Alibaba HPN: A Data Center Network for Large Language Model Training

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, Dennis Cai
Alibaba Cloud

Abstract

This paper presents HPN, Alibaba Cloud's data center network for large language model (LLM) training. Due to the differences between LLMs and general cloud computing (*e.g.*, in terms of traffic patterns and fault tolerance), traditional data center networks are not well-suited for LLM training. LLM training produces a small number of periodic, bursty flows (*e.g.*, 400Gbps) on each host. This characteristic of LLM training predisposes Equal-Cost Multi-Path (ECMP) to hash polarization, causing issues such as uneven traffic distribution. HPN introduces a 2-tier, dual-plane architecture capable of interconnecting 15K GPUs within one Pod, typically accommodated by the traditional 3-tier Clos architecture. Such a new architecture design not only avoids hash polarization but also greatly reduces the search space for path selection. Another challenge in LLM training is that its requirement for GPUs to complete iterations in synchronization makes it more sensitive to single-point failure (typically occurring on ToR). HPN proposes a new dual-ToR design to replace the single-ToR in traditional data center networks. HPN has been deployed in our production for more than eight months. We share our experience in designing, and building HPN, as well as the operational lessons of HPN in production.

CCS Concepts

• **Networks** → **Network architectures**; **Data center networks**; **Network components**.

Keywords

Network Architecture, AI Infrastructure, Large Language Model, Model Training, Data Center Networks.

ACM Reference Format:

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, Dennis Cai, *Alibaba Cloud* . 2024. Alibaba HPN: A Data Center Network for Large Language Model Training. In *ACM SIGCOMM 2024 Conference (ACM SIGCOMM '24), August 4–8, 2024, Sydney, NSW, Australia.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3651890.3672265

ACM SIGCOMM '24, August 4–8, 2024, Sydney, NSW, Australia

ACM ISBN 979-8-4007-0614-1/24/08
https://doi.org/10.1145/3651890.3672265

1 Introduction

The large language model (or LLM) [13, 15–17, 24] has brought about tremendous revolutions to today's AI and cloud services. The training of an LLM, which has hundreds of billions of parameters, relies on a large-scale distributed training cluster, typically equipped with tens of millions of GPUs. Due to its unique characteristics, LLM training presents new challenges to the design of data center networks.

Problem 1: Traffic patterns. The traffic patterns of LLM training are different from those of general cloud computing in terms of (1) low entropy [19, 22] and (2) bursty traffic. Specifically, general cloud computing generates millions of flows, which gives the network high entropy. Each flow is continuous and low-utilization (*e.g.*, typically below 20% of the NIC capacity), as shown in Figure 1. On the contrary, LLM training generates very few but periodically bursty flows, resulting in low entropy and high utilization for the network. The burst can directly reach the NIC capacity, which is 400 Gbps in our production clusters (as shown in Figure 2). Such traffic pattern undermines the Equal-Cost Multi-Path (ECMP) load balancing scheme widely deployed in our traditional data center networks. Since ECMP employs hash algorithms to distribute traffic evenly across all equivalent paths, ECMP can work well in a network with high-entropy and low-utilization traffic pattern (*i.e.*, the traditional data center network), but not in the case of LLM training which is the opposite. As a result of LLM training's traffic pattern, our traditional data center networks have recently encountered multiple performance issues resulting from the above-mentioned traffic pattern.

Problem 2: Higher sensitivity to faults, especially single-point failures. The LLM training is a synchronous process, where all GPUs cooperate to finish a series of iterations; thus, an anomaly in any GPU can delay or crash the entire training process. This means LLM training is more sensitive to a fault than traditional cloud computing. We found that the most significant impact on our LLM training was caused by Top-of-Rack (ToR)-related single-point failures, which can affect a wide range of GPUs. Further, the failures in LLM training are costly. Our production statistics show that a fault in LLM training can cost the company 20× more than when it occurs in general cloud computing.

Our contribution: Alibaba High Performance Network (HPN). In this paper, we share our new data center network architecture,

HPN, which is designed for LLM training.[1] Compared to traditional data centers [19, 21, 53, 58], HPN makes the following contributions:

- HPN proposes a novel ToR deploying design (*i.e.*, the non-stacked dual-ToR design) to avoid ToR-related single-point failure (§4.2). Compared with the stacked dual-ToR solutions proposed by switch vendors (§4.1), non-stacked dual-ToR eliminates the direct synchronization between two switches, greatly improving the reliability in the large-scale deployment.
- By addressing the challenge in adopting the latest 51.2Tbps switching chip (§5.1) and employing rail-optimized network (§5.2), 1K GPUs can be contained in a tier1 network, making 96.3% of training jobs enjoy the best network performance. More importantly, HPN accommodates 15K GPUs—which is the conventional size of a training cluster—in a 2-tier dual-plane architecture (§6), rather than a 3-tier Clos architecture, significantly decreasing the occurrence of ECMPs. Such a design feature not only avoids hash polarization [18, 72] in the Aggregation layer (by dual-plane design), but also decreases the searching space by 1-2 magnitudes for selecting ideal paths carrying different elephant flows.
- We further share the design considerations of supporting larger scale (§7) and inference (§8) in the coming future, as well as experience during the design, deployment and operation of HPN (§10).

Deployment. HPN has been built and used in our production for over eight months, and we have not encountered any ToR-related single-node failure. Our experience shows that the LLM training throughput with HPN is 14.9% higher than that of traditional data center networks (§9).
Ethics: *This work does not raise any ethical issues.*

2 Background and Our Goals

2.1 Large Language Model (LLM) Training

LLMs would contain more than $100B$ parameters and are constructed with multiple layers. The efficient training of these models requires dozens to thousands of GPUs. Mainstream training frameworks (*e.g.*, Megatron-LM [41, 67] and Deepspeed [30]) employ a hybrid of parallel strategies to coordinate all GPUs efficiently.

Data parallelism (DP). The training dataset is evenly distributed among all GPUs, where each GPU has a replicate of the entire model. In each iteration, all GPUs employ AllReduce to synchronize calculated gradients.

Pipeline parallelism (PP). A model is divided into multiple stages, each containing a series of continuous layers of the model and served by different GPUs. Each GPU in the pipeline receives the input from the previous stage and sends the output to the next stage in the pipeline.

Tensor parallelism (TP). The entire model or each layer in PP can further be horizontally split. Therefore, each layer is distributed across a group of GPUs. In each iteration, GPUs in the same TP group employ AllReduce/AllGather to synchronize calculated outputs and corresponding gradients.

Considering (1) the large training scale and (2) various parallel strategies, traffic patterns observed in LLM training are quite different from traffic patterns in either elastic cloud computing or traditional DNN training.

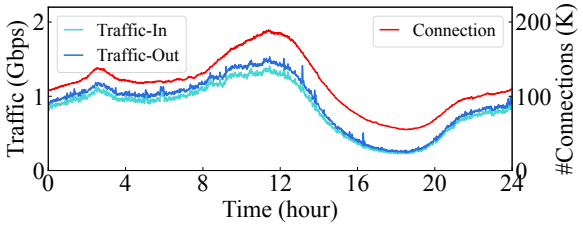


Figure 1: Traditional cloud computing traffic pattern.

2.2 Traffic Patterns in LLM Training

Traditional data center network architectures (*e.g.*, fat tree [53]) are mainly designed for general, elastic cloud computing. We observed that the traffic patterns of LLM training are different from those of general cloud computing.

Periodic burst in network utilization. In our production, general cloud computing generates millions of flows, and traffic utilization generally stays below 20%. The overall traffic pattern is relatively continuous and steady, which slowly changes on the hourly scale (as shown in Figure 1). On the contrary, the LLM training generates very few, but periodically bursty flows, as shown in Figure 2 and Figure 3. More specifically, Figure 2 shows the throughput of a NIC (2×200Gbps) in our in-production LLM training. The NIC periodically transmits a large amount of data, which instantly fulfills the network capacity (400Gbps) and lasts from a few seconds to tens of seconds. This traffic pattern arises from the need for gradient synchronization. A typical LLM training involves a series of iterations, and during each iteration, data synchronization is required between different parallel groups (each with many GPUs). The bursts occur during the backward phase of each training iteration, where all data parallel groups need to synchronize gradients through the AllReduce collective communication operations.

The sudden bursts of network utilization imply that LLM training requires extremely high network bandwidth. We, therefore, need to ensure that the network for LLM training can provide sufficient physical bandwidth for the bursts to avoid packet loss. In addition, the synchronicity of traffic indicates that LLM training is particularly sensitive to long-tail delays. Any long-tail flow would be an obstacle to the completion of the entire collective communication operations, putting all parallel groups on hold.

Small number of flows. As shown in Figure 1, a general cloud computing instance typically generates hundreds of thousands of connections; on the contrary, each node in the LLM training generates very few connections. As shown in Figure 3, a GPU uses only a few dozen to hundreds of connections. Combined with the previously mentioned bursty high network utilization in the training process, the actual data volume required to be sent per flow is substantial.

[1] We choose to design and build a new data center network based on Ethernet rather than adopting an InfiniBand-based solution (*e.g.*, DGX SuperPod [21]) for two reasons: (1) avoiding vendor lock-in and (2) leveraging the power of the entire Ethernet Alliance for faster evolution.

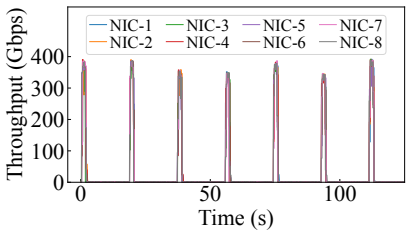


Figure 2: NIC egress traffic pattern during production model training.

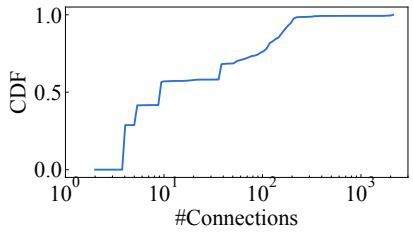


Figure 3: Number of connections per host.

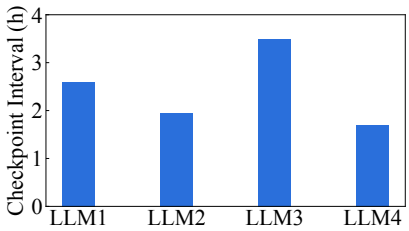


Figure 4: Checkpoint intervals of representative LLM jobs.

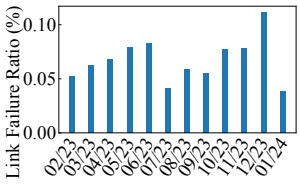


Figure 5: Monthly link failure ratio.

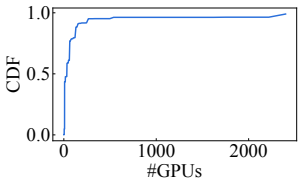


Figure 6: #GPUs used in production training jobs.

Traditional data centers are a poor fit for the traffic patterns of LLM training. Traditional data center networks employ ECMP as the load-balancing scheme. ECMP assumes that the hash algorithm can effectively distribute the traffic evenly across all equivalent paths when there are a large number of flows in the network. This assumption holds under general cloud computing traffic patterns, which typically generate millions of flows (as shown in Figure 1). However, such an assumption is no longer valid in the case of LLM training, which involves a few large flows (also known as having an elephant flow distribution). In our practice using traditional data centers for LLM training, we have already encountered multiple performance issues due to hash polarization, severely affecting the LLM training efficiency.

Even worse, due to the 3-tier architecture nature of traditional data center networks, the forwarding of an elephant flow would go through three times of hashing (*i.e.*, ToR, Aggregation, and Core layers). Since the input for each hashing (*i.e.*, the five-tuple of the flow) remains the same, the effect of such "cascading" hashing can lead to an even more severe load imbalance (*i.e.* hash polarization [18, 72]). We have observed many load imbalance cases in our production caused by hash polarization, especially in the cross-pod communication scenarios where traffic needs to pass through all three layers of switches to arrive at the destination.

2.3 LLM Training is Sensitive to Faults

Compared to general cloud computing, failures have a more severe impact on LLM training.

First, LLM training is more sensitive to failures. In LLM training, multiple GPUs cooperate to finish each iteration, and we need many iterations (lasting dozens of days) to complete the entire training process. Therefore, a failure on any GPU or host can directly slow down the current iteration, or even crash the entire LLM training process, (detailed evaluations are in §9.3).

Second, failures in LLM training can result in significant costs. When a failure occurs, the LLM training typically leverages checkpoints to recover from the failure [41, 43]; however, since generating a checkpoint in LLM training requires substantial storage (*e.g.*, 30GB per GPU) and high overhead (*e.g.*, 100s), our customers choose to generate a checkpoint only every few hours. For example, in Figure 4, we record the checkpoint generation intervals in four representative production LLMs, which typically range from two to four hours, (even at these high intervals, the overhead introduced by checkpointing is still around 5%). This means that once a failure occurs, the entire training must roll back to several hours earlier and be retrained. Given that training costs are 20K dollars per hour for a training task utilizing 3K GPUs, a failure could lead to a financial loss of 30K dollars.

Single-point failure matters. While the tier2 and tier3 layers have abundant redundant links, each NIC connects to a ToR via a single link, posing a single point of failure risk. When the accessing link (*i.e.*, the link connecting a NIC and a ToR) is down, it causes the corresponding host to disconnect. Even worse, the failure of a ToR can make dozens or even hundreds of hosts unavailable, leading to severe service quality degradation. LLM training requires thousands of GPUs to train collaboratively, involving dozens of ToRs and thousands of optical modules and links. With such a large scale, it is extremely hard to guarantee that no network device is down. Tools like monitoring and troubleshooting systems work for reactively localizing root causes of failures, but cannot prevent training from crashing. In our operating clusters, as shown in Figure 5, 0.057% of NIC-ToR links fail each month, and about 0.051% of ToR switches encounter critical errors and crashes. Under this high failure rate, a single LLM training job would encounter 1-2 crashes each month. Furthermore, 5K-60K link flapping cases happen each day, introducing temporary performance degradation as well.

2.4 Goals with Practical Considerations

Based on the unique characteristics of LLM training, we decided to build a new network architecture specifically for LLM training. We should meet the following goals.

G1: Scalability. Figure 6 shows that the number of GPUs required by a single training job in production is less than 3K. To accommodate the requirement evolution in the coming future, we set the *primary capacity goal* of containing 15K GPUs, which is also in line with mainstream LLM training providers (*e.g.*, Google, AWS, Azure, and NVIDIA) offering each training cluster with 10K-30K GPUs [12, 20, 21, 23]. Based on our experience, the number of model parameters may continue to rise by an order of magnitude in the next several years (*i.e.*, from 1 trillion to 10 trillion parameters). The *additional capacity goal* of our data center, therefore, is to be able to support the scale at 100K GPUs.

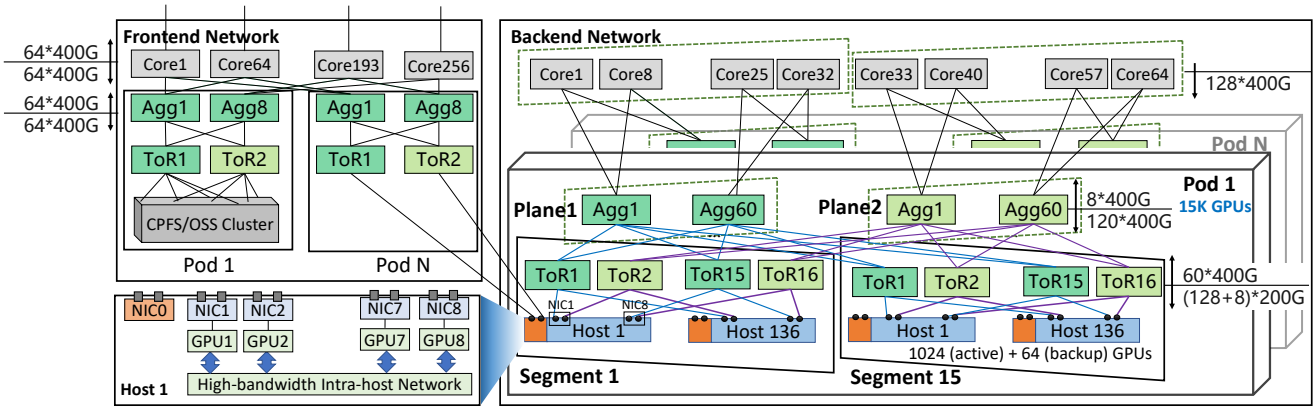


Figure 7: HPN overview. A solid parallelogram represents a segment (containing 1024 active GPUs and 64 backup GPUs). Two dotted parallelograms represent dual-plane. A cube contains an entire Pod (containing 15K GPUs).

G2: High performance. The performance is important. To enhance the performance, our design should minimize network hops as much as possible. Reducing the number of hops is not only to lower latency but also to decrease the times of ECMP hashing, making the path selection scheme more precise. Furthermore, our design should allow as much direct GPU-to-GPU communication as possible, rather than through the network.

G3: Single-ToR fault tolerance. Based on our observations in §2.3, the most critical risk potentially affecting the reliability of the LLM training is the fault of single-ToR. Our new network architecture, therefore, should fundamentally avoid the failure of single-ToR at the topology level.

3 HPN Architecture Overview

We design and build HPN, our new data center network specifically for LLM training. HPN meets our goals in §2.4. Figure 7 presents the overview of HPN.

HPN consists of frontend and backend networks. The backend network mainly supports the traffic during the training process, while the frontend network carries the others (*e.g.*, the traffic for management, inference and storage). For LLM training, we mainly focus on the backend network of HPN. In HPN's backend network, each host is equipped with 8 GPUs, each connected through a dedicated high-bandwidth intra-host network (*e.g.*, NVLINK [49]). Each GPU can directly communicate with other GPUs via this intra-host network with 400GBps-900GBps (bidirectional).

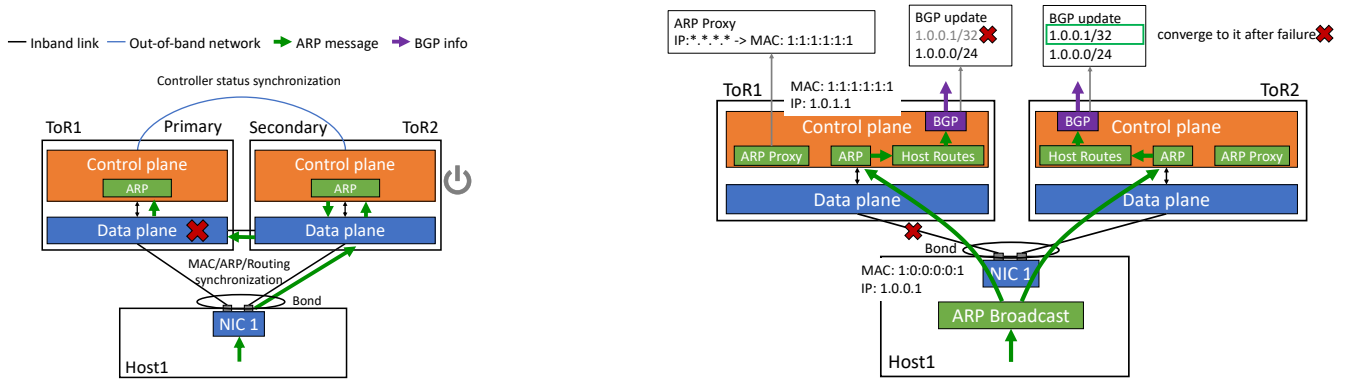
To offer the maximum network capacity, we equip each host with 9 NICs, each with 2×200Gbps. One of these nine NICs (*i.e.*, NIC0 in Figure 7) is connected to the frontend network, while the remaining eight NICs connect to the backend network to carry traffic during the LLM training. Each of these eight NICs serves for a dedicated GPU (named rail), and thus each GPU has a dedicated 400Gbps of RDMA network throughput, resulting in a total bandwidth of 3.2Tbps. Such a design aims to maximize the utilization of the GPU's PCIe capabilities (PCIe Gen5×16), thus pushing the network send/receive capacity to the limit. The two ports of each NIC are connected to different ToRs, respectively, forming a dual-ToR design. This dual-ToR design aims to avoid the single-ToR failure issue, which is a key design detailed in §4.

In tier1, given that the intra-host network (*e.g.*, NVLINK) has higher capacity than the inter-host Ethernet network, we employ the rail-optimized design [21], where NICs belong to different rails are connect to different set of ToRs. Combined with the abovementioned dual-ToR design, a host connects to 16 ToRs in the backend network. By fully utilizing the 51.2Tbps switching chip's capability, HPN enables 1024 GPUs to be interconnected through a single-layer network, called a segment (§5). Figure 6 quantifies our benefit: about 96.3% of in-production LLM training jobs take less than 1K GPUs; thus, in HPN, these jobs can be put in one segment, achieving the utmost network performance.

Tier2 interconnects multiple segments. We incorporated the dual-ToR feature to design a dual-plane forwarding at this layer (as shown in Figure 7). The traffic sent from port 0 of the source NIC can be forwarded through the network and eventually received only by port 0 of the destination NIC, physically isolated from traffic from port 1 of source NICs. This dual-plane design avoids the issue of hash polarization at the Aggregation layer, without affecting the 1:1 network bisection bandwidth. Furthermore, the dual-plane design doubles the number of GPUs covered by a Pod, supporting the interconnection of 15K GPUs (§6).

For a larger scale that may be required by a single job in the future, we also designed Core layer interconnections between different Pods. Since a single Pod scale has already reached 15K GPUs, jobs that need to go through the Core layer for coordination are rare. In our design, we opted for an oversubscription of 15:1 in the Aggregation-Core layer. Based on the traffic characteristics of LLM training, we assign the PP communication across Pods, ensuring that cross-Pod transmission minimally impacts on the end-to-end training performance (§7).

For the frontend, it is mainly used to carry traffic for management, inference and storage. The physical decoupling of the frontend and backend networks ensures that the frontend traffic does not affect the main procedures of training jobs. Furthermore, the frontend network is designed with an oversubscription of 1:1, making it extensible to more scenarios, such as the mixed deployment of LLM training and inference (§8).



(a) Stacked dual-ToR.

(b) Non-stacked dual-ToR.

Figure 8: Dual-ToR solutions. In stacked dual-ToR (a), the malfunction of ToR1's data plane would eventually trigger the offline of ToR2. In non-stacked dual-ToR (b), two ToRs run independently.

4 Access: Non-stacked Dual-ToR

In traditional data center networks, two ports of each NIC are aggregated through one cable/fiber connecting to a ToR switch, called single-ToR design (widely used in the majority of current cloud providers [21, 53, 55, 63]). Single-ToR is, however, vulnerable to switch/link failures, significantly affecting the LLM training (mentioned in §2.3).

Dual-ToR design, on the contrary, connects two ports of each NIC to different ToRs in an active-active way. These two ports are configured with the same IP and MAC addresses. If one ToR (or a port) is down, the other can still work. Furthermore, since two ports in the same NIC share the same Queue Pair (QP) contexts, the switching of traffic would not lead to the break of active flows and be transparent to upper-layer applications. Figure 8a shows a typical dual-ToR design, called *stacked dual-ToR*, which has been realized by commodity solutions such as virtual Port Channel (vPC) [29], M-LAG [8, 42, 44] and stacking [26, 33, 37]. In the stacked dual-ToR, the two ToRs are connected through a direct link, an essential design used to synchronize data plane forwarding information such as ARP and MAC. The host employs Link Aggregation Control Protocol (LACP) [2, 39] to aggregate two ToRs as one. In principle, the stacked dual-ToR solution would significantly reduce the degradation caused by the independent ToR switch failure, which is widespread in production.

Nevertheless, directly applying these commodity solutions causes many issues (detailed in §4.1). We propose a new dual-ToR scheme called non-stacked dual-ToR (§4.2).

4.1 Stacked Dual-ToR and Problems

As shown in Figure 8a, a typical dual-ToR design uses a link to connect two ToRs to synchronize the states. The control planes of the two ToRs operate in a primary-secondary way, synchronizing the controller states via an out-of-band network to ensure the correct primary selection. On the host side, dual-ToR access can be implemented using bond [40] (a built-in Linux module). Specifically, bonding mode 4 (dynamic link aggregation) [38] enables automatic load balance between two ports and automatic traffic rerouting when one link/ToR switch fails. Stacked dual-ToR minimizes modifications on the host side, providing convenience for the deployment and upgrade. However, in practice, we found that the stacked dual-ToR design introduced many risks.

Stack failure. Issues with the direct link between ToRs or a failure on one ToR can lead to anomalies, causing all NICs under the dual-ToR offline. For example, as shown in Figure 8a, ToR1 and ToR2 are primary and secondary, respectively. Suppose that the data plane of ToR1 does not function due to the MMU overflow [7], while its control plane does not recognize this root cause. Therefore, ToR1 and ToR2 can no longer synchronize ARP or MAC through the direct link. Since the out-of-band network is functional, the control planes of ToR1 and ToR2 still negotiate normally. From ToR1's perspective, ToR2 is unreachable in the data plane, and ToR1 should continue working in primary states. From ToR2's perspective, ToR1's data plane is unreachable, which means that the forwarding information cannot be synchronized anymore, but ToR1 still runs in Primary states. To prevent inconsistent forwarding in the data plane, ToR2 chooses to shut down itself. However, since the data plane of the remaining ToR1 is down, all NICs under this dual-ToR become unavailable. This rack-level failure causes severe unavailability issues in production.

Issues resulting from ToR upgrades. During dual-ToR upgrades, it may happen that one ToR runs the newly upgraded version, while the other runs the old version. This situation can lead to incompatibility issues during the synchronization of control-plane information via RPC, since the values and fields of RPCs of the two versions may not match at this point in time. ToRs can be down if such an incompatibility issue happens. While switch vendors have proposed solutions for In-Service Software Upgrade (ISSU) [3, 25, 52], it only works when the diff between the old and new versions is sufficiently small. Nevertheless, in the past three years, we observed that 70% of upgrades in our ToRs do not meet this assumption of ISSU, resulting in poor upgrading effectiveness.

Summary: Failures caused by stacked dual-ToRs. According to our internal failure reports, over 40% of critical failures in our traditional data centers were caused by the abovementioned two

categories of issues introduced by stacked dual-ToR in the past three years.

4.2 Our Solution: Non-stacked Dual-ToR

The root cause of failures in stacked dual-ToR is the strong dependency on synchronization via the direct link between two ToRs. To eliminate the culprit of failures, we decide to construct a non-stacked dual-ToR scheme, as shown in Figure 8b, which removes the direct link between two ToRs.

Such an idea introduces a new challenge: how to synchronize ToRs without a link directly connecting them? Solving this challenge is not straightforward. In the existing stacked dual-ToR schemes, because two ToRs are directly connected by a link, they can negotiate a shared sysID via the direct link. This enables the host to talk to stacked dual-ToR switches as one device through LACP [2]. However, since we want to remove the direct link between the ToRs, causing them to be independent, this means that they can no longer use the LACP to negotiate. Therefore, we need to design a new technology to "disguise" the two ToRs as a virtual single device through an implicit approach, enabling the host to talk to dual-ToR via LACP.

The original LACP works as follows. Two ToRs' LACP modules receive the LACP Data Unit (LACPDU) from the downstream host (Host1), respectively. Before generating the response LACPDU, to make dual-ToR functional, two ToRs need to negotiate through the direct link to ensure that they use (1) the same MAC address and (2) different portIDs as inputs. The LACP module fills the Actor information fields in LACPDU including sysID (generated from the negotiated MAC address), portID and other fields, and then sends it back to the host.

Bundling two independent links. Building non-stacked dual-ToR is non-trivial since we must guarantee that the same MAC address and different portIDs are used in dual-ToR switches during the LACP negotiation. We deeply cooperate with our switch vendors to implement a customized LACP module to achieve this goal.

① **Same MAC address:** When the ToR's LACP module receives a LACPDU, it generates sysID according to a pre-configured MAC address. Choosing this pre-configured MAC address is not straightforward. This MAC address should be the same between two switches in the same dual-ToR set. In addition, this MAC address must not be used by any hosts; otherwise, a conflict may happen. Thus, we choose an RFC-reserved virtual router MAC address, `00:00:5E:00:01:01` [1], as the pre-configured MAC address.

Principally, in the same layer2 subnet, ToR switches in different dual-ToR sets must not use the same pre-configured MAC address to avoid MAC address conflict. In Alibaba Cloud, we completely employ layer3 forwarding (via BGP) among different dual-ToR switches. Therefore, different dual-ToR sets naturally belong to different layer2 subnets, eliminating the possibility of MAC address conflicts.

② **Different portIDs:** By default, two ToR switches would generate the same portID for the same host (since their wiring is similar). To generate different portIDs, each ToR switch performs a bit shift operation on the original portID. The operation is: $p' = p + offset_i$, where p is the original portID. We set different $offset_i$ for switches in the same dual-ToR set, and it is an integer higher than 256 (*e.g.*, 300). Since the total port number on a ToR is less than 256, the calculated portID (p') would not conflict with other system settings.

Last but not least, the host should be able to update ARP information concurrently on the two ToRs via duplicating each ARP message to both ports on the NIC (*i.e.*, ARP Broadcast module in Figure 8b). and loads by default. By far, all hosts can support our non-stacked dual-ToR scheme.

Leveraging BGP under failures. In normal situations, each ToR switch updates BGP with the default subnet route (1.0.0.0/24 in Figure 8b), making both ToR switches equal-cost paths. When a failure happens, we need to ensure that routing converges to the remaining functional path, which is a primary feature of BGP. Therefore, we decide to maximally multiplex BGP in the failure handling.

In HPN, all ARPs learned on the ToRs are converted to /32 host routes in BGP (Host Routes module) and recognized in the entire network. When the NIC-ToR link fails, the corresponding ARP item will be withdrawn by the switch, as well as the corresponding host route, triggering the updating of BGP. As visualized in Figure 8b, 1.0.0.1/32 is withdrawn by ToR1 when the failure happens. As a result, in the entire cluster, the forwarding path to Host1 would converge to pass through ToR2, since only ToR2 publishes the longest prefix route (1.0.0.1/32) in BGP.

Specifically, intra-segment traffic should be carefully handled, since it can be directly forwarded in layer2 by default. The de-facto aging time of items in the MAC address table is 5 minutes, which leads to a black hole during the NIC-ToR link failure. To solve this problem, we turn off the layer2 broadcast. In addition, we further implement an ARP proxy in the switch. ARP proxy uses the switch's own MAC to respond to all ARP requests from hosts, forcing all layer2 forwarding to terminate at the ToR switch and enabling layer3 forwarding for intra-segment traffic. Therefore, intra-segment traffic can be correctly routed according to BGP to avoid the black hole.

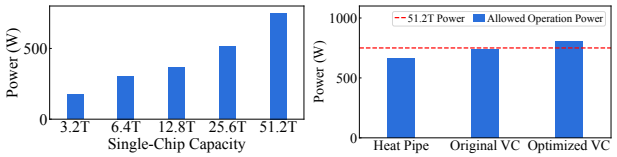
Lessons. First, why not use BGP on the host to replace LACP for NIC→ToR negotiation? Using BGP requires all hosts to participate in the entire BGP updating in the cluster, slowing the convergence speed and greatly complicating the cluster-wide upgrade of BGP daemons (*e.g.*, from operating O(1K) devices to O(10K) devices). Second, using configurable LACP and converting ARP to host routes result in inherent support from vendors in the Ethernet community. Therefore, this design leads to minimal implementation complexity, which significantly benefits our production-scale deployment.

5 Tier1: 1K GPUs in one Segment

This section presents how to interconnect 1024 GPUs in a tier1 network by employing the latest 51.2Tbps single-chip switch (§5.1) and rail-optimized network (§5.2).

5.1 Fully Utilizing 51.2Tbps Single-Chip

We employ the latest 51.2Tbps Ethernet single-chip switch (first released in early 2023) in HPN. In tier1, each switch possesses 128 active + 8 backup 200Gbps downstream ports and 60 upstream 400Gbps ports. This design ensures a near 1:1 oversubscription (actually 1.067:1). Each ToR switch reserves 8 backup downstream ports. We use these ports to connect backup hosts, enabling the



(a) Power consumption. (b) Cooling efficiency.

Figure 9: Power consumption of 51.2Tbps single-chip switch and the efficiency of different cooling solutions.

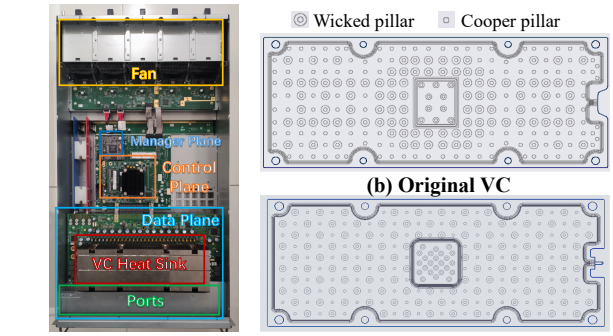


Figure 10: Customized 51.2Tbps single-chip switch. In (b) and (c), wicked pillar works for cooling and cooper pillar works for load-bearing.

quick replacement of hosts under host-side failures (including CPU, Memory, GPU, PCIe, NVLINK and NIC).

Why single-chip switch? The bandwidth capacity of the ToR switch directly determines the number of GPUs in the same tier1 network. There have been multi-chip chassis switches supporting higher bandwidth capacity [9, 27, 28, 32, 50]. However, Alibaba Cloud's long-term experience in operating data center networks reveals that multi-chip chassis switches introduce more stability risks than single-chip switches. Specifically, our operational single-chip switches outnumber multi-chip switches by 32.6×. On the contrary, the total number of critical hardware failures in multi-chip switches is 3.77× higher than in single-chip switches. The root cause is that the multi-chip switch is a distributed switching system, with multiple chips interconnecting through a chip fabric. Failures in the internal fabric, inter-chip interactions, and chip-to-CPU communication all contribute to the overall critical outages. We, therefore, decide to take single-chip switches for all newly designed network architectures.

Challenges introduced by high-throughput single-chip switch. A single chip supporting higher throughput means more traffic is handled per unit area, leading to increased power consumption in practice. As shown in Figure 9a, the power consumption of the 51.2Tbps switching chip has increased by 45% compared to the previous generation 25.6Tbps chip.[2] However, the chip's max junction temperature (T_{jmax}) remains unchanged (105°C). If the chip's working temperature exceeds T_{jmax}, it will immediately trigger the over-temperature protection, shutting down all data transmission.

[2]Data from chip vendor B, we omit the name due to confidentiality.

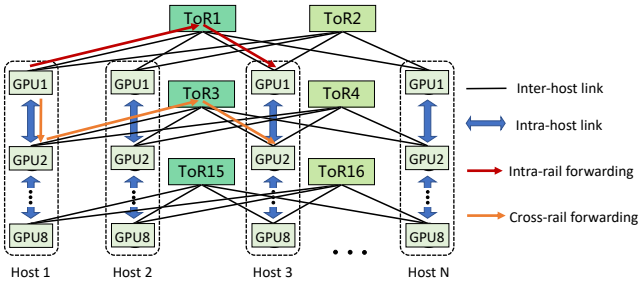


Figure 11: Rail-optimized network under dual-ToR.

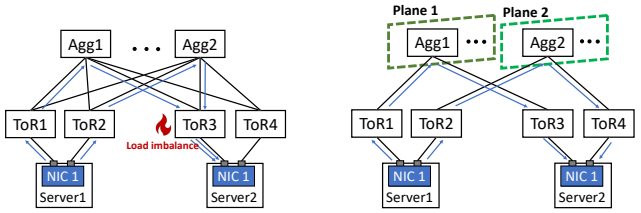
As shown in Figure 9b, none of the existing cooling solutions, including basic heat pipe solutions [34, 51] and the recommended vapor chamber (VC) heat sink [35, 36] provided by switch vendors, could support the 51.2Tbps single-chip work at full power (see Heat Pipe and Original VC in Figure 9b). In our experiments, we generated various high-pressure scenarios that could happen online, and the above solutions encountered shutdowns triggered by exceeding T_{jmax}. Leaving this cooling issue unsolved would lead to server outages in large-scale deployments.

Through comprehensive investigation, we find out the root cause: the heat generated at the center of the chip could not be effectively carried out. As shown in Figure 10, we design a new VC heat sink to resolve this issue. By optimizing the wick structure and deploying more wicked pillars at the center of the chip, heat could be carried out more efficiently. This design increases the cooling efficiency by 15% (see Optimized VC in Figure 9b). We deeply cooperate with our switch vendors to customize switches to equip this optimized VC, ensuring no overheating in all pressure cases.

5.2 Rail-Optimized Network

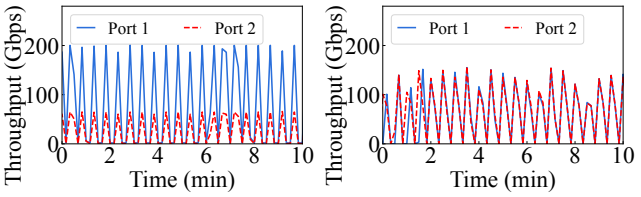
The de-facto configuration is that 8 GPUs inside the host are connected with a high bandwidth intra-host network (*e.g.*, NVLINK). Although the intra-host network bandwidth varies under different types of GPUs, it is 4-9× greater than the 2×200Gbps bandwidth provided by a NIC. To fully leverage the different forwarding capabilities, NVIDIA is the first to propose the concept of the rail-optimized network [21], which has been widely adopted in training clusters. In a rail-optimized network, NICs in the same rail are connected through the same set of dual-ToR switches. NICs in different rails can communicate via a combination of intra-host + inter-host forwarding. For example, in Figure 11, if GPU1 in host1 wants to talk to GPU2 in host 3, the forwarding path is GPU1 in host1 → GPU2 in host1 → ToR3 → GPU2 in host3.

Figure 11 shows how we employ the rail-optimized network in practice, which allows the 3.2Tbps (8×400Gbps) of a single host to be served across up to 16 ToR switches (under dual-ToR). The number of GPUs a segment can contain is increased by 8×, compared to the original topology, where 3.2Tbps for a host is served by 2 ToR switches. Each set of dual-ToR switches can serve 128 GPUs, and the 16 ToRs collectively connect 1024 GPUs in a segment, substantially reducing the forwarding latency and providing the utmost performance. More importantly, it significantly reduces traffic crossing the Aggregation layer, lowering the possibility of load imbalance in the network.



(a) Typical Clos. (b) Dual-plane.

Figure 12: Tier2 network architecture.



(a) Typical Clos. (b) Dual-plane.

Figure 13: Traffic on ToRs' ports towards the same NIC.

6 Tier2: 15K GPUs in one Pod

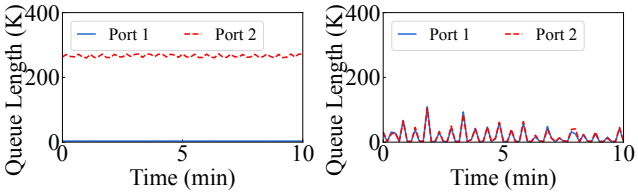
This section presents how HPN minimizes load imbalance (§6.1) and contains 15K GPUs (§6.2) in one Pod.

6.1 Conquering Load Imbalance

With dual-ToR in tier1 network, if we simply deploy a typical Clos topology between ToR and Aggregation, as shown in Figure 12a, hash polarization would still exist. In the downstream direction, owing to dual-ToR, there is a high convergence of traffic from 60 Aggregation switches to 2 ToR switches. Figure 13a shows the egress traffic of two downstream ports in the dual-ToR set towards the same NIC, measured during the real training job of a variant GPT-3 175B in production. The load of these two ports is significantly different (3×), degrading training performance as well.

Dual-plane: Eliminating hash polarization in a Pod. As shown in Figure 12b, in dual-plane, ToR switches in each dual-ToR set are categorized into two separate groups. With this design, once a flow enters one of the uplinks in the ToR, its forwarding path inside the Pod is completely determined. Therefore, hash polarization is eliminated in the Pod. After deploying the dual-plane design, as shown in Figure 13b, the ingress traffic of different ports becomes even, and the queue length at the ToR downstream ports decreases by 91.8%. Further ablation study reveals that the dual-plane design contributes up to 71.6% performance improvement for cross-segment traffic. We statistic the queue length of two downstream ports in the dual-ToR set towards the same NIC, measured during the real training job of a variant GPT-3 175B in production. As shown in Figure 14, when the tier2 network deploys typical Clos connecting, the imbalanced load would cause consistent congestion on the switch. The queue length is continuously kept at 267KB and 3KB, respectively. After deploying dual-plane, the load of two ToRs becomes even and the average queue length is 20KB.

Optimized path selection. Many efforts have been focused on solving the load imbalance caused by ECMP [54, 56, 57, 60, 61, 64, 66, 68, 70, 71]. However, many switch-based solutions [54, 57, 61, 66, 68, 70] require switches executing load-aware stateful packet forwarding, which is unavailable (*e.g.*, dynamic flow bin-packing or cross-switch negotiation) or unverified in large-scale deployment (*e.g.* flowlet splitting or per-packet spray). Most host-based solutions [56, 60, 64, 71] blindly select multiple paths and balance load on them according to congestion signals (*e.g.*, ECN, RTT, and INT). However, they may fall to suboptimal results if selected paths are not entirely disjoint. Furthermore, most host-based solutions require modifications in the transport layer, which is impractical in deploying commodity RDMA, whose transport layer is fully implemented in hardware. [3]



(a) Typical Clos. (b) Dual-plane.

Figure 14: Queue length at downstream ports of ToR.

To overcome shortcomings of existing solutions, HPN gets precise disjoint equal paths efficiently and balances the load on them in the collective communication library. First, for each new connection request, we generate a set of connections passing through disjoint paths. With large-scale deployed RePaC [72], the host can directly reprint the exact hash results in each switch. Based on the results, we find all disjoint paths and corresponding 5-tuples, and establish RDMA connections accordingly. We deploy a host-switch collaborating system to ensure all hosts maintain the latest link states and calculate the correct disjoint paths, which is out of the scope of this paper. Second, we implement a simple yet effective application layer load-balancing scheme to fully utilize all RDMA connections. Specifically, for each connection, HPN maintains a counter that records the total number of bytes in the current active Work Queue Elements (WQEs). The counter reveals the congestion status of the current connection: a congested connection drains the Work Queue slower. Therefore, our scheme (in the collective communication library) sends messages via the connection with the smallest counter (detailed pseudo-code is in Appendix B).

With a test of four AllReduce tasks concurrently running on 512 GPUs, this optimized path selection can improve the collective communication performance by up to 34.7%.

Thanks to the dual-plane design, when searching for disjoint paths, we only need to search the links in each ToR switch (*i.e.*, search at most 60 links), significantly decreasing the time consumption. Table 1 illustrates the calculation costs under different network architectures. HPN can greatly decrease the computation complexity by 1-2 magnitudes. More importantly, when failures happen, a host only needs to get the new ECMP group from ToR switches to recalculate disjoint paths (rather than maintaining ECMP groups from different tiers in a global controller).

[3] PLB [64] is a large-scale deployed load balance in Google, but it only works in TCP, rather than hardware-offloaded RDMA.

Table 1: Complexity of path selection

	Supported #GPUs	#Tiers	Switches participating load balance	Path selection complexity
Pod in HPN	15360	2	ToR	O(60)
SuperPod [21]†	16384	3	ToR+Aggregation+Core	O(32×32×4)=O(4096)
Jupiter [63]	26000 [12]	3	ToR+Aggregation	O(8×256)=O(2048)
Fat tree (k=48) [53]	27648	3	ToR+Aggregation	O(48×48)=O(2304)

† SuperPod is representing the network architecture of many in-production clusters (*e.g.*, NVIDIA DGX Cloud [47], Meta's AI supercomputer [6] and CoreWeave's cloud platform [10]).

Table 2: Key mechanisms affecting maximal scale

	Tier1 scale	Tier2 scale
51.2Tbps Clos	64	2K
Dual-ToR	128 (×2)	4K (×2)
Rail-optimzed	1K (×8)	-
Dual-plane	–	8K (×2)
Oversubscription of 15:1	–	15K (×1.875)

Table 3: Traffic patterns of different parallelisms

	DP	PP	TP
Traffic volume	5.5GB	6MB	560MB
Operations	AllReduce	Send/Recv	AllReduce/AllGather

In addition, there are 60 equivalent Agg switches in each plane. Therefore, even if one Aggregation switch fails, the other 59 Aggregation switches can still work to balance loads in the same plane.

6.2 15K GPUs in One Pod

The dual-plane design brings another important benefit: it halves the number of link connections between ToR and Aggregation, allowing Aggregation switches to support more segments in the same Pod. As a result, the scale of the tier2 network is doubled. In addition, we set the Aggregation-Core oversubscription to be 15:1, which releases 87.5% more ports on the Aggregation switches for interconnecting extra segments. Finally, we achieve the goal of containing 15K cards within the same Pod and we provide each GPU with 400Gbps network accessing capacity.

Table 2 summarizes key mechanisms in HPN that contribute to expanding the network scale. First, dual-ToR allows the single NIC's 2×200Gbps bandwidth to be served by two ToR switches, doubling the network scale. Coupled with the latest 51.2Tbps single-chip switches and rail-optimized network, a single segment can cover 1K GPUs. The dual-plane design at the Aggregation layer liberates half of the ports in each Aggregation switch and doubles the coverage of tier2. Finally, with the Aggregation-Core oversubscription of 15:1, HPN achieves the design goal of supporting 15K GPUs in one Pod. As shown in Figure 6, a Pod covers 100% of the training jobs we have served to date. Supporting 15K GPUs with a single Pod rather than multiple Pods further cuts unnecessary links and switches used for connecting multiple Pods, saving the overall network building cost by around 30% according to our statistics.

7 Supporting Larger Scale

To meet the long-term planning for supporting more GPUs (*e.g.*, O(100K) GPUs), we have also designed the tier3 network connecting multiple Pods. There is a trade-off between the Aggregation-Core oversubscription and the scale of the entire cluster. As aforementioned, we compromise the Aggregation-Core oversubscription (15:1) to increase the Pod scale. Deep diving into the communication pattern in the LLM training, we find that the single training job across tens of thousands of GPUs does not require excessive tier3 bandwidth capacity.

As shown in §2.1, the model training is distributed by multiple parallel strategies. Different parallel strategies introduce different volumes of data transmission. Take the training of GPT-3 175B with TP=8, PP=8, DP=512 as an example (requiring 32K GPUs). As shown in Table 3, PP generates the lowest traffic and utilizes the basic Send/Recv for communication, which is insensitive to network bandwidth. Thus, by proper cooperation with the worker scheduler, we can ensure that only PP traffic passes through Core layers, minimizing side effects introduced by multi-hop forwarding.

Minimizing load imbalance in tier3. Deploying tier3 may introduce additional load imbalance risks. To minimize this side effect, we make two enhancements. (1) We carry on the dual-plane design in the Core layer. (2) In each Core switch, we employ a prior per-port hash [69] to ensure traffic towards Pod i from physical port j would uniquely forward to port k (5-tuple irrelevant), eliminating hash polarization. If link failure happens at port k, traffic would fall back to execute the default 5-tuple-based hash. Potential small (affecting PP) performance degradation only occurs under failure cases, which is acceptable in production.

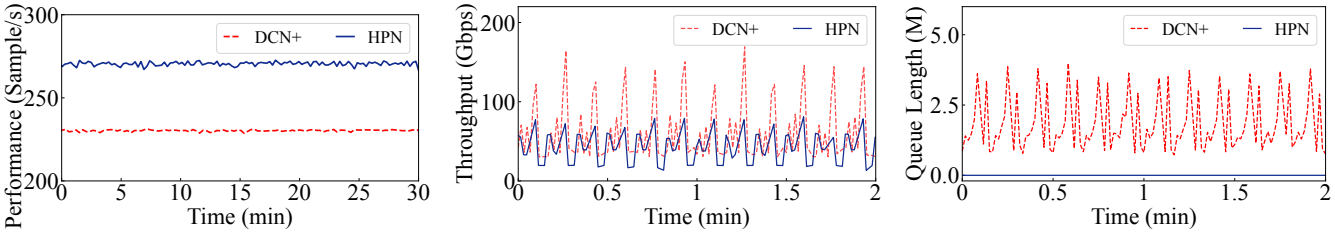
Till now, we have comprehensively presented the entire design of the HPN backend network.

8 Independent Frontend Network

In HPN, each host equips an additional 2×200Gbps NIC for frontend network access, and we design an independent frontend network. The frontend network primarily handles management and storage traffic (*e.g.*, cluster management, dataset loading, image loading and checkpoint saving/loading). It can also carry inference requests/responses while serving model inferences. Since the traffic supported by the frontend network is similar to traditional cloud computing scenarios, as shown in Figure 7, we adopted the classic 3-tier topology for the frontend network. To ensure reliability, each frontend NIC connects to two ToRs in the non-stacked dual-ToR way. In the frontend network, we design the convergence ratio to be 1:1 at both Aggregation and Core layers, guaranteeing maximal bisection bandwidth.

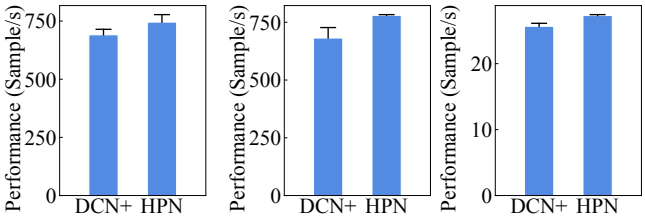
Isolating storage traffic from training. To provide optimal training performance, the storage cluster is settled in the frontend network (detailed discussion about the location of the storage cluster is shown in §10). A storage cluster consists of 96-128 storage hosts, running CPFS and OSS storage services to store training datasets, container images needed for training, and checkpoints saved during the training process.

Supporting inference compatibly. In the past few years, the mainstream GPUs for training and inference have been different.



(a) End-to-end training performance. **(b) Aggregation switch ingress traffic (averaged every 10s).** **(c) Queue length in Aggregation switches (max every 10s).**

Figure 15: Model training performance on 2300+ GPUs under different network architectures.



(a) LLaMa-7B. **(b) LLaMa-13B.** **(c) GPT-175B.**

Figure 16: Performance of training representative LLMs on different network architectures.

For example, NVIDIA V100, A100 and H100 are designed for training, and NVIDIA T4, A10 and H10 are mainly for inference. However, there is a trend to use training GPUs in inference. The reasons are twofold: (1) As the size of models increases, requiring GPUs with higher memory and performance for inference services, the specifications for GPUs used for inference are becoming increasingly similar to those used for training. (2) We observe that many customers prefer deploying both training and inference jobs on the same rented cluster for better GPU utilization.

When designing the HPN frontend network, we fully considered the above requirements, and the 2×200Gbps frontend network offers good performance. Therefore, such a design ensures that hosts in HPN can be flexibly used for both training and inference, building a unified platform supporting users' various demands.

9 Evaluation

This section presents the evaluations of HPN. HPN is deployed in multiple clusters connecting O(10K) GPUs in Alibaba Cloud, and serves thousands of model training jobs from dozens of customers. We compared HPN with our previous generation of training network architecture (DCN+). DCN+'s backend network is a traditional 3-tier Clos Data Center Network with full bisection bandwidth and dual-ToR enabled. In DCN+, each segment contains 128 GPUs, and each Pod contains 4 segments (detailed topology in Appendix §C). First, we show that HPN can improve the end-to-end large-scale training (taking up to 2300+ GPUs) performances by 14.9% compared with DCN+ (§9.1). Furthermore, we conduct network layer monitoring and different collective communication operations to show HPN's network layer performance improvement (§9.2). Last but not least, we quantify the reliability improvement (§9.3).

In evaluations, each host is equipped with 8 NVIDIA H800 GPUs [48] and 9 NVIDIA BlueField3 2×200Gbps DPUs [45]. GPUs in the same host are interconnected with 400GBps (bidirectional) NVLINK.

9.1 LLM Training Performance

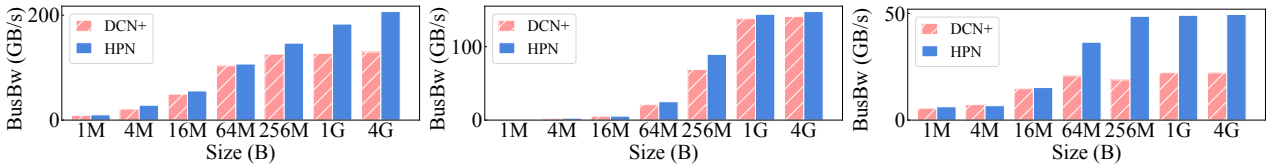
Large-scale training performance in production. One of Alibaba Cloud's proprietary LLMs was trained with 2300+ GPUs (288+ hosts) for several months. This model was first trained on DCN+, and then migrated to HPN. In DCN+, the training job spans 19 segments, while HPN fits the training job within 3 segments. We observed significant performance improvement after the migration. As shown in Figure 15a, the end-to-end training performance is improved by more than 14.9%. This end-to-end performance gain is actually a big value in production. Considering the construction of the entire training cluster may consume billions of dollars. 14.9% performance improvement contributes to significant cost savings. We further collect the statistics of Aggregation switches during training. Aggregation switches carry traffic across segments and their statistics directly reflect the network status. As shown in Figure 15b, the cross-segment traffic is decreased by 37% on average. Less cross-segment traffic triggers less congestion in the network. Figure 15c illustrates the queue length distribution of Aggregation switches' downlinks. The high traffic volume and hash collisions constantly build up queues in DCN+. While in HPN, this problem is greatly alleviated.

Representative LLM training performance. We further evaluate the training performance of three popular LLMs: LLaMa-7B, LLaMa-13B and GPT3-175B under 448 GPUs (56 hosts). As shown in Figure 16, by employing HPN, the end-to-end training performance is improved by 7.9%, 14.4% and 6.3%, respectively.

9.2 Network-Level Performance

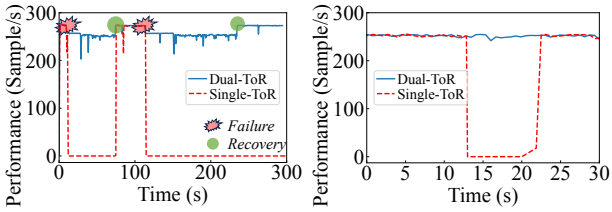
We evaluate the collective communication performance of HPN with NCCL 2.18.3 [46].

Collective communication results. We evaluate the performance of typical collective communication operations (AllReduce and AllGather) with 448 GPUs (56 hosts), where AllReduce is the predominant operation in LLM training jobs. Figure 17 lists the results. HPN increases the performance of AllReduce by up to 59.3%. Since a segment in HPN contains 1K GPUs, there is no contention between traffic. The performance of AllGather is similar between HPN and



(a) AllReduce. (b) AllGather. (c) Multi-AllReduce.

Figure 17: Collective communication performance.



(a) Link failure. (b) Link flapping.

Figure 18: Performance under different types of NIC-ToR link malfunctions.

DCN+. The main reason is that we employ NVLS in AllReduce. It can conduct aggregation in the NVSwitch, therefore providing higher intra-host throughput for AllReduce. NVLS, however, cannot accelerate AllGather. Therefore, the results of AllGather are all bounded by NVSwitch.

Multi-AllReduce is mainly used for gradient synchronization in the Megatron framework when setting TP=8. In Multi-AllReduce, GPUs with the same index in the same DP group conduct AllReduce in parallel. All data is exchanged through the inter-host network rather than the NVLink. HPN can increase Multi-AllReduce's performance by up to 158.2%, as shown in Figure 17c. The root cause is that HPN balances load better in the inter-host network and maximizes the network utilization.

9.3 Reliability

During our eight months of operation in production, no single-point failure is observed in HPN. In this subsection, we train LLaMa-7B with 256 GPUs (32 hosts), and inject link malfunctions (link failure and link flapping) on a NIC-ToR link. We compare dual-ToR with typical single-ToR design to validate the reliability improvement.

Case study1: Link failure and repair. As shown in Figure 18a, a link failure occurs at 10s, and the training halts immediately in single-ToR topology. If the failure can be located and repaired within 1 minute, the training can recover. However, if the repair takes more than two minutes, then training cannot recover. On the contrary, with dual-ToR, the failure of a single link only causes 6.25% performance degradation. While the failure is repaired, the training throughput returns to normal immediately.

Case study2: Link flapping. Figure 18b shows the impact of link flapping. In single-ToR, the temporary link flapping halts the training for more than nine seconds. In dual-ToR, the performance degradation is negligible.

10 Experience & Lessons

One Pod in a single data center building. All data center buildings in commission in Alibaba Cloud have an overall power constraint of 18MW, and an 18MW building can accommodate approximately 15K GPUs. In conjunction with HPN, each single building perfectly houses an entire Pod, making predominant links inside the same building. All optic fibers within the building are less than 100 meters, significantly simplifying the wiring complexity and allowing for the use of lower-cost multi-mode optical transceivers (cutting 70% cost compared with single-mode optical transceivers). Detailed data center layout can be seen in Appendix §D.

The forwarding capacity of a single Ethernet chip doubles every two years. While operating the current HPN, we are designing the next-generation network architecture equipping the higher-capacity single-chip switch. In the land construction planning of our next-generation data centers, the total power constraints for a single building have been adjusted to cover more GPUs. Thus, when the new data center is delivered, it can be directly equipped with 102.4Tbps single-chip switches and the next-generation HPN.

Asymmetric link states are possible. In the majority of cases, the link status observed from both sides of the link is identical. However, the deployments in practice still encounter exceptions. A case causing a large-scale performance degradation is due to the invalidation of Link Fault Signaling (LFS) negotiation. The optical signal quality in the NIC->ToR direction is abnormal, while the quality in the ToR->NIC direction is normal. The ToR switch detects this issue and attempts to notify the NIC through the standard LFS negotiation [38]. However, due to a bug in the NIC firmware, the notification has not responded correctly. The NIC still perceives the link as normal and sends data through the problematic link, ultimately leading to substantial packet loss. Owing to the low occurring possibility, this asymmetrical link states issue is difficult to discover in the testbed. It only emerges in large-scale deployments. Thanks to our dual-ToR design, this link fault leads to training performance degradation rather than the entire training job crashes.

HPN complicates wiring. HPN introduces extra designs (*i.e.*, rail-optimized and dual-plane designs) compared with DCN+, making wiring much more complex. Especially at the nascent stage of constructing HPN, on-site staff make a lot of wiring mistakes, leading to wired end-to-end network performances. To eradicate wiring mistakes before end-to-end testing, we employ INT-based probes to check that each hop (switchID and PortID) in paths precisely aligns with HPN's blueprint definition.

Why not employ the rail-optimized idea on tier2 to support larger scale? In HPN, we design a fully interconnected network on tier2. Meta [14] suggests that, during the training of LLMs, there is

Table 4: Any-to-any tier2 vs. Rail-only tier2

	Any-to-any tier2	Rail-only tier2
# Tier2 planes	2	16
# GPUs in a Pod	15360	122880
Communication limitations	None	Rail-only

completely no requirement for cross-rail network communication; thus, the network could be designed in a pure rail-only way.

If we take this assumption, most ToR-Aggregation links are unnecessary, which could lead to a great scale expansion of one Pod. As shown in Table 4, if HPN is modified to a tier2 rail-only design, a Pod could support over 120K GPUs.

However, rail-only tier2 heavily relies on models only generating intra-rail traffic. In current mainstream dense large models, all traffic patterns have been specifically optimized to satisfy this constraint. However, the evolution of new models would break this assumption. For example, training the increasingly popular MoE models [65] involves substantial all-to-all traffic towards different Experts, where the source and destination may inherently reside on different rails. Furthermore, in the serverless scenario, a host is shared by multiple tenants (*i.e.*, different tenants possess different NICs in a host). Relaying traffic across the intra-host network would be greatly limited, making the cross-rail network vital. Therefore, we decide to construct any-to-any tier2, and leverage tier3 to support a larger scale.

The location of the storage cluster. During the design of HPN, we conducted detailed analysis and discussions on the location of the storage cluster (in the frontend or backend network). From the perspective of a host, the backend network bandwidth is significantly higher than the frontend network (3.2Tbps versus 400Gbps), which is attractive for bandwidth-intensive storage services.

However, there are three main disadvantages to placing the storage cluster in the backend network: (1) Typically, the container images and dataset used by customers are usually stored in other data centers or customers' self-built clusters. This external data cannot be directly accessed through the backend network. Hence, this network design requires traffic to pass through a proxy between the frontend and backend networks, increasing software development complexity and associated stability risks. (2) As aforementioned, injecting storage traffic in the backend network would result in fluctuations in training performance. (3) Deploying a storage cluster in the backend consumes ToR ports, reducing the number of GPUs the backend network can support. Therefore, we finally chose to place the storage cluster in the frontend network.

Why not leverage rail-optimized topology for handling ToR-related failures? With rail-optimized topology, different NICs on the same host are connected to different ToR switches, naturally providing an opportunity to reroute traffic among different rails bypassing single-point failures mentioned in §2.3. With comprehensive assessments, we do not adopt this solution in production for two main reasons. First, proactively rerouting requires significant modifications on NCCL, which makes it hard to be employed by customers. A new relay module implemented on the critical data path is necessary for rerouting. Furthermore, manipulating the I/O direction of collective communication would introduce extra risks in production. For example, monitoring noise would cause misjudgment of the port status and trigger unnecessary traffic rerouting. As a result, the training performance would be greatly damaged (e.g., doubling communication time). We actually implemented a prototype of this solution, but it was abolished after our comprehensive internal evaluations.

11 Related Work

Topology for LLM training. Many companies have published their network architecture for LLM training. Google designs a 3D torus topology [62] to connect 4K TPUv4 with six uniform Ethernet ports [59]. This torus topology, however, is unsuitable for commodity GPUs, which are equipped with heterogeneous ports with vastly different speeds (e.g., PCIe and NVLINK). NVIDIA [21], Google [12], and Meta [19] disclose their 3-tier network architecture deployed in their GPU clusters, making hash polarization unavoidable. Meta also proposes a radical rail-only topology [14] achieving a larger scale in one layer network but sacrificing the performance of all cross-rail traffic. Furthermore, it relies on the large-scale intra-host network (*e.g.*, NVLINK Switch System connecting 256 GPUs), which is not yet a commodity as of now.

Dual-ToR solutions. The disadvantages of stacked dual-ToR solutions [8, 26, 29, 33, 37, 42, 44] are illustrated in §4 in detail. In 2022, Microsoft cooperates with Credo to implement the hardware-based active-standby dual-ToR [4, 5], and it evolves to active-active dual-ToR in 2023 [31]. However, it heavily relies on deploying FPGA-based SmartNIC and dedicated cables from the sole cable vendor [4], which is hard for large-scale deployment and prone to vendor lock-in. Besides equipping switches at the top-of-rack, another design is equipping switches at the middle-of-rack (MoR) [11], where each switch can serve hosts in multiple racks. Actually, with the employment of the dual-ToR solution and the rail-optimized topology, each ToR switch is worked in the MoR way.

Load balance. Tremendous research efforts have been devoted to improving load balance through schedule flows/packets to the optimal paths [54, 56, 57, 60, 61, 64, 66, 68–71]. The primary contribution of HPN is that it simplifies the search space for finding optimal paths. We also share the accurate path selection scheme deployed in our production.

12 Conclusion

This paper presents HPN, a new network architecture for training GPU clusters, which has been largely deployed in Alibaba Cloud for over eight months. HPN avoids single-point failure caused by single-ToR design in traditional data center topologies. HPN interconnects 15K GPUs with a 2-tier dual-plane network, eliminating hash polarization and simplifying the optimal path selections. HPN improves the end-to-end LLM training performance by 14.9%.

Acknowledgements

We acknowledge all teams within Alibaba Cloud that contributed to the success of HPN, including the High-Performance Network, Lingjun Production, Network Automation, Network Operation, Network Systems, Optical Network, and PAI teams, to name a few. We also thank our shepherd Ganesh Ananthanarayanan, and the SIGCOMM reviewers for their insightful comments. Ennan Zhai and Dennis Cai are the co-corresponding authors.

References

[1] 2004. Virtual Router Redundancy Protocol (VRRP) . https://datatracker.ietf.org/doc/html/rfc3768#page-19. (2004).

[2] 2010. IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 3: CSMA/CD Access Method and Physical Layer Specifications Amendment 5: Media Access Control Parameters, Physical Layers, and Management Parameters for Energy-Efficient Ethernet. https://standards.ieee.org/ieee/802.3az/4270/. (2010).

[3] 2019. In-Service Software Upgrade (ISSU). https://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst_standalones/b-in-service-software-upgrade-issu.html. (2019).

[4] 2021. Credo Announces HiWire SWITCH AEC – Enabling Simpler, Faster and More Reliable Dual TOR Connectivity. http://go.aussiebum.com/1KiYZf. (2021).

[5] 2021. Demystifying Dual TOR – Credo's SWITCH AEC and Upstream NOS Support. https://stordis.com/wp-content/uploads/documents/Credo-Stordis-Webinar-5-19-22.pdf. (2021).

[6] 2022. Meta Works with NVIDIA to Build Massive AI Research Supercomputer. https://blogs.nvidia.com/blog/meta-ai-supercomputer-dgx/. (2022).

[7] 2022. Troubleshoot The SwitchPort Packet Loss. https://img-en.fs.com/file/user_manual/switch-port-packet-loss-troubleshooting.pdf. (2022).

[8] 2022. Understanding Multichassis Link Aggregation Groups. https://www.juniper.net/documentation/us/en/software/junos/mc-lag/topics/concept/mc-lag-feature-summary-best-practices.html. (2022).

[9] 2023. Cisco Nexus 9800 Series Switches Data Sheet Data Sheet. https://www.cisco.com/c/en/us/products/collateral/switches/nexus-9000-series-switches/nexus9800-series-switches-ds.html. (2023).

[10] 2023. CoreWeave partners with Vast Data for AI cloud. http://go.aussiebum.com/xmGt7T. (2023).

[11] 2023. Data center cabling, EoR, MoR, or ToR? https://www.anfkomftth.com/data-center-cabling-eor-mor-or-tor/. (2023).

[12] 2023. Google unveils A3 supercomputer VMs capable of scaling to 26,000 Nvidia H100 GPUs. http://go.aussiebum.com/WrjpXtng. (2023).

[13] 2023. GPT-4 Technical Report. (2023). arXiv:cs.CL/2303.08774

[14] 2023. How to Build Low-cost Networks for Large Language Models (without Sacrificing Performance)? https://arxiv.org/abs/2307.12169. (2023).

[15] 2023. Introducing ChatGPT. https://openai.com/blog/chatgpt. (2023).

[16] 2023. Introducing Gemini: our largest and most capable AI model. https://blog.google/technology/ai/google-gemini-ai/sundar-note. (2023).

[17] 2023. LLaMA: Open and Efficient Foundation Language Models. (2023). arXiv:cs.CL/2302.13971

[18] 2023. 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. (2023).

[19] 2023. Meta's evolution of network for AI - presented by Meta. https://www.youtube.com/watch?v=5gOOtFySrqA. (2023).

[20] 2023. Microsoft Azure Eagle is a Paradigm Shifting Cloud Supercomputer. https://www.servethehome.com. (2023).

[21] 2023. NVIDIA DGX SuperPOD: Next Generation Scalable Infrastructure for AI Leadership. https://docs.nvidia.com/https:/docs.nvidia.com/dgx-superpod-reference-architecture-dgx-h100.pdf. (2023).

[22] 2023. NVIDIA Spectrum-X Network Platform Architecture. https://www.virtualgraffiti.com.au/datasheets/Nvidia/pdf7.pdf. (2023).

[23] 2023. Project Ceiba: AWS and Nvidia plan to build world's largest cloud AI supercomputer. http://go.aussiebum.com/kx21Ey4b. (2023).

[24] 2023. Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90% ChatGPT Quality. https://lmsys.org/blog/2023-03-30-vicuna/. (2023).

[25] 2023. What Is ISSU? https://info.support.huawei.com/info-finder/encyclopedia/en/ISSU.html. (2023).

[26] 2023. What Is Stacking? https://info.support.huawei.com/info-finder/encyclopedia/en/Stacking.html. (2023).

[27] 2024. 7800 Series. https://www.arista.com/en/qsg-7800-series/7800-series-overview. (2024).

[28] 2024. CloudEngine 16800 Series Data Center Switches. https://e.huawei.com/en/products/switches/data-center-switches/ce16800. (2024).

[29] 2024. Configuring Virtual Port Channels. https://www.cisco.com/c/en/us/td/docs/switches/datacenter/nexus5000/sw/layer2/503_n2_1/503_n2_1nw/Cisco_n5k_layer2_config_gd_rel_503_N2_1_chapter8.html. (2024).

[30] 2024. DeepSpeed. https://www.microsoft.com/en-us/research/project/deepspeed/. (2024).

[31] 2024. Dual ToR Evolution: Active-Active ToR Deep Dive. https://drive.google.com/file/d/1RtYYk1JHv7WnyABrUVj7FiRMZu2KEYtL/view?usp=drive_link. (2024).

[32] 2024. H3C S12500X-AF Data Center Cloud Core Series Switches. https://www.h3c.com/en/Products_and_Solutions/InterConnect/Switches/Products/Data_Center/Core/S12500/H3C_S12500/. (2024).

[33] 2024. H3C S5500-SI Series Ethernet Switches Operation Manual(V1.01) 36-Stack Management Configuration. https://www.h3c.com/en/d_200712/211816_294551_0.htm. (2024).

[34] 2024. Heat Pipe Technology. https://www.heatpipe.com/. (2024).

[35] 2024. Heat Pipes & Vapor Chambers – What's the Difference? https://celsiainc.com/heat-sink-blog/heat-pipes-vapor-chambers-difference/. (2024).

[36] 2024. HEATSINK TECHNOLOGY VAPOR CHAMBER HEATSINK. https://radianheatsinks.com/vapor-chamber-heatsink/. (2024).

[37] 2024. HPE E5500 - Stacking with HUAWEI or H3C Brand Switches. https://support.hpe.com/hpesc/public/docDisplay?docId=c03662589&docLocale=en_US. (2024).

[38] 2024. IEEE 802.3ad. https://standards.ieee.org/ieee/802.3ad/1088/. (2024).

[39] 2024. Link Aggregation Overview. https://www.cisco.com/assets/sol/sb/Switches_Emulators_v2_3_5_xx/help/250/index.html/page/tesla_250_olh/aggregating_ports.html. (2024).

[40] 2024. Linux Bonding Modes. https://thelinuxcluster.com/2010/01/08/linux-bonding-modes/. (2024).

[41] 2024. Megatron-LM. https://github.com/NVIDIA/Megatron-LM. (2024).

[42] 2024. MLAG vs. Stacking vs. LACP. https://community.fs.com/article/mlag-vs-stacking-vs-lacp.html. (2024).

[43] 2024. Model Checkpointing. https://deepspeed.readthedocs.io/en/latest/model-checkpointing.html. (2024).

[44] 2024. Multi-Chassis Link Aggregation - MLAG. https://docs.nvidia.com/networking-ethernet-software/cumulus-linux-55/Layer-2/Multi-Chassis-Link-Aggregation-MLAG/. (2024).

[45] 2024. NVIDIA BLUEFIELD-3 DPU PROGRAMMABLE DATA CENTER INFRASTRUCTURE ON-A-CHIP. https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/documents/datasheet-nvidia-bluefield-3-dpu.pdf. (2024).

[46] 2024. NVIDIA Collective Communications Library (NCCL). https://developer.nvidia.com/nccl. (2024).

[47] 2024. NVIDIA DGX Cloud. https://www.nvidia.com/en-us/data-center/dgx-cloud/. (2024).

[48] 2024. NVIDIA H800 PCIe 80 GB. https://www.techpowerup.com/gpu-specs/h800-pcie-80-gb.c4181. (2024).

[49] 2024. NVLink and NVSwitch. https://www.nvidia.com/en-us/data-center/nvlink/. (2024).

[50] 2024. RG-N18010-XH – Next Generation Data Center Network High-Density Centralized Modular Core Switch with 100GE/400GE Line Cards and Eight Service Slots. https://www.ruijienetworks.com/products/switches/data-center-switches/RG-N18010-XH/. (2024).

[51] 2024. The Basics of Heat Pipes – Their History, Principle, and Varieties explained. https://www.global.dnp/biz/column/detail/10162360_4117.html. (2024).

[52] 2024. Understanding In-Service Software Upgrade (ISSU) in ACX5000 Series Routers. https://www.juniper.net/documentation/us/en/software/junos/high-availability/topics/concept/issu-on-acx5000-overview.html. (2024).

[53] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. 2008. A scalable, commodity data center network architecture. In *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication (SIGCOMM '08)*. Association for Computing Machinery, New York, NY, USA, 63–74. https://doi.org/10.1145/1402958.1402967

[54] Mohammad Alizadeh, Tom Edsall, Sarang Dharmapurikar, Ramanan Vaidyanathan, Kevin Chu, Andy Fingerhut, Vinh The Lam, Francis Matus, Rong Pan, Navindra Yadav, and George Varghese. 2014. CONGA: distributed congestion-aware load balancing for datacenters. In *Proceedings of the 2014 ACM Conference on SIGCOMM (SIGCOMM '14)*. Association for Computing Machinery, New York, NY, USA, 503–514. https://doi.org/10.1145/2619239.2626316

[55] Wei Bai, Shanim Sainul Abdeen, Ankit Agrawal, Krishan Kumar Attre, Paramvir Bahl, Ameya Bhagat, Gowri Bhaskara, Tanya Brokhman, Lei Cao, Ahmad Cheema, Rebecca Chow, Jeff Cohen, Mahmoud Elhaddad, Vivek Ette, Igal Figlin, Daniel Firestone, Mathew George, Ilya German, Lakhmeet Ghai, Eric Green, Albert Greenberg, Manish Gupta, Randy Haagens, Matthew Hendel, Ridwan Howlader, Neetha John, Julia Johnstone, Tom Jolly, Greg Kramer, David Kruse, Ankit Kumar, Erica Lan, Ivan Lee, Avi Levy, Marina Lipshteyn, Xin Liu, Chen Liu, Guohan Lu, Yuemin Lu, Xiakun Lu, Vadim Makhervaks, Ulad Malashanka, David A. Maltz, Ilias Marinos, Rohan Mehta, Sharda Murthi, Anup Namdhari, Aaron Ogus, Jitendra Padhye, Madhav Pandya, Douglas Phillips, Adrian Power, Suraj Puri, Shachar Raindel, Jordan Rhee, Anthony Russo, Maneesh Sah, Ali Sheriff, Chris Sparacino, Ashutosh Srivastava, Weixiang Sun, Nick Swanson, Fuhou Tian, Lukasz Tomczyk, Vamsi Vadlamuri, Alec Wolman, Ying Xie, Joyce Yom, Lihua Yuan, Yanzhao Zhang, and Brian Zill. 2023. Empowering Azure Storage with RDMA. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. USENIX Association, Boston, MA, 49–67. https://www.usenix.org/conference/nsdi23/presentation/bai

[56] Advait Dixit, Pawan Prakash, Y. Charlie Hu, and Ramana Rao Kompella. 2013. On the impact of packet spraying in data center networks. In *2013 Proceedings IEEE INFOCOM*. 2130–2138. https://doi.org/10.1109/INFCOM.2013.6567015

[57] Soudeh Ghorbani, Zibin Yang, P. Brighten Godfrey, Yashar Ganjali, and Amin Firoozshahian. 2017. DRILL: Micro Load Balancing for Low-latency Data Center Networks. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '17)*. Association for Computing Machinery,

New York, NY, USA, 225–238. https://doi.org/10.1145/3098822.3098839

[58] Albert Greenberg, James R. Hamilton, Navendu Jain, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, David A. Maltz, Parveen Patel, and Sudipta Sengupta. 2009. VL2: a scalable and flexible data center network. In *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication (SIGCOMM '09)*. New York, NY, USA, 51–62. https://doi.org/10.1145/1592568.1592576

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

[60] Naga Katta, Mukesh Hira, Aditi Ghag, Changhoon Kim, Isaac Keslassy, and Jennifer Rexford. 2016. CLOVE: How I learned to stop worrying about the core and love the edge. In *Proceedings of the 15th ACM Workshop on Hot Topics in Networks (HotNets '16)*. Association for Computing Machinery, New York, NY, USA, 155–161. https://doi.org/10.1145/3005745.3005751

[61] Naga Katta, Mukesh Hira, Changhoon Kim, Anirudh Sivaraman, and Jennifer Rexford. 2016. HULA: Scalable Load Balancing Using Programmable Data Planes. In *Proceedings of the Symposium on SDN Research (SOSR '16)*. Association for Computing Machinery, New York, NY, USA, Article 10, 12 pages. https://doi.org/10.1145/2890955.2890968

[62] 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 (ACM SIGCOMM '23)*. Association for Computing Machinery, New York, NY, USA, 499–515. https://doi.org/10.1145/3603269.3604836

[63] Leon Poutievski, Omid Mashayekhi, Joon Ong, Arjun Singh, Mukarram Tariq, Rui Wang, Jianan Zhang, Virginia Beauregard, Patrick Conner, Steve Gribble, Rishi Kapoor, Stephen Kratzer, Nanfang Li, Hong Liu, Karthik Nagaraj, Jason Ornstein, Samir Sawhney, Ryohei Urata, Lorenzo Vicisano, Kevin Yasumura, Shidong Zhang, Junlan Zhou, and Amin Vahdat. 2022. Jupiter evolving: transforming google's datacenter network via optical circuit switches and software-defined networking. In *Proceedings of the ACM SIGCOMM 2022 Conference (SIGCOMM '22)*. Association for Computing Machinery, New York, NY, USA, 66–85. https://doi.org/10.1145/3544216.3544265

[64] Mubashir Adnan Qureshi, Yuchung Cheng, Qianwen Yin, Qiaobin Fu, Gautam Kumar, Masoud Moshref, Junhua Yan, Van Jacobson, David Wetherall, and Abdul Kabbani. 2022. PLB: congestion signals are simple and effective for network load balancing. In *Proceedings of the ACM SIGCOMM 2022 Conference (SIGCOMM '22)*. Association for Computing Machinery, New York, NY, USA, 207–218. https://doi.org/10.1145/3544216.3544226

[65] Samyam Rajbhandari, Conglong Li, Zhewei Yao, Minjia Zhang, Reza Yazdani Aminabadi, Ammar Ahmad Awan, Jeff Rasley, and Yuxiong He. 2022. DeepSpeed-MoE: Advancing Mixture-of-Experts Inference and Training to Power Next-Generation AI Scale. In *Proceedings of the 39th International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (Eds.), Vol. 162. PMLR, 18332–18346. https://proceedings.mlr.press/v162/rajbhandari22a.html

[66] Siddhartha Sen, David Shue, Sunghwan Ihm, and Michael J. Freedman. 2013. Scalable, optimal flow routing in datacenters via local link balancing. In *Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies (CoNEXT '13)*. Association for Computing Machinery, New York, NY, USA, 151–162. https://doi.org/10.1145/2535372.2535397

[67] 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. *CoRR* abs/1909.08053 (2019). arXiv:1909.08053 http://arxiv.org/abs/1909.08053

[68] Erico Vanini, Rong Pan, Mohammad Alizadeh, Parvin Taheri, and Tom Edsall. 2017. Let it flow: resilient asymmetric load balancing with flowlet switching. In *Proceedings of the 14th USENIX Conference on Networked Systems Design and Implementation (NSDI'17)*. USENIX Association, USA, 407–420.

[69] Yunhong Xu, Keqiang He, Rui Wang, Minlan Yu, Nick Duffield, Hassan Wassel, Shidong Zhang, Leon Poutievski, Junlan Zhou, and Amin Vahdat. 2022. Hashing Design in Modern Networks: Challenges and Mitigation Techniques. In *2022 USENIX Annual Technical Conference (USENIX ATC 22)*. USENIX Association, Carlsbad, CA, 805–818. https://www.usenix.org/conference/atc22/presentation/xu

[70] David Zats, Tathagata Das, Prashanth Mohan, Dhruba Borthakur, and Randy Katz. 2012. DeTail: reducing the flow completion time tail in datacenter networks. In *Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM '12)*. Association for Computing Machinery, New York, NY, USA, 139–150. https://doi.org/10.1145/2342356.2342390

[71] Hong Zhang, Junxue Zhang, Wei Bai, Kai Chen, and Mosharaf Chowdhury. 2017. Resilient Datacenter Load Balancing in the Wild. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '17)*. Association for Computing Machinery, New York, NY, USA, 253–266. https://doi.org/10.1145/3098822.3098841

[72] 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

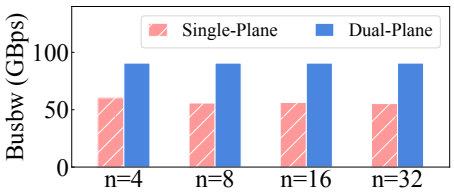


Figure 19: AllReduce performance of dual-plane.

Algorithm 1 EstablishConns

```
1: function EstablishConns(sip, dip, dport)
2:     sport_list ← findPaths(sip, dip, dport)
3:     conn_list ← []
4:     for sport in sport_list do
5:         conn ← Connect(sip, dip, sport, dport)
6:         Add(conn_list, conn)
7:     end for
8:     return conn_list
9: end function
```

Algorithm 2 PathSelection

```
1: function PathSelection(conn_list)
2:     conn ← getLeastLoad(conn_list)
3:     return conn
4: end function
```

APPENDIX

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

A Performance of Dual-plane

This part compares HPN's collective communication performance with and without dual-plane. The scales of experiments vary from 32 to 256 GPUs. We split the GPUs used in each experiment evenly into two different segments, so that all experiments generate cross-segment traffic. Results are shown in Figure 19. We test the performance of AllReduce under different scales and present the performance under a large message size (4GB). By employing dual-plane, the performance of AllRedcue increases by 50.1% to 63.7%.

B Path Selection Pseudo-code

Algorithm 1 illustrates the detailed procedure of establishing multiple disjoint paths. In line 2, findPaths() leverages RePaC [72] to calculate disjoint paths. As the source IP (*sip*), destination IP (*dip*) and destination port (*dport*) are determined, findPaths() returns a list of source ports (*sport_list*) for establishing connections passing through disjoint paths. In line 4-7, we generate connections accordingly and return the final *conn_list*.

Algorithm 1 illustrates the detailed procedure of the optimized path selection scheme. We maintain a counter (WQE_i) for each connection $conn_i$, representing the bytes of unfinished WQEs on the connection. WQE_i increases when a new WQE is injected to $conn_i$. WQE_i decreases when a new CQE is returned from c_i. In line 2, getLeastLoad() search paths in the *conn_list* to find out the $conn_i$ with the minimal WQE_i.

C DCN+ Topology

Figure 20 illustrates the DCN+ network architecture, which is Alibaba Cloud's previous version of network architecture deployed in training clusters. In the backend network, DCN+ deploys a typical three-layer Clos network with the enhancement of dual-ToR. With the deployment of dual-ToR, each segment is composed of 128 GPUs (16 servers). Each Pod is equipped with 8 Aggregation switches and each ToR switch connects to all Aggregation switches with a 400Gbps link. Therefore, each Pod contains 4 segments (*i.e.*, 512 GPUs). With DCN+, the entire cluster contains up to 32 Pods, making the scale up to 16384 GPUs. The frontend network in DCN+ remains the same compared with that in HPN.

D Data Center Layout

Figure 21 visualizes the layout of buildings in Alibaba Cloud's data center. In the backend network, a Pod (containing 15K GPUs) is completely accommodated by a backend building. Pods in different backend buildings are interconnected through a set of Core switches. There is an independent frontend building work for containing the entire frontend network and storage (CPFS/OSS) clusters. All hosts access the frontend network through the cross-building links. This data center layout leads to some benefits. (1) Cross-building links only occupy about 12.9% of the overall links, greatly decreasing the overall cost and simplifying the wiring and operating complexity. (2) GPUs in different backend Pods can share storage services at the data center scope, greatly decreasing the overall storage data volume.

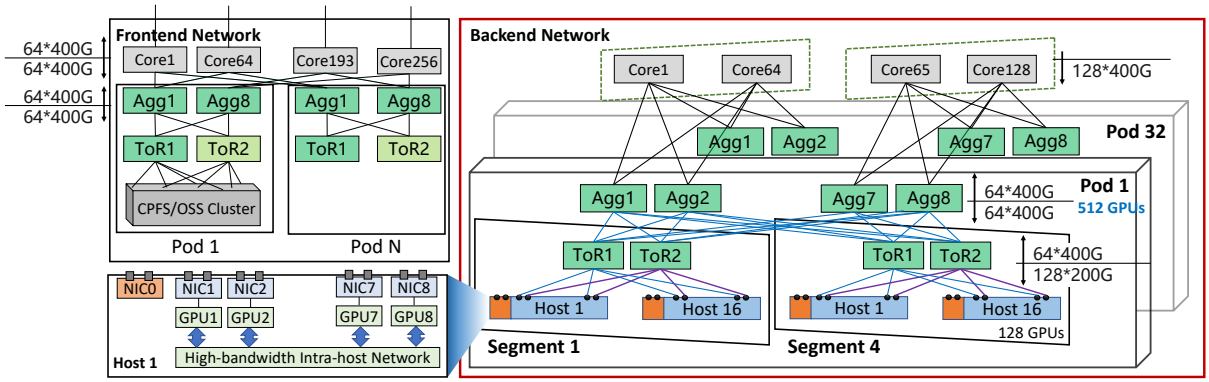


Figure 20: DCN+ topology. The frontend network is the same as that in HPN. In backend network, a segment contains 128 GPUs and a Pod contains 4 segments.

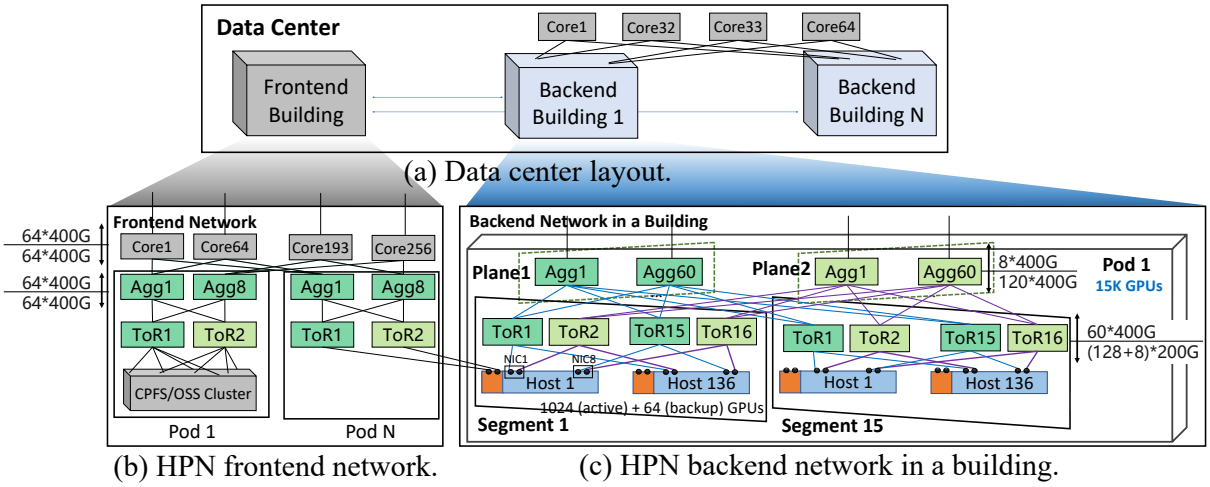


Figure 21: The layout of the entire data center that employs HPN. Each Pod in the backend is contained by an independent backend building. The frontend building contains the frontend network of the entire data center, as well as the storage cluster.