A Cost Comparison of Data Center Network Architectures

Lucian Popa[*†] Sylvia Ratnasamy[‡] Gianluca Iannaccone[‡] Arvind Krishnamurthy[§] Ion Stoica[*]

ABSTRACT

There is a growing body of research exploring new network architectures for the data center. These proposals all seek to improve the scalability and cost-effectiveness of current data center networks, but adopt very different approaches to doing so. For example, some proposals build networks entirely out of switches while others do so using a combination of switches and servers. How do these different network architectures compare? For that matter, by what metrics should we even begin to compare these architectures?

Understanding the tradeoffs between different approaches is important both for operators making deployment decisions and to guide future research. In this paper, we take a first step toward understanding the tradeoffs between different data center network architectures. We use high-level models of different classes of data center networks and compare them on cost using both current and predicted trends in cost and power consumption.

1. INTRODUCTION

The network infrastructure is a first order design concern for data center operators. It represents a significant fraction of the initial capital investment while not contributing directly to future revenues. For this reason, reducing the network infrastructure cost is seen by operators as a key driver for maximizing data center profits. The search for efficient and low-cost data center network fabrics has motivated several research proposals in recent years [14,15,23–25]. These proposals address a similar set of challenges (*e.g.*, reducing cost, improving bisection bandwidth, and increasing fault tolerance) but adopt very different approaches. For example, some architectures use only switches to forward traffic between servers [15, 23] while others require that servers participate in packet forwarding [14, 24, 25].

While there is a clear need for a sound methodology for comparing different network architectures, there has to date been little work in this direction. Such a comparison is challenging for multiple reasons. First, there are many dimensions that characterize performance, *e.g.*, latency, bandwidth, power, cost, failure resilience and so forth. A second challenge is due to the fundamental differences in proposed network designs that make it difficult to identify a level playing field from which to compare different solutions. For example, in traditional networks, one might simply count switch costs but for architectures that route packets through servers one must determine how to apportion the cost of servers that participate in such packet forwarding. A final complication is the various factors that drive costs and the impact future trends in server and network equipment may have on any comparison drawn from the capabilities and costs of current equipment.

In this paper, we make two contributions that together constitute a first attempt at understanding the trade-offs involved in the design of data center network architectures.

First, we propose a methodology for estimating and comparing the costs of several data center network architectures (§3). To a first approximation, our approach is based on setting a target performance level in terms of the key performance metrics of network capacity and latency and then comparing the network costs incurred by different design options when provisioned to sustain this target performance.

Second, using the proposed methodology, we conduct a cost analysis of several representative architectures (§4). In our analysis, we use current market prices and power consumption figures for hardware equipment and explore how technology trends may impact our conclusions. In addition, we analyze relative costs in a manner that is agnostic to absolute price values and in an asymptotic context.

The remainder of this paper is organized as follows. In Section 2 we provide an overview of the design space for data center networks and justify our selection of a small set of representative designs. In Section 3, we describe our methodology for comparing different network designs and then present our results in Section 4. We conclude with a discussion of the implications of our findings and potential directions for future work.

[*]University of California, Berkeley
[†]ICSI, Berkeley
[‡]Intel Labs, Berkeley
[§]University of Washington

ACM CoNEXT 2010, November 30 – December 3 2010, Philadelphia, USA.

2. DATA CENTER NETWORK DESIGNS

Our goal is to compare different data center network architectures. Toward this, we start with a high-level classification of data center network designs based on which we narrow our study to several architecture instances. The many proposals for next-generation data center network architectures [14, 15, 23–25] target a similar goal—namely, supporting high bisection bandwidth between very large numbers of servers in a cost-effective manner—but adopt very different approaches to achieving this. A key dimension along which these proposals differ is the form of hardware equipment used to forward or process network traffic. This leads us to classify designs into three broad categories along this dimension.

In **switch-only** architectures, packet forwarding is implemented exclusively using switches. Traditional data center networks that organize switches in a simple tree topology [19] fall under this category as do recent proposals such as VL2 [23] and the work of El-Fares *et al.* [15] that interconnect switches in more sophisticated topologies such as fat trees. Our second category, which we term **hybrid** architectures, include designs in which packets are forwarded using a combination of switches and servers as exemplified by systems such as BCube [24] and DCell [25]. Finally, we have **server-only** data center architectures which do not rely on switches for packet forwarding. Instead, each server plays a dual role – running regular applications and also relaying traffic between servers. Every server is directly connected to a few other servers to form a data center-wide interconnect topology of server-to-server links. Abu-Libdeh *et al.* [14] propose such a design in which servers are interconnected in a three dimensional torus topology. In some sense, switch-only and server-only designs can be viewed as the two extreme points in the design spectrum of hybrid architectures.

In scoping our analysis, we made two key choices. First, we chose to focus on data centers in which each server has 10Gbps connectivity rather than the current 1Gbps. Our rationale in this was simply that discussions around upgrading server connectivity from 1Gbps to 10Gbps are already underway in industry. In fact, some cloud computing providers have already started to offer 10Gbps connectivity with full bisection bandwidth for high performance computing [3]. Moreover, existing tree-based network architectures appear adequate for servers with 1Gbps connectivity and hence any overhaul of data center networks (at which point comparing design options has greater relevance) is more likely to accompany an upgrade to server connectivity. Because scaling traditional tree-based architectures to servers with 10Gbps connectivity is problematic (for reasons described in [15]), we eliminate this design from consideration.

Our second methodological decision has to do with routing in switch-only architectures. Current Ethernet routing protocols based on constructing spanning trees scale poorly to large data center sizes; the issues involved have been frequently documented and several ongoing efforts propose modified routing algorithms that address these problems [15, 23, 29, 31, 33]. In our study, we assume an idealized routing protocol that forwards packets along the shortest path(s) to the destination, with load-balancing across the set of such paths to make full use of the available network capacity. Implicitly, this assumes that current problems with Ethernet routing will be addressed in next-generation data center networks and that the resulting design will efficiently exploit available network capacity. Further, by using idealized route selection, the performance of a switch-based design becomes largely dependent on the inter-switch topology selected and less on issues such as addressing. Most proposals for new switch-only designs share similar topologies (typically a form of Clos network), differing more on how routing and addressing are implemented. Hence we can narrow our focus to a candidate switch-only architecture – we select the fat-tree based proposal of El-Fares *et al.* – and expect that our results are roughly representative of switch-only proposals based on similar topologies.

With these methodological choices in place, our study compares the following four data center network designs.

(1) FatTree-based switch-only networks: As mentioned above, we choose the fat-tree based proposal of El-Fares *et al.* from the class of switch-only network designs. We refer to this as SW-FatTree. In SW-FatTree, switches are interconnected in a folded-Clos topology; servers form the leaves in this tree and each server is connected to one switch port. We compute routes over this topology as described above. We believe that using VL2 would not change our high-level conclusions, since VL2 also uses a Clos topology.[1]

(2) de Bruijn-based server-only networks: In considering server-only network designs, we started with servers interconnected in a 3D-torus as in CamCube [14]. However we found that this design consistently suffered (in both performance and cost) relative to our other designs and that this was due to the relatively long routing paths in a torus—$O(N^{1/3})$ hops, with N servers.[2]

Since the notion of a server-only network design is broader than the specific choice of topology made in CamCube, we looked for topologies that would overcome this limitation particular to a low-dimensional torus. A natural alternate is a de Bruijn topology because it has an optimal diameter ($\log N$) given a constant degree of the nodes. In a de Bruijn graph of dimension n and base k, each vertex has a representation of length n in base k (*i.e.*, with values 0,...,$k-1$). Thus, the set of vertices is $V = \{(0, ..., 0), (0, ..., 1), \ldots, (k-$

[1] As originally proposed in [23], VL2 uses heterogeneous link capacities and is more difficult to scale to 10Gbps bandwidth per server. For simplicity, we narrow the scope of our comparison and only use homogeneous link capacities in this paper.

[2] In addition to the performance penalty, longer paths result in higher cost because longer path lengths lead to a higher total volume of traffic and hence the network must be provisioned for higher capacity.

$1, ..., k-1)\}$. The edges are between nodes of the form $((a_1, ..., a_n), (b_1, ..., b_n))$ where $a_2 = b_1$, $a_3 = b_2$,..., $a_n = b_{n-1}$, *i.e.,* the endpoint identifiers are shifted by one digit. Thus, nodes have k neighbors. Since de Bruijn is a directed graph while network links are bidirectional, we use a modified version of a de Bruijn graph where links are considered undirected (*e.g.,* see [36]). In this graph, nodes have (at most) $2k$ neighbors, similar to a de Bruijn graph where the identifiers of the neighbors of each node are shifted both left and right.

Like the torus, a de Bruijn graph requires a small and constant number of connections per server but has a much better worst-case path length of $\log_k N$. When requiring full bisection bandwidth, we found that a de Bruijn topology reduced costs by 2-3$\times$ relative to a 3D-torus, while providing equivalent performance even for small network sizes such as 15,000 hosts. As expected, due to the different scaling properties, these differences increase with the network size.

We thus introduce a new "SRV-deBruijn" server-only network design in which servers within a rack are connected with a de Bruijn graph. Between racks we also use de Bruijn graphs. To provide the necessary bisection bandwidth, we create r identical de Bruijn graphs between racks, where r is the number of servers in a rack. More precisely, if we order the servers in each rack from 1 to r, servers labeled 1 from each rack are part of one de Bruijn graph between the racks, servers labeled 2 are part of another identical de Bruijn graph, and so forth. Thus, our architecture consists of $r + C$ de Bruijn graphs, where C is the number of racks. C of these graphs are inside racks (typically smaller), and r of them are between racks.

Note that SRV-deBruijn, similar to CamCube, achieves a low fanout at the *rack* level in the sense that for a base-k inter-rack de Bruijn topology, (servers in) a given rack is connected to (servers in) $2k$ other racks. Intuitively, this simplifies wiring complexity since the cables exiting a rack are "aggregatable" into a small number of inter-rack aggregates. One drawback of the de Bruijn topology relative to the 3D-torus is that some inter-rack links can be longer than in a torus, where most inter-rack links are between immediately adjacent racks. Our results (in §4) suggest that this additional cost is small, and hence we deemed the tradeoff worthwhile.

Finally, in addition to the de Bruijn topology, we also considered the class of expander graphs since these offer high bisection bandwidth, low diameter and low degree topologies. In particular, we have experimented with random graphs (known to be good expanders) and record degree/diameter graphs [4] with equal degree and diameter. We typically found that these offered an additional $\sim$10% cost savings relative to the de Bruijn topologies. However, because routing over such a topology is more complex than over a de Bruijn or torus and because these graphs are more difficult to analyze, we opted for the de Bruijn as our representative server-only network design in this paper.[3]

(3) BCube and (4) de Bruijn-based hybrid networks: From the class of hybrid network architectures, we started with the DCell [25] and BCube [24] proposals. In our tests, DCell consistently achieved similar performance at a higher cost compared to at least one of the other topologies, and hence we do not consider DCell further in this paper.

Closer examination of BCube revealed that, as a point in the design space of hybrid architectures, BCube skews more toward the "switch heavy" end of the spectrum in the sense that switches take on a greater share of the job of packet forwarding than do servers. For example, in the particular instantiation of BCube that we are using ($K = 2$), over 60% of packet forwarding is done by switches. To better cover the spectrum of hybrid network designs, we thus introduce a new hybrid network architecture inspired by the de Bruijn-based server-only network introduced earlier. Starting with the SRV-deBruijn design, we replace the *intra*-rack server-based de Bruijn topology with a single Top-of-Rack (ToR) switch; *i.e.,* every server is connected to the ToR switch within its rack and all traffic between servers in the same rack flows through the ToR switch. Thus a server has no direct links to any other server within its rack; the inter-rack connectivity between servers remains identical to that in SRV-deBruijn. We term this hybrid architecture as HY-deBruijn. In a HY-deBruijn topology, each server has $2k+1$ links and a worst-case path length of $log_k C + 2$, where k is the base of the inter-rack de Bruijn graph and C is the number of racks. For example, in a scenario with $k = 3$ and $C = 243$, HY-deBruijn handles 70% of packet forwarding in servers and only 30% in switches thus offering a useful counterweight to the HY-BCube hybrid design.

Discussion: We thus compare four specific network designs – a fat-tree interconnect of only switches (SW-FatTree), a de Bruijn interconnect of only servers (SRV-deBruijn), the switch-heavy BCube hybrid (HY-BCube) and the server-heavy de Bruijn+ToR hybrid network (HY-deBruijn). It is important to note that we do not claim that these solutions are the best, or even broadly representative, of their class of solutions; rather, we pick these as they represent the current state-of-the-art from each design class and look to compare them as a starting point for future work on refining and understanding design options. In particular, we expect our work to be relevant because all the four topologies that we use have similar asymptotic behaviors, in that the dominant

[3] In both a de Bruijn or torus topology, routing decisions are made greedily based on local state (*e.g.,* the identifiers of a node's immediate neighbors). By contrast, routing over a random graph would require a computation based on knowledge of the entire network, or a precomputation that holds forwarding state entries for each node in the graph. On the other hand, the architectures based on random graphs have the ability to scale more easily compared to for example the architectures based on de Bruijn graphs, which ideally require the number of hosts/racks to be a strict power of their base. (Techniques using virtual nodes could alleviate some of these concerns.) We leave an exploration of this topic to future work.

resource used for forwarding (switch ports, NIC ports, CPU cores) scales as $NlogN$ where N is the number of nodes in the topology (§4.3). Since interconnect topologies have been studied extensively in the literature (and thus few big surprises are expected in designing fundamentally different topologies), we expect our findings to be useful even for future data center architectures using other topologies.

3. METHODOLOGY

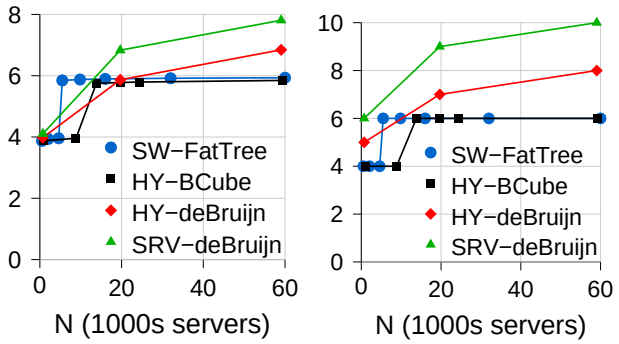
When comparing data center networks, the two broad axes along which we would like to evaluate different architectures are cost and performance. In accounting costs, we consider both the capital expenditure and power consumption due to the network-related infrastructure. These are derived from the cost and power profile of the networks' various component parts, which for the solutions we consider include: switches, cables, server network interface cards (NICs) and server CPU cores used to forward packets by hybrid and server-only architectures. We measure performance in terms of two metrics: the capacity (bisection bandwidth) that the network supports and the latency in routing between servers. An additional performance metric that might be considered is resilience or the network's ability to "route around failure" of individual components. We leave an evaluation of resilience to future work, but we note that the topologies we study (fat-trees, de Bruijn graphs and BCube graph) all have a large number of paths between any two end-points; hence we expect all these topologies to be reasonably resilient at low failure rates.

The challenge in attempting a comparative study stems from the large number of parameters simultaneously at play. The network in each of our solutions is constructed from multiple hardware components, each with its corresponding performance, power and cost characteristics, leading to a range of possible performance-to-cost tradeoffs. For example, in a server-only network, if we double the number of NICs at each server (increasing NIC-related cost), we obtain a topology with higher per-node degree and hence shorter paths (improving performance) but also lower hop-count (incurring lower CPU costs since servers forwards less traffic). In addition, we must consider how different solutions behave for different data center sizes and application workloads.

At a high level, our approach to bringing order to the parameter space is to first arrange (by construction) for the different solutions to achieve roughly similar performance and then compare the total cost of the resultant constructed networks. Effectively, we ask what the cost of the network is to achieve a particular target performance level. Thus we can evaluate and compare the cost of our four solutions for varying network size and target performance.

Our methodology thus consists of the following three steps.

(1) Equalize latency (path lengths): We first arrange for the topologies in the four designs we consider to have roughly similar routing path lengths. We do so by fine-tuning different topology parameters such as the number of levels and



(a) Average (b) Max

Figure 1: Hop count

ports per switch (for SW-FatTree, HY-BCube and HY-deBruijn) and the fanout at every server (for SRV-deBruijn and HY-deBruijn). But we also have to make sure the resulting topologies are feasible in terms of the necessary hardware requirements. In particular, we have to make sure that the switch port count and the number of NICs per server can be practically achieved. In this paper, we require the switch port count to be at most 96 and the maximum number of network ports per server to be on the order of 12 (corresponding to 2 NICs/server, assuming 6-port NICs as available today [5]).

Fig. 1 presents the resultant average and maximum paths we obtain for each of the solutions for increasing data center size (*i.e.*, number of servers). Given the previous constraints and the different topological designs, it is difficult to exactly match the latency between the different architectures, but we believe the results in Fig. 1 are sufficiently close for our purpose of a high-order-bit comparison. The HY-BCube and SW-FatTree maximum paths are practically represented by step functions, increasing every time the fixed "level" parameter increases, *i.e.*, the K parameter of BCube and the number of levels for the folded Clos; we use 2 and 3 as the values for these two parameters in Fig. 1. We use $k = 3$ for the de Bruijn topologies, for both the intra and inter rack graphs. In general, the path length as well as the component requirements scale logarithmically with the number of nodes for all the selected topologies (see 4.3), and thus similar behavior, although in larger jumps, can be expected for all data center sizes.

Hence, after this step we construct topologies of roughly equal path length for each of the solutions we consider, and this effectively normalizes the latency metric.

(2) Equalize capacity: Given topologies with similar path lengths, we next arrange for these topologies to have equal capacity. For this, we first select a target input traffic workload and then *provision* the network with the capacity required to support this input load. Provisioning in this context implies ensuring sufficient switch capacity (for switch-based solutions) or that servers have sufficient NICs and cores to accommodate the desired traffic load. For example, say that in a de Bruijn topology, the traffic demands are such that

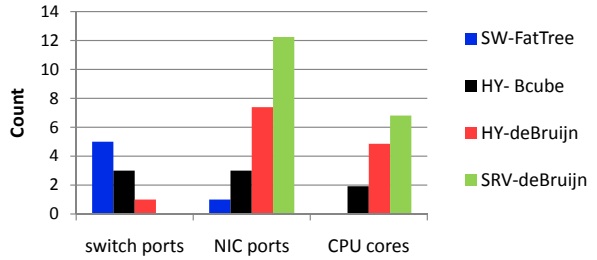


Figure 2: Component requirements per server, 60k server network

17Gbps of traffic should be forwarded between two servers A and B; in this case, we must provision for two 10G links between A and B. Likewise, say that A ends up relaying/forwarding 29Gbps of traffic; then we must ensure A has a number of cores capable of processing 29Gbps. If we say that one core can process a value of α Gbps, then server A must be provisioned with $29/\alpha$ cores. In this paper, we use a value of $\alpha = 10$ Gbps, based on recently published empirical results [26, 34]. To determine the resources (ports, NICs and cores) each switch or server must be provisioned with, we built a high-level simulator that routes the specified input workload over the target data center topology and computes the resultant traffic load at each server and switch.

As before, we must ensure that the provisioned topology is feasible—ensuring that the port count (for switches and servers) and core count (for servers) is possible using current technology. We described our limits on port counts previously. With respect to core counts, current off-the-shelf servers have at most 24-32 cores. Hence we deem any topology that requires more than $\sim$10 cores per server for handling communications as infeasible since we expect data center designers to be not willing to sacrifice a significant fraction of the processing power for networking tasks.

In this paper, we require the data center to have sufficient bisection bandwidth to allow any pair of servers to communicate at full 10Gbps rate. We focus on this capacity requirement as it reflects the driving assumption/requirement of prior work [15, 23–25]—namely that supporting high bisection bandwidth is key to managing complexity and cost in modern data centers. While we provision the networks for full bisection bandwidth, our analysis also considers traffic workloads that do not fully utilize this maximum available capacity; such scenarios incur lower power and CPU costs for network forwarding. Our overall methodology can be applied to different capacity subscription levels, and we leave it to future work to explore these different set of requirements.

Given our capacity requirement, Figure 2 plots the resultant number of component parts required by the appropriately provisioned topology in each of the four network designs for a data center size of 60,000 servers (some components are not required by some topologies).[4] We observe that in all cases the number of components are well within the upper limits on core and port counts mentioned above; this reveals that it is *feasible* to construct equally efficient (as measured by hop count) and moderately-sized networks with full-rate any-to-any bisection bandwidth using any of the solutions we consider. Hence, from the standpoint of performance alone, any of the solutions we consider appear roughly equivalent, under-scoring the importance of understanding costs and alternate dimensions along which the desirability of different solutions might diverge.

Note that in Figure 2, the presented topologies range in the degree of "hybridness" from left (switch-only) to right (server-only), using fewer switch ports and more NICs and cores. However, there are other topologies that make different tradeoffs, for example DCell uses fewer NIC ports than the HY-deBruijn but more cores, since its topology results in longer path lengths.

(3) Count costs: Having normalized the different solutions to achieve equivalent performance, we are left with comparing their costs. We consider the cost due to both the capital expenditure for, and power consumption of, networking-related equipment. For this, we consider the appropriately provisioned topology (from steps 1 and 2 above) and sum up the price and power consumption of each type of component used. The resultant cost requires information on the per-component price and power consumption and, in what follows, we elaborate on our assumptions in this regard. Note that we do not account for other types of cost such as management costs, which are difficult to quantify but might in fact tilt the balance between choosing one network design against another (see §5 for a discussion on this topic).

3.1 Cost Model

Equipment Costs: With regard to capital expenditure, we consider the prices for four components: (i) 10Gbps switches, (ii) 10Gbps server NICs, (iii) server cores and (iv) cables. These prices represent costs to the data center operator. We introduce our specific price numbers in the following sections and here explain the assumptions underlying our cost calculations.

Regarding switches, a key question is how we assume the per-port price scales given the overall switch port count. In this paper, we assume switch price is linear in the number of ports; *i.e.*, the per-port price is constant. Intuitively, one expects some non-linearity as the capacity of the switch fabric is scaled. However, we require our solutions to use switches with fewer than 96 ports and the vendor quotes we received (described in Sec. 4) offer switches up to 48/64 ports at a fixed per-port price. Moreover, from discussions with industry experts, this trend is expected to extend to be-

[4]We use 60,000 as maximum data center size since it is at the higher end of the spectrum for data center size [22], since we could match up all the architecture sizes almost exactly to this value, and because it is difficult for the simulator to scale up to significantly larger sizes.

tween 100-150 ports.[5] Note that the scaling target of our solutions is not accidental; architectures like HY-BCube and SW-FatTree were designed to operate with relatively low-port-count switches for cost-effective scaling. Hence, we think this is a reasonable approximation for the range of port counts necessary to scale to up to 60,000 servers. For example, at this size, the switch fan out required by SW-FatTree is 62 (three level fat tree), for HY-BCube is 39 (K=2) and for the HY-deBruijn is around 81 (we use 729 racks with base 3 de Bruijn graphs between racks).

With respect to server cores, a key question is the extent to which a core should be viewed as dedicated to network forwarding. The question arises because, unlike network interfaces and switches, CPUs can be used to run regular application jobs when not used for forwarding. We thus consider two charging models for cores. In the first *"reserved"* model, we assume that the number of cores required to process the maximum target input traffic (*i.e.,* 10Gbps per server for our assumed input workload) are dedicated for network forwarding and cannot be reclaimed even if the actual network utilization drops below this provisioned level.

In the second *"shared"* model, we assume cores can be used for other tasks when not busy forwarding packets and hence assume core usage in proportion to network utilization levels. Since the average network utilization is typically low [17, 30] this can offer significant savings to server-centric network designs. Note that this cost model assumes a more *elastic* data center workload, where long running tasks can be preempted or migrated under network traffic spikes. Elastic workloads wherein certain jobs do not require strict short-term performance guarantees are common in private data centers/clouds with batch computing jobs, though less common in data centers hosting public web services or public clouds. Other drawbacks of this cost model might arise as performance penalties due to the shared cache and I/O between the forwarding processes and the rest of the jobs.

To account for the network interfaces, we count the number of ports required per server and use NIC list prices in a straightforward manner (§4).

Finally, we must account for cabling costs. In this regard, we are not aware of any publicly documented cost models and hence we consider a model that emerged based on discussions with industry collaborators. Our cable model is "labor-centric" and reflects discussions with operators that suggested the dominant expense in cabling is due to the human cost of manually wiring equipment. Our labor-centric model thus quantifies the intuition that: (a) it is cheaper/ simpler to run cables between equipment within the same rack and (b) given k inter-rack cables, it is simpler/cheaper to run these cables between a smaller number of racks. For example, if rack A has 5 inter-rack cables emanating from it, then a topology in which all 5 cables go to a single other rack B is preferable to a topology in which each cable is run to a distinct rack, say B, C, D, E and F. Thus, specifically, our labor-centric model consists of a higher fixed price for the first cable connecting two racks and a lower incremental cost for each additional cable between a pair of racks (§4).

Power Costs: Similar to above, we compute power consumption costs by adding up the power consumption due to each of our components. For switches and NICs we use a linear power model with a constant idle-time power draw and an active power draw that is directly proportional to utilization as reported by prior empirical studies [32]. For the power consumption of CPU cores, we likewise use a linear model [16, 28]. In estimating the server power consumption, we only account for the incremental power consumed by the CPU and the network interface due to packet forwarding.

4. COST COMPARISON

Using the methodology described in the previous section, we now compare the cost of different data center designs. We explore three broad questions. First, how do the different designs compare given current prices? We explore this in Section 4.1. Next, in Section 4.2, we ask whether/how future technology trends might alter the relative rankings we observe in Section 4.1? Finally, in Section 4.3, we look to compare designs in a manner that is agnostic to absolute price values and in an asymptotic context.

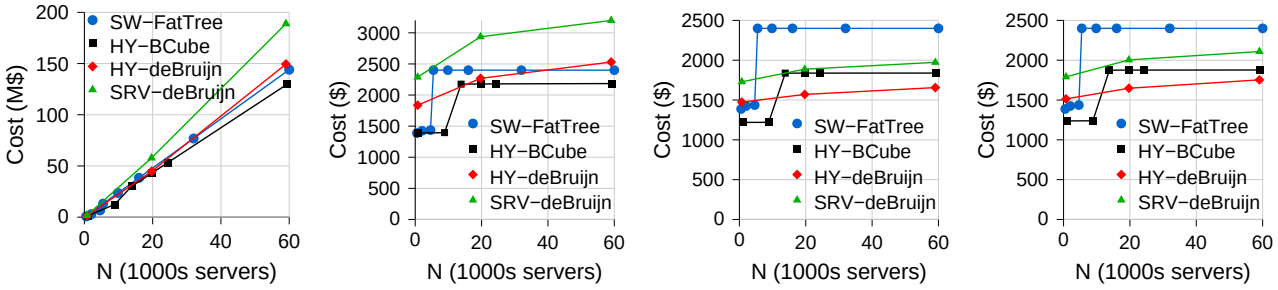
4.1 Current Prices

4.1.1 Equipment Cost

The prices we use are based on quotes from vendors and follow-up discussions with them. A challenge in any cost analysis such as ours is that prices can vary greatly across vendors depending on feature sets and various market factors. We address this as best as we can by considering vendors that are commonly seen in data center deployments [7] or that have explicitly positioned themselves as aggressive on pricing for the data center market [2, 10]. We do not account for factors such as volume discount, the impact of equipment depreciation, and so forth. Such information can be hard to obtain (and often confidential even if obtainable); moreover, we expect these factors would have a similar effect across components—whether switches, servers or NICs—and hence expect this omission to not affect our relative ranking of solutions.

As representative of current prices, we use values of $450 per 10Gbps Ethernet switch port, $150 per 10G port on an Ethernet NIC, and $200 per server core based on quotes we obtained from Arista Networks [2], HotLava [5] and Super-Micro [11] respectively. (These are well-known vendors and generally viewed as price-competitive.) For simplicity, we

[5]One reason for the increase in the number of ports for common switches is the increase in the number of servers in a rack. Most racks have 42 rack units and newer blades occupy less than a rack unit, a common example being to fit 12 blades in 5 rack units; as a consequence, up to 120 servers have been placed in a single rack [18, 35, 38].



(a) Data center, reserved cores (b) Per-server, reserved cores (c) Per-server, shared cores, 10% u (d) Per-server, shared cores, 20% u

Figure 3: Cost comparison - current prices

first consider the capital expenditure due to equipment only and ignore cabling costs.

Fig. 3(a) presents the resultant capital expenditure for the different data center designs using our reserved price model for cores. (Recall that, under this reserved-core model, we add the entire cost of the CPUs needed to forward *peak* network traffic load corresponding to each server generating 10 Gbps.) Fig. 3(b) presents the same results but from the perspective of the cost per server. At a high level, we see that most topologies achieve comparable costs with current prices. HY-BCube achieves the lowest cost, followed closely by SW-FatTree and HY-deBruijn that are 10-15% more expensive, while SRV-deBruijn can be up to 45% more expensive. An additional point worth noting from Figure 3(b) is that the per-server cost grows slowly with increasing data center size.

Fig. 3(c) and Fig. 3(d) present capital expenditure under the shared-core cost model, where CPU cores are assumed to do other useful work when not forwarding packets. Here the accounted cost is proportional to the average network utilization. Fig. 3(c) assumes an average network utilization of 10%, while we use a 20% average utilization in Fig. 3(d). These values are picked based on measurement studies of data center network utilization [17, 30, 37]. We see that the ability to multiplex cores across applications and network-related processing (as implied by the shared-core cost model) makes the server-centric solutions more attractive; HY-deBruijn now achieves the lowest cost although its advantage is relatively modest compared to the other server-centric approaches. For example, at 20% utilization and for 60,000 servers, SW-FatTree is about 37% more expensive than HY-deBruijn, while HY-BCube is about 7% more expensive and SRV-deBruijn is about 20% more expensive.

Fig. 4 breaks down the total capital expenditure for a 60,000 server data center, assuming the reserved-core model. We can extrapolate to the corresponding breakdown under the shared-core model by scaling down all CPU costs in proportion to network utilization; this is shown by whiskers on the bars in Fig. 4 for a 20% average utilization. It is worth noting the different tradeoffs made by the two hybrid designs, with one having a profile similar to that of switch-based architectures (HY-BCube) and the other closer to that of server-only architectures (HY-deBruijn).

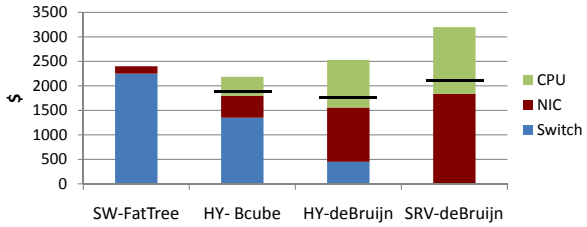


Figure 4: Equipment cost breakdown for a 60k network

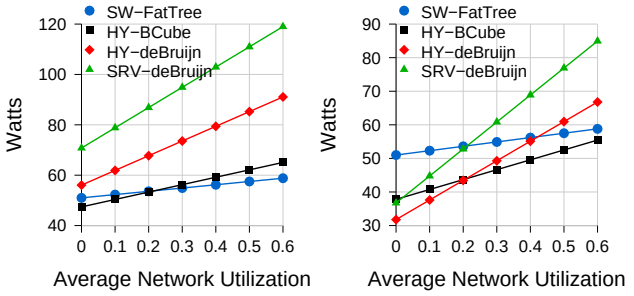
Thus, with our estimates of current prices, hybrid designs achieve the lowest equipment costs, and the particular hybrid design achieving the lowest cost depends on the extent to which cores can be shared between applications and network processing. However, none of the designs achieve a significantly lower cost.

4.1.2 Cabling Cost

To compute cabling costs, we use a price of: (i) $10 for each intra-rack cable, (ii) $50 for each inter-rack cable, and (iii) an additional $300 penalty for the first cable that connects a pair of racks (in keeping with the labor-centric model described in Sec. 3). We select these based on current cable prices, with $50 corresponding to a 70 feet category 6 10GbE cable.

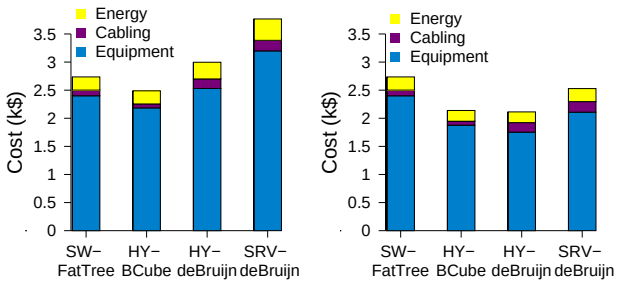
A brief note on the details of the cabling layouts we assume: we make the simplifying assumption that racks contain the same number of servers as the switch port fan-out (when used). For SW-FatTree, we use the approach proposed in [15] and combine the leftover core switches from each "pond" into racks. For HY-BCube we assume the size of one rack to be the size of a BCube_0, and we divide the remaining switches (optimally) into racks of the same size.[6] The de Bruijn-based architectures do not need any special

[6]The level 1 switches from each BCube_1 were placed in a rack of their own, then the first n level 2 switches were placed in a rack, the next n level 2 switches in another rack and so on. (n is the switch fan out.)



(a) Reserved Cores (b) Shared Cores

Figure 5: Per server network power consumption



(a) Reserved Cores (b) Shared Cores (20% util.)

Figure 6: Total network cost normalized by number of servers after 5 years of operation at current prices

optimization, with each rack typically connected to 6 other racks for base 3 de Bruijn networks.

SW-FatTree	HY-BCube	HY-deBruijn	SRV-deBruijn
$101	$70	$170	$188

Table 1: Per Server Cabling Cost Estimates.

Table 1 presents the resultant per-server cabling cost. We present a single value since we found that per-server cost varies little for different data center sizes. We see that (somewhat surprisingly, given its use of low-radix switches) HY-BCube achieves the lowest cost, about 40% lower than SW-FatTree and almost three times less than the cost for SRV-deBruijn. In the absolute however, we see that cabling costs are small compared to equipment costs – typically between 3-8% of the latter (in the reserved core model). Hence, in the bigger picture, the impact of cabling complexity appears unlikely to be a deciding factor across different architectures.

4.1.3 Power Consumption

We next estimate the power consumption of networking related equipment. For switch and NIC power consumption, we use available data sheets for 10GbE equipment. In particular, we use 12 Watts as the maximum power consumption per switch port [1] and 4 Watts for the NIC port maximum power consumption [6]. Since the technical datasheets present only the maximum power consumption (*i.e.*, for full utilization) we use an energy proportionality index of 20%—meaning that the idle power consumption represents 80% of the maximum power—as suggested by published empirical data [27, 32].

To estimate the power due to packet forwarding at a server, we measured the power consumption of a server when some of the CPU cores were running and measured the additional power usage when one or several of the cores were forwarding at full capacity. We used a software router technology [20] and measured a 12 core server, with Intel Xeon (X5660) cores. As a result of this experiment, we use the value of 10 Watts per core at maximum utilization. We assume the per-core power consumption to be proportional with utilization, as suggested by [16, 28]. In the shared core model, we only add the value of the above active power. In the reserved model, we also add a value of 5W for the idle power of a core.

Fig. 5 shows the resultant total power consumption across designs. In the reserved model, the server-heavy designs (SRV-deBruijn and HY-deBruijn) are penalized by CPU idle power leading to higher power consumption than the switch-heavy designs. In the shared cost model however, the server-centric approaches dominate SW-FatTree at low utilizations. This suggests once again that the ability to use CPU cores for processing when not forwarding in the server-centric topologies (*i.e.*, the ability to use the shared cost model) is likely to be an important differentiator across network designs. We also see that the power draw of server-centric architectures scales better with utilization. This result is not surprising given that switches are often considered notoriously power-inefficient, with power draws that do not scale with utilization [27, 32], while servers incorporate relatively sophisticated power management.

4.1.4 Overall Cost and Discussion

We now consider the total cost as the sum of equipment, cabling and power costs. To convert power consumption to costs, we assume a rate of $0.1 per kWh [13]. Fig. 6 presents the resultant costs after 5 years of operation at a 20% average network utilization, for both the reserved and shared core cost models.

We draw/recap three high-level conclusions from these results. The first has to do with the relative importance of equipment costs, cabling charges and power costs. We see that, in all the data center designs, power and cabling are relatively small contributors to the total cost—between 10-16%. One reason for this is that 10Gbps equipment is still relatively expensive since it has not yet reached the deployment volumes needed to dramatically drive down prices (although switches might soon be there, as we discuss in the following section).

Our second conclusion points to the value of research aimed at running clusters at higher utilizations and at using more "elastic" workloads; techniques such as contention-aware

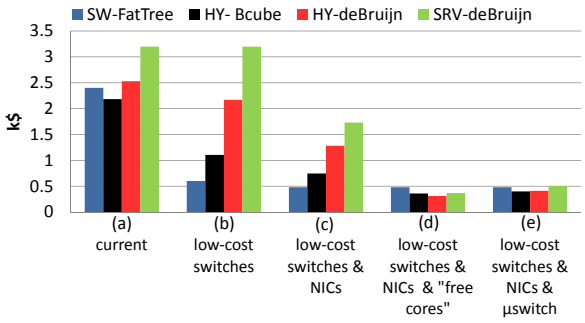


Figure 7: Networking cost per server – future scenarios

scheduling and job placement allow operators to run servers and switches at higher utilization without compromising performance. These techniques would reduce network costs by (a) enabling the shared-core cost model and (b) by reducing the relative cost of the equipment, the dominant factor in the network cost as identified before.

Our third high-level conclusion has to do with the relative cost across different data center designs, which are not huge regardless of the cost model. With a reserved-core model, we conclude that the more switch-centric architectures are superior to server-centric ones, with the server-only architecture being over 40% more expensive than the lowest-cost architecture. With a shared-core model, server-heavier architectures achieve lower costs, with the switch-only architecture being over 40% more expensive than the lowest cost architecture. In both of these scenarios however, a hybrid architecture achieves the lowest cost and the relative differences to the architectures achieving the second and third lowest prices are marginal; discussions with vendors suggest that these differences are within feasible ranges of price cuts that vendors might make in order to gain in market share.

4.2 Future Trends

In this section, we look at how the cost analysis presented before might be affected by future technology and price trends.

Our results in the previous section revealed the dominant effect of equipment costs and that per-server costs vary little with data center size (Fig. 3) and hence, in this section, we focus on the capital expenditure due to equipment for a fixed data center size of 60,000 hosts.

We consider four potential trends in pricing: one based on historical price trends, the remaining based on price reductions in switches, NICs and cores that one might anticipate will follow high volume deployments.

Price trend#1: historical Our cost analysis from the previous section represents a particular snapshot in time. More generally, one might look to historical price trends as indicative of future ones. Historically, CPU prices fall in proportion to Moore's law (a 1.58× price drop per year) while network equipment prices drop by between 1.47×–1.5× per year (the former based on historical sales reports we were able to obtain, the latter on the widely cited Nielsen's law [9]). Our projections (extrapolating our previous results based on the above price trend–we omit graphs) show that, over any reasonable time period, this trend has minimal impact on the relative assessment of switch *vs.* server based designs. This is both because the difference in price decline is small and because server-centric designs bear a high NIC-related cost which we assume scales as network equipment.

Price trend#2: low-cost switches The phenomenal growth in data centers and cloud computing has led to an aggressively competitive market for 10G Ethernet switches and the expectation that switch prices are poised for dramatic reductions [12]. More specifically, discussions with enterprise data center operators reveal that switches from Quanta [10] at a price of approximately $100 per 10G port are now on the market (we were not able to obtain a formal quote) and it is reasonable to expect other switch vendors will follow suit. Fig. 7 plots the impact of such low-cost switches—we assume $90 per 10G switch port (1/5 of our current price estimate) and the reserved-core cost model. Not surprisingly, low-cost switches puts the switch-based SW-FatTree far ahead of the pack—over 5× cheaper than SRV-deBruijn and 2× cheaper than even the best hybrid design.

Price trend #3: low-cost NICs Can hybrid and server-based designs recover from the blow due to low-cost switches? A reasonable expectation is that NIC prices will also drop with growing deployments as servers are upgraded to 10Gbps NICs (which, of course, can only follows upgrades to the switching infrastructure). Discussions with NIC designers reinforce the possibility of such a trend since the manufacturing costs of NICs are low and hence the price is largely of demand. We thus assume a similar 5× reduction in NIC prices (to $30 per 10G port, or $180 for a 6-port NIC which is not much higher than current prices for 1Gbps NICs) and plot the resultant costs in Figure 7(c). As expected, cheap NICs help server-based designs but cannot fully close the gap to SW-FatTree with cheap switches which retain a between 1.5×-3.5× savings over hybrid and server-only designs.

Price trend #4: "for free" cores A common reaction we encountered in sharing the results of our study was that we were wrong/unfair to charge server-only based designs for the cost of cores. That core counts are expected to scale rapidly and that, even today, server operators do not fully utilize the multicore capability of their servers and hence any cores used for packet processing essentially come "for free". There are both justified and debatable aspects to this line of reasoning and even some disagreement amongst the authors regarding the validity of this argument. Hence, for completeness we consider the impact of ignoring core costs altogether, shown in Figure 7(d). We see that in this scenario,

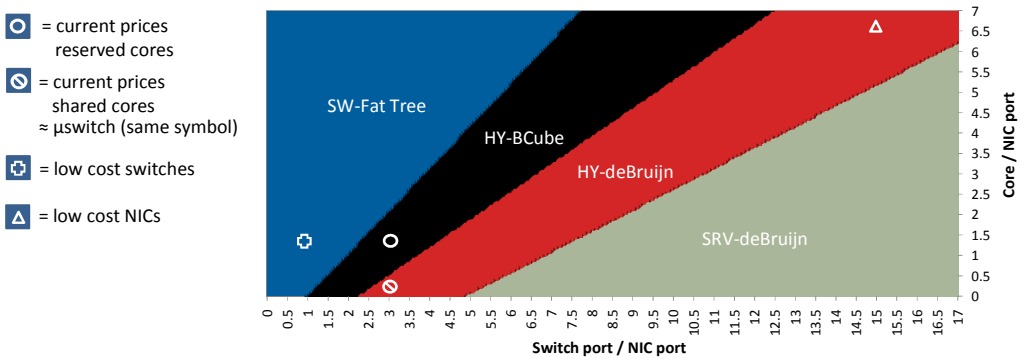


Figure 8: Comparison between switch-based and server-based architectures for different price ratios of cores / NIC ports (Y-axis) and switch ports / NIC ports (X-axis)

server-based approaches do achieve lower costs, achieving cost drops of 25-35% compared to SW-FatTree.

Technology trend: integrated μswitches, hybrid cores? Given the impact of low-cost switches, what avenues might server vendors explore to get back in the game? We speculate on two possibilities. The first is that server vendors incorporate some form of switching capability inside the server– we call this a server μswitch. Such a μswitch might be located within a server's PCIe or I/O hub. Discussions with architects suggest this is achievable at virtually no additional manufacturing cost; in fact, recent integrated "PCIe bridging" technologies are a step in this direction [8]. The effect of a server μswitch is that packets that are not destined for the server and that require little/no sophisticated processing can be "turned around" at the μswitch without being copied to memory or imposing on the CPUs. Such a μswitch is akin to Paxson *et al.*'s proposal for a "shunting" NIC [21]. Such a μswitch would be programmable so that the CPUs (and hence software) have ultimate control over which flows get processed by the μswitch *vs.* the cores. An alternate approach would be for servers to have heterogeneous cores where sophisticated (and expensive) cores are used for application processing while simpler, low-cost cores can be dedicated to packet forwarding. In both these proposals, the effect is to offload some traffic from the CPUs. We quantify this impact by assuming this leads to a β% reduction in core costs; we use $\beta = 90\%$ (this is admittedly a somewhat arbitrary choice but alternate values can be considered in a straightforward manner). Figure 7(e) plots the resulting costs. We see that such technology could close the gap in the cost of switch *vs.* server-based designs.

Discussion Ultra low cost switches appear imminent and offer switch-based architectures a decisive cost advantage. If we take low-cost switches as a given, can server-based designs compete? Our results here suggest server-based designs can be cost competitive only if (a) we have correspondingly low-cost NICs *and* (b) server cores come for (close to) free or new features that integrate low-cost switching capability into servers emerge. Even with these changes however, server-based architectures are mostly equivalent and not highly superior to switch-based solutions. Given that server-based architectures represent a radical shift in deployed infrastructure, we speculate that they are unlikely to see significant adoption *unless* they offer a significant advantage along a design dimension not considered here. We speculate that two directions worth exploring are: (1) the relative manageability of different designs and (2) a network's ability to support richer in-network processing as recently proposed by Symbiotic Routing [14].

4.3 Analysis

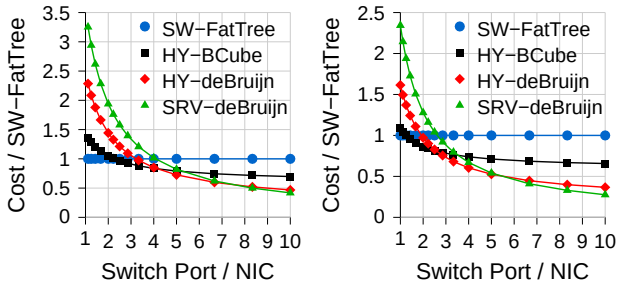
In this section, we investigate cost in a more abstract manner, attempting to decouple our relative ranking of solutions from specific absolute equipment prices. In other words, we try to answer the question: what are the pricing conditions under which a given architecture outperforms others?

4.3.1 Qualitative Analysis

We focus on a data center of 60,000 hosts. In the following section, we analyze the asymptotic behavior and show that similar behaviors are expected for any data center size.

Fig. 8 qualitatively shows the architecture that achieves the lowest equipment cost over the full space of cost metrics (*i.e.*, switch, NIC, and CPU). For readability purposes, we flatten the three-dimensional price space by normalizing the switch and CPU costs with respect to NIC costs. That is, we consider the impact of varying the ratio of the price of a switch port to that of a NIC port (X-axis) and the ratio between the price of a CPU core and that of a NIC port (Y-axis). Figure 8 also plots the particular data points corresponding to current prices with reserved and shared cores, low-cost switches, low-cost NICs, and μswitches, providing some insight behind the results presented in the previous sections.

To understand these results, let us analyze the boundary between SW-FatTree and HY-BCube in Fig. 8. For this network size, SW-FatTree uses a three-level fat-tree topology



(a) Reserved cores (b) Shared cores

Figure 9: Quantitative Data for Fig. 8

SW-FatTree	HY-BCube	HY-deBruijn	SRV-deBruijn
S_P=$O(\log N)$	S_P=$O(\log N)$	S_P=$O(1)$	S_P=0
N_P=$O(1)$	N_P=$O(\log N)$	N_P=$O(\log N)$	N_P=$O(\log N)$
C=0	C=$O(\log N)$	C=$O(\log N)$	C=$O(\log N)$

Table 2: Asymptotic hardware requirements per server. N is the number of hosts, S_P is the number of switch ports, N_P is the number of NIC ports, and C is the number of cores.

(as in [15]) with 5 switch ports per server and one NIC port; on the other hand, HY-BCube uses 3 switch ports, 3 NIC ports, and 2 cores. (This data is also presented in Fig. 2.) Using this breakdown, one can easily derive the equation to delimit SW-FatTree from HY-BCube as $5S_{price} + N_{price} = 3S_{price} + 3N_{price} + 2C_{price}$, where S_{price} is the price for a switch port, C_{price} is the price for a core, and N_{price} is the price of a NIC port. Dividing by N_{price}, this equation results in the boundary separating the two architectures in Fig. 8. Due to space limitations, we do not explain the remaining boundaries, but we consider the asymptotic behavior next.

4.3.2 *Quantitative Analysis*

After seeing which architecture is better in which circumstance, we now measure the amplitude of the results shown previously. Figure 9 presents a quantitative view for two "horizontal" lines in Fig. 8, one that crosses through the marker for current prices with the reserved cores model (Fig. 9(a)) and one through the marker for current prices with the shared cores model (Fig. 9(b)). This chart shows that the server based architectures can be quite expensive compared to switch based ones when the cost of a switch port is very close to that of a NIC port. It also shows that the cost of a switch port has to be just a bit more (1.5-2.5×) than that of a NIC port to make server-based approaches cost efficient when using the shared cores cost model.

4.3.3 *Asymptotic Behavior Discussion*

In the previous sections we analyzed data center sizes up to 60k servers, and for these sizes we used simulations to compute precise numerical estimates. A natural question is how do these results scale asymptotically, especially for very large data center sizes? Table 2 presents the high level asymptotic behavior of the networking equipment requirements for the tested topologies. (We use the same assumptions on required performance.) For simplicity, in Table 2 we ignore all constants including the switch fan out and terms smaller than $\log N$.[7] Note that the performance metrics have the same asymptotic behavior for all topologies, *i.e.*, the maximum path length is $O(\log N)$ hops.

SW-FatTree	HY-BCube	HY-deBruijn	SRV-deBruijn
S_P=L	S_P=$1/2L$	S_P=1	S_P=0
N_P=1	N_P=$1/2L$	N_P=L	N_P=L
C=0	C=$1/2L$	C=L	C=L

Table 3: Asymptotic hardware requirements per server in relation to L, the path length.

Table 2 does not allow us to infer asymptotic behavior in a format similar to Fig. 8 because the latter requires the associated constants. In order to arrive at a similar representation, we proceed as follows. First we express the equipment requirements from Table 2 in terms of the average path length, *i.e.*, replacing N by a function of L (the path length), and with constants obtained from the specific topologies we construct. We then set the path length to be equal for all architectures and show the resulting equipment requirements in Table 3. Note that we ignore the additive constants with respect to L, and the multiplicative constants we use are the asymptotic values, which approximate the exact values corresponding to a specific data center size. Also, these constants are optimistic (in particular for the de Bruijn-based topologies) in that the data center sizes can be matched with the optimum parameters for the topology and switch fan-out.

The expected boundary between SW-FatTree and HY-BCube as computed from Table 3 matches closely the one in Fig. 8. On the other hand, the boundary between HY-BCube and HY-deBruijn converges to the same boundary as between SW-FatTree and HY-BCube, thus "squeezing out" HY-BCube, when considered asymptotically.

In order to identify the boundary between HY-deBruijn and SRV-deBruijn, we need a further level of discrimination. Intuitively, SRV-deBruijn uses $2K_R-1$ more NIC ports than HY-deBruijn, where K_R is the degree of the de Bruijn graph for inside the rack; K_R should be on the order of $\log R$ where R is the number of servers in a rack. This provides us the point where the boundary between HY-deBruijn and SRV-deBruijn intersects with the X-axis (which is $2K_R-1$). Also, SRV-deBruijn uses L_R more cores for forwarding traffic, where L_R is the average path within a rack, while HY-deBruijn uses one switch port. This provides the slope of the boundary, which is $\arctan(1/L_R)$, where L_R is also on the order of $\log R$.

Hence, we expect a qualitatively similar behavior as presented in Fig. 8 for all data center sizes, but with particular boundaries determined by the exact parameters used for each of the topologies to achieve the desired performance goals for a given data center size.

[7] In Table 2, a more exact approximation when selecting the best parameters for the de Bruijn topologies is $N_P = \log N/W(\log N)$, where W is the Lambert W function, and $C = \log N/\log\log N$.

5. CONCLUSIONS AND FUTURE WORK

We presented a cost comparison of various data center network architectures spanning from switch-only to server-only designs. Based on our results, hybrid designs achieve lower costs with current prices, but the margins are not overwhelming. The results also depend on the assumptions regarding the application workload: an elastic workload that is able to reuse CPU cores for traditional computing when not forwarding packets would favor server-heavy designs, while an inelastic workload would favor switch-heavy designs.

In the near future, however, given the various announcements of very low-cost switches, switch based designs are likely to obtain a cost advantage over server-only or hybrid designs. To be competitive, server-based architectures would need to ensure that CPU cores can be dynamically shared by both computing and networking tasks. An alternative would be to include specific on-die features or heterogeneous cores that perform high speed packet switching, effectively bypassing CPU cores.

However, we believe that there are several other technical and non-technical factors that need to be taken into account, and these may favor or penalize some of the designs. For example, servers often exhibit lower MBTF compared to switches, which may reduce the desirability of hybrid and server-based designs. Also, a server-based architecture would represent such a radical shift from traditional network designs that it may encounter resistance as an "untested" option. To overcome these concerns, server-based architectures should exhibit a significant advantage on dimensions other than cost.

We believe one of the possible dimensions could be manageability. Indeed, a potential advantage of server-only architectures is that the data center could be managed by a single engineering team. One of the well known inefficiencies in current data center operations is that multiple engineering teams are required, *e.g.*, one team for networking management, one for server/application management, *etc*. Maintaining multiple engineering teams often results in reduced levels of productivity because of the need to cross team boundaries in order to troubleshoot common problems. We consider examining the management costs of the various architectures an interesting direction for future research. Another interesting research direction opened by server-centric network designs is the ability to implement more powerful network services such as in-network aggregation, caching, TCP optimization or routing enhancements, as recently investigated by [14].

6. REFERENCES

[1] Arista 7148sx. http://www.aristanetworks.com/media/system/pdf/7100_Datasheet.pdf.
[2] Arista networks. http://www.aristanetworks.com.
[3] AWS - High Performance Computing. http://aws.amazon.com/hpc-applications/.
[4] The degree/diameter problem. http://www-mat.upc.es/grup_de_grafs/table_g.html.
[5] Hotlava systems. http://www.hotlavasystems.com.
[6] Hotlava tambora 120g6. http://www.hotlavasystems.com/pdfs/HLS_TamboraDS.pdf.
[7] Intel corporation. http://www.intel.com.
[8] Intel shares vision for the future - "sandy bridge" project. http://www.intel.com/pressroom/archive/releases/20100412comp.htm.
[9] J. Nielsen. Nielsen's Law of Internet Bandwidth, http://www.useit.com/alertbox/980405.html, 1998.
[10] Quanta. http://www.quantatw.com.
[11] Supermicro. http://www.supermicro.com.
[12] Cisco's Switch Market Grows but Competition Increases, 2010. http://seekingalpha.com/article/201675-cisco-s-switch-market-grows-but-competition-increases.
[13] Us energy information administration, 2010. http://www.eia.doe.gov/cneaf/electricity/epm/table5_6_a.html.
[14] H. Abu-Libdeh, P. Costa, A. Rowstron, G. O'Shea, and A. Donnelly. Symbiotic Routing in Future Data Centers. In *ACM SIGCOMM*, 2010.
[15] M. Al-Fares, A. Loukissas, and A. Vahdat. A scalable, commodity data center network architecture. In *SIGCOMM*. ACM, 2008.
[16] L. A. Barroso and U. Hölzle. The Case for Energy-Proportional Computing. *Computer*, 2007.
[17] T. Benson, A. Anand, A. Akella, and M. Zhang. Understanding Data Center Traffic Characteristics. In *WREN*, pages 65–72, New York, NY, USA, 2009. ACM.
[18] Carrie Higbie. Are You Dense?, 2009. http://www.siemon.com/us/white_papers/05-09-26_dense.asp.
[19] Cisco. Data center infrastructure 2.5 design guide. 2010.
[20] M. Dobrescu, N. Egi, K. Argyraki, B.-g. Chun, K. Fall, G. Iannaccone, A. Knies, M. Manesh, and S. Ratnasamy. RouteBricks: Exploiting Parallelism to Scale Software Routers. In *ACM SOSP*, 2009.
[21] J. M. González, V. Paxson, and N. Weaver. Shunting: a hardware/software architecture for flexible, high-performance network intrusion prevention. In *ACM Conference on Computer and Communications Security*. ACM, 2007.
[22] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel. The Cost of a Cloud: Research Problems in Data Center Networks. *SIGCOMM Comput. Commun. Rev.*, 2009.
[23] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta. VL2: A Scalable and Flexible Data Center Network. *ACM SIGCOMM*, August 17 - 21 2009.
[24] 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. *ACM SIGCOMM*, 2009.
[25] 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 *SIGCOMM*. ACM, 2008.
[26] S. Han, K. Jang, K. Park, and S. Moon. PacketShader: a GPU-Accelerated Software Router. In *ACM SIGCOMM*, 2010.
[27] R. Hays. Active/Idle Toggling with Low-Power Idle, http://www.ieee802.org/3/az/public/jan08/hays_01_0108 .pdf, 2008.
[28] U. Hoelzle and L. A. Barroso. *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*. Morgan and Claypool Publishers, 2009.
[29] J. Touch and R. Perlman. RFC5556 - Transparent Interconnection of Lots of Links (TRILL):Problem and Applicability Statement, May 2009. http://tools.ietf.org/html/rfc5556.
[30] S. Kandula, J. Padhye, and P. Bahl. Flyways to de-congest data center networks. In *HotNets*, 2009.
[31] C. Kim, M. Caesar, and J. Rexford. Floodless in seattle: a scalable ethernet architecture for large enterprises. In *SIGCOMM*. ACM, 2008.
[32] P. Mahadevan, P. Sharma, S. Banerjee, and P. Ranganathan. A power benchmarking framework for network devices. In L. Fratta, H. Schulzrinne, Y. Takahashi, and O. Spaniol, editors, *Networking*, volume 5550 of *Lecture Notes in Computer Science*. Springer, 2009.
[33] R. N. Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat. Portland: a scalable fault-tolerant layer 2 data center network fabric. In *SIGCOMM*, New York, NY, USA, 2009. ACM.
[34] L. Popa, N. Egi, S. Ratnasamy, and I. Stoica. Building Extensible Networks with Rule-Based Forwarding. In *USENIX OSDI*, 2010.
[35] SGI. http://www.sgi.com/.
[36] M. Sridar. The Undirected de Bruijn Graph: Fault Tolerance and Routing Algorithms. *IEEE Trans. on Circuits and Systems-1:FundamentrJ Theory and Applications*, 1992.
[37] Srikanth K and Sudipta Sengupta and Albert Greenberg and Parveen Patel and Ronnie Chaiken. The Nature of Datacenter Traffic: Measurements & Analysis. In *IMC: Internet Measurement Conference*. ACM, November 2009.
[38] Thinkmate Blade Servers. http://www.thinkmate.com/Computer_Systems/Blade_Servers.