Balanced Sparse Tree: A Scalable Network Topology for Large Language Models

Shaoteng Liu*
liushaoteng@huawei.com
Huawei
Shenzhen, China

Dejun Kong*
kdjkdjkdj99@sjtu.edu.cn
Shanghai Jiao Tong Univ.
Univ. of New South Wales

Huitian Wang
wanghuitian@huawei.com
Huawei
Hangzhou, China

Hongji Dong
harlin671@sjtu.edu.cn
Shanghai Jiao Tong Univ.
Shanghai, China

Xiangyu Chen
chenxiangyu9@huawei.com
Huawei
Shenzhen, China

Yi Zhang
zhangyi418@huawei.com
Huawei
Nanjing, China

Xiaotian Zhou
zhouxiaotian7@huawei.com
Huawei
Shanghai, China

Peng Dong
dongpeng6@huawei.com
Huawei
Nanjing, China

Rui Meng
mengrui@huawei.com
Huawei
Beijing, China

Fuguang Huang
eric.huangfuguang@huawei.com
Huawei
Nanjing, China

Xia Zhu
zhuxia1@huawei.com
Huawei
Nanjing, China

Xiaofeng Gao✉
gao-xf@cs.sjtu.edu.cn
Shanghai Jiao Tong Univ.
Shanghai, China

Bingyang Liu
liubingyang@huawei.com
Huawei
Shenzhen, China

Guihai Chen
gchen@cs.sjtu.edu.cn
Shanghai Jiao Tong Univ.
Shanghai, China

Abstract

The development of large language models (LLMs) has catalyzed unprecedented demand on the computing network, specifically for large-scale, few-hops, and low-latency, which directly underpin LLM task efficiency. However, mainstream topologies such as Clos suffer from costs and latency, while topologies with good scalability have symmetric or collective communication issues. In order to achieve a favorable balance among design metrics, we propose a novel topology named the Balanced Sparse Tree (BST), which is a topology characterized by symmetric design and sparse connections, motivated by hypergraph theory and Steiner Systems. Its degree-diameter upper-bound approaches the Moore Bound for Bipartite Biregular graphs, larger than other known dia-meter-2 topologies. Furthermore, we incorporate differentiated routing, deadlock freedom, and topology-affined deployment into BST. Testbed experiments, simulations, together with modeling analysis, demonstrate the superiority of BST over the state-of-the-art in network scale, latency, bandwidth, and cost. With equivalent scales, BST outperforms Clos with a 50% cost reduction while maintaining comparable performance for AI workloads. Furthermore, BST delivers a 3.9%–11.8% gain in collective communications and has 13.4% improvement over state-of-the-art topologies.

*Both authors contributed equally to this research.

SIGCOMM '26, Denver, CO, USA

ACM ISBN 979-8-4007-2467-1/26/08
https://doi.org/10.1145/3789240.3829208

CCS Concepts

• **Networks** → **Data center networks**; **Topology analysis and generation**; • **Mathematics of computing** → **Hypergraphs**.

Keywords

Data center networks, Network topology, AI training and inference

ACM Reference Format:
Shaoteng Liu, Dejun Kong, Huitian Wang, Hongji Dong, Xiangyu Chen, Yi Zhang, Xiaotian Zhou, Peng Dong, Rui Meng, Fuguang Huang, Xia Zhu, Xiaofeng Gao, Bingyang Liu, and Guihai Chen. 2026. Balanced Sparse Tree: A Scalable Network Topology for Large Language Models. In *ACM SIGCOMM 2026 Conference (SIGCOMM '26), August 17–21, 2026, Denver, CO, USA.* ACM, New York, NY, USA, 20 pages. https://doi.org/10.1145/3789240.3829208

1 Introduction

The scaling law for training large language models (LLMs) has led to the rapid expansion of computing clusters, which now involve tens of thousands of GPUs/NPUs. For example, ByteDance has built clusters with 12,288 GPUs to train a model with 175B parameters [17]. Meanwhile, inference workloads are increasing, emerging as a major driver of cluster scaling. For instance, Deepseek V3 [23] was released with a maximum parameter scale of 671B and a sequence length of 128K, while Kimi K2.5 [2] has 1T parameters and 256K length[1]. Additionally, when combined with the recently proposed Attention-MoE [34], the number of computing clusters is expected to scale to $K \sim 10K$.

Computing clusters are connected via switches in the Data Center Network (DCN) with specific topologies. However, conventional network topologies have inherent limitations when scaling clusters.

[1]The Top-1 open-source LLM in llm-stats, released on Jan. 27th, 2026.

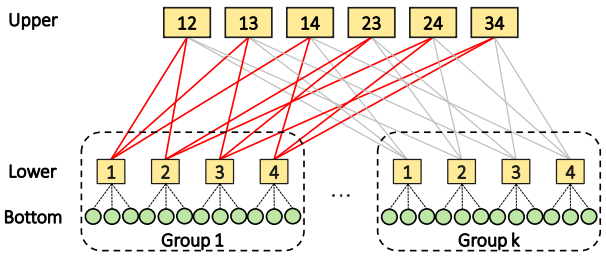


Figure 1: Balanced Sparse Tree (BST): Three Layers.

For instance, the widely deployed *Clos* topology [3, 32, 33] suffers from cost inefficiency and latency, since scaling typically requires adding more switches (and additional network layers) to maintain the topology, thereby increasing switching costs and hop count between pairwise clusters and resulting in greater network delay.

Unlike Clos, *Dragonfly* [19] replaces hierarchical switching with direct inter-switch connections. However, due to their asymmetric connectivity, they are incompatible with certain collective communication patterns, such as Pairwise All-to-All and HD All-Reduce, used in training and inference.

Moreover, in practical scenarios, scaling clusters is constrained by hardware limitations, including the number of switches and the switch radix (i.e., the number of ports), which is limited by semiconductor technology to a maximum of 256. A theoretical *degree-diameter upper-bound*, also named the *Moore Bound* [13], quantifies the maximum scale of cluster communication that a topology can support under certain limitations. Either Clos or Dragonfly+ remains considerably distant from this bound.

These limitations highlight the need for a new topology that better satisfies the requirements for low latency, collective communication, and approaches the theoretical upper bound. We refer to it as a *scalability requirement for LLMs*. Zettafly [8] is a recent scalable topology for LLMs that improves on Clos and Dragonfly, but it remains one-third to half of the theoretical upper bound.

To address the above challenges, we propose *Balanced Sparse Tree* (BST), a sparse and balanced network topology for LLMs, as shown in Fig. 1. BST has three key properties: ① **Balance**. BST is a biregular topology satisfying a Steiner System [16], so that the connection between pairwise clusters at each level is equivalent and symmetric, well-supporting the collective communication of LLMs. ② **Sparsity**. We define a sparsity factor to precisely utilize the switch radix, so that the network topology achieves horizontal expansion while maintaining a fixed 4-hop communication between pairwise clusters, thereby preserving low-latency communication and avoiding the need for additional network layers, such as Clos. ③ **Grouped Tree**. BST forms a switch-centric structure with three layers: bottom-level clusters (also referred to as nodes), lower-level switches (ToRs), and upper-level switches, with ToRs organized into equal-sized groups. Such a strategy appropriately addresses the scalability requirement.

In sum, the main contributions are shown as follows.

- We design the BST topology based on hypergraph theory with good characteristics on scalability, low latency and cost efficiency. The techniques of balance, sparsity and grouped tree design are introduced in the structure. BST topology enlarges the network scale by 1.7× compared with Zettafly topology, and achieves near-optimal performance for All-to-All/All-Reduce communications.
- We systematically design differentiated routing, deadlock avoidance, and topology-affined deployment for the BST, approaching the theoretical throughput limits of the BST.
- We build a testbed with Ascend servers and switches for real-world evaluation. Through VPN-based port isolation, the network topology can be virtualized and seamlessly switch between BST and Clos. The HCCL test tool is deployed in the prototype for collective communication tests.
- We conduct comprehensive experiments with simulation, testbed, and modeling. BST outperforms baseline methods including Clos and ZettaFly on theoretical analysis and AI workloads under collective communication scenarios. Fault tolerance and resilience are tested and discussed for the robustness of BST.

2 Background and Motivation

2.1 Distributed Training and Inference

As LLM processing data and model sizes continue to increase, model parallelism is required to enable distributed training and inference. As shown in Fig. 2, different model parallelisms are utilized, i.e., Sequence Parallel (SP), Tensor Parallel (TP), Expert Parallel (EP), Pipeline Parallel (PP), and Data Parallel (DP). Generally, TP, PP are used to offload model parameters to different NPUs so that the volume of parameters is within the memory capacity of each NPU. DP is used to distribute data batches across multiple NPUs to accelerate training and inference. For MoE model, EP is a general tool to spread out different experts among NPUs. The trend of large-scale EP propagates All-to-All communication across the entire network, potentially leading to network congestion. For TP or DP, communication is achieved using All-Reduce primitives. Therefore, communication in LLM training and inference is implemented via collective communication, dominated by All2All or AllReduce.

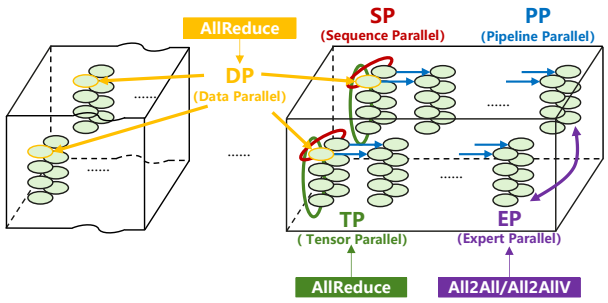


Figure 2: Illustration of different model parallelisms.

2.2 Motivation of Topology Design

The design of network topology for LLM training and inference is fundamentally constrained by hardware constraint, communication patterns, and scale requirement. Existing topologies usually have shortcoming on some aspects, leading to bottlenecks or inefficiencies when applied to LLM training and inference. This section outlines three key motivations that guide the design of BST.

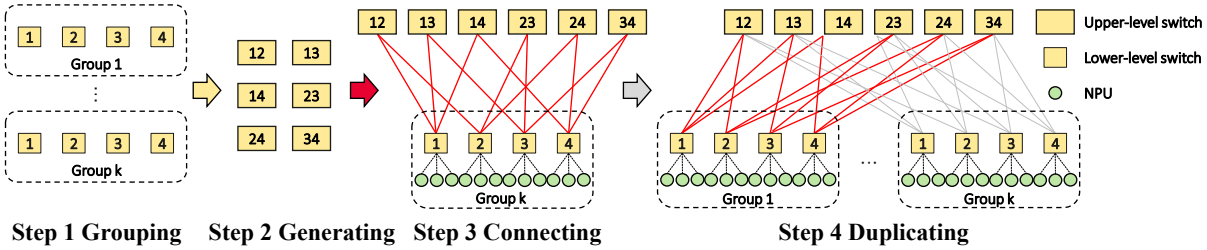


Figure 3: The process of constructing a BST: Grouping, generating, connecting and duplicating.

Efficient Collective Communication. Communication in LLM training and inference includes collective operations such as All2all and AllReduce, rather than arbitrary point-to-point traffic. Asymmetric design of the topology leads to unnecessary structural complexity and failing to fully utilize the characteristics of collective communication.

Moderate Structure Sparsity. Traditional multi-layer topologies (e.g., Clos) scale by increasing the number of layers, but this increases path length and cost, while flattened or expander-based designs still fall considerably short of the theoretical degree-diameter limit. These motivations necessitate a sparse topology that contributes to better scalability.

Topology Design with Hardware Constraints. Network topology is strictly constrained by hardware limitations. Besides, the topology must support practical system mechanisms, including routing, deadlock avoidance, and deployment strategies. Highly dense or irregular topologies complicate routing design and make system behavior difficult to analyze or control at scale.

3 Design of Balanced Sparse Tree

3.1 Overview and Construction

The basic structure of BST is presented as follows.

Definition 1 (Bipartite Graph)**.** A BST topology is a bipartite graph with two levels. Correspondingly, we define a graph $G = (U, V, E)$, where U denotes the set of nodes in the upper level and V denotes the set of nodes in the lower level. Let $|U| = m$, $|V| = n$. Edge set $E = \{(u_i, v_j)\} \subseteq U \times V$ connecting two sets as a bipartite graph.

Definition 2 (Biregularity)**.** Nodes at the upper or lower levels of a BST have the same degree. Say, $d(u_i) = d(u_{i'}), \forall i, i' \in [m]$. Similarly we have, $d(v_j) = d(v_{j'}), \forall j, j' \in [n]$. Here $d(\cdot)$ is the degree function. Define δ as the degree setting of the upper level, and ζ that of the lower level, respectively.

We consider a simple graph without self-loops and parallel edges. Thus $\delta \le n$, and $\zeta \le m$. Further, G is a complete bipartite graph if $\delta = n$ or $\zeta = m$.

Definition 3 (Sparsity Factor)**.** Define $\lambda = \frac{\delta}{n}$ as the sparsity factor of G, $\lambda \in [0, 1]$. $\lambda = 0$ denotes an edgeless graph, whereas $\lambda = 1$ denotes a complete bipartite graph.

To sum up, BST is a Sparse Bipartite Biregular graph, satisfying the biregularity with a sparsity factor $\lambda \in (0, 1)$.

With the basic structure of BST, we can construct a BST topology G bottom-up w.r.t. the lower level nodes V. We denote this topology as $G = \text{BST}(\omega, \delta, \kappa)$, which is constructed in the following steps, illustrated in Fig. 3.

- **Step 1. Grouping.** Split V into κ groups, each group with $\omega = \frac{n}{\kappa}$ nodes. n and κ are properly determined to satisfy the requirement in Sec. 3.2.
- **Step 2. Generating.** Generate upper-level nodes to form the upper-level node set U. Every two nodes among all the ω lower-level nodes correspond to only one upper-level node; each upper-level node connects δ lower-level nodes in a single group.
- **Step 3. Connecting.** Connect pairwise u_i, v_j by the corresponding relationship. Connect the NPUs to each v_j with the same bandwidth between u_i and v_j.
- **Step 4. Duplicating.** Construct the topology of the other groups with the same pattern towards the first group.

The definitions and notations used are shown in Table 1.

Table 1: Definitions and Notations

Symbols	Definitions
$G = (U, V, E)$	A Sparse Bipartite Biregular Graph.
$U = \{u_i\}_{i=1}^{m}$	Upper level set with m nodes.
$V = \{v_j\}_{j=1}^{n}$	Lower level set with n nodes.
$E = \{(u_i, v_j)\}$	Edges between U and V, $E \subseteq U \times V$.
$B_l(i, i')$	Bandwidth between $v_i, v_{i'}$ via two hops.
δ	Fixed degree setting of each u_i.
κ	Group number of lower-level nodes.
ω	Node number of one node group.
$\text{BST}(\omega, \delta, \kappa)$	BST construction rules w.r.t. δ, κ, ω.

3.2 Preliminary: Hypergraph Partitioning

As shown in Sec. 3.1, the construction process of a BST topology can be interpreted from the perspective of hypergraph partitioning. A hypergraph is a generalization of a graph in which edges (called hyperedges) can connect any number of vertices. In BST, the lower level node set V corresponds to the vertex set of the hypergraph, while the upper level node set U corresponds to the hyperedge set. Specifically, each upper-level node u_i and its connected δ lower-level nodes together constitute a hyperedge.

From this perspective, the core task of constructing a BST topology $G = \text{BST}(\omega, \delta, \kappa)$ is equivalent to designing a specific hypergraph: dividing $n = \kappa \times \omega$ vertices (lower level nodes) into κ groups, each group containing ω vertices, and designing a set of hyperedges

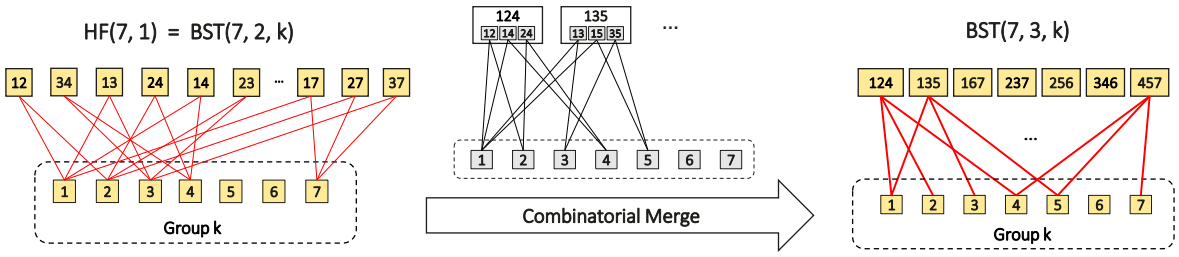


Figure 4: Combinatorial Merge: From $\text{HF}(7,1)$ to $\text{BST}(7,3,\kappa)$.

(upper level nodes) for each group, such that the connections within the group satisfy specific regularity and performance requirements.

To ensure optimal regularity and communication performance in this topology, an ideal construction goal is to guarantee that within the same group, any two lower-level nodes are contained within exactly one and only one hyperedge. An example is illustrated in Fig. 5. This property guarantees the communication path between any two points in the group is unique and deterministic, fundamentally avoiding hash collisions and providing an optimal symmetry basis for load balancing and efficient routing.

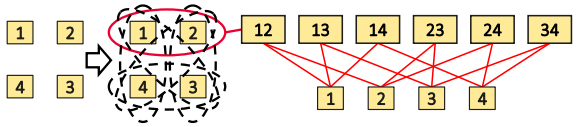


Figure 5: Hypergraph Partitioning: 6 Hyperedges.

When the degree of the hyperedge is set to $\delta = 3$, this construction goal perfectly aligns with the definition of the well-known Steiner System in combinatorial design theory. A Steiner System $S(2,3,\omega)$ is a set of triples (i.e., hyperedges with degree 3) on ω points, satisfying that any two points are contained within exactly one triple. The necessary and sufficient condition for the existence of a Steiner System is that the size ω of the group must satisfy $\omega \equiv 1 \text{ or } 3 \pmod 6$.

The derivation of this number-theoretic condition comes from the requirements of balance and regularity: the number of times each point appears, $(\omega-1)/2$, and the total number of triples, $\omega(\omega-1)/6$, must be integers. The parameters satisfying this condition (such as $\omega = 3, 7, 9, 13, 15, 19, 21, 25, \cdots$) provide a theoretically rigorous and infinitely expandable sequence for grouping schemes of BST. For example, when $\omega = 7$, the famous Fano Plane is obtained, which contains 7 points and 7 hyperedges, and is the minimal nontrivial structure satisfying this property.

BST's intuition conceptualized a full-mesh structure in hypergraph, where strict pairwise connectivity is maintained via uniform hyperedges. This also brings full-mesh like bisection bandwidth and collective symmetry.

3.3 BST Extension: Combinatorial Merge

To achieve a larger network size, even approaching the Moore bound for BST, we introduce combinatorial merge in this subsection. We first introduce Hiddenfly topology as follows.

Definition 4 (Hiddenfly Topology)**.** A Hiddenfly topology is also a Sparse Bipartite Biregular graph with two levels, which can be abstracted as a hypergraph. There are κ groups of nodes with ω nodes in each group. In each group, every pair of nodes is connected by exactly one hyperedge, excluding other nodes, to ensure full connectivity. Hence, there are $C_\omega^2 = \frac{\omega(\omega-1)}{2}$ hyperedges in each group. The Hiddenfly topology is denoted as $\text{HF}(\omega,\kappa)$.

With Def. 4, we define the combinatorial merge.

Definition 5. With $\text{HF}(\omega,\kappa)$ and an integer δ, the δ-merging (or δ-combinatorial merge) of $\text{HF}(\omega,\kappa)$ refers to merging the switches in the Spine layer of $\text{HF}(\omega,\kappa)$ so that each switch in the Spine layer connects to δ Tor switches within a group.

The combinatorial merge is explained as follows from the perspective of a hypergraph. Given a Hiddenfly hypergraph $G_{hf} = (V,E)$ and an integer δ satisfying $\delta > 2$, δ-merging refers to merging a set of binary hyperedges in the hyperedge set E into higher-order hyperedges with degree δ, in which each node can be connected to another via only one hyperedges. The existence problem of δ-merging is equivalent to the clique covering problem in hypergraph theory: determining whether a complete graph can be uniformly covered exactly by a δ-clique. This problem is NP-hard, reflecting the computational complexity of combinatorial merging.

If a δ-merge exists in a Hiddenfly hypergraph, the resulting structure is denoted as $\text{BST}(\omega,\delta,\kappa)$. The BST topology preserves the hypergraph's sparsity and symmetry and has diameter 2. To intuitively understand δ-merge, we illustrate it with an example as shown in Fig. 4. Through δ-merging, the binary hyperedges $\{1,2\}$, $\{1,4\}$, and $\{2,4\}$ are merged into a ternary hyperedge $\{1,2,4\}$, allowing vertices 1, 2, and 4 to exchange data packets freely. This merging significantly improves network efficiency by reducing hardware requirements. Simultaneously, the hypergraph maintains symmetry at each vertex and good collective communication characteristics, keeping load balancing and high throughput.

The following theorem (proved in Appendix A) shows the relationship between the network scale and δ-merging.

Theorem 1. Let R be the switch radix, r_{up} be the number of lower-level upward ports, and suppose there exists δ-merging for $n = r_{up}(\delta-1)+1$. The network scale is $V_{tor} = (r_{up}(\delta-1)+1) \times \lfloor R/\delta \rfloor$.

In a BST group, each ToR has r_{up} upward ports. Every two ToRs share exactly one common Spine switch. Each Spine connects to $\delta - 1$ other ToRs. Therefore, the total number of ToR switches reachable from one ToR switch is n, and the total number among

groups is V_{tor}. A natural question then arises: can we maximize the network scale? We propose the following problem.

Problem 1. Given r_{up} and R, we aim to find an integer δ, such that there exists a δ-merging and the network scale is maximized. The problem can be formally stated by:

$$\begin{aligned} &\text{Maximize: } (r_{up}(\delta-1)+1)\cdot\lfloor\tfrac{R}{\delta}\rfloor \\ &\text{subject to: } \delta\text{-merging exists} \end{aligned} \tag{1}$$

To solve this problem, we need to determine whether a complete graph with $V = r_{up}(\delta-1)+1$ nodes can be uniformly covered exactly once by a δ-clique. After obtaining the BST structure via the δ-merging operation, we next prove the non-blocking condition of the BST structure and analyze its bisection bandwidth. The detailed proof of Theorem 2 is provided in Appendix B. Additionally, Theorem 3 proved in Appendix C and Theorem 4 proved in Appendix D.

Theorem 2. Let r_{up} be the total upward bandwidth of each ToR, r_{down} be the total downward one. Collective communication can be non-blocking for any $\text{BST}(\omega,\delta,\kappa)$ when the speedup ratio $r_{up}/r_{down} = 1$. When the speedup ratio $r_{up}/r_{down} = 2$, the BST topology can be generally non-blocking.

When speedup = 1, BST's symmetric structure gives each node pair a unique minimal path with no link contention, enabling non-blocking collective communication. At speedup = 2, the doubled uplink capacity suffices to handle any permutation traffic, making the topology rearrangeably non-blocking.

Theorem 3. The maximum network size of a BST is naturally larger than a diameter-2 direct graph.

For a diameter-2 direct graph, the Moore bound is r_{up}^2+1. BST's two-tier structure yields up to $(r_{up}+1)R$ nodes, exceeding this bound with the same radix and speedup.

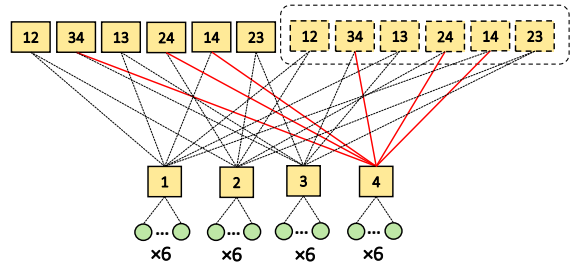
Theorem 4. The bisection bandwidth of each lower-level group in $\text{BST}(\omega,\delta,\kappa)$ is $B_{\text{bisect}} = \lfloor\omega^2/4\rfloor B_0$, where B_0 denotes the per-pair logical bandwidth.

Intuitively, the hypergraph structure makes each group behave as a logical full mesh: every pair of lower-level nodes is connected through one upper-level hyperedge. This design gives BST an asymptotic bisection factor of 1/4, which theoretically accounts for its efficient communication in LLM.

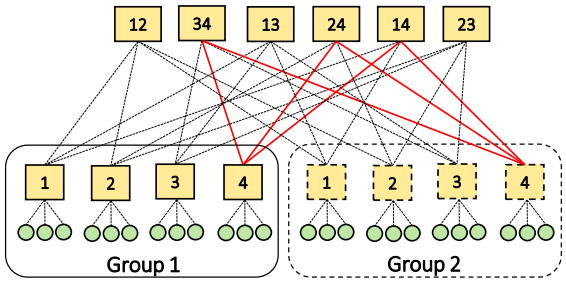
3.4 BST Expansion: Scalable BST

To improve scalability, we develop a BST topology comprising two switching layers. The upper layer is deployed as the *Spine layer*, while the lower layer serves as the *Leaf layer*. Fig. 6 exhibits a scalable BST achieving lossless collective communication using two non-blocking network-scaling methods, as depicted in Fig. 6(a) and Fig. 6(b), respectively.

Spine fabric scaling. To fully utilize the ports of ToR switches, we proportionally increase the number of Spine switches and the number of upward ports on ToR switches, increasing the total network bandwidth. The newly added Spine switches are interconnected to the ToR switches using the same BST topology. Hence, the number of equal paths between different groups is increased. Collective communication and P2P performance remain unchanged.



(a) Spine fabric scaling: Duplicate at Spine layer.



(b) ToR fabric scaling: Duplicate at ToR layer.

Figure 6: Topological scaling of Scalable BST.

ToR fabric scaling. Contrary to Spine fabric scaling, ToR fabric scaling is designed to saturate the ports of Spine switches to increase network scale. Then, we proportionally increase the number of downward ports on Spine switches and the number of ToR switches. The newly added ToR switches are interconnected to the Spine switches with the same BST. ToR fabric scaling has following merits:

- Switches labeled with the identical number in different groups are interconnected via a Clos fabric.
- The Scalable BST topology in each group is equivalent.
- Both collective communication performance and P2P performance are fully maintained.

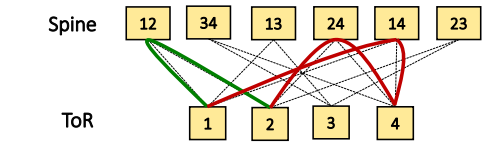
A concern for BST is its Steiner-based construction leading to significant port stranding on standard radices (64, 128, 256). A feasible BST parameter table is shown in Appendix M.

- Across three radices, a notable fraction of configurations achieve no waste: 31% (11/36) for radix-64, 17% (14/82) for radix-128, and 10% (17/173) for radix-256. For designs with non-zero waste, the total port waste rate is tightly bounded below 5% in every case.
- The worst-case waste is only 3.1% (radix-64), decreasing to 2.3% (radix-128) and 1.2% (radix-256). This 1–3% waste on the Spine is well within industry norms, and idle ports can serve cable management or expansion.
- The connectivity between ToR and spine switches remains governed by the same deterministic incidence matrix regardless of dark ports, ensuring a reproducible cabling plan with no irregular wiring. Thus, BST incurs negligible port stranding and is readily deployable on real hardware.

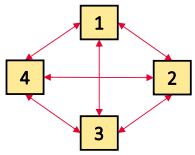
Besides, Extension on logical topologies combining BST with Clos is presented in Appendix G. An example of BST with $\delta > 3$ is presented and discussed in Appendix H.

4 Routing Scheme

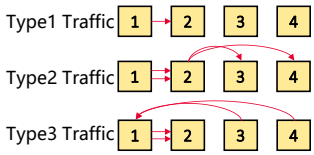
Unlike densely connected Clos networks, which primarily use minimal paths, sparsely connected topologies benefit from using both minimal and detour paths to maximize bandwidth utilization in Fig. 7(a). For example, in Fig. 7(b), if the three endpoints under node 1 communicate with those under node 2, the bandwidth of the minimal path converges, resulting in congestion. In such cases, detour routing, such as the traffic via node 3, can resolve this bottleneck.



(a) Two BST Routings: Minimal and Detour Paths.



(b) Minimal-paths.



(c) Paths of Different Traffic Types.

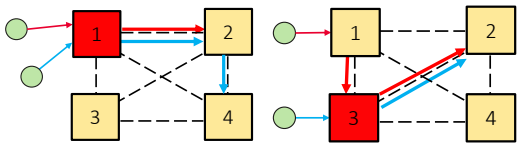
Figure 7: (a) Toy examples of minimal (green) and detour (red) routing in BST. (b) All2all with minimal routing. The throughput between each node is the link bandwidth BW. (c) All2all with detour routing, showing three traffic types: one-to-one/one-to-many/many-to-one. Each link supports 3 flow types with 5 flows; hence, the bandwidth per flow is $BW/5$, and the total bandwidth between each node is $3/5BW$.

On the other hand, detour paths introduce additional hops, hence consuming more aggregate network bandwidth (i.e., utilizing link bandwidth across more nodes). For simplicity, we use a 4-node full mesh to abstract the BST in Fig. 7(b), omitting the Spine-layer nodes to demonstrate the routing methods. While minimal paths yield a throughput of BW per pair for All2All communication (Fig. 7(b)), incorporating detour paths reduces it to $3/5BW$ (Fig. 7(c)). This highlights a key trade-off: minimal paths are more preferable for dense traffic (e.g., All2All) to ensure efficiency, whereas sparse traffic (e.g., P2P) benefits from detour paths to increase communication bandwidth between nodes.

To address this, we propose Differentiated Routing (DiffR) for BST-like sparsely connected networks. DiffR utilizes both minimal and detour paths, and adaptively inhibits detour path traffic in response to real-time traffic and network conditions.[2] Additionally, we present a deadlock avoidance mechanism for detour in BST.

4.1 Differentiated Routing (DiffR)

Let $T_{\max}$ denote the Priority Flow Control (PFC) threshold or the port buffer size, and $C_{\max}$ denote the maximum credit count in credit-based flow control. DiffR introduces a new threshold for detour path routing. We define $t_q < T_{\max}$ and $t_c < C_{\max}$ as the corresponding detour path routing thresholds. Every two thresholds divide the traffic into three stages. When the current port queue depth is less than t_q or the available credit count exceeds t_c, the port is considered lightly loaded and eligible for both minimal and detour routing; otherwise, only minimal routing is permitted. In the following sections, we use queue depth as a representative metric to elucidate the two main mechanisms of DiffR. Experiments towards t_q are conducted in Section 7.4.



(a) Local congestion aware. (b) Remote congestion aware.

Figure 8: DiffR monitors two types of congestion.

Local Congestion-aware Routing. Each node monitors the queue depths of its local ports. When forwarding packets, a node adaptively selects the ports of minimal paths and lightly loaded ports (i.e., queue depths $< t_q$) on detour paths. Taking Fig. 8(a) as an example, there are three paths from node 1 to 2: the minimal path $1 \rightarrow 2$, and the detour paths $1 \rightarrow 3 \rightarrow 2$ and $1 \rightarrow 4 \rightarrow 2$. Similarly, there are three routing paths from node 1 to 4: the minimal path $1 \rightarrow 4$, and the detour paths $1 \rightarrow 2 \rightarrow 4$ and $1 \rightarrow 3 \rightarrow 4$. Initially, if node 1 communicates only with node 2, all three paths are utilized. However, as 1 initiates traffic to 4, flows $1 \rightarrow 4 \rightarrow 2$ contend with the minimal-path flow $1 \rightarrow 4$ for the 1-4 link, causing the queue depth on this link to exceed t_q. Therefore, the detour path $1 \rightarrow 4 \rightarrow 2$ is inhibited from prioritizing the minimal path $1 \rightarrow 4$.

Remote Congestion-aware Routing. Detour path traffic can also encounter congestion on remote links. As shown in Fig. 8(b), there are three paths from node 3 to 2: the minimal path $3 \rightarrow 2$, and the detour paths $3 \rightarrow 1 \rightarrow 2$ and $3 \rightarrow 4 \rightarrow 2$. Flows $3 \rightarrow 2$ and $1 \rightarrow 3 \rightarrow 2$ contend for the link from 3 to 2. To mitigate remote congestion, when the queue depth of link 3-2 exceeds t_q, node 3 sends a notification signal to its upstream nodes (e.g., node 1), specifying the destination prefixes affected by this congestion. Upon receiving this notification, node 1 stops forwarding 1-to-2 traffic to node 3, thereby avoiding the congested path.

In practice, DiffR can use either flow-level ECMP or packet-level spraying. We recommand packet-level spraying, since it distributes packets across eligible paths and provides finer-grained load balancing, which better matches DiffR's goal of utilizing lightly loaded detour paths. In our packet-spray experiments, the maximum observed reordering depth is 72 packets, below the 128-packet reorder tolerance of the chip.

4.2 Credit Loop Deadlock Avoidance

Building on the above analysis of the BST structure, we further derive two corollaries regarding deadlock. More detailed proof of Corollary 1 and Corollary 2 can be found in Appendix E and Appendix F, respectively.

[2]Given the dynamism of AI traffic, such as that induced by expert parallelism in mixture-of-experts models, performing optimized static routing is challenging, and is beyond the scope of this work.

Corollary 1. To form a deadlock cycle in a BST topology, at least three detour flows are required.

Corollary 2. Minimal-path flows neither form cycles by themselves nor form cycles with detour flows.

Based on the above observations, we propose a simple approach: use two Virtual Lanes/Channels per port to avoid deadlock. When a detour flow enters ToR from an upward spine and then exits toward another spine, it transitions to VL2 and continues to use VL2 for the remainder of its path. In other situations, flows always use VL1.

5 Collective Communication

All2All. To fully utilize the upgoing paths, we use a one-shot (single-cycle) All2All/All2Allv algorithm, in which each sender NPU simultaneously sends messages to all other receiver NPUs. In this traffic pattern, all the traffic is distributed in the BST topology in a balanced manner, exhibiting non-blocking communication performance (Fig. 9).

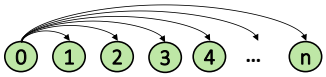


Figure 9: One-shot All2All.

Note that the pairwise algorithm, a multi-step algorithm in which each sender NPU communicates with only one receiver NPU in each step, is widely used for All2All communication in Clos topologies. When a pair-wise algorithm is employed in a BST topology, *topology-affined communication scheduling* should be considered to maintain the load balancing in the network. Therefore, a one-shot algorithm is preferred when AlltoAll is run in a BST topology.

All. It corresponds to a two-round application of All2All. In the first round, each node scatters $\frac{1}{n}$ of its local data to the other $n-1$ nodes. In the second round, each node broadcasts the aggregated $\frac{1}{n}$ dataset to other $n-1$ nodes.

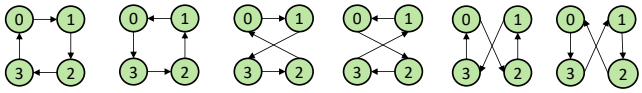


Figure 10: Ring with 4 nodes.

In the context of specific collective communication algorithms such as All, it may result in inter-group congestion without proper rank scheduling. We orchestrate the Ranktable to distribute traffic evenly across all links and avoid flow congestion. In Fig.10, we leverage Ranktable to generate 6 distinct ring schemes with 4 nodes for AllReduce, ensuring congestion-free communication.

For real-world deployment, suppose a network consists of 5 Pods, each with 8 nodes, as shown in Fig. 11. The nodes with identical indices across all pods form eight distinct AllReduce communication domains. Among the total 8 communication domains, 6 communication domains can be arranged in the 6 ring configurations as shown in Fig. 10, while the remaining two communication domains utilize the bandwidth of the 5th pod to complete the routing.

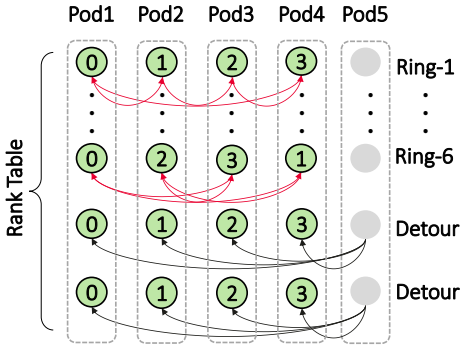


Figure 11: AllReduce with 5 Pods and each has 8 NPUs.

6 Modeling Analysis

In this section, we present the core analytical innovations of our model, demonstrating its advantages in large-scale deployment and high resource efficiency. The following baselines are selected for the model analysis and subsequent simulated evaluation experiments.

Clos[3]: A classic multi-stage, non-blocking network topo-logy which employs multiple stages of switches to provide high bisection bandwidth and fault tolerance.

HyperX[1]: A topology based on a hypergraph structure, forming a generalized hypercube. It provides flexible dimensionality and high path diversity, allowing it to be tailored for specific workload and cost constraints.

Zettafly[8]: A high-radix, low-diameter network topology designed for extreme-scale systems. It optimizes for both performance and cost by leveraging high-port switches.

Dragonfly+[26]: An enhanced version of the Dragonfly topology with a Clos-like intra-group fabric and spine switches carrying global inter-group links.

To quantitatively compare the scalability and cost efficiency of different network topologies, we use several widely used metrics: network scale, cost per NPU, and hop count. The cost per NPU includes optical modules and switches. To ensure a fair comparison, we further analyze the network scale of these topologies at the same radix. Based on the experimental results in Table 2, BST has achieved significant scale expansion while maintaining cost efficiency per NPU and per network hop, outperforming mainstream solutions. While it outperforms HyperX in both dimensions, BST achieves a comparable scale to the 3-layer Clos topology but with an approximate 50% reduction in cost. When measured against Zettafly, BST scales 1.7× further while maintaining a similar cost profile. The results indicate BST offers comprehensive advantages in switch radix count, optical device costs, and network latency.

In addition, we analyze the resource utilization efficiency of Clos and BST using a task-scheduling simulation under varying task-size distributions. We demonstrate that BST achieves a higher average NPU utilization rate (6.74% better than Clos). Under the same radix constraint, network layers, and total NPUs, BST provides larger per-cluster scheduling domains than Clos thereby reducing resource fragmentation when placing tasks of different sizes. Furthermore, this efficiency advantage of BST becomes more pronounced as the

Table 2: Topology Cost and Scale Comparison.

Topology	Hop	Cost (/NPU)		NPU by Switch Radix			Scalability
		Optical Modules	Switches	32	64	128	
2-layer Clos	3	2	$3/R$	512	2K	8K	$\frac{1}{2}R^2$
3-layer Clos	5	4	$5/R$	8K	64K	512K	$\frac{1}{4}R^3$
HyperX	3	~ 4	$5/(R+2)$	1.2K	9K	68K	$\frac{4}{125}(R+2)^3$
DF+	4	3	$4/R$	64K	1M	16M	$\frac{1}{4}R^2(\frac{1}{4}R^2+1)$
Zettafly	3	2	$3/R$	4.2K	33K	260K	$\frac{1}{4}R^2(\frac{1}{2}R+1)$
BST	3	2	$3/R$	6.5K	48.5K	449K	$\frac{1}{4}R^3(1-\frac{1}{\delta})+\frac{1}{2}\frac{R^2}{\delta}$

Note. All computations are performed under the non-blocking assumption. To consistent with BST, the hop count for HyperX and Zettafly is set to 3.

cluster scale increases. Detailed modeling setup and results are shown in Appendix I.

7 Simulation Evaluation

In this section, we present a comprehensive simulation-based evaluation of the proposed topology. Our findings demonstrate that BST achieves superior scalability while delivering comparable or superior performance. Our experiments are conducted on *Netisim*, an independently developed simulation platform that supports kilo-scale topology construction and load-balancing algorithms, including Packet-spray and ECMP. Beyond minimal path routing, *Netisim* incorporates our differentiated routing for BST. We evaluate BST topologies at different scales: a 320-node configuration using 40 servers and a 2.5k-node configuration using 320 servers. Each server comprises 8 NPUs interconnected via a 1D-FullMesh architecture, providing 800 Gbps of bidirectional bandwidth between any pair. Each NPU connects to a switch via sixteen 400G ports, and all switches have a radix of 128. All topology designs comply with the aforementioned constraints. The simulation results are also validated using open-source platform *Open-usim* [15].

7.1 Single-Task Collective Communication

To evaluate the performance under collective communication workloads, we conduct All2All, AllReduce at scales of 320 nodes and All2allv and AllReduce at scales of 2560 nodes, using the collective communication schemes in Sec. 5. We analyze how the structural characteristics of BST determine the flow completion time (FCT) across message sizes and communication patterns.

For the 320-node All2All experiments, the results presented in Fig. 12(b) indicate that the BST topology demonstrates strong convergence with Clos across both latency sensitive and bandwidth-sensitive regions. Within the latency sensitive range ($10KB$ to $800KB$) in Fig. 12(a), topologies with lower forwarding hop counts achieve superior performance, improving by approximately 10.7%. While Zettafly and HyperX remain close to Clos, BST maintains optimal latency due to its efficient routing. As message size enters the bandwidth-sensitive range ($800KB$ to $200MB$), the performance of Zettafly and HyperX begins to show signs of diminished convergence. In contrast, BST maintains high throughput even for large message sizes, outperforming by at least 3.9%. This is because BST's ToR-layer architecture naturally separates upstream traffic before it enters the Spine layer, providing physical-layer load balancing.

Regarding the AllReduce performance at the 320-node scale, we evaluate the Ring and Mesh algorithms as illustrated in Fig. 12(c). BST outperforms the baseline by approximately 11.8%. BST and Zettafly feature a ToR layer domain of up to 64 NPUs, enabling a significant portion of AllReduce traffic to be handled by the ToR switches, thereby meeting communication requirements efficiently. While Clos is bottlenecked because a vast majority of traffic must traverse the Spine layer. For Dragonfly+ and HyperX, the smaller size of their respective lossless domains imposes physical constraints that degrade AllReduce performance.

To verify scalability, we extend the evaluation to a 2560-node BST deployment. Fig. 12(d) presents performance evaluation with uniform All2Allv communication. The lighter the traffic load, the greater the latency advantage of BST over other topologies. As the traffic increases, the performance curves of BST and Clos nearly converge, both eventually saturating the available bandwidth. In Fig. 12(e), the RHD algorithm is adopted for Allreduce in large-scale scenarios to enhance communication efficiency. The overall trend is similar to the results with 320 nodes. For small message sizes, BST has a lower latency advantage; for large message sizes, the advantage depends on whether the topology can fully utilize the bandwidth. In these experiments, BST outperforms Zettafly by 8.2%, with Clos ranking third.

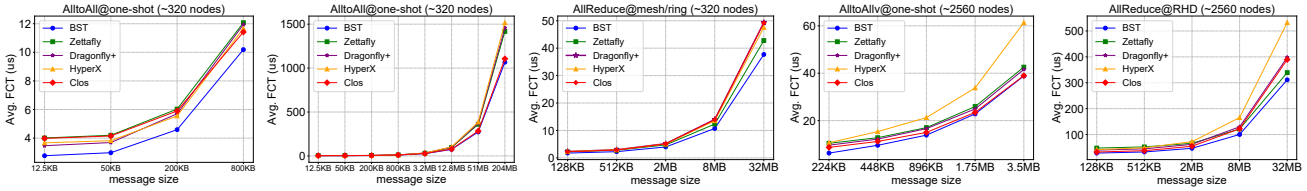
In addition to collective communication, we also evaluate the end-to-end latency of typical traffic patterns with uniform random and adversarial traffic. BST achieves the lowest latency under uniform random load, and ranks second only to Clos under adversarial traffic in latency stability. Detailed setup and results are in Appendix J.

7.2 Real-World Workload Simulation

To further validate the efficacy of BST in real-world AI environments, we evaluate its performance on end-to-end workload traces generated by the DeepSeek-V3 framework.

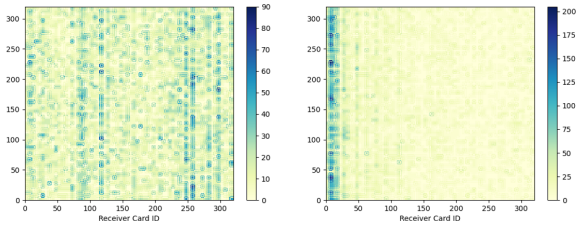
Workload Profiling and Trace Synthesis. We collected fine-grained profiling data from a 64-NPU cluster running DeepSeek-V3, which employs an Mixture-of-Experts (MoE) architecture with 256 experts and a Top-8 routing policy. The profiling captures the access frequencies of all 256 experts across different Transformer layers with uniform All2Allv communication. As shown in Fig. 13, there are significant variations in load distribution across layers. To capture this diversity, we selected two representative layers for our simulation: a **balanced layer** (exhibiting balanced recognition across diverse expert domains) and an **unbalanced layer** (exhibiting significant expert-popularity skewness). We then synthesized the traffic distribution by fitting these empirical frequencies to model the data-dependent routing behavior.

Incorporating Production Mechanisms. To faithfully replicate the traffic patterns of large-scale deployment, we first sample from an expert-selected CDF and subsequently incorporate two key mechanisms used in DeepSeek-V3. **Redundant Expert:** To mitigate hotspots, we simulated the deployment of 24 redundant experts on additional nodes. This mechanism splits the traffic of the original hot experts or the topology's load-balancing capability. **Shared Expert:** We introduced 40 shared experts, stochastically mapped to NPUs. This approach ensures a more uniform utilization of network bandwidth as shown in Fig. 14.



(a) All2All (Latency-Sensitive). (b) All2All with 320 nodes. (c) AllReduce with 320 nodes. (d) All2Allv with 2560 nodes. (e) AllReduce with 2560 nodes.

Figure 12: Performance of AllReduce and All2All on different topologies with 320 nodes and 2560 nodes.



(a) Deepseek Balanced Layer. (b) Deepseek Unalanced Layer.

Figure 13: Deepseek-V3 Workload Profiling.

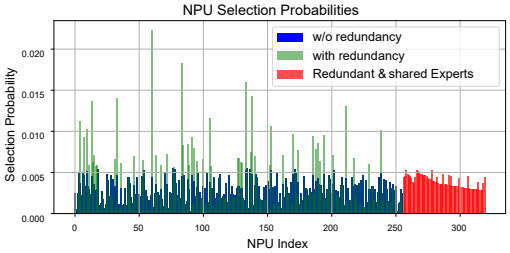
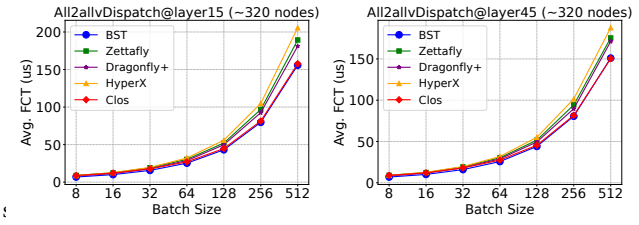


Figure 14: 320 MoE Experts Profiling.

By overlaying these mechanisms onto the profiled distributions, we constructed a comprehensive experimental setup comprising 320 experts (256 primary, 24 redundant, and 40 shared) mapped to 320 NPUs. This configuration generates a suite of synthetic yet realistic All2Allv traffic matrices, enabling rigorous evaluation of BST's ability to handle the dynamic and skewed communication patterns characteristic of next-generation MoE training at scale.



(a) Balanced Layer FCT. (b) Unbalanced Layer FCT.

Figure 15: Performance of Deepseek-V3 workload on different topologies with 320 nodes.

As shown in Fig. 15, across both balanced and unbalanced workload layers, BST achieves comparable performance with Clos, while delivering a 13.4% improvement over other topologies. This efficiency arises from BST's structural symmetry and differentiated routing, which optimize throughput for both balanced and skewed traffic patterns. Furthermore, its sparse interconnection fabric facilitates robust load balancing across physical links, ensuring high utilization even under intense MoE workloads.

7.3 Fault Resilience

To evaluate resilience, we run simulations on 320-node BST and Clos and measure the average and p99 flow completion time with results normalized to the failure-free topology. We first evaluated All2Allv from DeepSeek-V3 MoE in Sec. 7.2, where BST differs from Clos by less than 1.6%, and performance degradation under failures is modest (0.3% for link failures and only 8.4% for a single switch failure). This can be explained by the skewed expert popularity in All2Allv, where it targets hot experts and may avoid many failed paths. To better expose adverse cases, we therefore supplement the evaluation with balanced All2All traffic and AllReduce.

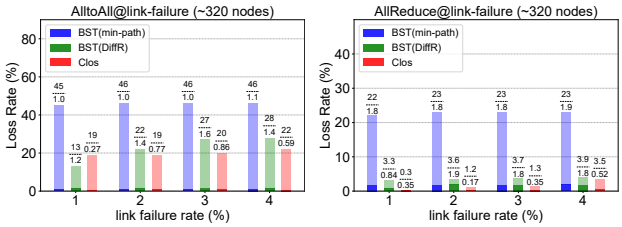


Figure 16: Performance Loss Rate under Link Failure. Bar labels use the $\frac{P99}{AVG}$ format.

Link failures. For link failures, failed uplinks are distributed across different leaf switches with link failure rate from 0% to 4%. As shown in Fig. 16, Clos is more stable under this mode because its path redundancy spreads each failed link over many equivalent spine paths where each affected leaf typically loses only a small fraction of its uplink capacity. BST with only minimal-path routing is more sensitive: when failures appear on multiple leaves, some source-destination pairs have fewer minimal paths available, so traffic may be forced to share the remaining link. Enabling BST differentiated routing (DiffR) reroutes part of the affected traffic onto detour paths, roughly halves the degradation, and keeps the p99 latency close to the Clos.

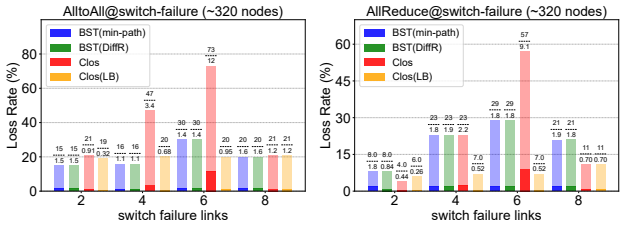


Figure 17: Performance Loss Rate under Switch Failure. Bar labels use the $\frac{\text{P99}}{\text{AVG}}$ format.

Switch failures. For switch failures, we inject failures on the uplinks of a single leaf switch. The affected leaf switch has eight uplinks, which we progressively disable 2-8 links. The 8 case is equivalent to a complete single switch failure and triggers route convergence. Both Minimal-path routing and DiffR of BST perform well because the affected BST leaf connects to multiple lossless domains, and route convergence quickly moves traffic away from failed uplinks. Clos is more sensitive before the switch is fully isolated if endpoints keep using failure-oblivious load balancing: traffic can be hashed unevenly onto the surviving uplinks, and the 6 uplink-failure case creates the worst many-to-one contention. We also evaluate an oracle endpoint load-balancing Clos(LB), where NPUs split traffic according to the remaining bandwidth of the remote failed leaf. This oracle scheme becomes optimal. In practice, however, it requires remote failure-induced load awareness, which current GPU/NPU collective stacks generally do not expose.

7.4 Sensitivity Analysis of DiffR

We further evaluate the sensitivity of the differentiated routing scheme. We use queue depth as the port-load indicator. Following the notation in Sec. 4.1, $T_{\max}$ acts as the PFC threshold that keeps the queue depth within the port buffer capacity. The queue depth threshold t_q controls whether a port is permitted to forward packets via detour routing. Therefore, we fix $T_{\max}$ to the empirical XOFF threshold and sweep t_q to observe DiffR's performance sensitivity.

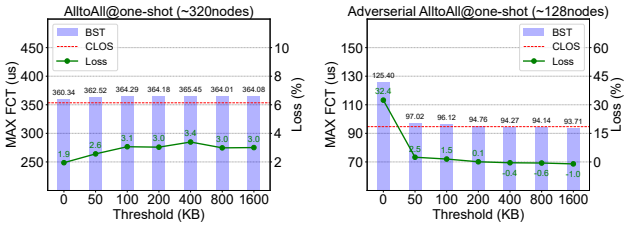


Figure 18: Performance of Different t_q under All2All.

For All2All, we use the one-shot algorithm where all flows start concurrently. DiffR($t_q = 0$) denotes BST minimal-path, where detour traffic is fully suppressed as shown in Fig. 18. The completion time of DiffR($t_q = 0$) is already close to Clos for genernal All2All case. As t_q increases, the overall performance fluctuates only within 1.9% to 3.4%, and quickly converges. We further construct adversarial collective traffic by running All2All over two BST domains. In this case, DiffR($t_q = 0$) completely suppresses detour traffic and makes BST lose about 32% performance compared with Clos. Increasing t_q allows detour flows to borrow other bandwidth, and DiffR starts to match Clos around t_q = 400KB. This result indicates that enabling detours is important for adversarial communication on BST, while t_q is not overly sensitive once it is large enough to expose useful detour paths.

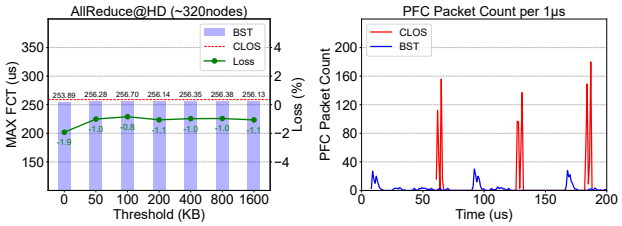


Figure 19: Performance of Different t_q under AllReduce and PFC Packet Count.

For AllReduce, we use the Halving-Doubling (HD) algorithm: eight phases total with six large intra-domain traffic phases and two samll cross-domain traffic phases. As shown in Fig. 19, BST stays close to Clos and even slightly outperforms it overall. Our analysis reveals that it benefits from one fewer switching layer in the last two latency-sensitive cross-domain phases. Consider the barrier synchronization phase in AllReduce, we also compare the PFC triggers with DiffR enabled on BST versus Clos. Clos exhibits large and concentrated PFC bursts, whereas BST shows intermittent small spikes with a comparable number of PFC triggers.

Overall, the performance is not highly sensitive to the DiffR settings. We therefore recommend enabling DiffR with $T_{\max}$ set to the empirical PFC threshold, while t_q can be set to 1/2 or 1/4 of $T_{\max}$, so that detour traffic can be exploited without severely contending with minimal-path traffic.

8 Testbed Evaluation

In this section, we construct a network topology using 4 Ascend 910B servers as shown in Fig. 20. While each server is equipped with 8 NPUs, only 6 per server (a total of 24 NPUs) are utilized in this network. Each of these NPUs is connected via a 200 Gbps port to its respective ToR switch. The ToR switches are then interconnected with the upper-layer Spine switches using two alternative topologies: a Clos network and the BST(4,2,1) structure. Both the ToR and Spine switches are equipped with 32,400 Gbps ports.

8.1 Collective and P2P Communication

We established virtualized switching units through VPN-based port isolation. This logical partitioning facilitated the deployment of BST and Clos, as shown in Fig. 21. The performance of these topologies was then evaluated using collective communication algorithms supported by HCCL. For a network-wide collective communication test, minimal-path routing tables are loaded to the switches; conversely, detour routing tables are uesd in inter-group P2P tests.

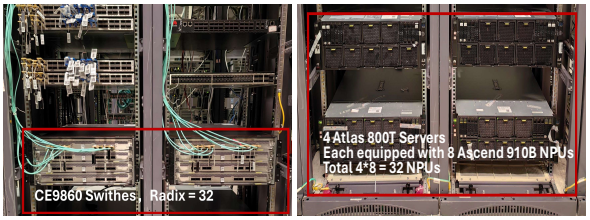
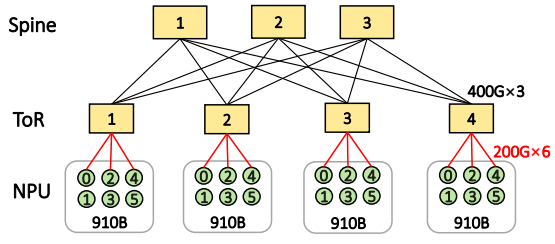
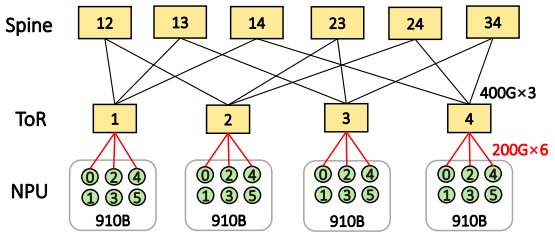


Figure 20: The testbed environment of BST.



(a) Clos.



(b) BST.

Figure 21: Logical topologies for testbed evaluation.

Table 3: HCCL Test results of BST and Clos

HCCL Test	Size	Data	Runtime(us)	
			Clos	BST(min-path)
All2All-Pairwise	24	1G	36793	37676
AllReduce-NHR	24	1G	74407	74755
AllReduce-Ring	24	1G	76914	69648

We use the HCCL test tool to evaluate the Pairwise algorithm for All2All, along with the Ring and Nonuniform Hierarchical Ring (NHR) algorithms for AllReduce, comparing their network-wide completion times for collective communication on BST and Clos topologies. To continuously make full use of the bandwidth for topology performance observation, we generate 1 G of data over 1000 iterations and compute the average completion time. In the experimental evaluation, the Clos topology utilizes the default-ordered Ranktable, whereas BST employs a load-balanced Ranktable. We set the communication domain size for both Clos and BST to 24. As empirically demonstrated in Table 3, BST achieves comparable completion time to Clos, with the Allreduce-Ring operation even exhibiting a 10% improvement, because the sparsity of BST mitigates issues such as ECMP hash collisions at the same scale.

Table 4: Inter-group P2P results of BST and Clos.

Allocation	Size	Runtime (μs)		
		Clos	BST(min-path)	BST(detour)
50%	12	26300	36182	27093
75%	16	26280	47299	31617
100%	24	26974	69890	43066

In addition, we conduct inter-group P2P communication experiments by partitioning 24 NPUs across 4 ToR switches into 2 distinct communication groups. Within each group, we vary the allocation fraction of NPUs allocated to P2P transmission (50%, 75%, 100%), while the remaining NPUs remain idle or are used for intra-group communication. This corresponds to a total active P2P scale of 24 × allocation fraction.

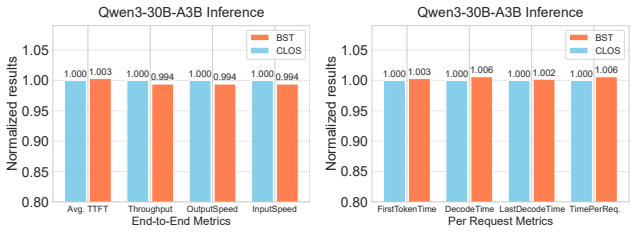
As shown in Table 4, when restricted to minimal-path routing, the BST topology's inability to resolve all hash collisions results in a 1.3× to 2.5× increase in completion time compared to Clos. This is primarily because non-detour routing results in wasted bandwidth on alternative paths, thereby limiting performance to the limited uplink bandwidth from ToR to Spine switches. Enabling detour routing in BST significantly improves performance: it is comparable to Clos with a 50% allocation fraction, and the performance gap is narrowed from 2.5× to 1.6× with a 100% allocation fraction.

Since the testbed uses current switches that only support BST detour routing through static routing-table, the P2P experiment is intended to quantify the benefit space of BST detour routing rather than to evaluate the full DiffR implementation. Hardware support for DiffR will be developed in next-generation switch chips, which further narrows the gap between BST and Clos (Appendix K).

8.2 Real-world AI Model Deployment

To validate the performance of BST in real-world AI workloads, we deployed the Qwen3-30B-A3B model [27] on Ascend 910B processors within our testbed. This model adopts a Mixture-of-Experts (MoE) architecture with 128 experts. We employed a distributed deployment strategy by configuring tensor parallelism (TP) and expert parallelism (EP). This configuration generates collective communication operations (AllReduce and All2AllV) across servers, rather than intra-server. The Qwen3 model was launched via the service mode of the MindIE inference engine [14]. We used the oasst1 dataset [20] to simulate human-like dialogue requests. The benchmark tool provided by MindIE was used to simulate concurrent multi-user requests and collect detailed performance, including throughput, average Time-To-First-Token, and decode time.

Since the dual-server deployment enables P2P communication, we enabled differentiated routing on BST. The normalized results against Clos are presented in Fig. 22. Specifically, Fig. 22(a) illustrates the end-to-end performance of the LLM inference service, which is strongly correlated with user experience. Excluding the interaction overhead between clients and the service, Fig. 22(b) presents the latency of individual inference requests. BST demonstrates performance closely aligned with Clos, with marginal deviations of only 0.6% to 0.2%. The results indicate that BST achieves LLM inference performance comparable to Clos, while using lower



(a) Inference metrics of LLMs, reflecting the actual user experience.

(b) Inference metrics for requests, representing the actual runtime.

Figure 22: Inference metrics for LLMs and requests.

radix switches to construct a network of the same scale as shown in Fig. 21 (Spine Radix: BST = 2, Clos = 4). This ensures that BST maintains a cost advantage in network deployment.

9 Related Work

Data Center Network (DCN) topologies are broadly classified into indirect-connect and direct-connect architectures. Clos [3] is the dominant indirect topology in both traditional DCNs and modern AI clusters, including Alibaba's HPN [25] and Nvidia's NVL72 [28]. Its key advantage is the ability to provide full bandwidth under arbitrary traffic patterns. However, scaling Clos requires additional network layers, leading to a rapid increase in deployment costs in large-scale systems. To address this issue, server-centric topologies are explored in which servers connect to a switching fabric, such as BCube [9], DCell [10], HS-DCell [30], BCCC [22].

BCube [9] accelerates one-to-several and one-to-all traffic by constructing edge-disjoint complete graphs and multiple edge-disjoint server spanning trees. DCell [10] is a recursively defined structure, in which a high-level DCell is constructed from many low-level DCells, and DCells at the same level are fully connected. HS-DCell [30] is a highly scalable DCell-based server-centric data center network topology with only 3 network port. BCCC [22] is a BCube-connected crossbar that can deliver good network performance using inexpensive, commodity off-the-shelf switches and servers with only two network interface card ports. Server-centric topologies can reduce the cost of network hardware but incur substantial performance losses, particularly for collective communication in AI applications. Zettafly [8] further studies the sparsity of the indirect topology and explores a new trade-off among scalability, throughput, and non-blocking regions. The Stacked Single-Path Tree (SSPT), including the Orthogonal Fat-Tree (OFT) [29], and the Multi-Layer Full-Mesh (MLFM) [18], focuses on cost-effective, diameter-two indirect topologies, which are special cases of BST with $\delta = 2$. They achieve high scalability at a constant, low cost per endpoint and are optimized for workloads like uniform random HPC traffic, whereas BST is specialized for large-scale collective communication and efficient scalability on LLM. In [7], Chang and Shao laid the groundwork with a combinatorial model for layered networks. Inspired by this work, we demonstrate that Steiner systems, acting as a novel generalization of the full-mesh topology, can provide an architectural breakthrough, yielding superior scalability, bisection bandwidth, symmetry, and routing simplicity compared to conventional designs.

Direct connection topologies enhance network scalability by directly connecting switching or computing devices. In AI computing, a well-known direct connection topology is the Torus [6, 31]. However, as the communication area expands, its All2All performance rapidly degrades, indicating that Torus is unsuitable for training or inference of MoE (Mixture of Experts) models. In the high-performance computing (HPC) domain, Dragonfly [19] is a commercialized direct connection topology. Compared to Clos, it offers a balance between cost and performance. Subsequently, Dragonfly+ [26] was proposed, extending Dragonfly by connecting switching nodes within the group in a fully bisectional manner. Another cost-effective direct connection topology is HyperX [1], which leverages high-radix switching capabilities to achieve performance comparable to folded Clos, with fewer switches. SlimFly [4, 5] and PolarFly [21] are the most typical representatives. They offer greater cost-effectiveness and communication performance, but their complex connectivity poses obstacle to application. Another emerging research direction is hybrid topologies (e.g., HammingMesh [12] and KNS [24]), which combine the cost advantages of direct-connection topologies with the performance advantages of indirect-connection topologies. However, due to deployment complexity, their scope of application remains relatively limited. 3D DCMesh [11] is a flexible direct-connect topology that uses OCS (Optical Circuit Switch) to arrange network links for the adaptation on deep neural network training workloads. Compared to Slim Fly and Dragonfly+, BST's construction uniquely provides modular cabling and deterministic collision-free route symmetry.

10 Conclusion

This paper proposes BST for LLM training and inference. Based on hypergraph, BST maintains sparse connectivity, differentiated routing, deadlock freedom, and topology-affined deployment with high scalability and low latency. Theoretical analysis shows BST matches the scale of a 3-layer Clos at $\tilde{5}0\%$ lower cost, and rivals ZettaFly's scale by 1.7× under similar cost, nearing the Moore bound. Experiments show 3.9%–11.8% gains in collective communication. BST sustains loads near 1 (uniform) and 0.5 (adversarial), matching Clos while outperforming SOTA by 13.4% on AI workloads.

Motivated by the low latency requirements of LLM inference, our ongoing research explores BST for single-tier, low-latency networks. Leveraging the multi fan-out capabilities of NPUs/GPUs, the devices and L1 Switches can be interconnected via BST. The network can thus scale-up by a factor of 8 to 16, depending on the fan-out of NPUs/GPUs. With NPUs of fan-out 16 and L1 switches of radix 128, the network scale can reach 1.5K. In contrast, a fully-connected design is limited to 128 by radix. BST over a single-tier topology aligns with EP scaling and supports future inference paradigms.

Acknowledgments

This work was supported by the National Key R&D Program of China [2024YFF0617700], the National Natural Science Foundation of China [U23A20309, 62272302, 62372296]. Dejun Kong and Xiaofeng Gao are with the Shanghai Key Laboratory of Scalable Computing and Systems. Xiaofeng Gao is the corresponding author. Thanks for Haitong Feng's help. This work does not raise any ethical issues.

References

[1] Jung Ho Ahn, Nathan Binkert, Al Davis, Moray McLaren, and Robert S Schreiber. 2009. HyperX: topology, routing, and packaging of efficient large-scale networks. In *The International Conference for High Performance Computing, Networking, Storage, and Analysis (SC)*. 1–11.

[2] Moonshot AI. 2026. Kimi K2.5. https://www.kimi.com/ai-models/kimi-k2-5.

[3] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. 2008. A scalable, commodity data center network architecture. *ACM SIGCOMM Computer Communication Review* 38, 4 (2008), 63–74.

[4] Maciej Besta and Torsten Hoefler. 2014. Slim Fly: A cost-effective low-diameter network topology. In *The International Conference for High Performance Computing, Networking, Storage, and Analysis (SC)*. 348–359.

[5] Nils Blach, Maciej Besta, Daniele De Sensi, Jens Domke, Hussein Harake, Shigang Li, Patrick Iff, Marek Konieczny, Kartik Lakhotia, Ales Kubicek, Marcel Ferrari, Fabrizio Petrini, and Torsten Hoefler. 2024. A High-Performance Design, Implementation, Deployment, and Evaluation of The Slim Fly Network. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. 1025–1044.

[6] Jose M Camara, Miquel Moreto, Enrique Vallejo, Ramon Beivide, Jose Miguel-Alonso, Carmen Martínez, and Javier Navaridas. 2010. Twisted torus topologies for enhanced interconnection networks. *IEEE Transactions on Parallel and Distributed Systems* 21, 12 (2010), 1765–1778.

[7] Yijia Chang, Xi Huang, Longxiulin Deng, Ziyu Shao, and Junshan Zhang. 2020. Systematic topology design for large-scale networks: A unified framework. In *IEEE Conference on Computer Communications (INFOCOM)*. 347–356.

[8] Dezun Dong, Ziyu Wang, and Fei Lei. 2025. Zettafly: A Network Topology with Flexible Non-blocking Regions for Large-scale AI and HPC Systems. In *ACM Annual International Symposium on Computer Architecture (ISCA)*. 835–848.

[9] Chuanxiong Guo, Guohan Lu, Dan Li, Haitao Wu, Xuan Zhang, Yunfeng Shi, Chen Tian, Yongguang Zhang, and Songwu Lu. 2009. BCube: a high-performance, server-centric network architecture for modular data centers. In *ACM SIGCOMM*. 63–74.

[10] Chuanxiong Guo, Haitao Wu, Kun Tan, Lei Shi, Yongguang Zhang, and Songwu Lu. 2008. Dcell: a scalable and fault-tolerant network structure for data centers. In *ACM SIGCOMM*. 75–86.

[11] Pengxing Guo, Xiangyu He, Yufei Yang, Kun Liu, Sijing Yu, Weigang Hou, and Lei Guo. 2023. Design of duel-core connected mesh topology and fine-grained fault-tolerant mechanism for 3D optical network-on-chip. *Science China Information Sciences* 66, 11 (2023), 212303.

[12] Torsten Hoefler, Tommaso Bonato, Daniele De Sensi, Salvatore Di Girolamo, Shigang Li, Marco Heddes, Deepak Goel, Miguel Castro, and Steve Scott. 2024. Hammingmesh: A network topology for large-scale deep learning. *Commun. ACM* 67, 12 (2024), 97–105.

[13] Jun-Ting Hsieh, Pravesh K. Kothari, and Sidhanth Mohanty. 2023. A simple and sharper proof of the hypergraph Moore bound. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2324–2344.

[14] Huawei. 2024. MindIE Inference Engine. https://gitcode.com/Ascend/MindIE-LLM/.

[15] Huawei. 2026. Open-usim. https://gitcode.com/open-usim/ns-3-ub.

[16] Vishesh Jain and Huy Tuan Pham. 2024. Optimal thresholds for Latin squares, Steiner Triple Systems, and edge colorings. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 1425–1436.

[17] 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 *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. 745–760.

[18] Georgios Kathareios, Cyriel Minkenberg, Bogdan Prisacari, German Rodriguez, and Torsten Hoefler. 2015. Cost-effective diameter-two topologies: Analysis and evaluation. In *The International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*. 1–11.

[19] John Kim, Wiliam J Dally, Steve Scott, and Dennis Abts. 2008. Technology-driven, highly-scalable dragonfly topology. *ACM SIGARCH Computer Architecture News* 36, 3 (2008), 77–88.

[20] Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi-Rui Tam, Keith Stevens, Abdullah Barhoum, Nguyen Minh Duc, Oliver Stanley, Richárd Nagyfi, Shahul ES, Sameer Suri, David Glushkov, Arnav Dantuluri, Andrew Maguire, Christoph Schuhmann, Huu Nguyen, and Alexander Mattick. 2023. OpenAssistant Conversations – Democratizing Large Language Model Alignment. arXiv:2304.07327 [cs.CL]

[21] Kartik Lakhotia, Maciej Besta, Laura Monroe, Kelly Isham, Patrick Iff, Torsten Hoefler, and Fabrizio Petrini. 2022. PolarFly: a cost-effective and flexible low-diameter topology. In *The International Conference for High Performance Computing, Networking, Storage, and Analysis (SC)*. 1–15.

[22] Xiao-Yan Li, Wanling Lin, Ximeng Liu, Cheng-Kuan Lin, Kung-Jui Pai, and Jou-Ming Chang. 2021. Completely independent spanning trees on BCCC data center networks with an application to fault-tolerant routing. *IEEE Transactions on Parallel and Distributed Systems* 33, 8 (2021), 1939–1952.

[23] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J.L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jiawei Wang, Jin Chen, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Litong Wang, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qinyu Chen, Qiushi Du, R.J. Chen, R.L. Jin, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runxin Xu, Ruoyu Zhang, Ruyi Chen, S.S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Shuting Pan, T. Wang, Tao Yun, Tian Pei, Tianyu Sun, W.L. Xiao, Wangding Zeng, Wanjia Zhao, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, X.Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaokang Zhang, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xinnan Song, Xinxia Shan, Xinyi Zhou, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, Y.K. Li, Y.Q. Wang, Y.X. Wei, Y.X. Zhu, Yang Zhang, Yanhong Xu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Yu, Yi Zheng, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Ying Tang, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yu Wu, Yuan Ou, Yuchen Zhu, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yukun Zha, Yunfan Xiong, Yunxian Ma, Yuting Yan, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z.F. Wu, Z.Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhipeng Xu, Zhiyu Wu, Zhongyu Zhang, Zhuoshu Li, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Ziyi Gao, and Zizheng Pan. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437* (2024).

[24] Roberto Peñaranda, Crispín Gómez, María E Gómez, Pedro López, and Jose Duato. 2016. The k-ary n-direct s-indirect family of topologies for large-scale interconnection networks. *The Journal of Supercomputing* 72, 3 (2016), 1035–1062.

[25] 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 *ACM SIGCOMM*. 691–706.

[26] Alexander Shpiner, Zachy Haramaty, Saar Eliad, Vladimir Zdornov, Barak Gafni, and Eitan Zahavi. 2017. Dragonfly+: Low cost topology for scaling datacenters. In *IEEE International Workshop on High-Performance Interconnection Networks in the Exascale and Big-Data Era (HiPINEB)*. 1–8.

[27] Qwen Team. 2025. Qwen3 Technical Report. arXiv:2505.09388 [cs.CL]

[28] Ajay Tirumala and Raymond Wong. 2024. Nvidia blackwell platform: Advancing generative ai and accelerated computing. In *IEEE Hot Chips 36 Symposium*. 1–33.

[29] Marcos Valerio, Louise E Moser, and P Michael Melliar-Smith. 1994. Recursively scalable fat-trees as interconnection networks. In *IEEE Annual International Phoenix Conference on Computers and Communications*. 40.

[30] Guijuan Wang, Yazhi Zhang, Jiguo Yu, Meijie Ma, Chunqiang Hu, Jianxi Fan, and Li Zhang. 2024. HS-DCell: A highly scalable DCell-based server-centric topology for data center networks. *IEEE/ACM Transactions on Networking* 32, 5 (2024), 3808–3823.

[31] Ting Wang, Zhiyang Su, Yu Xia, Bo Qin, and Mounir Hamdi. 2014. NovaCube: A low latency Torus-based network architecture for data centers. In *IEEE Global Communications Conference*. 2252–2257.

[32] Renjie Yao and Yaoyao Ye. 2020. Toward a High-Performance and Low-Loss Clos-Benes-Based Optical Network-on-Chip Architecture. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 39, 12 (2020), 4695–4706.

[33] Shizhen Zhao, Qizhou Zhang, Peirui Cao, Xiao Zhang, Xinbing Wang, and Chenghu Zhou. 2023. Flattened Clos: Designing High-performance Deadlock-free Expander Data Center Networks Using Graph Contraction. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. 663–683.

[34] Ruidong Zhu, Ziheng Jiang, Chao Jin, Peng Wu, Cesar A. Stuardo, Dongyang Wang, Xinlei Zhang, Huaping Zhou, Haoran Wei, Yang Cheng, Jianzhe Xiao, Xinyi Zhang, Lingjun Liu, Haibin Lin, Li-Wen Chang, Jianxi Ye, Xiao Yu, Xuanzhe Liu, Xin Jin, and Xin Liu. 2025. MegaScale-Infer: Serving Mixture-of-Experts at Scale with Disaggregated Expert Parallelism. arXiv:2504.02263 [cs.DC]

Appendix

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

A Proof of Theorem 1

Proof. Each switch in the Spine layer is connected to node groups in the ToR layer, with a maximum of $\lfloor R/\delta \rfloor$ groups connected per Spine switch. The number of ToR nodes in each group is $r_{up}(\delta-1)+1$; thus, the network scale is $V_{tor} = (r_{up}(\delta-1)+1)\times\lfloor R/\delta \rfloor$, completing the proof. □

B Proof of Theorem 2

Proof. First, to prove that any collective communication is non-blocking on BST topology, it suffices to prove that the BST topology is non-blocking for All2All traffic when $r_{up}/r_{down} = 1$. For any node pair (i, i'), there is a unique minimal path between i and i', The traffic from node i to node i' only needs to pass through the minimal path between i and i', and the minimal paths between different node pairs will not be completely duplicated. Each ToR switch sends a fixed amount of traffic to a specific Spine switch and receives a fixed amount of traffic from Spine switches. All communication between endpoints takes the shortest path, and upward bandwidth equals downward bandwidth, so every link between Spine and ToR switches carries the same traffic load. Therefore, when the speedup ratio $r_{up}/r_{down} = 1$, collective communication can achieve non-blocking transmission.

Then, according to the Birkhoff-Von Neumann Theorem, if we prove that the BST topology is rearrangeably non-blocking, it implies that the BST topology must be rearrangeably non-blocking whenever the traffic matrix is any permutation matrix. Consider any set of peer-to-peer (P2P) permutation traffic. For each pair of nodes, the bandwidth between them equals the direct bandwidth plus the detour bandwidth. This is because there exists a unique minimal path between each pair of nodes, along with multiple detour paths that pass through another Tor node. The bandwidth between each pair of nodes is given by $1+\frac{\omega-\delta}{2(\delta-1)}$. In total, the bandwidth occupied by all ω node pairs is $\frac{\omega(\omega+\delta-2)}{2(\delta-1)}$. The total bandwidth in the network is $\omega r_{up} = \frac{\omega(\omega-1)}{\delta-1}$. Therefore, the speedup ratio is $\frac{2(\omega-1)}{\omega+\delta-2} < 2$. When the speedup ratio $r_{up}/r_{down} = 2$, the BST topology is rearrangeably non-blocking, which completes the proof. □

C Proof of Theorem 3

Proof. As is known, the Moore bound for a diameter-2 direct graph is r_{up}^2+1. However, for BST, the lower (ToR) layer can contain a maximum of $(r_{up}+1)*R$ nodes, where R is the radix of a switch. Since we have $r_{up} = R - r_{down}$, the network topology using BST is larger than using a direct graph. As a result, the maximum size of BST is larger than that of the diameter-2 topologies, such as Slimfly. Normally, we have a speedup ratio from 1-2, which means the maximum size of the BST is around 1.5-2 times larger than other known diameter-2 topologies.

We will present an example to illustrate the observation above. As claimed in [5], the non-blocking Slimfly achieves a speedup of 2[3]. The Slimfly topology, which features a hyperparameter $q = 4w + \theta$ (where $w \in \mathbb{N}$ and $\theta \in \{-1, 0, 1\}$). We assume $R = 36$ to facilitate a more intuitive comparison in the subsequent analysis. For the SlimFly topology, the number of its switch ports is $R \approx \frac{9}{4}q = 36$: each switch has $q = \frac{4}{9}R = 16$ ports connected between racks, $\frac{q-\theta}{2} \approx \frac{2}{9}R = 8$ ports connected within a rack, and $p = \lceil \frac{3q-\theta}{4} \rceil \approx \frac{1}{3}R = 12$ ports connected to computing nodes. Consequently, the total number of switches is $2q^2 = \frac{32}{81}R^2 = 512$, and the total number of computing nodes is $2q^2p \approx \frac{32}{243}R^3 = 6144$. For the BST topology, when $R = 32$, the maximum number of computing nodes is $\frac{2}{9}R^3 = 8712$ based on the previous conclusions. Thus, under the same radix constraints and non-blocking conditions, the BST topology achieves a larger theoretical maximum networking scale than SlimFly. □

[3]Typically, as the radix R grows larger, the speedup ratio of 2-hop direct-connect non-blocking networks typically converges to 2 when these networks approach the Moore bound.

D Proof of Theorem 4

Proof. Consider one lower-level group in $\text{BST}(\omega, \delta, \kappa)$ with vertex set V_g and $|V_g| = \omega$. Let B_0 denote the per-pair logical bandwidth provided by one upper-level switch. For any bisection $(S, \bar{S})$ of this group, the two parts satisfy

$$|S| = \left\lfloor \frac{\omega}{2} \right\rfloor, \qquad |\bar{S}| = \left\lceil \frac{\omega}{2} \right\rceil.$$

Hence, the number of lower-level node pairs crossing the bisection is

$$|S||\bar{S}| = \left\lfloor \frac{\omega}{2} \right\rfloor \left\lceil \frac{\omega}{2} \right\rceil = \left\lfloor \frac{\omega^2}{4} \right\rfloor.$$

By the construction of $\text{BST}(\omega, \delta, \kappa)$, every pair of lower-level nodes in the same group is contained in exactly one degree-δ hyperedge, i.e., one upper-level switch-mediated logical connection. If the bandwidth of each such two-hop logical connection is B_0, the bisection bandwidth within one group is therefore

$$B_{\text{bisect}} = |S||\bar{S}|B_0 = \left\lfloor \frac{\omega^2}{4} \right\rfloor B_0.$$

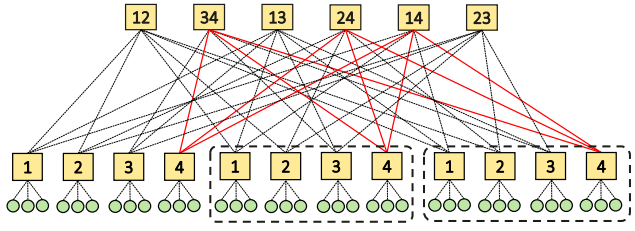
Under the all-pairs logical traffic model, each lower-level node injects traffic to the other $\omega - 1$ nodes, so the aggregate logical injection bandwidth is $\omega(\omega-1)B_0$. Thus, normalizing the bisection bandwidth by this aggregate injection bandwidth gives the same asymptotic factor as a complete graph:

$$\frac{B_{\text{bisect}}}{\omega(\omega-1)B_0} = \frac{\left\lfloor \frac{\omega^2}{4} \right\rfloor}{\omega(\omega-1)} \to \frac{1}{4}.$$

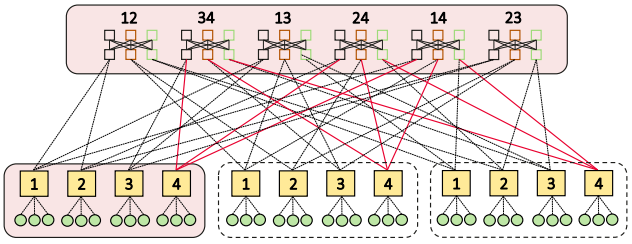
□

E Proof of Corollary 1

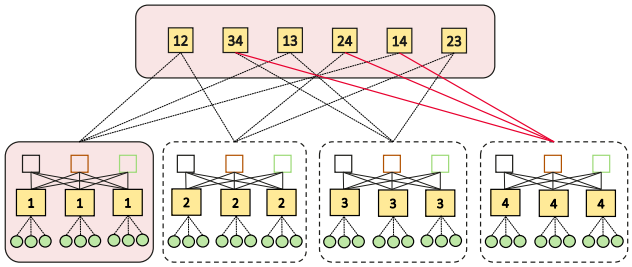
Proof. We prove the corollary in three cases. If there is only one detour flow, no deadlock occurs. If there are two detour flows, let $F_1 = (v_1, v_2, v_3, v_4, v_5)$ and $F_2 = (\hat{v}_1, \hat{v}_2, \hat{v}_3, \hat{v}_4, \hat{v}_5)$. If F_1 holds the link (v_i, v_{i+1}) and waits for the remaining links including $(v_{i'}, v_{i'+1})$ (where $i < i'$), while F_2 holds $v_{i'}, v_{i'+1})$ and waits for the remaining links including (v_i, v_{i+1}), this would theoretically satisfy the conditions for a deadlock. Next, we show that this would not happen in 3 subcases. (i) If $i + 1 = i'$, then v_{i+1} would appear twice in F_2, which conflicts with the definition of detour paths in the BST topology. (ii) If $i+2 = i'$, we only consider the scenario where $i = 1$, as the analysis



(a) Example of Scalable BST(4, 2, 3).



(b) Clos over BST.



(c) BST over Clos.

Figure 23: Logical topology of two-layer BST.

for $i = 2$ is analogous. If $i = 1$, F_2 would be $(v_3, v_4, v_1, v_2, \hat{v}_5)$. According to the definition of BST topology, $v_4 = v_2$, contradicting the definition of detours. (iii) If $i+3 = i'$, we have $F_2 = (\hat{v}_1, v_4, v_5 = v_1, v_2, \hat{v}_5)$, which means $v_1 = v_5$, and conflicts with definition. If there are three detour flows, let $F_1 = (v_1, v_2, v_3, v_4, v_5)$, $F_2 = (v_3, v_4, v_5, v_6, v_7)$, and $F_3 = (v_8, v_6, v_7, v_2, v_3)$. Let F_1 holds (v_2, v_3) and waits for the remaining unoccupied segments of its path, including (v_3, v_4), F_2 holds (v_3, v_4) and waits for the remaining unoccupied segments of its path, including (v_6, v_7), and F_3 holds (v_6, v_7) and waits for the remaining unoccupied segments of its path, including (v_2, v_3). A closed dependence loop exists among flows F_1, F_2, and F_3, and this loop thereby gives rise to a deadlock, which completes the proof. □

F Proof of Corollary 2

Proof. All minimal-path flows consist of an upward flow (from a ToR node to a Spine node) followed by a downward flow (from the Spine node to a ToR node), adhering to the order of "upward first, then downward." Thus, they obviously cannot form cycles with other minimal-path flows.

Since two detour flows cannot form a deadlock cycle, and a minimal-path flow can be regarded as a component of a detour flow, a minimal-path flow will not form a deadlock cycle with a detour flow. □

G Logical Topologies: Combining BST with Clos

In Fig. 23, BST(4, 2, 3) can be formulated as a lossless task partition problem, corresponding to two following equivalent logical topologies. Collective communication is lossless on both topologies.

Clos over BST. The upper-layer employs a Clos topology, while the lower-layer employs a BST topology. The partition granularity is set to a factor of the number of NPU cards per BST group. In Fig. 23(b), taking node 1 of the first BST group as an example, Spine 12 forms a Clos fabric by interconnecting node 1 and 2 from each group; Spine 13, node 1 and 3; Spine 23, node 2 and 3. On the one hand, there is a single minimal path from node 1 to another intra-group or inter-group node, respectively, 2/3/4. On the other hand, node 1 has three minimal paths to inter-group nodes with the same number. Clos over BST allows node 1 to achieve full bisectional bandwidth to the entire network by load-balancing across its three uplinks, applicable to large-scale tasks ($4 \times 8 \times n$, where $n \leq 3$).

BST over Clos. The upper-level layer employs a BST topology, while the lower-level layer utilizes a Clos topology. In Fig. 23(c), Spine 12, 13, and 14 interconnect node 1 of each group in Clos-like topology. Meanwhile, there are 3 minimal paths from node 1 to the other inter-group nodes, each with the same number. Communication between the same-numbered nodes can utilize the full bandwidth of a Clos network, making the interconnect non-blocking for this traffic pattern. We set the partition granularity to be no greater than the number of groups. The BST over Clos topology is suitable for small-scale tasks ($\leq 3 \times 8$), in which communication between nodes with the same number across different groups can be load-balanced to fully saturate the bandwidth of their three uplinks.

H An Example of BST for $\delta = 4$

This part provides a worked example illustrating the δ-merge construction for $\delta > 3$ and its deployment on switches.

Since the Hiddenfly topology HF(ω, κ) is a Sparse Bipartite Biregular graph where each group of ω lower-level nodes (ToRs) is fully connected by $\binom{\omega}{2}$ binary hyperedges (each representing a Spine switch of degree 2). Applying the δ-combinatorial merge merges sets of binary hyperedges into higher-order hyperedges of δ, so that each pair of lower-level nodes belongs to only one hyperedge.

For $\delta = 4$, choose $r_{up} = 4$ upward ports per ToR yields a group size $\omega = 13$. This corresponds to a 13-point Steiner system $S(2, 4, 13)$, a balanced incomplete block design with parameters ($v = 13, k = 4, \lambda = 1$). The structure is illustrated in Fig. 24. The right picture shows the hypergraph representation; the left picture depicts the corresponding Spine-ToR base topology. In this base sub-network, 13 Spine switches and 13 ToR switches each use exactly 4 ports, a parameter set that inherently avoids port fragmentation.

To deploy the base topology on real 64-port switches, we apply the Scalable BST construction using ToR fabric scaling. The Spine layer is replicated 8 times and the ToR layer 16 times, yielding 104 physical Spine switches and 208 physical ToR switches. Each ToR is configured with 32 downlinks (to NPUs) and 32 uplinks, partitioned into 8 groups corresponding to the 8 Spine copies. Within each group, a ToR connects to exactly 4 Spine switches following the incidence pattern of the base Steiner system. This expansion supports $208 \times 32 = 6656$ NPUs. The topology is sketched in Fig. 25.

The example exhibits three desirable characteristics:

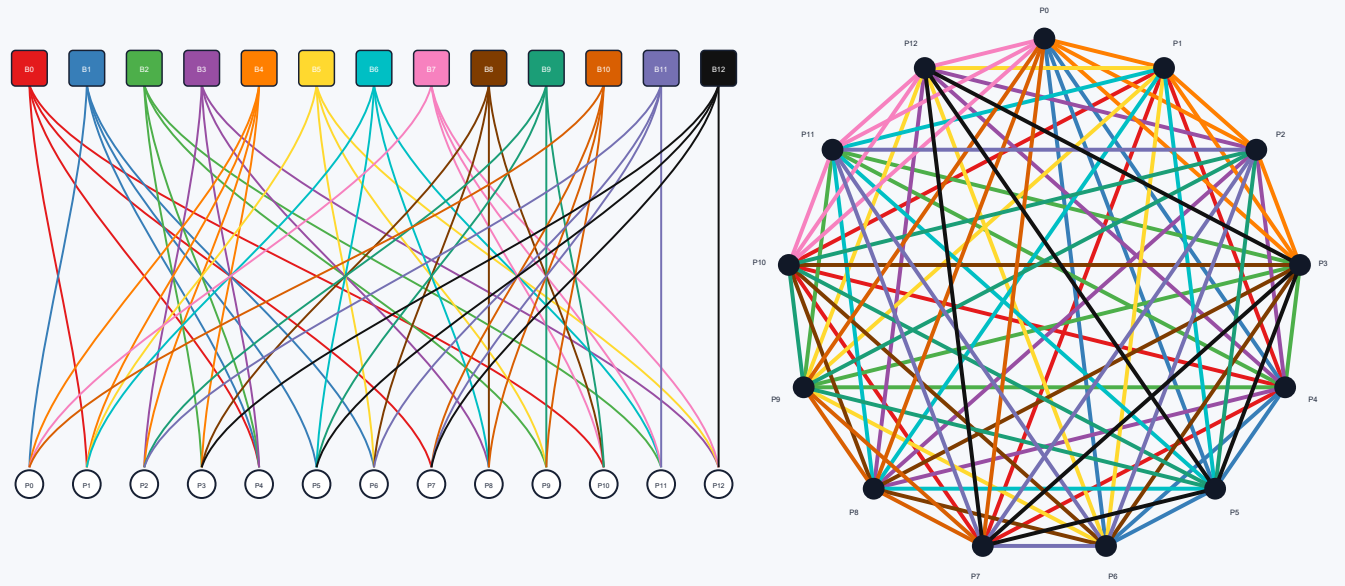


Figure 24: An Worked Example for $\delta = 4$.

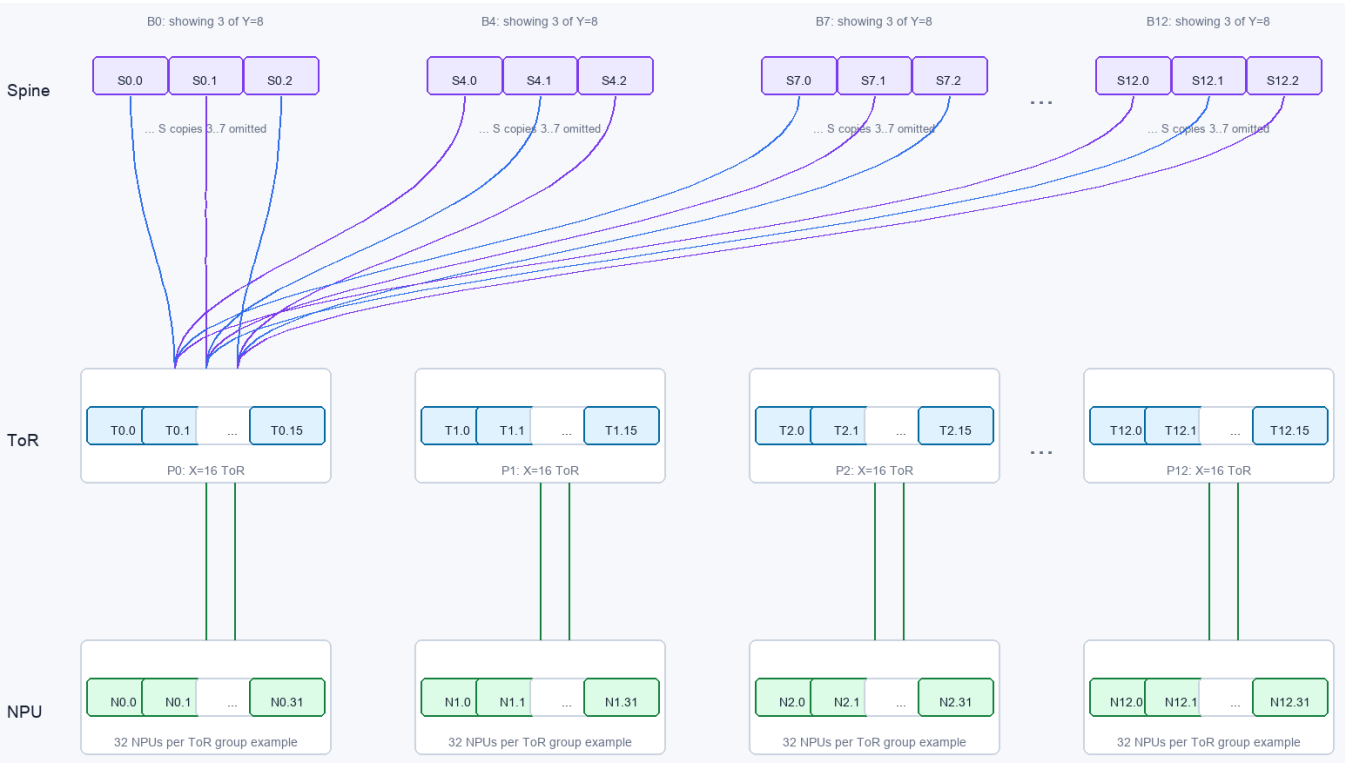
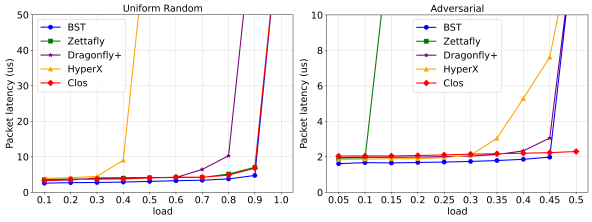


Figure 25: The Scaling of the Example for Deployment.

- **No port fragmentation.** The base parameters $\omega = 13$ and $\delta = 4$ divide evenly into the switch radix $R = 64$, so every port is fully utilized.
- **Preserved symmetry and non-blocking property.** The expansion replicates the exact incidence pattern of the base Steiner system, retaining BST properties such as balance, sparsity, and diameter 2 within a group.



(a) Uniform random. (b) Adversarial.

Figure 26: Latency-Throughput Comparison.

- **Generality.** The same methodology applies to any switch radix R that is a multiple of δ (e.g., 128 or 256).

I Resource Utilization Analysis

In this subsection, we simulate resource utilization under various task-size distributions. The NPU utilization rate is calculated as the average of the ratio of the NPU's non-idle time to its total task time. Within the operational size range of the NPU, which accommodates AI training tasks requiring NPU quantities from 8 to $2K$ and from 16 to $4K$, we construct the following four task size distributions: Uniform distribution: equal probability for all task sizes. Small-task-centric distribution: higher likelihood of small-sized tasks. Large-task-centric distribution: higher likelihood of large-sized tasks. Medium-task-centric distribution: higher likelihood of medium-sized tasks.

We compared the utilization rate between BST$(5, 2, 1)$ ($20K$ NPUs per cluster) and the largest 2-layer Clos scale ($8K$ NPUs per cluster) under the radix $R = 128$ condition. To maintain a total of 80K NPUs and enable direct comparison, we constructed a system comprising 4 BST topologies and compared it with a baseline system comprising 10 Clos topologies. To simulate real-world scenarios under varying task distributions, we generated and submitted random task queues to both topologies, with runtime profiles from actual workloads. The results are shown in Table 5. BST achieved a 6.74% higher resource utilization rate than the baseline, highlighting its superior scalability and efficiency in large-scale environments. In addition, Table 6 compares the resource utilization of BST with $2.5K$ NPUs per cluster and Clos with $1K$ NPUs per cluster. To maintain a unified total of $10K$ NPUs, we ultimately used four BST clusters and ten Clos clusters. Comparing with Table 5, we find that the advantage of BST over Clos increases with the size of the network.

J Typical Traffic Pattern Injection

To characterize steady-state congestion under variable-load scenarios, the network topology is seeded with two typical traffic patterns (uniform random and adversarial), and end-to-end latency is analyzed statistically on a per-packet basis. The simulation is conducted on a topology with 320 nodes. Specifically, we simulate different load conditions by varying the packet transmission rate of the source nodes. The load is defined as the total packet transmission rate divided by the total NPU bandwidth. The switch is configured with an infinite buffer, so no packet loss occurs during congestion. Meanwhile, the switch is configured with an infinite buffer, so no packet loss occurs during congestion.

Uniform random: In the experimental configuration, each computing node is programmed to stochastically select communication peers as destinations and subsequently generate an arbitrary data payload for transmission. We utilize minimal-path routing as the fundamental forwarding strategy and present the results in Fig. 26(a). Within the bounds of latency convergence, the BST topology exhibits lower packet latency than other topologies because it requires only a single layer of switching. HyperX saturates the bandwidth at $load = 0.5$ because the 3-hop minimal path experiences fair resource competition with the 2-hop minimal path for inter-group traffic. In contrast, BST and the other 3 topologies reach saturation when the load approaches 1.

Adversarial: We partition the total number of network nodes into two equal halves and generate peer-to-peer (P2P) flows. For example, nodes $0 \sim 159$ communicate with nodes $160 \sim 319$. In these experiments, all topologies use detour routing except Clos. As depicted in Fig. 26(b), an inflection point appears in BST and DragonFly+ at $load = 0.45$, and the packet latency will increase indefinitely as it approaches 0.5. This is because the detour traffic occupies additional bandwidth within the topology, and the link is already saturated at $load = 0.5$. On the contrary, an inflection point appears in HyperX at $load = 0.3$, for which some of its minimal paths also consume duplicate bandwidth. What's more, an inflection point appears in Zettafly at $load < 0.15$. The underlying reason is that the communication bottleneck for cross-rail traffic occurs on the directly connected link of the receiving NPU, which has a bandwidth of only 800 G. Meanwhile, the source transmits at a rate of $16 * 400G * load$, resulting in full capacity at $load = 0.125$. A key advantage of the Clos topology is its use of minimal routing, which ensures stable latency even under high loads ($load > 0.5$).

K Multi-Task Deployment Test

BST can be viewed as a sparse fabric over multiple Clos domains. Fig. 27 illustrates BST$(5, 2, 1)$, where five Clos domains are connected pairwise through the BST fabric. Under first-come, first-served scheduling with uniform priority, we design a multi-task affined deployment strategy that follows a simple topology-aware rule. If a task size is divisible by 4 or 5, the task is evenly distributed across 4 or 5 Clos domains within BST; otherwise, it is placed within a single Clos domain. This strategy fully utilizes server egress links in the non-blocking setting.

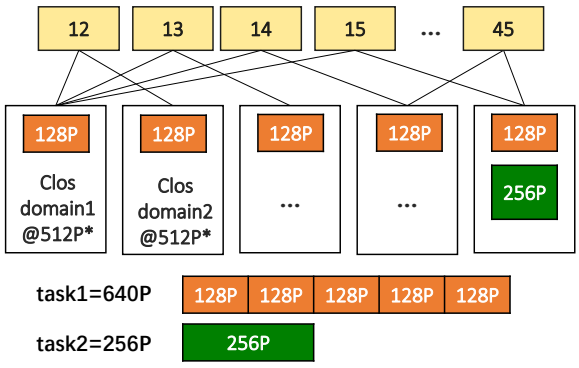


Figure 27: Deployment of Tasks in BST$(5, 2, 1)$.

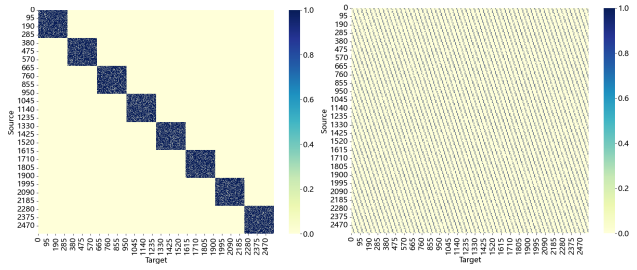
Table 5: Predictive NPU Utilization Rate of BST with $20K$ NPUs and Clos with $8K$ NPUs.

Task size	Distribution	Uniform		Small-task-centric		Large-task-centric		Medium-task-centric	
	Topo	BST	Clos	BST	Clos	BST	Clos	BST	Clos
8~2k	Task 1	98.02%	95.72%	98.57%	96.85%	97.96%	95.38%	98.37%	96.87%
	Task 2	97.91%	95.67%	98.48%	97.16%	97.55%	95.31%	98.46%	96.55%
	Task 3	98.07%	96.06%	98.33%	96.75%	97.64%	95.35%	98.40%	96.85%
16~4k	Task 4	94.83%	88.93%	96.00%	90.92%	94.36%	87.19%	95.58%	90.15%
	Task 5	94.77%	88.57%	95.92%	90.57%	94.14%	87.07%	95.43%	90.47%
	Task 6	94.71%	88.58%	95.84%	90.80%	94.15%	88.16%	95.58%	90.46%

Table 6: Predictive results of BST with $2.5K$ NPUs and Clos with $1K$ nodes.

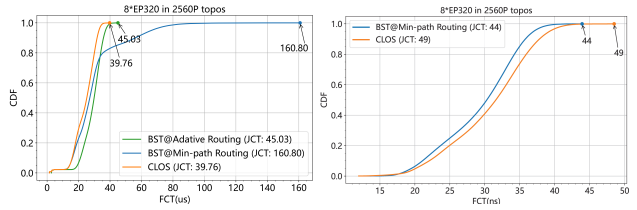
Task size	Distribution	Uniform		Small-task-centric		Large-task-centric		Medium-task-centric	
	Topo	BST	Clos	BST	Clos	BST	Clos	BST	Clos
8~320	Task 1	95.75%	90.57%	96.32%	92.01%	95.16%	89.64%	96.14%	92.20%
	Task 2	95.56%	90.39%	96.10%	92.22%	95.13%	89.99%	96.38%	92.67%
	Task 3	95.50%	90.47%	96.12%	92.13%	95.23%	89.47%	96.09%	92.78%
8~640	Task 4	90.94%	81.13%	92.26%	83.48%	90.15%	79.21%	91.85%	83.00%
	Task 5	91.05%	80.61%	92.63%	83.68%	90.18%	78.94%	91.84%	83.10%
	Task 6	90.93%	81.15%	92.60%	84.54%	90.82%	79.55%	91.64%	82.62%

To evaluate the performance of BST and affinity deployment, with Flow Completion Time (FCT) serving as the primary performance metric, we used Clos and sequential deployment as baselines and conducted extensive experiments across 8 EP320 domains.



(a) Sequential deployment. (b) Affined deployment.

Figure 28: Traffic heatmap of sequential deployment and affined deployment.



(a) Sequential deployment. (b) Affined deployment.

Figure 29: All2Allv on BST and Clos with sequential deployment and affined deployment, respectively.

The BST and Clos used in the following experiments comprises 5 groups, each with 512 NPUs. For sequential deployment, we assign 8 EP tasks to groups sequentially, with a subset of tasks requiring communication between adjacent groups. As for affined deployment, each communication domain of EP320 is deployed across 5 groups and each group consists 64 computing nodes. The traffic distribution diagram of sequential and affined deployment is shown in Fig. 28. Sequential deployment results in hotspot traffic, whereas affined deployment ensures uniform distribution across the entire network. Therefore, sequential deployment enables detour routing to optimize performance, whereas affined deployment uses only the minimal path.

In Fig. 29(a), we observe that the detour scheme significantly enhances the overall network transmission performance under sequential deployment. Specifically, it achieves a 3.57 improvement over minimal-path routing and outperforms Clos by 13%. In Fig. 29(b), BST achieves a 9.46% improvement over Clos under affined deployment because BST requires one fewer switching level than Clos.

L Topologies of the simulation

Fig. 30 presents the topologies used in the simulation evaluation. HRS stands for high radix switch and LRS stands for low radix switch.

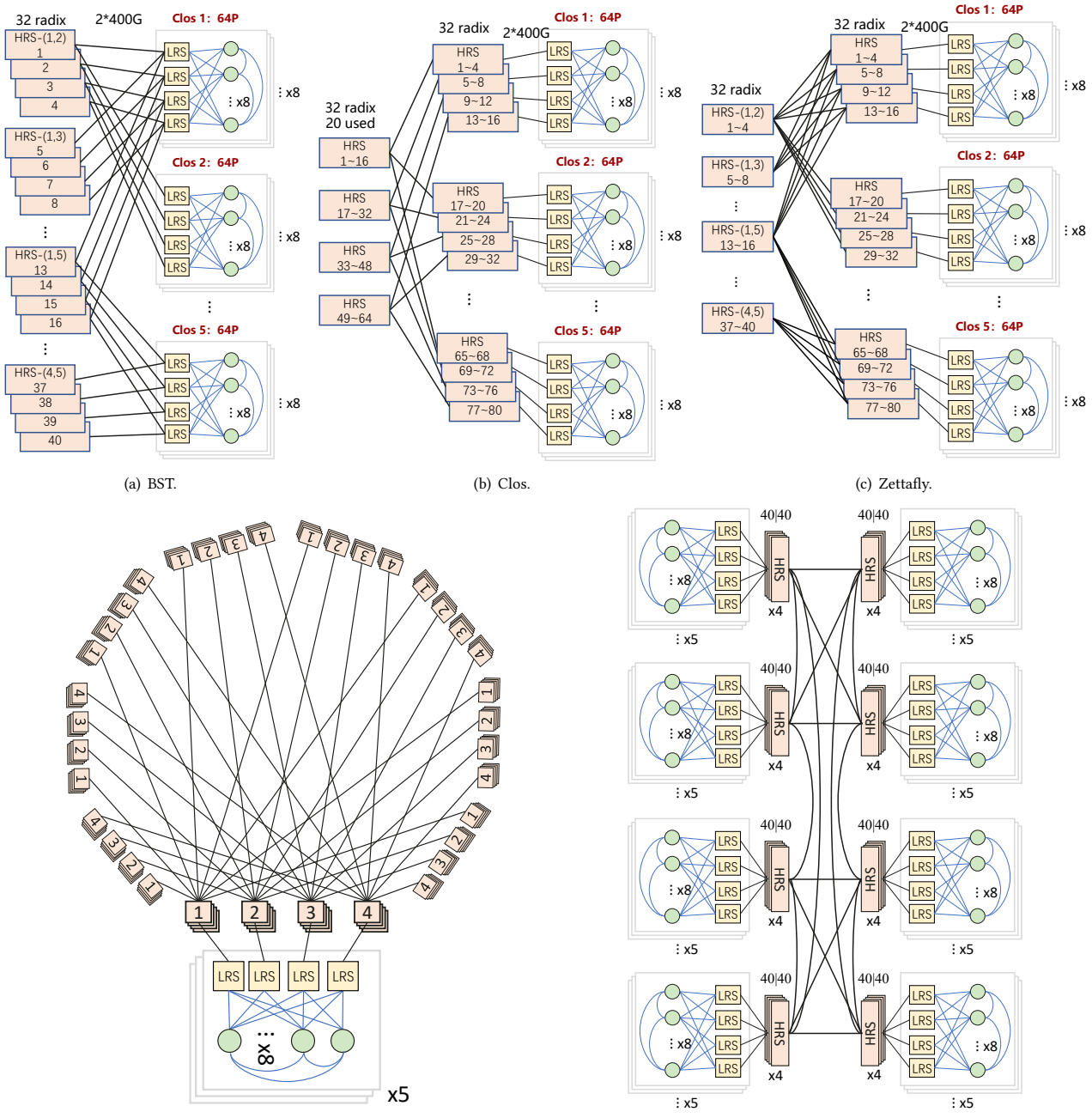
M BST Deployability Analysis

Table 7 presents a partial list of feasible Balanced Sparse Tree (BST) topology configurations. The table details the network parameters for different network scales, including radix, network parameters, strand waste and so on.

Received 6 February 2026; revised 19 June 2026; accepted 29 June 2026

Table 7: The Configurations of Feasible BST Topologies (Partially)

Radix	v	u	r_{up}	δ	X ToR Copy	Y Spine Copy	ToR Count	Spine Count	NPU Count	Tor Up/ Down Ports	Spine Down Used	Spine Waste	Spine Waste Rate	Total Waste Ports	Total Network Ports	Total Waste Rate	Total Switch	Total Link
64	13	13	4	4	16	8	208	104	6656	32	64	0	0	0	13312	0	312	6656
64	9	36	8	2	32	4	288	144	9216	32	64	0	0	0	18432	0	432	9216
64	5	10	4	2	32	8	160	80	5120	32	64	0	0	0	10240	0	240	5120
64	3	3	2	2	32	16	96	48	3072	32	64	0	0	0	6144	0	144	3072
128	57	57	8	8	16	8	912	456	58368	64	128	0	0	0	116736	0	1368	58368
128	25	50	8	4	32	8	800	400	51200	64	128	0	0	0	102400	0	1200	51200
128	17	136	16	2	64	4	1088	544	69632	64	128	0	0	0	139264	0	1632	69632
128	9	36	8	2	64	8	576	288	36864	64	128	0	0	0	73728	0	864	36864
256	97	776	32	4	64	4	6208	3104	794624	128	256	0	0	0	1589248	0	9312	794624
256	33	528	32	2	128	4	4224	2112	540672	128	256	0	0	0	1081344	0	6336	540672
256	49	196	16	4	64	8	3136	1568	401408	128	256	0	0	0	802816	0	4704	401408
256	17	136	16	2	128	8	2176	1088	278528	128	256	0	0	0	557056	0	3264	278528
64	33	176	16	3	21	2	693	352	22176	32	63	1	0.0156	352	44704	0.0079	1045	22176
64	49	56	8	7	9	4	441	224	14112	32	63	1	0.0156	224	28448	0.0079	665	14112
64	9	12	4	3	21	8	189	96	6048	32	63	1	0.0156	96	12192	0.0079	285	6048
128	129	2752	64	3	42	1	5418	2752	346752	64	126	2	0.0156	5504	699008	0.0079	8170	346752
128	33	176	16	3	42	4	1386	704	88704	64	126	2	0.0156	1408	178816	0.0079	2090	88704
128	49	56	8	7	18	8	882	448	56448	64	126	2	0.0156	896	113792	0.0079	1330	56448
128	9	12	4	3	42	16	378	192	24192	64	126	2	0.0156	384	48768	0.0079	570	24192
256	129	2752	64	3	85	2	10965	5504	1403520	128	255	1	0.0039	5504	2812544	0.0020	16469	1403520
256	65	208	16	5	51	8	3315	1664	424320	128	255	1	0.0039	1664	850304	0.0020	4979	424320
256	49	56	8	7	36	16	1764	896	225792	128	252	4	0.0156	3584	455168	0.0079	2660	225792
256	33	176	16	3	85	8	2805	1408	359040	128	255	1	0.0039	1408	719488	0.0020	4213	359040
256	9	12	4	3	85	32	765	384	97920	128	255	1	0.0039	384	196224	0.0020	1149	97920



(a) BST. (b) Clos. (c) Zettafly.

(d) Dragonfly+. (e) HyperX.

Figure 30: Topologies used in evaluation experiments: Hierarchical or direct topologies above with Clos and direct connection at bottom.