Programmable optical interconnects for server-scale multi-GPU systems

Paper #8, 12 pages body, 18 pages total

Abstract

We develop a collective communication framework, LUMORPH, for photonic server- and rack-scale multi-GPU interconnects. We inform the design of LUMORPH with an experimental understanding of the capabilities of a novel silicon photonic server-scale interconnect, LIGHTPATH. LUMORPH leverages the programmability of optical switches on LIGHTPATH to construct on-demand circuits between GPUs performing collective communication for distributed ML. Our experiments show that it takes 3.7μs to reconfigure optical switches on LIGHTPATH. This delay leads to high circuit set up costs, potentially hampering the performance of multi-GPU systems. Moreover, finding optimal collective communications schedules on a programmable photonic interconnect is computationally hard. Our key insight is that despite the heterogeneity of optical circuits underpinning inter-GPU links, the links themselves have homogeneous data transmission characteristics. LUMORPH leverages the homogeneous data transmission characteristics of optical circuits on LIGHTPATH to adapt known optimal collective communication algorithms to a programmable optical interconnect. LUMORPH constructs on-demand end-to-end optical circuits between GPUs participating in collective communication. Despite incurring optical reconfiguration delays, LUMORPH achieves 74% faster ALLREDUCE communication in rack-scale multi-GPU interconnects. This translates to a 1.7X speedup in the end-to-end training throughput of popular ML models.[1]

1 Introduction

Modern machine learning (ML) models have trillions of parameters, making them too large to fit in the memory of individual GPUs. This trend is necessitating distributed training and inference from ML models across several GPUs. For instance, large language models like GPT-4 are trained using thousands of GPUs in private datacenters [13]. The efficiency of distributed ML training depends on the performance of the inter-GPU interconnect. Inefficient inter-GPU communication leads to GPUs idling while waiting for data or parameters to arrive over the interconnect. In fact, recent work has shown that communication has already become a bottleneck in distributed ML training [2, 9–11, 14, 32].

Limitations of server-scale interconnects. Today, server-scale multi-GPU interconnects are *electrical packet-switched* [37, 40]. Packet-switching has been the defacto design choice for interconnects since it enables fine-grained statistical multiplexing — the ability to effectively share the network's resources (*e.g.,* link capacities, buffers on switches) among multiple flows [41]. Moreover, packet-switching is application-agnostic and can rapidly switch packets of an arbitrary set of applications with unknown communication patterns. However, recent work has shown performance limitations of packet-switching on electrical multi-GPU interconnects (*e.g.,* PCIe [37], NVlinks [40]) used by modern server-scale systems like Nvidia DGXes [38]. These electrical packet-switched interconnects suffer from queuing delays that reduce achievable bandwidth between chips [44].

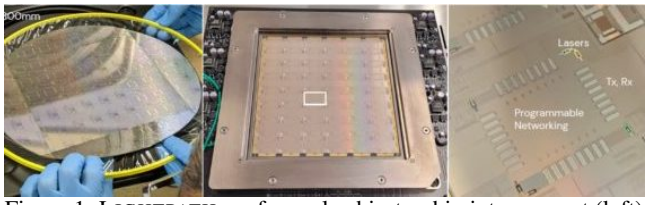


Figure 1: LIGHTPATH: wafer-scale chip-to-chip interconnect (left). Our 200mm x 200mm wafer-scale prototype wirebonded in a socket (center). Micrograph of one site on the wafer showing lasers, transceivers and a programmable optical switches (right).

Photonic server-scale interconnects. We propose a transition to *photonic circuit-switched* interconnects at server and rack-scale where GPUs communicate over dedicated circuits established on-demand with optical wavelengths. By design, circuit switching eliminates contention for bandwidth once circuits are established and provides predictable performance due to the absence of network queueing delays — a property essential for performance-sensitive ML workloads. To achieve our vision, we leverage recent advances in silicon photonics that have made server-scale photonic interconnect hardware commercially viable [22]. Figure 1 shows a commercial wafer-scale photonic interconnect, LIGHTPATH. Chips (*e.g.,* GPUs, CPUs) are bonded to *tiles* on the interconnect, forming a grid (Figure 2c). Each tile on the interconnect grid has silicon-based photonic components like lasers, waveguides, switches and photodetectors (Figure 2) to generate, transmit, switch and receive optical signals between chips.

However, achieving on-demand photonic circuits on multi-GPU optical interconnects is challenging in practice. First, silicon photonic switches are slow to reconfigure, requiring several microseconds to establish optical circuits between chips (§2). This delay introduces high circuit setup costs, potentially reducing the efficiency of resulting multi-chip systems. Second, finding optimal communication schedules for

[1] LUMORPH's code is at: https://lumorphnetwork.github.io/.

communication primitives used in distributed ML has been shown to be NP-Hard. In fact, recent work has grappled with developing collective communication schedules in static electrical multi-GPU interconnects even at the small scale of 32 interconnected GPUs [44]. This task is rendered more challenging in the context of programmable optical interconnects where a communication schedule must also arrive at the appropriate trade-off between optical reconfiguration delay and end-to-end interconnect performance.

Our contribution. To tackle these challenges, we develop LUMORPH, a system that implements circuit-switching on chip-to-chip photonic interconnects to accelerate distributed ML workloads. Distributed ML workloads use collective communication primitives to train and infer from large models. LUMORPH computes near-optimal *collective communication schedules* between GPUs such that data transfers in the schedule occur over on-demand photonic circuits between GPUs.

Key Insight. Our key insight is that ML training and inference applications that are prominent users of server-scale interconnects today [13, 23] have *predictable* patterns of communication. For instance, distributed training using model, data and expert parallelism requires GPUs to collectively *reduce* gradients over the interconnect using a collective communication primitive called ALLREDUCE [44]. Thus, we compensate for the slow creation of photonic tunnels between GPUs by leveraging the predictability in communication patterns of distributed ML workloads to proactively construct photonic tunnels. We make three main research contributions:

1. Hardware testbed. We develop a hardware prototype of LIGHTPATH, a silicon photonic chip-to-chip interconnect. Using the prototype, we derive the data transmission characteristics on LIGHTPATH. We show that LIGHTPATH can connect up to 32 GPUs where each GPU is bonded on a LIGHTPATH tile equipped with 16 lasers and photodiodes. A wavelength from a laser on can sustain up to 224 Gbps data rate. Finally, programming optical circuits between tiles on LIGHTPATH can take up to 3.7μs (§2, §7).

2. Faster collective communication. We use data transmission characteristics of LIGHTPATH to inform the design of LUMORPH. LUMORPH programs silicon photonic switches to dynamically construct end-to-end optical circuits between GPUs participating in collective communication. LUMORPH computes communication schedules that are up to 74% faster than state-of-the-art (§3, §5).

3. Faster end-to-end training. Optical circuits on LIGHTPATH have homogeneous data rates despite differing in path length. This homogeneity of inter-GPU links underpinned by optical circuits allows LUMORPH to adapt well-known optimal collective algorithms to programmable interconnects with up to 256 GPUs. Overall, LUMORPH improves the end-to-end training throughput of ML models by 1.7X (§4, §6).

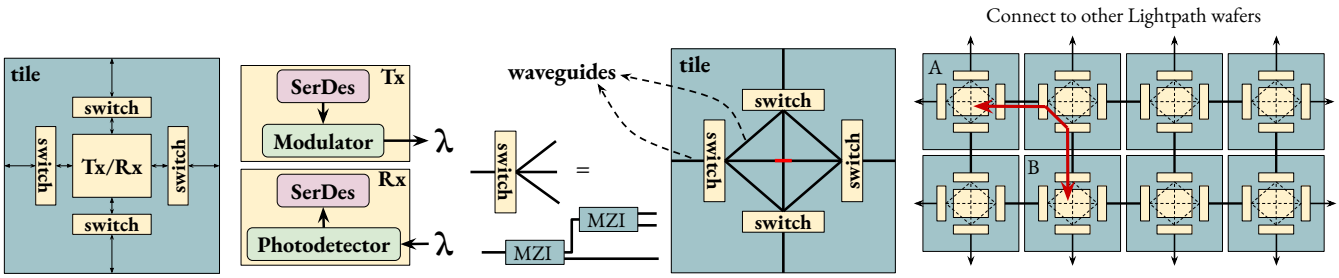
2 Server-scale optical interconnects

Light has been the physical carrier of data in long-haul and datacenter interconnects for several years [46, 47, 49, 60]. In contrast, modern chip-to-chip or *server-scale* interconnects are electrical (*e.g.,* PCIe [37], NVLinks [40]). Despite being more susceptible to noise than optical signals, electrical transmission has been the design choice for short range interconnects within servers. However, the need for high bandwidth and low power in multi-chip ecosystems has led to the development of commercial silicon photonic chip-to-chip interconnects [22, 34]. Silicon photonics is an emerging technology that uses silicon as the optical medium to develop photonic components. It is enabling large-scale manufacturing of optical communication components using semiconductor fabrication processes. Recently, silicon photonics has enabled the commercialization of server-scale chip-to-chip photonic interconnects [5, 6, 8, 26, 27, 31].

Rationale for server-scale photonic interconnects. There are several advantages to adopting photonic interconnects at the server-scale. First, photonic interconnects can support higher bandwidths compared to electrical interconnects by multiplexing data on multiple wavelengths of light, making them a good choice for high-bandwidth multi-GPU interconnects. Second, photonic server-scale interconnects consume a magnitude lower power than their electrical counterparts [38, 40]. Third, switching components of photonic server-scale interconnects are *programmable*. In this work, we leverage this programmability of photonic interconnects to adapt the interconnect to the characteristics of bandwidth-intensive networked workloads like distributed ML training.

Fast reconfiguration of server-scale interconnects. Recent work has studied reconfiguration of topology in datacenter-scale optical interconnects [29, 46, 56]. By design, this work reconfigures the datacenter optical interconnect infrequently by programming micro-electromechanical systems or MEMS-based optical circuit switches at the beginning of an entire ML training job. This separation between optical topology reconfiguration and traffic routing over the fixed topology after reconfiguration is called *topology engineering* [47] and *traffic engineering*, respectively. In this work, we show that high frequency reconfiguration of the interconnect in multi-GPU systems unlocks near-optimal latency and bandwidth for collective communication primitives that are the workhorses of distributed machine learning [44].

Key insight. We leverage the ability of novel photonic server-scale interconnects to reconfigure on-chip optical switches within $\approx 4\mu$s (§7). This allows us to dynamically adapt the interconnect topology to fit communication patterns at millisecond timescales, blurring the boundary between traffic vs. topology engineering. While faster than MEMS switches, microseconds spent to reprogram optical switches in server-scale interconnects are still significant. This poses an important trade-off between using high frequency adaptation of the in-



(a) Tx/Rx on a LIGHTPATH tile. (b) Optical switches on a LIGHTPATH tile. (c) $2X4$ tiles on a LIGHTPATH wafer.

Figure 2: Figure 2a shows one tile on the LIGHTPATH wafer. Each tile has a Tx/Rx block and four switches. The Tx/Rx block (zoomed in) modulates data from chips on to wavelengths of light (λ) in the transmit direction and detects bits from the modulated light in the receive direction. Figure 2b hides the Tx/Rx block to reveal the optical waveguide connectivity between switches and across tiles in LIGHTPATH. The red bar shows optical crossings. Each switch on the tile has a degree of $1X3$ and is constructed using Mach-Zehnder Interferometers (MZIs). Figure 2c shows $2X4$ tiles from a LIGHTPATH wafer. In addition to waveguides that connect switches on a tile to each other (shown in dashed lines), waveguides also connect tiles to each other (solid lines). Optical switches on tiles can be programmed to configure circuits like the one between tile *A* and *B* (in red). Optical fibers (solid lines with arrows) connect one LIGHTPATH wafer to others.

terconnect for efficient communication and the latency cost of reconfiguring the interconnect. This trade-off is central to our work. Our key insight is performance-sensitive workloads like distributed ML training have highly predictable patterns of communication. For instance, given the parallelization strategy used for training an ML model, the communication between GPUs training the model is highly predictable (§5), allowing us to proactively reconfigure photonic interconnects to match the communication schedule.

2.1 Components of server-scale optical interconnects

In this section, we introduce a novel server-scale photonic interconnect, LIGHTPATH. LIGHTPATH is a switchable photonic fabric in a hybrid Complementary Metal Oxide Semiconductor (CMOS) photonics process (Figure 1). A LIGHTPATH wafer consists of 32 *tiles*. ASICs (*e.g.,* CPU, GPU) are bonded on top of LIGHTPATH tiles. Thus, one LIGHTPATH interconnect can connect up to 32 chips. We evaluate the interconnect using a LIGHTPATH testbed in §7.

Modulators and Photodetectors. At the center of each LIGHTPATH tile is a Transmitter and a Receiver (Figure 2a). The optical transmitter (Tx) converts data from the chip to optical signals by modulating a wavelength of light. In LIGHTPATH, we use micro-ring modulators (MRRs) to modulate light signals with data. The optical receiver (Rx) demultiplexes multiple wavelengths of light, converts modulated wavelengths back to electronic data using photodetectors and sends them to the Serializer/Deserializer (SerDes) module.

Light sources and waveguides. Each LIGHTPATH tile has 16 wavelength-multiplexed lasers. Optical waveguides transport optical wavelengths generated by the lasers across tiles. Waveguides form edges of a two-dimensional grid that connects LIGHTPATH tiles. The pitch of each waveguide is 3μm, allowing LIGHTPATH to support over 10,000 pairs of bus waveguides from a tile. While each waveguide can support multiplexed wavelengths that carry data, the number of connections that can be made by one LIGHTPATH tile is limited by the number of I/O SerDes available in the electrical chip. This is an important factor in the design of our algorithm for collective communication on LIGHTPATH (§3).

Optical switches. A LIGHTPATH tile is equipped with four optical switches where each switch has a degree of $1X3$. We construct these optical switches using Mach-Zehnder Interferometers (MZIs) as shown in Figure 2b. Each switch connects to inter-tile optical waveguides and the three remaining optical switches on the same LIGHTPATH tile. We can program the MZIs to route wavelengths arriving on a tile to one of three neighboring tiles via an optical switch. This routing behavior can be reconfigured, allowing us to create on-demand optical circuits between GPUs on LIGHTPATH.

Fiber connectivity between LIGHTPATH wafers. One LIGHTPATH wafer connects to others using fiber connections. Using the attached fibers, we can cascade several LIGHTPATH wafers to create a rack-scale photonic interconnect.

2.2 Circuit-switched photonic server-scale interconnects

Traditional interconnects between chips on a server are electrical packet-switched (PCIe [37], NVLinks [40]). Recent work has shown that electrical packet-switched host interconnects become bottlenecks at high data rates [3, 44]. In this work, we propose an programmable photonic interconnects like LIGHTPATH as an alternative to electrical packet-switched server-scale interconnects.

Microsecond reconfiguration. We fabricated a testbed that demonstrates the performance of optical devices on LIGHTPATH in GlobalFoundries [16]. A detailed evaluation of the testbed is in §7. We show that MZIs of optical switches on LIGHTPATH can be reconfigured within 3.7 microseconds. The fast reconfiguration of optical switches allows us to establish on-demand optical circuits between a pair of chips bonded on to LIGHTPATH. By design, optical circuits or *photonic tunnels* between chips eliminate contention at intermediate hops

on the path between two chips, once the tunnel is established.

Chip-to-chip optical circuits. Figure 2c shows an example optical circuit constructed on LIGHTPATH using MZIs that direct signals from one transceiver to another through a series of horizontal and vertical bus waveguides. A group of MZIs are configured such that a pair of bus waveguides (one from A to B and vice versa) make a direct connection from a transceiver of tile A to a transceiver of tile B. Low-loss optical crossings enable routing within the same active silicon device layer.

2.3 Implications for distributed ML

While photonic server-scale interconnects can be general-purpose and application agnostic, in this work, we focus on a specific application that is increasingly important: distributed machine learning. Due to the importance of efficient multi-GPU communication for distributed ML, cloud providers have recently deployed special purpose datacenter interconnects [29] by replacing electrical packet switches with optical circuit switches (OCSes). For instance, in modern TPUv4 datacenters, racks of ML accelerator chips are interconnected using OCSes. OCSes in these interconnects enable high bandwidth connectivity between racks, at lower infrastructure cost and power consumption [12, 15, 19, 55, 56].

However, bulk of the innovation in this space has integrated optics into the datacenter interconnect by replacing or augmenting top-of-rack (ToR), leaf and spine packet switches of the clos topology [17] with OCSes [47], leaving *datacenter racks* as monolithic electrical blocks. For instance, racks in Google's TPUv4 datacenters consist of hundreds of electrically-connected accelerator chips. Instead, we aim to deepen the penetration of optics in the datacenter by enabling *chip-to-chip photonic interconnects* within server racks to improve the performance of distributed ML workloads.

Why now? Recent advances in silicon photonics have enabled the commercialization of server-scale chip-to-chip optical interconnects [5, 6, 8, 26, 27, 31]. Thus, LUMORPH is timely proposal for using photonics for making data routing decisions in high-bandwidth server-scale interconnects.

3 Photonic collective communication

We formalize the algorithmic challenge of using server-scale photonic interconnects for collective communication (Algorithm 1). The solution to this challenge forms the core of LUMORPH, a system that creates on-demand photonic circuits between GPUs in a server and rack-scale interconnect to achieve near-optimal optimal collective communication.

Scale of the interconnect. Each LIGHTPATH-like interconnect connects up to 32 modern GPUs. Several LIGHTPATH boards can then be serially connected using point-to-point fiber connections to form a rack-scale programmable optical interconnect with up to 256 GPUs. We denote the interconnected GPUs with a graph G with GPUs as nodes (N) and waveguides as edges (E). Waveguides physically connect the nodes into a two dimensional grid, as shown in Figure 2c. In comparison, state-of-the-art Nvidia DGX-like systems connect up to 32 GPUs in a 4-rack unit server. However, these systems connect the GPUs using a high-bandwidth electrical packet-switched interconnect using proprietary NVLink and NVSwitch technology. Recent work has shown that in high fan-out communication patterns, *i.e.,* communication where several GPUs send data to the same GPU, packet switches suffer from queueing delays, consequently reducing the bandwidth between GPUs [44].

Algorithm 1 Collective communication with LUMORPH

Inputs:

$G\langle V, E\rangle$:	Photonic interconnect with V GPUs, E edges
L:	Number of lasers and photodiodes per tile
W:	Number of waveguides per edge
Pre:	Pre-condition for the communication;
$Post$:	Post-condition for the communication
P_{sd}:	Set of routes between s and d
$\text{WAVE}(P^i_{sd})$:	Wavelength used by path P^i_{sd}
R_{max}:	Maximum number of communication rounds
R_{min}:	Minimum number of communication rounds

Outputs:

$C_{r,s,d,c,i}$	$\in \{0,1\}$	indicator variable; 1 if chunk c is sent from s to d on the i^{th} path in round r
R_r	$\in \{0,1\}$	indicator variable; 1 if any chunk is sent in the r^{th} *round*

Minimize: $max_{R_{min}<r<R_{max}}(r \cdot R_r)$

subject to:

(1) $\sum_{i=1}^{|P_{sd}|} C_{r,s,d,c,i} <= 1, \quad \forall r,c,s,d$

(2) $C_{r,s,d,c,i} \implies \sum_{l<r} \sum_{o \in Post[c]} \sum_{i \in P_{os}} C_{l,o,s,c,i} = 1, \quad \forall c$

(3) $\sum_r \sum_s \sum_{d:d \in Post(c)} \sum_i C_{r,s,d,c,i} = 1, \quad \forall c$

(4) $\sum_d \sum_{i:\text{WAVE}(P^i_{sd})=\lambda} \sum_c C_{r,s,d,c,i} = 1, \quad \forall r,s,\lambda$

(5) $\sum_{i:\text{WAVE}(P^i_{sd})=\lambda} \sum_s \sum_c C_{r,s,d,c,i} = 1, \quad \forall r,d,\lambda$

(6) $\sum_{i:P^i_{sd} \ni e \& \text{WAVE}(P^i_{sd})=\lambda} \sum_s \sum_d \sum_c C_{r,s,d,c,i} \leq W, \forall r, e \in E, \lambda$

(7) $\text{ANY}\{C_{r,s,d,c,i}\} \implies R_r = 1, \quad \forall r$

(8) $R_i = 1, \quad \forall 0 < i < R_{min}$

(9) $R_i \implies R_{i-1}, \quad \forall 1 < i < R_{max}$

No host-based forwarding. These limitations of packet-switched interconnects inspire the design of LUMORPH. LUMORPH leverages programmability of photonic switches on server-scale interconnects like LIGHTPATH to establish on-demand wavelength-switched circuits between GPUs. LUMORPH constructs end-to-end optical circuits between a communicating pair of GPUs, eliminating the need for *host-based forwarding* where intermediate GPUs buffer data destined for another GPU. This design aims to fully benefit from fast reconfiguration of photonic switches and reduce the software stack overheads necessitated by packet switching for data buffering, congestion control *etc.* [56].

Collective communication primitives. GPUs performing distributed ML training and inference use collective communication primitives (*e.g.,* AllToAll, AllReduce) to share local state with each other. This communication is crucial in enabling the speed-up from distributed training and inference. Overall, each GPU has a local buffer which is divided into discrete *data chunks*. A collective communication primitive has a *pre-condition* (*Pre*) that identifies the data chunks available at each GPU before the communication starts. The collective communication primitive also defines a *post-condition* (*Post*) that identifies data chunks available at all GPUs after the communication completes. An optimal collective communication algorithm computes the *communication schedule* that achieves the post-condition in the least time.

Goal of the algorithm. The goal of LUMORPH is compute optimal communication schedules that achieve collective communication primitives on programmable photonic interconnects. A communication schedules describes an ordered sequence of data transfers between GPUs, identified by the source GPU, destination GPU and data chunk. Several transfers can happen concurrently to utilize more of the interconnect resources and the goal of a communication schedule is to finish in as little time as possible.

Establishing on-demand circuits. LUMORPH computes communication schedules for general collective communication primitives. Each pair-wise data transfer in these schedules requires LUMORPH to establish an optical circuit between the GPUs by programming a set of optical switches on the interconnect. Despite the fast microsecond-speed reconfiguration of optical switches on LIGHTPATH, establishing on-demand photonic circuits incurs a significant reconfiguration delay. Thus LUMORPH's algorithm aims to minimize the latency cost of switch reconfiguration. LUMORPH needs a route between a pair of GPUs s and d on the interconnect to establish a circuit between them. P_{sd} represents a set of input routes between the nodes in the interconnect graph. It turns out, selecting these routes has important implications for the optimality of the algorithm. We discuss this in detail in §3.1.

Limited lasers and photodiodes. Each GPU is bonded on a LIGHTPATH tile that has a fixed number of lasers. A laser can launch a light signal of a specific wavelength. Similarly, each tile has the same number of photodiodes as lasers. One photodiode can detect a light signal of a specific wavelength. In addition to selecting a route, establishing a circuit between GPUs also requires selecting a laser at the source s and a photodiode of matching wavelength at the destination d. We refer to the wavelength used by the i^{th} path between s and d with $\text{WAVE}(P^i_{sd})$. Based on our experimental testbed, we assume 16 lasers and photodiodes on each tile of LIGHTPATH. Each signal from a laser can be modulated with electrical data from the chip using different modulation formats. These formats decide the data rate of the circuit. Recall from §2 that adding more lasers does not scale the bandwidth beyond the number of I/O ports available to the GPU.

Rounds of communication. We split the communication schedule into discrete steps that we call *rounds*. In a round, LUMORPH establishes circuits by programming optical switches for all transfers within the round. In a round, each GPU transfers a single data chunk on the established circuits. In this way, the LUMORPH's communication schedules incur the reconfiguration delay only once per round. By minimizing the number of rounds, we can reduce the effect of reconfiguration delay on the overall communication schedule latency. We note that LUMORPH does not need to implement an explicit teardown of circuits since in every round, it reconfigures the state of optical switches inherited from the previous round.

Decision Variables and Objective. The main decision variable of our algorithm for computing optimal collective communication schedules is $c_{r,s,d,c,i}$. $c_{r,s,d,c,i}$ is a binary indicator variable. It is 1 if GPU s sends chunk c to GPU d in round r using the i^{th} route between the GPUs. We keep track of the rounds used by a communication schedule using an indicator variable R_r. If a communication round is used for any data transfer, then R_r is 1. The goal of the optimization is to minimize the count of rounds used for data transfers. The assumption baked into this objective is that all transfers within a round take roughly the same time and therefore, as long as the number of rounds is minimized, the communication schedule will take the lowest end-to-end time. The objective is subject to a number of constraints:

Unique path constraint. This constraint ensures that of the several possible circuits that be established between GPUs s and d for the transfer of chunk c, only one is used in a round. It helps LUMORPH achieve the design goal of not splitting data chunks across different paths to avoid the overhead of storing and recombining them at the destination GPU:

$$\sum_{i=1}^{|P_{sd}|} C_{r,s,d,c,i} <= 1, \qquad \forall r,c,s,d \tag{1}$$

Receive before sending. This constraint ensures that only the GPUs who have a data chunk *i.e.,* either they were sent the chunk in a previous round or they had the chunk in the pre-condition, can send the chunk to other GPUs:

$$C_{r,s,d,c,i} \implies \sum_{l<r} \sum_{o \in Post[c]} \sum_{i \in P_{os}} C_{l,o,s,c,i} = 1, \qquad \forall c \tag{2}$$

No redundant transfers. By design, LUMORPH eliminates transfers that are not needed as per the post-condition. In contrsast, packet-switched networks may allow intermediate GPUs to hold on to data before finding the bandwidth to forward it to the destination [45, 56]. However, this host-based forwarding necessitates extra buffer memory on the intermediate host and software overheads of managing chunks at intermediate hops. Thus, LUMORPH only transfers chunks to GPUs that are essential as per the post condition:

$$\sum_r \sum_s \sum_{d:d\in Post(c)} \sum_i C_{r,s,d,c,i} = 1, \qquad \forall c \tag{3}$$

Unique use of laser and photodiode constraints. In a round, a laser on a tile, identified by its wavelength, can be used at most once across all transfers. Similarly, in a round, a photodiode on a tile, identified by its wavelength, can be used at most once across all transfers:

$$\sum_d \sum_{i:\text{WAVE}(P^i_{sd})=\lambda} \sum_c C_{r,s,d,c,i} = 1, \qquad \forall r,s,\lambda \tag{4}$$

$$\sum_{i:\text{WAVE}(P^i_{sd})=\lambda} \sum_s \sum_c C_{r,s,d,c,i} = 1, \qquad \forall r,d,\lambda \tag{5}$$

Unique wavelength per waveguide constraint. While photonic interconnects like LIGHTPATH can have several waveguides on every edge of the interconnect, each waveguide can accommodate one light signal at a specific wavelength. Therefore, an edge on the interconnect grid can sustain as many circuits of a wavelength as the number of waveguides:

$$\sum_{i:P^i_{sd}\ni e \,\&\, \text{WAVE}(P^i_{sd})=\lambda} \sum_s \sum_d \sum_c C_{r,s,d,c,i} \le W, \quad \forall r,e\in E,\lambda \tag{6}$$

Couting useful rounds. LUMORPH allows collective communication schedules to span a number of rounds between R_{min} and R_{max}. We use the following constraint to count the number of rounds it takes for an optimal schedule to complete the collective communication primitive. This constraint marks a round as active if any data transfers occur in this round:

$$\text{ANY}\{C_{r,s,d,c,i}\} \implies R_r = 1, \qquad \forall r \tag{7}$$

Minimum rounds must be used. To prevent the optimization solver from wasting time looking for optimal solutions that are unachievable, we use the input R_{min} to bound the number of rounds in the collective communication from below. We input values of R_{min} based on the rounds in the optimal communication schedule of a smaller interconnect size:

$$R_i = 1, \qquad \forall 0 < i < R_{min} \tag{8}$$

Round count constraint. This ensures that if a higher round is active, earlier rounds will also be considered active:

$$R_i \implies R_{i-1}, \qquad \forall 1 < i < R_{max} \tag{9}$$

What prevents collective completion in one round? In photonic interconnects like LIGHTPATH, GPUs can drive the eight lasers on the tile using the SerDes. This means that a GPU can simultaneously transfer data chunks along eight different circuits, provided photodiodes of matching wavelengths are available at the receiving ends of these circuits. In a grid of 8 GPUs, establishing circuits like this would lead to completion of the ALLTOALL collective in a single round. However, as the interconnect scales to rack dimensions, it has upto 256 GPUs. There are not enough lasers and photodiodes on each tile to complete ALLTOALL communication in a single round. Recall that adding more lasers on LIGHTPATH tiles is not the solution since the bottleneck is the number of available I/O ports on GPUs that can drive lasers.

Importance of more waveguides. The other constraint, aside from available lasers and photodiodes, that prevents Algorithm 1 from completing communication in one round is the number of waveguides between tiles. A waveguide can accommodate only one circuit of a specific wavelength. Overlapping circuits that use the same waveguides must be staggered across rounds. Fortunately, LIGHTPATH can implement thousands of waveguides at little cost, allowing algorithm designers to achieve faster collective communication by leveraging several waveguides. We evaluate how number of waveguides influence optimality of communication schedules in §5.

3.1 Minimize circuit overlap on waveguides

The LUMORPH optimization formulation takes as input a candidate set of routes between GPU pairs. It selects routes from this candidate set to establish photonic circuits. We keep route selection outside of the main algorithm to reduce the search space of Alg. 1 and help it scale. We use two different strategies of selecting routes to provide as input to Alg. 1.

K-shortest paths. We use Yen's algorithm to efficiently compute k-shortest paths between GPU pairs [58]. While an intuitive choice, the advantage of using shortest paths is limited in server-scale photonic interconnects because longer paths do not degrade optical signals enough to reduce the achievable data rate. We show this experimentally by measuring signal loss with distance on LIGHTPATH (§7). Moreover, circuits using longer paths use the same resources as shorter paths — a laser and a photodiode. However, we still prefer shorter paths since longer paths occupy wavelengths on more waveguides compared to shorter paths.

Minimize circuit overlap or *congestion*. By preventing redundant transfers LUMORPH ensures that lasers and photodiodes — scarce resource of the interconnect, are used only to perform useful transfers. The other remaining resource are the optical waveguides which can carry only one wavelength of a color (Eqn 6). Our second mechanism of route selection reduces the overlap of circuits on waveguides to prevent a waveguide from becoming the bottleneck that prevents the collective from completing. The notion of preventing transfers that overlap on the same edge of a network has been explored in theoretical computer science in the context of minimizing the *makespan* of data transfers by reducing the overlap of transfers along one edge (called *congestion*) and length of the longest path (called dilation) [33,43]. We implement a second method for computing routes between GPUs to minimize the resulting overlap of circuits on waveguides. This algorithm encodes route selection with minimal overlap as an optimization. The optimization closely resembles the multi-commodity flow problem with an optimization objective different from traditional flow problems that maximize flow. Details of the encoding are in Alg. 2 in the Appendix A.

3.2 Leveraging symmetry to scale

We implement Algorithm 1 in Python 3 with Gurobi's Python bindings [18]. Similarly, we implement Algorithm 2 to pre-compute routes and provide them as input to Algorithm 1. We find that Algorithm 2 produces routes between GPUs that minimize overlap over waveguides within tens of seconds for rack-scale interconnect topologies of up to 128 nodes (Figure 3). However, Algorithm 1 fails to converge to a solution even for an interconnect with only 8 GPUs on a grid. Related work has found that the problem of computing optimal schedules for collective communication even on fixed topology electrical interconnects is NP-Hard [44] since the problem involves finding optimal routes and scheduling transfers on routes with overlapping edges. In addition to solving these challenges, LUMORPH's optimization (Algorithm 1) must also manage the implications of establishing on-demand circuits on latency optimality of the communication schedule.

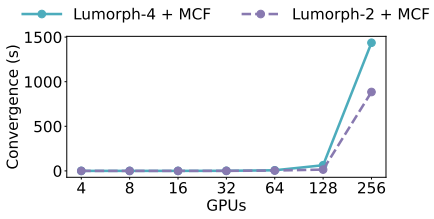


Figure 3: Time taken by LUMORPH to compute routes such that the overlap of optical circuits on waveguides is minimized.

Symmetry with ALLREDUCE. We tackle the scaling challenge of LUMORPH's optimization by taking advantage of the symmetry in the topology of photonic interconnects like LIGHTPATH. Our key insight is that the interconnect grid is composed of squares of four GPUs. We first *divide* the interconnect grid into squares with four nodes. We use Algorithm 1 to compute the optimal communication schedule for the ALLREDUCE collective on a four-GPU square. We recursively *conquer* by sharing the reduced state from one GPU square with another and then from one GPU 8-tuple with another and so on. We note that the circuits over which the conquer step occurs are established along paths computed by Algorithm 2. We implement the divide-and-conquer algorithm and find that while it is sub-optimal in performance, it scales to rack-scale interconnects with up to 256 GPUs. We quantify the optimality gap of this approach in §5.

4 Leveraging homogeneity of photonic circuits

We developed an optimization formulation to find schedules for general collective communication primitives (Algorithm 1) in §3. We reduced the search space of the encoding by providing pre-computed paths (§3.1), bounded the number of communication rounds (Eqn. 8) and used several other techniques to scale the algorithm to large interconnect topologies. These techniques did not significantly reduce the convergence time of the encoding and we developed a divide and conquer algorithm that trades off optimality for runtime (§3.2).

A general optimization formulation like Algorithm 1 is ideally suited for two purposes. First, it makes no assumptions about the pre- and post- conditions of collective communication. Second, it makes no assumptions about the differences in bandwidths of different links. For instance, the optimization can find communication schedules for bespoke communication patterns on a network with heterogenous link bandwidths, like the NVlink + Infiniband network of NVIDIA DGXes [38]. However, this generality of the optimization problem comes at the cost of computational complexity that prevents Algorithm 1 from converging in a reasonable time.

Homogeneity of inter-GPU optical circuits. Our key insight is that inter-GPU links resulting from circuits on server-scale photonic interconnects are *homogeneous*. The circuits underpinning the links can have different path lengths and hops but due to the short reach of the interconnect, such differences do not cause reduction in end-to-end data rates (§7). We use this property to our advantage since there are well-known communication schedules for common collective communication primitives on homogenous networks [53]. We discuss how LUMORPH adapts these algorithms for optical interconnects, thereby computing near-optimal communication schedules without needing to trade-off optimality for algorithm convergence time. We focus on the two main collective communication primitives that are workhorses of distributed ML training and inference — ALLREDUCE and ALLTOALL.

$\alpha-\beta$ cost model. The well-known α-β [21] cost model captures the time taken by collective communication to finish. The α constant captures the fixed cost of sending one data chunk on a link. For instance, sending a data chunk incurs software overheads of preparing the memory buffer. Such costs are independent of the size of the chunk and are paid once for every transfer. The β cost captures the time it takes to put the bits of the data chunk on the wire. Commonly called the transmission delay, *beta* costs are a function of the size of the buffer as well as the bandwidth of the link. Overall, the cost of sending a chunk of size s along a link is $\alpha+\beta\cdot s$, where β is the inverse of the link's bandwidth.

Lower bound on α cost. Previous work has computed lower bounds on the α cost of algorithms for well-known collective communication primitives executing on homogenous electrical interconnects [53]. In such interconnects, one GPU can talk to only one other GPU in a round of communication. In all collective communication, there is one node which has data that is needed by all other nodes. Since, in each round, the number of nodes that have the data can *at most* double since a GPU can only talk to one other GPU, it takes at least $log_2 N$ rounds of communication between N GPUs where fixed costs are paid once every round. Thus, the lower bound on α costs a collective communication is $log_2 N \cdot \alpha$.

Lower bound on β cost. For β cost, we focus primarily on the ALLREDUCE collective which is of crucial importance for distributed ML training. Each GPU has a buffer of size

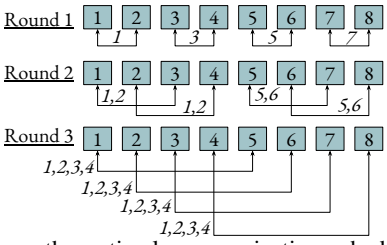


Figure 4: shows the optimal communication schedule for ALLGATHER where initially each GPU has one data chunk and the post condition requires that all GPUs have all data chunks ($1-8$). For clarity, we annotate the arrows with the transfers from the left GPU to the right GPU only, omitting transfers in the reverse direction.

B. In the best case, each GPU has a fair fraction of the buffer *i.e.,* $\frac{B}{N}$ and it has to send $(N-1)$ chunks, one to every GPU to perform the reduction, leading to $(N-1)\cdot\frac{B}{N}$ transfers. After the reduction, each GPU must receive $(N-1)$ chunks of size $\frac{B}{N}$ from others. Overall, previous work has shown that the lower bound on ALLREDUCE β cost is $2(N-1)\cdot\frac{B}{N}$ [53].

To achieve these lower bounds for ALLREDUCE, previous work has used *recursive halving* to first perform REDUCESCATTER, followed by *recursive doubling* to perform ALLGATHER. Figure 4 shows how recursive doubling takes log_2N steps to finish ALLGATHER. The combination of REDUCESCATTER and ALLGATHER achieves the ALLREDUCE collective. We adopt these algorithms in LUMORPH where instead of using pre-existing links, we establish on-demand circuits to make transfers to achieve recursive doubling/halving. Since establishing circuits incurs a reconfiguration delay, the overall α cost of using this algorithm on LIGHTPATH is the usual α cost plus the reconfiguration delay. We adopt a similar algorithm [7, 53] for AllToAll that achieves lower bound on α of log_2N steps.

4.1 Adapting optimal ALLREDUCE for LIGHTPATH

Photonic interconnects allow us to challenge the key assumption of well-known algorithms like recursive doubling and halving: in photonic interconnects, each GPU can talk to multiple GPUs in a given round. A GPU can on a LIGHTPATH tile can simultaneously talk to as many neighbors as the number of lasers on the tile. LUMORPH leverages this ability to generalize the recursive doubling/halving algorithms to *recursive quadrupling/quartering*. Figure 13 in Appendix B shows recursive quadrupling using an example of 16 GPUs. The associated tradeoff is important. Splitting a GPU's total egress bandwidth across multiple wavelength-switched circuits allows LUMORPH to complete the collective in fewer rounds, reducing the α cost. However, using *thinner* connections of less bandwidth instead of *fatter* ones increases the β constant, leading to higher bandwidth cost. We note that since LIGHTPATH has 16 lasers, we can generalize this idea even further by splitting egress bandwidth from a GPU into 7 simultaneous transfers. However, not only this is complex, it also has diminishing returns, as we discuss in §5.

5 Micro-benchmarking LUMORPH

In this section we evaluate LUMORPH's algorithms for collective communication (§3 and §4). We note that the integer linear program (Algorithm 1) produces optimal schedules but does not scale to LIGHTPATH grids with more than 8 GPUs even after several days of run time. We call the divide and conquer approximation to the encoding D&C in the graphs. We call the recursive doubling/halving adapted with optical reconfiguration (§4) LUMORPH-2 and recursive quadrupling/quartering with optical reconfiguration LUMORPH-4.

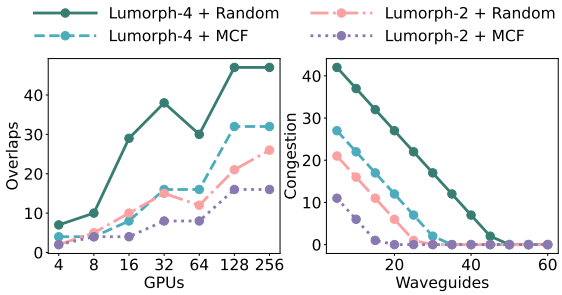


Figure 5: The Left figure shows the maximum number of overlaps on an edge during All-reduce. The Right figure shows the congestion on that edge for 256 GPUs with increasing number of waveguides.

Circuits overlapping on waveguides. First, we evaluate the effect of path selection algorithms in LUMORPH. Recall that LUMORPH has two approaches to selecting routes for constructing optical circuits. First, it computes shortest paths between nodes and select one path from the shortest paths at random (called LUMORPH + Random). In the second approach, LUMORPH computes routes that minimize the maximum overlap of circuits along any edge in the interconnect (called LUMORPH + MCF). Figure 5(left) shows that as the interconnect scales to 256 GPUs, the number of overlaps of circuits on waveguides increases for both path selection algorithms in LUMORPH. However, shortest paths do significantly worse than the minimum congestion paths computed by Algorithm 2. We also note that while using recursive quadrupling/quartering is likely to achieve better performance, it does cause more circuit overlaps on waveguides in LIGHTPATH (LUMORPH-4 + MCF and LUMORPH-4 + Random)

How many waveguides is enough? Overlapping circuits on the same waveguide or edge in LIGHTPATH can slow the completion of a collective since each waveguide allows only one circuit of a given wavelength. Overlapping transfers that use the same waveguide must be serialized across rounds, lengthening the collective schedule. We compute how many waveguides are needed to drive the overlap on edges or congestion [33, 43] to 0. Figure 5(right) shows that 30 waveguides between tiles are enough eliminate congestion for all algorithms in LUMORPH. Since pitch of silicon nitride waveguides is very small (3μm), making it feasible for LIGHTPATH to fabricate thousands of waveguides between tiles.

Effect of waveguides due to contention. Figure 6 shows

the impact of number of waveguides on the performance of LUMORPH's schedules. We find that fewer waveguides (*e.g.,* 20) between a pair of GPUs on LIGHTPATH lead to more reconfigurations of the interconnect whereas the number of reconfigurations are much lower on an interconnect with 40 waveguides. Thus waveguides have a direct effect on the performance of LUMORPH's collective schedules. Given they are cheap to integrate on server-scale interconnects, we hope our analysis will encourage future server-scale interconnects to include tens of waveguides per node pair.

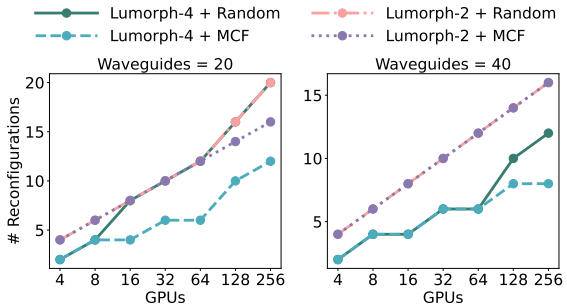


Figure 6: Number of reconfigurations in All-reduce calls with 20 (left) and 40 (right) waveguides on each edge.

Improvement in collective run times. We evaluate the ability of LUMORPH's collective communication schedules to accelerate the run time of individual ALLREDUCE communication on a server-scale interconnect. We compare LUMORPH with the hardest baseline — an electrical interconnect that does not need reconfiguration and has an ideal switch with no queuing delays. We note that scaling an electrical packet switched interconnect to 256 GPUs in a rack with no queueing delays is impractical [44]. In addition, we use state-of-the-art Ring and Tree algorithms for ALLREDUCE on the ideal switch-based interconnect. These algorithms are implemented by Nvidia's collective communication library (NCCL) [39] which is used in all practical multi-GPU environments training environments. Figure 7 shows the run time of collective communication schedules in microseconds as a function of the size of GPU buffers for 64, 128 and 256 GPU interconnects. We note that LUMORPH-2 and LUMORPH-4 achieve the lowest time to complete the collective despite having to pay reconfiguration latency to establish circuits in each round of the collective. Related research on optical topology engineering in datacenter-scale interconnects [30,56] also uses the Ring algorithm to evaluate their performance. Therefore, benefits from LUMORPH's collective communication algorithm should further improve the performance of such systems.

Trade-off between $\alpha-\beta$ costs. The two best performing collective algorithms are LUMORPH-2 and LUMORPH-4. LUMORPH's collectives complete in nearly 80% lesser time compared to both Ring and Tree algorithms with an ideal switch despite paying for reconfiguration of the interconnect several times during the communication schedule for 256 GPUs at 64 MB buffer size. We note that while D&C beats the popular Ring ALLREDUCE algorithm at small buffer sizes, it becomes worse than Ring ALLREDUCE for buffers larger than 16 MB. The main reason for this is the trade-off between α and β costs inherent in collective communication. Smaller buffers incur small β costs which depend on the size of the buffer and thus, the α costs dominate at small buffer sizes. D&C produces schedules that minimize the number of rounds in the communication schedule, consequently minimizing the α costs. However, at higher buffer sizes, β costs dominate and the Ring algorithm is β cost-optimal. We see similar trends across other algorithms in Figure 7.

Effect of photonic switch reconfiguration delay. For the experiments in Figure 7, we consider the reconfiguration latency for establishing circuits in every round of communication as 3.7μs. We derive this value experimentally by building a testbed for LIGHTPATH (§7). However, recent work has shown the possibility of achieving nanosecond-speed reconfiguration of photonic switches [4]. Systems like SiP-ML [30] assume a reconfiguration delay of 25μs. We evaluate the performance of an ALLREDUCE using LUMORPH with different switch reconfiguration delays in Figure 8a. We find that LUMORPH does 4X better than an ideal interconnect (no reconfiguration delay and no queueing delays) when switches can be reconfigured at 3.7μs. However, if the reconfiguration delay goes as high as 25μs, as previous work has discussed, on-demand circuit switching does not buy an end-to-end performance gain over the Ring ALLREDUCE on an ideal interconnect.

Effect of bandwidth of on-chip laser. Finally, we evaluate the effect of the bandwidth of individual lasers of the LIGHTPATH wafer on the performance of LUMORPH. Figure 8b shows the duration of one ALLREDUCE collective communication for different buffer sizes as we scale the per-laser bandwidth. Higher bandwidth lasers significantly improve LUMORPH's performance especially at high buffer sizes. The vertical line shows that 16 lasers each of $\approx$ 18 GBps data rate achieve the total of state-of-the-art Nvidia NVLinks, 300GBps. In §7 we show the feasibility of achieving per laser data rates in the range of 112–228 Gbps on LIGHTPATH.

6 Distributed ML with server-scale photonics

In this section, we evaluate the benefit of training ML models (BERT, NCF and InceptionV3) [11,20,50] across hundreds of GPUs connected using a programmable optical interconnect where LUMORPH programs the silicon photonic switches to establish inter-GPU circuits for communication. We first use existing frameworks to calculate optimal parallelization strategies for three ML models. We then compute the time it would take to train these models as per the optimal parallelization strategy over a LUMORPH-managed rack-scale interconnect. Distributed ML training frameworks model neural networks as directed graphs where nodes represent layers in the network and directed edges represent the flow of data from one layer to another during the forward and backward

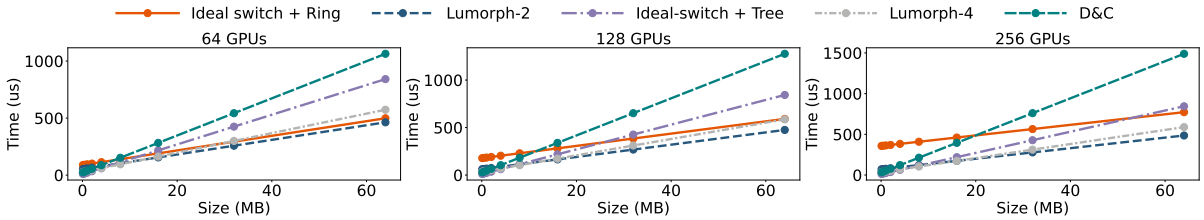
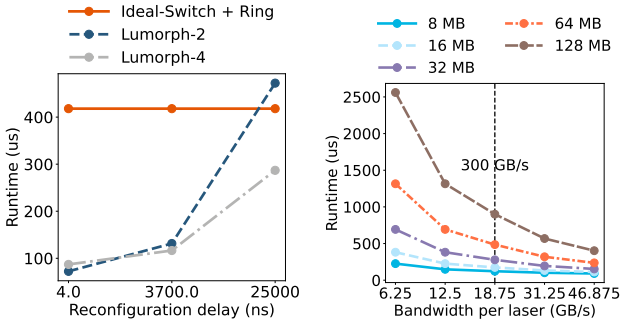


Figure 7: Runtime of AllReduce collective with different algorithms across interconnects with different GPUs.



(a) (b)

Figure 8: (a) shows reconf. delay vs. runtime of AllReduce (256 GPUs). (b) shows effect of laser bandwidth on collective runtime.

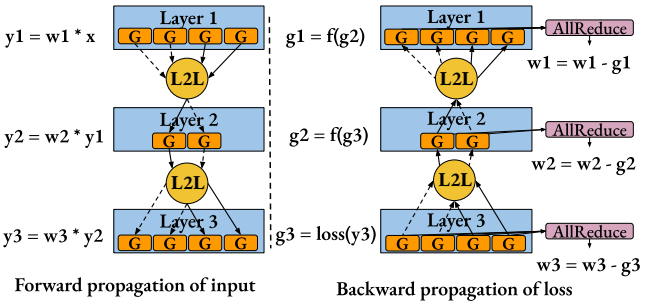


Figure 9: Directed graph of a neural network's iteration. (G = GPU)

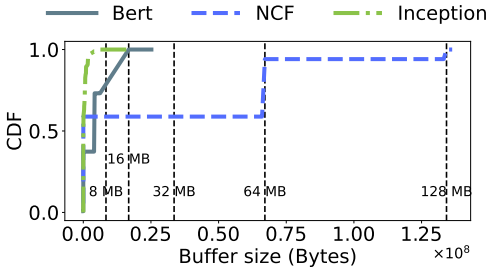


Figure 10: CDF of AllReduce buffer sizes across the 3 models.

pass. The model training process has two parts. In the first, called the forward pass, input data propagates from the first layer of the neural network to the last layer to calculate the loss with respect to ground truth. In the second, the loss gradient back-propagates from the last layer to the first layer while updating the weights on each layer [25].

Distributed training. ML models have been growing in size, making it impossible to fit an entire model in the memory of a single GPU. As a result, both the model and the data it uses to train must be distributed across GPUs for training. The architecture of the model and the size of the minibatch *i.e.,* number of input data points processed in one iteration of forward and backward pass, decide the how much memory is required to execute a layer of a model.

Parallelization strategies. Data parallelism and model parallelism are two common techniques used in distributed ML training. Data parallelism parallelizes the model across the data dimension. It replicates the entire neural network on each GPU where each GPU processes a subset of the input training data. The gradients are synchronized across GPUs during back-propagation. Model parallelism parallelizes across the operator or layer dimension where layers are split across multiple GPUs and the data flows across the GPUs during both forward and backward propagation. Hybrid parallelization utilizes both data and model-parallelism on each layer [25].

Optimal parallelization strategies and predictable communication. Recent work has developed systems that help determine the best parallelization strategy for a given ML model [25, 54]. These systems compute the runtime of different potential parallel strategies by simulating the directed computation and communication graph on a single GPU (Figure 9). The parallelization strategy determines which GPUs must communicate to propagate input and gradients during the forward and backward propagation as shown in the Figure 9. The end-to-end iteration time for training a model is computed by traversing this graph in a topologically sorted order. The training process involves repeating the same directed graph multiple times until the target accuracy is reached, leading to predictable communication patterns [25].

Types of inter-GPU communication There are two main types of communication in a training iteration. The first is called a **LayerToLayer** transfer that occurs during both the forward and backward passes. The set of GPUs that communicate using LayerToLayer transfers depends on the model's architecture and the parallelization strategy. The pattern can vary anywhere from one-to-one to all-to-all communication across the GPUs. For example, recommender models such as NCF and DLRM result in an AllToAll communication pattern between the embedding layers and the subsequent layer. The second type is the AllReduce collective communication resulting from gradient synchronization during the backward pass. Pipelining layers across different GPUs results in one-to-one communication pattern. The Layer-

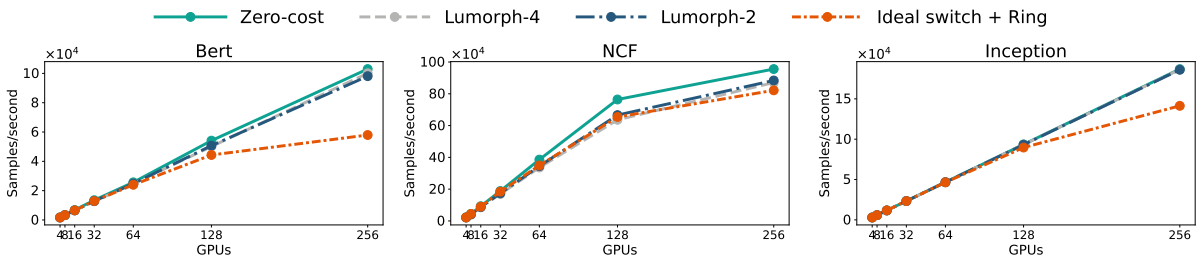
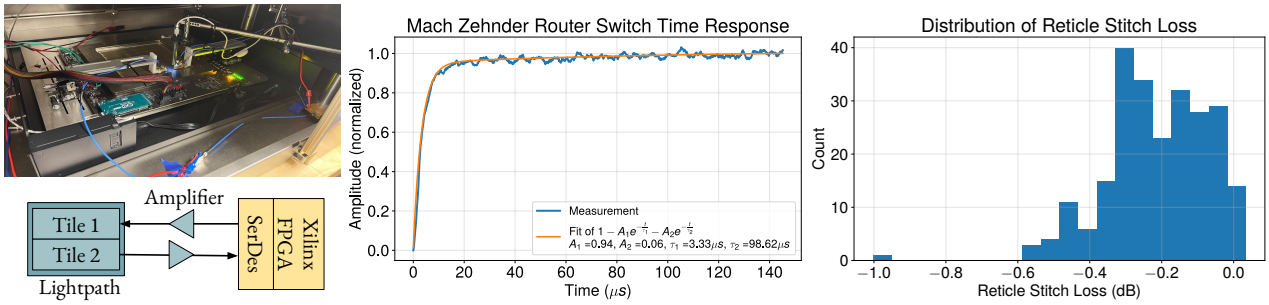


Figure 11: End-to-end throughput of three representative machine learning models.



(a) Testbed. (b) Reconfiguration latency. (c) Stitching loss.

Figure 12: 12a shows a photograph of our LIGHTPATH hardware evaluation setup. We connect two tiles on the LIGHTPATH wafer to a Xilinx FPGA through amplifiers. We use the FPGA to generate traffic that is sent to Tile 1 on LIGHTPATH. Modulators on the tile transmit the traffic to Tile 2. We measure characteristics (*e.g.,* bit error rate) using this transfer. We use an Arduino controller to program the LIGHTPATH device through the JTAG interface [1]. 12b shows that it takes roughly 3.7 microseconds to reconfigure an optical switch on LIGHTPATH. 12c shows there is a 0.25 db loss when signals cross LIGHTPATH tile boundaries.

TOLAYER transfers are critical to complete the forward and backward propagation and must be executed in order before the performing computation of the next layer in the graph (Figure 9). Additionally, ALLREDUCE transfers need to complete before the end of the iteration and in no particular order. This is because the compute in the backward propagation has no dependency on gradient synchronization done by the ALLREDUCE calls. The weights need to be updated with the gradient only before the forward pass of the next iteration.

6.1 Improving training throughput with LUMORPH

We use the FlexFlow simulator to create the compute graph for three machine learning models representative of language models (BERT [11]), vision models (Inception [50]) and recommendation models (NCF [20]). We simulate the end-to-end iteration time by executing each layer on the GPU and determining the communication costs through hardware benchmarks of LIGHTPATH. We provide the implementation details of our evaluation setup in Appendix C.

We compare the training throughput of models trained by LUMORPH with the popular Ring algorithm used by NCCL [39]. We use the α value of 0.7 μs, derived by recent work [44]. For LUMORPH the α value is 0.7 + 3.7 μs to account for the reconfiguration delay. Figure 11 shows that LUMORPH performs up to 1.7X better than the Ring algorithm. BERT shows the highest throughput improvement because the parallelization strategy has many ALLREDUCE calls of small sizes (Fig. 10) and a few LAYERTOLAYER transfers. Therefore, LUMORPH algorithms that optimize both latency and bandwidth perform better than Ring at larger number of GPUs. Inception also shows an improvement of 1.31X with LUMORPH-4 compared to the Ring algorithm since it uses data parallelism and has at least one ALLREDUCE call per layer with comparable sizes to BERT (Fig. 10). The impact of lower collective communication time on throughput is higher for BERT than Inception since Inception has more compute to overlap with communication than BERT. This also explains why LUMORPH-2 and 4 are very close to zero-cost communication in Inception. LUMORPH algorithms show around 8% improvement in throughput in NCF because NCF has fewer ALLREDUCE calls but with larger buffer sizes which makes it more bandwidth bound. So, the gap between the Ring algorithm and the LUMORPH algorithms is lower.

7 Hardware benchmarks

We fabricated a testbed that demonstrates the performance of optical devices on LIGHTPATH at GlobalFoundries [16]. In this section, we use the testbed to benchmark the data transmission properties of LIGHTPATH. Results from these experiments have informed the design of LUMORPH's algorithms for collective communication (§3).

Experimental setup. We connect two tiles on a LIGHTPATH wafer to each other through an FPGA. Figure 12a shows a photograph of our hardware evaluation setup and the schematic of the same setup. We configure optical switches on LIGHTPATH

tiles using an Arduino controller through the JTAG interface [1]. The entire periphery of the LIGHTPATH wafer die is wirebonded to the socket. JTAG signals, along with electrical power, are transferred into LIGHTPATH through the wirebonds. In the testbed, six indium-phosphide (InP) 1310nm lasers are bonded to the silicon photonic wafer. Lasers power the optical links in LIGHTPATH. Although in this prototype the lasers are directly bonded on chip, external lasers can also be used instead by coupling the light through optical fibers.

Error-free data transmission. We perform a transmission loopback experiment where a Xilinx VCU128 FPGA Evaluation board drives the transmitter on a LIGHTPATH tile. Modulated signals from this tile propagate on LIGHTPATH to the receiving tile. The received data is sent back to the same FPGA for bit error rate (BER) checking. We send pseudo random bit sequence (PRBS) signals from the FPGA's 28Gbps SerDes to the optical modulator on LIGHTPATH. The light signals traverse across LIGHTPATH tiles, passing through optical switches, and are finally detected by a Germanium photodetector, whose signals are looped back to the FPGA Evaluation board. At data rates of 10 Gbps, 15 Gbps, and 20 Gbps, we observe very low bit error rates at $6.96X10^{-13}$, $6.62X10^{-13}$, and $5.60X10^{-14}$, respectively. The results demonstrate the feasibility of optical switching on LIGHTPATH and high-speed data links with low error rates across a silicon photonic wafer.

Bi-directional bandwidth. We modulate the signals in the LIGHTPATH testbed with the non-return zero (NRZ) and Pulse Amplitude Modulation 4-level (PAM4) modulation formats. The loopback experiment shows the photonic interconnect's ability to enable low error rate transmission at these modulation formats. We note that a BER of $1e^{-4}$ is sufficient to drive a signal with spectral width of 56 GHz at PAM-4 modulation, achieving a data rate of 112 Gbps. Similarly, at the same modulation format, increased spectral width of 112 GHz achieves the data rate of 224 Gbps per wavelength. Therefore, we conclude that inter-tile communication on LIGHTPATH can sustain high data rates of over 200 Gbps per wavelength. We use these data rates to parameterize the bandwidth per laser in LUMORPH's evaluation (§5).

Dynamic connectivity. In this experiment we experimentally derive the latency of reconfiguring optical switches on LIGHTPATH. This reconfiguration latency is the cost LUMORPH pays to adapt the interconnect to the communication patterns of distributed ML workloads. We use the Arduino controller as the host device for programming optical switches on LIGHTPATH. The controller issues commands to the digital circuits within LIGHTPATH on how to configure the MZI routers that constitute optical circuit switches (Figure 2b in §2). We measure the time it takes to reconfigure the switching behavior of MZIs on LIGHTPATH. Figure 12b shows that the MZI reconfiguration delay *i.e.,* time taken for the signal amplitude to reach 1 from 0, is roughly 3.7 microseconds. Recent work has shown that this delay can be further reduced by driving the MZI with pre-emphasis or pulsed excitation [4]. We note that the MZI we have fabricated in this experimental testbed is one of many possible thermo-optic MZIs that can be optimized for either switching time, power consumption, or area/footprint [24, 35].

Low signal loss. We measure the loss in signal quality as modulated light propagates on LIGHTPATH. Transmission losses in the silicon photonic components (*e.g.,* waveguides, switches) can limit the size of the LIGHTPATH device. The main contributors to the transmission losses are the *reticle stitching loss* at the boundary of two LIGHTPATH tiles and the *propagation loss* in waveguides. Figure 12c shows a distribution over 200 loopback measurements of reticle stitching across the entire LIGHTPATH wafer, where light crosses the reticle boundary twice: light is input into one reticle, crosses to another reticle, and then crosses back to the original input reticle before being measured. The nominal reticle-stitching loss is 0.25 dB per crossing. Waveguides on LIGHTPATH use Silicon Nitride (SiN) since the material suffers low loss. Our experiments show that signals propagating in LIGHTPATH's SiN waveguides suffer a loss less than 0.4db/cm.

8 Related Work

Multi-accelerator systems for distributed training. Closely related work uses program synthesis to develop communication schedules on heterogenous interconnects like the Infiniband + NVLink interconnect of Nvidia's DGX boxes [44]. In contrast, LUMORPH develops collective communication schedules for programmable optical rack-scale interconnects.

Long-haul optics. Recent work leverages optical components (*e.g.,* modulators, ROADMs) for failure resilience [60], capacity augmentation [48, 49] and long-running bulk data transfers [28]. While it uses similar ideas of programming optical components as LUMORPH, the domains are different. LUMORPH applies to novel server-scale interconnects whereas this work focusses on long-haul wide-area networks.

Datacenter-scale optics. Recently there has been growing commercial interest in making optics an active part of routing in datacenter interconnects. For instance, Google has recently replaced the spine layer packet switches in their datacenter interconnects with optical circuit switches [29, 46, 47]. Researchers have used optics for reconfiguring the datacenter interconnect [19, 36, 51, 56]. Researchers have used silicon photonic interconnects for improving ML training [30, 42, 52, 57]. This work has mainly focussed on relatively slow and infrequent reconfiguration of the interconnect, called topology engineering. Unlike them, LUMORPH actively reconfigures the interconnect even within a single communication collective.

Silicon photonics for ML. Recent work has began to use silicon photonic components for machine learning workloads [30, 59]. This work focusses on datacenter-scale interconnects and uses the Ring algorithm for collective communication. LUMORPH develops communication schedules for server and rack-scale optical interconnects.

References

[1] Ieee standard for reduced-pin and enhanced-functionality test access port and boundary-scan architecture. *IEEE Std 1149.7-2009*, pages 1–985, 2010.

[2] Scaling distributed machine learning with In-Network aggregation. In *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*, pages 785–808. USENIX Association, April 2021.

[3] Saksham Agarwal, Arvind Krishnamurthy, and Rachit Agarwal. Host congestion control. In *Proceedings of the ACM SIGCOMM 2023 Conference*, ACM SIGCOMM '23, page 275–287, New York, NY, USA, 2023. Association for Computing Machinery.

[4] Amir H Atabaki, Ali A Eftekhar, Siva Yegnanarayanan, and Ali Adibi. Sub-100-nanosecond thermal reconfiguration of silicon photonic devices. *Optics express*, 21(13):15706–15718, 2013.

[5] Daniel J Blumenthal. Integrated combs drive extreme data rates. *Nature Photonics*, 12(8):447–450, 2018.

[6] Grant M Brodnik, Mark W Harrington, John H Dallyn, Debapam Bose, Wei Zhang, Liron Stern, Paul A Morton, Ryan O Behunin, Scott B Papp, and Daniel J Blumenthal. Optically synchronized fibre links using spectrally pure chip-scale lasers. *Nature Photonics*, 15(8):588–593, 2021.

[7] J. Bruck, Ching-Tien Ho, S. Kipnis, E. Upfal, and D. Weathersby. Efficient algorithms for all-to-all communications in multiport message-passing systems. *IEEE Transactions on Parallel and Distributed Systems*, 8(11):1143–1156, 1997.

[8] Nitesh Chauhan, Jiawei Wang, Debapam Bose, Kaikai Liu, RL Compton, Chad Fertig, CW Hoyt, and Daniel J Blumenthal. Ultra-low loss visible light waveguides for integrated atomic, molecular, and quantum photonics. *Optics express*, 30(5):6960–6969, 2022.

[9] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

[10] Wei Deng, Junwei Pan, Tian Zhou, Deguang Kong, Aaron Flores, and Guang Lin. Deeplight: Deep lightweight feature interactions for accelerating ctr predictions in ad serving. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, WSDM '21, page 922–930, New York, NY, USA, 2021. Association for Computing Machinery.

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[12] 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.

[13] Fierce Electronics. ChatGPT runs 10K Nvidia training GPUs with potential for thousands more. https://www.fierceelectronics.com/sensors/chatgpt-runs-10k-nvidia-training-gpus-potential-thous (Accessed on 2023-05-26).

[14] Nadeen Gebara, Manya Ghobadi, and Paolo Costa. In-network aggregation for shared machine learning clusters. In A. Smola, A. Dimakis, and I. Stoica, editors, *Proceedings of Machine Learning and Systems*, volume 3, pages 829–844, 2021.

[15] 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.

[16] Global Foundries. Global Foundries. https://gf.com/, (Accessed on 2023-06-16).

[17] Albert Greenberg, James R. Hamilton, Navendu Jain, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, David A. Maltz, Parveen Patel, and Sudipta Sengupta. Vl2: A scalable and flexible data center network. In *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication*, SIGCOMM '09, page 51–62, New York, NY, USA, 2009. Association for Computing Machinery.

[18] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2023.

[19] Navid Hamedazimi, Zafar Qazi, Himanshu Gupta, Vyas Sekar, Samir R. Das, Jon P. Longtin, Himanshu Shah, and Ashish Tanwer. Firefly: A reconfigurable wireless data center fabric using free-space optics. In *Proceedings of the 2014 ACM Conference on SIGCOMM*, SIGCOMM '14, page 319–330, New York, NY, USA, 2014. Association for Computing Machinery.

[20] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, page 173–182, Republic and Canton of Geneva, CHE, 2017. International World Wide Web Conferences Steering Committee.

[21] Roger W. Hockney. The communication challenge for mpp: Intel paragon and meiko cs-2. *Parallel Computing*, 20(3):389–398, 1994.

[22] HotChips 34. Passage—A Wafer-Scale, Programmable Photonic Communication Substrate. https://hc34.hotchips.org/assets/program/conference/day1, (Accessed on 2023-05-26).

[23] Infinera. AI, Optical Networking, and Indium Phosphide. https://www.infinera.com/blog/ai-optical-networking-and-indium-phosphide/tag/optical/?utm_source=LI&utm_medium=Blog&utm_campaign=AIMLInP, (Accessed on 2023-06-16).

[24] Maxime Jacques, Alireza Samani, Eslam El-Fiky, David Patel, Zhenping Xing, and David V Plant. Optimization of thermo-optic phase-shifter design and mitigation of thermal crosstalk on the soi platform. *Optics express*, 27(8):10456–10471, 2019.

[25] Zhihao Jia, Matei Zaharia, and Alex Aiken. Beyond data and model parallelism for deep neural networks. *Proceedings of Machine Learning and Systems*, 1:1–13, 2019.

[26] Warren Jin, Avi Feshali, Mario Paniccia, and John E. Bowers. Seamless multi-reticle photonics. *Opt. Lett.*, 46(12):2984–2987, Jun 2021.

[27] Warren Jin, Demis D John, Jared F Bauters, Tony Bosch, Brian J Thibeault, and John E Bowers. Deuterated silicon dioxide for heterogeneous integration of ultra-low-loss waveguides. *Optics Letters*, 45(12):3340–3343, 2020.

[28] Xin Jin, Yiran Li, Da Wei, Siming Li, Jie Gao, Lei Xu, Guangzhi Li, Wei Xu, and Jennifer Rexford. Optimizing bulk transfers with software-defined optical wan. In *Proceedings of the 2016 ACM SIGCOMM Conference*, SIGCOMM '16, page 87–100, New York, NY, USA, 2016. Association for Computing Machinery.

[29] Norman P. Jouppi, George Kurian, Sheng Li, Peter Ma, Rahul Nagarajan, Lifeng Nai, Nishant Patil, Suvinay Subramanian, Andy Swing, Brian Towles, Cliff Young, Xiang Zhou, Zongwei Zhou, and David Patterson. Tpu v4: An optically reconfigurable supercomputer for machine learning with hardware support for embeddings, 2023.

[30] 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*.

[31] Tin Komljenovic, Duanni Huang, Paolo Pintus, Minh A. Tran, Michael L. Davenport, and John E. Bowers. Photonic integrated circuits using heterogeneous integration on silicon. *Proceedings of the IEEE*, 106(12):2246–2257, 2018.

[32] ChonLam Lao, Yanfang Le, Kshiteej Mahajan, Yixi Chen, Wenfei Wu, Aditya Akella, and Michael Swift. ATP: In-network aggregation for multi-tenant learning. In *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*, pages 741–761. USENIX Association, April 2021.

[33] Frank Thomson Leighton, Bruce M. Maggs, and Satish Rao. Packet routing and job-shop scheduling in (congestion + dilation) steps. *Combinatorica*, 14(2):167–186, 1994.

[34] Lightelligence. Hummingbird ONOC. https://www.lightelligence.ai/index.php/product/hummingbird.html, (Accessed on 2023-05-26).

[35] Shengping Liu, Junbo Feng, Ye Tian, Heng Zhao, Li Jin, Boling Ouyang, Jiguang Zhu, and Jin Guo. Thermo-optic phase shifters based on silicon-on-insulator platform: State-of-the-art and a review. *Frontiers of Optoelectronics*, 15(1):9, 2022.

[36] William M. Mellette, Rob McGuinness, Arjun Roy, Alex Forencich, George Papen, Alex C. Snoeren, and George Porter. Rotornet: A scalable, low-complexity, optical datacenter network. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '17, page 267–280, New York, NY, USA, 2017. Association for Computing Machinery.

[37] Rolf Neugebauer, Gianni Antichi, José Fernando Zazo, Yury Audzevich, Sergio López-Buedo, and Andrew W. Moore. Understanding pcie performance for end host networking. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '18, page 327–341, New York, NY, USA, 2018. Association for Computing Machinery.

[38] Nvidia DGX Systems, 2021. https://www.nvidia.com/en-us/data-center/dgx-systems/.

[39] Nvidia NCCL, 2021. https://github.com/nvidia/nccl.

[40] Nvidia NVLink and NVSwitch, 2021. https://www.nvidia.com/en-us/data-center/nvlink/.

[41] Larry L. Peterson and Bruce S. Davie. *Computer Networks, Fifth Edition: A Systems Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 5th edition, 2011.

[42] Anthony Rizzo and Keren Bergman. Realizing pb/s io with silicon photonic chiplets. In *Optica Advanced Photonics Congress 2022*, page NeTu1D.1. Optica Publishing Group, 2022.

[43] Thomas Rothvoss. A simpler proof for o(congestion + dilation) packet routing, 2012.

[44] Aashaka Shah, Vijay Chidambaram, Meghan Cowan, Saeed Maleki, Madan Musuvathi, Todd Mytkowicz, Jacob Nelson, and Olli Saarikivi. 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.

[45] Vishal Shrivastav, Asaf Valadarsky, Hitesh Ballani, Paolo Costa, Ki Suh Lee, Han Wang, Rachit Agarwal, and Hakim Weatherspoon. Shoal: A network architecture for disaggregated racks. In *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*, pages 255–270, Boston, MA, February 2019. USENIX Association.

[46] Arjun Singh, Joon Ong, Amit Agarwal, Glen Anderson, Ashby Armistead, Roy Bannon, Seb Boving, Gaurav Desai, Bob Felderman, Paulie Germano, Anand Kanagala, Jeff Provost, Jason Simmons, Eiichi Tanda, Jim Wanderer, Urs Hölzle, Stephen Stuart, and Amin Vahdat. Jupiter rising: A decade of clos topologies and centralized control in google's datacenter network. In *Sigcomm '15*, 2015.

[47] Arjun Singh, Joon Ong, Amit Agarwal, Glen Anderson, Ashby Armistead, Roy Bannon, Seb Boving, Gaurav Desai, Bob Felderman, Paulie Germano, Anand Kanagala, Jeff Provost, Jason Simmons, Eiichi Tanda, Jim Wanderer, Urs Hölzle, Stephen Stuart, and Amin Vahdat. Jupiter rising: A decade of clos topologies and centralized control in google's datacenter network. In *Sigcomm '15*, 2015.

[48] Rachee Singh, Nikolaj Bjorner, Sharon Shoham, Yawei Yin, John Arnold, and Jamie Gaudette. Cost-effective capacity provisioning in wide area networks with shoofly. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, SIGCOMM '21, page 534–546, New York, NY, USA, 2021. Association for Computing Machinery.

[49] Rachee Singh, Manya Ghobadi, Klaus-Tycho Foerster, Mark Filer, and Phillipa Gill. Radwan: Rate adaptive wide area network. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '18, page 547–560, New York, NY, USA, 2018. Association for Computing Machinery.

[50] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[51] Min Yee Teh, Zhenguo Wu, Madeleine Glick, Sebastien Rumley, Manya Ghobadi, and Keren Bergman. Performance trade-offs in reconfigurable networks for hpc. *Journal of Optical Communications and Networking*, 14(6):454–468, 2022.

[52] Min Yee Teh, Shizhen Zhao, Peirui Cao, and Keren Bergman. Enabling quasi-static reconfigurable networks with robust topology engineering. *IEEE/ACM Transactions on Networking*, 31(3):1056–1070, 2023.

[53] Rajeev Thakur, Rolf Rabenseifner, and William Gropp. Optimization of collective communication operations in mpich. *Int. J. High Perform. Comput. Appl.*, 19(1):49–66, feb 2005.

[54] Colin Unger, Zhihao Jia, Wei Wu, Sina Lin, Mandeep Baines, Carlos Efrain Quintero Narvaez, Vinay Ramakrishnaiah, Nirmal Prajapati, Pat McCormick, Jamaludin Mohd-Yusof, Xi Luo, Dheevatsa Mudigere, Jongsoo Park, Misha Smelyanskiy, and Alex Aiken. Unity: Accelerating DNN training through joint optimization of algebraic transformations and parallelization. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*, pages 267–284, Carlsbad, CA, July 2022. USENIX Association.

[55] Guohui Wang, David G. Andersen, Michael Kaminsky, Konstantina Papagiannaki, T.S. Eugene Ng, Michael Kozuch, and Michael Ryan. C-through: Part-time optics in data centers. In *Proceedings of the ACM SIGCOMM 2010 Conference*, SIGCOMM '10, page 327–338, New York, NY, USA, 2010. Association for Computing Machinery.

[56] 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.

Algorithm 2 Minimal overlap of routes on waveguides

Inputs:

$G\langle V,E\rangle$:	Directed graph of photonic interconnect; V GPUs, E edges
$in(v)$:	Set of directed edges incident on node v
$out(v)$:	Set of directed outgoing edges from node v
$L = \{(s,d) \mid s,d \in V\}$:	List of source, destination pairs

Outputs:

x_e^{sd}:	indicator variable; 1 if e in on the path between s and d

Minimize: $\max_e\{\sum_{s,d} x_e^{sd}\}$

subject to:

(10) $\sum_{e\in in(v)} x_e^{sd} - \sum_{e\in out(v)} x_e^{sd} = 0, \forall v \in V - \{s,d\}, \forall sd$

(11) $\sum_{e\in in(v)} x_e^{sd} \leq 1, \quad \forall s,d,\forall v \in V - \{s,d\}$

(12) $\sum_{e\in out(v)} x_e^{sd} \leq 1, \quad \forall s,d,\forall v \in V - \{s,d\}$

(13) $\sum_{e\in out(v)} x_e^{sd} = 1, \quad \forall s,d,v = s$

(14) $\sum_{e\in in(v)} x_e^{sd} \leq 0, \quad \forall s,d,v = s$

(15) $\sum_{e\in out(v)} x_e^{sd} \leq 0, \quad \forall s,d,v = d$

(16) $\sum_{e\in in(v)} x_e^{sd} = 0, \quad \forall s,d,v = d$

[57] Zhenguo Wu, Liang Yuan Dai, Ziyi Zhu, Asher Novick, Madeleine Glick, and Keren Bergman. Sip architecture for accelerating collective communication in distributed deep learning. In *2023 Optical Fiber Communications Conference and Exhibition (OFC)*, pages 1–3, 2023.

[58] Jin Y. Yen. An algorithm for finding shortest routes from all source nodes to a given destination in general networks. *Quarterly of Applied Mathematics*, 27:526–530, 1970.

[59] Yawei Yin, Mingyang Zhang, Zuqing Zhu, and S. J. B. Yoo. Fragmentation-aware routing, modulation and spectrum assignment algorithms in elastic optical networks. In *Optical Fiber Communication Conference/National Fiber Optic Engineers Conference 2013*, page OW3A.5. Optica Publishing Group, 2013.

[60] Zhizhen Zhong, Manya Ghobadi, Alaa Khaddaj, Jonathan Leach, Yiting Xia, and Ying Zhang. Arrow: Restoration-aware traffic engineering. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, SIGCOMM '21, page 560–579, New York, NY, USA, 2021. Association for Computing Machinery.

A Minimal circuit overlap on waveguides

$$\sum_{e\in in(v)} x_e^{sd} - \sum_{e\in out(v)} x_e^{sd} = 0, \quad \forall v \in V - \{s,d\}, \forall s,d \quad (10)$$

$$\sum_{e\in in(v)} x_e^{sd} \leq 1, \quad \forall s,d,\forall v \in V - \{s,d\} \quad (11)$$

$$\sum_{e\in out(v)} x_e^{sd} \leq 1, \quad \forall s,d,\forall v \in V - \{s,d\} \quad (12)$$

$$\sum_{e\in out(v)} x_e^{sd} = 1, \quad \forall s,d,v = s \quad (13)$$

$$\sum_{e\in in(v)} x_e^{sd} \leq 0, \quad \forall s,d,v = s \quad (14)$$

$$\sum_{e\in out(v)} x_e^{sd} \leq 0, \quad \forall s,d,v = d \quad (15)$$

$$\sum_{e\in in(v)} x_e^{sd} = 0, \quad \forall s,d,v = d \quad (16)$$

The algorithm in 2 describes the Multi Commodity Flow routing problem. The input is the graph G that represents the physical topology of the interconnect and a list of route requests $p = (s,d)$. The decision variables are binary variables $x^v_{e^p_{in}}$ and $x^v_{e^p_{out}}$ for each incoming and outgoing edge on every node v for each route p. The constraints are as follows:

- For each source node s, the sum of outgoing edges is 1 and the sum of incoming edges is 0.
- For each destination node d, the sum of incoming edges is 1 and the sum of outgoing edges is 0.
- For every other node v, if there is an incoming edge to the node, then there has to be an outgoing edge from the node.

The algorithm's objective is to minimize congestion across all the edges in the graph while generating paths for every route request in the input. We perform a DFS search on each route request (s,d) to trace the generated path. The DFS algorithm follows only the outgoing edges with a positive decision variable until the destination is reached.

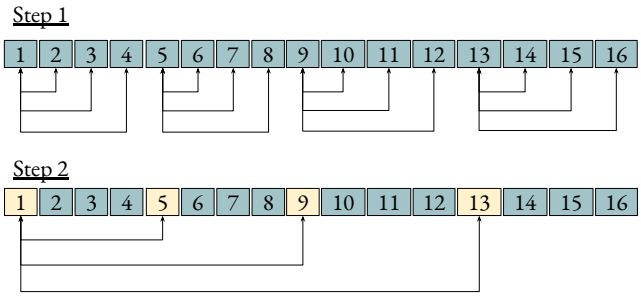


Figure 13: shows the steps of the recursive quadrupling algorithm that LUMORPH proposes. The first step shows the set of communication GPUs. At the end of first step, GPUs $1-4$ have all data chunks $1-4$. Similarly, GPUs $5-8$ have all data chunks $5-8$ and so on. In the second step 1 sends exchanges its local state with 5, 9 and 13. We have omitted the other transfers for clarity. But in two steps $(log_4 N)$, the collective completes.

Algorithm 3 Greedy algorithm for LAYERTOLAYER transfers which are not All-to-All

Initialize *round_id* as 1
while True **do**
  Initialize *links_in_round* as an empty set
  for each *dst* in *circuits* **do**
    Initialize *sources* as the list of sources for *dst*
    Initialize *picked* as an empty list
    while *picked* has less than *lasers* elements and *sources* is not empty **do**
      Choose a random element *source* from *sources*
      Create a tuple *picked_link* as $(source, dst)$
      Add *picked_link* to *picked* and *links_in_round*
      Remove *source* from *sources*
    end while
  end for

  GENERATEMCFROUTES(*links_in_round*)
  Increment *round_id* by 1
  Initialize *should_break* as True
  for each *dst* in *circuits* **do**
    if $circuits[dst]$ is not empty **then**
      Set *should_break* to False
    end if
  end for
  if *should_break* is True **then**
    Break the loop
  end if
end while
round_id is the number of rounds taken by the algorithm.

B Recursive quadrupling

Figure 13 shows how all-gather works with quadrupling instead of doubling. The data sent in the first round is $B/16$ where B is the buffer size. The data sent in the second round is $B/4$, so the size of the data quadruples every round forming a geometric progression. The sum of the this geometric progression for $(log_4 N)$ terms is the total amount of data sent.

The recursive quartering algorithm follows the same traffic pattern as the quadrupling algorithm. The main thing that changes is the amount of data sent in each round. In the first round, GPU 1 sends chunks $4, 8, 12, 16$ to 4. Similarly 4 receives the same chunks from GPUs $2, 3$. Therefore, after the first round GPU 4 has all the chunks from GPUs $1, 2, 3, 4$ related to GPUs $4, 8, 12, 16$ and 1 will have all the chunks from GPUs $1, 2, 3, 4$ related to GPUs $1, 5, 9, 13$. In the subsequent round, GPU 1 sends the reduced data to 5. The data transferred in the first round is $B/4$, second round is $B/16$ and so on which again forms the same geometric progression in the reverse order.

C Details of end-to-end evaluation

We use the FlexFlow simulator [25, 54] to evaluate the end-to-end gains of LUMORPH. The simulator calculates the runtime of the graph by queueing communication calls on the communication devices attached to each GPU object and compute calls on GPU device queues. The simulator starts by queuing tasks which have no dependencies. A queued task is simulated by updating the current time of the simulation if the sum of the ready time of the device and the task's execution time is higher than the current runtime. The ready time of the device is updated after each task to the sum of the previous ready time and the current task's execution time. Once the task is marked complete, it is removed as a dependency of all the tasks that depend on it. If a task has no dependency after this, it is enqueued. We modify the simulator to add an ideal switch so that each GPU has a fixed outgoing bandwidth and incoming bandwidth. We added the LUMORPH-2 algorithm to translate ALLREDUCE calls to individual transfers and enqueue these transfers onto the device queues. We then execute the search algorithm to generate an optimal graph with the least time per iteration. We export the parallelization strategy of each layer and all the LAYERTOLAYER and ALLREDUCE calls including the buffer size, GPUs involved and the layer at which they were issued.

Each ALLREDUCE and LAYERTOLAYER call is tagged with a unique key. The LAYERTOLAYER calls are in two directions, forward and backward. We group the transfers between a pair of layers in each direction and consider the transfers only if the are between two different GPUs.

LAYERTOLAYER algorithms. The communication pattern of the group can be all-to-all or a random pattern. If it is all to all, we again leverage the homogenous circuits as described in §4 and use an algorithm that takes $(log_2 N)$ rounds as proposed in the index algorithm of [7, 53]. We then use MCF to calculate the routes for these circuits. If the pattern is arbitrary, we use a greedy algorithm to generate the schedule. As mentioned earlier, the main bottleneck here is the number of lasers. So, we resort to use multiple thin pipes in the greedy algorithm which is informed by the observation that most of the LAYERTOLAYER transfer are small in size. We describe the greedy algorithm in Algorithm 3.

For ALLREDUCE calls we calculate the required circuits using different algorithms described in the previous section. All the circuits in a given step can be established in parallel, so we need to add the reconfiguration delay only once across each round. Once we have the circuits in each round, we feed that to the MCF route generation algorithm to generate routes that can fit within the waveguide constraint. If the congestion exceeds the number of waveguides, we need to split this round into two and repeat the process until all the circuits in the round are established.

Computing α and β costs. We use the experimentally determined α ($0.7\mu s$) from previous work [44] and add the

reconfiguration delay to it. The α cost is the product of this and the number of rounds. We use the results from hardware measurements to determine the time taken to establish each circuit and transfer the buffer to determine the β cost. We export these communication times to a format which is fed to the graph simulator to replay the graph with LIGHTPATH device instead of the ideal switch.

We add a new graph executor to simulate the graph with a single LIGHTPATH device. Every LAYERTOLAYER and ALLREDUCE call is issued to the LIGHTPATH device and the device queue is executed in that order. We assume a centralized controller per rack and the GPUs communicate with the controller across round boundaries before reconfiguring the switch to which they are bonded. The circuit to the controller are along dedicated waveguides which are pre-determined and this doesn't affect the overlapping constraint since we have free waveguides (fig: 5). We add the time taken for this synchronization between rounds. The compute tasks are still issued to the GPU devices as before. We run this new executor with the exported parallelization strategy and the exported communication times to calculate the iteration time.

Evaluation setup. We execute the simulator on a server with 1 TB DRAM and 128 cores that has two NVIDIA A100 GPUs with 80 GB memory each. The simulator uses one of the GPUs to measure the compute time of the parallelization strategies. We assume 16 lasers per GPU and 150Gbps per laser which to match the existing NVSwitch [40] bandwidth of 300GB/s per GPU. We also show in 7 that speeds up-to 224Gbps per lane are possible which makes 150Gbps well within reason.

Co-scheduling multiple ALLREDUCE invocations with LUMORPH. During the backward pass, a layer can enqueue LAYERTOLAYER transfers and an ALLREDUCE call at the same time. If the sets of GPUs involved in the two calls are not disjoint, then the peers a given GPU speaks with will likely be different across the issued LAYERTOLAYER and ALLREDUCE calls. The transfers on these overlapping GPUs need to be scheduled optimally. The variance is exacerbated by model architectures that have different LAYERTOLAYER patterns. Optimizing the schedule for the entire iteration together is challenging because the schedule of the current layer's calls is required to determine the ready time of the next layer's calls. This is because of the dependency of the next layer's compute on the current layer's LAYERTOLAYER call. This is an NP-hard problem in scheduling, we tackle it by prioritizing LAYERTOLAYER calls over ALLREDUCE calls. The naive approach is to delay all the ALLREDUCE calls to the end of the iteration and execute them sequentially. However, this approach misses the two opportunities, first is overlapping ALLREDUCE calls that execute on disjoint sets of GPUs, second is overlapping ALLREDUCE with compute. We address these challenges by batching ALLREDUCE calls at the boundaries of layers which have LAYERTOLAYER transfers and optimizing the schedule of the batch. We model the ALLREDUCE calls as a graph where the nodes are the calls and the edges connect nodes which have at least one common GPU. After this, the problem is finding maximally independent sets of nodes greedily until a cover set over the set of GPUs is obtained. Each set in the cover-set is a disjoint from each other and the all-reduce calls on them can be executed simultaneously.