Impact of Photonic Switch Radix on Realizing Optical Interconnection Networks for Exascale Systems

Sébastien Rumley[1], Madeleine Glick[2] , Raj Dutt[2] and Keren Bergman[1]
1: Department of Electrical Engineering, Columbia University, 500 W. 120th St., New York, NY 10027, USA
2: APIC Corporation, 5800 Uplander Way, Culver City, CA 90230, USA
sr3061@columbia.edu

Abstract: We investigate the realization of large scale, 100,000-node optical interconnection networks with photonic switch fabrics of varying radices. Although such interconnection networks are realizable with 16-radix switches, radices greater than 40 provide a significant advantage.

1. Introduction

Designing Exaflop capable computing systems is on the agenda of an ever increasing list of government agencies, institutions and industries worldwide. Today's machines reach 20 Petaflop/s (as of February 2014) [1] but are already showing limitations in terms of programmability, resilience, power consumption and cost. Incremental evolutions of existing platforms may not be sufficient to achieve the 50x factor required for Exascale [2]. Nevertheless, several "strawman" designs estimating the number of CPU cores, memory sizes, number of nodes, etc. have been published (e.g. [3]). Through increased multi-core parallelism, a single computing chip is expected to deliver approximately 1 Teraflop/s [4]. A simple approximation that assumes approximately 10 sockets (i.e. chips) per compute node translates to an Exascale system with around 100,000 compute nodes. These thousands of nodes must be interconnected with a network supporting injection bandwidths and link rates at least of the order of 1 Terabit/s (keeping the same *injection bandwidth/flops per node* ratio in today's top system). With such high bandwidth, and given that the interconnect will span several meters, the first Exascale machines will leverage optical transmission, as in current leading supercomputers.

However, optics is also envisioned to perform switching operations (transparent switching). This transparency, by allowing the signal to stay in the optical domain, has the potential to reduce the number of per bit electrical operations, and therefore the network power consumption. Several challenges still need to be addressed, for instance, achieving energy proportionality of optical links [5], and the relatively slow switching times. Nevertheless, studies have shown that transparent switching can be exploited provided adequate architectural changes are implemented [6]. Moreover, recent demonstration of sub-nanosecond switching in silicon photonics [7] opens the door to per-packet switch reconfigurations, as for example in [8].

The emergence of optical switching as a potential solution for Exascale computing leads us to the following question: what would a 100,000 node optical network look like? In a similar way to current electrical networks, large scale optical interconnection networks would be composed of a combination of low *radix* switches. We therefore investigate the implication of having to realize such an interconnection topology. We start by considering three classical ways of constructing a large scale interconnection network: fat-tree, torus and flattened butterfly. For each architecture, we analyze how increasing switch radices leads to lower hop counts, lower number of switches, and less links in the network. From these numbers, we derive conclusions about the requirements in terms of radices for Exascale optical interconnects.

2. Interconnection network architectures

High node count interconnects supporting N compute nodes can be constructed by combining multiple low-radix switches in a multi-stage architecture to obtain a virtual high port count switch, forming an *indirect network*. Alternatively, switches and compute nodes can be interleaved, for instance in a mesh, forming a so called *direct network* [9]. In the former case, compute nodes are connected to a portion of the switches only (entry level switches). The remaining switches are employed to ensure connectivity among these entry level switches. This connectivity is achieved by successively interconnecting group of switches or increasing sizes. To keep the bandwidth constant along the resulting hierarchy, a necessary feature to accommodate any traffic pattern, switches and links supporting increasingly higher bandwidths have to be used, leading to a *fat-tree* structure. If such higher bandwidth equipment is not available, parallel lower rate links and switches can be used instead, resulting in a folded Clos. The designation *fat-tree* is however often kept.

In *direct networks*, an equal number of compute nodes are attached to each switch. These switches are then interconnected among themselves following a particular structure. The n-dimensional torus and flattened-butterfly (FB)[10] are the most common representatives of such structures. Both organize S switches as a hyper-cube with around $\sqrt[n]{S}$ switches along each edge. They differ in the wiring. In the torus, nodes sharing a dimension are wired in a ring. In the FB, all nodes sharing a dimension are interconnected.

We consider here three degrees of freedom in the design of both torus and FB architectures: a) the number of dimensions n b) the number of compute nodes attached to each switch M c) the number of parallel links connecting two switches p (note that p can be made dimension dependent. Here we keep it constant across dimensions due to space constraints). For the fat-tree, we adapt the topology at the first level, to perform aggregation: for instance, assuming 32-ports switches, and that each compute node may send traffic to its entry switch at full-link rate, only 16 compute nodes can be connected per switch. On the contrary, if each compute node is expected to send *on average* no more than 10% of the link rate, 29 compute nodes can be accommodated per entry switch (leading to a relative load of 290% *on average*, which can be served by the remaining 3 ports). Apart from this adaptation, there is no freedom in the fat-tree design as long as bandwidth is kept constant across the levels.

3. Traffic model and architecture comparison

To compare the architectures in distinct traffic conditions, we introduce the traffic locality vector V with $v[i]$ representing the part of the traffic injected by each node that is destined to the ith closest other node in the topology. We retain two cases here: uniform traffic ($v[i] = 1/N\text{-}1 \, \forall\, i \neq 0$, $v[0] = 0$) and adversarial traffic ($v[i] = 0 \,\forall\, i \neq N\text{-}1$, $v[N\text{-}1] = 1$).

Considering a number of compute nodes N, a given radix R and a traffic locality V, we sweep all possible combinations of n, p and M for both torus and FB. For each combination, we check if each combination can support the traffic. If it can, we store a triplet *{average hops, number of switches, number of links}*. Once the sweep is done, we keep only the triplets that minimize one of the criteria. We then compute the same triplet for the fat-tree. We count a hop as a traversed switch. Links are assumed bidirectional.

Results of the comparisons are displayed in Fig. 1. Note that no results are available for the FB with a radix of 16 as100,000 clients is beyond the scalability limit (such limits are investigated in [11] for 1-dimensional topologies). Fig. 1a) shows the average hops achieved by the traffic of interest (uniform or adversarial). The FB dominates the other topologies: the number of hops corresponds to the number of dimensions n plus one (entry switch). As n is limited to 4 for radices 48 and 64 and to 3 for radices 96 and 128, the number of hops is thus 5 and 4. In these two cases, resulting hyper-cubes have *[11,11,11,11]* and *[16,16,17]* switches on each edge, respectively. This low number of dimensions is made possible by relatively high values of 7 and 23 for M, i.e. compute nodes connected to each switch. For the fat-tree, even with a radix of 128, three levels of switches are required, translating into 5 hops, and four levels (7 hops) for radixes < 74. Hence, in terms of travelled distance (which is a proxy for latency), the FB shows more flexibility and a radix of 48 already provides a relative compactness.

The number of links and switches (Fig. 1b and 1c) are closely interrelated, showing that in all cases most ports are utilized. The fat-tree topologies generally dominates, except for radices 32, 48 and 64 with uniform traffic, where the FB is more advantageous. As we assume a constant bandwidth along the fat-tree level, the uniformity of the traffic provides no savings (it would be the case using universal fat-trees [12] – to be developed in future work). In contrast, traffic uniformity does have an impact on the FB.

If hardware is the main cost driver, and radices of 92 or larger are available, a fat-tree is the best option.

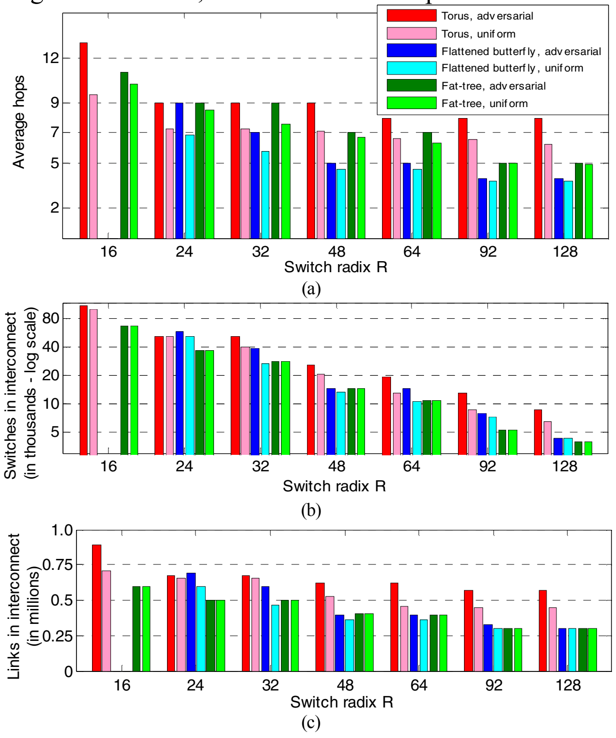

Fig 1. Impact of the radix on the interconnect size, for 100,000 compute nodes.

4. Conclusion

We investigate the influence of the switch radix on interconnection network structure. Though an Exascale topology is realizable with 16-radix switches, a radix of 32 is required to exploit the potential of a FB. With radices of 48 and 64, the FB shows interesting benefits compared to the fat-tree. For larger radices, our results show that the fat-tree is the most hardware efficient.. However, the fat-tree forces traffic through an additional hop, which may be detrimental to application performance, in which case the FB would be favored.

5. References

[1] www.top500.org
[2] P.M. Kogge et al. “Yearly Update: Exascale Projections for 2013”, Sandia Report.
[3] J. Shalf, et, a., LNCS Vol. 6449, 2011.
[4] P.M. Kogge, et al. “ExaScale Computing Study: Technology Challenges in Achieving Exascale Systems “, 2008.
[5] S. Borkar, JLT, 2013.
[6] J. Shalf, et al. SuperComputing’07
[7] K. Nozaki, et al. CLEO 2013.
[8] S. Rumley, et al. ECOC 2013.
[9] A. Agarwal, IEEE Trans. on P. and Distributed Systems, 1991.
[10] J. Kim, et al. ISCA'07.
[11] Q. Li, et al. JOCN, 2013.
[12] C. E. Leiserson, IEEE Transactions on Computers, 1985.