# An FPGA-based Simulator for Datacenter Networks

Zhangxi Tan
Computer Science Division
UC Berkeley, CA
xtan@eecs.berkeley.edu

Krste Asanović
Computer Science Division
UC Berkeley, CA
krste@eecs.berkeley.edu

David Patterson
Computer Science Division
UC Berkeley, CA
pattrsn@eecs.berkeley.edu

## ABSTRACT

We describe an FPGA-based datacenter network simulator for researchers to rapidly experiment with O(10,000) node datacenter network architectures. Our simulation approach configures the FPGA hardware to implement abstract models of key datacenter building blocks, including all levels of switches and servers. We model servers using a complete SPARC v8 ISA implementation, enabling each node to run real node software, such as LAMP and Hadoop. Our initial implementation simulates a 64-server system and has successfully reproduced the TCP incast throughput collapse problem. When running a modern parallel benchmark, simulation performance is two-orders of magnitude faster than a popular full-system software simulator. We plan to scale up our testbed to run on multiple BEE3 FPGA boards, where each board is capable of simulating 1500 servers with switches.

## 1. INTRODUCTION

In recent years, datacenters have been growing rapidly to scales of 10,000 to 100,000 servers [18]. Many key technologies make such incredible scaling possible, including modularized container-based datacenter construction and server virtualization. Traditionally, datacenter networks employ a fat-tree-like three-tier hierarchy containing thousands of switches at all levels: rack level, aggregate level, and core level [1].

As observed in [13], the network infrastructure is one of the most vital optimizations in a datacenter. First, networking infrastructure has a significant impact on server utilization, which is an important factor in datacenter power consumption. Second, network infrastructure is crucial for supporting data intensive Map-Reduce jobs. Finally, network infrastructure accounts for 18% of the monthly datacenter costs, which is the third largest contributing factor. In addition, existing large commercial switches and routers command very healthy margins, despite being relatively unreliable [26]. Sometimes, correlated failures are found in replicated million-dollar units [26]. Therefore, many researchers have proposed novel datacenter network architectures [14, 15, 17, 22, 25, 26] with most of them focusing on new switch designs. There are also several new network products emphasizing low latency and simple switch designs [3, 4].

When comparing these new network architectures, we found a wide variety of design choices in almost every aspect of the design space, such as switch designs, network topology, protocols, and applications. For example, there is an ongoing debate between low-radix and high-radix switch design. We believe these basic disagreements about fundamental design decisions are due to the different observations and assumptions taken from various existing datacenter infrastructures and applications, and the lack of a sound methodology to evaluate new options. Most proposed designs have only been tested with a very small testbed running unrealistic microbenchmarks, as it is very difficult to evaluate network architecture innovations at scale without first building a large datacenter.

To address the above issue, we propose using Field-Programmable Gate Arrays (FPGAs) to build a reconfigurable simulation testbed at the scale of O(10,000) nodes. Each node in the testbed is capable of running real datacenter applications. Furthermore, network elements in our testbed provide detailed visibility so that we can examine the complex network behavior that administrators see when deploying equivalently scaled datacenter software. We built the testbed on top of a cost-efficient FPGA-based full-system manycore simulator, RAMP Gold [24]. Instead of mapping the real target hardware directly, we build several abstracted models of key datacenter components and compose them together in FPGAs. We can then construct a 10,000-node system from a rack of multi-FPGA boards, e.g., the BEE3 [10] system. To the best of our knowledge, our approach will probably be the first to simulate datacenter hardware along with real software at such a scale. The testbed also provides an excellent environment to quantitatively analyze and compare existing network architecture proposals.

We show that although the simulation performance is slower than prototyping a datacenter using real hardware, abstract FPGA models allow flexible parameterization and are still two orders of magnitude faster than software simulators at the equivalent level of detail. As a proof of concept, we built a prototype of our simulator in a single Xilinx Virtex 5 LX110 FPGA simulating 64 servers connecting to a 64-port rack switch. Employing this testbed, we have successfully reproduced the TCP Incast throughput collapse effect [27], which occurs in real datacenters. We also show the importance of simulating real node software when studying the TCP Incast problem.

| Network Architecture | Testbed | Scale | Workload |
|---|---|---|---|
| Policy away switching layer [17] | Click software router | Single switch | Microbenchmark |
| DCell [16] | Commercial hardware | ∼20 nodes | Synthetic workload |
| Portland (v1) [6] | Virtual machine+commercial switch | 20 switches+16 servers | Microbenchmark |
| Portland (v2) [22] | Virtual machine+NetFPGA | 20 switches+16 servers | Synthetic workload |
| BCube [15] | Commercial hardware+NetFPGA | 8 switches+16 servers | Microbenchmark |
| VL2 [14] | Commercial hardware | 10 servers+10 switches | Microbenchmark |
| Thacker's container network [26] | Prototyping with FPGA boards | - | - |

**Table 1: Datacenter network architecture proposals and their evaluations**

## 2. EVALUATING DATACENTER NETWORKS

We begin by identifying the key issues in evaluating datacenter networks. Several recent novel network architectures employ a simple, low-latency, supercomputer-like interconnect. For example, the Sun Infiniband datacenter switch [3] has a 300 ns port-port latency as opposed to the 7–8 $\mu$s of common Gigabit Ethernet switches. As a result, evaluating datacenter network architectures really requires simulating a computer system with the following three features.

1. *Scale*: Datacenters contains O(10,000) servers or more.
2. *Performance*: Large datacenter switches have 48/96 ports, and are massively parallel. Each port has 1–4 K flow tables and several input/output packet buffers. In the worst case, there are ∼200 concurrent events every clock cycle.
3. *Accuracy*: A datacenter network operates at nanosecond time scales. For example, transmitting a 64-byte packet on a 10 Gbps link takes only ∼50 ns, which is comparable to DRAM access time. This precision implies many fine-grained synchronizations during simulation if models are to be accurate.

Table 1 summarizes evaluation methodologies in recent network design research. Clearly, the biggest issue is evaluation scale. Although a mid-size datacenter contains tens of thousands of servers and thousands of switches, recent evaluations have been limited to relatively tiny testbeds with less than 100 servers and 10–20 switches. Small-scale networks are usually quite understandable, but results obtained may not be predictive of systems deployed at large scale.

For workloads, most evaluations run synthetic programs, microbenchmarks, or even pattern generators, while real datacenter workloads include web search, email, and MapReduce jobs. In large companies, like Google and Microsoft, trace-driven simulation is often used, due to the abundance of production traces. But production traces are collected on existing systems with drastically different network architectures. They cannot capture the effects of timing-dependent execution on a new proposed architecture.

Finally, many evaluations make use of existing commercial off-the-shelf switches. The architectural details of these commercial products are proprietary, with poor documentation of existing structure and little opportunity to change parameters such as link speed and switch buffer configurations, which may have significant impact on fundamental design decisions.

### 2.1 The Potential of FPGA-based Simulation

As the RAMP [28] project observed, FPGAs have become a promising vehicle for architectural investigation of massively parallel computer systems. We propose building a datacenter simulator based on the RAMP Gold FPGA simulator [24], to model up to O(10,000) nodes and O(1,000) switches running real datacenter software. Figure 1 abstractly compares our RAMP-based approach to four existing approaches in terms of experiment scale and accuracy.

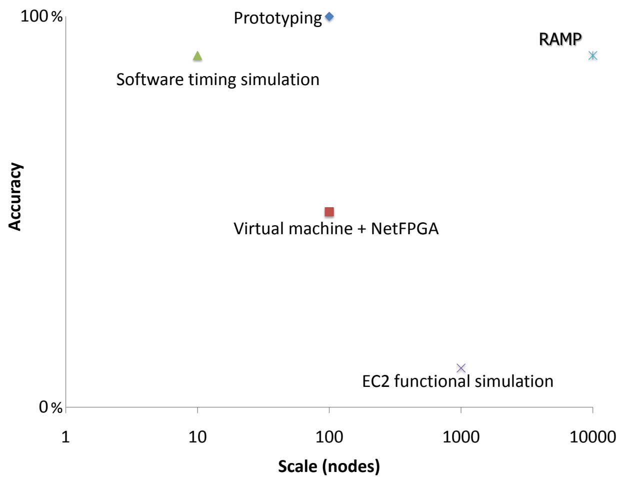

**Figure 1: RAMP vs. Existing Evaluations**

Prototyping has the highest accuracy, but it very expensive to scale beyond O(100) nodes. To increase the number of tested end-hosts, many evaluations [22] utilize virtual machines (VMs) along with programmable network devices, such as NetFPGA [21]. However, multiple VMs time-multiplex on a single physical machine and share the same switch port resource. Hence, true concurrency and switch timing is not faithfully modeled. In addition, it is still very expensive to reach the scale of O(1,000) in practice.

To accurately model architectural details, computer architects often use full-system software timing simulators, for example M5 [8] and Simics [20]. Programs running on these simulators are hundreds of thousands of times slower than running on a real system. To keep simulation time reasonable, such simulators are rarely used to simulate more than a few dozen nodes.

Recently, cloud computing platforms such as Amazon EC2 offer a pay-per-use service to enable users to share their datacenter infrastructure at an O(1,000) scale. Researchers can rapidly deploy a functional-only testbed for network management and control plane studies. Such services, however, provide almost no visibility into the network and have no mechanism for accurately experimenting with new switch architectures.

## 3. RAMP GOLD FOR DATACENTER SIMULATION

In this section, we first review the RAMP Gold CPU simulation model before describing how we extend it to model a complete datacenter including switches, and then provide predictions of scaled-up simulator performance.

### 3.1 RAMP Gold

RAMP Gold is an economical FPGA-based cycle-accurate full-system architecture simulator that allows rapid early design-space exploration of manycore systems. RAMP Gold employs many FPGA-friendly optimizations and has high simulation throughput. RAMP Gold supports the full 32-bit SPARC v8 ISA in hardware, including floating-point and precise exceptions. It also models sufficient hardware to run an operating system including MMUs, timers, and interrupt controllers. Currently, we can boot the Linux 2.6.21 kernel and a manycore research OS[19].

We term the computer system being simulated the *target*, and the FPGA environment running the simulation the *host*. RAMP Gold uses the following three key techniques to simulate a large number of cores efficiently:

1. *Abstracted Models*: A full RTL implementation of a target system ensures precise cycle-accurate timing, but it requires considerable effort to implement the hardware of a full-fledged datacenter in FPGAs. In addition, the intended new switch implementations are usually not known during the early design stage. Instead of full RTL, we employ high-level abstract models that greatly reduce both model construction effort and FPGA resource requirements.

2. *Decoupled functional/timing models* RAMP Gold decouples the modeling of target timing and functionality. For example, in server modeling, the *functional model* is responsible for executing the target software correctly and maintaining architectural state, while the *timing model* determines how long an instruction takes to execute in the target machine. Decoupling simplifies the FPGA mapping of the functional model and allows complex operations to take multiple FPGA cycles. It also improves modeling flexibility and model reuse. For instance, we can use the same switch functional model to simulate both 10 Gbps switches and 100 Gbps switches, by changing only the timing model.

3. *Multithreading* Since we are simulating a large number of cores, RAMP Gold applies multithreading to both the functional and timing models. Instead of replicating hardware models to model multiple instances in the target, we use multiple hardware model threads running in a single host model to simulate different target cores. Multithreading significantly improves the FPGA resource utilization and hides simulation latencies, such as those from host DRAM access and timing model synchronization. The timing model correctly models the true concurrency in the target, independent of the time-multiplexing effect of multithreading in the host simulation model.

The prototype of RAMP Gold runs on a single Xilinx Virtex-5 LX110T FPGA, simulating a 64-core multiprocessor target with a detailed memory timing model. We ran six programs from a popular parallel benchmark, PARSEC [7], on a research OS. Figure 2 shows the geometric mean of the simulation speedup on RAMP Gold compared to a popular software architecture simulator, Simics [20], as we scale the number of cores and the level of detail. We configure the software simulator with three timing models of different levels of detail. Under the 64-core configuration with the most accurate timing model, RAMP Gold is 263x faster than Simics. Note that at this accuracy level Simics performance degrades super-linearly with the number of cores simulated: 64 cores is almost 40x slower than 4 cores.

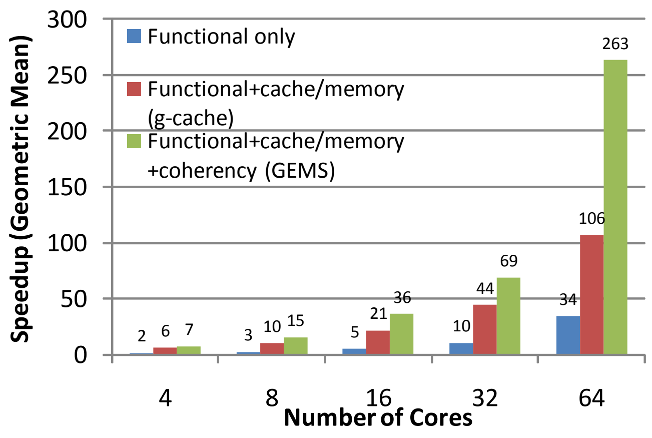

**Figure 2: RAMP Gold speedup over Simics running the PARSEC benchmark**

### 3.2 Modeling a Datacenter with RAMP Gold

Our datacenter simulator contains two types of models: node and switch. The node models a networked server in a datacenter, which talks over some network fabric (e.g. Gigabit Ethernet) to a switch. We assume each target server executes the SPARC v8 ISA, which is simulated with one hardware thread in RAMP Gold. By default, we assign a simple in-order issue CPU timing model with a fixed CPI for each target server. The target core frequency is adjustable by configuring the timing model at runtime, which simulates scaling of node performance. We can also add more detailed processor and memory timing models for points of interest.

Similar to the server model, the switch models are also host-threaded, with decoupled timing and functional models. Each hardware thread simulates a single target switch port, while the switch packet buffer is modeled using DRAM. The model also supports changing architectural parameters—such as link bandwidth, delays, and switch buffer size—without time-consuming FPGA resynthesis. The current switch model simulates a simple output-buffered source-routed architecture. We plan to add a conventional switch model soon. We use a ring to physically connect all switches and node models on one host FPGA, but can model any arbitrary target topology.

Each functional/timing model pipeline supports up to 64 hardware threads, simulating 64 target servers. We can also configure fewer threads per pipeline to improve single-thread performance. To reach O(10,000) scale, we plan to put down multiple threaded RAMP Gold pipelines in a rack of 10 BEE3 boards as shown in Figure 3. Each BEE3 board has four Xilinx Virtex-5 LX155T FPGAs connected with a 72-bit wide LVDS ring interconnect. Each FPGA supports 16 GB of DDR2 memory in two independent channels, resulting in up to 64 GB of memory for the whole board. Each FPGA provides two CX4 ports, which can be used as two 10 Gbps Ethernet interfaces to connect multiple boards.

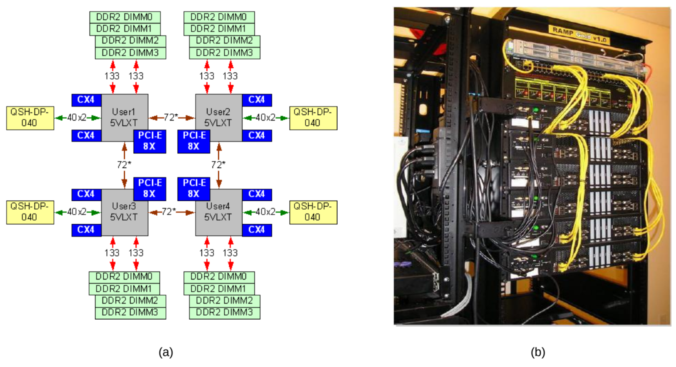

**Figure 3: a) Architecture of a BEE3 board. b) A rack of six BEE3 boards, each with 4 FPGAs.**

On each FPGA, we can fit six pipelines to simulate up to 384 servers. We then can simulate 1,536 servers on one BEE3 board, since there are four FPGAs on each board. The onboard 64 GB DRAM simulates the target memory, with each simulated node having a share of ∼40 MB.

We are looking at expanding memory capacity by employing a hybrid memory hierarchy including both DRAM and flash. Using the BEE3 SLC flash DIMM [11], we can build a target system with a 32 GB DRAM cache and 256 GB flash memory on every BEE3.

In terms of the host memory bandwidth utilization, when running the PARSEC benchmark, one 64-thread pipeline only uses <15% of the peak bandwidth of a single-channel DDR2 memory controller. Each BEE3 FPGA has two memory channels, so it should have sufficient host memory bandwidth to support six pipelines.

In terms of FPGA utilization for networking, the switch models consume trivial resources. Our 64-port output-buffered switch only takes ∼300 LUTs on a Xilinx Virtex-5 FPGA. Moreover, novel datacenter switches are much simpler than traditional designs. Even real prototyping takes only < 10% of the resources on a midsize Virtex-5 FPGA [12].

Each simulated node in our system runs Debian Linux with a full Linux 2.6 kernel. LAMP (Linux, Apache, Mysql, PHP) and Java support is from the binary packages of Debian Linux. We plan to run Map-Reduce benchmarks from Hadoop [5] as well as three-tiered Web 2.0 benchmarks, e.g. Cloudstone [23]. Since each node is SPARC v8 compatible and has a full GNU development environment, the platform is capable of running other datacenter research codes compiled from scratch.

### 3.3 Predicted Simulator Performance

One major datacenter application is running Map-Reduce jobs, where each job contains the two types of tasks: map tasks and reduce tasks. According to the production data from Facebook and Yahoo datacenters [29], the medium map task length at Facebook and Yahoo is 19 seconds and 26 seconds respectively, while the medium reduce task length is 231 seconds and 76 seconds respectively. Moreover, small and short jobs dominate, while there are more map tasks than reduce tasks. In reality, most tasks will finish sooner than medium length tasks.

Table 2 shows the simulation time with different number of hardware threads on RAMP Gold, if we simulate these medium length tasks till completion. To predict the Map-Reduce performance, we use the simulator performance data gathered while running the PARSEC benchmark. Map tasks can be finished in a few hours, while reduce tasks take longer, ranging from a few hours to a few days. Using fewer threads per pipeline gives better performance at the cost of simulating fewer servers. Note the simulation time in Table 2 is only for a single task. Multiple tasks can run simultaneously, because the testbed can simulate a large number of servers. The simulation slowdown compared to a real datacenter is roughly around 1,000x under the 64-thread configuration. This is comparable to a software network simulator used at Google, which has a slowdown of 600× [2] but doesn't simulate any node software.

| Target System | Map Task | Reduce Task |
|---|---|---|
| Facebook (64 threads/pipeline) | 5 hours | 64 hours |
| Yahoo (64 threads/pipeline) | 7 hours | 21 hours |
| Facebook (16 threads/pipeline) | 1 hours | 16 hours |
| Yahoo (16 threads/pipeline) | 2 hours | 5 hours |

**Table 2: Simulation time of a single median length task on RAMP Gold**

In the next section, we present a small case study to show that simulating the software stack on the server node can be vital for uncovering network problems.

## 4. CASE STUDY: REPRODUCING THE TCP INCAST PROBLEM

As a proof of concept, we use RAMP Gold to study the TCP Incast throughput collapse problem [27]. The TCP Incast problem refers to the application-level throughput collapse that occurs as the number of servers sending data to a client increases past the ability of an Ethernet switch to buffer packets. Such a situation is common within a rack, where multiple clients connecting to the same switch share a single storage server, such as for NFS servers.

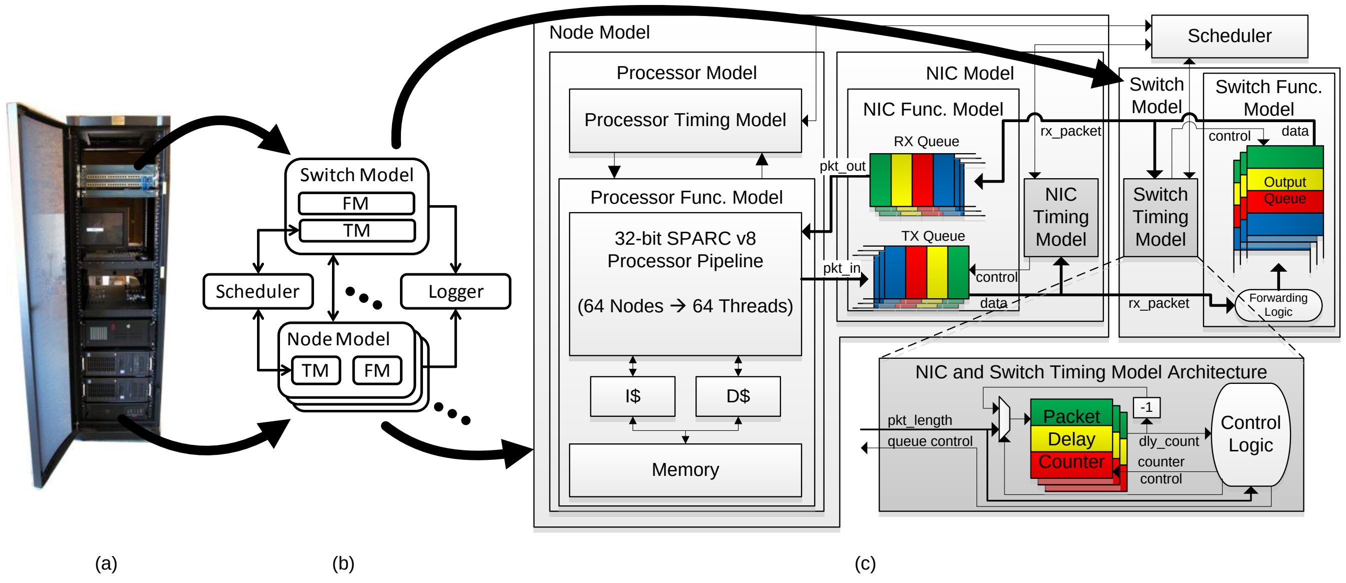

**Figure 4: Mapping the TCP Incast problem to RAMP Gold. a) The target is a 64-server datacenter rack. b) High-level RAMP Gold models. c) Detailed RAMP Gold models.**

Figure 4 illustrates the mapping of the TCP Incast problem on our FPGA simulation platform. The target system is a 64-node datacenter rack with a single output-buffered layer-2 gigabit switch. The node model is simulated with a single 64-thread SPARC v8 RAMP Gold pipeline. The NIC and switch timing models are similar, both computing packet delays based on the packet length, link speed and queuing states.

Figure 5 shows the RAMP simulation results compared to those from the real measurements in [9], when varying the TCP retransmission time out (RTO). As seen from the graph, the simulation results differ from the measured data in terms of absolute values. This difference is mainly because commercial switch architectures are proprietary, so that we lack the information needed to create an accurate model. Nevertheless, the shapes of the throughput curves are similar. This similarity suggests that using an abstract switch model can still successfully reproduce the throughput collapse effect and the trend with more senders. Moreover, the original measurement data contained only up to 16 senders due to the many practical engineering issues of building a larger testbed. In contrast, it is very easy to scale using our RAMP simulation.

In order to show the importance of simulating node software, we replace the RPC-like application sending logic with a simple naïve FSM-based sender to imitate more conventional network evaluations. The FSM-based sender does not simulate any computation and directly connects to the TCP protocol stack. Figure 6 shows the receiver throughput of FSM senders versus normal senders. We configure a 200 ms TCP RTO and 256 KB switch port buffer. Illustrated clearly on the graph, FSM senders cannot reproduce the throughput collapse, as the throughput gradually recovers with more

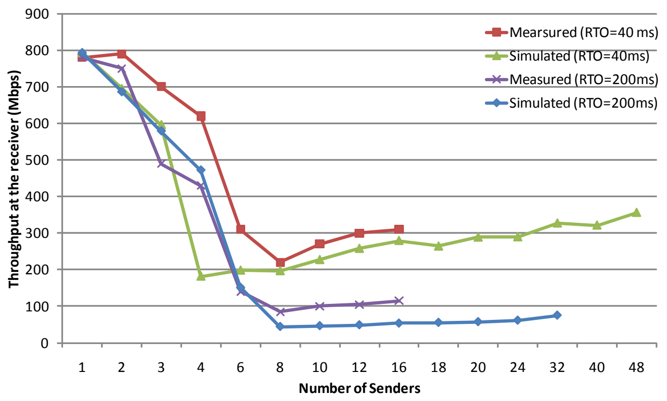

**Figure 5: RAMP Gold simulation vs. real measurement**

FSM senders. The throughput collapse is also not as significant as that of normal senders. In conclusion, the absence of node software and application logic in simulation may lead to a very different result.

## 5. CONCLUSION AND FUTURE WORK

Our initial results show that simulating datacenter network architecture is not only a networking problem, but also a computer system problem. Real node software significantly affects the simulation results even at the rack level. Our FPGA-based simulation can improve both the scale and the accuracy of network evaluation. We believe it will be promising for datacenter-level network experiments, helping to evaluate novel protocols and software at scale.

Our future work primarily involves developing system func-

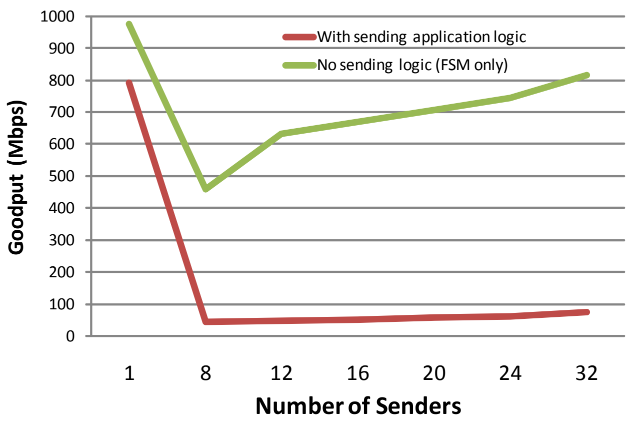

**Figure 6: Importance of simulating node software**

tionality and scaling the testbed to multiple FPGAs. We also plan to use the testbed to quantitatively analyze previously proposed datacenter network architectures running real applications.

## 6. ACKNOWLEDGEMENT

We would like to thank Jonathan Ellithorpe for the contribution of modeling TCP Incast and Glen Anderson for helpful discussion. This research is supported in part by gifts from Sun Microsystems, Google, Microsoft, Amazon Web Services, Cisco Systems, Cloudera, eBay, Facebook, Fujitsu, Hewlett-Packard, Intel, Network Appliance, SAP, VMWare and Yahoo! and by matching funds from the State of California's MICRO program (grants 06-152, 07-010, 06-148, 07-012, 06-146, 07-009, 06-147, 07-013, 06-149, 06-150, and 07-008), the National Science Foundation (grant #CNS-0509559), and the University of California Industry/University Cooperative Research Program (UC Discovery) grant COM07-10240.

## 7. REFERENCES

[1] Cisco data center: Load balancing data center services, 2004.
[2] Glen Anderson, private communications, 2009.
[3] Sun Datacenter InfiniBand Switch 648, http://www.sun.com/products/networking/infiniband.jsp, 2009.
[4] Switching Architectures for Cloud Network Designs, http://www.aristanetworks.com/en/SwitchingArchitecture_wp.pdf, 2009.
[5] Hadoop, http://hadoop.apache.org/, 2010.
[6] M. Al-Fares, A. Loukissas, and A. Vahdat. A scalable, commodity data center network architecture. In *SIGCOMM '08: Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, pages 63–74, New York, NY, USA, 2008. ACM.
[7] C. Bienia et al. The PARSEC Benchmark Suite: Characterization and Architectural Implications. In *PACT '08*, pages 72–81, New York, NY, USA, 2008. ACM.
[8] N. L. Binkert, R. G. Dreslinski, L. R. Hsu, K. T. Lim, A. G. Saidi, and S. K. Reinhardt. The M5 simulator: Modeling networked systems. *IEEE Micro*, 26(4):52–60, 2006.
[9] Y. Chen, R. Griffith, J. Liu, R. H. Katz, and A. D. Joseph. Understanding TCP incast throughput collapse in datacenter networks. In *WREN '09: Proceedings of the 1st ACM workshop on Research on enterprise networking*, pages 73–82, New York, NY, USA, 2009. ACM.
[10] J. Davis, C. Thacker, and C. Chang. BEE3: Revitalizing Computer Architecture Research. Technical Report MSR-TR-2009-45, Microsoft Research, Apr 2009.
[11] J. D. Davis and L. Zhang. FRP: a Nonvolatile Memory Research Platform Targeting NAND Flash. In *The First Workshop on Integrating Solid-state Memory into the Storage Hierarchy, Held in Conjunction with ASPLOS 2009*, March 2009.
[12] N. Farrington, E. Rubow, and A. Vahdat. Data center switch architecture in the age of merchant silicon. In *HOTI '09: Proceedings of the 2009 17th IEEE Symposium on High Performance Interconnects*, pages 93–102, Washington, DC, USA, 2009. IEEE Computer Society.
[13] 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.*, 39(1):68–73, 2009.
[14] 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 *SIGCOMM '09: Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, pages 51–62, New York, NY, USA, 2009. ACM.
[15] 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 *SIGCOMM '09: Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, pages 63–74, New York, NY, USA, 2009. ACM.
[16] 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 '08: Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, pages 75–86, New York, NY, USA, 2008. ACM.
[17] D. A. Joseph, A. Tavakoli, and I. Stoica. A policy-aware switching layer for data centers. In *SIGCOMM '08: Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, pages 51–62, New York, NY, USA, 2008. ACM.
[18] R. Katz. Tech titans building boom: The architecture of internet datacenters. *IEEE Spectrum*, February 2009.
[19] R. Liu et al. Tessellation: Space-Time partitioning in a manycore client OS. In *HotPar09*, Berkeley, CA, 03/2009 2009.
[20] P. S. Magnusson et al. Simics: A Full System Simulation Platform. *IEEE Computer*, 35, 2002.
[21] J. Naous, G. Gibb, S. Bolouki, and N. McKeown. NetFPGA: Reusable router architecture for experimental research. In *PRESTO '08: Proceedings of the ACM workshop on Programmable routers for extensible services of tomorrow*, pages 1–7, New York, NY, USA, 2008. ACM.
[22] R. Niranjan 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 '09: Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, pages 39–50, New York, NY, USA, 2009. ACM.
[23] W. Sobel, S. Subramanyam, A. Sucharitakul, J. Nguyen, H. Wong, A. Klepchukov, S. Patil, A. Fox, and D. Patterson. Cloudstone: Multi-platform, multi-language benchmark and measurement tools for web 2.0. In *CCA '08: Proceedings of the 2008 Cloud Computing and Its Applications*, Chicago, IL, USA, 2008.
[24] Z. Tan, A. Waterman, R. Avizienis, Y. Lee, D. Patterson, and K. Asanović. Ramp Gold: An FPGA-based architecture simulator for multiprocessors. In *4th Workshop on Architectural Research Prototyping (WARP-2009), at 36th International Symposium on Computer Architecture (ISCA-36)*, 2009.
[25] A. Tavakoli, M. Casado, T. Koponen, and S. Shenker. Applying NOX to the datacenter. In *HotNets*, 2009.
[26] C. Thacker. Rethinking data centers. October 2007.
[27] V. Vasudevan, A. Phanishayee, H. Shah, E. Krevat, D. G. Andersen, G. R. Ganger, G. A. Gibson, and B. Mueller. Safe and effective fine-grained TCP retransmissions for datacenter communication. In *SIGCOMM '09: Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, pages 303–314, New York, NY, USA, 2009. ACM.
[28] J. Wawrzynek et al. RAMP: Research Accelerator for Multiple Processors. *IEEE Micro*, 27(2):46–57, 2007.
[29] M. Zaharia, D. Borthakur, J. Sen Sarma, K. Elmeleegy, S. Shenker, and I. Stoica. Job scheduling for multi-user mapreduce clusters. Technical Report UCB/EECS-2009-55, EECS Department, University of California, Berkeley, Apr 2009.