Helios: A Hybrid Electrical/Optical Switch Architecture for Modular Data Centers

Nathan Farrington, George Porter, Sivasankar Radhakrishnan,
Hamid Hajabdolali Bazzaz, Vikram Subramanya, Yeshaiahu Fainman,
George Papen, and Amin Vahdat

University of California, San Diego

Abstract

The basic building block of ever larger data centers has shifted from a rack to a modular container with hundreds or even thousands of servers. Delivering scalable bandwidth among such containers is a challenge. A number of recent efforts promise full bisection bandwidth between all servers, though with significant cost, complexity, and power consumption. We present HELIOS, a hybrid electrical/optical switch architecture that can deliver significant reductions in the number of switching elements, cabling, cost, and power consumption relative to recently proposed data center network architectures. We explore architectural trade offs and challenges associated with realizing these benefits through the evaluation of a fully functional HELIOS prototype.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*circuit-switching networks, packet-switching networks, network topology*

General Terms

Design, Experimentation, Measurement, Performance

Keywords

Data Center Networks, Optical Networks

1. INTRODUCTION

The past few years have seen the emergence of the modular data center [12], a self-contained shipping container complete with servers, network, and cooling, which many refer to as a *pod*. Organizations like Google and Microsoft have begun constructing large data centers out of pods, and many traditional server vendors now offer products in this space [6, 33, 35, 36]. Pods are now the focus of systems [27] and networking research [10]. Each pod typically holds between 250 and 1,000 servers. At these scales, it is possible to construct a non-blocking switch fabric to interconnect all of the servers within an individual pod. However, interconnecting hundreds to thousands of such pods to form a larger data center remains a significant challenge.

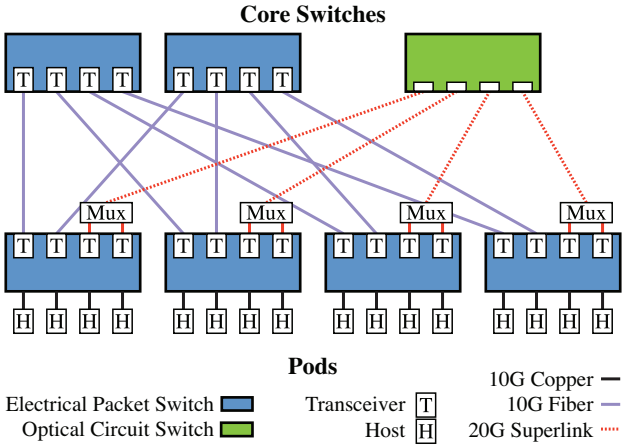

Figure 1: Helios is a 2-level multi-rooted tree of pod switches and core switches. The core consists of both traditional electrical packet switches and MEMS-based optical circuit switches. Superlinks are defined in §3.1.

Supporting efficient inter-pod communication is important because a key requirement in data centers is flexibility in placement of computation and services. For example, a cloud computing service such as EC2 may wish to place the multiple virtual machines comprising a customer's service on the physical machines with the most capacity irrespective of their location in the data center. Unfortunately, if these physical machines happen to be spread across multiple pods, network bottlenecks may result in unacceptable performance. Similarly, a large-scale Internet search engine may run on thousands of servers spread across multiple pods with significant inter-pod bandwidth requirements. Finally, there may be periodic bandwidth requirements for virtual machine backup or hotspots between partitioned computation and storage (e.g., between EC2 and S3).

In general, as services and access patterns evolve, different subsets of nodes within the data center may become tightly coupled, and require significant bandwidth to prevent bottlenecks. One way to achieve good performance is to statically provision bandwidth between sets of nodes with known communication locality. Unfortunately, given

SIGCOMM'10, August 30–September 3, 2010, New Delhi, India.

current data center network architectures, the only way to provision required bandwidth between dynamically changing sets of nodes is to build a non-blocking switch fabric at the scale of an entire data center, with potentially hundreds of thousands of ports.

While recent proposals [1, 9–11] are capable of delivering this bandwidth, they introduce significant cost and complexity. Worse, full bisection bandwidth at the scale of an entire data center is rarely required, even though it is assumed to be the only way to deliver on-demand bandwidth between arbitrary hosts in the face of any localized congestion. One common approach to bring down networking cost and complexity is to reduce the capacity of the inter-pod network by an oversubscription ratio and to then distribute this capacity evenly among all pods [32]. For a network oversubscribed by a factor of ten, there will be bandwidth bottlenecks if more than 10% of hosts in one pod wish to communicate with hosts in a remote pod.

Optical circuit switching and wavelength division multiplexing (WDM) are promising technologies for more flexibly allocating bandwidth across the data center. A single optical port can carry many multiples of 10 Gb/s assuming that all traffic is traveling to the same destination. The key limitation is switching time, which can take as long as tens of milliseconds. Thus, it makes little sense to apply optical switching at the granularity of end hosts that transmit no faster than 10 Gb/s and that carry out bursty communication to a range of hosts. However, the pod-based design of the data center presents the opportunity to effectively leverage optical switching due to higher stability in the pod-level aggregated traffic demands. When there is a lot of burstiness of aggregated traffic demand between different pairs of pods, the number of circuits required to support this bursty communication in the data center would be prohibitive. Such bursty communication is better suited to electrical packet switching.

We thus propose Helios, a hybrid electrical/optical data center switch architecture capable of effectively combining the benefits of both technologies, and delivering the same performance as a fully-provisioned packet-switched network for many workloads but at significantly less cost, less complexity (number of cables, footprint, labor), and less power consumption. Helios identifies the subset of traffic best suited to circuit switching and dynamically reconfigures the network topology at runtime based on shifting communication patterns. Helios requires no modifications to end hosts and only straightforward software modifications to switches. Further, the electrical and optical interconnects in the core array can be assembled from existing commercial products.

We have completed a fully functional Helios prototype using commercial 10 GigE packet switches and MEMS-based optical circuit switches. In §2, we describe how copper links are no longer viable for 10 Gb/s links beyond interconnect distances of 10 m. This gap must be filled by optical interconnects, with new trade offs and opportunities. In §3, we extrapolate Helios to a large-scale deployment and find an opportunity for significant benefits, e.g., up to a factor of 3 reduction in cost, a factor of 6 reduction in complexity, and a factor of 9 reduction in power consumption. In §4, we describe a control algorithm for measuring and estimating the actual pod-level traffic demands and computing the optimal topology for these demands, irrespective of the current network topology and the challenges it imposes on demand estimation. In §5, we find that the appropriate mix of optical and electrical switches depends not only on the volume of communication but on the specific communication patterns. And in §6, we describe the insights we gained from building Helios.

2. BACKGROUND AND MOTIVATION

In this section we discuss the problem of oversubscription, and how existing data center network architectures can only solve that problem through over-provisioning, i.e., providing enough bisection bandwidth for the worst case, not the common case. Then we discuss the technologies such as optical circuit switches and wavelength division multiplexing that make Helios possible.

2.1 The Oversubscription Problem

To make our discussion concrete, we first consider the architecture of a large scale data center built from pods. Each pod has 1,024 servers and each server has a 10 GigE NIC. We assume that the task of interconnecting individual servers within a pod is largely solved with existing [37] and soon-to-be-released dense 10 GigE switches. A number of vendors are releasing commodity 64-port 10 GigE switches on a chip at price points in the hundreds of dollars per chip. Laying these chips out in a fully-connected mesh [8, 30] makes it possible to build a non-oversubscribed modular pod switch with 1,024 10 GigE server-facing ports and 1,024 10 GigE uplinks for communication to other pods, by way of a *core* switching layer. Bandwidth through the core may be limited by some *oversubscription ratio*.

A number of recent architectures [1,9–11] can be employed to provide full bisection bandwidth among all hosts, with an oversubscription ratio of 1. For instance, a core switching layer consisting of 1,024 64-port 10 GigE switches could provide non-blocking bandwidth (655 Tb/sec) among 64 pods (each with 1,024 10 GigE uplinks) using 65,536 physical wires between the 64 pods and the core switching layer. The cost and complexity of deploying such a core switching layer and the relatively rare need for aggregate bandwidth at this scale often leads to oversubscribed networks. In our running example, a pod switch with 100 10 GigE uplinks would mean that server communication would be oversubscribed by a factor of 10, leading to 1 Gb/sec for worst-case communication patterns. One recent study of data center network deployments claimed an oversubscription ratio of 240 for GigE-connected end hosts [9].

There is an interesting subtlety regarding fully provisioned inter-pod data center network topologies. It can be claimed that provisioning full bandwidth for arbitrary all-to-all communication patterns at the scale of tens of thousands of servers is typically an overkill since not many applications run at such scale or have such continuous communication requirements. However, it is worth noting that a fully provisioned large-scale topology is required to support localized bursts of communication even between two pods as long as the set of inter-communicating pods is not fixed. In our example data center with an oversubscription ratio of 10, the topology as a whole may support 65 Tb/s of global bisection bandwidth. However, if at a particular point in time, all hosts within a pod wish to communicate with hosts in remote pods, they would be limited to 1 Tb/s of aggregate bandwidth even when there is no other communication anywhere else in the data center.

The key observation is that the available bisection bandwidth is inflexible. It cannot be allocated to the points in the data center that would most benefit from it, since it is fixed to a pre-defined topology. While supporting arbitrary all-to-all communication may not be required, supporting bursty inter-pod communication requires a non-blocking topology using traditional techniques. This results in a dilemma: unfortunately, network designers must pay for the expense and complexity of a non-blocking topology despite the fact that vast, though dynamically changing, portions of the topology will sit idle if the network designer wishes to prevent localized bottlenecks.

The goal of our work is to address this dilemma. Rather than provision for the worst case communication requirements, we wish to enable a pool of available bandwidth to be allocated when and where it is required based on dynamically changing communication patterns. As an interesting side effect, we find that the technologies we leverage can deliver even full bisection bandwidth at significantly lower cost, complexity, and energy than existing data center interconnect technologies as long as one is willing to make certain assumptions about stability in communication patterns.

2.2 Enabling Technologies

The Trend of Optics in the Data Center.

The electronics industry has standardized on silicon as their common substrate, meaning that many VLSI fabs are optimized for manufacturing Si-based semiconductors [34]. Unfortunately for optics, it is not currently possible to fabricate many important optical devices, such as lasers and photodetectors, using only Si. Optics have traditionally used more exotic group III-V compounds like GaAs and group III-V quaternary semiconductor alloys like InGaAsP [21]. These materials are more expensive to process, primarily because they cannot piggyback on the economies of scale present in the electronics market.

Transceivers for copper cables are Si-based semiconductors, and are ideal for short-reach interconnects. However, the definition of "short" has been trending downward over time. There is a fundamental trade off between the length of a copper cable and available bandwidth for a given power budget [15]. For 10 GigE, this "power wall" limits links to approximately 10 m. Longer cables are possible, though power can exceed 6 W/port, unsustainable in large-scale data centers. Fortunately, 10 m corresponds to the distance between a host and its pod switch. So for the short-term, we assume all host links inside of a pod will be copper.

Large data centers require a significant number of long links to interconnect pods, making it essential to use optical interconnects. Unfortunately, optical transceivers are much more expensive than copper transceivers. For example, an SFP+ 10 GigE optical transceiver can cost up to $200, compared to about $10 for a comparable copper transceiver. This high cost is one factor limiting the scalability and performance of modern data centers. For the fully provisioned topology we have been considering, the 65,536 optical fibers between pods and core switches would incur a cost of $26M just for the required 131,072 transceivers (not accounting for the switches or the complexity of managing such an interconnect). It is possible that volume manufacturing will drive down this price in the future, and emerging technologies such as silicon nanophotonics [26] may further reduce cost by integrating optical components on standard Si substrates. Despite these industry trends, data center networking has reached a point where the use of optics is required, and not optional.

MEMS-based Optical Circuit Switching.

A MEMS-based optical circuit switch (OCS) [24] is fundamentally different from an electrical packet switch. The OCS is a Layer 0 switch — it operates directly on light beams without decoding any packets. An OCS uses an $N \times N$ crossbar of mirrors to direct a beam of light from any input port to any output port. The mirrors themselves are attached to tiny motors, each of which is approximately 1 mm^2 [16]. An embedded control processor positions the mirrors to implement a particular connection matrix and accepts remote commands to reconfigure the mirrors into a new connection matrix. Mechanically repositioning the mirrors imposes a switching time, typically on the order of milliseconds [16].

Despite the switching-time disadvantage, an OCS possesses other attributes that prove advantageous in the data center. First, an OCS does not require transceivers since it does not convert between light and electricity. This provides significant cost savings compared to electrical packet switches. Second, an OCS uses significantly less power than an electrical packet switch. Our Glimmerglass OCS consumes 240 mW/port, whereas a 10 GigE switch such as the 48-port Arista 7148SW [30] consumes 12.5 W per port, in addition to the 1 W of power consumed by an SFP+ transceiver. Third, since an OCS does not process packets, it is data rate agnostic; as the data center is upgraded to 40 GigE and 100 GigE, the OCS need not be upgraded. Fourth, WDM (§2.2) can be used to switch an aggregate of channels simultaneously through a single port, whereas a packet switch would first have to demultiplex all of the channels and then switch each channel on an individual port.

Optical circuit switches have been commercially available for the past decade. Switches with as many as 320 ports [31] are commercially available, with as many as 1,000 ports being feasible. We assume an OCS cost of $500/port and power consumption of 240 mW/port. As a point of comparison, the 48-port 10 GigE Arista 7148SX switch has a per-port cost of $500. We summarize these values in Table 1.

Wavelength Division Multiplexing.

WDM is a technique to encode multiple non-interfering channels of information onto a single optical fiber simultaneously. WDM is used extensively in WAN networks to leverage available fiber given the cost of trenching new cables over long distances. WDM has traditionally not been used in data center networks where fiber links are shorter and the cost of deploying additional fiber is low.

Coarse WDM (CWDM) technology is less expensive than Dense WDM (DWDM) because it uses a wider channel spacing (20 nm channels across the 1270 nm - 1630 nm C-band). CWDM lasers do not require expensive temperature stabilization. However, there is little demand for CWDM transceivers because they are not compatible with erbium-doped fiber amplifiers used for long-haul communications [23]. Such amplification is not required for short data center distances. CWDM SFP+ modules are practically the same as "standard" 1310 nm SFP+ transceivers, except for a different color of laser. With sufficient volume, CWDM transceivers should also drop to about $200.

Component	Cost	Power
Packet Switch Port	$500	12.5 W
Circuit Switch Port	$500	0.24 W
Transceiver ($w \leq 8$)	$200	1 W
Transceiver ($w = 16$)	$400	1 W
Transceiver ($w = 32$)	$800	3.5 W
Fiber	$50	0

Table 1: Cost and power consumption of data center networking components.

DWDM transceivers have a much narrower channel spacing (0.4 nm across the C-band) which allows for 40 independent channels. They are more expensive because narrow channels require temperature stabilization. Finisar and others are developing DWDM SFP+ modules. We estimate they would cost about $800 in volume.

There are two major technology trends happening in optics. The CWDM and DWDM transceivers mentioned earlier use edge-emitting lasers. There also exists a competing technology called Vertical-Cavity Surface-Emitting Lasers (VCSELs) that are currently less expensive to manufacture and test. Prior research [4] has even demonstrated 4 CWDM channels using VCSELs, and 8 channels may be possible. If multiple channels could be integrated into a single chip, then the cost of deploying HELIOS might be lower than using edge-emitting laser technology. However, HELIOS is independent of technology choice and we leave exploration of dense transceiver integration to future work.

3. ARCHITECTURE

In this section, we present a model of the HELIOS architecture and analyze cost, power, complexity, and ideal bisection bandwidth trade offs. Some relevant measures of complexity include the human management overhead, physical footprint, and the number of long interconnection cables for the switching infrastructure. Due to the distances involved, we assume that all inter-pod 10 GigE links are optical (§2.2).

3.1 Overview

Fig. 1 shows a small example of the HELIOS architecture. HELIOS is a 2-level multi-rooted tree of pod switches and core switches. Core switches can be either electrical packet switches or optical circuit switches; the strengths of one type of switch compensate for the weaknesses of the other type. The circuit-switched portion handles baseline, slowly changing inter-pod communication. The packet-switched portion delivers all-to-all bandwidth for the bursty portion of inter-pod communication. The optimal mix is a trade off of cost, power consumption, complexity, and performance for a given set of workloads.

In Fig. 1, each pod has a number of hosts (labeled 'H") connected to the pod switch by short copper links. The pod switch contains a number of optical transceivers (labeled "T") to connect to the core switching array. In this example, half of the uplinks from each pod are connected to packet switches, each of which also requires an optical transceiver. The other half of uplinks from each pod switch pass through a passive optical multiplexer (labeled "M") before connecting to a single optical circuit switch. We call these superlinks, and in this example they carry 20G of capacity ($w = 2$ wavelengths). We refer to w as the size of a superlink and it is bounded by the number of WDM wavelengths supported by the underlying technology. In this paper we assume $w \in 1, 2, 4, 8, 16, 32$.

This example delivers full bisection bandwidth. However, we differentiate the bandwidth and say that 50% of the bisection bandwidth is *shared* between pods at packet timescales, and the remaining 50% is *allocated* to particular source-destination pod pairs, and can be reallocated on millisecond timescales. As long as a given workload has at least 50% of its inter-pod traffic changing over multi second timescales, it should work well with this particular mix of packet and circuit switches.

3.2 Simulation Study at Scale

We sought to understand the behavior of a large HELIOS deployment. Specifically, we wanted to know the best allocation of packet switches and circuit switches. We also wanted to know the best choice of w for superlinks. In order to answer these questions, we created a simplified model of HELIOS and implemented a simulator and a traffic generator to analyze this model.

The simulator performs a static analysis, meaning that the circuit switch is set to a particular, fixed configuration for the duration of the simulation. In this way, the simulation results will predict a lower bound to the cost, power, and complexity savings of an actual HELIOS deployment that is able to change the circuit switch configuration at runtime to match the dynamically changing workload.

3.2.1 *Simplified Model of* HELIOS

The simplified topology model consists of $N = 64$ pods, each with $H = 1,024$ hosts. All core packet switches are combined into a single packet switch with a variable number of ports, P. The same is true for the circuit switch with C ports. Instead of dynamically shifting traffic between these two switches over time, we statically assign traffic to one switch or the other. P and C expand as needed to completely encompass the communication demands of the hosts. There will be communication patterns where $C = 0$ is optimal (e.g., highly bursty, rapidly shifting traffic) and others where $P = 0$ is optimal (e.g., all hosts in one pod communicate only with hosts in exactly one other pod). We wish to explore the space in between.

Table 1 shows the component costs. We list three different types of transceivers, showing the increase in cost for transceivers that support a greater number of wavelengths. The transceiver for $w = 32$ is an XFP DWDM transceiver, which costs more and consumes more power than an SFP+, since there currently are no SFP+ DWDM transceivers.

We use a bijective traffic model which we define as follows: the number of flows a host transmits always equals the number of flows the host receives. Analysis would be complicated by non-bijective patterns since certain bottlenecks would be at end hosts rather than in the network. The traffic model consists of a variable called `num_dests`, the maximum number of destination pods a given host can communicate with throughout the course of the simulation. For simplicity and to maximize burstiness, we assume that when hosts change the destinations that they send traffic to, they do so simultaneously. We vary `num_dests` between 1 and N-1 and observe the effect on system cost, power consumption, and number of required switch ports.

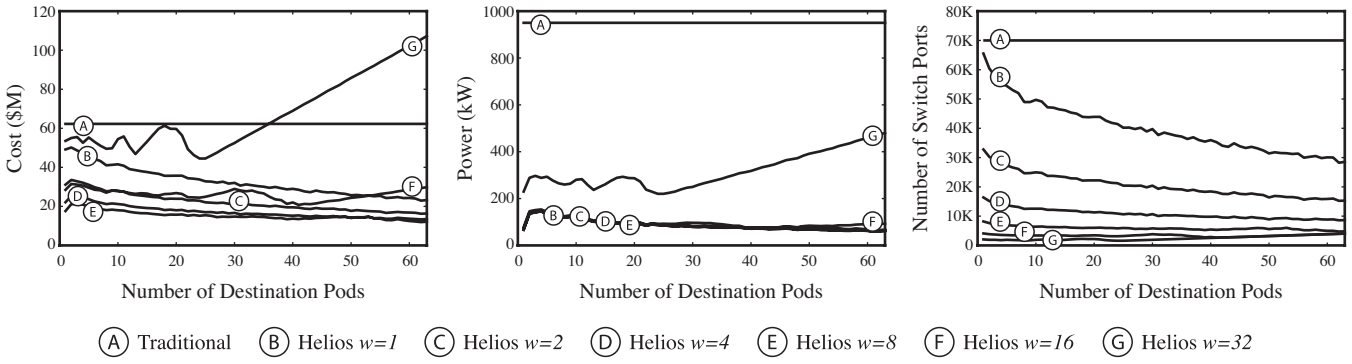

Figure 2: Simulation results comparing a traditional topology to a Helios-like system.

3.2.2 Simulation Methodology

We generate multiple communication patterns for all 63 values of `num_dests` and simulate them across the 6 topologies corresponding to different values of w. We also consider a traditional entirely-electrical switch consisting of a sufficient number of 10 GigE ports to deliver non-blocking bandwidth among all pods.

For each communication pattern, we generate a matrix of functions, $[f_{i,j}(t)]_{N\times N}$, where the value of the function is the instantaneous number of flows that pod i is sending to pod j. For each $f_{i,j}(t)$ over the simulation time interval $t \in [0, T]$, we record the minimum number of flows $m_{i,j} = min_t(f_{i,j}(t))$ and allocate this number of ports to the core circuit switch. Our rationale is as follows.

Consider a single large electrical packet switch. Each pod switch connects to the packet switch using H links. However, since we know that pod i will always send at least $m_{i,j}$ flows to pod j, we will not reduce the total bisection bandwidth by disconnecting $m_{i,j}$ of pod i's physical links from the core packet switch and connecting them directly to pod j's transceivers. Since the circuit switch consumes less power and requires fewer transceivers and fibers, it is always advantageous to send this minimum traffic over the circuit switch (assuming equal cost per port). We repeat this process for every pair of pods for a total of $M = \sum_i^N \sum_j^N m_{i,j}$ ports, transceivers, and fibers. We allocate the remaining $(NH - M)$ links to the core packet switch.

3.2.3 Simulation Results

Fig. 2 shows the simulation results for varying the number of destination pods for the different values of w to see the impact on cost, power consumption, and number of switch ports. Cost necessarily reflects a snapshot in time and we include sample values to have a reasonable basis for comparison. We believe that the cost values are conservative.

We compare these results to a traditional architecture consisting entirely of electrical packet switches. Even without the ability to dynamically change the topology during runtime, our simulation results indicate a factor of three reduction in cost ($40M savings) and a factor of 9 reduction in power consumption (800 kW savings). We also achieve a factor of 6.5 reduction in port count, or 55,000 fewer ports. Port count is a proxy for system complexity as fewer ports means less cables to interconnect, a smaller physical footprint for the switching infrastructure and less human management overhead.

Smaller values of `num_dests` result in a more expensive implementation because of the wider variation between the maximum number of flows and the minimum number of flows in $[f_{i,j}(t)]_{N\times N}$. When `num_dests` is large, these variations are minimized, reducing the number of links assigned to the more expensive core packet switch.

The parameter w influences cost both positively and negatively. Larger values of w reduce the number of fibers and core circuit switch ports, reducing cost. But larger values of w also lead to more *internal fragmentation*, which is unused capacity on a superlink resulting from insufficient demand. The effect of w on system cost is related to the number of flows between pods. In our configuration, between $w = 4$ and $w = 8$ was optimal.

Surprisingly, if communication is randomly distributed over a varying number of destination pods, even statically configuring a certain number of optical circuits can deliver bandwidth equal to a fully provisioned electric switch, but with significantly less cost, power, and complexity. Much of this benefit stems from our ability to aggregate traffic at the granularity of an entire pod. Certainly this simple analysis does not consider adversarial communication patterns, such as when all hosts in a pod synchronously shift traffic from one pod to another at the granularity of individual packets.

4. DESIGN AND IMPLEMENTATION

This section describes the design and implementation of Helios, its major components, algorithms, and protocols.

4.1 Hardware

Our prototype (Figs. 3 and 4) consists of 24 servers, one Glimmerglass 64-port optical circuit switch, three Fulcrum Monaco 24-port 10 GigE packet switches, and one Dell 48-port GigE packet switch for control traffic and out-of-band provisioning (not shown). In the WDM configuration, there are 2 superlinks from each pod and the first and second transceivers of each superlink use the 1270 nm and 1290 nm CWDM wavelengths, respectively. The 24 hosts are HP ProLiant DL380 rackmount servers, each with two quad-core Intel Xeon E5520 Nehalem 2.26 GHz processors, 24 GB of DDR3-1066 RAM, and a Myricom dual-port 10 GigE NIC. They run Debian Linux with kernel version 2.6.32.8.

The 64-port Glimmerglass crossbar optical circuit switch is partitioned into multiple (3 or 5) virtual 4-port circuit switches, which are reconfigured at runtime. The Monaco packet switch is a programmable Linux PowerPC-based plat-

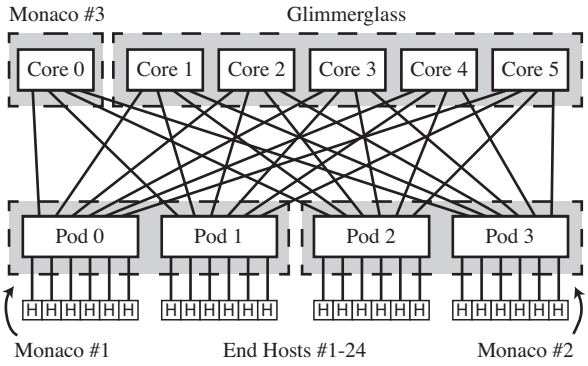

Figure 3: Helios prototype with five circuit switches.

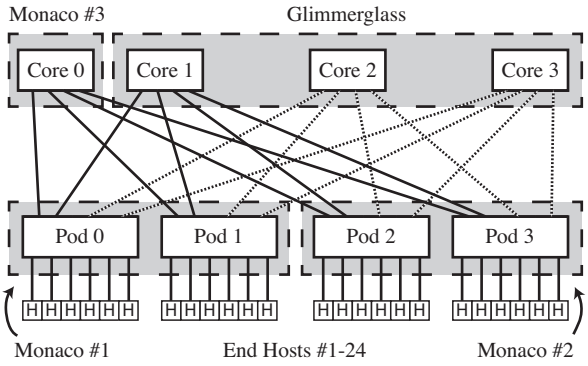

Figure 4: Helios prototype with three circuit switches. Dotted lines are $w = 2$ superlinks.

form that uses an FM4224 ASIC for the fast path. We partitioned two Monaco switches into two 12-port virtual pod switches each, with 6 downlinks to hosts, 1 uplink to a core packet switch, and 5 uplinks to core circuit switches. We used four ports of a third Monaco switch as the core packet switch. Virtualization allowed us to construct a more balanced prototype with less hardware.

4.2 Software

The Helios software consists of three cooperating components (Fig. 5): a Topology Manager (TM), a Circuit Switch Manager (CSM), and a Pod Switch Manager (PSM). The TM is the heart of Helios, monitoring dynamically shifting communication patterns, estimating the inter-pod traffic demands, and calculating new topologies/circuit configurations. The TM is a user-level process totalling 5,049 lines of Python with another 1,400 lines of C to implement the TCP fixpoint algorithm (§4.3.2) and the Edmonds maximum weighted matching algorithm (§4.3.3). In our prototype, it runs on a single server, but in practice it can be partitioned and replicated across multiple servers for scalability and fault tolerance. We discuss the details of the TM's control loop in §4.3.

The CSM is the unmodified control software provided by Glimmerglass, which exports a TL1 command-line interface. In synchronous mode, the RPC does not return until after the operation has completed. In asynchronous mode, the RPC may return before the command has completed, and a second message may be generated after the operation finally

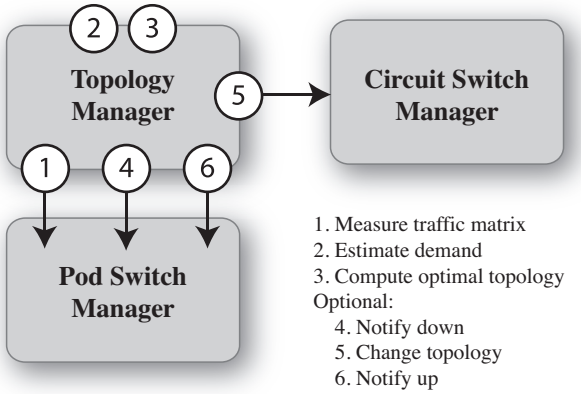

Figure 5: Helios control loop.

completes. We used synchronous mode for our evaluation. Neither mode has the ability to notify when the circuit configuration has begun to change or has finished changing.

The PSM runs on each pod switch, initializing the pod switch hardware, managing the flow table, and interfacing with the TM. We chose Layer 2 forwarding rather than Layer 3 routing for our prototype, but our design is general to both approaches. The PSM supports multipath forwarding using Link Aggregation Groups (LAGs). We assign each destination pod its own LAG, which can contain zero or more physical circuit switch uplink ports. If a LAG contains zero ports, then the PSM assigns the forwarding table entries for that destination pod to the packet switch uplink ports. The number of ports in a LAG grows or shrinks with topology reconfigurations. We use the Monaco switch's default hashing over Layer 2, 3 and 4 headers to distribute flows among ports in a LAG.

A limitation of our prototype is that a pod switch cannot split traffic over both the packet switch port as well as the circuit switch ports for traffic traveling to the same destination pod. This is due to a software restriction in the Monaco where a switch port cannot be a member of multiple LAGs.

Each PSM maintains flow counters for traffic originating from its pod and destined to a different pod. Most packet switches include TCAMs for matching on the appropriate packet headers, and SRAM to maintain the actual counters. The Monaco switch supports up to 16,384 flow counters.

The PSM is a user-level process, totalling 1,184 lines of C++. It uses the Apache Thrift RPC library to communicate with the TM. It uses Fulcrum's software library to communicate with the Monaco's FM4224 switch ASIC.

4.3 Control Loop

The Helios control loop is illustrated in Fig. 5.

4.3.1 Measure Traffic Matrix

The TM issues an RPC to each PSM and retrieves a list of flow counters. It combines them into an octet-counter matrix and uses the previous octet-counter matrix to compute the flow-rate matrix. Each flow is then classified as either a mouse flow (<15 Mb/s) or an elephant flow and the mice are removed from the matrix. This static cutoff was chosen empirically for our prototype, and more sophisticated mice-elephant classification methods [17] can also be

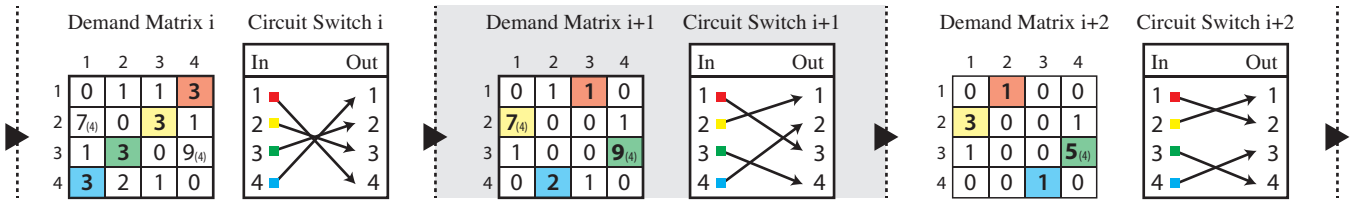

Figure 6: Example of COMPUTE NEW TOPOLOGY with 4 pods and $w = 4$.

used. Intuitively, elephant flows would grow larger if they could, whereas mice flows probably would not; and most of the bandwidth in the network is consumed by elephants, so the mice flows can be ignored while allocating circuits.

4.3.2 ESTIMATE DEMAND

The flow-rate matrix is a poor estimator for the true inter-pod traffic demand since it is heavily biased by bottlenecks in the current network topology. When used directly, we found that the topology changed too infrequently, missing opportunities to achieve higher throughput. A better demand estimator is the max-min fair bandwidth allocation for TCP flows on an ideal non-oversubscribed packet switch. These computed fixpoint values are the same values that the TCP flows would oscillate around on such an ideal switch. We use an earlier algorithm [2] to compute these fixpoint values. The algorithm takes the flow-rate matrix as input and produces a modified flow-rate matrix as output, which is then reduced down to a pod-rate (demand) matrix by summing flow rates between pairs of pods.

4.3.3 COMPUTE NEW TOPOLOGY

Our goal for computing a new topology is to maximize throughput. We formulate this problem as a series of max-weighted matching problems on bipartite graphs, illustrated with an example in Fig. 6. Each individual matching problem corresponds to particular circuit switch, and the bipartite graph used as input to one problem is modified before being passed as input to the next problem. The first set of graph vertices (rows) represents source pods, the second set (columns) represents destination pods, and the directed edge weights represent the source-destination demand or the superlink capacity of the current circuit switch (in parentheses), whichever is smaller. The sum of the edge weights of a particular maximum-weighted matching represents the expected throughput of that circuit switch. Note that we configure circuits unidirectionally, meaning that having a circuit from pod i to pod j does not imply the presence of a circuit from pod j to pod i, as shown in Fig. 6, stage $i + 1$.

We chose the Edmonds algorithm [7] to compute the max-weighted matching because it is optimal, runs in polynomial time ($O(n^3)$ for our bipartite graph), and because there are libraries readily available. Faster algorithms exist; however as we show in §5.6, the Edmonds algorithm is fast enough for our prototype.

4.3.4 NOTIFY DOWN, CHANGE TOPOLOGY, NOTIFY UP

If the new topology is different from the current topology, then the TM starts the reconfiguration process. First, the TM notifies the PSMs that certain circuits will soon be disconnected. The PSMs act on this message by removing the affected uplinks from their current LAGs, and migrating

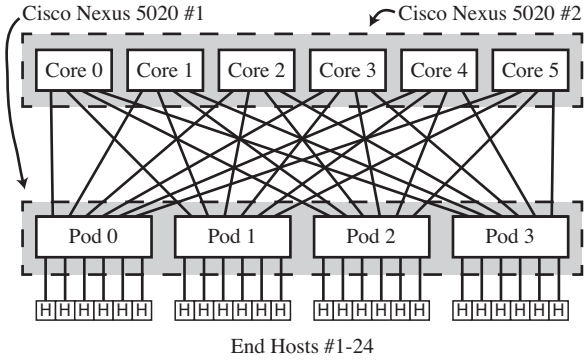

Figure 7: Traditional multi-rooted network of packet switches.

forwarding entries to the core packet switch uplinks if required. This step is necessary to prevent pod switches from forwarding packets into a black hole while the topology is being reconfigured. Second, the TM instructs the CSMs to actually change the topology. Third, the TM notifies the PSMs that the circuits have been re-established (but to different destination pods). The PSMs then add the affected uplinks to the correct LAGs and migrate forwarding entries to the LAGs if possible.

During both NOTIFY DOWN and NOTIFY UP, it is possible for some packets to be delivered out of order at the receiver. However, our experiments did not show any reduction in throughput due to spurious TCP retransmissions.

5. EVALUATION

We evaluated the performance of HELIOS against a fully-provisioned electrical packet-switched network arranged in a traditional multi-rooted tree topology (Fig. 7). We expect this traditional multirooted network to serve as an upper bound for our prototype's performance, since it does not have to pay the overhead of circuit reconfiguration. We partitioned one Cisco Nexus 5020 52-port 10 GigE switch into four virtual 12-port pod switches, with one virtual switch per untagged VLAN. All six uplinks from each pod switch were combined into a single Link Aggregation Group (EtherChannel), hashing over the Layer 2 and 3 headers by default. We partitioned a second such switch into six virtual 4-port core switches.

5.1 Communication Patterns

In this section, we describe the communication patterns we employed to evaluate HELIOS. Since no data center network benchmarks have yet been established, we initially tried running communication-intensive applications such as

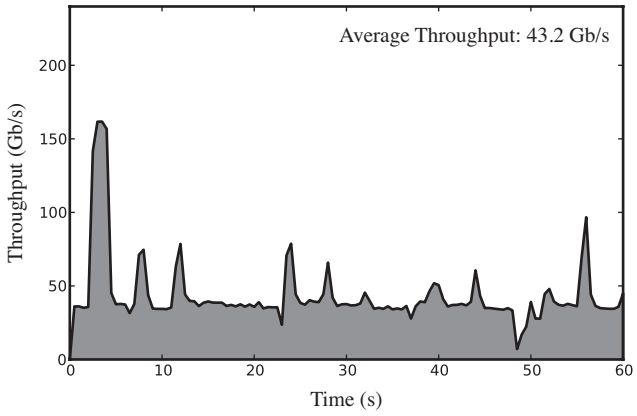

Figure 8: PStride (4s) on Helios using the baseline Fulcrum software.

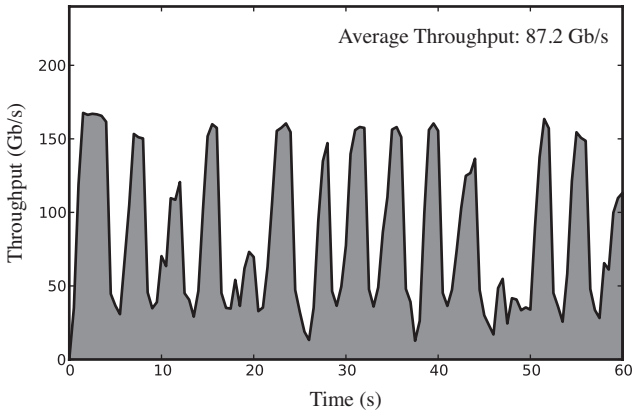

Figure 9: PStride (4s) on Helios after disabling the 2s "debouncing" feature.

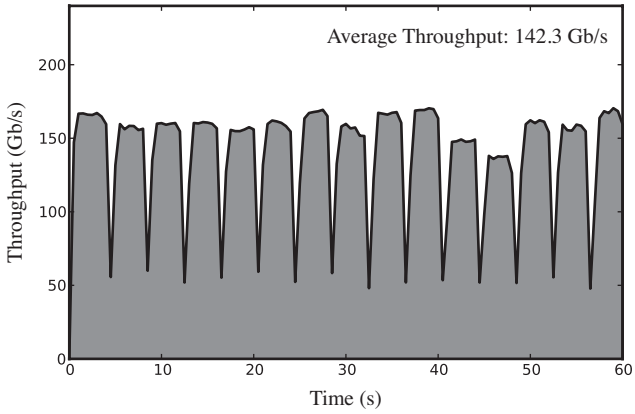

Figure 10: PStride (4s) on Helios after disabling both the 2s "debouncing" feature as well as EDC.

Hadoop's Terasort. We found that Hadoop only achieved a peak aggregate throughput of 50 Gb/s for both configurations of HELIOS and the experimental comparison network. We sought to stress HELIOS beyond what was possible using Hadoop, so we turned our attention to synthetic communication patterns that were not CPU or disk I/O bound. Each pattern is parameterized by its *stability*, the lifetime in seconds of a TCP flow. After the stability period, a new TCP flow is created to a different destination. In all communication patterns, we had the same number of simultaneous flows from each host to minimize differences in hashing effects.

Pod-Level Stride (PStride). Each host in a source pod i sends 1 TCP flow to each host in a destination pod $j = (i+k) \bmod 4$ with k rotating from 1 to 3 after each stability period. All 6 hosts in a pod i communicate with all 6 hosts in pod j; each host sources and sinks 6 flows simultaneously. The goal of PStride is to stress the responsiveness of the TM's control loop, since after each stability period, the new flows can no longer utilize the previously established circuits.

Host-Level Stride (HStride). Each host i (numbered from 0 to 23) sends 6 TCP flows simultaneously to host $j = (i + 6 + k) \bmod 24$ with k rotating from 0 to 12 after each stability period. The goal of HStride is to gradually shift the communication demands from one pod to the next. HStride's pod-to-pod demand does not change as abruptly as PStride's and should not require as many circuits to be reconfigured at once. Intuitively, the aggregated pod-to-pod demand is more stable than individual flows.

Random. Each host sends 6 TCP flows simultaneously to one random destination host in a remote pod. After the stability period, the process repeats and different destination hosts are chosen. Multiple source hosts can choose the same destination host, causing a hotspot.

5.2 Debouncing and EDC

While building our prototype, our initial measurements showed very poor performance, with throughput barely above that of the core packet switch used in isolation, when stability is less than 4 seconds. Fig. 8 shows a PStride pattern with a stability of 4 seconds. Upon closer inspection, we found phases of time where individual TCP flows were idle for more than 2 seconds.

We identified *debouncing* as the source of the problem. Debouncing is the technique of cleaning up a signal generated from a mechanical connection. When plugging a cable into a switch, the link goes up and down rapidly over a short time frame. Without debouncing, there is a potential for these rapid insertion and removal events to invoke multiple expensive operations, such as causing a routing protocol to issue multiple broadcasts.

Debouncing works by detecting a signal and then waiting a fixed period of time before acting on the signal. In case of the Monaco switch, it was waiting two seconds before actually enabling a switch port after detecting the link. Since switch designers typically assume link insertion and removal are rare, they do not optimize for debouncing time. However, in HELIOS, we rapidly disconnect and reconnect ports as we dynamically reconfigure circuits to match shifting communication patterns.

We programmed Fulcrum's switch software to disable this functionality. The result was a dramatic performance improvement as shown in Fig. 9. The periods of circuit switch reconfiguration were now visible, but performance was still below our expectations.

After many measurements, we discovered that the Layer 1 PHY chip on the Monaco, a NetLogic AEL2005, took 600 ms after a circuit switch reconfiguration before it was usable. Most of this time was spent by an *Electronic Dispersion Compensation (EDC)* algorithm, which removes the noise

introduced by light travelling over a long strand of fiber. We disabled EDC and our system still worked correctly because we were using relatively short fibers and transceivers with a relatively high power output. After disabling EDC, performance again improved (Fig. 10). Now each change in flow destinations and corresponding circuit switch reconfiguration was clearly visible, but the temporary periods of lost capacity during circuit reconfigurations were much shorter. Even with EDC disabled, the PHY still took 15ms after first light before it was usable. In general, our work suggests opportunities for optimizing PHY algorithms such as EDC for the case where physical topologies may change rapidly.

One problem with multi-rooted trees, including Helios, is that throughput is lost due to hotspots: poor hashing decisions over the multiple paths through the network. This can be seen in Fig. 10 where the heights of the pillars change after every stability period. These types of hashing effects should be less pronounced with more flows in the network. Additionally, complementary systems such as Hedera [2] can be used to remap elephant flows to less congested paths.

5.3 Throughput versus Stability

Having considered some of the baseline issues that must be addressed in constructing a hybrid optical/electrical switching infrastructure, we now turn our attention to Helios performance as a function of communication characteristics. The most important factor affecting Helios performance is inter-pod traffix matrix stability, which is how long a pair of pods sustains a given rate of communication. Communication patterns that shift too rapidly would require either too many under-utilized circuits or circuit-switching times not available from current technology.

Fig. 11 shows the throughput delivered by Helios for the communication patterns in §5.1. Each bar represents the mean across 5 trials of the average throughput over 60 second experiments. We varied the stability parameter from 0.5 seconds up to 16 seconds. First, we observed that higher stability leads to higher average throughput because more stable communication patterns require fewer circuit switch reconfigurations. Second, the throughput achieved by Helios with WDM is comparable to the throughput achieved without WDM. This indicates that WDM may be a good way to reduce the cost, power consumption, and cabling complexity of a data center network without hurting performance for certain communication patterns. Third, for the same value of stability, throughput is generally better for HStride compared to PStride since the inter-pod traffic matrix is more stable. This suggests that even for low stability of individual flows, Helios can still perform well.

5.4 Unidirectional Circuits

We designed Helios to use either unidirectional circuits or bidirectional circuits. If port A in pod 1 connects to port B in pod 2, then bidirectional circuits would have port B in pod 2 connected back to port A in pod 1 through another circuit. Traditional approaches to establishing circuits employ bidirectional circuits because of the assumption of full duplex communication. However, unidirectional circuits have no such constraint and can better adapt to asymmetric traffic demands. For example, when transferring a large index from one pod to another, most of the traffic will flow in one direction.

This is illustrated in Fig. 12, which shows throughput for

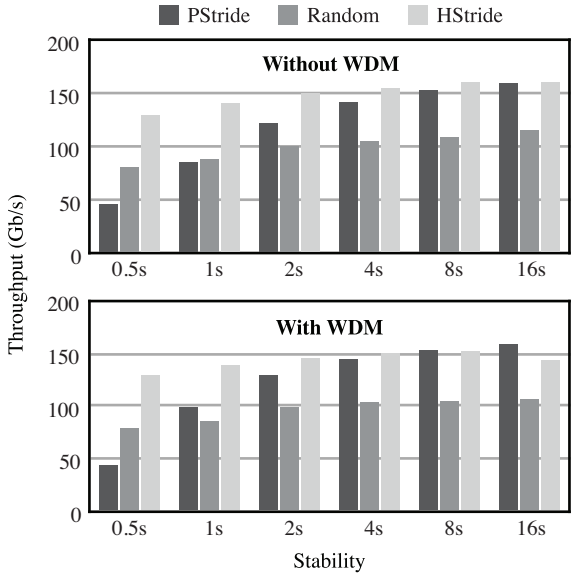

Figure 11: Throughput as a function of stability.

a PStride communication pattern with 4 seconds of stability. Throughput with unidirectional circuits closely matches that of the traditional network during the stable periods, whereas bidirectional circuit scheduling underperforms when the pod-level traffic demands are not symmetric. During intervals with asymmetric communication patterns, bidirectional circuit scheduling is only half as efficient as unidirectional scheduling in this experiment. The higher throughput of the traditional network comes from two factors: first, our prototype does not split traffic between circuit switches and packet switches, and second, the traditional network does not suffer from temporary capacity reductions due to circuit switch reconfigurations. All other Helios results in this paper use unidirectional circuits.

The 10G Ethernet standard includes a feature that complicates unidirectional circuits. If a Layer 1 receiver stops receiving a valid signal, it shuts off its paired Layer 1 transmitter. The assumption is that all links are bidirectional, so the loss of signal in one direction should disable both sides of the link in order to simplify fault management in software. In practice, we did not see any performance degradation from this feature since we reconfigure a set of circuits in parallel, and since we keep all links active. However, this feature increases the size of the failure domain: if one transceiver fails, all other transceivers in the same daisy-chained cycle are disabled. This problem can be solved by having the TM connect a failed transceiver in loopback, thus preventing it from joining a cycle with other transceivers.

5.5 How Responsive is Helios?

The length of the control loop is the most important factor in determining how quickly Helios can react to changing communication patterns. As discussed in §4.3, the control loop is composed of three mandatory stages (1-3) executed every cycle, and three optional stages (4-6) executed only when changing the topology (see Fig. 5). Measurements of our prototype, summarized in Fig. 13, show that the mandatory portion of a cycle takes approximately 96.5 ms, whereas the optional portion takes an additional 168.4 ms.The two