Accelerating HPC Workloads with Dynamic Adaptation of a Software-Defined Hybrid Electronic/Optical Interconnect

K. Christodoulopoulos[1], **K. Katrinis**[2], **M. Ruffini**[1], **D. O'Mahony**[1]
[1] *CTVR, Trinity College Dublin, Ireland, Email: {christok,ruffini,Donal.OMahony}@tcd.ie*
[2]*IBM Research, Ireland, Email: katrinisk@ie.ibm.com*

Abstract: We prototyped a dynamically adaptable hybrid electronic/optical interconnect using commodity switches and measured its reconfiguration delay. By reconfiguring the network at runtime to avoid congestion we accelerated the execution of parallel workloads in a real testbed.

OCIS codes: (060.4256) Fiber Optics, network optimization; (060.4253) Networks, circuit-switched; (200.4650) Optical interconnects

1. Introduction

High performance computing (HPC) and datacenter (DC) systems are being built out of ever increasing numbers of servers. Evolving network capacity in these environments to support increasing computation density incurs massive capital and management costs: DCs and HPC clusters are designed with full-bisection or slightly oversubscribed fat-tree topologies of commodity switches, which exhibit super-linear cost scaling.

In response to this challenge, prior research [1-4] has proposed using commodity optical switches for carrying aggregated traffic between racks or collections of racks, partly or entirely replacing the higher levels of the electronic tree networks. These proposals leverage some interesting features exhibited by optical switches, such as being protocol and rate agnostic, having low cost per port, and being reconfigurable. The optical network envisioned is typically implemented with Micro Electro-Mechanical Systems (MEMS) switches. MEMS are space switches operating at PHY, whereby they reflect a light beam from any input to any output port (crossbar) without processing the signal, and 320x320 switches are commercially available. Although MEMS are reconfigurable, they exhibit circuit configuration times typically in the order of tens of milliseconds. This technology shortcoming is circumvented in the proposed architectures by employing a hybrid of two switching paradigms, namely electronic packet switching (EPS) and optical circuit switching (OCS). The hybrid network is typically designed with a low degree optical part to keep the cost low, but since it is reconfigurable it can adapt and provide capacity wherever it is needed, while the EPS network provides full connectivity at packet timescales - albeit at lower rates. The motivation behind employing such a low-degree and reconfigurable network is that many studies suggest that this fits the requirements of HPC and DC applications [1],[2].

The hybrid optical/electrical network can operate in two modes. Assuming a system with limited prior knowledge of the applications running on it, a periodic control process monitors live traffic and configures the optical network to fit traffic patterns [2-4]. Moreover, such a network can support a true bandwidth-on-demand service and thus enable a second operation mode, where applications are granted access to the network controller (e.g. through an API) and configure the network according to their needs. Applications ranging from typical parallel MPI HPC applications [1] to MapReduce schedulers that place mappers and reducers in a pool of servers can take advantage of a network tailored to their needs. Note that the two above described modes can coexist.

Although hybrid networks were proposed several years ago, very little is still known about their impact on applications' performance; in particular, how dynamic reconfiguration affects them, and how fast this can be performed with commodity equipment. In our previous work [5] we reported on a functional hybrid network prototype including a network controller and used it to configure the network and accelerate the execution of parallel workloads, when compared to equal cost electronic-only fat-tree networks. However, the previous version of our prototype supported only "static" network configurations: the network was configured prior to application execution, not supporting dynamic adaptation during application runtime. In this paper we focus on the dynamic reconfiguration aspect and measure for the first time 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 parallel workloads by dynamically reconfiguring the network during runtime.

2. System Architecture

We depict an example of our reference system architecture in Fig. 1, comprising 320 racks and a dual network: (a) a high-speed single-level optical network built out of an array of commodity MEMS switches and driven by 10Gbps Ethernet Top of Rack (ToR) switches, and (b) a lower-rate, packet-switched and highly over-subscribed Ethernet network. The optical network is circuit switched and carries high-volume, long-lived flows between racks. Lower-rate or bursty communication is served over the electronic network. All network devices and servers connect via a low-rate management network to a dedicated server, where our *hybrid network controller* is running. The main role of the network controller is to periodically or upon request configure the network to benefit the execution of applications.

978-1-55752-993-0/14/$31.00 ©2014 Optical Society of America

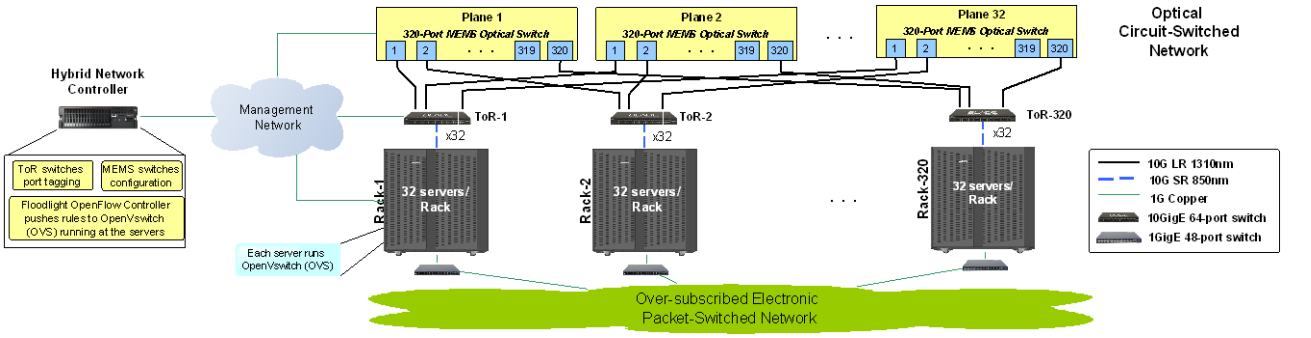

Fig. 1: Proposed architecture comprising a hybrid optical/electrical network spanning 320 racks and 10,240 servers.

Packets to be transferred over the optical network are aggregated at the ToRs and then switched as optical circuits through the MEMS array. Multi-hop optical circuits can reduce the need for frequent reconfiguration. In addition, multiple physical paths between racks can increase throughput. Conventional spanning-tree protocol (STP) cannot satisfy these requirements, while STP convergence time when topology changes is high. Therefore, in our prototype we fully disable STP and implement our forwarding substrate using VLAN: each rack-to-rack (single or multi-hop) circuit is tagged with a distinct VLAN-ID, ensuring that no cycles occur on each VLAN overlay. To realize our VLAN-based forwarding scheme, a server tags a packet with the VLAN-ID corresponding to the circuit that brings it to its destination rack. To implement this we used OpenVswitch (OVS) [6] at the servers, controlled by a Floodlight (FL) OpenFlow (OF) controller [7] which we enclosed in our network controller. We prototyped appropriate FL modules and interfaces between our controller and FL. Note that our approach is application-agnostic, i.e. it does not impose any changes to existing applications, while also hiding the underlying function from the user/developer. The network controller calculates the topology [9] and orchestrates the reconfiguration that is performed in 3 steps:

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.
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- or multi-hop circuits. In parallel, the controller also pushes the new routing rules to the OVS switches, which have however lower priority than the rules written in step 1, so they are inactive.
3. After step 2 has finished, the network controller (via the FL instance) removes all the OVS rules written in step 1, so that the rules written in step 2 become active and traffic is forwarded over the newly configured optical topology.

There are a number of delays introduced when reconfiguring the hybrid network. First, the reconfiguration time of MEMS switches (step 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 step 2 (assuming topology calculation is done in advance), and also the time to switch the traffic to the electronic and back to the optical network taking place in steps 1 and 3. The number of rules to be pushed in step 2 is, in the worst case, square to the number of racks (one circuit for each pair of racks), while the rules pushed in step 1 and 3 are linear to the number of racks. Moreover, ToR VLAN port tagging, performed in step 2 is rather slow. Thus, step 2 dominates the whole reconfiguration cycle. In our controller implementation rules were proactively pushed using a custom developed FL module, measured to be at least 50 times faster than the Static Flow Pusher API. However, these rules were sequentially pushed to each OVS and thus there is still space to optimize the module and reduce the reconfiguration delay.

Note that, instead of a VLAN-based forwarding scheme, a “full OpenFlow” solution could be used, where OF-enabled ToRs and appropriate OF rules would create the circuits between the ToRs, while we would not have to use OVS at the servers. Although the full OF solution would be sleeker (removes the complexity of OVS source routing) and faster (flow rule installation is considerably faster than VLAN tagging [8]), the solution precludes availability of native OF switches across the datacenter. The latter is still far from being a reality and therefore our solution focuses on systems deployed today. Still, we are porting our prototype to embark on an OF-only datacenter era.

3. Performance Experiments

Fig. 2a presents the topologies used to measure the reconfiguration delay of our hybrid electrical/optical prototype. We used 2 servers connected to 2x 10Gbps Ethernet ToR switches (IBM RackSwitch G8264R), that were in turn connected to an optical MEMS switch (Crossfiber LiteSwiich-96). The servers were also connected via a low-end 1Gbps Ethernet switch (the electrical network). We used “iperf” to measure the bandwidth (bw) between the two servers. Using topology 1 of Fig. 2a we forwarded traffic via the ToRs and the optical switch and measured an average bw of 9.21 Gbps (blue line in Fig. 3a). Next, we used our network controller to reconfigure the network between

topologies 1 and 2 of Fig. 2a every 10 sec. Note that during reconfiguration the traffic was forwarded over the 1Gbps electronic network (see steps 1 and 3 above). We did the same experiment twice, pushing 100 and 5000 OF rules to the OVS switches running at the servers to emulate a small and a medium size DC. Tagging the ToR ports with VLAN-IDs is performed in parallel with OF rule pushing (step 2). The red and green lines in Fig 3a show the measured bw as a function of time. We see that the bw dives and jumps are quite sharp, indicating that switching between the 10 Gbps optical and the 1 Gbps electrical network is swift. We measured the average bw to be 8.37 and 7.86 Gbps, respectively. The average time to tag the ToR ports was measured to be 1.2 sec, 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 2.2 sec average reconfiguration delay in that experiment.

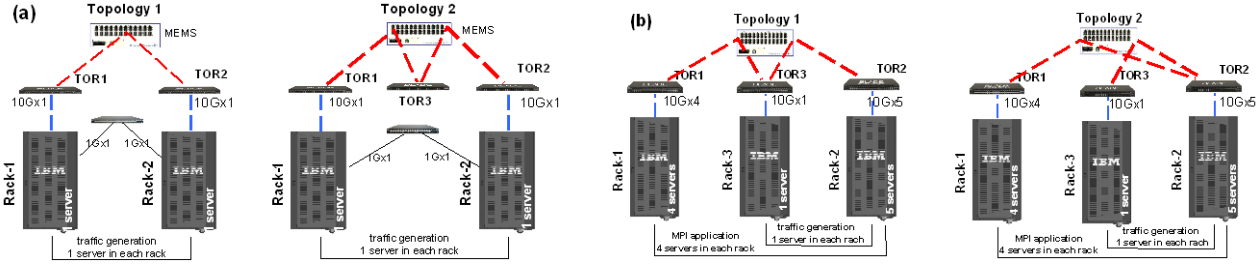

Fig. 2 (a) Topologies used to measure reconfiguration delays (b) Topologies used to measure applications' execution times.

Next we run the FFTW MPI application (3D Fast Fourier Transformation) on racks 1 and 2 (4 servers/rack) that were connected via the 10Gbps optical network (topology 1 of Fig. 2b). The blue line of Fig. 3b shows the measured bw as a function of time. We see that the bw peaks at 3.54 Gbps, while the application finished at 28.7 sec. 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. 3b shows the bw as a function of time (peak: 1.55 Gbps), with the application finishing at 62.2 sec. Finally we did the same experiment using our network controller to reconfigure from topology 1 to 2 of Fig 2b after 15 sec. The green line of Fig. 3b shows the bw 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 sec. Note that the acceleration depends on the reconfiguration trigger that was given at 15 sec in this experiment. We did not discuss here mechanisms to detect congestion and trigger the reconfiguration, but focused on the reconfiguration delay and showed that even though in our implementation this is high - in the order of secs - we can still achieve significant performance improvement. Similar results were obtained for other HPC applications, now shown here due to space limitations.

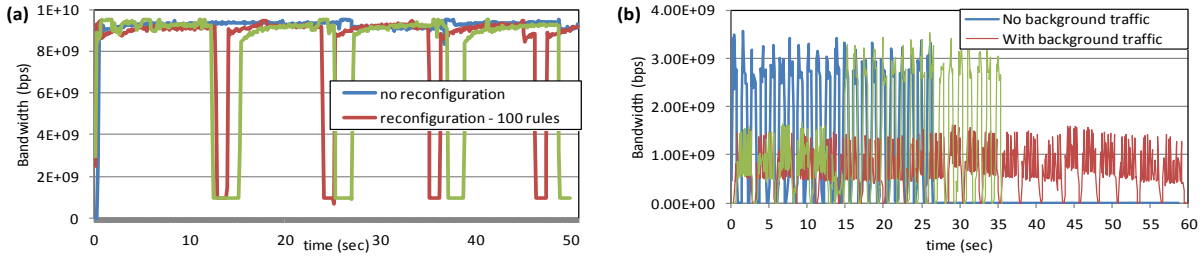

Fig. 3 (a) Topologies used to measure reconfiguration delays (b) Topologies used to measure applications' execution time.

4. Conclusion

We reported on a functional hybrid electrical/optical interconnect and a controller that can dynamically reconfigure it. We measured the reconfiguration delay and observed the modules that comprise it. Even though our controller was not fully optimized we demonstrated a substantial acceleration in execution time of parallel MPI applications.

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

References

[1] K. J. Barker et al.: On the feasibility of optical circuit switching for high performance computing systems. Supercomputing (SC), 2005.
[2] N. Farrington et al.: Helios: a hybrid electrical/optical switch architecture for modular data centers. ACM SIGCOMM, 2010.
[3] G. Wang et al.: c-through: part-time optics in data centers. ACM SIGCOMM, 2010.
[4] K. Chen et al.: OSA: an optical switching architecture for data center networks with unprecedented exibility. NSDI, 2012.
[5] D. Lugones et. al: Accelerating communication-intensive parallel workloads using commodity optical switches and a software-configurable control stack, Europar, 2013.
[6] OpenVswitch: an open virtual switch: http://openvswitch.org/
[7] Project Floodlight: http://www.projectfloodlight.org/floodlight/
[8] K. Katrinis et al.: SDN control for hybrid OCS/electrical datacenter networks: An enabler or just a convenience?, IEEE Photonics Society Summer Topical, 2013.
[9] K. Christodoulopoulos et al.: Tailoring the network to the problem: Topology Configuration in Hybrid EPS/OCS Interconnects, to appear Concurrency and Computation: Practice and Experience.