Evaluating the Impact of Data Center Network Architectures on Application Performance in Virtualized Environments

Yueping Zhang
NEC Labs America, Inc.
Princeton, NJ 08540, USA
Email: yueping@nec-labs.com

Ao-Jan Su
Northwestern University
Evanston, IL 60208, USA
Email: ajsu@u.northwestern.edu

Guofei Jiang
NEC Labs America, Inc.
Princeton, NJ 08540, USA
Email: gfj@nec-labs.com

***Abstract*—In recent years, data center network (DCN) architectures (*e.g.,* DCell [5], FiConn [6], BCube [4], FatTree [1], and VL2 [2]) received a surge of interest from both the industry and academia. However, none of existing efforts provide an in-depth understanding of the impact of these architectures on application performance in practical multi-tier systems under realistic workload. Moreover, it is also unclear how these architectures are affected in virtualized environments. In this paper, we fill this void by conducting an experimental evaluation of FiConn and FatTree, each respectively as a representative of *hierarchical* and *flat* architectures, in a three-tier transaction system using virtual machine (VM) based implementation. We observe several fundamental characteristics that are embedded in both classes of network topologies and cast a new light on the implication of virtualization in DCN architectures. Issues observed in this paper are generic and should be properly addressed by any DCN architectures before being considered for actual deployment, especially in mission-critical real-time transaction systems.**

I. Introduction

Driven by the recent proliferation of Cloud services and trend of consolidating enterprise IT systems, data centers are experiencing a rapid growth in both scale and complexity. At the same time, it is widely recognized that traditional tree-like structures of data center networks (DCN) encounter a variety of challenges, such as limited server to server connectivity, vulnerability to single point of failure, lack of agility, insufficient scalability, and resource fragmentation [2]. To address these problems, in the recent two years several network architectures [1], [2], [4], [5], [6], [7] have been proposed for large-scale data centers and gained a significant amount of attention from both industry practitioners and the research community.

However, all existing work assumes general-purpose underlying systems and does not explicitly consider performance of complicated (e.g., multi-tier) applications or interactions between different application components. In contrast, data centers are usually shared systems hosting a set of applications and the internal traffic is highly correlated due to the embedded application logic. Moreover, there does not exist an experimental comparison of these DCN architectures for a better understanding of their characteristics in a common physical setting. Finally, an in-depth study of the impact of server virtualization on DCN architectures is also missing in the current picture.

In this paper, we seek to fill the void by conducting an experimental evaluation of two DCN architectures, FiConn and FatTree (which are respectively a representative of the hierarchical and flat architectures), in a three-tier transaction system with a cluster-based virtualized implementation. Towards this end, we first implement FiConn and FatTree in a fully virtualized testbed and then justify correctness of our implementation by comparing the experiment results to those obtained from a non-virtualized testbed. We then deploy RUBiS [9], a three-tier eBay-like on-line auction system, on both FiConn and FatTree and examine application performance in different scenarios. We focus on two test cases, *service fragmentation* and *failure resilience*, from which we observe several fundamental properties that are embedded in both architecture classes and shed a new light on the impact of server virtualization on DCN's and application performance. We believe that issues observed in this paper are generic and should be properly addressed by any DCN architectures before being considered for actual deployment, especially in mission-critical real-time transaction systems.

Main contributions of this paper can be summarized into the following three aspects. First, we perform an experimental comparison of newly proposed DCN architectures in a common system setting. Second, this paper employs a fully virtualized implementation and explicitly examines the impact of server virtualization on DCN architectures. Third, all experiments are performed in a cluster-based three-tier system where application performance (e.g., request throughput and response latency) is the major measurement metric. At the time of this writing, we believe that none of the above three points have been properly studied in existing literature and this paper is the first to address all of them simultaneously. Even though our experiments focus on FiConn and FatTree, most of our results can be generalized and applied as guidelines for designing any DCN architectures purported to be deployed in modern virtualized data centers hosting mission-critical real-time transaction systems. However, we understand that this work is by no means a comprehensive experimental study of

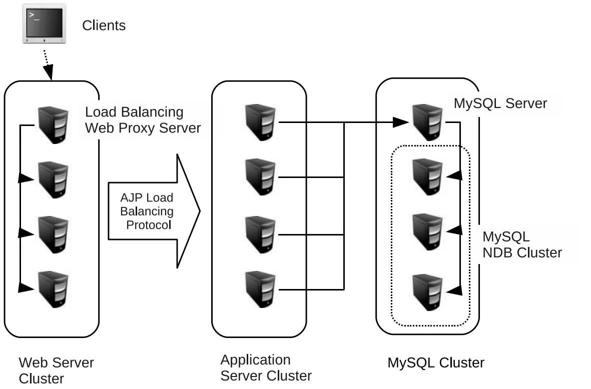


Fig. 1. Three-tier Web service.

all existing DCN architectures. Instead, we encourage readers to consider it as an initial step towards a deeper understanding of these architectures in a more realistic environment.

II. Related Work

In this section, we briefly review the conventional and newly proposed data center network architectures. Based on how networking functionalities are realized, we can classify existing DCN architectures into two categories, *server-centric* and *network-centric*.

In a server-centric design, servers are not only computing units but also routing nodes that actively participate in packet forwarding and load balancing. To the best of our knowledge, the first server-centric DCN architecture is DCell [5], which is designed to improve DCN's robustness to link/server failures and scales to a large number of servers. In DCell, each server is equipped with multiple Ethernet ports for packet forwarding. Guo *et al.* further propose BCube [4], which uses several layers of mini-switches to connect different levels of BCube nodes. Using BCube-based containerized data centers, Wu *et al.* [10] develop an inter-container architecture called MDCube. Another DCN architecture, called FiConn[6], shares the same design principle as DCell and BCube, but uses only two ports for each server.

In a network-centric design, network traffic routing and forwarding is conducted purely in switches/routers. One representative of this class is the FatTree architecture introduced by Al-Fares *et al.* [1]. FatTree applies the Clos topologies originally designed for telephone networks and organizes commodity switches into a k-ary fat-tree. Bearing the same design principle as FatTree, VL2 [2] (or its predecessor Monsoon [3]) is another example of network-centric architectures. In VL2, interconnection topology of switches follows a folded Clos network [2] where intermediate and aggregation switches are respectively organized into either side of a bipartite graph.

III. System Architecture

A. Virtualized Three-Tier Transaction System

In the paper, we conduct experiments in a three-tier transaction system running an eBay-like auction service RUBiS [9] as illustrated in Figure 1. The system is composed of a workload generator and a three-tier processing system. We set up RUBiS workload generator on a separate physical machine to avoid performance interference with the back-end servers. The generated workload is represented by a number of concurrent user sessions with a series of interactions following a predefined Markov transition matrix. This three-tier architecture is a stereotype for many Web services that consist of a Web server layer, an application server layer, and a database layer. We implement each of the three tiers using a cluster of four virtual machines with load balancing mechanisms.

We employ a fully virtualized implementation of DCN architectures. Specifically, for each physical machine, we install a CentOS 5.3 64-bit operating system and Xen 3.1 hypervisor [11]. All physical machines are equipped with Intel Core 2 Duo 2.33 GHz CPU and 4 GB RAM. In addition, we install on each physical machine an Intel quad-port Gigabit network interface card to provide abundant link capacities and isolated network channels for the VM's. System monitoring is accomplished by by combining standard utilities such as `tcpstat`, `vmstat`, and `TCPdump`. For clients, we utilize tools provided by RUBiS to monitor various application QoS metrics, such as response time, throughput, and loss rate.

B. DCN Setup

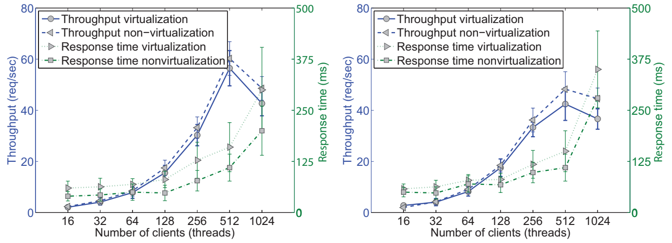
We implement two DCN architectures, FiConn and FatTree, which are respectively a representative of the hierarchical and flat DCN topologies. We implement a level-1 FiConn using three physical machines (PM), each representing a level-0 FiConn. Each PM hosts four virtual machines and utilizes three Gigabit Ethernet ports, one of which serves as the network bridge for inter-connecting the four VM's and the other two serve as router ports. The three PM's are directly connected with each other using CAT6 cross-over cables. We implement FiConn's routing algorithm using statically configured IP routing table in each VM.

We implement a FatTree topology using 12 servers grouped into three Pod's. Each Pod is contained in a single physical server hosting four virtual machines. The Pod switches and the corresponding two-level routing algorithm are implemented using the Click modular router [8] with user-level configurations. The physical machines are equipped with four Gigabit Ethernet ports, each of which is assigned to one of the four up-facing links connecting the Pod and Core switches. The four Core switches are implemented as four separate VLANs on a single 24-port Gigabit switch.

IV. Experimental Evaluation

A. System Validation

Before conducting further evaluation, we first need to ensure that our virtualized implementation does not introduce undesirable artifacts and is able to faithfully reveal characteristics of the original DCN architectures. Towards this end, we implement FiConn and FatTree in a separate non-virtualized testbed. Then, we conduct on both the virtualized and physical testbeds the same set of experiments. Experiment



(a) FiConn (b) FatTree

Fig. 2. Implementation validation.

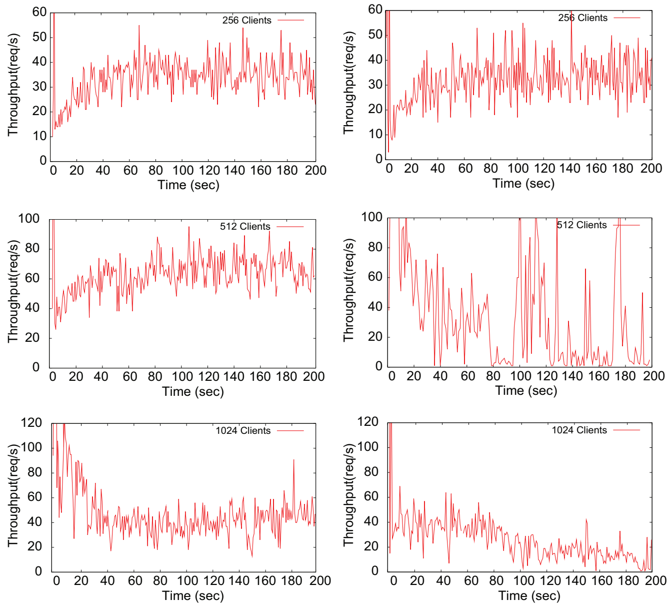
results are shown in Fig. 2. As seen from the figures, FiConn and FatTree produce consistent behavior in the virtualized and non-virtualized testbeds, thereby demonstrating that our virtualized implementation does not introduce fundamental artifacts or performance overhead and therefore is able to faithfully capture properties of DCN architectures with non-virtualized implementation. As virtualization becomes the driving technology of today's data centers, we focus in this paper on evaluating DCN architectures in fully virtualized environments. Thus, in the following two subsections, we examine FiConn and FatTree in more testing scenarios using only the virtualized implementation.

B. FiConn Experiments

We start with FiConn and evaluate it in two testing cases, *service fragmentation* and *failure resilience*.

1) Service Fragmentation: As mentioned in Section III, a level-1 FiConn includes three level-0 FiConn's, each consisting of four servers. At the same time, from the perspective of application setup, we organize the 12 servers into three tiers (i.e., Web, application, and database), each of which also has four servers. Clearly, performance of the system is optimized if the three tiers are exactly mapped to the three level-0 FiConn's, i.e., placing the three clusters of Web servers, application servers, and database servers in $FiConn_0[0]$, $FiConn_0[1]$, and $FiConn_0[2]$, respectively. This is because this way intra-tier network traffic (which is oftentimes data-intensive, such as in the database layer) is delivered by efficient memory swapping between VM's residing on the same physical machine. Experiments performed in the previous subsection (see Fig. 2) follow this VM placement scheme, which we call *ideal placement*[1] for ease of reference.

In practice, however, it is very difficult to guarantee that servers with intensive communication are always placed close to each other. This is especially true in Cloud environments where applications arrive/depart dynamically and resources are allocated/recycled on-demand. In such environments, servers with intensive communication or belonging to the same service tier are very likely to be scattered into different network segments. We call this situation *service fragmentation* and are interested in understanding its impact on different DCN architectures. To achieve this goal, we mix together servers



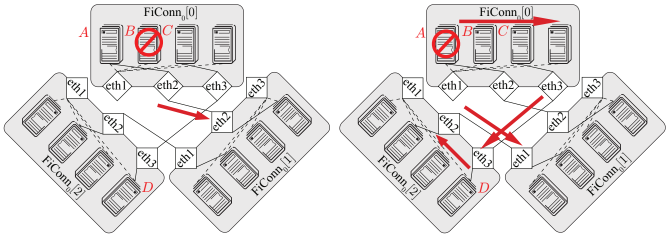
(a) Ideal placement (b) Service fragmentation

Fig. 3. FiConn service fragmentation test.

of different service tiers and rerun experiments conducted in Fig. 2(a). We plot results for 256, 512, and 1024 clients in Fig. 3(b). Comparing these results with those obtained from ideal placement illustrated in Fig. 3(a), it is clear that under heavy workloads (i.e., 512 and 1024 clients), throughput in FiConn exhibits severe fluctuations and low average value under service fragmentation than under the ideal placement. The underlying cause is twofold. First, when VM's belonging to the same service layer are placed onto different physical machine, a significant amount of communication traffic is converted from low-overhead intra-PM memory copying to high-overhead inter-PM network I/O, which slows down the system's performance. Second, placing multiple resource-consuming VM's onto the same PM may lead to contention problems. Particularly, in our setting, two database servers and one Web server are placed on a single physical server. Both types of servers are very CPU-intensive and are subject to competition of CPU under large numbers of concurrent clients. Specifically, under 512 clients, both the Web and database servers experience CPU saturation, which slows down processing of Web requests and database queries and translates into throughput fluctuation depicted in Fig. 3(b).

2) Failure Resilience: A common feature of both FiConn and FatTree is that there exist multiple routing paths between every pair of nodes. As a result, both architectures provide certain resilience to link/node failures. We consider two failure patterns as demonstrated in Fig. 4. Specifically, we construct the first failure scenario (which we call Case I) by bringing down a non-routing node B (which is a Web server in our case) in $FiConn_0[0]$. As a consequence of this action, Web requests will be evenly distributed to the other three Web servers; but traffic between $FiConn_0[0]$ and $FiConn_0[1]$ will still go

[1] We keep this naming for ease of reference and note that it may not be the *ideal placement* for other application settings or network architectures.



(a) Case I

(b) Case II

Fig. 4. Failure patterns of FiConn.

through VM A. In the second failure scenario (which we call Case II), we bring down the routing node A. As a consequence, traffic traveling from $FiConn_0[0]$ to $FiConn_0[1]$ is no longer able to directly go through interface `eth2` due to failure of its corresponding routing node. Instead, it takes a detour by first going to VM C, then to VM D in $FiConn_0[2]$ through `eth3`, then to the other routing node, and finally to $FiConn_0[1]$.

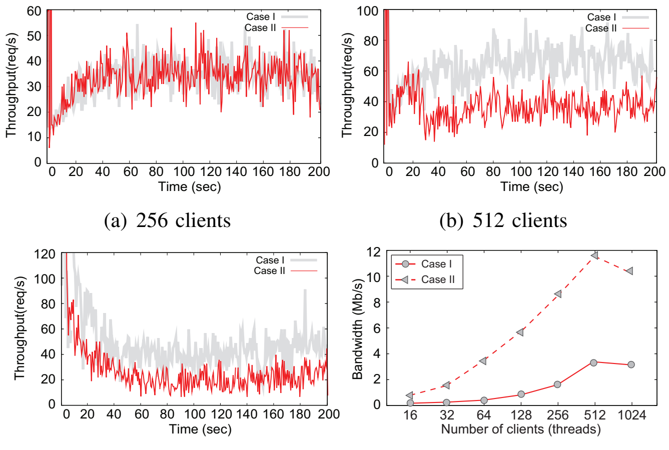
The experiment results in both testing cases are given in Fig. 5. It can be easily seen from the figures that throughput of FiConn under 512 and 1024 clients is much lower in Case II than in Case I. This is because, in Case II, failure of node A leads to traffic redirection and significantly increases the load on the other routing node C. Under larger numbers of concurrent clients, the increased traffic load saturates the routing node and slows down performance of the entire system. The effect of this traffic redirection can be seen from Fig. 5(d), where network traffic encountered by node C in Case II is almost 4 times of that in Case I.

C. FatTree Experiments

We next rerun the above experiments in FatTree.

1) Service Fragmentation: Similar to FiConn, the ideal placement of FatTree is defined as a setting where servers belonging to the same service tier (i.e., Web, application, and database) are placed in the same Pod. Accordingly, service fragmentation occurs when these servers are blended into different Pod's. Fig. 6 illustrates FatTree's performance in the ideal placement and service fragmentation. Two observations can be obtained from the plots. First, comparing Fig. 6(a) and Fig. 6(b), it is evident that these two server placement schemes does not impose a significant impact on application performance in the FatTree architecture. This is in contrast to FiConn as shown in Fig. 3 (or grey curves in Fig. 6), which exhibits a noticeable performance degradation due to resource contention between VM's under service fragmentation. This difference is because, compared to FiConn, FatTree servers are relatively lightly loaded due to isolation of computing and routing and are less prone to resource contention.

Second, in the ideal placement, FatTree yields lower throughput under 512 and 1024 clients than FiConn. This can be partly attributed to the overhead of Click router's user-level emulation of FatTree's two-level routing algorithm. However, a more fundamental reason lies in the design principle of FatTree. In FatTree, all network traffic is first aggregated at and processed by Pod switches, which differentiate intra- and inter-Pod traffic by maintaining a list of terminating prefixes for subnets contained in the same Pod. If a packet belongs to intra-Pod traffic, it is directly sent to the destination by the Pod switches; otherwise, it is forwarded to Core switches for further processing. This scheme works well in physical environments, but may have challenges in virtualized systems. For instance, if a packet is destined at another VM residing on the same PM, it has to travel up to the Pod switches and then come back, even though a much more efficient path (i.e., memory swapping within a PM) exists. Thus, an extra amount of traffic is introduced into the network, resulting in increased network load, reduced goodput, and prolonged response latency. In comparison, in FiConn, VM's collocated in the same PM are connected via a network bridge and are able to directly communicate with each other.



(a) 256 clients

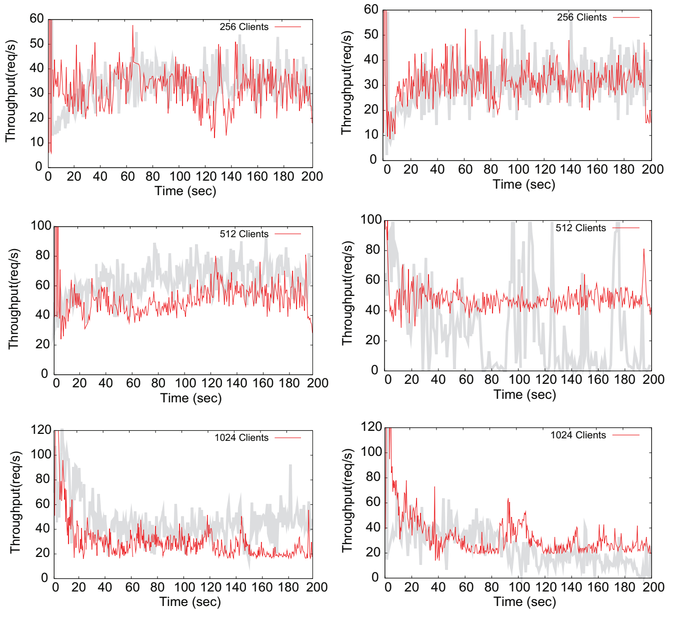
(b) 512 clients

(c) 1024 clients

(d) Network traffic of node C

Fig. 5. Failure resilience of FiConn.

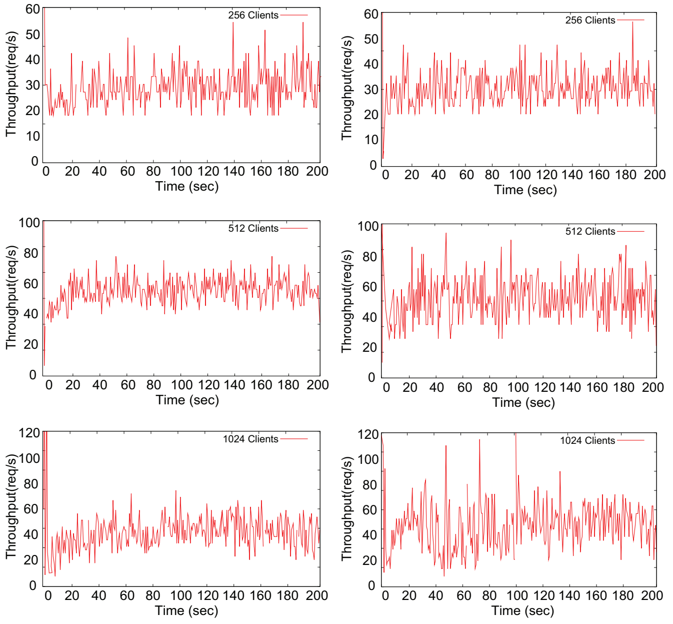
2) Failure Resilience: We next proceed to examine FatTree in two failure scenarios. Different from FiConn, FatTree does not have the concept of *routing node* and routes network traffic purely through switches. Thus, we cannot reproduce in FatTree the same two failure patterns as shown in Fig. 4. Instead, we create two scenarios each with failure of a single node. In the first case, we bring down one of the four Web servers. Then, all Web requests will be evenly distributed to the other three Web servers. Since these Web servers are all lightly used, we expect application performance in this case should not significantly deviate from that illustrated in Fig. 6(a). Experiment results shown in Fig. 7(a) justifies our speculation. In the second case, we bring down one database server, which is more data-intensive than a Web server and therefore more likely to introduce performance bottlenecks. As demonstrated in Fig. 7(b), application throughput in this case indeed exhibits more pronounced fluctuations. These fluctuations are a result of increased workload on the other three database servers and lookup failures of queries destined at records located in the failed database server. Despite this oscillatory behavior, the average throughput is maintained the same as that in Fig. 6(a) and 7(a), demonstrating FatTree's robustness to link/node



(a) Ideal placement

(b) Service fragmentation

Fig. 6. Service fragmentation test of FatTree (grey curves are results of FiConn illustrated in Fig. 3).



(a) Web server failure

(b) Database server failure

Fig. 7. Failure resilience of FatTree.

faults. Similar observations can be obtained under failures of other nodes and are omitted for brevity.

V. Discussion and Future Work

Even though it is still unclear which class of DCN architectures, server-centric or network-centric, will prevail, there are certain fundamental issues that both classes should properly address. On one hand, the concept of *locality* is naturally embedded in server-centric architectures in that servers belonging to the same-level component (e.g., $FiConn_0[0]$ or $DCell_0[0]$) are physically close to each other and are able to communicate efficiently. This property can be utilized by VM placement schemes to avoid traffic propagation to higher-level networks. On the other hand, due to the hierarchical layout of servers, machines (e.g., routing nodes in FiConn) aggregating traffic between components belonging to the same or different hierarchies are more prone to overloading and performance bottleneck. This problem is mitigated in network-centric architectures, where all servers are equally treated and communication between any pair of servers can almost achieve full link speed. However, these architectures may not be able to fully take advantage of locality (e.g., in FatTree) due to their flat layouts. Both problems become more profound when virtualization is introduced, which not only blurs isolation between physical resources, but also expands the edge of a network from to physical servers to virtual machines. As a consequence, a variety of problems, such as resource contention, performance isolation, and inter-VM communication, emerge. We believe that any DCN architectures designed for modern (virtualized) data centers have to properly address the above issues before being considered for actual deployment. Our future work involves evaluating other DCN architectures in a larger-scale testbed and an actual operational data center, and expanding the background environment with more multi-tier applications. In addition, designing new data center network architectures that explicitly address issues raised in this paper forms another line of our future work.

References

[1] M. Al-Fares, A. Loukasass, and A. Vahdat, "A Scalable, Commodity Data Center Network Architecture," in *Proc. ACM SIGCOMM*, Aug. 2008, pp. 63–74.

[2] 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," in *Proc. ACM SIGCOMM*, Aug. 2009, pp. 51–62.

[3] A. Greenberg, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "Towards a Next Generation Data Center Architecture: Scalability and Commoditization," in *Proc. ACM Workshop on Programmable Routers for Extensible Services of Tomorrow*, Aug. 2008, pp. 57–62.

[4] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu, "BCube: a High Performance, Server-Centric Network Architecture for Modular Data Centers," in *Proc. ACM SIGCOMM*, Aug. 2009, pp. 63–74.

[5] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu, "DCell: A Scalable and Fault-Tolerant Network Structure for Data Centers," in *Proc. ACM SIGCOMM*, Aug. 2008, pp. 75–86.

[6] D. Li, C. Guo, H. Wu, K. Tan, Y. Zhang, and S. Lu, "FiConn: Using Backup Port for Server Interconnection in Data Centers," in *Proc. IEEE INFOCOM*, Apr. 2009, pp. 2276–2285.

[7] 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 *Proc. ACM SIGCOMM*, Aug. 2009, pp. 39–50.

[8] The Click Modular Router Project. [Online]. Available: http://read.cs.ucla.edu/click/.

[9] RUBiS: Rice University Bidding System. [Online]. Available: http://rubis.ow2.org/.

[10] H. Wu, G. Lu, D. Li, C. Guo, and Y. Zhang, "MDCube: A High Performance Network Structure for Modular Data Center Interconnection," in *Proc. ACM SIGCOMM CoNEXT*, Dec. 2009.

[11] Xen. [Online]. Available: http://www.xen.org/.