c-Through: Part-time Optics in Data Centers

Guohui Wang[*], David G. Andersen[†], Michael Kaminsky[‡], Konstantina Papagiannaki[‡],
T. S. Eugene Ng[*], Michael Kozuch[‡], Michael Ryan[‡]
[*]Rice University, [†]Carnegie Mellon University, [‡]Intel Labs Pittsburgh

ABSTRACT

Data-intensive applications that operate on large volumes of data have motivated a fresh look at the design of data center networks. The first wave of proposals focused on designing pure packet-switched networks that provide full bisection bandwidth. However, these proposals significantly increase network complexity in terms of the number of links and switches required and the restricted rules to wire them up. On the other hand, optical circuit switching technology holds a very large bandwidth advantage over packet switching technology. This fact motivates us to explore how optical circuit switching technology could benefit a data center network. In particular, we propose a hybrid packet and circuit switched data center network architecture (or HyPaC for short) which augments the traditional hierarchy of packet switches with a high speed, low complexity, rack-to-rack optical circuit-switched network to supply high bandwidth to applications. We discuss the fundamental requirements of this hybrid architecture and their design options. To demonstrate the potential benefits of the hybrid architecture, we have built a prototype system called c-Through. c-Through represents a design point where the responsibility for traffic demand estimation and traffic demultiplexing resides in end hosts, making it compatible with existing packet switches. Our emulation experiments show that the hybrid architecture can provide large benefits to unmodified popular data center applications at a modest scale. Furthermore, our experimental experience provides useful insights on the applicability of the hybrid architecture across a range of deployment scenarios.

Categories and Subject Descriptors

C.2.1[**Network Architecture and Design]:** Network topology, Packet switching networks, Circuit switching networks

General Terms

Design, Experimentation, Performance

Keywords

Data center networking, optical circuit switching, hybrid network

1. INTRODUCTION

The rising tide of data-intensive, massive scale cluster computing is creating new challenges for datacenter networks. In this paper, we explore the question of integrating optical circuit-switched technologies into this traditionally packet-switched environment to create a "HyPaC network"—Hybrid Packet and Circuit—asking both *how* and *when* such an approach might prove viable. We ask this question in the hope of being able to identify a solution that combines the best of both worlds, exploiting the differing characteristics of optical and electrical switching: optics provides higher bandwidth, but suffers slower switching speed. Our results suggest in particular that data-intensive workloads such as those generated by MapReduce, Hadoop, or Dryad are sufficiently latency-insensitive that much of their traffic can be carried on slowly-switching paths. Our results both raise many questions for future work regarding the design details for hybrid networks, and motivate the need to answer those questions by showing that such designs are feasible, exploring the application characteristics that render them so, and providing a first-cut design to exploit them.

To understand these goals, first consider today's hierarchical, electrically-switched datacenter networks. They typically place 10–40 servers in a rack, with an aggregation ("Top of Rack", or ToR) switch in each. The ToR switches are the leaves in a tree of Ethernet switches that connects all of the racks. The bandwidth at the top of the tree is typically a fraction of the incoming capacity, creating a bottleneck. In response to this now-well-known limitation, the research community has begun exploring novel interconnect topologies to provide high bisection bandwidth using commodity Ethernet switches—examples include Fat trees [12, 34, 25], DCell [26], and BCube [27], among a rapidly growing set of alternatives, many adapted from earlier solutions from the telecom and supercomputing areas. These new designs provide optimal switching capacity, but they require a large number of links and switches. For example, a k-level fat tree used to connect N servers needs at least $N \times k$ switch ports and wires. While research is ongoing in this area, physically constructing these topologies at present requires complex, structured wiring, and expanding the networks after construction is challenging.

For many years, optical circuit switching has led electrical packet switching in high bandwidth transmission. A single optical fiber can carry hundreds of gigabits per second. However, this capacity must be allocated between a source and a destination at coarse granularity—much longer than the duration of a single packet transmission. "All-optical" packet switched networks have been a goal as elusive as they are important; despite numerous innovations, today's commodity optical switching technologies require on the order of milliseconds to establish a new circuit. They provide high bandwidth, but cannot provide full bisection bandwidth at the packet granularity.

Why, therefore, do we believe optical circuit switching is worth considering in data centers? Because the characteristics of many data-centric workloads suggest that many data center applications may not need full bisection bandwidth at the *packet granularity*. Recent measurement studies show substantial traffic "concentration" in data center workloads: In many scientific computing applications *"the bulk of inter-processor communication was bounded in degree*

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

and changed very slowly" [14]. Microsoft researchers measured the traffic characteristics of production data centers, finding *"evidence of ON-OFF traffic behavior"* [16] and *"only a few ToRs are hot and most of their traffic goes to a few other ToRs"* [29]. As we expand upon later, these patterns and others require high bandwidth, but the concentration of traffic makes it more suited to a network in which at any time, only a subset of the paths are accelerated.

In this work, our goal is to develop an architecture that can exploit these more fundamental differences between packet-switched electrical networks and circuit-switched optical networks (or, more generally, any circuit-switched technology with a bandwidth advantage) in the context of a modern datacenter. We only give a brief discussion of the relative cost of these networks: we believe it is a question that follows those that we ask in this paper about feasibility and the technical requirements to their use.[1] Parallel work [22] provides a more detailed discussion on the cost issue. This paper presents a first effort exploring this design space and makes the following contributions:

(1) We present a hybrid packet and circuit switched data center network architecture (or HyPaC for short) that augments a traditional electrical packet switch hierarchy with a second, high-speed *rack-to-rack* circuit-switched optical network. The optical network is reconfigured relatively slowly compared to the per-packet electrical switches, connecting at any point in time each rack to exactly one other rack. As a result, pairs of racks experience transient high capacity links, and the set of racks that are paired changes over time based upon traffic demand. Among numerous design choices that realize the HyPaC architecture, we present a prototype system called c-Through in which the responsibility for traffic demand estimation and traffic demultiplexing resides in end hosts, making it compatible with existing packet switches. In order to make the best use of the transient high capacity optical circuits, c-Through recruits servers to buffer traffic so as to collect sufficient volumes for high speed transmission. Traffic buffering is done by enlarging individual socket buffer limits so as to do it without introducing head-of-line blocking or extra delay. By performing this buffering in-kernel, we explore the benefits of the HyPaC architecture without substantial application modification. Experiments with a c-Through prototype on an emulated testbed show that optical circuits can be integrated efficiently with traditional Ethernet and TCP/IP based protocols. While we emphasize that there are many other ways that optical circuit switching might be used to enhance data center networks, the c-Through design shows that the general approach is feasible, even without modifying applications or Ethernet switches.

(2) By experimenting with different kinds of applications over the c-Through prototype, we provide insight into how the HyPaC architecture applies to several usage scenarios. We find that HyPaC can benefit many kinds of applications, but particularly those with bulk transfer components, skewed traffic patterns, and loose synchronization. We provide guidelines on how to maximize the benefits of the HyPaC architecture in large scale data centers.

The rest of this paper is organized as follows: Section 2 provides background on optical technologies. We discuss the HyPaC architecture and present the c-Through system design in Section 3 and Section 4. We evaluate the system and application performance on c-Through prototype in Section 5 and Section 6. Section 7 discusses the applicability and scalability of the HyPaC architecture. We introduce the related work in Section 8 and conclude in Section 9.

2. OPTICS: PRO AND CON

Optical circuit switching can provide substantially higher bandwidth than electrical packet switching. Today's fastest switches and routers are limited to roughly 40Gb/s per port; in contrast, 100Gb/s optical links have been developed [8], and WDM techniques can multiplex terabits/s onto a single fiber in the lab [9]. Today's market already offers 320x320 optical circuit switches with 40Gb/s transceivers [1]. The cost it pays is requiring about 20ms to switch to a new mapping of input ports to output ports. In contrast, the CRS-1 router from Cisco supports only sixteen 40Gb/s line cards in a full-rack unit—but, of course, provides packet-granularity switching.

Slow switching is a lasting challenge for optical networking, and affects all commercially available technologies. MEMS (Micro-Electro-Mechanical Systems) optical switches reconfigure by physically rotating mirror arrays that redirect carrier laser beams to create connections between input and output ports. The *reconfiguration time* for such devices is a few milliseconds.[2] Tunable lasers combined with an Arrayed Waveguide Grating Router (AWGR) can potentially provide faster switching. Tunable lasers can switch channels in tens of nanoseconds, with a significant caveat: "dark tuning". To avoid spilling garbage into the network during switching, the laser must be optically isolated from the network. With the practical constraints involved, the best switching speeds available today are still in the 1 to 10ms range [17]. Tunable lasers will likely switch more rapidly in the future, which would likely improve the performance provided by a system such as c-Through, but, of course, such improvements over time must also be balanced against the increased number of packets per second transmitted on the links.

Optics has traditionally been viewed as more expensive than its electrical counterparts, but this gap has been narrowing over time, particularly as newer Ethernet standards require increasingly heavy and expensive cables (such as CX4) for high bandwidth over even modest distances. For example, the price of optical transceivers has dropped more than 90 percent in the last decade [28], and the price of MEMS optical switches has dropped to a few hundred dollars per port [6]. Even at recent prices, the cost of optical networking components is comparable with existing solutions. For example, it has been estimated that constructing a BCube with 2048 servers costs around $92k for switches and NICs and requires 8192 wires [27]. Today, MEMS switches are mostly aimed at the low-volume, high-margin test and measurement market, but even so, using a MEMS switch to connect 52 48-port switches, each switch connecting 40 servers, would cost approximately $110k at most (On an 80-port MEMS optical switch, each port costs $200-$700, single 10Gbit optical transceiver modules cost under $350, and 48-port switches cost under $700). We expect the cost of these switches would drop substantially were they to be used in commodity settings, and the cost of the transceivers drops continuously. The increasing bandwidth demand and dropping prices of optical devices make optical circuits a viable choice for high bandwidth transmission in data centers.

3. HYPAC NETWORK REQUIREMENTS

Figure 1 depicts the HyPaC configuration we use in the rest of this paper: The packet-switched network (top) uses a traditional hierarchy of Ethernet switches arranged in a tree. The circuit-switched network (bottom) connects the top-of-rack switches. Optically connecting racks instead of nodes reduces the number of (still expensive) optical components required, but can potentially still provide high

[1] This question also falls outside our bailiwick: Price is very sensitive to volume and market conditions that may change drastically if optical technologies were to become widely deployed in the datacenter.

[2] e.g., opneti's 1x8 MEMS switch requires 2ms typical, 5ms max to switch with multi-mode fiber. `http://www.opneti.com/right/1x8switch.htm`

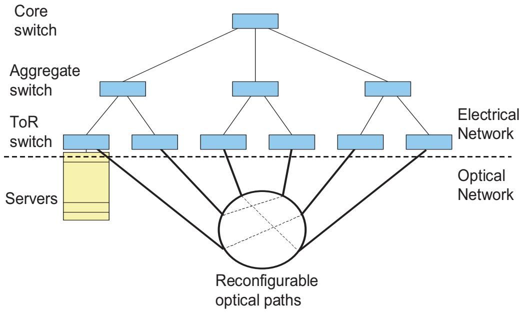


Figure 1: HyPaC network architecture

	System requirements
Control plane	1. Estimating cross-rack traffic demands 2. Managing circuit configuration
Data plane	1. De-multiplexing traffic in dual-path network 2. Maximizing the utilization of circuits when available (optimization)

Table 1: Fundamental requirements of HyPaC architecture.

capacity because a single optical path can handle tens of servers sending at full capacity over conventional gigabit Ethernet links.

The circuit-switched network can only provide a *matching* on the graph of racks: Each rack can have at most one high-bandwidth connection to another rack at a time. The switch can be reconfigured to match different racks at a later time; as noted earlier, this reconfiguration takes a few milliseconds, during which time the fast paths are unusable. To ensure that latency sensitive applications can make progress, HyPaC retains the packet-switched network. Any node can therefore talk to any other node at any time over potentially over-subscribed packet-switched links.

For the circuits to provide benefits, the traffic must be "pair-wise concentrated"—there must exist pairs of racks with high bandwidth demands between them and lower demand to others. Fortunately, such concentration has been observed by numerous prior studies [14, 16, 29]. This concentration exists for several reasons: time-varying traffic, biased distributions, and—our focus in later sections—amenability to batching. First, applications whose traffic demands vary over time (e.g. hitting other bottlenecks, multi-phase operation) can contribute to a non-uniform traffic matrix. Second, other applications have intrinsic communication skew in which most nodes only communicate with a small number of partners. This limited out-degree leads to concentrated communication. Finally, latency-insensitive applications such as MapReduce-style computations may be amenable to batched data delivery: instead of sending data to destinations in a fine-grained manner (e.g., 1, 2, 3, 2, 3, 1, 2), sufficient buffering can be provided to batch this delivery (1, 1, 2, 2, 2, 3, 3). These patterns do not require arbitrary full-bisection capacity.

3.1 System Requirement

Table 1 summarizes functions needed for a generic HyPaC-style network. In the *control plane*, effective use of the circuit-switched paths requires determining rack-to-rack traffic demands and timely circuit reconfiguration to match these demands.

In the *data plane*, a HyPaC network has two properties: First, when a circuit is established between two racks, there exist two paths between them—the circuit-switched link and the always-present packet-switched path. Second, when the circuits are reconfigured, the network topology changes. Reconfiguration in a large data center causes hundreds of simultaneous link up/down events, a level of dynamism much higher than usually found in data centers. A HyPaC network therefore requires traffic control mechanisms to dynamically de-multiplex traffic onto the circuit or packet switched network, as appropriate. Finally, if applications do not send traffic rapidly enough to fill the circuit-switched paths when they become available, a HyPaC design may need to implement additional mechanisms, such as extra batching, to allow them to do so.

3.2 Design Choices and Trade-offs

These system requirements can be achieved on either end-hosts or switches. For designs on end-hosts, the system components can be at different software layers (e.g, applications layer or kernel layer).

Traffic demand estimation: One simple choice is to let applications explicitly indicate their demands. Applications have the most accurate information about their demands, but this design requires modifying applications. As we discuss in Section 4, our c-Through design estimates traffic demand by increasing the per-connection socket buffer sizes and observing end-host buffer occupancy at runtime. This design requires additional kernel memory for buffering, but is transparent to applications and does not require switch changes. The Helios design [22], in contrast, estimates traffic demands at switches by borrowing from Hedera [13] an iterative algorithm to estimate traffic demands from flow information.

Traffic demultiplexing: Traditional Ethernet mechanisms handle multiple paths poorly. Spanning tree, for example, will block either the circuit-switched or the packet-switched network instead of allowing each to be used concurrently. The major design choice in traffic demultiplexing is between emerging link-layer routing protocols [33, 30, 25, 34, 10] and partition-based approaches that view the two networks as separate.

The advantage of a routing-based design is that, by treating the circuit and packet-switched networks as a single network, it operates transparently to hosts and applications. Its drawback is that it requires switch modification, and most existing routing protocols impose a relatively long convergence time when the topology changes. For example, in link state routing protocols, re-convergence following hundreds of simultaneous link changes could require seconds or even minutes [15]. To be viable, routing-based designs may require further work in rapidly converging routing protocols.

A second option, and the one we choose for c-Through, is to isolate the two networks and to de-multiplex traffic at either the end-hosts or at the ToR switches. We discuss our particular design choice further in Section 4. The advantage of separating the networks is that rapid circuit reconfiguration does not destabilize the packet-switched network. Its drawback is a potential increase in configuration complexity.

Circuit utilization optimizing, if necessary, can be similarly accomplished in several ways. An application-integrated approach could signal to applications to increase their transmission rate when the circuits are available; the application-transparent mechanism we choose for c-Through is to buffer additional data in TCP socket buffers, relying on TCP to ramp up quickly when bandwidth becomes available. Such buffering could also be accomplished in the ToR switches.

In the remainder of this paper, we do not attempt to cover all of the possible design choices for constructing a HyPaC network. The following section introduces the c-Through design, which represents one set of choices, and demonstrates that the HyPaC architecture can be feasibly implemented in today's datacenters without the need to modify switches or applications.

4. C-THROUGH DESIGN AND IMPLEMENTATION

c-Through[3] is a HyPaC network design that recruits end-hosts to perform traffic monitoring, and uses a partition approach to separate the circuit (optical) and packet (electrical) networks. c-Through addresses all the architectural requirements outlined in Table 1. In our current testbed, we emulate a circuit switch's connectivity properties with a conventional packet switch. We therefore omit the implementation details on how the c-Through control software interfaces with a commercial circuit switch. However, existing interfaces for circuit configuration should be usable easily.

4.1 Managing Optical Paths

Traffic measurement: c-Through estimates rack-to-rack traffic demands in an application-transparent manner by increasing the per-connection socket buffer limit and observing per-connection buffer occupancy at runtime. This approach has two benefits: First, an application with a lot of data to send will fill its socket buffer, allowing us to identify paths with high demand. Second, as discussed below, it serves as the basis for optimizing use of the circuits.

The use of TCP socket buffers ensures that data is queued on a per-flow basis, thus avoiding head-of-line blocking between concurrent flows. A low-bandwidth, latency-sensitive control flow will therefore not experience high latency due to high-bandwidth data flows.

We buffer at end hosts, not switches, so that the system scales well with increasing node count (DRAM at the end hosts is relatively cheap and more available than on the ToR switches). Each server computes for each destination rack the total number of bytes waiting in socket buffers and reports these per-destination-rack demands to the optical manager.

Utilization optimization: Optical circuits take time to set up and tear down. c-Through buffers data in the hosts' socket buffers so that it can batch traffic and fill the optical link when it is available. As we show in Section 5, buffering a few tens to hundreds of megabytes of data at end hosts increases network utilization and application performance. The end-host TCP can ramp up to fill the increased available bandwidth quickly after the network has been reconfigured.

Optical configuration manager: The optical configuration manager collects traffic measurements, determines how optical paths should be configured, issues configuration directives to the switches, and informs hosts which paths are optically connected. The initial design of this component is a small, central manager attached to the optical switch (equivalent to a router control plane).

Given the cross-rack traffic matrix, the optical manager must determine how to connect the server racks by optical paths in order to maximize the amount of traffic offloaded to the optical network. This can be formulated as a maximum weight perfect matching problem. The cross rack traffic matrix is a graph $G = (E, V)$. V is the vertex set in which each vertex represents one rack and E is the edge set. The weight of an edge e, $w(e)$, is the traffic volume between the racks. A *matching* M in G is a set of pairwise non-adjacent edges. That is, no two edges share a common vertex. A *perfect matching* is a matching that matches all vertices of the graph. From this formulation, the optical configuration is a perfect matching with the maximum aggregated weight. The solution can be computed in polynomial time by Edmonds' algorithm [20]. As we previously reported, Edmonds' algorithm is fast [39], computing the configuration of 1000 racks within a few hundred milliseconds.[4]

4.2 Traffic De-multiplexing

VLAN based network isolation: c-Through solves the traffic de-multiplexing problem by using *VLAN-based routing* (similar ideas using different mechanisms have been used to leverage multiple paths in Ethernet networks [37, 41]). c-Through assigns ToR switches two different VLANs that logically isolate the optical network from the electrical network. VLAN-s handles packets destined for the packet-switched electrical network, and VLAN-c handles packets going directly to one other rack via the optical path. In c-Through, the end-host is responsible for tagging packets with the appropriate VLAN ID, for ease of deployment.

In this topology, observe that the topology of VLAN-s (packet-switched) does not change frequently, but that of VLAN-c could change a few times per second. Rapid reconfiguration is challenging for many Ethernet control protocols, such as spanning tree or other protocols with long convergence time [21]. Therefore, spanning tree protocol should be disabled in VLAN-c and c-Through guarantees that the optical network is loop-free (by construction, it can provide only a matching). However, either existing or future protocols can still be used to manage the hierarchical electrical network.

Other designs are certainly viable depending on the technological constraints of the implementer: we do not believe that end-host VLAN selection is the only, or even the best, way to accomplish this goal in the long term. Our focus is more on demonstrating the fundamental feasibility of HyPaC networks for datacenters. Future advances in routing protocols and programmable switches (e.g., Click or OpenFlow) could provide a transparent, switch-based mechanism for traffic demultiplexing.

Traffic de-multiplexing on hosts: Each host runs a management daemon that informs the kernel about the inter-rack connectivity. The kernel then de-multiplexes traffic to the optical and electrical paths appropriately. As shown in Figure 2, each server controls its outgoing traffic using a per-rack output traffic scheduler. When TCP/IP transmits a packet, it goes to a destination-rack classifier, and is placed into a small (several packets) queue. Based upon their destination rack, the packets are then assigned either to the electrical or optical VLAN output queue (this queue is small—approximately the size of the Ethernet output buffer). Broadcast/multicast packets are always scheduled over the electrical network.

In this design, c-Through must give higher transmission priority to the optically-connected destinations than to the electrically-connected destinations. It does so using a weighted round-robin policy to assign weight 0.9 to the per-rack queue that is currently optically connected, and splits the remaining 0.1 weight among the non-optically-connected links. Because the policy is work-conserving and most traffic is running over TCP, this simple approach works well to ensure that both the optical and electrical networks are highly utilized and no flow is starved.

[3] A conjunction of c, the speed of light, and "cut-through".

[4] At larger scales or if switching times drop drastically, faster but slightly heuristic algorithms such as iSLIP could be used [32].

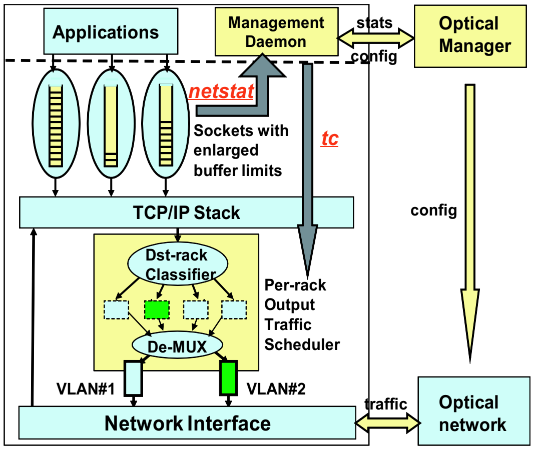


Figure 2: The structure of the optical management system

4.3 c-Through System Implementation

c-Through implements traffic measurement and control in-kernel to make the system transparent to applications and easy to deploy. Figure 2 shows the architecture of the optical management system.

Two VLAN interfaces are configured on the physical NIC connected to the ToR switch. All packets sent through a VLAN interface are tagged with the associated VLAN number. As illustrated in Figure 2, packets going through interface VLAN#1 are tagged with VLAN#1, and forwarded via electrical paths by the ToR switch; packets sent out on interface VLAN#2 are tagged with VLAN#2 and forwarded via optical paths. Each server is configured with two virtual interfaces, bonded through a bonding driver. This approach is therefore transparent to the upper layers of the stack—the two virtual interfaces have the same MAC and IP addresses.

c-Through buffers traffic by enlarging the TCP socket buffer limits, allowing applications to push more data into the kernel. We use *netstat* to extract the buffered data sizes of all sockets and then sum them based on the destination rack.

In practice, the memory consumption from enlarged socket buffer limits is moderate. The buffers need not be huge in order to infer demand and to batch traffic for optical transfer: our results in Section 6 show that limiting the TCP send buffer to 100 MB provides good performance for many applications. Furthermore, the limit is not a lower bound: only as much memory is consumed as the application generates data to fill the socket buffer. The applications we observed do not generate unbounded amounts of outstanding data. For the data intensive applications we have tried, such as VM migration, MapReduce and MPI FFT, the total memory consumption of all socket buffers on each server rarely goes beyond 200MB.

A user-space management daemon on each node reads socket statistics using *netstat* and reports them to the central configuration manager. The statistics report how much traffic is buffered for each destination rack, permitting the optical configuration manager to assign an optimal configuration of the optical paths. To reduce the amount of traffic sent to the centralized manager, servers only report traffic statistics when the queue size is larger than a threshold and the queue size variation exceeds a second threshold. We empirically set both thresholds to 1MB. Since optical paths are configured for applications with high bandwidth demands, omitting a small amount of buffered data will not significantly impact the configuration decision.

When a new configuration is computed, the manager sends reconfiguration commands to the optical switches and notifies the server daemons about the new configuration. The notification messages can be multicast to reduce overhead. The management daemon notifies the per-destination-rack output queue scheduler about the new optical path using *tc* (a Linux tool for configuring the kernel network stack). The scheduler then dequeues and de-multiplexes packets according to the new configuration. In a very large data center, the control traffic between individual servers and the central manager could be significant. We discuss this issue in Section 7.

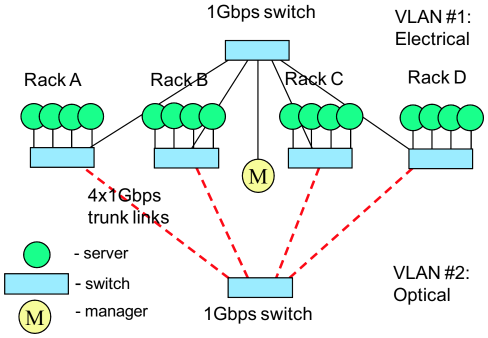


Figure 3: The logical testbed topology.

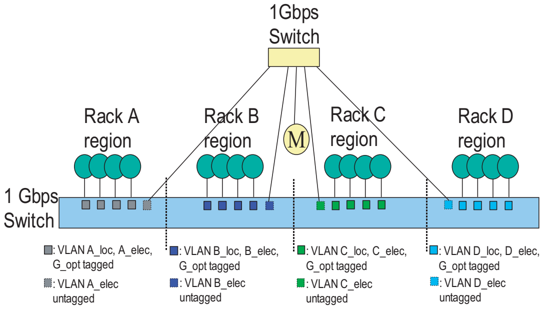


Figure 4: The physical testbed configuration.

5. SYSTEM EVALUATION

We implemented a c-Through prototype to study how well packet-switched and circuit-switched networks coexist and how applications perform. Due to the expense of prototyping a hybrid electrical/optical network, we emulate a HyPaC topology on a conventional packet-switched network, enabling and disabling communications to emulate the availability of high-speed optical links between switches. A controller introduces a reconfiguration delay to emulate the reconfiguration and settling time that the optical components would experience. Section 5.2 validates that this framework accurately reproduces network properties of interest. Although we choose parameters for these links and intervals based upon one particular optical technology (MEMS optical switches), our results should generalize as long as two assumptions continue to hold about optical and electrical networking: First, that the reconfiguration time of the optical switch remains long relative to the number of packets that can be sent (e.g., a 5ms reconfiguration interval is 16,000 packets of 1500 bytes on a 40 Gbps network). Second, that a circuit-switched optical path will retain a substantial bandwidth advantage over an electrically-switched path.

5.1 Testbed Setup

While there is no canonical data center design, one popular configuration includes 40 servers per rack connected at 1 Gbps to a ToR switch. Although a single rack can generate as much as 40 Gbps, the ToR switches are often interconnected at 1, 2, 4, or 10 Gbps, offering *over-subscription ratios* from 40:1 to 4:1 [7, 25].

We emulate comparable over-subscription ratios in our experiments. The logical topology emulated is shown in Figure 3. Physically, our emulation testbed consists of 16 servers and two Ethernet switches. Our switches are Dell PowerConnect 5448 Gigabit switches, and the servers have two 4-core Intel Xeon 2.5GHz processors and 8GB of memory. They run Ubuntu 8.04.3 LTS with the Linux 2.6.30 kernel. The 16 servers are connected to the first Gigabit Ethernet switch at 1 Gbps. We use VLANs to isolate the servers into four logical server racks as shown in Figure 4. The VLAN *X_loc* is used for intra-logical-rack traffic. Packets tagged with *X_loc* can only reach other servers within the same logical rack. Within each logical rack, one switch port is connected to the second Gigabit Ethernet switch (as shown in the top of Figure 4) which emulates a low speed packet-switched inter-rack network. By rate-limiting the ports, over-subscription ratios from 40:1 to 4:1 can be emulated. VLAN *X_elec* is used for traffic going through this low speed packet-switched inter-rack network. On the other hand, we emulate an optical circuit switch connecting the logical racks using the internal switching fabric of the first Ethernet switch. When the optical manager decides two logical racks are connected via an emulated circuit, communications through the emulated optical switch between these logical racks are allowed; otherwise, they are disallowed. An emulated optical circuit provides 4 Gbps of capacity between logical racks when communication is allowed. VLAN *G_opt* is used for traffic going through the emulated optical circuit-switched network.

Using an Ethernet switch to emulate the optical network creates a few differences between our testbed and a real optical network. First, we must artificially restrict communication through the emulated optical switch to emulate the limited rack-to-rack circuits. Second, we must make the network unavailable during optical switch reconfiguration. We estimate this delay based upon the switching time of MEMS switches plus a small settling time for the optical transceivers to re-synchronize before talking to a new destination. During this delay, no traffic is sent through the emulated optical switch.

For comparison, we also emulate a full bisection bandwidth packet-switched network in the testbed by allowing traffic to flow freely through the switching fabric of the first Gigabit Ethernet switch.

We implement the per-rack output scheduler in the kernel. The optical manager runs on a separate server. It collects traffic statistics from the rack servers and dynamically reconfigures the emulated optical paths. Because we use an Ethernet switch to emulate the optical switch, the optical manager does not command switches to set up optical paths. Instead, it communicates with the servers' management daemons to reconfigure the scheduler. When the manager starts to reconfigure optical paths, it first tells all servers to disable the optical paths by configuring the scheduler modules. It then delays for 10ms to emulate the switch reconfiguration, during which time no servers can use any optical path. The manager then instructs the servers to activate the new paths.

5.2 Micro-benchmark Evaluation

The goal of our micro-benchmarks is to understand how well today's TCP/IP stack can use the dynamic optical paths. In Section 6, we measure the benefit applications gain from the optical paths.

5.2.1 TCP behavior during optical path reconfiguration

We evaluate the effect of optical network emulation using electrical switches. We examine the TCP throughput between two servers in our testbed over several reconfiguration epochs. The servers transfer data as rapidly as possible from one to the other using the default TCP CUBIC. While the flow is running, we periodically set up and tear down the optical path between them. We emulate a 40:1 over-subscription ratio (100Mb/s between racks) to maximize the capacity change upon reconfiguration.

	FIFO		VLAN+output scheduling	
	Optical	Electrical	Optical	Electrical
TCP (Mbps)	940	94	921	94
UDP (Mbps)	945	94	945	94

Table 2: Throughput with and without output scheduling.

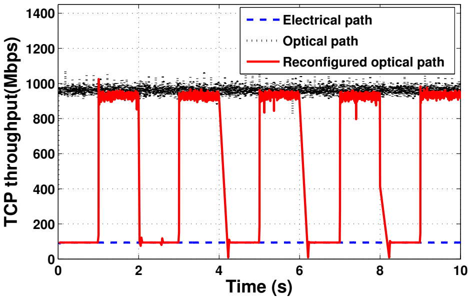


Figure 5: TCP throughput on a reconfiguring path.

Figure 5 shows the TCP throughput at the receiver across the entire experiment, along with what that throughput would look like if the flow was routed on the electrical or the optical path alone. The testbed correctly emulates the changes in network capacity before, during, and after reconfiguration. At time=1s, for instance, TCP correctly increases its throughput from 100 Mbps to 1 Gbps.

In this environment, TCP can increase its throughput to the available bandwidth within 5 ms of reconfiguration. When the optical path is torn down, TCP experiences packet loss since the available bandwidth drops rapidly. Although throughput briefly drops below the electrical network's capacity, it re-reaches the electrical capacity within a few tens of milliseconds. These experiments confirm that the optical emulation reflects the expected outcome of reconfiguration, and that TCP adapts rapidly to dynamically re-provisioned paths. This adaptation is possible because of the low RTT among servers. Even for a high bandwidth transmission, the TCP window size remains small.

5.2.2 How does the scheduler affect throughput?

The optical management system adds an output scheduler in the server kernel. The imposition of this component does not significantly affect TCP or UDP throughput. Table 2 shows the throughput achieved between a single sender and receiver using both 100 Mbps and 1 Gbps links (electrical and optical capacities, respectively), over TCP and UDP, and with and without the output scheduler. We gathered the results using *iperf* to send 800MB of data from memory as rapidly as possible.

5.2.3 Do large buffers affect packet delays?

c-Through is designed to avoid head-of-line (HoL) blocking by using per-flow socket buffers to buffer traffic, preventing the large buffers from imposing excess delay on latency-sensitive, small flows. Traffic from different flows to the same per-rack output queue share bandwidth according to TCP congestion control. The networks *can*, of course, become congested when applications send a large amount of data. This congestion, however, is not unique to c-Through.

App	Traffic pattern	Synchronization
VM Migration	one-to-many	NONE
MapReduce	all-to-all	Loose
MPI FFT	all-to-all	Global barrier

Table 3: Benchmark applications

To confirm that our per-flow buffering avoids HoL blocking and does not unfairly increase latency, we send both TCP and ICMP probes together with an aggressive TCP flow to servers in the same destination rack. The probe flows sends 1 probe every second and the heavy TCP flow sends 800MB of data as fast as possible, saturating the 100MB socket buffer and transiently filling the relatively small buffers on the Ethernet switch (until the optical network provides it with more bandwidth in response to its increased demand). Both TCP and ICMP probes observe a 0.15 to 4–5ms increase in RTT due to the congestion at the switch, but do *not* experience the huge delay increases it would observe were it queued behind hundreds of megabytes of data.[5]

6. APPLICATIONS ON C-THROUGH

The benefits from using HyPaC depends on the application communication requirements. The traffic pattern, its throughput requirements, and the frequency of synchronization all affect the resulting gain. We evaluated three benchmark applications, summarized in Table 3. We chose these applications to represent three types of cluster activity: bulk transfer (VM migration), loosely-synchronized computation (Map Reduce), and tightly-synchronized computation (MPI/HPC). To understand how applications perform when optical circuits are combined with an electrical network at different speeds, we emulated 40:1, 10:1, and 4:1 over subscription ratios by setting the cross-rack packet-switched bandwidth to 100Mbps, 400Mbps and 1Gbps. Although in practice a data center may run mixed applications, we run each application separately for ease of understanding the applicability of c-Through to different application patterns.

6.1 Case study 1: VM Migration

Virtual machine (VM) migration—in which a live VM (including a running operating system) moves from one physical host to another—is a management task in many data centers. The core of migration involves sending the virtual machine memory image (usually compressed) from one host to another. This application represents a point-to-point bulk data transfer and does not require synchronization with machines other than the sender and receiver. Because the VM memory image may be large (multiple GB) and the migration needs to be done rapidly, VM migration requires high bandwidth. Intuitively, we expect this application to be an ideal case for c-Through: the high bandwidth demand will make good use of the optical network, the pairwise concentrated traffic (point-to-point) is amenable to circuit-switching, and the lack of synchronization means that accelerating part of the traffic should improve the overall performance.

We deployed the KVM [5] VMM on our testbed to study the performance of virtual machine migration with c-Through. Our experiment emulates a management task in which the data center manager wants to shut down an entire rack; prior to shutdown, all the VMs running on that rack need to be migrated to other racks. In our experiments, we migrate all the VMs evenly to servers in other racks based on the VM ID. The starting state has eight VMs per physical node; each VM is configured with 1GB of RAM. Since we have four servers per rack, we need to migrate 32 VMs in total.[6]

Figure 6 shows the average completion time of the task, averaged across 3 runs, using c-Through, the bottlenecked electrical network, and a full bisection bandwidth network. Results are further presented as a function of the TCP send buffer size (Figure 6(a)) and the reconfiguration interval (Figure 6(b)). The three bars correspond to 40:1, 10:1, and 4:1 over subscription ratios (flat for the case of full bisection bandwidth).

We begin by examining the results when the electrical network is 40:1 over-subscribed. In Figure 6(a) we fix the reconfiguration interval to be 1s and vary the TCP socket buffer sizes on the servers. Using only the bottlenecked electrical network, the migration takes 280 seconds. Even with default 128KB socket buffers, c-Through accelerates the task to 120 seconds. With larger socket buffers, c-Through improves performance even more. For example, with 300MB TCP socket buffers, the job completes within 60 seconds—very close to the 52 second completion time with a full bisection bandwidth network. The reasons for this near optimal performance are twofold: First, during VM migration, each source host will tend to have some traffic destined for each rack. As a result, an optical path, once configured, can be fully utilized. Second, the performance of the full bisection bandwidth network is closely determined by the time to initiate the task and compress the VM images, operations that are CPU-limited and not bandwidth-limited. As a result, the bandwidth provided by the optical path, while not always full bisection, is nonetheless sufficient to accelerate performance.

Figure 6(b) shows the effect of the reconfiguration interval on VM migration performance. In this figure, we fix the socket buffer size to 100MB, as a reasonable choice from the previous results. Since VM migration is a point-to-point bulk transfer application, it does not require fine-grained optical path reconfiguration. c-Through achieves good VM migration performance even with a 5 second optical reconfiguration interval.

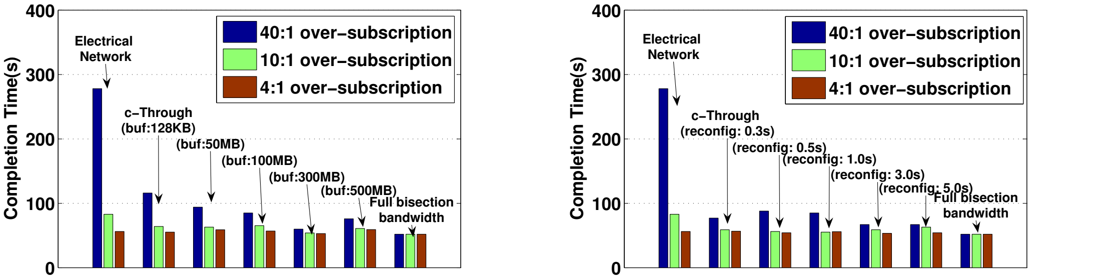
As expected, when the electrical network is less oversubscribed, the performance gap among the three network designs shrinks. That is, when the bandwidth of electrical network is higher, the benefit of adding more bandwidth is relatively smaller. More importantly, however, our results clearly demonstrate that even *augmenting a slow electrical network with c-Through could achieve near optimal performance for VM migration/bulk data transfer applications*.

6.2 Case study 2: MapReduce

MapReduce [18] is a widely-used parallel computation approach in today's data centers. Many production data centers, such as Amazon EC2 and Google, support MapReduce-style applications. In MapReduce applications, jobs are split into Map tasks and Reduce tasks. During the Map phase, the master node assigns the job to different mappers. Each mapper reads its input from a data source (often the local disk) as key-value pairs and produces one or more intermediate values with an output key. All intermediate values for a given output key are transfered over the network to a reducer. The reducer sorts these intermediate values before processing them; as a result, it cannot start until all mappers have sent it their intermediate values. At the end of the Reduce task, the final output values are written to stable storage, usually a distributed file system, such as the Hadoop filesystem (HDFS).

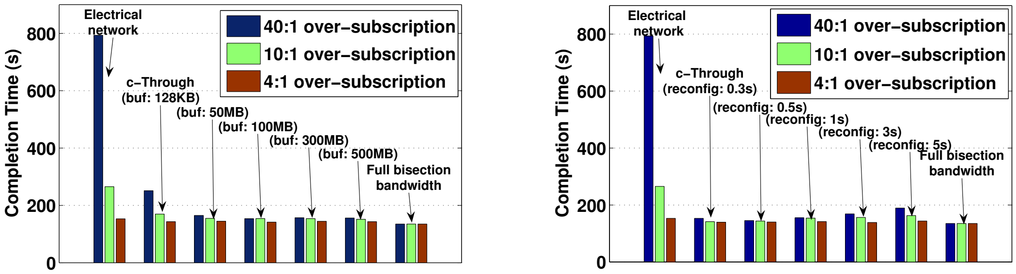
[5] This design still could increase latency of application control messages *within a flow* if the application sends data and control messages over the same socket.

[6] Although each VM is configured with 1GB RAM, the virtual machine may not be using all the RAM at one time. Thus, before migration, the VMM will compress the memory image, reducing the amount of data it needs to send.



(a) Varying socket buffer size (reconfigure: 1s) (b) Varying reconfiguration interval (buffer size: 100MB)

Figure 6: Virtual machine migration performance



(a) Varying socket buffer size (reconfigure: 1s) (b) Varying reconfigure interval (buffer size: 100MB)

Figure 7: The performance of Hadoop sort

In the context of c-Through, MapReduce has two important features: all-to-all traffic and coarse-grained synchronization. The mappers must often shuffle a large amount of data to the reducers. As a result, MapReduce applications may not have traffic that is as concentrated as that in a bulk data transfer such as VM migration.[7] Second, MapReduce applications have only a single system-wide synchronization point between the Map and Reduce phase.

To explore the effect of c-Through on MapReduce-style applications, we deployed Hadoop (an open source version of MapReduce) on our testbed. The first Hadoop application that we ran was a distributed *sort*. A feature of Hadoop *sort* is that the intermediate data generated by the mappers and the final output generated by the reducers are at least as large as the input data set. Therefore, Hadoop's sort requires high inter-rack network bandwidth. Our sort uses 10GB of random data as its input set, and we vary the c-Through parameters as before.

Figure 7 shows the performance of Hadoop sort on c-Through with different buffer sizes, reconfiguration intervals. Figure 7(a) shows the effect of the TCP send buffer size on job completion time (as before we fix the optical network reconfiguration time to 1s). Enlarging the TCP socket buffers has very little impact on performance when the buffer size is larger than 50MB. The reason is that the block size of HDFS is 64MB, which also forms its transmission unit. Consequently, Hadoop does not expose enough traffic to the kernel when socket buffers are set to large values.

In Figure 7(b), we fix the TCP send buffer size to 100MB and vary the optical path reconfiguration interval. The key result from this experiment is that more frequent optical path reconfiguration can improve Hadoop's sort performance (this should be expected, since faster reconfiguration allows for more frequent draining of the slowly filling buffers).

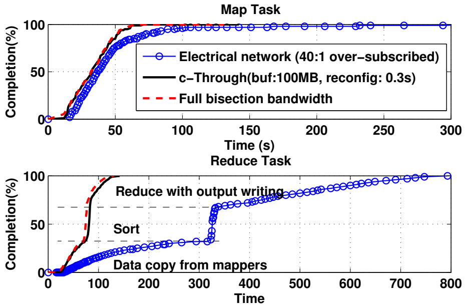


Figure 8: The completion of Hadoop sort tasks

Coupling c-Through with a slow (40:1 over-subscribed) electrical network provides near-optimal performance for Hadoop sort. To understand the source of this performance improvement, we plot the benchmark's execution timeline in Figure 8. The top portion shows the Map phase and indicates the completion of the map tasks as a function of time. The solid line shows Hadoop's performance on the bottlenecked electrical network with 40:1 over subscription ratio; note how the curve has a long tail. This tail comes from a few mappers that are reading from a non-local data source and are hitting the electrical network's bottleneck capacity. By adding optical paths, c-Through eliminates the long tail, significantly reducing the completion time of the map phase (seen when the c-Through curve reaches 100%). This reduction, in turn, improves overall Hadoop sort performance because the reduce tasks cannot start processing until all of the map tasks have completed.

The bottom portion of Figure 8 shows the execution timeline of

[7] With one important exception: During the output phase, the final values are written to a distributed filesystem, often in a rack-aware manner.

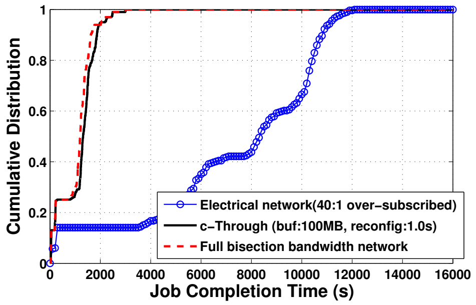


Figure 9: The completion of Hadoop Gridmix tasks

the Reduce phase, which has three steps: copying the intermediate values from the mappers and grouping by key, sorting, and the execution of the reduce function. (The execution of the reduce function also includes writing the output back to HDFS.) c-Through can significantly speed up the data copying and output writing. During the data copying step, data is shuffled among all of the mappers and reducers. Since the use of key space and the value sizes are uniform, there is an equivalent traffic pattern across all racks. Nevertheless, our results shows that c-Through can still accelerate this data shuffling step. Since reducers can start pulling data from mappers as soon as it is available, intermediate data is shuffled among mappers and reducers while the remaining mappers are still running. There is not a blocking serialization point in the shuffle phase because the map tasks and reduce tasks are running in parallel. This property makes it suitable for batch transfer over optical paths.

As reduce tasks complete, the final output is written to HDFS, which maintains (by default) three replicas of each data block. After an invocation of reduce, HDFS will write one copy of the data locally and send two additional copies of the data to other servers. This replication generates significant network traffic and saturates the electrical network, which is clearly visible in the bottom figure by the long tail of the output writing step. Again, this traffic does not exhibit significant skew, but nevertheless, c-Through can accelerate this bulk data transfer step significantly.

Finally, we summarize MapReduce's performance on the three network designs. When Hadoop sort runs on a full bisection bandwidth network, it can still be bottlenecked by intensive disk I/O operations. Consequently, as before, we observe that even using reconfigurable optical paths with a slow electrical network can still provide close to optimal performance to data-intensive, MapReduce-style applications—despite relatively uniform traffic patterns. Similarly, when the electrical network becomes faster, the performance gap among bottlenecked electrical network, c-Through and full bisection bandwidth network is smaller.

Gridmix: To understand how c-Through works for more realistic applications with complicated traffic patterns, we study the performance of the Hadoop Gridmix benchmark on c-Through. Gridmix [4] mimics the MapReduce workload of production data centers. The workload is simulated by generating random data and submitting MapReduce tasks based on the data access patterns observed in real user jobs. Gridmix simulates many kinds of tasks with various sizes, such as web data scanning, key value queries and streaming sort. We generate a 200GB uncompressed data set and a 50GB compressed data set for our experiments. Each experiment launches 100 tasks running on these data sets. We run the same Gridmix experiment 3 times for each network architecture. Figure 9 shows the cumulative distribution of the completion time of Gridmix tasks on the bottlenecked electrical network, full bisection bandwidth network and c-Through network respectively. In our experiment, c-Through uses 100MB socket buffer size and 1 second optical reconfiguration time. The over-subscription ratio of the electrical network is 40:1. The results show that even when the electrical network is highly over-subscribed, the completion time of Gridmix jobs on c-Through network is very close to that on full bisection bandwidth network.

6.3 Case study 3: MPI Fast Fourier

Fast Fourier Transforms (FFTs) are frequently used in scientific computing; efficient parallel FFT computation is important for many large scale applications such as weather prediction and Earth simulation. Parallel FFT algorithms use matrix transpose for FFT transformation. To perform a matrix transpose using MPI, a master node divides the input matrix into sub-matrices based on the first dimension rows. Each sub-matrix is assigned to one worker. Matrix transpose requires each worker to exchange intermediate results with all other workers. Parallel FFT on a large matrix therefore requires data intensive all-to-all communication and periodic global synchronization among all of the workers. We expected this style of application to be challenging to a hybrid network because of the poor traffic concentration and strict synchronization among servers. Surprisingly, c-Through substantially improved performance.

We studied the performance of parallel FFT on our testbed using FFTW [2]. FFTW is a C library for computing discrete Fourier transform in one or more dimensions. It provides both single node and MPI-based implementations to compute FFT with real or complex data matrices. We ran MPI FFTW on 15 nodes of our testbed to compute the Fourier transform of a complex matrix with 256M elements. The resulting matrix occupies 4GB. Figure 10 reports the MPI FFT completion time with different network settings.

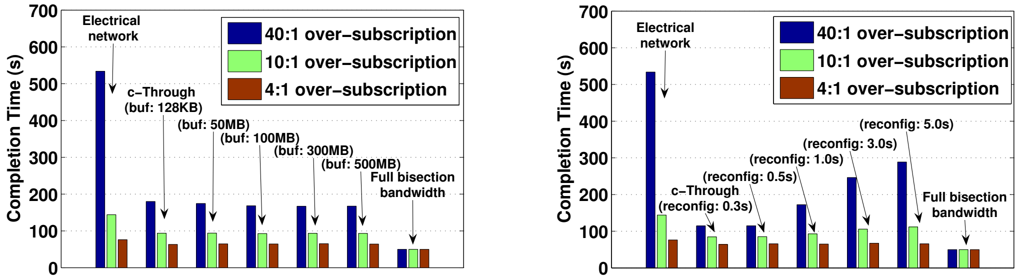
We follow the same method as with previous experiments to study the effect of socket buffer size and optical reconfiguration interval on application performance. Because MPI FFT exchanges small blocks (20MB) of the matrix among servers, larger buffers do not substantially improve performance. The benefits of c-Through for MPI FFT depend more on rapid optical path reconfiguration. As shown in Figure 10, with 100MB socket buffers and a 0.3 second optical reconfiguration interval, the job took twice as long using c-Through with a 40:1 oversubscribed electrical network as it did on the full bisection bandwidth network. When the optical path is reconfigured slowly (e.g. 5 seconds), the performance of MPI FFT is much worse than when the reconfiguration interval is 0.3 seconds. The frequent global barrier synchronization, combined with the uniform traffic matrix, make it impossible to saturate the optical network for the entire period. *To maximize performance, applications must ensure there is enough data available in the buffers to saturate the optical link when it becomes available.* Reconfiguration periods should not be large without reason: The period should be chosen just long enough to ensure efficiency (the network experiences downtime during the reconfiguration), but small enough so that links do not fully drain due to lack of data to the chosen destination rack.

7. DISCUSSION

7.1 Applicability of the HyPaC Architecture

The case studies provide several insights into the conditions under which a HyPaC network can or cannot provide benefits.

Traffic concentration: Bulk transfers can be accelerated by high capacity optical circuits. c-Through buffers at sources to create



(a) Varying socket buffer size (reconfigure:1s) (b) Varying reconfigure interval (buffer size: 100MB)

Figure 10: The performance of MPI FFT

bulk transfer opportunities, but applications may not be able to make full use of such a feature if they internally operate on small sized data units. An example of this comes from our experience with Hadoop: In many cases, Hadoop's default configuration limited the nodes' ability to generate and buffer large amounts of data, even when TCP provided sufficient buffer space. Fortunately, the performance tuning was straightforward—and similar to the techniques needed to achieve high throughput on high bandwidth-delay networks.

Even applications that may seem on the surface to have uniform traffic matrices may experience transiently concentrated traffic, such as the data output phase in Hadoop. By diverting even just this one phase of the transfer over the optical links, all applications running on the cluster—even those with uniform traffic—can benefit from the reduced load on the oversubscribed electrical network. Second, even uniform applications such as the sort phase of Hadoop may experience traffic concentration on shorter timescales due to statistical variation in their traffic. If those timescales are long enough, the hybrid network can exploit the transient imbalance to accelerate the transfer.

Synchronization: HyPaC networks are more suited for applications with loose or no synchronization. The pairwise connections and reconfiguration interval impose a minimum time to contact all racks of interest called the *circuit visit delay.* If the time between application synchronization events is substantially smaller than the circuit visit delay, the benefits of a HyPaC network decrease rapidly. There are two ways to reduce the circuit visit delay: Cluster the application traffic so that it must visit fewer racks, or move to faster optical switching technologies; until these technologies become available, the HyPaC architecture may not be an appropriate choice for tightly synchronized applications that require all-to-all communication.

Latency sensitivity: All the applications we have studied so far are not sensitive to the latency of particular messages. Some data center applications, such as Dynamo [19], do not operate on bulk data. Instead, they need to handle a large number of small queries. These applications are sensitive to the latency of each query message. For such applications, a HyPaC network can improve query throughput and relieve congestion in the electrical network, mostly because these applications also perform concentrated bulk transfers during reconfiguration and failover. However, it does not reduce the non-congested query latency.

7.2 Making Applications Optics Aware

Our current design makes the optical paths transparent to applications. Applications might take better advantage of the HyPaC architecture by *informing* the optical manager of their bandwidth needs, or by *adapting* their traffic demands to the availability of the optical network. We view both of these questions as interesting avenues for future exploration. The applications we have studied so far have largely stationary traffic demands that would benefit little from telegraphing their intent to the manager. Other applications, however, may generate more bursty traffic that could benefit from advance scheduling. The second approach, allowing applications to actively adapt to the reconfiguring bandwidth, could potentially leverage optical paths in much better ways.

For example, applications could buffer more data on their own, or generate data in a more bursty fashion based upon the availability of the network. Several datacenter applications are already topology-aware, and it may be possible to make such applications (e.g., Hadoop) adapt to the changing topology just by modifying the scheduling algorithms. Finally, the optical component manager might be integrated into a cluster-wide job/physical resource manager that controls longer-term, high level job placement, to improve traffic concentration as discussed below.

7.3 Scaling

There are three scaling challenges to realize a large hybrid network. First, the size of the data center may exceed the port capacity of today's optical switches, requiring a more sophisticated design than the one considered so far. Second, the measurement and matching computation overhead scales with the number of racks. Third, if racks talk to more and more other racks, the *circuit visit delay*, i.e. the amount of time it takes to connect one rack to a particular other rack, might increase. To make the discussion concrete, we consider a large data center with 1000 racks (a total of 40,000 servers).

Optical network construction: Given a 1000-rack data center, c-Through would require a 1000-port optical circuit switch. Research prototype MEMS optical switches already scale to a few thousand input and output ports [31], but they are not available commercially; today's largest MEMS switches has 320 input and output ports [1]. In the future, however, this approach could provide a very simple and flexible way to add optical paths into large data centers.

Without such dense switches, one choice is to divide the datacenter into zones of 320 racks (12800 servers). As we discuss below, both anecdotes and recent datacenter utilization data from Google suggest that the vast majority of jobs run on clusters smaller than this. However, this answer is both intellectually dissatisfying and is based only on *today's* use patterns. An alternate approach is to chain multiple MEMS switches using the same fat-tree or butterfly topologies used in recent research to scale electrical networks, at the cost of both increased cost and the requirement for fast, coordinated circuit switching. Fortunately, constructing a rack-to-rack circuit switched optical fat tree is much simpler than a node-to-node packet switched fat tree (it operates at roughly $\frac{1}{40}$ the scale). Eleven 320x320 optical MEMS switches could cover a 1000-rack data center, at a cost three times higher per node than simply dividing the network into zones.

Optical network management: The VLAN based traffic control can remain unchanged when c-Through is used in large data centers since it operates on a per-switch or per-server basis. If we use a single switch or a fat-tree to construct the optical network, Edmonds' algorithm can still be used to compute the optical path configuration. Edmonds' algorithm is very efficient: For a 1000-rack data center, the computation time to generate a new configuration/schedule is only 640ms on a Xeon 3.2GHz processor, which is sufficiently fast for a large class of data center applications [39]. If the optical network topology is a hierarchy or a ring, the configuration is not a perfect matching anymore; nevertheless, it is still a max weighted matching. Several algorithms can compute the maximum matching efficiently [23], and several approximation algorithms [23, 32] can compute near-optimal solutions even faster.

Scaling also increases the demands on collecting and reporting traffic measurements, as well as disseminating reconfiguration notices. If optical paths are reconfigured every second and all of the rack queues have a large amount of data buffered, each server needs to report its traffic demands (4 bytes each) for each of the 1000 possible destination racks. Thus, each server will send 4KB of traffic statistics data to the optical manager. Across the whole data center, the optical manager will receive 160MB of traffic measurement data (40 servers per rack) every second. We believe that our existing heuristic for discarding under-occupied buffer slots will be sufficient to handle this aspect of scaling (Section 4). For example, based on our MapReduce experiment trace, when servers only report queue sizes larger than 1MB, the traffic measurement data is only 22% of the worst case volume. As discussed below, we believe that as the network scales, more and more of the rack-to-rack paths are likely to be unused.

Finally, a number of standard techniques, such as hierarchical aggregation trees and MAC-layer multicast, can help alleviate the messaging overhead. Multiple message relay nodes can be organized into a logically hierarchical overlay. Each leaf node collects traffic measurement data from a subset of servers, aggregating the data through the hierarchy to the root that computes the optical configuration. Notifications of configuration changes can also be disseminated efficiently through the hierarchy to the servers. Intra-rack MAC layer multicast could be used to further improve efficiency.

Circuit visit delay: The circuit visit delay depends on the number of destination racks to which servers spread traffic. In a 1000-rack data center, the worst case scenario is an application that simultaneously sends large amounts of traffic to destinations in all 1000 racks. The last rack will have to wait for 999 reconfiguration periods before it can be provisioned with an optical path.

The problems of long *circuit visit delay* are twofold. First, the amount of data that must be queued could grow quite large—up to a full circuit visit delay times the server's link capacity. At 1 Gb/s, 999 seconds worth of data (128 GB) would substantially exceed the memory available in most servers. Worse, this memory is pinned in kernel. During the wait for reconfiguration, the application's traffic will be routed over the electrical network, causing congestion for other, possibly latency-sensitive, traffic.[8] For this worst case scenario, data center designers may require other solutions, such as a full bisection bandwidth electrical network.

In practice, however, many factors can help keep the circuit visit delay low. First, although some applications (e.g. MapReduce, MPI) require all-to-all communication, the application may only use a smaller set of racks during any particular phase of its execution. Second, in a large data center, a single application usually does not use the entire data center. In many cases, a large data center is separated into many subnets. Different applications are allocated into different subnets. Although we do not have exact numbers about the job sizes in production data centers, recently released workload traces from a Google production data center show that their largest job uses 12% and the medium size job only uses 0.8% of cores in the data center [3].[9] Therefore, even in a large data center, there are likely to exist smaller cliques of communicating nodes. In fact, Microsoft researchers have also observed this to be true in their large production data center [29]. A second path to scaling, then, is to ensure that the job placement algorithms assign these applications to as few racks as possible. This is actually a simplification of the job placement algorithms required today, which must not only group at the rack level, but at the remaining levels of the network hierarchy.

8. RELATED WORK

Our previous papers [24, 39] proposed the basic ideas of using optical circuits to augment an under-provisioned packet-switched network in data centers. Two recent proposals sketched designs that are similar in spirit to ours. One observed that many applications that run in their production datacenters do not require full bisection bandwidth. They instead proposed the use of 60GHz wireless "flyways" to augment an electrical network provisioned for average-case use, using the wireless links to selectively add capacity where needed [29]. The second one is parallel work Helios [22], which explores a similar hybrid electrical/optical data center architecture. A key difference between Helios and c-Through is that Helios implements its traffic estimation and traffic demultiplexing features on switches. This approach makes traffic control transparent to end-hosts, but it requires modifying all the switches. An advantage of the c-Through design is that by buffering data in the hosts, c-Through can batch traffic and fill the optical link effectively when it is available. Helios and c-Through demonstrate different design points and performance trade-offs in the hybrid data center design space.

The supercomputing community has also extensively examined the use of optical circuits, though their goals often differ substantially from our focus. Within a supercomputer, several papers examined the use of node-to-node circuit switched optics [14]; in contrast, our work deliberately amortizes the potentially high cost of optical ports and transceivers by providing only rack-to-rack connectivity, a design we feel more appropriate for a commodity datacenter. Our work by necessity then focuses more on the application and operating systems challenges of effectively harnessing this restricted pattern of communication. IBM researchers explored the use of hybrid networks in a stream computing system [36]. While it provides no design details, this work focused primarily routing and job management in stream computing.

Using large per-destination queues at the edge to aggregate traffic and make use of optical paths is related to optical burst switching [35] in optical networks. Many research efforts examined the use of optical burst switching in the Internet backbone (e.g [38, 40]),

[8] We have deliberately avoided a design in which some traffic is delayed until the optical network becomes available, feeling that such an approach makes little sense without tight application integration to appropriately classify the traffic.

[9] The data do not specify the exact size of the datacenter.

but they focus mostly on distributed scheduling and contention resolution in order to correctly integrate the optical paths. Similarly, UCLP (User Controlled Lightpaths) [11] and numerous other technologies (e.g., MP-λ-S) switch optical circuits on hour-and-longer timescales for wide-area connectivity and computing. In contrast, our datacenter focus limits the amount of traffic available to statistically multiplex onto the optical links, but simultaneously grants the flexibility to induce traffic skew and to incorporate the end-hosts into the circuit switched network. Our results suggest that the increased control and information from this integration substantially improves the throughput gains from optical links.

9. CONCLUSION

Building full bisection bandwidth networks using packet switches imposes significant complexity. It may also aim to provision more than today's and future's applications require. This paper explores the use of optical circuit switching technology to provide high bandwidth to data center applications at low network complexity. We present a HyPaC architecture that integrates optical circuits into today's data centers, and demonstrate the feasibility of such a network by building a prototype system called c-Through.

By studying several modern datacenter applications, we assess the expected gain from integrating optical circuits in target data center scenarios. Our results suggest that a HyPaC network offers the potential to significantly speed many applications in today's datacenters, even when the applications may not intuitively seem to be promising candidates for acceleration through circuits. While there remain many significant questions about—and options for—the design of future datacenter networks, we believe the HyPaC architecture represents a credible alternative that should be considered along-side more conventional packet switched technologies.

Acknowledgments: We thank Richard Gass of Intel Labs Pittsburgh for his help setuping up the experiments. We thank our shepherd Jitendra Padhye and the anonymous reviewers for their valuable feedback. This research was supported in part by NSF awards 0546551 and 0619525. Wang and Ng are sponsored by the NSF through CAREER Award CNS-0448546, CNS-0721990, and by an Alfred P. Sloan Research Fellowship.

References

[1] Calient networks diamondwave fiberconnect, . http://www.calient.com/products/diamondwave_fiberconnect.php.

[2] FFTW: Fastest fourier transform in the west, . http://www.fftw.org/.

[3] Google cluster data, . http://code.google.com/p/googleclusterdata/.

[4] Gridmix: Trace-based benchmark for mapreduce, . https://issues.apache.org/jira/browse/MAPREDUCE-776.

[5] KVM: Kernel based virtual machine, . http://www.linux-kvm.org/page/Main_Page.

[6] Glimmerglass 80x80 MEMS switch. Website, 2003. http://electronicdesign.com/article/test-and-measurement/3d-mems-based-optical-switch-handles-80-by-80-fibe.aspx.

[7] OpenCirrus, . https://opencirrus.org/.

[8] OpVista CX8 optical networking system for 40G and 100G services to debut at NXTComm08, . http://www.encyclopedia.com/doc/1G1-180302517.html.

[9] Alcatel-Lucent Bell labs announces new optical transmission record, . http://tinyurl.com/yau3epd.

[10] IETF transparent interconnection of lots of links (TRILL) working group, . http://datatracker.ietf.org/wg/trill/charter/.

[11] UCLP project, . http://www.uclp.ca/.

[12] M. Al-Fares, A. Loukissas, and A. Vahdat. A scalable, commodity, data center network architecture. In *Proc. ACM SIGCOMM*, Aug. 2008.

[13] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat. Hedera: Dynamic flow scheduling for data center networks. In *Proc. 7th USENIX NSDI*, Apr. 2010.

[14] K. Barker, A. Benner, and R. H. et al. On the feasibility of optical circuit switching for high performance computing systems. In *Proc. SC05*, 2005.

[15] A. Basu and J. G. Riecke. Stability issues in OSPF routing. In *Proc. ACM SIGCOMM*, Aug. 2001.

[16] T. A. Benson, A. Anand, A. Akella, and M. Zhang. Understanding data center traffic characteristics. In *Proc. Workshop: Research on Enterprise Networking*, Aug. 2009.

[17] J. Buus and E. J. Murphy. Tunable lasers in optical networks. *Journal of Lightwave Technology*, 24(1), 2006.

[18] J. Dean and S. Ghemawat. MapReduce: Simplified data processing on large clusters. In *Proc. 6th USENIX OSDI*, Dec. 2004.

[19] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels. Dynamo: Amazon's highly available key-value store. In *Proc. 21st ACM Symposium on Operating Systems Principles (SOSP)*, Oct. 2007.

[20] J. Edmonds. Paths, trees and flowers. *Canadian Journal on Mathematics*, pages 449–467, 1965.

[21] K. Elmeleegy, A. Cox, and T. E. Ng. On count-to-infinity induced forwarding loops in ethernet networks. In *Proc. IEEE INFOCOM*, Mar. 2006.

[22] N. Farrington, G. Porter, S. Radhakrishnan, H. Bazzaz, V. Subramanya, Y. Fainman, G. Papen, and A. Vahdat. A hybrid electrical/optical switch architecture for modular data centers. In *Proc. ACM SIGCOMM*, Aug. 2010.

[23] Z. Galil. Efficient algorithms for finding maximum matching in graphs. *ACM Computing Survey*, 18:23–38, 1986.

[24] M. Glick, D. G. Andersen, M. Kaminsky, and L. Mummert. Dynamically reconfigurable optical links for high-bandwidth data center networks. In *Optical Fiber Comm. Conference (OFC)*, Mar. 2009. (invited paper).

[25] A. Greenberg, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. Maltz, P. Patel, and S. Sengupta. VL2: A scalable and flexible data center network. In *Proc. ACM SIGCOMM*, Aug. 2009.

[26] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu. DCell: A scalable and fault-tolerant network structure for data centers. In *Proc. ACM SIGCOMM*, Aug. 2008.

[27] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu. BCube: A high performance, server-centric network architecture for modular data centers. In *Proc. ACM SIGCOMM*, Aug. 2009.

[28] Joseph Berthold. Optical networking for data center interconnects across wide area networks, 2009. http://www.hoti.org/hoti17/program/slides/SpecialSession/HotI-2009-Berthold.pdf.

[29] S. Kandula, J. Padhye, and P. Bahl. Flyways to de-congest data center networks. In *Proc. ACM Hotnets-VIII*, Oct. 2009.

[30] C. Kim, M. Caesar, and J. Rexford. Floodless in SEATTLE: A scalable ethernet architecture for large enterprises. In *Proc. ACM SIGCOMM*, Aug. 2008.

[31] J. Kim, C. Nuzman, B. Kumar, D. Lieuwen, et al. 1100x1100 port MEMS-based optical crossconnect with 4-dB maximum loss. *IEEE Photonics Technology Letters*, pages 1537–1539, 2003.

[32] N. Mckeown. The iSLIP scheduling algorithm for input-queued switches. *IEEE/ACM Transactions on Networking*, 7(2):188–201, Apr. 1999.

[33] A. Myers, T. E. Ng, and H. Zhang. Rethinking the service model: Scaling ethernet to a million nodes. In *Proc. 3nd ACM Workshop on Hot Topics in Networks (Hotnets-III)*, Nov. 2004.

[34] R. N. Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat. Portland: A scalable fault-tolerant layer2 data center network fabric. In *Proc. ACM SIGCOMM*, Aug. 2009.

[35] C. Qiao and M. Yoo. Optical burst switching (OBS) - a new paradigm for an optical Internet. *Journal of High Speed Networks*, 8(1):69–84, 1999.

[36] L. Schares, X. Zhang, R. Wagle, D. Rajan, P. Selo, S. P. Chang, J. Giles, K. Hildrum, D. Kuchta, J. Wolf, and E. Schenfeld. A reconfigurable interconnect fabric with optical cicuit switch and software optimizer for stream computing systems. In *Optical Fiber Comm. Conference (OFC)*, Mar. 2009.

[37] S. Sharma, K. Gopalan, S. Nanda, and T. Chiueh. Viking: A multi-spanning-tree ethernet architecture for metropolitan area and cluster networks. In *Proc. IEEE INFOCOM*, Mar. 2004.

[38] J. Turner. Terabit burst switching. *J. of High Speed Networks*, 8(1):3–16, 1999.

[39] G. Wang, D. G. Andersen, M. Kaminsky, M. Kozuch, T. S. E. Ng, K. Papagiannaki, M. Glick, and L. Mummert. Your data center is a router: The case for reconfigurable optical circuit switched paths. In *Proc. ACM Hotnets-VIII*, Oct. 2009.

[40] M. Yoo, C. Qiao, and S. Dixit. QoS performance of optical burst switching in IP-over-WDM networks. *Journal on Selected Area in Communications*, 18: 2062–2071, 2000.

[41] X. Zhang, R. Wagle, and J. Giles. VLAN-based routing infrastructure for an all-optical circuit switched LAN. In *IBM T.J. Watson Research Report*, July 2009.