Performance Evaluation of a Hybrid Optical/Electrical Interconnect

Kostas Christodoulopoulos, Diego Lugones, Kostas Katrinis, Marco Ruffini, and Donal O'Mahony

***Abstract*—In response to the need for higher-speed and affordable-cost datacenter interconnection networks, hybrid optical/electrical architectures have been proposed. Even still, a number of implementation issues remain open and little is known about the performance of real applications over such networks. To fill this gap, we present a hybrid network architecture, called HydRA, comprising commodity off-the-shelf equipment, calculate its total price using current list prices, and show its price competitiveness against fat-tree alternatives. We also report on a prototype implementation of our HydRA network that uses Ethernet switches and a custom-built software controller. We show that our HydRA network prototype manages to accelerate the execution of real parallel workloads, when compared to equal-cost electronic-only fat-tree networks.**

***Index Terms*—Bandwidth on demand; Control plane prototype; Dynamic reconfiguration; Hybrid optical network; Optical circuit switching.**

I. Introduction

The adoption of the cloud application-, platform-, and infrastructure-as-a-service models has substantially increased over the past few years and motivates the construction of new and more powerful datacenters (DCs). Apart from the higher-speed networks required to access the cloud DCs [1], higher-speed networks are also required to interconnect the servers within them. To address this requirement, DCs are typically designed with a fat-tree or oversubscribed fat-tree interconnection topology that can provide abundant bandwidth. Figure 1 presents an example of a fat-tree network built with commodity top-of-rack (ToR) Ethernet switches following the architecture proposed in [2]. However, fat-tree networks exhibit super-linear cost scaling and thus raise concerns when considered on a large scale, magnifying the contribution of the network to total cost.

In response to this challenge, prior research [3–6] has proposed using commercial off-the-shelf optical switches for transferring aggregated traffic between racks or collections of racks, partly or entirely replacing the higher levels of electronic tree networks. These proposals leverage some interesting features exhibited by optical switches, such as being protocol and rate agnostic, having relatively low cost per port, and being reconfigurable.

The optical network envisioned is typically implemented with commodity micro-electro-mechanical systems (MEMS) switches. MEMS switches are space switches operating at layer-1, whereby they reflect a light beam from an input to an output port without processing the switched signal. MEMS switches can forward any incoming to any outgoing port in a nonblocking fashion independent of the switching permutation. Currently, switches of up to 320 × 320 ports are commercially available. Although these optical switches are reconfigurable, they exhibit circuit configuration times that are typically in the order of tens to hundreds of milliseconds and thus would result in prohibitively high per-packet switching overhead should they be operated as packet switches (e.g., at 10 Gbps, the duration of a 1500-byte packet is 1.2 μs, at least four orders of magnitude shorter than the reconfiguration time). Also, even if the reconfiguration time decreases, it would be hard to operate the network at packet level granularity. The reason is that optical buffering is not available (still an open research topic), which means that optical packets need to be "on the fly," requiring global scheduling and synchronization among all the transmitters in the network to avoid contention, which is hard to achieve at packet level granularity. Taking this into account, the network architectures proposed in the literature are hybrids of two switching technologies, namely electronic packet and optical circuit switching.

The proposed hybrid networks are typically designed with a low-degree optical part (i.e., the number of optical ports is much smaller than the sum of server ports) so as to maintain the low cost, but since the optical network is reconfigurable it can adapt and provide capacity wherever it is needed, as opposed to the overprovisioned and thus high-cost design of conventional electronic-only tree networks. The motivation behind employing such a low-degree and reconfigurable network is that many studies suggest this would fit the requirements of applications in both high-performance computing (HPC) and DC worlds. In particular, studies on HPC parallel workloads reveal that in such applications the communication is performed between specific sets of servers and these communication

Manuscript received August 14, 2014; revised November 20, 2014; accepted January 12, 2015; published February 27, 2015 (Doc. ID 220822).

K. Christodoulopoulos (e-mail: kchristodou@ceid.upatras.gr) is with the Computer Engineering and Informatics Department, University of Patras, Patras, Greece.

D. Lugones is with Bell Labs, Dublin, Ireland.

K. Katrinis is with IBM Research, Dublin, Ireland.

M. Ruffini and D. O'Mahony are with CTVR and the School of Computer Science and Statistics, Trinity College Dublin, Ireland.

http://dx.doi.org/10.1364/JOCN.7.000193

patterns do not change or change very slowly over the duration of an application instance [3,7]. Moreover, it has been observed that traffic in DCs occurs mainly between specific racks and that the higher levels of tree networks in DCs are not efficiently utilized [8].

The hybrid optical/electrical network can operate in two modes. Assuming a DC environment with limited prior knowledge of the type of applications running on it, a periodic control process can collect monitoring information and configure the optical network to fit the traffic fluctuations [4,5]. In addition, such a reconfigurable network can support a true bandwidth-on-demand service and thus enable an alternative operation mode, where applications are granted access to the network controller (e.g., through a well-defined API) and direct it to configure the network according to their needs. Applications ranging from typical parallel HPC applications using message passing interfaces [9] to MapReduce/Hadoop schedulers that place mappers and reducers in a pool of servers can take advantage of a reconfigurable network that can be tailored to their needs. Moreover, new applications can be written to take advantage of this operation mode, since it is expected that joint consideration of the computation and communication resources would boost their performance. Note that the two above-described modes are not mutually exclusive; i.e., they can coexist in a cloud DC environment. However, although hybrid optical/electrical networks have been proposed before, very little is still known about what their impact on existing applications' performance can be and at what cost.

This paper attempts to bring together the work that we have performed on hybrid optical/electrical networks [9–12] and present a holistic view of such systems, answering key questions regarding their price, performance, and implementation. We present previously proposed architectures and comment on their characteristics. Then, we present our approach called hybrid reconfigurable architecture (HydRA) and its prototype deployment built out of currently available commodity equipment. We report on its total price using current list prices and compare it against conventional fat-tree networks. Furthermore, we discuss implementation issues and present a software stack and a HydRA network controller that uses VLANs to forward traffic over standard Ethernet switches. The network controller can be used to configure the topology before the application execution, but also to reconfigure the network at runtime implementing the two modes of operation outlined above.

We deployed our HydRA network controller in a 40-server/four-rack testbed in our lab and used it to compare the performance of communication-intensive HPC kernels and pseudo-applications running over our architecture against the performance obtained over equal-cost fat-tree setups. Our results show that statically configuring the network before application execution, we can accelerate the workload by up to 35%. We also experimented with dynamic reconfiguration and measure, for the first time, to the best of our knowledge, the reconfiguration delay of a fully functional prototype, taking into account delays introduced by our software control stack and hardware. Although our control stack can be further optimized, we managed to accelerate the execution of real parallel workloads to avoid congestion by dynamically reconfiguring the network during runtime.

II. Related Work

Various hybrid optical/electrical interconnect architectures have been proposed for HPC [3] and DC systems [4–6]. In these systems the electronic network typically connects all servers together in a multilevel electronic hierarchy (e.g., tree), while the optical network, implemented with a single or an array of optical switches, is connected at a specific level of the electronic network, for example, at the ToR switches (see Fig. 1) or at a higher level. Traffic aggregated at that level destined toward distant servers can be served over two alternate networks, namely the optical network and the part of the electronic network that is above that point. Connecting the optical switches at the lower levels of the system (i.e., directly to each server) is not considered efficient, due to scalability issues regarding the port count of optical switches, but also to the cost of optical transceivers. Moreover, optical switches exhibit slow reconfiguration times, and thus it makes sense to use them as switching elements of high-rate, long-lived, point-to-point flows, traffic characteristics that are currently exhibited at the rack level or at higher levels of the network hierarchy.

The basic differences among the proposals in [4–6] can be found in the level at which the optical network is connected, in the number and rate of the optical ports per connection point, and in the use of single or multihop connections over the optical network. In Helios [4], the optical network interconnects pods (i.e., sets of several racks containing up to 1000 servers), while in c-Through [5] and OSA [6] the interconnected blocks are the racks. Both Helios and c-Through support only single-hop transmissions over the optical network, while OSA considers multihop connectivity, however, without optimizing this feature in its topology reconfiguration algorithm and also without explaining how to apply multihop forwarding decisions on the switches. It must be noted that including multihop connections in the optimization makes the topology computation intractable for large systems [9] and the implementation of the control software more challenging. Note also that c-Through and OSA assume a single optical switch, while

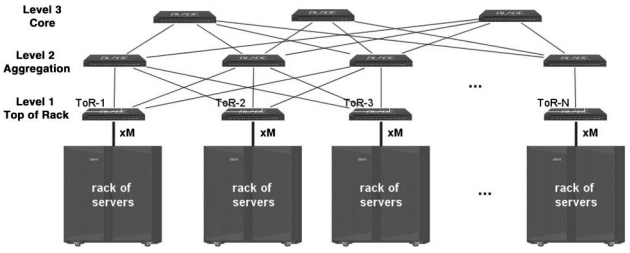

Fig. 1. Example of an electronic-only fat-tree build-out of commodity Ethernet switches.

Helios extends to scenarios in which more than one switch is used. An interesting architectural feature of OSA is that apart from a central optical switch, it uses additional active optical components [wavelength selective switches (WSSs)] and multiwavelength transponders at the ToR switches to enable the establishment of variable rate optical connections. Since the support of multihop and rate adaptable circuits improves the efficiency of the optical network, OSA proposes to completely remove the second level of the electronic network and serve all traffic between racks over the optical core. A quite different architecture that is based on optical WSSs, not MEMS used as fiber/space switches, is the Mordia architecture, which achieves low reconfiguration times in the order of microseconds [13]. The authors proposed to use such a network in a time division multiplexing access (TDMA) manner to share the optical bandwidth among the hosts, which differs significantly from the previously proposed solutions. However, since commercial WSSs do not support many ports (1×9 is the typical size, with 1×20 also available), the scalability of the proposed solution is an open issue.

In this paper we present a hybrid optical/electrical network we call HydRA that follows a similar architecture to [4,5], in the sense that it uses only commodity MEMS optical switches, but also assumes the use of more than one optical switch as in [4] and supports multihop connections over the optical network as in [6]. As opposed to previous solutions [4–6] that did not reach a functioning system, we describe implementation details and present a network controller for our HydRA system that supports multihop optical connections using standard Ethernet switches at the edges of the optical network. Our prototype HydRA network and custom-built controller is used with real applications and is shown to accelerate their execution.

III. System Architecture

A. Data Plane

We depict an example of our reference HydRA system architecture in Fig. 2, comprising $N = 320$ server racks and a dual network option: 1) a high-speed single-level optical network driven by high-speed (10 Gbps) Ethernet ToR switches, and 2) a lower-rate, packet-switched Ethernet network. The server integration factor (32 servers/rack) stems from the currently "standard" 64-port density of high-end 10 Gbps Ethernet ToR switches, allowing construction of full bisection bandwidth fat-trees for racks of such integration.

The high-speed optical network is implemented with an array of $K = 32$ commodity MEMS optical switches. By appropriately cross connecting the latter, we can connect any pair of ToR switches and thus implement direct low-latency connectivity between racks, as opposed to the cumbersome cabling/switching required by multistage electronic interconnects (fat-trees—Fig. 1). Multihop circuit connections can be established over the optical (MEMS) network by appropriately configuring its edges, that is, the Ethernet ToR switches, as will be explained in the following sections. Our control prototype and algorithm provisions bidirectional circuits, but unidirectional circuits can also be provisioned

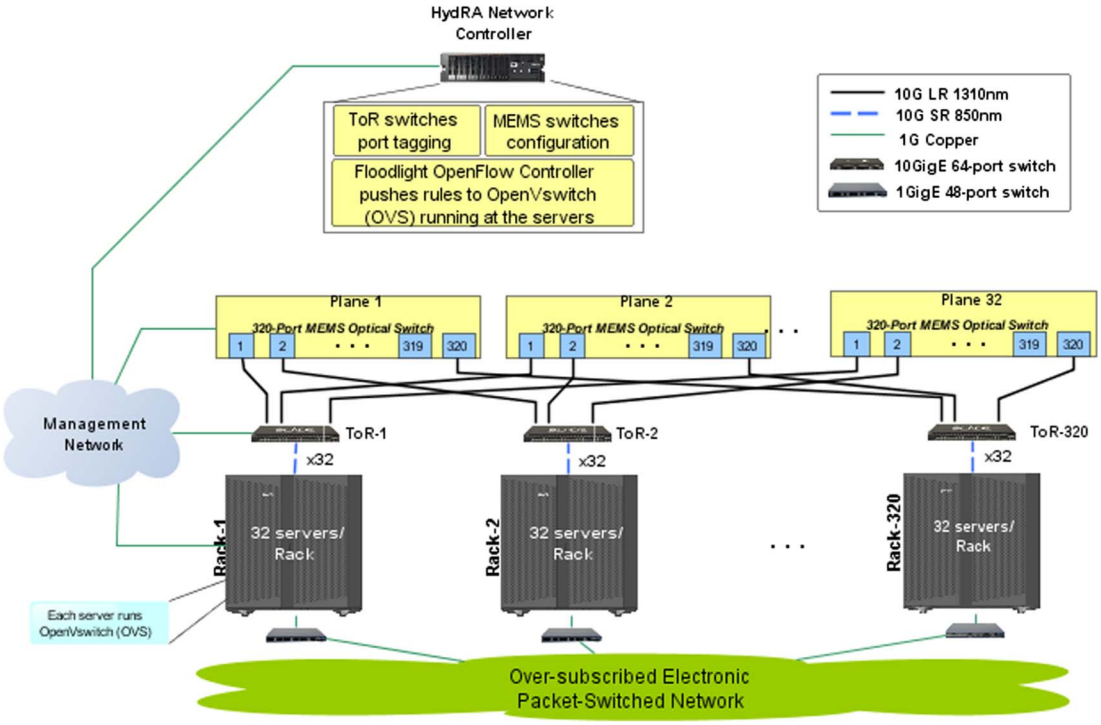

Fig. 2. Proposed HydRA architecture comprising a hybrid optical/electrical network spanning 10,240 servers at an integration factor of 32 servers per rack.

with few changes. MEMS optical switches inherently suffer from high (relative to packet transmission time) switching delay, and therefore are used as circuit-switching elements to carry high-volume, long-lived flows between pairs of racks in our system. Lower-rate or bursty communication occurs in our system via the lower-end electronic network built out of inexpensive Ethernet switches that are aligned in a highly oversubscribed tree topology, to keep a low cost. In Fig. 2 the lower-end electronic network is assumed to be connected directly to the servers (and thus dual-home servers are required), but a variation of the proposed architecture includes the case in which that network is connected to the high-end 10 Gbps Ethernet ToR switches.

The low-rate electronic part of our network architecture can be realized with well-researched solutions (e.g., [2]) for implementing large-scale networks over redundant topologies of Ethernet switches. Since solutions for this part have matured, we focus our research and system prototyping efforts on the control plane of the optical part of our network architecture.

B. Control Plane

All devices making up the network (MEMS optical switches and ToR switches) and servers connect via a low-rate network (typically the provisioning/management network of the datacenter) to a dedicated server, where the *HydRA network controller* is running. The main role of the HydRA network controller is to periodically or upon application request configure the optical part of the network in a manner that benefits the execution of parallel/distributed applications.

At a high level, the controller delivers this as the result of a three-step cycle (Fig. 3):

1) monitors network traffic/gathers application demands;
2) calculates the new logical network topology that maximizes throughput, given the available network resources, and translates that to the physical network;
3) reconfigures the network: configures the array of MEMS optical switches and the edge ToR switches such that the desired topology is realized. Moreover, to realize the forwarding substrate of our implementation the controller has to also communicate appropriate information to the end-systems/servers (as will be explained in the following section).

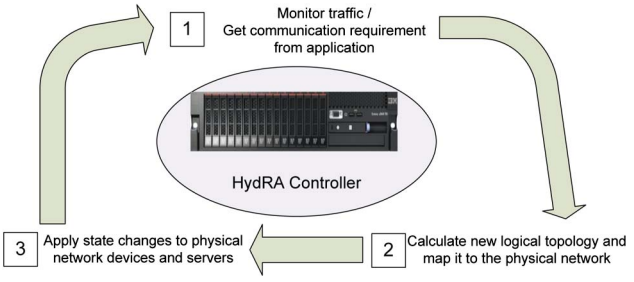

Fig. 3. Steps taken by the HydRA network controller in each reconfiguration cycle.

As we can see in Fig. 3 there are two different modes of operation for step 1. The controller can monitor the network (this can be a periodic process) and configure the optical network accordingly to fit the traffic fluctuations. An alternative way of operating the HydRA network is to have applications interacting with the controller asking for bandwidth in an *on demand* manner. This would be enabled by an appropriate API at the HydRA controller that would enable the applications to configure the network according to their needs. New applications can be written to take advantage of such bandwidth-on-demand service. Moreover, current applications such as MPI applications or MapReduce/Hadoop would require a meta-scheduler to be placed between the application scheduler and the HydRA network controller to translate the application specifications to network requirements, similar to the job that Mesos [14] does for the computation workload.

1) Forwarding Substrate Implementation: In our previous work [11], we showed that supporting multihop optical circuits can reduce the need for frequent reconfiguration of the optical switch array, while preserving network throughput. In addition, we want to support the ability for multiple physical paths (involving either identical or distinct sets of links) between rack pairs for increased throughput. These requirements cannot be satisfied using the conventional spanning-tree protocol (STP). Moreover, when the topology changes, the STP, even in its rapid version, converges relatively slowly (a few seconds or higher). So instead of using the STP, we fully disabled it and implemented our forwarding substrate using VLAN: each rack-to-rack circuit is tagged with a distinct VLAN-ID, and servers tag packets with the VLAN-ID of the optical circuit that they will use to reach the destination rack. The approach is depicted in Fig. 4. Note that intermediate racks of a circuit can also be accessed using the same VLAN-ID. A similar technique has been employed in [15] using static VLAN allocation, which has scalability issues. We measured that we can tag up to 32 ToR switch ports in less than 1.5 s [16], and therefore we propose a dynamic VLAN allocation for increased scalability. Note that the proposed solution is similar to SPAIN [17], with the key difference that VLAN paths are calculated and implemented dynamically when the optical network is reconfigured, since the underlying physical topology changes.

To realize VLAN-based forwarding, flows should be tagged appropriately at the servers with the VLAN-ID corresponding to the circuit that brings it to its destination rack, ideally in an application-unaware fashion (in order to support existing applications, hide details from users/developers). To do this, we inject a translation layer into the network stack of each server. In particular, in our implementation we used OpenVswitch (OVS) [18] at the servers that were centrally controlled by a floodlight (FL) OpenFlow (OF) controller [19] that resided in our HydRA network controller. We prototyped appropriate FL modules and interfaces between our HydRA network controller and FL.

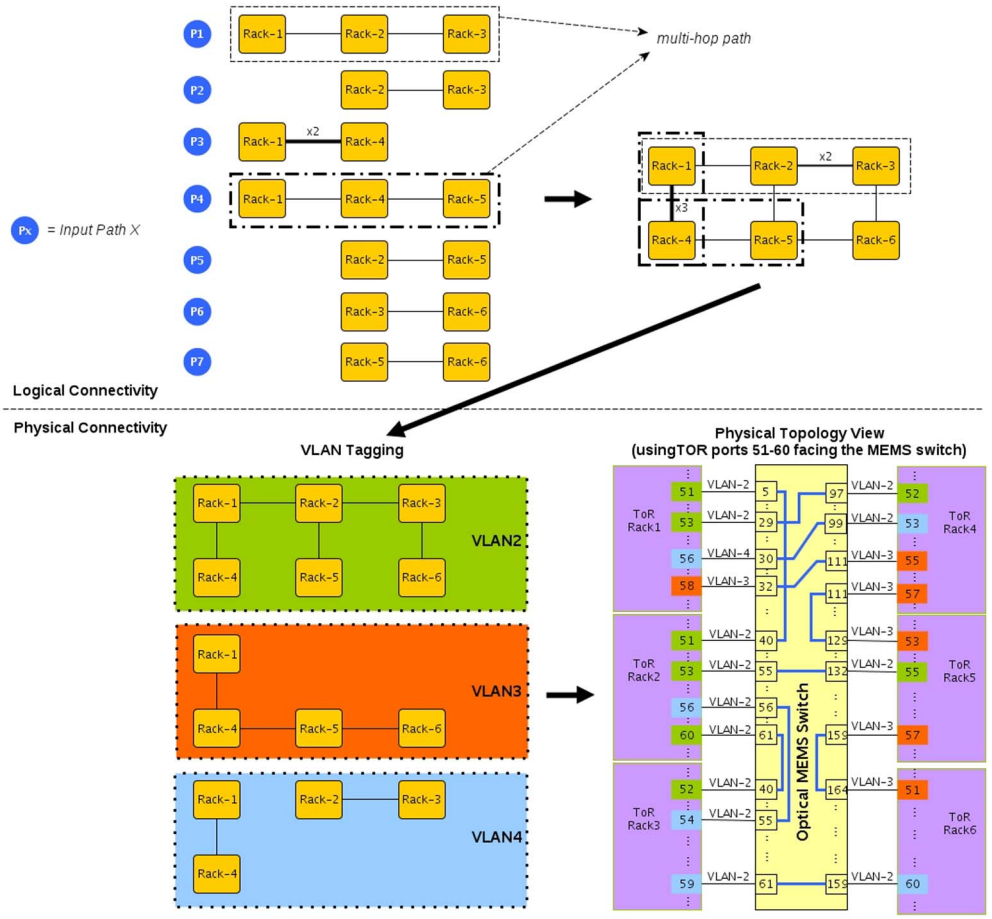

Fig. 4. Example of forwarding based on VLANs. The top part of the figure shows logical connectivity as presented by applications to the controller, while the bottom part shows a mapping of the logical connectivity to VLAN-IDs and the resulting physical connectivity across ToR switches and a MEMS optical switch. In particular, the seven initial required connection paths between six racks (shown on the top left) are merged (top right) and mapped into three VLANs (bottom left). Note that more than one path can be mapped into one VLAN so as to minimize the number of VLAN-IDs used. The constraint for the allocation is that each VLAN should be loop free.

a) Network reconfiguration: We now focus on the third step of the control cycle and explain how the network is reconfigured and the new network topology is realized. The HydRA network controller orchestrates the reconfiguration in three substeps:

3-1) The network controller pushes rules to the OVS switches running at the servers (via the FL instance), so as to route traffic that is affected by the reconfiguration through the electronic network.

3-2) The network controller reconfigures the array of MEMS (optical network) and in parallel tags the ports of the ToR switches with appropriate VLAN-IDs, so as to implement the desired forwarding substrate, over single-hop or multihop circuits. In parallel, the controller also pushes (via the FL instance) the new routing rules to the OVS switches, which have, however, lower priority than the rules written in substep 3-1, so they are inactive.

3-3) The network controller (via the FL instance) removes all the OVS rules written in substep 3-1, so that the rules written in substep 3-2 become active and traffic is forwarded over the newly configured optical topology.

There are a number of delays introduced when reconfiguring the HydRA network. First, the reconfiguration time of MEMS switches (substep 3-2) is relatively high, typically in the order of tens to hundreds of milliseconds. This defines the lower bound, but there are more delays that are introduced by the implementation of our network controller, i.e., the time to install the forwarding state to the network at substep 3-1 (assuming topology calculation is done in advance—substep 3-2) and also the time to switch the traffic to the electronic and back to the optical network taking place in substeps 3-1 and 3-3. The number of rules to be pushed in substep 3-2 is, in the worst case, equal to $M \cdot N^2$, where N is the number of racks and M is the number of servers per rack (the worst case includes having one dedicated circuit for each pair of racks). Viewed differently we have to install N rules (for each destination rack) at each of the $M \cdot N$ servers of the system. So the worst case calculation assumes that the whole network is reconfigured, all racks communicate with all other racks, and N

optical MEMS are available, which seems exceptionally demanding. We target systems that will never have to perform such extensive reconfiguration, but the fact is that the rules pushed in substep 3-2 are more than those of substeps 3-1 and 3-3, which are, in the worst case, linear to the number of racks and equal to $M \cdot N$. Moreover, ToR VLAN port tagging, performed in substep 3-2, is rather slow, and thus it is evident that substep 3-2 dominates the whole reconfiguration cycle. In our controller implementation the rules were proactively pushed using a custom-developed FL module, measured to be at least 50 times faster than the static flow pusher API of FL. However, these rules were sequentially pushed to each OVS, and thus there is still space to optimize the module and reduce the reconfiguration delay. If the rules were pushed in parallel to all servers of the system, then we would have to divide their number by $M \cdot N$, which is the number of servers in the system, although we believe that it would be complicated to perfectly parallelize this process.

2) OpenFlow Forwarding Substrate: The forwarding scheme we implemented in our controller is based on the VLAN-IDs, as presented in Subsection III.B.1, and uses OVS and OF at the end-system servers. A "full OF" solution could also be used, where OF-enabled ToRs and appropriate OF rules would create the circuits between the ToR switches. In this case, there is no need to use OVS at the servers, but all switching functionality would be installed in the OF-enabled ToR switches. Although the full OF solution would be much sleeker (it removes the complexity of OVS source routing) and faster (flow rule installation is considerably faster than VLAN tagging [16]), the solution requires native OF switches across the datacenter. The latter is still far from being a reality, and therefore the solution we adopted focuses on systems deployed today, based on Ethernet commodity switches and built-in VLAN functions. Still, we are porting our prototype to support an OF-only datacenter solution.

3) Topology Calculation Algorithm: The role of the network controller is to compute a "good" configuration of the optical network, given input on communication requirements and the current network configuration, and take all necessary actions to apply the logical topology on the physical network. The controller uses an appropriate heuristic algorithm to calculate the new topology (substep 3-2 of the control cycle). The input comes in the form of a traffic matrix between the racks of the system. This can be defined by knowing in advance the communication requirements of the workload that is going to be executed. However, a more complex operation mode can be envisaged that involves monitoring the traffic at the rack level and predicting the future traffic characteristics. For the former case we can have the execution of a static HPC application for which we know the processor-to-processor communication pattern [9] or an application that specifically requests the establishment of connections, e.g., a MapReduce scheduler. For the latter case we can envision a multitenant DC environment in which multiple applications are simultaneously executed and the network is optimized to follow their traffic variations. Monitoring a large network, which includes collecting and processing such information at close to real time, would be quite demanding and is expected to not scale well. A detailed description of a solution that monitors traffic at end-hosts (servers) was presented in [5], requiring control of buffers and seamless modifications to the network protocol stack at end-hosts. Closing the loop and providing monitoring feedback to the HydRA network and evaluating such solution is part of our future plans.

The algorithm, apart from the traffic matrix, is also given the network resources that will be reconfigured in that cycle (can be the full network or a subset defined in the form of specific network resources allocated to the application that we optimize). The algorithm computes a connectivity graph between the racks involved and the paths to be followed for these communications, aiming at maximizing the network throughput and thus speeding up application(s) completion. Note that the optical network can be fully or partially reconfigured, by specifying appropriately the traffic matrix and the resources to be reconfigured at each cycle.

The heuristic topology configuration algorithm we devised is a variation of the algorithm presented in [9]. The algorithm takes the traffic matrix, orders the demands, and examines them one-by-one in rounds. At each round the algorithm establishes paths of a specific hop distance that is increased at the next round up to a given threshold distance H. In the first round, for a demand s-d (s is the source ToR switch and d is the destination ToR switch) the algorithm examines whether it can establish a cross connection on some optical switch to which s and d are connected (this optical switch has to have available both ports accessing the ToRs). If it finds a cross connection, it establishes it, stores this as the path to be followed for s-d, marks these optical ports as used, removes s-d from the ordering, and moves to the next demand. On the other hand, if it does not find a cross connection, it moves to the next demand and considers s-d again in the next round. After it finishes examining all connections it goes to the second round and examines the establishment of two-hop connections, serving demands left over from the first round (demands that were not served by single-hop connections). For a demand s-d it runs the breadth-first-search (BFS) algorithm on an auxiliary graph. The auxiliary graph comprises N nodes (representing the ToR switches) and has links between two ToR switches that either 1) have free optical ports in at least one of the optical switches or 2) were previously cross connected and have remaining capacity higher than that required. The BFS algorithm is stopped after two iterations, meaning that we find only two-hop shortest paths. If d is reached in these two iterations of the BFS, we find the exact ports to cross connect, and we update the network state and move to the next demand. If d is not reached in two hops, then demand s-d is not served in this second round and we move to the next demand. After examining all demands for two hops we move to the next round, where we examine three-hop connections. This process is continued until there are no more demands or optical ports left, or we reach the hop distance threshold H.

As discussed the devised algorithm is a variation of the algorithm proposed in [9], where instead of the Johnson algorithm we use the BFS algorithm, assuming a Boolean traffic matrix input representing the required ToR pairs' connectivity. The complexity of the algorithm is $O(H \cdot N^3 + H \cdot K \cdot Z^2 \cdot N^2)$, where N is the number of racks, K is the number of OCS planes (number of MEMS switches), Z is the number of ports per ToR per OCS plane, and H is the maximum path length threshold. Parameter H can be controlled, so we can trade off the running time for performance, which is particularly useful when the calculation has to be performed in a limited time budget. Previous hybrid proposals [3,4] considered only single-hop connections over the optical network. Although we consider multihop connections that make the related optimization problem intractable [9], the proposed heuristic is designed to have comparable complexity to algorithms optimizing only single-hop connections.

C. Scalability and Resilience

The reference architecture presented in Fig. 2 is for a system of $N = 320$ racks, a size that is mainly driven by the maximum port number (radix) of MEMS switches used. Higher radix optical switches have been prototyped or can be created by connecting several MEMS switches in a Clos network, although such an approach would be very expensive. Alternative solutions to scale the HydRA architecture are 1) to create an oversubscribed network in which the MEMS switches are connected in a regular manner to subsets and not all the racks or 2) to use the reference 320-rack system as a building block and define a second layer of MEMS switches or another networking solution to interconnect such blocks. With respect to reliability, the proposed architecture, since it relies on a centralized controller, has a single point of failure, and the known solutions for centralized architectures are applicable (e.g., a replicated controller). Failures of the MEMS or the ToR switches can be taken into account in the resource allocation process, requiring a mechanism to forward such failures to the controller. Such a mechanism has not been included in our prototype system design, but a standardized solution can be used.

IV. Price Considerations

We estimate the total list price of our proposed HydRA network and compare it against a conventional electronic fat-tree network implemented with top-end Ethernet switches [2] (see Fig. 1). Our architecture employs solely commodity off-the-shelf equipment, and we used in our comparison current list prices drawn from publicly available sources [20] (collected in early 2013), with the exception of the optical MEMS switches, for which we used an averaged representative list price after discussions we had with vendors.

Table I shows the amount of items required to build a hybrid optical/electrical network such as HydRA for a 10,240 server cluster (320 racks of 32 servers each, as presented in Fig. 2), as a function of the size K of the array of MEMS switches that it is built with. The pricing of the lower-rate electronic network—although quite low—is also included in our price calculations graphed in the following figure, though we omit elaborating on its price structure in Table I. Note that according to the definition of bisection bandwidth [21], a network build with an array of $K = 32$ MEMS switches provides full bisection bandwidth, since for any bisection (partition of the servers in two equal parts and a choice of sources/sinks), the array of optical switches can be configured so that all sources/sinks communicate at the full bandwidth (10 Gpbs).

Table I also shows the amount of items required to build an electronic-only fat-tree network, this time parametric to its oversubscription ratio β. Taking the approach of [2] we assume that we can build a network that provides full bisection bandwidth, denoted here by $\beta = 1$, by interconnecting three levels of 800 64-port 10G Ethernet ToR switches in total. In the general case, to build a β-over subscribed fat-tree network we assume that the bisection bandwidth is reduced by $\sqrt{\beta}$ at each level, to get the overall ratio of β between the first and third levels of the tree.

Using the price values listed in Table I and the item quantities calculated for each network separately above, we calculated the total list price of HydRA and a fat-tree network interconnecting a 10,240 server cluster and plot the results parametric to K for the hybrid and to β for the fat-tree network, respectively, in Fig. 5. For networks

TABLE I
List of Equipment and Corresponding List Price per Item Used in the Analysis

Symbol	Equipment Name	Equipment Description	Price/Item (US $)	Number of Items for HydRA of 10,240 Servers	Number of Items for a β Oversubscribed Fat-Tree Network of 10,240 Servers
S_{10G}	IBM RackSwitch G8264	64 port 10 Gbps Ethernet switch	30,000	320	$320 + 320/\sqrt{\beta} + 160/\beta$
T_{SR}	IBM SFP+ SR transceiver	10 Gbps short-reach (SR) 850 nm transceiver	665	320×32	$10,240 + 20,480/\sqrt{\beta} + 20,480/\beta$
T_{LR}	IBM SFP+ LR transceiver	10 Gbps long-reach (LR) 1350 nm transceiver	1600	$320 \times K$	—
O_{320}	320×320 MEMS optical switch	320×320 nonblocking optical switch	110,000	K	—
C_{MM}	LC-LC 50 μm fiber cable	Multimode fiber cable	28	320×32	$5,120 + 10,240/\sqrt{\beta} + 10,240/\beta$
C_{SM}	9 μm fiber cable	Single-mode fiber cable	25	$320 \times K$	—

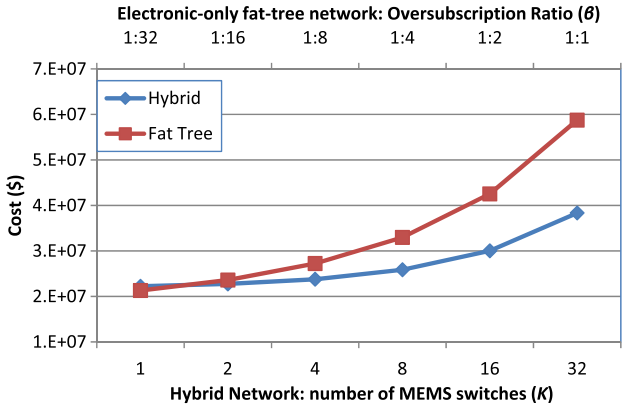

Fig. 5. List price of a hybrid network such as HydRA and electronic-only fat-tree network interconnecting a 10,240 server cluster at various capacity levels. The parameter K denotes the number of optical MEMS switches used to build the hybrid network, and the parameter β denotes the oversubscription ratio of the electronic-only network.

that provide full bisection bandwidth, that is, for $K = 32$ for the hybrid and for $\beta = 1$ for the electronic-only fat-tree, we observe that the HydRA network is about 30% cheaper. Note that the optical part of our hybrid network is assumed to be built with long-reach (LR) single-mode transceivers and optical MEMS switches operating with single-mode fibers, since these switches scale to higher port count (i.e., 320×320 assumed here) than multimode ones. This restricts our transceiver choice, since short-reach (SR) single-mode transceivers are not very common due to the scarcity of applications for such components. LR transceivers are much more expensive than SR ones (about 3×), while there is no need in terms of reach and power budget to use them in a datacenter environment. Thus, the savings that the hybrid network could bring forward would be much higher if the price discrepancy of the transmitters employed was not that high.

Note that to increase the capacity of the hybrid network (e.g., go to 40 Gbps) we only need to upgrade the electronic edge that accesses the OCS network, since the MEMS switches are protocol and rate agnostic. Thus, the upgrade cost would be much smaller for the hybrid than for the fat-tree network. Lastly, note that the optical technology, due to its passive forwarding nature, consumes much less energy compared to active electronic switching, a cost-saving factor not captured in the above model.

In summary, a full bisection bandwidth hybrid network ($K = 32$) is cheaper than the equivalent bandwidth fat-tree network ($\beta = 1$). However, in addition we want to see how the two networks compare against application performance. Thus, in the next section we invert the problem and examine the performance of applications on equal-cost hybrid and fat-tree networks.

V. Prototype Implementation

We prototyped HydRA, our reference hybrid optical/electrical architecture including the network controller and VLAN-based forwarding (as described in Section III) and conducted various trials to validate our implementation. Our testbed comprised 40 servers (12 cores each) mounted in four racks with a 10G Ethernet ToR switch (IBM RackSwitch G8264) per rack and one 96×96 MEMS optical switch (Crossfiber Liteswitch 96). Each server connects via a 10G SR transceiver to its rack's ToR switch, and in turn each ToR switch connects via 10 LR single-mode transceivers to the optical switch. We also used five additional 10G Ethernet switches to implement slices of three-level electronic-only fat-tree networks. Our network controller ran on a dedicated server that connects via a 1 Gbps network to the management ports of the ToR switches and the optical switch, as well as to all servers. The same 1 Gbps network was used as the electronic part of the hybrid network.

We performed two sets of experiments. In the first set we evaluate the performance of the HydRA network when we statically configure it to serve the communication requirements of an application before its execution and compare it against equal-cost electronic-only fat-tree networks. In the second set we focus on the dynamic reconfiguration of HydRA. We measure the reconfiguration delay of our fully functional prototype, taking into account delays introduced by our software control stack and the hardware. We also demonstrate that by dynamically reconfiguring the network to avoid congestion we can speed up the execution of the parallel workloads.

A. Static Network Topology Configuration Experiments

The rationale behind the first set of our experimental scenarios is to compare instantiations of our HydRA network against equal-cost instantiations of an electronic-only fat-tree, and in fact do so using real applications. To this end, we used our cost models to obtain two scenarios: scenario 1 compares a HydRA network with an array of 20 optical switches against a 1:4 oversubscribed fat-tree, and scenario 2 compares a HydRA network with an array of six optical switches against a 1:25 oversubscribed fat-tree.

Our use case involves a 10,240 server cloud datacenter with 32 servers per rack and a user requesting to execute a parallel job assuming an infrastructure or a platform-as-a-service environment. To address the general case in which this may lead to the allocation of servers without physical proximity—due to virtualization, or fragmentation of the resource—we assume that the user gets servers in different racks and inter-rack communication in the three-level fat-tree case traverses the root of the tree. For scenarios 1 and 2 we assume the use of 8 and 10 servers in four racks, respectively, and a uniform bandwidth allocation to servers in each rack. As such, the 8 servers in each rack in scenario 1 are allocated 1/4 of the available inter-rack bandwidth, while the 10 servers in scenario 2 are allocated 1/3 of the available inter-rack bandwidth, in both the HydRA and fat-tree networks. Figure 6 presents in detail our comparison scenarios. Note that to create these scenarios we assumed datacenter networks of the same total cost for

the electronic packet and hybrid architectures and took the same "slice" of those networks consisting of a number of servers in different racks and the relative slice of the capacity. The packet-switched network provides less capacity to the examined slice (and to any other slice spanning many racks) than the circuit-switched network due to the cost discrepancy of the two solutions as shown in Fig. 5.

We executed the following MPI parallel applications over all four network configurations shown in Fig. 6: FFTW [22], which is a discrete fast-Fourier transform kernel; the FT (discrete 3D fast Fourier transform) kernel; the MG (multigrid on a sequence of meshes) kernel; and the SP (scalar penta-diagonal solver) pseudo-application—the last three are part of the NAS Parallel Benchmarks (NPB) suite [23]. For FFTW we solved 3D FFT problems, the size of which were multiples of the total number of servers that were used in each scenario (32 in scenario 1, and 40 in scenario 2), while for NPB we executed class D and E problem sizes. In the case of the HydRA network, the execution involved using our network controller stack and utilizing our VLAN and end-system translation solution. Note that in the HydRA network configurations of both scenarios, the constructed topologies include loops, which would be broken by disabling one or more links, if standard Ethernet/STP switching was used. Instead, our VLAN-based routing enabled the loops, thus yielding higher throughput over the same network configuration. Figures 7(a) and 7(b) show the speedup results obtained for scenarios 1 and 2, respectively. Speedup is defined as the ratio of the completion time of the application execution in the tree network over the completion time in the HydRA network. Results show acceleration across almost all of the examined workloads up to a peak of 35%.

In the scenarios examined above we assumed that the HydRA network is "statically" configured before the user starts executing applications. However, other use cases may require dynamic reconfiguration of the network during application execution. In the following section we measure the reconfiguration time of our prototype and also show how this affects the execution of an application.

B. Network Reconfiguration Experiments

Figure 8(a) presents the topologies used to measure the reconfiguration delay of our HydRA electrical/optical

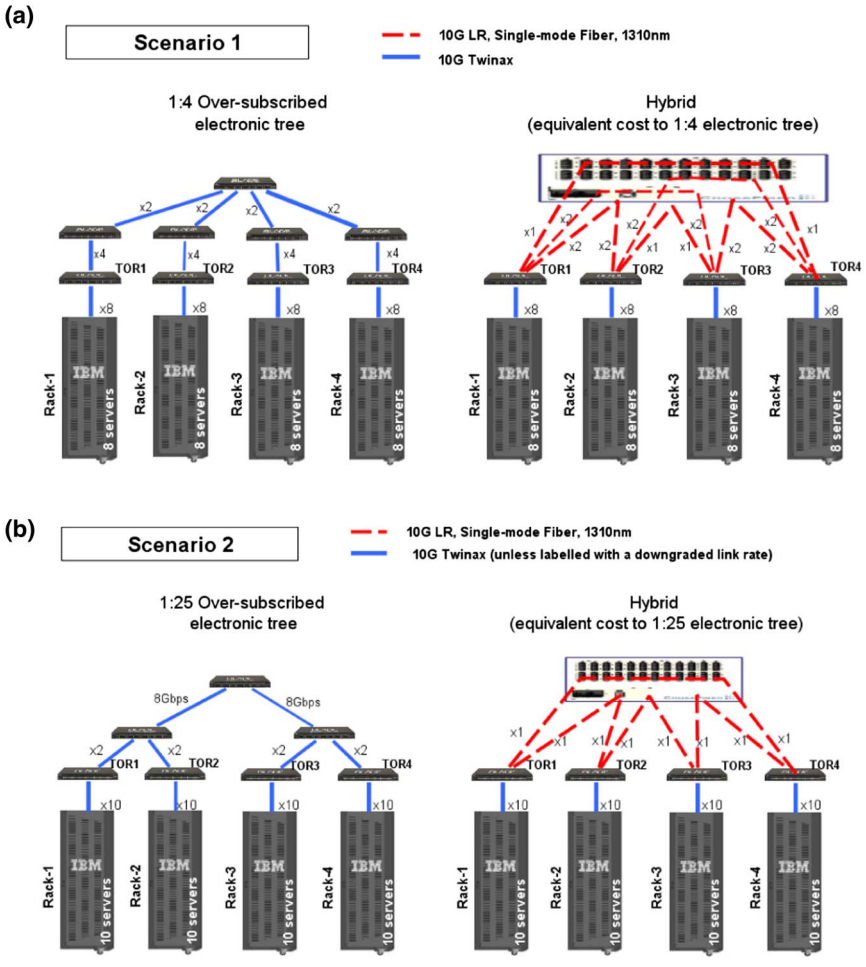

Fig. 6. Network configurations implementing the two experiment scenarios for the two network types: scenario 1 shown in (a) compares a 1:4 electronic-only tree network to a cost-equivalent HydRA network, and scenario 2 shown in (b) compares a 1:25 electronic-only tree network to a cost-equivalent HydRA network.

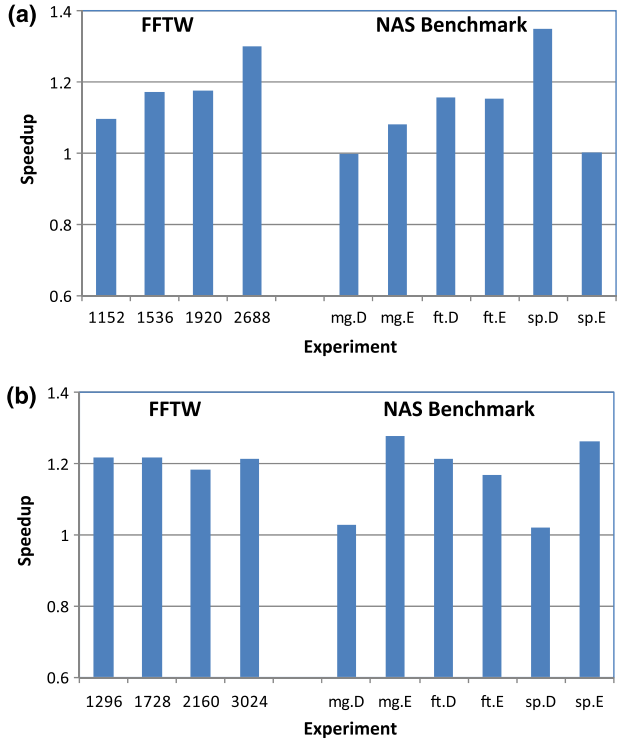

Fig. 7. Speedup achieved by the HydRA hybrid optical/electrical network over the electronic-only fat-tree of equivalent cost to the various workloads tested in scenario 1 [shown in (a)] and scenario 2 [shown in (b)].

prototype. We used two servers connected to 2×10 Gbps Ethernet ToR switches, which were in turn connected to the optical MEMS switch. The servers were also connected via the 1 Gbps Ethernet switch (the electrical part of the HydRA network). We used "*iperf*" to measure the bandwidth between the two servers. Using topology 1 of Fig. 8(a) we forwarded traffic via the ToRs and the optical switch and measured an average bandwidth of 9.21 Gbps [blue line in Fig. 9(a)]. Next, we used our network controller to reconfigure the network between topologies 1 and 2 of Fig. 8(a) every 10 s. Note that during reconfiguration the traffic was forwarded over the 1 Gbps electronic network (see substeps 3-1 and 3-3 of Subsection III.B). We performed the same experiment twice, pushing 100 and 5000 OF rules at substep 3-2 to the OVS switches running at the servers to emulate a small and a medium size datacenter. Note that as also discussed in Subsection III.B.1.a the number of rules pushed at substep 3-2 in the implemented control cycle corresponds to the number of different circuits established between the ToR switches and can be in the worst case $M \cdot N^2$, where N is the number of racks and M is the number of servers per rack (assuming that the whole network is reconfigured, all racks communicate with all other racks, and we do not parallelize the process of writing the rules at the servers). Pushing 5000 rules corresponds to the case in which 1/10 of the 320 racks of the system participate in the reconfiguration, each rack communicates with five other racks, and all these are new rules that are installed, so none of the rules previously installed

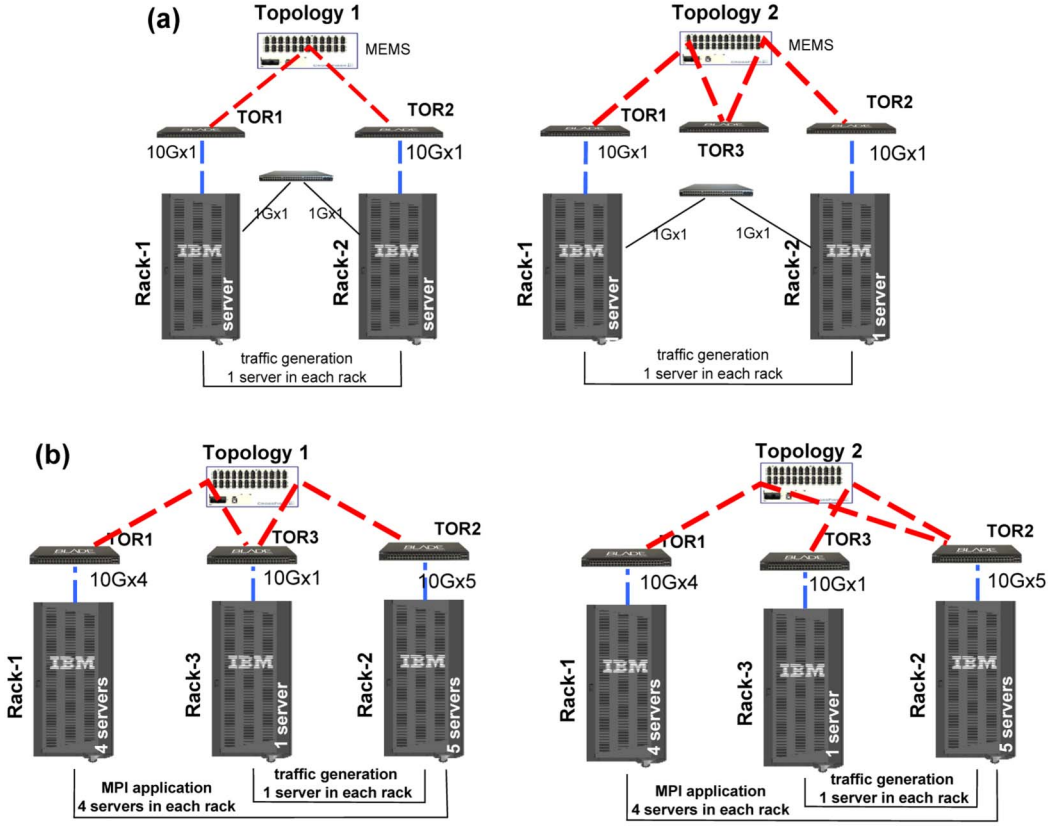

Fig. 8. (a) Topologies used to measure reconfiguration delays. (b) Topologies used to measure applications' execution times.

are kept. We consider this as a demanding reconfiguration scenario and expect a 320-rack system to perform quite smaller reconfiguration cycles. Note that tagging the ToR ports with VLAN-IDs is performed in parallel with OF rule pushing (substep 3-2) and that this substep dominates the whole reconfiguration cycle.

The red and green lines in Fig. 9(a) show the measured bandwidth as a function of time. We see that the bandwidth dives and jumps are quite sharp, indicating that switching between the 10 Gbps optical and the 1 Gbps electrical network and the opposite (substeps 3-1 and 3-3), are swift. We measured the average bandwidth to be 8.37 and 7.86 Gbps, respectively. The average time to tag the ToR ports was measured to be 1.2 s, and this was the reconfiguration delay we measured when pushing 100 OF rules, since tagging dominated the reconfiguration cycle. However, the delay of pushing 5000 rules was higher, which explains our finding of a 2.2 s average reconfiguration delay in that experiment.

Next we run the FFTW MPI application on two racks (four servers/rack) that were connected via the 10 Gbps optical network [racks 1 and 2—topology 1 of Fig. 8(b)]. The blue line of Fig. 9(b) shows the measured bandwidth as a function of time. We see that the bandwidth peaks at 3.54 Gbps, while the application finished at 28.7 s. We did the same experiment, but we added background traffic between racks 2 and 3 this time, creating network congestion that affected the application. The red line of Fig. 9(b) shows the bandwidth as a function of time (peak: 1.55 Gbps), with the application finishing at 62.2 s. Finally, we did the same experiment using our HydRA network controller to reconfigure from topology 1 to 2 of Fig. 8(b) after 15 s. The green line of Fig. 9(b) shows the bandwidth as a function of time for this final run. It can be clearly seen that initially the application is running slowly due to network congestion, while after the reconfiguration it immediately increases its network utilization and finishes at 36.7 s. Note that the acceleration depends on the reconfiguration trigger that was given at 15 s in this experiment. Similar results were obtained for other HPC applications. We did not discuss here mechanisms to detect congestion and trigger the reconfiguration or how applications can interact with the controller to request bandwidth on demand. We also did not focus much on the topology calculation phase, but outlined the algorithm in Subsection III.B.3, but instead we consider a realistic operational environment aiming to show the potential that this architecture can achieve. So we prototyped the system and we focused on the achieved reconfiguration delay and showed that even though in our implementation this is high—in the order of seconds—we can still achieve significant performance improvement.

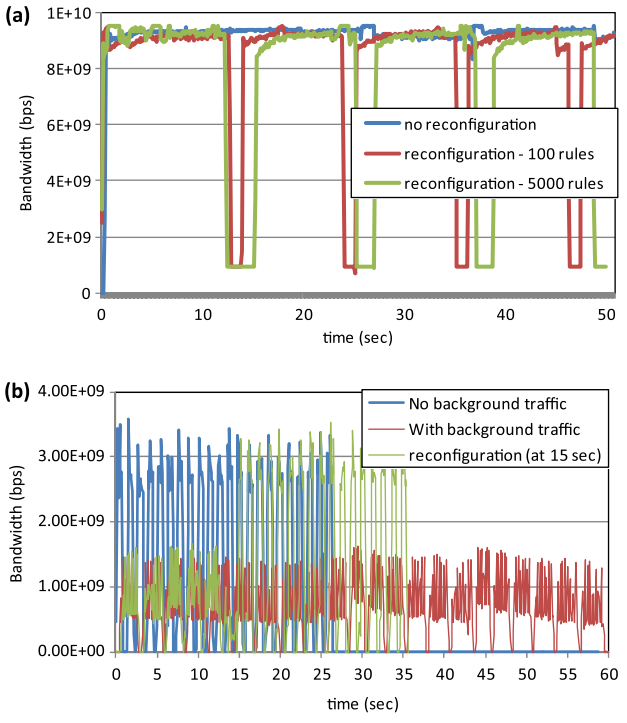

Fig. 9. (a) Maximum achieved bandwidth as a function of time when no reconfiguration happens, when we reconfigure every 10 s and push 100 rules, and when we push 500 rules. (b) Used bandwidth as a function of time by FFTW application, when application is executed without background traffic, with background traffic, and with background traffic that is avoided by reconfiguring the network after 15 s.

VI. Conclusions

In response to the increased communication density and high bandwidth of datacenters, hybrid optical circuits/electrical packet-switched networks are affordable candidates compared to conventional dense multistage electronic-only solutions. We delivered a concise price analysis of a hybrid interconnect comprising commodity parts and showed that it is cheaper compared to its most prominent competitor, namely a full electronic fat-tree providing full bisection bandwidth. As technology evolution migrates to 40 Gbps or even to 100 Gbps, we expect that the advantages of optical technologies would be even more magnified, making their adoption more of a necessity than an option. We prototyped our hybrid network called HydRA focusing mainly on its network controller. The role of the HydRA controller is to compute efficient workload-input specific topologies and orchestrate the control planes of the various network devices and the network stack of end-systems involved to create an optical substrate transparent to applications using it. We deployed our HydRA prototype in a four-rack testbed and showed through real experimentation that tested parallel workloads are accelerated at an equal network investment with an electronic-only tree solution, when considering static network configuration scenarios. We also measured the reconfiguration delay of our prototype and observed the modules that compose it. Even though our controller was not fully optimized we demonstrated a substantial acceleration in execution time of parallel workloads when the network is reconfigured to avoid congestion. The migration of our network controller to a full OF solution and evaluation against a broader workload domain are part of our future research agenda.

Acknowledgments

This work has been partially supported by the Industrial Development Agency (IDA), Ireland, and the Irish Research Council for Science, Engineering and Technology (IRCSET).

References

[1] "Cisco Global Cloud Index: Forecast and Methodology, 2011–2016," Cisco White Paper [Online]. Available: http://www.cisco.com.

[2] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *Proc. ACM SIGCOMM*, Seatle, USA, 2008, pp. 63–74.

[3] K. J. Barker, A. Benner, R. Hoare, A. Hoisie, A. K. Jones, D. K. Kerbyson, D. Li, R. Melhem, R. Rajamony, E. Schenfeld, S. Shao, C. Stunkel, and P. Walker, "On the feasibility of optical circuit switching for high performance computing systems," in *Proc. ACM/IEEE Conf. on Supercomputing*, 2005, p. 16.

[4] N. Farrington, G. Porter, S. Radhakrishnan, H. H. Bazzaz, V. Subramanya, Y. Fainman, G. Papen, and A. Vahdat, "Helios: A hybrid electrical/optical switch architecture for modular data centers," in *Proc. ACM SIGCOMM*, New Delhi, India, 2010, pp. 339–350.

[5] G. Wang, D. G. Andersen, M. Kaminsky, K. Papagiannaki, T. S. E. Ng, M. Kozuch, and M. Ryan, "c-Through: Part-time optics in data centers," in *Proc. ACM SIGCOMM*, New Delhi, India, 2010, pp. 327–338.

[6] K. Chen, A. Singla, A. Singh, K. Ramachandran, L. Xu, Y. Zhang, X. Wen, and Y. Chen, "OSA: An optical switching architecture for data center networks with unprecedented flexibility," in *USENIX Conf. on Networked Systems Design and Implementation (NSDI)*, San Jose, CA, 2012.

[7] S. Kamil, L. Oliker, A. Pinar, and J. Shalf, "Communication requirements and interconnect optimization for high-end scientific applications," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 2, pp. 188–202, 2009.

[8] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Conf. on Internet Measurement (IMC)*, 2010, pp. 267–280.

[9] K. Christodoulopoulos, K. Katrinis, M. Ruffini, and D. O'Mahony, "Tailoring the network to the problem: Topology configuration in hybrid electronic packet switched/optical circuit switched interconnects," *Concurr. Comput. Pract. Exper.*, vol. 25, no. 17, pp. 2412–2432, Dec. 2013.

[10] D. Lugones, K. Christodoulopoulos, K. Katrinis, M. Ruffini, D. O'Mahony, and M. Collier, "Accelerating communication-intensive parallel workloads using commodity optical switches and a software-configurable control stack," in *European Conf. on Parallel and Distributed Computing (Euro-Par)*, Aachen, Germany, 2013, pp. 713–724.

[11] D. Lugones, K. Katrinis, and M. Collier, "A reconfigurable optical/electrical interconnect architecture for large-scale clusters and datacenters," in *Conf. on Computing Frontiers*, Cagliary, Italy, 2012, pp. 13–22.

[12] K. Christodoulopoulos, K. Katrinis, M. Ruffini, and D. O'Mahony, "Accelerating HPC workloads with dynamic adaptation of a software-defined hybrid electronic/optical interconnect," in *Optical Fiber Communication Conf. (OFC)*, 2014, paper Th2A.11.

[13] N. Farrington, A. Forencich, P. C. Sun, S. Fainman, J. Ford, A. Vahdat, G. Porter, and G. Papen, "A 10 ms hybrid optical-circuit/electrical-packet network for datacenters," in *Optical Fiber Communication Conf. (OFC)*, Anaheim, CA, 2013.

[14] B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A. Joseph, R. Katz, S. Shenker, and I. Stoica, "Mesos: A platform for fine-grained resource sharing in the data center," in *USENIX Conf. on Networked Systems Design and Implementation (NSDI)*, Boston, MA, 2011.

[15] X. J. Zhang, R. Wagle, and J. Giles, "VLAN-based routing infrastructure for an all-optical circuit switched LAN," in *IEEE Conf. on Global Telecommunications (IEEE GLOBECOM)*, Honolulu, HI, 2009.

[16] K. Katrinis, G. Wang, and L. Schares, "SDN control for hybrid OCS/electrical datacenter networks: An enabler or just a convenience?" in *IEEE Photonics Society Summer Topical Meeting Series*, 2013, p. 243.

[17] J. Mudigonda, P. Yalagandula, M. Al-Fares, and J. C. Mogul, "SPAIN: COTS data-center Ethernet for multipathing over arbitrary topologies," in *Symp. on Networked Systems Design and Implementation (NSDI)*, 2010.

[18] OpenVswitch: An Open Virtual Switch, http://openvswitch.org/.

[19] Project Floodlight, http://www.projectfloodlight.org/floodlight/.

[20] IBM, http://www.ibm.com.

[21] W. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. Morgan Kaufmann, 2003.

[22] M. Frigo and S. Johnson, "The design and implementation of FFTW3," *Proc. IEEE*, vol. 93, no. 2, pp. 216–231, 2005.

[23] NAS Parallel Benchmarks, http://www.nas.nasa.gov/publications/npb.html.