When Light Bends to the Collective Will: A Theory and Vision for Adaptive Photonic Scale-up Domains

Vamsi Addanki
Purdue University
USA

Abstract

As chip-to-chip silicon photonics gain traction for their bandwidth and energy efficiency, collective communication has emerged as a critical bottleneck in scale-up systems. Programmable photonic interconnects offer a promising path forward: by dynamically reconfiguring the fabric, they can establish direct, high-bandwidth optical paths between communicating endpoints — *synchronously and guided by the structure of collective operations* (e.g., AllReduce). However, realizing this vision — *when light bends to the collective will* — requires navigating a fundamental trade-off between reconfiguration delay and the performance gains of adaptive topologies.

In this paper, we present a simple theoretical framework for adaptive photonic scale-up domains that makes this trade-off explicit and clarifies when reconfiguration is worthwhile. Along the way, we highlight a connection — not surprising but still powerful — between the Birkhoff–von Neumann (BvN) decomposition, maximum concurrent flow (a classic measure of network throughput), and the well-known α–β cost model for collectives. Finally, we outline a research agenda in algorithm design and systems integration that can build on this foundation.

1 Introduction

The massive scale of modern distributed computing [5, 10, 15, 26, 29, 35, 42] — spanning hyperscale data centers to tightly-coupled HPC clusters — has made efficient collective communication a critical bottleneck [18, 21, 40]. Scale-up[1] networks typically connect multiple GPUs using high-bandwidth electrical links, often through electrical switches (e.g., NVSwitches) or PCIe memory interconnects [27]. While these designs have served well for decades, they now face fundamental limits: the bandwidth demands of modern AI workloads and the sheer scale of distributed systems are pushing electrical interconnects beyond what they can practically deliver [12]. As the number of GPUs grows, these links become a bottleneck, creating a bandwidth wall alongside rising power consumption and heat dissipation [38]. The slowdown of Moore's Law for CMOS only compounds this challenge [3, 24]. As scale-up systems grow larger and more heterogeneous, the need for fundamentally more efficient, scalable, and low-power communication fabrics has never been greater.

In this context, silicon photonics offers a promising path forward, delivering orders-of-magnitude improvements in bandwidth density and energy efficiency compared to electrical interconnects [20, 39]. By using optical signals for chip-to-chip communication, silicon photonics can dramatically boost data throughput while reducing power draw. Yet despite its promise, much of this potential remains untapped in scale-up systems—primarily because current designs are rigid, lacking the adaptability needed to match dynamic workload patterns such as collective communication.

Photonic interconnects have traditionally been built as static circuit-switched topologies, tuned for specific, often predictable, communication patterns. But this rigidity is starting to crack: programmable silicon photonic fabrics are emerging [8], enabling dynamic reconfiguration of optical paths to adapt to shifting workload demands. These fabrics can establish direct, high-bandwidth optical links between endpoints, unlocking more efficient data exchange and synchronization across GPUs within a scale-up domain. Recent work has shown how to schedule circuit-switch configurations that align with communication patterns, using Birkhoff–von Neumann (BvN) decompositions or by solving optimization problems on the aggregate demand matrix [9, 19, 22, 28, 41]. Yet despite this progress, we know surprisingly little about how reconfiguration delays shape collective performance — or when it is worth reconfiguring at all.

In this paper, we make two simple yet striking observations. First, many collective communication algorithms naturally induce BvN decompositions: each algorithm step can be seen as a matching, and together these matchings form a convex combination of the aggregate demand. This connection has hovered in the literature for years, thanks to the inherently point-to-point, step-wise design of collective algorithms [7]. Second, this perspective bridges neatly to performance modeling: the classic α–β cost model for collectives emerges naturally when each step is viewed through the lens of maximum concurrent flow, capturing both network throughput and congestion. Together, these insights ground the familiar α–β model in physical topologies, generalizing collective completion time to account for real-world network constraints.

Leveraging these insights, we present a theoretical framework for optimizing circuit switching in adaptive

[1] "Scale-up" refers to networks within a single server or a single memory domain.

photonic interconnects. Our focus is the fundamental trade-off between reconfiguration delay and the performance gains enabled by dynamically matching the topology to the communication pattern. We formulate an optimization problem that captures this trade-off, allowing us to systematically decide when to reconfigure the interconnect and when to maintain a static topology, with the objective of minimizing collective completion time. This framework provides a principled way to design circuit-switching schedules that balance the benefits of reconfiguration against its costs, while explicitly accounting for both network throughput and the structure of collective communication.

Our preliminary results show that adaptive circuit switching can unlock substantial performance gains for collective communication — but only when used wisely. In regimes with high reconfiguration delays or small messages, naive per-step reconfiguration can add more latency than it saves; here, our framework shows when it is better to stay static. Conversely, when delays are low and message sizes are large, carefully chosen reconfigurations can fully tap the available photonic bandwidth, outperforming static designs by a wide margin. Most importantly, we uncover a practical middle ground where neither always reconfiguring nor always staying static is sufficient: this regime clearly requires optimized schedules that decide when reconfiguration is worth the cost and when it is not.

These results expose rich new questions at the intersection of theory and practice: how to design fast heuristics, develop collective algorithms that are reconfiguration-aware, and build photonic fabrics that can adapt on the fly. We outline a research agenda addressing these challenges at the end of this paper. We believe this line of work pushes us closer to interconnects where light truly bends to the collective will.

2 Background & Motivation

Collective operations, such as `AllReduce` and `Broadcast`, are foundational in distributed computing [7, 37]. These operations progress in structured stages with predictable communication patterns and data dependencies. Yet static interconnects — even when combined with topology-aware algorithms — often fail to fully leverage this structure.

Limits of topology-aware collectives: Prior work has developed collective algorithms tailored for specific static topologies (e.g., torus, DGX), guided by the classic α–β cost model [6, 23, 33]. While these designs improve efficiency for fixed topologies, they fundamentally inherit the rigidity of static networks. For example, multi-step collectives like halving/doubling for `AllReduce` [30] require repeated pairwise exchanges, but a fixed topology forces some pairs to traverse longer or congested paths, increasing both latency and bandwidth requirements [32]. Worse, static networks must provision for worst-case demand, leading to underutilization when traffic is sparse or staged. This is often tackled by pipelining and mirroring the collectives for multi-ported networks [32], but this approach only partially mitigates the inefficiencies of static designs.

Throughput modeling and BvN decompositions: The maximum concurrent flow [34] framework has long been used to analyze network throughput and congestion [1, 2, 16, 25, 36]. It connects naturally to Birkhoff–von Neumann (BvN) decompositions, which express an aggregate traffic matrix as a convex combination of matchings [25]. Many works use an aggregate traffic matrix as an input to synthesize circuit-switching schedules for demand-aware networks [19, 28, 41]. Yet, traffic matrices, and BvN decompositions assume all traffic is available simultaneously — which is not true for collectives that generate and exchange data in a strict sequence. This mismatch means static traffic matrix decompositions alone cannot capture the temporal dependencies that real collectives impose.

Programmable but costly reconfiguration: Reconfigurable photonic fabrics hold the promise of tailoring the network topology to each step of a collective communication pattern, reducing congestion and boosting throughput [8, 20]. But this flexibility comes at a price: practical designs introduce non-trivial reconfiguration delays, which can easily wipe out any performance gains if applied without care [8]. Yet much of the existing work sidesteps this trade-off altogether, either assuming that reconfiguration overheads are negligible or simply falling back to static networks when they are not.

Together, these gaps motivate a more principled perspective — one that bridges the staged structure of collective algorithms, the limits of network throughput, and the real costs of reconfiguration. In the next section, we present a framework that connects BvN decompositions, maximum concurrent flow, and the α–β model, providing fresh insight into when and how adaptive photonic interconnects can truly pay off.

3 Theory for Adaptive Scaleup Domains

We present a theoretical framework for optimizing circuit switching in adaptive photonic interconnects, focusing on the trade-off between reconfiguration delay and performance gains of adaptive topologies.

3.1 Architecture and Assumptions

We consider a scale-up domain with n GPUS, each equipped with an electrical-to-optical tranceiver (e.g., TeraPhy [39]) with bandwidth b. All the n tranceivers are connected to a photonic interconnect with n ports. Light enters through these ports and can be routed through the interconnect that establishes direct optical paths between pairs of ports — essentially connecting two GPUs. The interconnect is programmable i.e., reconfigurable, allowing it to dynamically reconfigure the optical paths on-demand [20]. Alternatively, if the tranceivers are capable of tuning the wavelength of the light they emit,

a passive wavelength switching photonic interconnect can establish direct paths between pairs of ports, without requiring a cental controller. In either designs, we consider a reconfiguration delay of α_r for the interconnect to reconfigure the optical paths. We note that several technologies today incur a reconfiguration delay that is dependent on the number of ports involved in the reconfiguration [8]. For simplicity, we assume the reconfiguration delay α_r is constant for all reconfigurations (e.g., for the total port count), but our framework can be extended to account for this variability. Importantly, we assume that all GPUs are within a single scale-up domain, and thus have fast access to a shared memory (e.g., DGX H100 server [14]). This allows the GPUs to rapidly synchronize e.g., using a barrier, before a particular step during a collective, so that they can perform the reconfiguration (if required) synchronously and proceed to the next step. We currently focus on collective communication across all n GPUs. A subset of GPUs can also be considered, and the interconnect simply reconfigures (if required) only the involved ports.

3.2 BvN, Concurrent Flow, and the α–β Cost Model

We model collective communication performed by an algorithm across n GPUs as a sequence of s communication steps. In each step i, a fixed amount of data m_i is exchanged between pairs of GPUs according to a matching, represented by a permutation matrix $\mathcal{M}_i$. Each entry $\mathcal{M}_i(j,k) = 1$ indicates that GPU j sends data to GPU k during step i; all other entries are zero. The full collective communication algorithm can thus be described as a sequence $\langle \mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_s \rangle$ of permutations, with associated data volumes $\langle m_1, m_2, \ldots, m_s \rangle$.

The total communication across all steps can be captured by the *aggregate demand matrix* $\mathcal{M}$, where each entry $\mathcal{M}(j,k)$ denotes the total volume of data sent from GPU j to GPU k over the entire operation. This matrix is simply the sum of all stepwise permutation matrices, weighted by their data volumes:

$$\mathcal{M} = m_1 \cdot \mathcal{M}_1 + m_2 \cdot \mathcal{M}_2 + \ldots + m_s \cdot \mathcal{M}_s. \quad (1)$$

This expression is, by definition, a *Birkhoff–von Neumann (BvN) decomposition* of $\mathcal{M}$: a convex combination of permutation matrices. In this view, the steps of the collective algorithm naturally correspond to matchings in the decomposition, with each m_i denoting the volume of data transferred during step i.

Observation 1 (Collectives Induce BvN Decompositions)**.** *Collective communication algorithms that proceed via a sequence of matchings naturally induce a BvN decomposition of their aggregate demand matrix.*

The reverse, however, does not hold: not all BvN decompositions correspond to valid collective algorithms. More critically, BvN decompositions fail to capture the *temporal structure* inherent in collective communication. In real algorithms, the ordering of permutations matters—steps cannot be arbitrarily rearranged. The data exchanged in step i is often *generated as a result of* the computation or communication in step $i-1$, creating a strict sequence of dependencies.

These temporal and data-flow constraints underscore an important limitation: the aggregate demand matrix, while useful in demand-aware network design [11, 22, 28], assumes all traffic is simultaneously available between source-destination pairs. This assumption breaks down in collectives, where data availability is staged and communication steps must follow a strict temporal order. As a result, designing interconnects for collective communication requires reasoning beyond static demand matrices and BvN decompositions alone.

Yet, the BvN decompositions induced by collective algorithms, as we show next, reveal a useful connection to both network throughput and the classic α–β cost model.

Consider a graph $G = (V,E)$, where V is the set of n GPUs and E represents the photonic links between them. The total completion time of the collective communication algorithm can be expressed as:

$$t_c = DCT(m_1 \cdot \mathcal{M}_1) + DCT(m_2 \cdot \mathcal{M}_2) + \ldots + DCT(m_s \cdot \mathcal{M}_s), \quad (2)$$

where $DCT(m_i \cdot \mathcal{M}_i)$ denotes the *demand completion time* of step i, corresponding to a data volume m_i and communication pattern $\mathcal{M}_i$.

The value of $DCT(m_i \cdot \mathcal{M}_i)$ depends on the structure and capacity of the underlying graph G. Specifically, we define the *maximum concurrent flow* $\theta(G, \mathcal{M}_i)$ as the largest fraction of the permutation demand matrix $\mathcal{M}_i$ that can be routed simultaneously without exceeding any link capacities. Intuitively, $\theta(G, \mathcal{M}_i)$ quantifies the achievable throughput for that step's communication pattern. This implies that the demand completion time can be written as:

$$DCT(m_i \cdot \mathcal{M}_i) = \frac{m_i}{b} \cdot \frac{1}{\theta(G, \mathcal{M}_i)},$$

where b is the link bandwidth. Here, $\frac{m_i}{b}$ represents the ideal transmission time assuming full throughput, while the factor $\frac{1}{\theta(G,\mathcal{M}_i)}$ accounts for congestion. By definition of the maximum concurrent flow, the effective bandwidth available for this permutation is $b \cdot \theta(G, \mathcal{M}_i)$, so the actual transmission time scales inversely with the achievable throughput.

In addition, each communication step i incurs a fixed overhead α, which captures startup latencies such as data preparation; latency $\delta \cdot \ell_i$ incurred due to per-link propagation delay δ and the path length ℓ_i of the most congested link in the corresponding step, which is often neglected and absorbed into the constant α. If the network offers bandwidth b per node, we define $\beta = \frac{1}{b}$. The demand completion time

for step i can then be written as:

$$DCT(m_i \cdot \mathcal{M}_i) = \underbrace{\alpha + \delta \cdot \ell_i}_{\text{latency factor}} + \overbrace{\beta}^{\text{bandwidth factor}} \cdot m_i \cdot \underbrace{\frac{1}{\theta(G, \mathcal{M}_i)}}_{\text{congestion factor}} \tag{3}$$

The total completion time of the collective for all the s steps can now be expressed as:

$$\begin{aligned} t_c &= \sum_{i=1}^{s} DCT(m_i \cdot \mathcal{M}_i) = \sum_{i=1}^{s} \left(\alpha + \delta \cdot \ell_i + \beta \cdot m_i \cdot \frac{1}{\theta(G, \mathcal{M}_i)} \right) \\ &= s \cdot \alpha + \sum_{i=1}^{s} \delta \cdot \ell_i + \beta \cdot \sum_{i=1}^{s} m_i \cdot \frac{1}{\theta(G, \mathcal{M}_i)} \end{aligned} \tag{4}$$

Observation 2 (Collective Completion Time as α–β Cost)**.** *The classic α–β cost model for collective communication emerges naturally when we express collective completion time in terms of latency factor α, bandwidth factor β, and importantly, propagation delay δ and congestion factor which is the inverse of concurrent flow θ. This formulation grounds the cost model in network throughput and reveals its dependence on both the underlying topology and the structure of the collective.*

While the α–β model is widely used in practice, network throughput, propagation delays, and congestion are rarely made explicit in its formulation. A few exceptions relate congestion to communication distance or the number of the messages on a link in structured topologies [7, 31, 32], but these are often limited to specific patterns or architectures assuming unsplittable flow. On the algorithm synthesis side, Liu et al. [23] recently extended the collective cost model using a multi-commodity flow formulation to capture capacity constraints and routing flexibility for the demand matrix represented by the overall collective operation. Although prior work has implicitly explored aspects of this connection, our formulation explicitly links the α–β model to network throughput via concurrent flow. This yields a more comprehensive understanding of performance that accounts for both communication structure and network topology. Notably, our formulation applies to arbitrary topologies, making it broadly applicable beyond structured or hierarchical networks.

3.3 Optimization Framework for Circuit Switching

The key insight from our observations is that the completion time of a collective communication algorithm is fundamentally tied to the path lengths, congestion and throughput of the underlying topology in each step. This is especially relevant for circuit switching in photonic interconnects: congestion and path lengths can be reduced to 1 — i.e., full throughput — by establishing direct, high-bandwidth optical paths that exactly match the communication pattern $\mathcal{M}_i$ for each step i.

However, realizing these direct paths requires reconfiguring the interconnect, which incurs a reconfiguration delay α_r. This creates a clear trade-off: reconfiguring reduces congestion and improves throughput but adds latency, while maintaining a static topology avoids reconfiguration costs but may suffer higher congestion.

This tension opens up an opportunity for optimization: how should we schedule interconnect reconfigurations to minimize the total completion time for any given collective? For example, one might choose to maintain a static topology to avoid reconfiguration overhead but pay persistent congestion costs, or reconfigure before every step to eliminate congestion while incurring the maximum reconfiguration penalty. An effective circuit switching schedule must strike a balance, reconfiguring only in steps when the throughput gain outweighs the cost.

Given any collective communication algorithm with s steps, each with a communication pattern $\mathcal{M}_i$ and data volume m_i, we can formulate the following optimization problem. We define two binary variables x_i and z_i as follows:

$$x_i = \begin{cases} 1 & \text{base topology } G \\ 0 & \text{matched topology } \mathcal{M}_i \textit{ for step } i \end{cases} \tag{5}$$

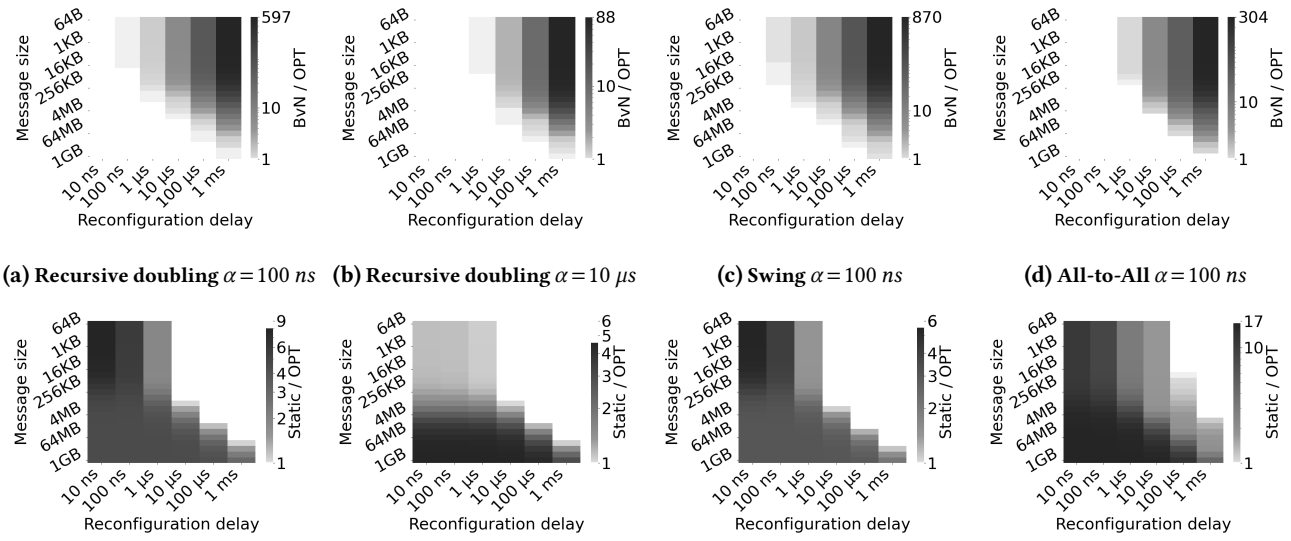
$$z_i = \begin{cases} 1 & \text{if step } i-1 \text{ and } i \text{ are both base topologies } G \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

Here, x_i defines the circuit switching schedule, i.e., whether each step uses the base topology G or a topology that perfectly matches the communication pattern $\mathcal{M}_i$ for step i of the collective. The variable z_i defines whether the interconnect incurs any reconfiguration delay between step $i-1$ and i.

The optimization problem can now be formulated as:

$$\begin{aligned} \min \quad & \delta \cdot \sum_{i=1}^{s} \left(\overbrace{x_i \cdot \ell_i}^{\substack{\text{propagation delay} \\ \text{w/o reconf.}}} \overbrace{+(1-x_i) \cdot 1}^{\substack{\text{direct} \\ \text{with reconf.}}} \right) + \sum_{1}^{s} \overbrace{(1-z_i) \cdot \alpha_r}^{\text{reconf. delay}} \\ & + s \cdot \alpha + \beta \cdot \sum_{i=1}^{s} m_i \cdot \Big(\underbrace{x_i \cdot \frac{1}{\theta(G, \mathcal{M}_i)}}_{\substack{\text{congestion} \\ \text{w/o reconf.}}} + \underbrace{(1-x_i) \cdot 1}_{\substack{\text{no congestion} \\ \text{with reconf.}}} \Big) \\ \text{subject to} \quad & z_i \geq x_i + x_{i-1} - 1 \\ & z_i \leq x_i; \quad z_i \leq x_{i-1} \quad \forall i \in [1, s],\ x_0 = 1 \\ \text{Variables} \quad & x_i \in \{0,1\}; \quad z_i \in \{0,1\} \end{aligned} \tag{7}$$

Our objective is to minimize the total completion time of the collective communication algorithm, which consists of four components: $(\delta \cdot \ell_i)$ the propagation delay as function of path lengths, (α) the fixed latency factor, (α_r) the total reconfiguration delay incurred by the interconnect, and $(\frac{1}{\theta})$ the congestion factor across all steps. The congestion



(a) Recursive doubling $\alpha = 100\ ns$ **(b) Recursive doubling $\alpha = 10\ \mu s$** **(c) Swing $\alpha = 100\ ns$** **(d) All-to-All $\alpha = 100\ ns$**

(e) Recursive doubling $\alpha = 100\ ns$ **(f) Recursive doubling $\alpha = 10\ \mu s$** **(g) Swing $\alpha = 100\ ns$** **(h) All-to-All $\alpha = 100\ ns$**

Figure 1: Heatmaps showing the speedup in collective completion times achieved by our optimized schedules, compared to BvN-based schedules (top row) and a static ring topology (bottom row).

and propagation delay depend on whether we choose to reconfigure the interconnect to match the communication pattern $\mathcal{M}_i$ or maintain the base topology G. The constraints ensure that z_i correctly captures whether a reconfiguration occurs between steps, and all variables are binary.

Overall, this formulation is a mixed integer program (0–1 ILP), which is NP-hard in the general case [17]. Interestingly, our model has a special sequential structure: the variables x_i (interconnect state) and z_i (reconfiguration event) depend only on the previous step. This structure admits an efficient dynamic programming solution and is polynomial-time solvable due to the principle of optimality [4].

This framework captures the fundamental trade-off between reconfiguration delay and congestion in adaptive photonic interconnects. It provides a systematic way to optimize circuit switching schedules for collective communication, balancing the benefits of reconfiguration against its costs. Notably, the optimization is aware of both the data volume in each step and the underlying network throughput. Furthermore, the formulation supports any base topology G and applies to any collective communication algorithm (including custom ones) that can be expressed as a sequence of matchings, or even a sequence of such collective communication operations e.g., All-to-All after an AllReduce operation. Our formulation can even be extended to account for a fixed pool of base topologies instead of a single base topology G that we current use e.g., using multiple co-prime rings as base topologies or a union of such rings for higher degree networks [41]. Optimizing the base topologies opens further opportunities for performance gains.

3.4 So What is the Δ After All? Reconfigure or Not?

Our focus so far has been on the underlying theoretical problem of optimizing circuit switching for collective communication. But the central question remains: *what performance gain can we actually expect from programmable silicon photonic interconnects?* In other words, for what range of reconfiguration delays does a *programmable* interconnect yield meaningful speedup for collective operations in scale-up domains?

To explore this question, we conduct preliminary evaluations using a flow-level simulator that implements the optimization framework described in § 3.3. We model a scale-up system with $n = 64$ GPUs, each equipped with a single link to a reconfigurable photonic interconnect as introduced in §3.1. We set the link bandwidth to 800 Gbps, propagation delay δ to 100ns [32], and vary the fixed per-step latency α, the reconfiguration delay α_r, and the message size. We run the AllReduce collective using recursive doubling and Swing algorithms [30, 32] (which are bandwidth-optimal); and All-to-All (transpose) collective. Due to space constraints, we omit the combinations of α, α_r, and bandwidth, but similar trends hold throughout the full parameter space. Since each GPU has a single fat link, we use a ring as the base topology G — a common choice for scale-up photonic interconnects. While our optimization framework is especially valuable for degree > 2 networks, we use this simple case to clearly illustrate the main trade-offs. We compare two approaches: (1) a static ring topology, and (2) a reconfigurable interconnect that follows BvN schedules matched to the communication pattern (see §3.2). We report speedup in terms of the completion time of the collective achieved by our optimized schedules (OPT) compared to these alternatives.

Figure 1 summarizes the results. Figures 1a—1d show the speedup relative to BvN schedules, while Figures 1e—1h show the speedup relative to the static ring. Each column (x-axis) in the heatmaps corresponds to a different value of reconfiguration delay α_r, and each row (y-axis) corresponds to a different message size. The color indicates the speedup achieved by our optimized schedule, with darker shades representing higher speedup and no color (or white) indicates speed up of 1.

Overall, we see that our framework captures two distinct regimes: significant performance gains (up to orders of magnitude) over BvN schedules appear when reconfiguration delay is high or message sizes are small, where naive per-step reconfiguration would otherwise incur excessive latency. In comparison to a static ring topology, we observe substantial speedup when reconfiguration delay is low and message sizes are large, where our optimized schedule fully exploits the available bandwidth. Interestingly, Figure 2 shows that there is also a transitional regime — visible as the diagonal region — where our optimized schedules outperform both static and naive BvN approaches by adaptively deciding when to reconfigure and when not to. This illustrates precisely *when* adaptive photonic interconnects should reconfigure and when they should not.

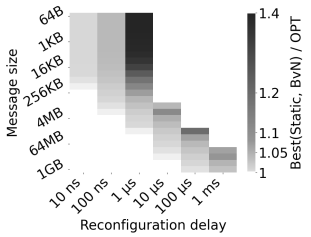


Figure 2: Our optimized schedules can significantly speed up collective communication even compared to the best of both worlds — BvN schedules and a static ring topology.

4 Research Agenda and Future Outlook

We see many opportunities for performance optimization from a theoretical perspective, along with practical challenges that must be addressed before adaptive photonic interconnects can be fully realized in scale-up domains. We outline a research agenda spanning algorithm design, and systems integration.

Fast heuristics for adaptive photonic interconnects: As scale-up domains grow, fast heuristics for optimizing circuit-switching schedules become paramount. While our framework offers insight into potential gains, practical implementations need algorithms that adapt quickly to arbitrary collectives. For example, threshold-based heuristics could switch between a static topology and a BvN-based schedule depending on when gains outweigh reconfiguration costs [13]. Balancing near-optimality with computational efficiency will be crucial for real adoption.

Simplifying the congestion factor in the cost model: Our framework relies on the maximum concurrent flow $\theta(G,\mathcal{M}_i)$ to capture congestion, but computing this exactly can be expensive, particularly for large topologies. Future work could explore approximations or simpler proxies that retain accuracy but reduce overhead. For example, an upper bound on throughput per permutation pattern based on graph degree could reduce the congestion factor to a function of maximum node degree and the number of communicating GPUs. Such simplifications could make scheduling practical at runtime while preserving useful performance insights.

Deeper understanding of the propagation delays: Our formulation in §3.2, indicates that the completion time of a collective communication algorithm is influenced by the path lengths and congestion. For AllReduce algorithms, this implies that the ring algorithm is optimal even for short messages if the propagation delays are high. Naturally, recursive doubling [30], or other algorithms like Swing [32] that finish in fewer steps become more attractive for reconfigurable interconnects, than for static interconnects. We leave it for future work, to design fast heuristics for AllReduce operations.

Routing challenges: Reconfigurable interconnects naturally introduce dynamic routing challenges. While a topology that matches a collective step's pattern allows simple one-hop routing, practical schedules may include intermediate topologies that balance reconfiguration cost against performance. This requires routing algorithms that adapt quickly while maintaining high throughput and low latency. Exploring lightweight, topology-aware routing techniques for dynamic configurations is an important direction.

Tackling variable reconfiguration delays: Our formulation assumes a constant reconfiguration delay α_r, but in practice, this may vary with the number of ports or the specific operation. We plan to extend our framework to account for variable delays by modeling them as a function of port count or reconfiguration complexity. This would enable more accurate scheduling that adapts to the interconnect's characteristics.

Overlapping reconfiguration with computation: Many collectives offer potential to overlap reconfiguration with computation, letting GPUs prepare data while the interconnect reconfigures. We plan to explore how to schedule these overlaps to minimize total completion time, by modeling computation phases as part of the optimization.

Many interesting questions remain open, including extending our model to multi-ported collectives where each step is not a single permutation but a union of multiple permutations; identifying optimal sets of base topologies; and addressing practical aspects such as synchronization.

We envision a future where scale-up GPU systems seamlessly harness the power of reconfigurable photonic fabrics to

break through today's bandwidth and energy walls. By bridging theory and practice — from fast scheduling heuristics to topology-aware routing and reconfiguration-aware collectives — we can unlock the full potential of adaptive photonic interconnects. Realizing this vision will require close collaboration across systems, networking, and photonics communities, but the payoff is compelling: a new class of datacenter and HPC architectures where communication is entirely in photonic domain, and light truly bends to the collective will.

References

[1] Vamsi Addanki, Chen Avin, and Stefan Schmid. Mars: Near-optimal throughput with shallow buffers in reconfigurable datacenter networks. *Proc. ACM Meas. Anal. Comput. Syst.*, 7(1), March 2023.

[2] Daniel Amir, Tegan Wilson, Vishal Shrivastav, Hakim Weatherspoon, Robert Kleinberg, and Rachit Agarwal. Optimal oblivious reconfigurable networks. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2022, page 1339–1352, New York, NY, USA, 2022. Association for Computing Machinery.

[3] Hitesh Ballani, Paolo Costa, Raphael Behrendt, Daniel Cletheroe, Istvan Haller, Krzysztof Jozwik, Fotini Karinou, Sophie Lange, Kai Shi, Benn Thomsen, and Hugh Williams. Sirius: A flat datacenter network with nanosecond optical switching. In *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '20, page 782–797, New York, NY, USA, 2020. Association for Computing Machinery.

[4] Dimitri Bertsekas. *Dynamic programming and optimal control: Volume I*, volume 4. Athena scientific, 2012.

[5] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[6] Zixian Cai, Zhengyang Liu, Saeed Maleki, Madanlal Musuvathi, Todd Mytkowicz, Jacob Nelson, and Olli Saarikivi. Synthesizing optimal collective algorithms. In *Proceedings of the 26th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, PPoPP '21, page 62–75, New York, NY, USA, 2021. Association for Computing Machinery.

[7] Ernie Chan, Marcel Heimlich, Avi Purkayastha, and Robert van de Geijn. Collective communication: theory, practice, and experience. *Concurrency and Computation: Practice and Experience*, 19(13):1749–1783, 2007.

[8] Eric Ding and Rachee Singh. Pipswitch: A circuit switch using programmable integrated photonics. In *Optical Fiber Communication Conference (OFC) 2025*, page W2A.41. Optica Publishing Group, 2025.

[9] Nathan Farrington, George Porter, Sivasankar Radhakrishnan, Hamid Hajabdolali Bazzaz, Vikram Subramanya, Yeshaiahu Fainman, George Papen, and Amin Vahdat. Helios: a hybrid electrical/optical switch architecture for modular data centers. In *Proceedings of the ACM SIGCOMM 2010 Conference*, SIGCOMM '10, page 339–350, New York, NY, USA, 2010. Association for Computing Machinery.

[10] Adithya Gangidi, Rui Miao, Shengbao Zheng, Sai Jayesh Bondu, Guilherme Goes, Hany Morsy, Rohit Puri, Mohammad Riftadi, Ashmitha Jeevaraj Shetty, Jingyi Yang, Shuqiang Zhang, Mikel Jimenez Fernandez, Shashidhar Gandham, and Hongyi Zeng. Rdma over ethernet for distributed training at meta scale. In *Proceedings of the ACM SIGCOMM 2024 Conference*, ACM SIGCOMM '24, page 57–70, New York, NY, USA, 2024. Association for Computing Machinery.

[11] Monia Ghobadi, Ratul Mahajan, Amar Phanishayee, Nikhil Devanur, Janardhan Kulkarni, Gireeja Ranade, Pierre-Alexandre Blanche, Houman Rastegarfar, Madeleine Glick, and Daniel Kilper. Projector: Agile reconfigurable data center interconnect. In *Proceedings of the 2016 ACM SIGCOMM Conference*, SIGCOMM '16, page 216–229, New York, NY, USA, 2016. Association for Computing Machinery.

[12] Amir Gholami, Zhewei Yao, Sehoon Kim, Coleman Hooper, Michael W. Mahoney, and Kurt Keutzer. Ai and memory wall. *IEEE Micro*, 44(3):33–39, 2024.

[13] Sarah-Michelle Hammar, Stefan Schmid, Rachee Singh, and Vamsi Addanki. Short-circuiting rings for low-latency allreduce. *arXiv preprint*, 2025.

[14] Alexander Ishii and Ryan Wells. The Nvlink-Network Switch: Nvidia's Switch Chip for High Communication-Bandwidth Superpods . In *2022 IEEE Hot Chips 34 Symposium (HCS)*, pages 1–23, Los Alamitos, CA, USA, August 2022. IEEE Computer Society.

[15] Norm Jouppi, George Kurian, Sheng Li, Peter Ma, Rahul Nagarajan, Lifeng Nai, Nishant Patil, Suvinay Subramanian, Andy Swing, Brian Towles, Clifford Young, Xiang Zhou, Zongwei Zhou, and David A Patterson. Tpu v4: An optically reconfigurable supercomputer for machine learning with hardware support for embeddings. In *Proceedings of the 50th Annual International Symposium on Computer Architecture*, ISCA '23, New York, NY, USA, 2023. Association for Computing Machinery.

[16] Sangeetha Abdu Jyothi, Ankit Singla, P Brighten Godfrey, and Alexandra Kolla. Measuring and understanding throughput of network topologies. In *SC'16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 761–772. IEEE, 2016.

[17] Richard M Karp. Reducibility among combinatorial problems. In *50 Years of Integer Programming 1958-2008: from the Early Years to the State-of-the-Art*, pages 219–241. Springer, 2009.

[18] Tarannum Khan, Saeed Rashidi, Srinivas Sridharan, Pallavi Shurpali, Aditya Akella, and Tushar Krishna. Impact of roce congestion control policies on distributed training of dnns. In *2022 IEEE Symposium on High-Performance Interconnects (HOTI)*, pages 39–48, 2022.

[19] Mehrdad Khani, Manya Ghobadi, Mohammad Alizadeh, Ziyi Zhu, Madeleine Glick, Keren Bergman, Amin Vahdat, Benjamin Klenk, and Eiman Ebrahimi. Sip-ml: high-bandwidth optical network interconnects for machine learning training. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, SIGCOMM '21, page 657–675, New York, NY, USA, 2021. Association for Computing Machinery.

[20] Abhishek Vijaya Kumar, Arjun Devraj, Darius Bunandar, and Rachee Singh. A case for server-scale photonic connectivity. In *Proceedings of the 23rd ACM Workshop on Hot Topics in Networks*, HotNets '24, page 290–299, New York, NY, USA, 2024. Association for Computing Machinery.

[21] Wenxue Li, Xiangzhou Liu, Yuxuan Li, Yilun Jin, Han Tian, Zhizhen Zhong, Guyue Liu, Ying Zhang, and Kai Chen. Understanding communication characteristics of distributed training. In *Proceedings of the 8th Asia-Pacific Workshop on Networking*, APNet '24, page 1–8, New York, NY, USA, 2024. Association for Computing Machinery.

[22] He Liu, Matthew K. Mukerjee, Conglong Li, Nicolas Feltman, George Papen, Stefan Savage, Srinivasan Seshan, Geoffrey M. Voelker, David G. Andersen, Michael Kaminsky, George Porter, and Alex C. Snoeren. Scheduling techniques for hybrid circuit/packet networks. In *Proceedings of the 11th ACM Conference on Emerging Networking Experiments*

and Technologies, CoNEXT '15, New York, NY, USA, 2015. Association for Computing Machinery.

[23] Xuting Liu, Behnaz Arzani, Siva Kesava Reddy Kakarla, Liangyu Zhao, Vincent Liu, Miguel Castro, Srikanth Kandula, and Luke Marshall. Rethinking machine learning collective communication as a multi-commodity flow problem. In *Proceedings of the ACM SIGCOMM 2024 Conference*, ACM SIGCOMM '24, page 16–37, New York, NY, USA, 2024. Association for Computing Machinery.

[24] William M. Mellette, Alex Forencich, Rukshani Athapathu, Alex C. Snoeren, George Papen, and George Porter. Realizing rotornet: Toward practical microsecond scale optical networking. In *Proceedings of the ACM SIGCOMM 2024 Conference*, ACM SIGCOMM '24, page 392–414, New York, NY, USA, 2024. Association for Computing Machinery.

[25] Pooria Namyar, Sucha Supittayapornpong, Mingyang Zhang, Minlan Yu, and Ramesh Govindan. A throughput-centric view of the performance of datacenter topologies. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, SIGCOMM '21, page 349–369, New York, NY, USA, 2021. Association for Computing Machinery.

[26] Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, Amar Phanishayee, and Matei Zaharia. Efficient large-scale language model training on gpu clusters using megatron-lm. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '21, New York, NY, USA, 2021. Association for Computing Machinery.

[27] DGX NVIDIA. Superpod: Next generation scalable infrastructure for ai leadership. *Retrieved from: https://docs.nvidia.com/https:/docs.nvidia.com/dgx-superpod-reference-architecture-dgx-h100.pdf*, 2023.

[28] George Porter, Richard Strong, Nathan Farrington, Alex Forencich, Pang Chen-Sun, Tajana Rosing, Yeshaiahu Fainman, George Papen, and Amin Vahdat. Integrating microsecond circuit switching into the data center. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, SIGCOMM '13, page 447–458, New York, NY, USA, 2013. Association for Computing Machinery.

[29] 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. Alibaba hpn: A data center network for large language model training. In *Proceedings of the ACM SIGCOMM 2024 Conference*, ACM SIGCOMM '24, page 691–706, New York, NY, USA, 2024. Association for Computing Machinery.

[30] Rolf Rabenseifner. Optimization of collective reduction operations. In Marian Bubak, Geert Dick van Albada, Peter M. A. Sloot, and Jack Dongarra, editors, *Computational Science - ICCS 2004*, pages 1–9, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[31] Paul Sack and William Gropp. Collective algorithms for multiported torus networks. *ACM Trans. Parallel Comput.*, 1(2), February 2015.

[32] Daniele De Sensi, Tommaso Bonato, David Saam, and Torsten Hoefler. Swing: Short-cutting rings for higher bandwidth allreduce. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, pages 1445–1462, Santa Clara, CA, April 2024. USENIX Association.

[33] Aashaka Shah, Vijay Chidambaram, Meghan Cowan, Saeed Maleki, Madan Musuvathi, Todd Mytkowicz, Jacob Nelson, Olli Saarikivi, and Rachee Singh. TACCL: Guiding collective algorithm synthesis using communication sketches. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pages 593–612, Boston, MA, April 2023. USENIX Association.

[34] Farhad Shahrokhi and D. W. Matula. The maximum concurrent flow problem. *J. ACM*, 37(2):318–334, April 1990.

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

[36] Ankit Singla, P. Brighten Godfrey, and Alexandra Kolla. High throughput data center topology design. In *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*, pages 29–41, Seattle, WA, April 2014. USENIX Association.

[37] Rajeev Thakur, Rolf Rabenseifner, and William Gropp. Optimization of collective communication operations in mpich. *The International Journal of High Performance Computing Applications*, 19(1):49–66, 2005.

[38] Arya Tschand, Arun Tejusve Raghunath Rajan, Sachin Idgunji, Anirban Ghosh, Jeremy Holleman, Csaba Kiraly, Pawan Ambalkar, Ritika Borkar, Ramesh Chukka, Trevor Cockrell, Oliver Curtis, Grigori Fursin, Miro Hodak, Hiwot Kassa, Anton Lokhmotov, Dejan Miskovic, Yuechao Pan, Manu Prasad Manmathan, Liz Raymond, Tom St. John, Arjun Suresh, Rowan Taubitz, Sean Zhan, Scott Wasson, David Kanter, and Vijay Janapa Reddi. Mlperf power: Benchmarking the energy efficiency of machine learning systems from μwatts to mwatts for sustainable ai. In *2025 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 1201–1216, 2025.

[39] Mark Wade, Erik Anderson, Shahab Ardalan, Pavan Bhargava, Sidney Buchbinder, Michael L. Davenport, John Fini, Haiwei Lu, Chen Li, Roy Meade, Chandru Ramamurthy, Michael Rust, Forrest Sedgwick, Vladimir Stojanovic, Derek Van Orden, Chong Zhang, Chen Sun, Sergey Y. Shumarayev, Conor O'Keeffe, Tim T. Hoang, David Kehlet, Ravi V. Mahajan, Matthew T. Guzy, Allen Chan, and Tina Tran. Teraphy: A chiplet technology for low-power, high-bandwidth in-package optical i/o. *IEEE Micro*, 40(2):63–71, 2020.

[40] Weiyang Wang, Manya Ghobadi, Kayvon Shakeri, Ying Zhang, and Naader Hasani. Rail-only: A low-cost high-performance network for training llms with trillion parameters. In *2024 IEEE Symposium on High-Performance Interconnects (HOTI)*, pages 1–10, 2024.

[41] Weiyang Wang, Moein Khazraee, Zhizhen Zhong, Manya Ghobadi, Zhihao Jia, Dheevatsa Mudigere, Ying Zhang, and Anthony Kewitsch. TopoOpt: Co-optimizing network topology and parallelization strategy for distributed training jobs. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pages 739–767, Boston, MA, April 2023. USENIX Association.

[42] Yazhou Zu, Alireza Ghaffarkhah, Hoang-Vu Dang, Brian Towles, Steven Hand, Safeen Huda, Adekunle Bello, Alexander Kolbasov, Arash Rezaei, Dayou Du, Steve Lacy, Hang Wang, Aaron Wisner, Chris Lewis, and Henri Bahini. Resiliency at scale: Managing Google's TPUv4 machine learning supercomputer. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, pages 761–774, Santa Clara, CA, April 2024. USENIX Association.