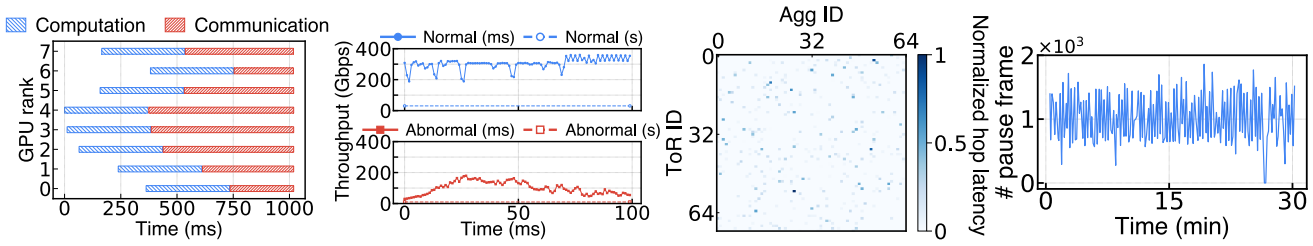
## 3.2 Astral Monitoring Architecture

We develop a full-stack monitoring system in Astral infrastructure, as shown in Figure 8. The key design of the system lies in its ability not only to monitor every component of the cluster, from the application layer down to the hardware level, but also to identify correlations across these layers. The monitoring system includes the following layers:

**Application layer: training progress monitoring.** Application layer monitoring provides a direct reflection of user-perceived training progress. Given that each iteration inherently involves collective communication, monitoring NCCL communication operators allows for precise tracking of both the iteration progress and the associated time costs across individual nodes. Iteration progress, indicated by work request start and finish counts, can be leveraged to identify which node's communication or computation within a specific iteration remains incomplete in the event of an exception. Furthermore, per-iteration computation and communication times serve as key metrics to evaluate whether the training task's performance meets expectations. We can cross-verify the monitored iteration time with the results from the expected operator timeline (*i.e.*, Seer in §4) to identify potential anomalies. Besides, when encountering failures that cannot be resolved online, we conduct offline training on some template models to perform end-to-end testing and in-depth analysis.

**Transport layer: millisecond-level flow monitoring.** Monitoring the transport layer to ensure high performance and reliable data transmission is essential during training. To achieve this, we focus on two primary types of information during transport-layer data transmission. First, we monitor NCCL and RDMA error logs, such as Completion Queue Entry (CQE) error events, which contain the Queue Pair (QP) information of failed transmission. Second, we employ Access Control Lists (ACLs) to filter the first packet of an RDMA Request and parse DMA length in the RDMA Extended Transport Header (RETH), which enables precise calculation of flow throughput with millisecond-level temporal resolution. As shown in Figure 9b, millisecond-level rate monitoring provides finer details compared to traditional second-level rate monitoring. The latter even fails to distinguish an abnormal transmission rate, as the average rate appears low in both cases due to the transmission interval. Millisecond-level monitoring introduces additional bandwidth overhead due to the need to mirror the first packet's header of each RDMA message; however, the overhead is negligible as discussed in Appendix C.

**Network layer: end-to-end path telemetry.** The network layer is responsible for traffic routing, reflected in end-to-end connectivity and latency. Congestion or failures may occur at any hop in the network, adversely affecting end-host functionality and performance. We propose two complementary path telemetry methods. First, we employ passive measurement using sFlow, where the flow path can be restored by sampling and collecting packets from each device. Second, we can use INT-armed ping packets to perform hop-by-hop telemetry of the flow path to obtain connectivity and latency at a hop-by-hop granularity. Compared to existing probing systems [23, 31], combining these two methods enables real-time inference of network flow paths and rapid identification of congested nodes via hop-by-hop information.




(a) NCCL Timeline (b) QP rate (c) INT per-hop latency (d) PFC count

**Figure 9: Anomalies location with hierarchical analyzer.**

**Physical layer: per-node internal state monitoring.** We devise a comprehensive data collection strategy tailored to heterogeneous hardware devices. For each device, we systematically collect internal logs and specialized counter metrics. On end-host systems, we acquire computational unit diagnostics (*e.g.*, CPU/GPU utilization), memory subsystem errors, high-speed interconnect metrics (*i.e.*, PCIe and NVLink), and OS-level event tracking. Network interface analysis includes CNP/PFC/ECN packet counters monitoring. We collect SNMP metrics for network devices while activating Mirror On Drop (MOD) for packet loss detection. Offline validation tools include environmental configuration checkers, stress testing suites (Hostping [32], GPU Burn [4]), and automated wire connection diagnostics.

To fully leverage the multi-layer monitoring metrics for hierarchical correlation analysis, we deliberately maintain the relationships between different layers of monitoring metrics. Specifically, at the application layer, we monitor the host lists engaged in each training task and the communication group information (including detailed QP data) based on the configured parallelism methods. By associating QP data with the five-tuple information (*i.e.*, source and destination IP addresses, source and destination UDP ports, and the IP protocol field), we can link the application layer to the transport-layer metrics. In the network layer, we utilize sFlow for traffic path reconstruction[2] and INT-pingmesh to validate the path and per-hop details. This not only connects upward to the transport layer but also downward to hop-by-hop physical nodes. This integrated hierarchical monitoring system facilitates log consolidation and fault analysis. It can delve into the underlying causes of failure manifestations and deduce the affected applications from the root-cause failures.

## 3.3 Hierarchical Analysis and Cases

Based on the full-stack monitoring system, we build a correlation analysis tool, which combines cross-host correlation analysis and hierarchical correlation analysis to detect failure manifestations and progressively locate root causes. Our analytical algorithm is based on two key observations. First, application-layer monitoring offers broader coverage, indicating that diverse anomalies will invariably manifest in user experience at the application layer. Second, abnormal indicators at the physical layer can often reveal the root causes of failures more directly.

[2]sFlow captures path information by mirroring sampled flow packets at the switches they traverse. In addition, the ECMP hash function that determines the path is provided by our switch vendor.

**Hierarchical Correlation Algorithm.** The algorithm commences with the application-layer monitoring, which is closest to the user's perception. By continuously monitoring each host's computation time and communication time in each iteration, thresholds and anomaly detection strategies are introduced to alert task-level anomalies. We use job-related thresholds obtained by fast forecasts using the Seer for abnormal judgment. In addition to threshold judgment, we also compare horizontally across hosts. If a node lags significantly behind other nodes during fail-slow or fail-stop, it is considered an abnormal node. However, to ascertain the root causes of these host anomalies, further analysis at the subsequent layer is necessary:

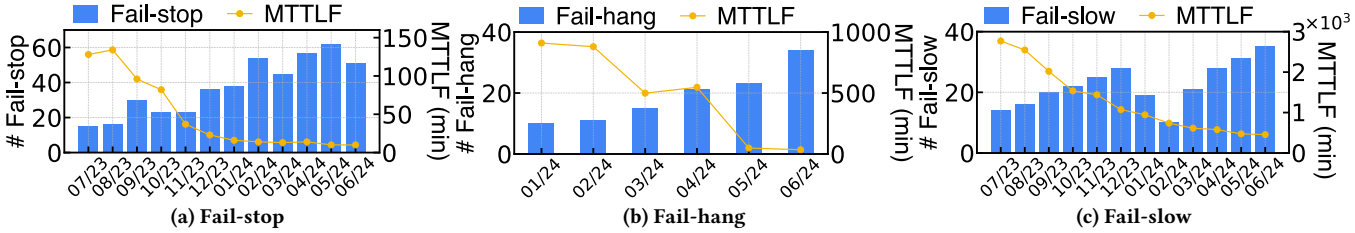
**Branch #1: handling computation anomalies**. When a computation anomaly occurs on a single host, it is associated with abnormal log information at the physical layer of that device. If a fatal error log is matched, actions including isolation, checkpoint loading, and restarting are performed. In cases where computation anomalies occur on multiple devices, empirical evidence suggests that they are typically caused by software or user code anomalies. An alarm will be triggered, requiring manual intervention to determine whether to halt or continue operations.

**Branch #2: handling communication anomalies.** If the application layer detects a communication anomaly, errCQE events and the QP rate information are collected through the maintained job information. To delve deeper into the root cause of the failure, network path information (*i.e.*, the sequence of switches and egress ports, and corresponding hop-by-hop latencies for a flow) is gathered based on the five-tuple information maintained in the QP metadata. Here, we develop two different correlation tools:

- *Identification of failure points through path overlapping.* Network device failures typically impact multiple passing network flows. If a set of errCQE events occurs, the failure points can be identified by locating the overlapping points of multiple affected flow paths.
- *Identification of congestion hotspots via INT per-hop delay.* If the QP rate is abnormal, INT ping detects the hop-by-hop delay and pinpoints the abnormal link. Subsequently, the switch's internal metrics are examined to determine whether there are behaviors of PFC pauses or packet drops.

When the root cause of the network-side abnormality is identified, we adopt a global optimization routing strategy to switch paths by modifying the UDP source port of the flow.

We use a real case to demonstrate the effectiveness of the monitoring system for abnormal locating:




(a) Fail-stop (b) Fail-hang (c) Fail-slow

**Figure 10: Stability improvement after deploying the monitoring system in production.**

**Step #1: Distinguishing computation and communication anomalies at the application layer.** As shown in Figure 9a, we can swiftly identify the device experiencing abnormal computation or communication using the NCCL timeline. In this particular case, our system detected that most devices communicated for a much longer time than the thresholds from the expected operator timeline in Seer.

**Step #2: Investigating anomalies in transport and network Layers.** Upon identifying irregularities in the communication rate, we conducted a detailed analysis of the millisecond-level rate data for each QP. As shown in Figure 9b, specific nodes exhibited QP rates below 50% of the designated link bandwidth. Leveraging the five-tuple information linked to each QP, the system interrogated the sFlow path database to trace the flow path. Subsequent analysis based on INT heatmap (*e.g.*, Figure 9c) revealed forwarding delays of 0.6µs, 179µs, and 266µs at each hop. From these observations, the system deduced congestion in the downlink between the aggregation and the ToR switch.

**Step #3: Identifying root causes of the fail-slow at the physical layer.** Through the collection and analysis of hardware counters from the switches, it was observed that the PFC pause counters significantly exceeded the normal range (in Figure 9d). By correlating this data with the type and volume of flows traversing the link during the specified time frame, the root cause was pinpointed as persistent downstream link congestion resulting from ECMP selecting inefficient paths at upstream links.

One year after deployment, Figure 10 summarizes changes in the location time for three types of faults: fail-stop, fail-hang, and fail-slow, since the implementation of the monitoring system. Although the number of faults has increased (due to the expansion of the network scale), the Mean Time To Locate Failure (MTTLF) for fail-stop and fail-hang faults has been reduced to minutes, achieving up to 12× and 25× reductions, respectively. And the location time for fail-slow anomalies has also been shortened by nearly 5 times. Still, there are always some anomalies that the automatic correlation system cannot recognize. We can only continuously evolve the monitoring system as discussed in Appendix D.

## 4 Astral Seer

The high cost of LLM training makes it essential to forecast the execution timeline of computation, communication, and memory access behaviors. By leveraging this foresight, we can determine optimal parameter selection, validate testbed results, and refine model frameworks offline to improve performance and reliability. To this end, we introduce a brand-new forecasting component Astral Seer.

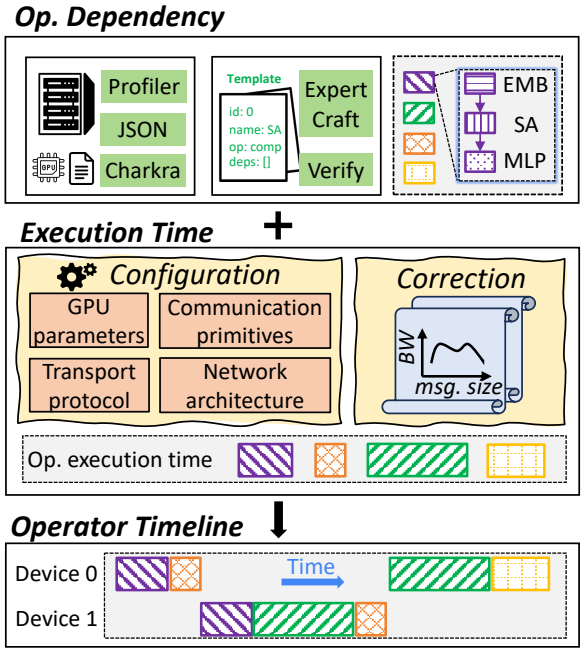



**Figure 11: Astral Seer framework.**

### 4.1 Goals of Seer

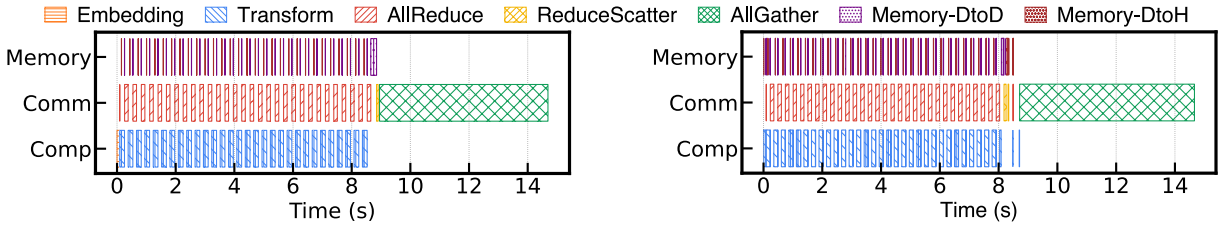
Astral Seer is designed to achieve three pivotal goals:

- *Tuning* the parameters of the model framework, *e.g.*, parallelism and overlap strategies, and configurations in the network, *e.g.*, link bandwidth and transport protocol for optimal performance *before practical deployment.*
- *Verifying* the execution time of in-production LLM training and inference at runtime, thereby aiding in diagnosing failures *during practical deployment.*
- *Exploring* novel model frameworks, *e.g.*, new operators and overlap optimization, and network architecture, *e.g.*, intra-host network and cross-datacenter network topologies, *to upgrade deployment.*

### 4.2 Key Properties for Seer

To achieve the above goals, Astral Seer adheres to the following properties:

- *High efficiency:* The LLM workflow can be decomposed into a series of operators involving computation, communication, and memory access. Efficiently obtaining the execution timeline of LLM operators is a fundamental prerequisite for finding the optimal




**(a) Astral Seer foresight** **(b) Testbed result**

**Figure 12: Timeline comparisons between Astral Seer foresight and testbed result.**

parameters and configurations before model deployment, as well as for verifying in-production results to serve as a reference for diagnosis during runtime.

•*Acceptable accuracy:* Accurately forecasting the execution timeline of each LLM operator is highly valuable. However, unlike simulators, which strive for complete consistency with production results (as they are mainly used in algorithm research), our Astral Seer only requires foresight into the timeline with acceptable accuracy as a reference.

•*Extensibility and configurability:* LLM experts frequently explore new operators and overlap strategies to improve training performance. Meanwhile, they favor configuring different setups for their specific demands. Hence, our Astral Seer should allow easy extension and configuration.

## 4.3 Astral Seer Framework

In Astral Seer, we effectively generate operator-granular timeline forecasts and calibrate our operator execution time with packet-level behaviors to achieve high accuracy. Figure 11 overviews our Astral Seer framework, which mainly consists of the following two components:

**Operator dependency generation.** Given that LLM developers either leverage existing LLM frameworks or introduce new operators to upgrade frameworks, we provide two methods for generating operator dependency:

*(i) Converting from realistic profiling data on GPUs:* One direct and precise way to generate operator dependency is converting from model instances of one iteration collected from GPUs. We use PyTorch profiler to collect GPU traces and export the profiling data to JSON files. By leveraging Pytorch Chakra, model execution can be converted into an executor graph, which includes operators and their dependencies. Our Astral Seer focuses on typical computation, communication, and memory access operators. Table 1 in Appendix showcases the operators used by LLaMA 3 in Seer.

*(ii) Extending with handcraft:* The essence of operator dependency generation is to build the workflow of AI model training and inference in the form of operators. Model research and development experts can follow our templates for operators and their dependencies to manually introduce their newly proposed operators and overlap with existing operators. This template lies in JSON file format, primarily involving the operator dependency, attribute, and corresponding execution time.

**Self-correcting operator execution.** Next, we obtain execution times corresponding to operators. Although using packet-level device simulators to obtain execution time can achieve fair accuracy, it significantly compromises efficiency and is even slower than conducting real experiments. Given that the primary goal of Seer is to enable high efficiency with acceptable accuracy, we obtained the operation execution time by modeling the LLM framework and then iteratively refined it through realistic experiments to improve accuracy.

*(i) Basic modeling:* The execution time of operator kernels can be deduced from the basic model. We formulate the operator execution time as the tensor size divided by the theoretical bandwidth. Specifically, we refine the atomic operation times of computation (*e.g.*, matrix multiplication and addition), memory access, and communication under different parallelism strategies (*e.g.*, Tensor Parallelism (TP), PP, and DP). Each operator can be decomposed into a combination of multiple atomic operations. See Appendix E for the detailed formulation.

*(ii) Self-correction of modeling:* Theoretical bandwidth often fails to accurately reflect actual throughput, as the latter is a realistic measurement resulting from packet-level behaviors influenced by on-training datapath I/O contention and network congestion [9, 30, 41, 45, 47]. Therefore, to improve accuracy, we use actual throughput instead of the theoretical bandwidth in our basic modeling. Specifically, we perform a polynomial curve fit on the throughput measured from the Astral infrastructure. Our bandwidth correction, which implicitly accounts for packet-level behaviors, primarily involves the correlations between arithmetic operations and measured GPU FLOPS, memory access traffic and measured High Bandwidth Memory (HBM) throughput, as well as message size and measured network throughput. Eventually, the operator execution time is calibrated with the actual throughput that inherently reflects packet-level effects.

Given that the actual throughput is affected by hardware parameters and software mechanisms. We offer typical modular hardware and software suites for configurations: *GPU configurations* include specific GPU devices for generating the GPU FLOPS, HBM size, and HBM bandwidth; and *Network configurations* involve network topology, congestion control, and load balance schemes, for generating the ReduceScatter, AllGather, and All-to-All bandwidth.

With operator dependencies and operator execution time, any discrete-event simulation tool can be used to construct the timeline of the end-to-end LLM training and inference process. In brief, Astral Seer can generate LLM operators' timelines within seconds. Also, it has been demonstrated to have high accuracy in production. Figure 12 compares the timeline of one iteration in our Hunyuan model between Seer's foresight and testbed result, showing an

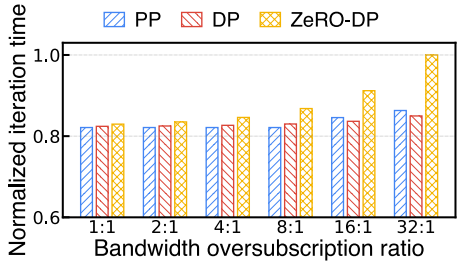



**Figure 13: Training efficiency across datacenters.**

acceptable 0.3% accuracy deviation. Seer maintains acceptable accuracy across dense models, *e.g.*, LLaMA 2 and LLaMA 3. But for MoE-based models, *e.g.*, DeepSeek R1, the accuracy deviation is relatively higher due to unpredictable expert selection and as-yet uncalibrated operators. We are continuously evolving Seer to better accommodate diverse LLM architectures.

## 4.4 Case Studies

In addition to diagnosis use cases in §3, we also present cases that benefit model tuning and network upgrades.

• **Case #1:** *Astral Seer facilitates determining which communication traffic across datacenters can minimally impact training efficiency, and what the bandwidth oversubscription ratio for cross-datacenter links should be.* Due to power supply constraints in data center areas and the need to aggregate remaining GPU resources, several LLM infrastructure providers are building networks that connect multiple datacenters [14, 15, 49]. This involves two natural concerns: First, which communication traffic across datacenters can minimally impact training efficiency? Previous literature shows that PP generates the least traffic (compared to TP and DP) and utilizes the basic Send/Recv for communication, which is insensitive to network bandwidth [39]. Then, the intuition is to make PP traffic pass through datacenters. Nevertheless, we notice that this intuition is not universally applicable. In practice, both PP and DP may be well-suited for cross-datacenter training. Figure 13 shows the cross-datacenter training efficiency on 1K GPUs. As we see, enabling DP traffic across datacenters can achieve better training efficiency in some cases. This is because DP communication is low frequency and can be easily overlapped in asynchronous communication, even though it generates relatively large traffic. We can also observe that enabling memory-optimized ZeRO-DP [42] traffic across datacenters results in worse performance due to its extremely heavy communication traffic. To conclude, allowing traffic across the data center is contingent on the specific case, and Astral Seer can provide recommendations. Second, what is the appropriate bandwidth for cross-datacenter links? Figure 13 illustrates the impacts of varying intra-datacenter to across-datacenter bandwidth oversubscription ratios on training efficiency. As it shows, the training efficiency does not drop significantly until the bandwidth oversubscription ratio reaches 16:1. These impacts could be case-specific. Fortunately, Astral Seer can assist infrastructure service providers in making bandwidth oversubscription decisions.

• **Case #2:** *Astral Seer aids in figuring out the training efficiency after upgrading the intra-host network.* Common wisdom tells us that

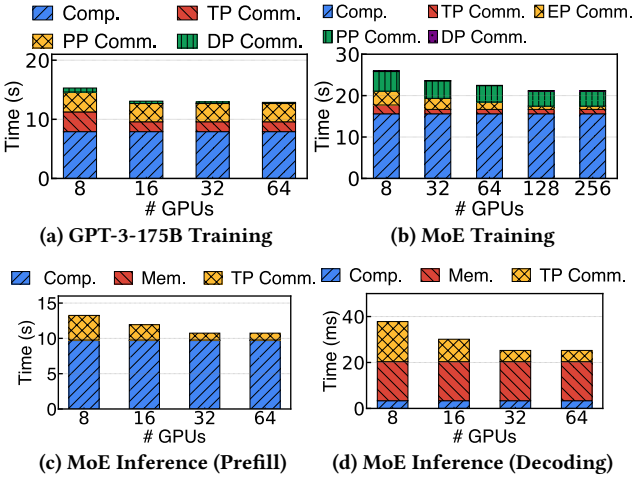


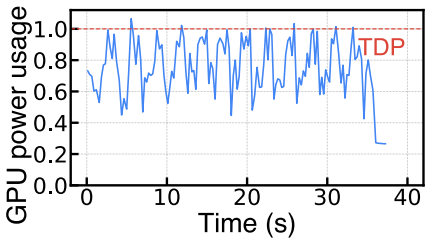

**Figure 14: Performance impacts of intra-host network scale.**

building a large intra-host network with more GPUs (*i.e.*, directly interconnected via NVSwitch) can effectively deliver high performance. However, there is no corresponding quantitative analysis on how large the intra-host network should be to accommodate the corresponding GPUs. By leveraging Astral Seer, we can figure out the scale of the intra-host network that is sufficient for performance. Figures 14a and 14b report the training performance of GPT-3-175B and our in-production MoE-based model under different scales of the intra-host network. As we observe, the MoE-based model benefits more from a large intra-host network because it introduces more all-to-all communication traffic compared to the GPT-3 model. Figures 14c and 14d demonstrate the inference performance of our MoE-based model during the prefill phase and decoding phase under different scales of the intra-host network. These results indicate that increasing the scale of the intra-host network can also contribute to higher inference efficiency.

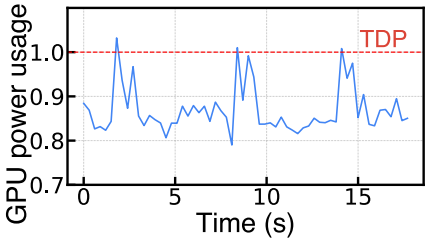
# 5 Deployment and Experience

Astral LLM infrastructure has been gradually deployed since July 2023. As of today, we have interconnected 128K GPUs within two Pods based on Astral network architecture and are extending to 512K-GPUs datacenter. In the following, we share our experience in physically deploying the Astral infrastructure and serving the LLM services.

**Power consumption patterns and our power allocation strategy.** We characterize the power usage patterns of LLM training and inference in production over several iterations and one day: *(i) Exceeding TDP in an iteration:* Figure 15 shows GPU power usage time series during multiple training and inference iterations. For LLM training, the peak power goes up to the GPU's TDP during both forward and backward computation phases, but decreases upon entering the communication phase. For LLM inference, the peak power goes up to the TDP during the prefill phase and drops well below the TDP during the decoding phase. Given that peak power often reaches or exceeds TDP, our distributed HVDC power system overprovisions GPU power to ensure power safety. Specifically, each HVDC power system provides the total TDP to the shared




(a) GPU power usage for training




(b) GPU power usage for inference

Figure 15: GPU power usage over multiple iterations.

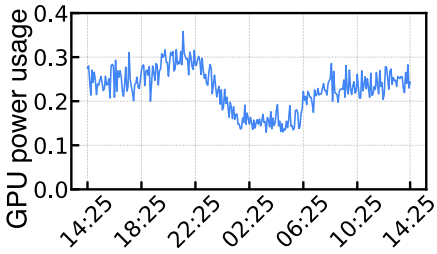



Figure 16: GPU power usage over a day.

racks, where each rack can elastically obtain up to 30% additional power (experience value) above the TDP. *(ii) Tidal effect during a day:* Figure 16 shows GPU power usage time series over one day in a production environment. It exhibits a tidal pattern characterized by high power consumption during the day and a gradual decline between 10 p.m. and 8 a.m. This is because LLM inference, which involves interaction with users, is seldom utilized during nighttime hours. Following the principle of maintaining stable power consumption (since we signed a constant power contract with utility companies), we schedule LLM training and inference tasks. Based on the tidal pattern of LLM inference workloads, our sales model incentivizes customers to perform LLM training at night by offering cheap rental prices.

**Cooling system selection for LLM.** As the company that owns the largest social software in China, we have a long history of building a data center, and its scale is now larger than most enterprise networks. Our cooling system has previously undergone three significant upgrades: from a direct expansion air conditioning system (in 2006), to a centralized chilled water system (in 2010), and then to a distributed air-cooling air handling unit architecture (in 2018). As LLM workloads emerge, air-cooling has reached its TDP and then liquid cooling becomes an inevitable choice. When choosing two advancing liquid cooling technologies, *i.e.*, immersion and cold plate, we first considered the technology with mature ecosystem and multiple supply chains for large-scale use. Then, we considered whether the operation and maintenance of technology is easy and sustainable. Last but not least, we considered compatibility with old data center facilities and air-cooled servers. Considering these factors, each service infrastructure provider can choose the most suitable technology for their situation. Since we have a large number of air-cooled servers and air-cooled data center facilities, cold plate liquid cooling technology is a more suitable choice in our datacenter. Given that the power ratio of liquid cooling to air cooling depends on workloads which is difficult to accurately predict during a datacenter's life cycle (nearly 10 years), we launched an air-liquid integrated cooling system.

**Aggregation of same-rail ToR switches facilitates scalability.** At the nascent stage of constructing Astral network architecture, we have tried a fully interconnected network on tier 2. However, our statistics in production reported that most traffic is within the same rail, benefiting from NVLink-optimized network communication PXN [2, 46]. As a result, a fully interconnected network on tier 2 not only decreased the number of GPUs for same-rail transmission but also exacerbated hash polarization, affecting training efficiency at scale. Then, we decided to aggregate the same-rail ToR switches on tier 2 to maximize the same-rank GPU interconnects for same-rail transmission. Our in-production Hunyuan-MoE model training statistics indicate that the improvement of training efficiency is almost consistent with the expansion of the GPU scale, with only a 0.6% performance loss on 8K GPUs (details are shown in Figure 19 in Appendix). We further monitored the network link utilization and did not see any hotspot paths under high traffic volume, benefiting from our Astral monitoring system that timely detects congestion and schedules the congested flow to a low-load path.

**Wiring and configuration consistency check.** Astral network architecture enables 64K GPUs within a single Pod and accommodates 512K GPUs within a three-tier switching network, complicating wiring. In the early stage of physically deploying Astral infrastructure, the on-site staff made a lot of wiring mistakes and were stuck in correcting them. We ran the wiring verify tool in Astral monitoring system's offline toolsets to address this issue, which collects the slot ID, MAC address, and IP address through the dmidecode command and address resolution protocol to establish the relationship between switch ports and host slots, and then compares it with the network topology rules. Also, since the servers are often rented by customers, the configurations, such as DCQCN and PFC parameters, NVIDIA drivers, and NCCL version, among servers are inconsistent. In production, we noticed that such inconsistency degraded training performance and caused failures. Hence, we used nvidia-smi and NCCL logs to check these configurations, which have also been integrated into our offline toolsets. In consequence, we employ Astral monitoring system to conduct essential checks before delivering hosts to customers.

**Specific driver version can affect the scaling of training efficiency.** During a large-scale LLM training involving 8K GPUs, we encountered an unexpected fail-hang fault. As was usual, we started with NCCL and RDMA error logs in the monitoring system but did not see anything particularly wrong, making it challenging to identify the root cause. Frustratingly, such a fail-hang did not manifest when the scalability of the GPU decreased, rendering binary search diagnosis ineffective for isolating the problematic machine. Therefore, we resort to a dumb way of replacing and rebooting machines in batches. Each iteration of this process took approximately one hour. Several dozen experts in our team worked continuously for 26 hours to replace the machine and this issue did not occur. Through correlation with our monitoring system maintenance records, we traced the issue to an NVIDIA driver update as the only suspicious change. Finally, after consulting with

NVIDIA experts, we confirmed that this issue stemmed from a specific version of the NVIDIA driver.

**PCIe issue causes PFC storms, halving the performance of the entire cluster running multiple jobs.** In this incident, we encountered a dramatic drop in training efficiency to 50% when multiple customers trained their models simultaneously on the Astral infrastructure. Some customers reported that their model training efficiency was reduced by half. Our original Astral monitoring system only identified the congested end-host that generated PFC. However, the original monitoring system could not monitor the physical layer of this PCIe anomaly, making it unable to pinpoint the root cause of the host congestion. After several hours of diagnosis, we figured out that the PCIe of one machine was broken, which eventually triggered PFC and caused congestion spreading, severely affecting training efficiency. Subsequently, we integrated physical layer monitoring for this PCIe anomaly into our monitoring system, empowering it to identify and handle such anomalies automatically.

**Effective scope of cross-host and hierarchical correlation analysis for abnormal behaviors.** Cross-host and hierarchical correlation analysis can effectively address most issues caused by single-device failures or performance degradation that disrupt the entire cluster. For example, when a failure occurs, the log of the abnormal device may be missing, or the training progress (*e.g.*, completed communication/computation operations) may be significantly slower. Nevertheless, our analysis has limitations when dealing with issues not tied to a specific device, such as incorrect strategies at the training framework layer that cause occasional resource preemption and underutilization of operators. It can only reveal that different devices experience random anomalies at different steps, but cannot identify the root cause.

**Self-correcting model improves accuracy.** Many LLM timeline tools have been provided in recent years. To avoid repeated development, we had full expectations to utilize existing solutions such as SimAI and ASTRA-sim. However, after several days of use, one crucial issue we noticed was low efficiency. Specifically, for one iteration of 1K GPU training, we ran ASTRA-sim for one day on a 48-core server to obtain the experiment results. Even though SimAI leverages UNISON [12] that offers a parallel-efficient and user-transparent network simulation kernel to speed up discrete-event simulation, it still requires several hours to complete. This remains insufficient to meet our requirements. In addition, in the currently open-sourced version of SimAI, PP communication traffic is not transmitted over the inter-host network, which may lead to a biased estimation of network communication time. We confirmed this issue with the SimAI team. In consequence, we built Astral Seer. In the beginning, we only constructed basic modeling without correction that used the full GPU FLOPs, HBM bandwidth, and network bandwidth. In some cases, the results between the testbed and the Seer were consistent. However, we realized that Seer's results could deviate from the testbed results by more than 5% when communications become a bottleneck. To this end, we introduced self-correcting modeling driven by realistic testbed data. One limitation could be the data in various cases. But, as an infrastructure provider, we serve many customers under various cases, which naturally have lots of data. Therefore, Astral Seer achieves acceptable accuracy after calibration.

## 6 Related Work

**Network architecture for LLM.** Many companies have released their network architecture tailored for LLM. Alibaba provides HPN [39], which interconnects 15K GPUs within one Pod and imposes bandwidth oversubscription across Pods. ByteDance and Meta have published their CLOS-like network architecture, accommodating 10K GPUs [27] and 32K GPUs [20], respectively. They do not account for reliability due to single-ToR failures. Google discloses torus topology [29] to connect 4K TPUv4, which, however, proves unsuitable for commodity GPUs due to the heterogeneous ports with significantly varying speeds. By contrast, we propose a 3-tier network architecture that can support 512K GPU interconnection without bandwidth oversubscription.

**Monitoring system for LLM.** Existing efforts have developed specialized monitoring tools tailored for LLM clusters. TRANSOM [48] and Minder [16] collect various KPI metrics (*e.g.*, GPU Utilization) and hardware counters (*e.g.*, PFC counts). MegaScale [27] and Aegis further enable progress-aware monitoring through CUDA events and probing of NCCL libraries. They use univariate or multivariate anomaly detection methods (*e.g.*, Z-score [8], LSTM-VAE [38]) to identify outlier hosts. By contrast, we share experience on our full-stack monitoring that empowers cross-host and hierarchical correlation.

**Operator timeline tools for LLM.** Existing AI simulators can forecast operator execution timelines. However, packet-granular simulators, such as ASTRA-sim [1] and SimAI [7], are inefficient at ultra-scale. Instead, operator-granular simulators, such as FlexFlow [26], Daydream [52], dPRO [24], DistSim [33], Proteus [18] and Echo [19], achieve high efficiency. However, they do not incorporate packet-level behaviors, which could raise accuracy concerns. Owing to their limits of being unable to achieve both efficiency and accuracy, we developed Astral Seer.

## 7 Conclusion

This paper presents Astral, our new datacenter infrastructure natively built for LLM, which is capable of interconnecting half a million GPUs. Astral features a same-rail interconnection network architecture, a full-stack monitoring system with cross-host and hierarchical correlation, and operator-granular performance forecasting. These components come together to deliver a high-performance, scalable, flexible, and reliable datacenter infrastructure for LLM workloads. We envision that Astral can be a powerful knob for exploring the extremes of LLM's capabilities.

## Acknowledgments

We thank our shepherd Soudeh Ghorbani and other anonymous SIGCOMM reviewers for their insightful comments. This work was supported by the National Natural Science Foundation of China under Grant Number 62325205, the Key Program of the Natural Science Foundation of Jiangsu Province under Grant Numbers BK20243053 and BK20243059, Gusu Innovation Project for People under Grant Number ZXL2024360, and the Nanjing University-China Mobile Communications Group Co., Ltd. Joint Institute. Yinben Xia and Chen Tian are corresponding authors.

## References


[1] 2025. ASTRA-sim. https://astra-sim.github.io/ Accessed on 2025-01-19.

[2] 2025. Doubling all2all Performance with NVIDIA Collective Communication Library 2.12. https://developer.nvidia.com/blog/doubling-all2all-performance-with-nvidia-collective-communication-library-2-12/ Accessed on 2025-01-19.

[3] 2025. GitHub Copilot. https://github.com/features/copilot. Accessed on 2025-01-19.

[4] 2025. GPU Burn. https://github.com/wilicc/gpu-burn Accessed on 2025-01-19.

[5] 2025. Load Balancing on Aggregated Ethernet Interfaces. https://www.juniper.net/documentation/us/en/software/junos/high-availability/topics/topic-map/load-balancing-aggregated-ethernet-interfaces.html Accessed on 2025-01-19.

[6] 2025. NVLink and NVSwitch. https://www.nvidia.com/en-us/data-center/nvlink/ Accessed on 2025-01-19.

[7] 2025. SimAI. https://ennanzhai.github.io/pub/nsdi25spring-simai.pdf Accessed on 2025-01-19.

[8] 2025. Z-Score: Meaning and Formula. https://www.investopedia.com/terms/z/zscore.asp#:~:text=Z%2Dscore%20is%20a%20statistical,traders%20to%20help%20determine%20volatility Accessed on 2025-01-19.

[9] Saksham Agarwal, Arvind Krishnamurthy, and Rachit Agarwal. 2023. Host Congestion Control. In *Proceedings of the ACM SIGCOMM 2023 Conference* (New York, NY, USA) *(ACM SIGCOMM '23)*. Association for Computing Machinery, New York, NY, USA, 275–287. https://doi.org/10.1145/3603269.3604878

[10] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millicah, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andrew Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. 2022. Flamingo: a visual language model for few-shot learning. 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 1723, 21 pages.

[11] Alibaba. 2025. Evolution of Aegis: Fault Diagnosis for AI Model Training Cloud Service in Production. In *available on request.*

[12] Songyuan Bai, Hao Zheng, Chen Tian, Xiaoliang Wang, Chang Liu, Xin Jin, Fu Xiao, Qiao Xiang, Wanchun Dou, and Guihai Chen. 2024. Unison: A Parallel-Efficient and User-Transparent Network Simulation Kernel. In *Proceedings of the Nineteenth European Conference on Computer Systems* (Athens, Greece) *(EuroSys '24)*. Association for Computing Machinery, New York, NY, USA, 115–131. https://doi.org/10.1145/3627703.3629574

[13] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems* (Vancouver, BC, Canada) *(NIPS '20)*. Curran Associates Inc., Red Hook, NY, USA, Article 159, 25 pages.

[14] Sangjin Choi, Inhoe Koo, Jeongseob Ahn, Myeongjae Jeon, and Youngjin Kwon. 2023. EnvPipe: Performance-preserving DNN Training Framework for Saving Energy. In *2023 USENIX Annual Technical Conference (USENIX ATC 23)*. USENIX Association, Boston, MA, 851–864. https://www.usenix.org/conference/atc23/presentation/choi

[15] Jae-Won Chung, Yile Gu, Insu Jang, Luoxi Meng, Nikhil Bansal, and Mosharaf Chowdhury. 2024. Reducing Energy Bloat in Large Model Training. 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, 144–159. https://doi.org/10.1145/3694715.3695970

[16] Yangtao Deng, Xiang Shi, Zhuo Jiang, Xingjian Zhang, Lei Zhang, Zhang Zhang, Bo Li, Zuquan Song, Hang Zhu, Gaohong Liu, Fuliang Li, Shuguang Wang, Haibin Lin, Jianxi Ye, and Minlan Yu. 2024. Minder: Faulty Machine Detection for Large-scale Distributed Model Training. arXiv:2411.01791 [cs.DC] https://arxiv.org/abs/2411.01791

[17] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805 [cs.CL] https://arxiv.org/abs/1810.04805

[18] Jiangfei Duan, Xiuhong Li, Ping Xu, Xingcheng Zhang, Shengen Yan, Yun Liang, and Dahua Lin. 2024. Proteus: Simulating the Performance of Distributed DNN Training. *IEEE Transactions on Parallel and Distributed Systems* 35, 10 (2024), 1867–1878. https://doi.org/10.1109/TPDS.2024.3443255

[19] Yicheng Feng, Yuetao Chen, Kaiwen Chen, Jingzong Li, Tianyuan Wu, Peng Cheng, Chuan Wu, Wei Wang, Tsung-Yi Ho, and Hong Xu. 2024. Echo: Simulating Distributed Training At Scale. arXiv:2412.12487 [cs.LG] https://arxiv.org/abs/2412.12487

[20] Adithya Gangidi, Rui Miao, Shengbao Zheng, Sai Jayesh Bondu, Guilherme Goes, Hany Morsy, Rohit Puri, Mohammad Riftadi, Ashmitha Jeevaraj Shetty, Jingyi Yang, Shuqiang Zhang, Mikel Jimenez Fernandez, Shashidhar Gandham, and Hongyi Zeng. 2024. RDMA over Ethernet for Distributed Training at Meta Scale. In *Proceedings of the ACM SIGCOMM 2024 Conference* (Sydney, NSW, Australia) *(ACM SIGCOMM '24)*. Association for Computing Machinery, New York, NY, USA, 57–70. https://doi.org/10.1145/3651890.3672233

[21] Talia Gershon, Seetharami Seelam, Brian Belgodere, Milton Bonilla, Lan Hoang, Danny Barnett, I-Hsin Chung, Apoorve Mohan, Ming-Hung Chen, Lixiang Luo, Robert Walkup, Constantinos Evangelinos, Shweta Salaria, Marc Dombrowa, Yoonho Park, Apo Kayi, Liran Schour, Alim Alim, Ali Sydney, Pavlos Maniotis, Laurent Schares, Bernard Metzler, Bengi Karacali-Akyamac, Sophia Wen, Tatsuhiro Chiba, Sunyanan Choochotkaew, Takeshi Yoshimura, Claudia Misale, Tonia Elengikal, Kevin O Connor, Zhuoran Liu, Richard Molina, Lars Schneidenbach, James Caden, Christopher Laibinis, Carlos Fonseca, Vasily Tarasov, Swaminathan Sundararaman, Frank Schmuck, Scott Guthridge, Jeremy Cohn, Marc Eshel, Paul Muench, Runyu Liu, William Pointer, Drew Wyskida, Bob Krull, Ray Rose, Brent Wolfe, William Cornejo, John Walter, Colm Malone, Clifford Perucci, Frank Franco, Nigel Hinds, Bob Calio, Pavel Druyan, Robert Kilduff, John Kienle, Connor McStay, Andrew Figueroa, Matthew Connolly, Edie Fost, Gina Roma, Jake Fonseca, Ido Levy, Michele Payne, Ryan Schenkel, Amir Malki, Lion Schneider, Aniruddha Narkhede, Shekeba Moshref, Alexandra Kisin, Olga Dodin, Bill Rippon, Henry Wrieth, John Ganci, Johnny Colino, Donna Habeger-Rose, Rakesh Pandey, Aditya Gidh, Aditya Gaur, Dennis Patterson, Samsuddin Salmani, Rambilas Varma, Rumana Rumana, Shubham Sharma, Aditya Gaur, Mayank Mishra, Rameswar Panda, Aditya Prasad, Matt Stallone, Gaoyuan Zhang, Yikang Shen, David Cox, Ruchir Puri, Dakshi Agrawal, Drew Thorstensen, Joel Belog, Brent Tang, Saurabh Kumar Gupta, Amitabha Biswas, Anup Maheshwari, Eran Gampel, Jason Van Patten, Matthew Runion, Sai Kaki, Yigal Bogin, Brian Reitz, Steve Pritko, Shahan Najam, Surya Nambala, Radhika Chirra, Rick Welp, Frank DiMitri, Felipe Telles, Amilcar Arvelo, King Chu, Ed Seminaro, Andrew Schram, Felix Eickhoff, William Hanson, Eric Mckeever, Michael Light, Dinakaran Joseph, Piyush Chaudhary, Piyush Shivam, Puneet Chaudhary, Wesley Jones, Robert Guthrie, Chris Bostic, Rezaul Islam, Steve Duersch, Wayne Sawdon, John Lewars, Matthew Klos, Michael Spriggs, Bill McMillan, George Gao, Ashish Kamra, Gaurav Singh, Marc Curry, Tushar Katarki, Joe Talerico, Zenghui Shi, Sai Sindhur Malleni, and Erwan Gallen. 2025. The infrastructure powering IBM's Gen AI model development. arXiv:2407.05467 [cs.DC] https://arxiv.org/abs/2407.05467

[22] 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

[23] Chuanxiong Guo, Lihua Yuan, Dong Xiang, Yingnong Dang, Ray Huang, Dave Maltz, Zhaoyi Liu, Vin Wang, Bin Pang, Hua Chen, Zhi-Wei Lin, and Varugis Kurien. 2015. Pingmesh: A Large-Scale System for Data Center Network Latency Measurement and Analysis. *SIGCOMM Comput. Commun. Rev.* 45, 4 (Aug. 2015), 139–152. https://doi.org/10.1145/2829988.2787496

[24] Hanpeng Hu, Chenyu Jiang, Yuchen Zhong, Yanghua Peng, Chuan Wu, Yibo Zhu, Haibin Lin, and Chuanxiong Guo. 2022. dPRO: A Generic Performance Diagnosis and Optimization Toolkit for Expediting Distributed DNN Training. In *Proceedings of Machine Learning and Systems*, D. Marculescu, Y. Chi, and C. Wu (Eds.), Vol. 4. 623–637. https://proceedings.mlsys.org/paper_files/paper/2022/file/b422680f3db0986ddd7f8f126baaf0fa-Paper.pdf

[25] Jun Huang, Zhen Zhang, Shuai Zheng, Feng Qin, and Yida Wang. 2024. DISTMM: Accelerating Distributed Multimodal Model Training. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*. USENIX Association, Santa Clara, CA, 1157–1171. https://www.usenix.org/conference/nsdi24/presentation/huang

[26] Zhihao Jia, Matei Zaharia, and Alex Aiken. 2019. Beyond Data and Model Parallelism for Deep Neural Networks.. In *Proceedings of Machine Learning and Systems*, A. Talwalkar, V. Smith, and M. Zaharia (Eds.), Vol. 1. 1–13. https://proceedings.mlsys.org/paper_files/paper/2019/file/b422680f3db0986ddd7f8f126baaf0fa-Paper.pdf

[27] Ziheng Jiang, Haibin Lin, Yinmin Zhong, Qi Huang, Yangrui Chen, Zhi Zhang, Yanghua Peng, Xiang Li, Cong Xie, Shibiao Nong, Yulu Jia, Sun He, Hongmin Chen, Zhihao Bai, Qi Hou, Shipeng Yan, Ding Zhou, Yiyao Sheng, Zhuo Jiang, Haohan Xu, Haoran Wei, Zhang Zhang, Pengfei Nie, Leqi Zou, Sida Zhao, Liang Xiang, Zherui Liu, Zhe Li, Xiaoying Jia, Jianxi Ye, Xin Jin, and Xin Liu. 2024. MegaScale: Scaling Large Language Model Training to More Than 10,000 GPUs. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*. USENIX Association, Santa Clara, CA, 745–760. https://www.usenix.org/conference/nsdi24/presentation/jiang-ziheng

[28] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. 2023. BLIP-2: bootstrapping language-image pre-training with frozen image encoders and large language models. In *Proceedings of the 40th International Conference on Machine Learning* (Honolulu, Hawaii, USA) *(ICML'23)*. JMLR.org, Article 814, 13 pages.

[29] Hong Liu, Ryohei Urata, Kevin Yasumura, Xiang Zhou, Roy Bannon, Jill Berger, Pedram Dashti, Norm Jouppi, Cedric Lam, Sheng Li, Erji Mao, Daniel Nelson, George Papen, Mukarram Tariq, and Amin Vahdat. 2023. Lightwave Fabrics: At-Scale Optical Circuit Switching for Datacenter and Machine Learning Systems. In *Proceedings of the ACM SIGCOMM 2023 Conference* (New York, NY, USA) *(ACM SIGCOMM '23)*. Association for Computing Machinery, New York, NY, USA, 499–515. https://doi.org/10.1145/3603269.3604836

[30] Juncai Liu, Jessie Hui Wang, and Yimin Jiang. 2023. Janus: A Unified Distributed Training Framework for Sparse Mixture-of-Experts Models. In *Proceedings of the ACM SIGCOMM 2023 Conference* (New York, NY, USA) *(ACM SIGCOMM '23)*. Association for Computing Machinery, New York, NY, USA, 486–498. https://doi.org/10.1145/3603269.3604869

[31] Kefei Liu, Zhuo Jiang, Jiao Zhang, Shixian Guo, Xuan Zhang, Yangyang Bai, Yongbin Dong, Feng Luo, Zhang Zhang, Lei Wang, Xiang Shi, Haohan Xu, Yang Bai, Dongyang Song, Haoran Wei, Bo Li, Yongchen Pan, Tian Pan, and Tao Huang. 2024. R-Pingmesh: A Service-Aware RoCE Network Monitoring and Diagnostic System. In *Proceedings of the ACM SIGCOMM 2024 Conference* (Sydney, NSW, Australia) *(ACM SIGCOMM '24)*. Association for Computing Machinery, New York, NY, USA, 554–567. https://doi.org/10.1145/3651890.3672264

[32] Kefei Liu, Zhuo Jiang, Jiao Zhang, Haoran Wei, Xiaolong Zhong, Lizhuang Tan, Tian Pan, and Tao Huang. 2023. Hostping: Diagnosing Intra-host Network Bottlenecks in RDMA Servers. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. USENIX Association, Boston, MA, 15–29. https://www.usenix.org/conference/nsdi23/presentation/liu-kefei

[33] Guandong Lu, Runzhe Chen, Yakai Wang, Yangjie Zhou, Rui Zhang, Zheng Hu, Yanming Miao, Zhifang Cai, Li Li, Jingwen Leng, and Minyi Guo. 2023. DistSim: A performance model of large-scale hybrid distributed DNN training. In *Proceedings of the 20th ACM International Conference on Computing Frontiers* (Bologna, Italy) *(CF '23)*. Association for Computing Machinery, New York, NY, USA, 112–122. https://doi.org/10.1145/3587135.3592200

[34] Kiran Kumar Matam, Hani Ramezani, Fan Wang, Zeliang Chen, Yue Dong, Maomao Ding, Zhiwei Zhao, Zhengyu Zhang, Ellie Wen, and Assaf Eisenman. 2024. QuickUpdate: a Real-Time Personalization System for Large-Scale Recommendation Models. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*. USENIX Association, Santa Clara, CA, 731–744. https://www.usenix.org/conference/nsdi24/presentation/matam

[35] Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, Amar Phanishayee, and Matei Zaharia. 2021. Efficient Large-Scale Language Model Training on GPU Clusters Using Megatron-LM. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (St. Louis, Missouri) *(SC '21)*. Association for Computing Machinery, New York, NY, USA, Article 58, 15 pages. https://doi.org/10.1145/3458817.3476209

[36] Maxim Naumov, Dheevatsa Mudigere, Hao-Jun Michael Shi, Jianyu Huang, Narayanan Sundaraman, Jongsoo Park, Xiaodong Wang, Udit Gupta, Carole-Jean Wu, Alisson G. Azzolini, Dmytro Dzhulgakov, Andrey Mallevich, Ilia Cherniavskii, Yinghai Lu, Raghuraman Krishnamoorthi, Ansha Yu, Volodymyr Kondratenko, Stephanie Pereira, Xianjie Chen, Wenlin Chen, Vijay Rao, Bill Jia, Liang Xiong, and Misha Smelyanskiy. 2019. Deep Learning Recommendation Model for Personalization and Recommendation Systems. arXiv:1906.00091 [cs.IR]

https://arxiv.org/abs/1906.00091

[37] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2024. GPT-4 Technical Report. arXiv:2303.08774 [cs.CL] https://arxiv.org/abs/2303.08774

[38] Daehyung Park, Yuuna Hoshi, and Charles C. Kemp. 2018. A Multimodal Anomaly Detector for Robot-Assisted Feeding Using an LSTM-Based Variational Autoencoder. *IEEE Robotics and Automation Letters* 3, 3 (2018), 1544–1551. https://doi.org/10.1109/LRA.2018.2801475

[39] Kun Qian, Yongqing Xi, Jiamin Cao, Jiaqi Gao, Yichi Xu, Yu Guan, Binzhang Fu, Xuemei Shi, Fangbo Zhu, Rui Miao, Chao Wang, Peng Wang, Pengcheng Zhang, Xianlong Zeng, Eddie Ruan, Zhiping Yao, Ennan Zhai, and Dennis Cai. 2024. Alibaba HPN: A Data Center Network for Large Language Model Training. In *Proceedings of the ACM SIGCOMM 2024 Conference* (Sydney, NSW, Australia) *(ACM SIGCOMM '24)*. Association for Computing Machinery, New York, NY, USA, 691–706. https://doi.org/10.1145/3651890.3672265

[40] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. (2019).

[41] Sudarsanan Rajasekaran, Manya Ghobadi, and Aditya Akella. 2024. CASSINI: Network-Aware Job Scheduling in Machine Learning Clusters. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*. USENIX Association, Santa Clara, CA. https://www.usenix.org/conference/nsdi24/presentation/rajasekaran

[42] Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2020. ZeRO: Memory optimizations Toward Training Trillion Parameter Models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*. 1–16. https://doi.org/10.1109/SC41405.2020.00024

[43] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053* (2019).

[44] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and Efficient Foundation Language Models. arXiv:2302.13971 [cs.CL] https://arxiv.org/abs/2302.13971

[45] Guanhua Wang, Shivaram Venkataraman, Amar Phanishayee, Nikhil Devanur, Jorgen Thelin, and Ion Stoica. 2020. Blink: Fast and Generic Collectives for Distributed ML. In *Proceedings of Machine Learning and Systems*, I. Dhillon, D. Papailiopoulos, and V. Sze (Eds.), Vol. 2. 172–186. https://proceedings.mlsys.org/paper_files/paper/2020/file/cd3a9a55f7f3723133fa4a13628cdf03-Paper.pdf

[46] Weiyang Wang, Manya Ghobadi, Kayvon Shakeri, Ying Zhang, and Naader Hasani. 2024. Rail-only: A Low-Cost High-Performance Network for Training LLMs with Trillion Parameters. In *2024 IEEE Symposium on High-Performance Interconnects (HOTI)*. 1–10. https://doi.org/10.1109/HOTI63208.2024.00013

[47] Weiyang Wang, Moein Khazraee, Zhizhen Zhong, Manya Ghobadi, Zhihao Jia, Dheevatsa Mudigere, Ying Zhang, and Anthony Kewitsch. 2023. TopoOpt: Co-optimizing Network Topology and Parallelization Strategy for Distributed Training Jobs. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. USENIX Association, Boston, MA, 739–767. https://www.usenix.org/conference/nsdi23/presentation/wang-weiyang

[48] Baodong Wu, Lei Xia, Qingping Li, Kangyu Li, Xu Chen, Yongqiang Guo, Tieyao Xiang, Yuheng Chen, and Shigang Li. 2023. TRANSOM: An Efficient Fault-Tolerant System for Training LLMs. arXiv:2310.10046 [cs.DC] https://arxiv.org/abs/2310.10046

[49] Jie You, Jae-Won Chung, and Mosharaf Chowdhury. 2023. Zeus: Understanding and Optimizing GPU Energy Consumption of DNN Training. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. USENIX Association, Boston, MA, 119–139. https://www.usenix.org/conference/nsdi23/presentation/you

[50] Zhehui Zhang, Haiyang Zheng, Jiayao Hu, Xiangning Yu, Chenchen Qi, Xuemei Shi, and Guohui Wang. 2021. Hashing Linearity Enables Relative Path Control in Data Centers. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*. USENIX Association, 855–862. https://www.usenix.org/conference/atc21/presentation/zhang-zhehui

[51] Zhehui Zhang, Haiyang Zheng, Jiayao Hu, Xiangning Yu, Chenchen Qi, Xuemei Shi, and Guohui Wang. 2021. Hashing Linearity Enables Relative Path Control in Data Centers. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*. USENIX Association, 855–862. https://www.usenix.org/conference/atc21/presentation/zhang-zhehui

[52] Hongyu Zhu, Amar Phanishayee, and Gennady Pekhimenko. 2020. Daydream: Accurately Estimating the Efficacy of Optimizations for DNN Training. In *2020 USENIX Annual Technical Conference (USENIX ATC 20)*. USENIX Association, 337–352. https://www.usenix.org/conference/atc20/presentation/zhu-hongyu

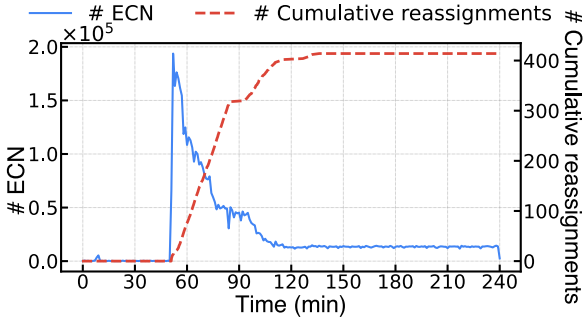



**Figure 17: The effectiveness of our optimized ECMP.**

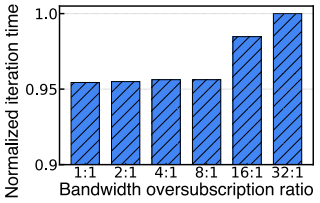



**Figure 18: Training performance across datacenter.**

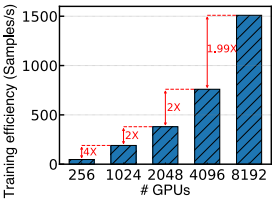



**Figure 19: Training performance at scale.**

# APPENDIX

Appendices are supporting material that has not been peer-reviewed.

## A Rationale for Adopting ECMP Load Balancing

Although ECMP has relatively low efficiency, we still adopt it in production because: First, compared to per-packet load balancing schemes, ECMP, which assigns a fixed path to each flow, can simplify fault diagnosis in practical operations. Also, our fault diagnosis tools rely on a fixed path and would require substantial redesign otherwise. Second, per-flow ECMP is compatible with existing hardware. For example, some legacy NICs, *e.g.*, the CX-6 standard (non-DX) version, only support in-order delivery. Others, such as CX-7 and BF-3, support out-of-order delivery only for RDMA Write operation, but not for RDMA Write with Immediate operation which is commonly used in LLM workloads. Finally, in contrast to per-packet load balancing schemes, per-flow ECMP confines the impact of failures to a limited set of flows. When a link fails, only those flows mapped to the failed path are affected.

## B Astral Network Extension

To consolidate computing power, we are extending the Astral network architecture to connect multiple LLM data centers separated by hundreds of kilometers. In general, the distance between LLM data centers can be hundreds kilometers, resulting in high prices of long-distance optical fiber (*e.g.*, ~70$/km each month and 250K$ of 300km in our rental records for one year) comparable to GPUs. Thus, there is a trade-off between optical fiber bandwidth oversubscription and training performance loss. From our experience with today's training framework, choosing either PP traffic or DP traffic instead of TP traffic across datacenters has the potential to mitigate the side effects of bandwidth oversubscription and long latency, depending on the amount of data volume and communication frequency as mentioned in §4.4. Figure 18 shows the realistic experiment results of using PP traffic across datacenters, indicating that intra-datacenter to across-datacenter bandwidth oversubscription of 8:1 does not affect performance, while 32:1 causes 4.6% performance degradation.

## C Astral Monitoring System Overheads

Millisecond-level monitoring introduces additional bandwidth overhead due to the need to mirror the first packet's header of each RDMA message. In our deployments, this results in an average bandwidth overhead of approximately 0.8Mbps per node. For a cluster with 100K GPUs, the total monitoring traffic is about 10Gbps. This represents only about 0.00005% of the total link bandwidth, rendering the overhead negligible. INT Ping also introduces additional storage overhead, as each ping packet carries extra metadata to detect hop-by-hop nodes and measure latency. For example, in a 10K-GPU cluster, this results in only 173 GB of storage usage per day, and we retain the data for only the most recent 15 days.

**Table 1: Lists of computation, memory access and communication operators in Seer for LLaMA 3.**

| | Operators | Types |
|---|---|---|
| Input Embedding | LoadWeight | Mem. |
| | EmbeddingComputation | Comp. |
| Transfomer Layer | PPRecv | Comm. |
| | RMSNormLoadWeight | Mem. |
| | RMSNormComputation | Comp. |
| | GQAQKVLoadWeight | Mem. |
| | GQAQKVComputation | Comp. |
| | GQACoreAttn | Comp. |
| | GQAAttnProjLoadWeight | Mem. |
| | GQAAttnProjComputation | Comp. |
| | AttnTPAllReduce | Comm. |
| | SwiMLPUpProj | Mem. + Comp. |
| | SwiMLPGateProj | Mem. + Comp. |
| | SwiMLPDownProj | Mem. + Comp. |
| | MLPTPAllReduce | Comm. |
| | PPSend | Comm. |
| Output Layer | Logit | Mem. + Comp. |

## D Evolving Astral Monitoring System

Astral monitoring system offers a viable solution for an automatic correlation analysis tool. This hierarchical correlation analysis essentially draws on the underlying operational habits of experienced operations engineers when locating anomalies. However, there are always some anomalies that the automatic correlation system cannot recognize. Specifically, there are certain failures that do not adhere to the preset logic, such as persistent congestion resulting from the abnormal response of network interface cards (NICs) to PFC frames. For these newly emerging anomalies, we need to use offline tools for in-depth analysis. To append the new anomaly to the automatic monitoring framework, we just need to patch the

new detector at the lower level (*i.e.*, physical layer). With layer-by-layer abstraction, upper-level monitoring is mainly responsible for identifying abnormal manifestations and locating abnormal nodes, introducing minimal changes when dealing with new failures. In this way, we can establish a continuously evolving monitoring system.

**There are edge-case failures that are challenging to handle.** A few failure types are difficult for the online monitoring system to diagnose. During operations, we found that certain software bugs and hardware anomalies require manual intervention. For example, a training task exhibited daily performance fluctuations across multiple nodes within a specific time range. Our monitoring system detected increased CPU utilization on a large number of devices, yet cross-host analysis failed to identify any root-cause nodes, and no critical abnormal logs were captured. After manual investigation, engineers identified the issue: routine port scanning by the network security team repeatedly connected to a port NCCL was actively listening on, causing abnormal CPU usage. The problem was solved after the port was added to the whitelist.

## E Basic Modeling in Astral Seer

Here, we present the formulation for atomic computation, memory access, and communication operator execution time. Specifically, we formulate the atomic computation operator time as:

$$T_{multiplication} = \frac{(2n-1) \times m \times p}{flops} \tag{1}$$

$$T_{addition} = \frac{m \times n}{flops} \tag{2}$$

where $mn$ is the demand matrix addition flops of matrix $A, B \in \mathbb{R}^{m \times n}$; and $(2n-1)mp$ is the demand matrix multiplication flops of matrix $A \in \mathbb{R}^{m \times n}$ and matrix $B \in \mathbb{R}^{n \times p}$. Each computation operator can be decomposed into multiple atomic operations.

The atomic memory access operator execution time of matrix $A \in \mathbb{R}^{m \times n}$ is expressed as:

$$T_{mem} = \frac{m \times n \times f}{hbm\_bw} \tag{3}$$

where $f$ is the binary floating-point format used by the computer, defined by its bit-width. For example, $f$ is 16 bits in the FP16 format.

Taking the basic 3D-parallelism, *i.e.*, TP, PP, and DP, in transformer models as an example [35, 43], the communication operator execution time can be categorized in accordance with communication types as:

$$T_{tp\_comm} = \frac{b \times s \times h \times f}{net\_bw} \tag{4}$$

$$T_{pp\_comm} = \frac{\frac{b \times s \times h \times f}{tp\_groups}}{net\_bw} \tag{5}$$

$$T_{dp\_comm} = \frac{\frac{model\_para\_num \times f}{tp\_groups \times pp\_groups}}{net\_bw} \tag{6}$$

where $b$, $s$, and $h$ represent batch size, sequence length and hidden size in model framework, respectively; $tp\_groups$ and $pp\_groups$ are TP group size and PP group size, respectively; $model\_para\_num$ is the total number of model parameters.