DIABLO: A Warehouse-Scale Computer Network Simulator using FPGAs

Zhangxi Tan Zhenghao Qian Xi Chen Krste Asanović David Patterson
University of California, Berkeley
{xtan,peterfly,c.xi,krste,pattrsn}@eecs.berkeley.edu

Abstract

Motivated by rapid software and hardware innovation in warehouse-scale computing (WSC), we visit the problem of warehouse-scale network design evaluation. A WSC is composed of about 30 arrays or clusters, each of which contains about 3000 servers, leading to a total of about 100,000 servers per WSC. We found many prior experiments have been conducted on relatively small physical testbeds, and they often assume the workload is static and that computations are only loosely coupled with the adaptive networking stack. We present a novel and cost-efficient FPGA-based evaluation methodology, called Datacenter-In-A-Box at LOw cost (DIABLO), which treats arrays as whole computers with tightly integrated hardware and software. We have built a 3,000-node prototype running the full WSC software stack. Using our prototype, we have successfully reproduced a few WSC phenomena, such as TCP Incast and memcached request latency long tail, and found that results do indeed change with both scale and with version of the full software stack.

Categories and Subject Descriptors C.5.3 [*Computer System Implementation*]: Microprocessors; I.6.8 [*Simulation and Modeling*]: Discrete Event

Keywords Warehouse-scale computing; FPGA; Performance Evaluation

1. Introduction

Modern datacenters are no longer just a collection of servers running traditional commercial workloads. The tremendous success of Internet services has led to the rise of Warehouse-Scale Computers (WSCs) [25] with 50,000 to 100,000 servers [41] treated as a massive computer system. The continuing push for greater cost efficiency and scalability has driven leading Internet companies, such as Facebook and Google, to innovate in new software infrastructure and even custom hardware [15, 18].

As noted in many recent papers [35, 58], the network is perhaps the most critical component in a WSC. The growing scale of WSCs leads to increasing and more variable network delays, while demand for more ports increases cost in an already expensive switching fabric. Consequently, many researchers have proposed novel datacenter network architectures, mostly focusing on new switch designs [36, 37, 40, 50, 57, 58], and several new networking products emphasize simple switch designs with low latency [9, 10].

When comparing these network proposals, we observe a wide variety of design choices made for almost every aspect of the design space, including switch design, network topology, protocol, and applications. We believe these basic disagreements about fundamental design decisions are due to the different assumptions taken by various existing WSC infrastructures and applications, and the lack of a sound methodology to evaluate new options. Most proposed designs have only been tested with a small testbed running unrealistic microbenchmarks, as it is very difficult to evaluate network infrastructure innovations at scale without first building a WSC. We surveyed WSC-related papers in the SIGCOMM conference from 2008 to 2013 and found that the median size of physical testbeds contained only 16 servers and 6 switches (see Figure 2).

In this paper, we present the design of DIABLO (Datacenter-In-A-Box at LOw cost), an FPGA-based WSC simulator that enables the at-scale and in-depth modeling of WSC arrays as complete computer systems with tightly integrated hardware and software. DIABLO can run unmodified WSC applications and operating systems at the scale of O(1,000) to O(10,000) nodes while providing detailed models of every instruction executed on every node, the operation of the network interfaces on each node, and the movement of every byte in every packet through multiple levels of datacenter switches.

ASPLOS '15, Month 14–18, 2015, Istanbul, Turkey.

http://dx.doi.org/10.1145/2694344.2694362

The specific target of DIABLO is the building block of the WSC, called an *array* [38] or *cluster* [25]. A WSC is composed of about 20–40 such arrays, each of which contains about 2000–3000 servers, summing to a total of about 50,000–100,000 servers per WSC. The array is often the target of an application, with multiple arrays used for both dependability and to distribute workload. DIABLO is fast enough to model O(10) seconds of runtime of a whole WSC array in a few hours instead of the weeks that would be required for an equivalent software simulator. To increase FPGA efficiency and flexibility, DIABLO is not based on FPGA prototyping, where hardware designs are directly mapped to FPGAs, but instead uses FPGAs to accelerate parameterized abstract performance models of the system [55].

The number of compute nodes in the DIABLO prototype is two orders-of-magnitude greater than recent testbeds used by academic researchers, but the FPGA-based simulation approach makes it plausible for many research groups to acquire their own private platform. The projected full-system hardware cost of an O(10,000)-node DIABLO system using modern state-of-the-art FPGAs is around $150K. An equivalent real WSC array would cost around $36M in CAPEX and $800K in OPEX/month. Furthermore, unlike real hardware, DIABLO is fully parameterizable and fully instrumented, and supports repeatable deterministic experiments.

In our evaluation, we show how results obtained from large-scale DIABLO runs can be dramatically different from those obtained from smaller testbeds. In particular, DIABLO allows us to record the same behavior administrators observe when deploying equivalently scaled WSC software, such as TCP Incast throughput collapse [60] and the long tails of memcached request latency.

2. Background and Related Work

In this section, we first review the structure of WSCs, then discuss existing evaluation approaches and previous studies, and conclude with a summary of WSC simulation needs.

2.1 WSC Network Architecture

WSCs use a hierarchy of local-area networks (LAN) and off-the-shelf switches. Figure 1 shows a typical WSC network arranged in a Clos topology with three networking layers. At the bottom layer, each rack typically holds 20–40 servers, each singly connected to a commodity *Top-of-Rack (ToR)* switch with a 1 Gbps link. These ToR switches usually offer two to eight uplinks, which leave the rack to connect up to several *array switches* to provide redundancy and bandwidth. At the top of the hierarchy, *datacenter switches* carry traffic between array switches usually using 10 Gbps links. All links use Ethernet physical-layer protocol, with either copper or fiber cabling depending on connection distance.

One of the most challenging design problems is that the bandwidth "over-subscription" ratio (i.e. bandwidth entering from below versus bandwidth available to level above) worsens rapidly as we move up the hierarchy. This imbalance is due to the cost of switch bandwidth, which grows quadratically in the number of switch ports. The resulting limited WSC bisection bandwidth significantly affects the design of software and the placement of services and data, hence the current active interest in improving network switch designs.

Recent novel network architectures employ a simple, low-latency, supercomputer-like interconnect. For example, the Sun Infiniband datacenter switch [10] has a 300 ns port-port latency as opposed to the 7–8 μs of common Gigabit Ethernet switches. Even at the WSC level, where huge traditional Cisco and Juniper switches or routers have dominated to handle inter-WSC traffic, simple switches with minimal software are now preferred. For example, the Google G-scale OpenFlow switch [47] runs almost no software except the OpenFlow agent using just BGP and ISIS protocols.

2.2 Evaluation Approaches

We next compare the different methodologies that have been used to evaluate designs for WSC networks.

Production prototyping: In industry, perhaps the ideal way to test new hardware is to build the real system at a smaller scale and run it alongside a production system [34]. In this way, new hardware evaluation will benefit from running production software and a more realistic workload. However, this approach suffers from scalability issues. First, it is expensive to build a large number of prototypes. Second, it is risky to deploy experimental hardware at a large-scale in a production environment. Third, the majority of the testing workload will be generated by older and slower computers.

Testing clusters: Another popular approach is to deploy the test equipment in medium-scale shared testing clusters. For instance, the Yahoo M45 [7] cluster has 4,000 processors and 1.5 petabytes of data storage, and is designed to run data-intensive distributed computing platforms such as Hadoop [13]. Larger Internet companies can afford much larger testing clusters, e.g. 20,000 nodes at Google [8]. Although these test clusters have enough computing nodes to help diagnose interesting scalability problems, their construction and maintenance costs are enormous. Bringing up an equivalent production software stack is also another practical issue [8]. Because of cost, researchers in academia use much smaller in-house clusters at the scale of 40–80 nodes (1–2 racks) [36, 51, 61] of off-the-shelf hardware. Commer-

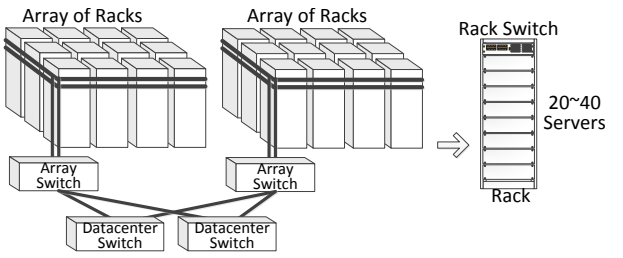


Figure 1. A typical WSC network architecture.

cial hardware usually has few controllable parameters, so experiments can only explore a limited design space, and has limited observability to help understand behavior, which complicates analysis given that experimental runs with real hardware are not generally reproducible.

Cloud computing: Cloud computing platforms such as Amazon EC2 offer a pay-per-use service based on virtual machine (VM) technology to enable users to share their WSC infrastructure at an O(1,000) scale. Researchers can pay $0.10/hour/node to rapidly deploy a functional-only testbed for network management and control plane studies [30, 63]. The cloud is a straightforward approach for software developers to acquire a large-scale infrastructure. Such services, however, provide almost no visibility or control over the network and have no mechanism for accurately experimenting with new networking architectures.

Full-system software simulations: To avoid the high capital cost of hardware prototyping, researchers have long used software simulators to explore new architectural ideas at all levels, from microarchitectures and instruction sets [46] to full systems [27]. Alas, the recent abrupt transitions to massively distributed architectures and high-radix switches have both increased the complexity and scale of simulation. The complex interaction of networking software with networking hardware renders popular techniques, such as sampling and simple analytical models, unreliable at reproducing actual steady-state behavior even in the simplest TCP/IP networking environments [39].

Analytical simulation models: Researchers have used stochastic queuing models in simulations [43, 48] to gain insight into existing systems. Instead of simulating the detailed microarchitecture, such analytical models raise the level of abstraction, provided measurements of distributions of arrival and service times for various tasks in the system are available. Although analytical models are much faster than full-scale simulation, it is challenging to build faithful models for new HW/SW that captures interesting interactions at the 100,000-server scale when there is no system to measure.

2.3 Recent WSC Network Evaluation Studies

We next survey previous WSC-network evaluation studies. Figure 2 plots the scale of the physical testbeds used by datacenter networking papers published in SIGCOMM from 2008 to 2013. To be generous in our survey, we count VMs as physical nodes, although VMs time-share hardware resources and might not reflect accurate timing in the real physical network. In addition, when EC2 is used, there is no visibility into network performance. We optimistically estimate the maximum number of switches assuming each physical machine only hosts one VM.

Clearly, the biggest issue is evaluation scale. Although a mid-size WSC contains tens of thousands of servers and thousands of switches, recent evaluations have been limited to relatively small testbeds with less than 100 servers and 10–20 switches. Small-scale systems are understandable, but results obtained may not be predictive of systems deployed at large scale. As we show later in this paper, sometimes design decisions drawn from O(100) testbeds are completely reversed when scaling to O(1,000) nodes.

For workloads, most evaluations run synthetic programs, microbenchmarks, or even pattern generators, but real WSC workloads include web search, email, and Map-Reduce jobs. In large companies, like Google and Microsoft, researchers typically use trace-driven simulation, due to the abundance of production traces. Nevertheless, 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. In the past, when looking at networking designs, researchers tend to focus on networking hardware and ignore server computation and client OS interaction for large-scale experiments. We find that not only does OS software affect the system results at scale, but it can be the dominant factor.

Types	*Microbenchmark*	*Trace*	*Application*
Number of Papers	16	3	2

Table 1. *Workload in recent SIGCOMM papers*

Finally, many evaluations make use of existing commercial off-the-shelf switches, with proprietary architectures that have poor documentation of their 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.

2.4 WSC Network Evaluation Needs

As pointed out in [25], the technical challenges of designing WSCs are no less worthy of the expertise of computer systems and networking architects than any other class of machines. Their size alone makes them difficult to experiment with or simulate efficiently, therefore, system designers must develop new techniques to guide design decisions. We believe to evaluate new WSC network architectures using high-performance supercomputing-style interconnects with a rapidly evolving set of application workloads and OS software requires simulations with the following four properties:

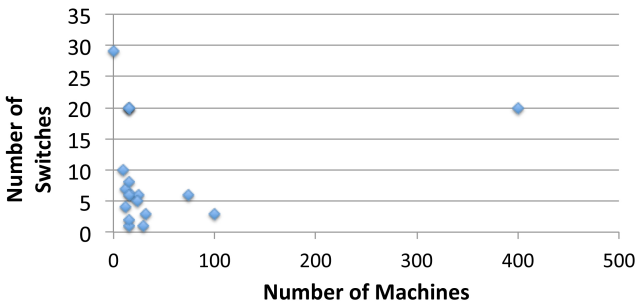


Figure 2. Size of physical testbeds used in recent SIGCOMM papers.

1. *Scale*: WSCs contain O(100,000) servers or more. Although few single apps use all servers in a WSC, O(10,000) nodes are desired to study networking phenomena at aggregate and array-level switches.
2. *Performance*: Current 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. In addition, high-bandwidth switch processors often employ multicore architectures [1].
3. *Accuracy*: A WSC 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 latency. This precision implies many fine-grained synchronizations during simulation if timing models are to be accurate.
4. *Flexibility*: The simulator should support experimentation with radical new switch designs and network software stacks. It is unlikely slight tweaks to existing designs will lead to large improvements.

3. DIABLO Implementation

Although software simulation is a flexible and low-cost technique for systems evaluation, it cannot scale to the size and performance levels required for WSC-scale network experiments while maintaining accuracy. In this section, we describe the design of DIABLO, which exploits FPGAs to provide programmable hardware acceleration of simulation models, achieving two orders of magnitude performance improvement over software simulators. In addition, DIABLO can be scaled using multiple tightly coupled FPGAs to provide scalable high-performance simulations that retain accuracy and flexibility. Overall, we designed DIABLO to answer questions about how new systems behave after changes to the hardware, the software, and/or the scale. We use *target* to refer to the proposed system under evaluation, and *host* to refer to the system running the simulation.

3.1 FAME-Based WSC Array Simulation

The most obvious approach to experimenting with WSC arrays on FPGAs is to build a WSC array *emulator* by mapping processor and switch RTL designs directly to the FPGAs. But this naive mapping is inefficient and inflexible, and makes it impractical to model key WSC array architecture features at scale. DIABLO instead uses FPGAs to accelerate *simulation models* of the various components, an approach known as FPGA Architecture Model Execution (FAME) [55]. Groups in both academia and industry have built various types of FAME simulators, which can be classified in five levels that are analogous to different RAID levels [55]. Higher FAME levels lower simulator cost and improve performance over lower levels, while moving further away from the concrete RTL design of the target. DIABLO is a FAME-7 simulator using the following three key techniques to improve efficiency [55].

1. *Abstracted Models:* To reduce host hardware resource requirements, we employ high-level abstract models of each simulated component, capturing important features but simplifying or removing features that are rarely used. We also separate functional models from timing models to simplify parameterization of target timing. For example, the functional model for a router interprets headers to send packets to the correct output port, while the router's timing model calculates how many target clock cycles this takes. Thus, we can change the timing without altering the router's functional model.
2. *Decoupled Design:* We use a variable number of FPGA host clock cycles and FPGA-resource-friendly structures to simulate a single target clock cycle. For example, a simple ring network on the host FPGA can model an expensive multiport crossbar in the target datacenter switch.
3. *Host Multithreading:* We run multiple target model threads simulating different target instances in a single host hardware model pipeline. Multithreading improves FPGA resource utilization and hides host platform latencies, such as those from host DRAM access and timing-model synchronization across different FPGAs.

3.2 DIABLO Modular Architecture

DIABLO employs a modular design with only two distinct FPGA configurations. Figure 3 shows the high-level simulator architecture for the typical target WSC array configuration presented in Figure 1. We map all server models along with the ToR switch models into *Rack FPGAs*, and array and datacenter switch models to separate *Switch FPGAs*. To further simplify switch model design, we keep any switch model within a single FPGA. Mirroring the physical topology of the target system, we connect Rack FPGAs to Switch FPGAs through several time-shared 2.5 Gbps serial transceivers. Each FPGA has its own simulation scheduler that synchronizes with adjacent FPGAs over the serial links at a fine granularity to satisfy the nanosecond-scale simulation accuracy requirements mentioned in Section 2.

To make the design simpler and more modular, we only use multi-gigabit serial transceivers for inter-FPGA connections. The serial transceivers provide enough bandwidth between FPGAs considering our overall simulation slowdown of between 250–1000× of real time. For example, the bandwidth of a 2.5 Gbps host transceiver translates to 625–2500 Gbps in the target, which far exceeds the bandwidth between a few racks and several array switches today. Moreover, recent FPGAs have significantly enhanced serial transceiver performance, supporting up to 28 Gbps bandwidth [14] in the 28 nm generation.

We reduce host communication latency by using our own protocol over the serial links. Including all packet payload

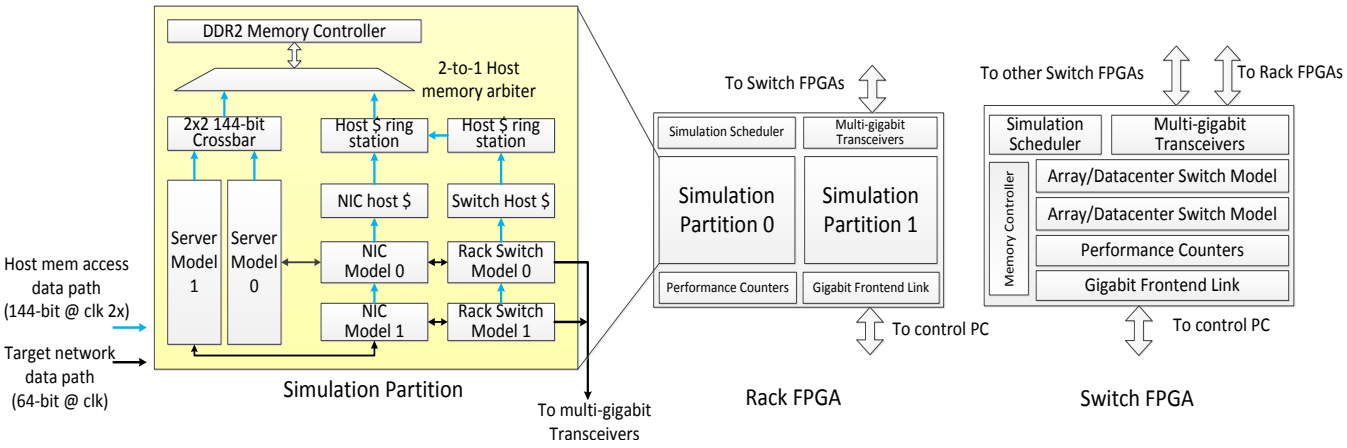


Figure 3. DIABLO FPGA simulator architecture. Only two FPGA configurations are used: Rack FPGA and Switch FPGA.

and overhead, the overall round-trip latency between FAME models on different FPGAs is only around 1.6 μs. In addition, the host-multithreaded FAME-7 design further helps to hide host communication latency, removing model synchronization latency as a simulator performance bottleneck.

3.3 DIABLO FPGA Models

DIABLO contains three FPGA models corresponding to three basic components in WSC array network infrastructure: compute servers, network interface cards, and network switches. To enable design-space exploration without time-consuming FPGA re-synthesis, all the models have runtime-configurable parameters. All the modules were custom-built in SystemVerilog with minimal use of 3rd-party IP blocks.

Server Model

The server model is built on top of RAMP Gold [54], which is an open-source cycle-level full-system FAME-7 architecture simulator supporting the full 32-bit SPARC v8 ISA. RAMP Gold also models MMUs, timers, and interrupt controllers, and boots Linux 2.6.39.3 and 3.5.7 kernels as well as a many-core research OS [45]. The server model is binary compatible with existing SPARC32 machines, and runs unmodified binaries from Debian Linux distributions.

We map one target server running an independent Linux instance to one hardware thread in the host server model pipeline, with each host pipeline in DIABLO simulating a WSC rack holding up to 32 servers. Each simulated server uses a simplified runtime-configurable fixed-CPI timing model, where all instructions take a fixed number of cycles. The goal of the simple server model is not to model WSC server microarchitecture with 100% accuracy but run a full software stack with an approximate performance estimate or bound. More detailed timing models could be implemented, but would require additional host hardware resources and would thus reduce simulation scale.

Even though the RAMP Gold pipeline improves server model performance by two orders-of-magnitude over state-of-the-art software simulators, the server models are still the simulation bottleneck for the whole DIABLO system.

Switch Model

To demonstrate the flexibility of our approach, we build FAME-7 models for two broad categories of WSC array switch: those using *connectionless packet switching*, also known as *datagram switching*, and those using *connection-oriented virtual-circuit switching*.

Although packet switches dominate current WSC networks, some researchers are proposing new circuit-switching designs for WSCs to provide more predictable latencies and to take advantage of new high-speed switching technologies. The proposed designs are only available as early research prototypes but details are open, and some have been directly implemented on FPGAs [59], simplifying construction of highly accurate models. An earlier publication [56] showed that the fully detailed model for a 128-port 10-Gbps high-radix array/datacenter circuit-switching switch can fit on a single FPGA.

In contrast, the packet switches used in existing production WSCs pose a considerable modeling challenge due to their design complexity and their hidden proprietary microarchitecture. To make modeling tractable, we simplify our abstract models by removing features that are seldom used in a WSC. Here are the simplifications we employed and the rationale behind our choice:

Ignore Ethernet QoS-related features: Although QoS features (e.g. support of IEEE 802.1p class of service (CoS)) are available on almost every switch today, many WSCs only utilize switches for basic connectivity without turning on QoS features.

Use simplified source routing: Many switches [11, 12] already support TCAM-based flow tables that have at least 32 K entries. Given the total number of machines in a WSC, the slow-path flow-table update is rarely executed, making the flow-table lookup time constant in practice. Besides, WSC topologies do not change frequently, and

routes can be pre-configured statically. We use source routing to simplify modeling of packet routing, and we note that source routing is actually a component of many WSC-switch research proposals. To emulate more complicated flow-table operations, we could implement d-left hash tables [49] using host DRAM. This technique has already been applied by recent datacenter switches implementing large flow tables [11].

Abstract packet processors: Commercial datacenter switches include many pipelined packet processors that handle different tasks such as MAC address learning, VLAN membership, and so on. The processing time of each stage is relatively constant regardless of packet size, and the time can be as short as a few hundred nanoseconds [31] to a few microseconds [11].

Although commercial switch implementation details are generally not publicly available, the fundamentals of these switch architectures are well known. Examples include the architecture of a virtual-output-queue switch and common scheduling algorithms. We build our abstracted model focusing on these central well-known architectural features, and allow other parts that are unclear or of special interest to researchers (for example, packet buffer layout) to be configurable during simulation. Specifically, in our switch models, we focus on the data path features, such as switch buffer management and configuration, which have become an active area for packet-switch researchers [26]. We base our packet buffer models after that of the Cisco Nexus 5000 switch, with configurable parameters selected according to a Broadcom switch design [42]. We also model high link bandwidth and cut-through switching fabrics with low port-to-port latencies. These are all essential features for exploring future high-performance WSC switches, which are not very easy to deploy at scale for testing in the real world because of high cost.

There has been an increasing research interest in the area of Software Defined Networks (SDNs) using Openflow-capable switches in the WSC. The datapaths of these SDN switches are actually identical to those of conventional switches, and our simplified source-routed switch models can be easily extended to support the flow-table architecture used by an SDN switch.

In DIABLO, we use a unified abstract virtual output-queue switch model with a simple round-robin scheduler for all levels of switch. Switch models in different layers of the network hierarchy differ only in their link latency, bandwidth, and buffer configuration parameters. An earlier publication provides more details of our switch model [56]. Our FAME-7-based switch model is four times faster than a software single-threaded network simulator used at Google [8] that does not simulate packet payloads or support full software-stack scaling to 10,000 nodes as does DIABLO.

NIC Models

The DIABLO NIC models an abstracted Ethernet device, whose internal architecture resembles that of the Intel 8254x Gigabit Ethernet controller on the popular Intel PRO/1000 MT server adapter. It is also popular among many VM implementations such as Virtual Box [20], VMWare Virtual Server [21], QEMU [19], and Microsoft Hyper-V [16]. Figure 4 shows the target architecture of our abstracted NIC model. The core feature of the NIC is a scatter/gather DMA with ring-based packet buffers stored in the main system DRAM. The scatter/gather DMA is used to support the zero-copy feature in Linux, and is essential for any high-performance networking interface. In our current prototype, we support only one hardware ring buffer for each of the receive (RX) and transmit (TX) queues. For model simplicity, we did not model any hardware checksum offloading. Instead, we turn off the packet checksum feature in the Linux kernel to emulate having a hardware checksum offloading engine in the NIC without taking additional CPU time. Our

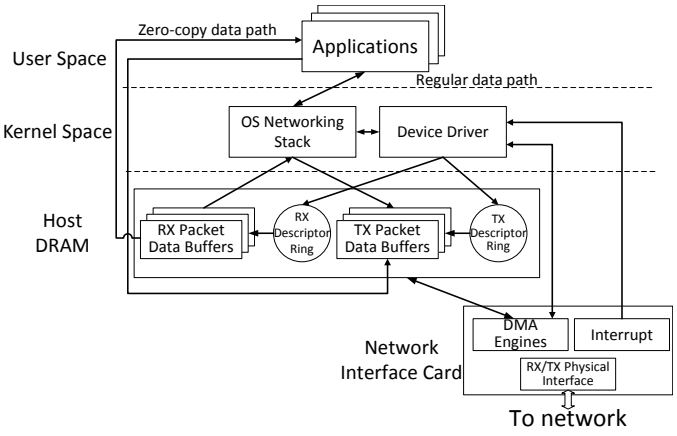


Figure 4. Software and hardware architecture of generic network interface cards

NIC device driver supports all features of a generic Linux Ethernet device driver. We can run unmodified TCP/IP user applications using the standard socket programming interface. To model high performance NICs, our driver supports advanced features such as Zero-copy, RX/TX interrupt mitigation and the NAPI polling interface [52].

The NIC model is a straightforward FAME-7 implementation of the target hardware. The basic functionality of a NIC is to manipulate packet data stored in the DRAM, but DRAM performance is much less an issue for FAME models so the NIC model is not on the simulator critical path.

3.4 DIABLO Cluster Prototype

The DIABLO prototype was built using a rack of BEE3 boards [32]. Each board has four Xilinx Virtex-5 LX155T FPGAs, each with 16 GB of DDR2 memory across two independent DRAM channels, resulting in 64 GB total per board. Each FPGA provides eight 2.5 Gbps SERDES lanes in two CX4 ports to connect to other FPGAs.

On each Rack FPGA of the current BEE3 boards, we evenly distribute four 32-thread server models to two host DRAM channels and partition the host DRAM for server computations, simulating four racks with 124 servers. Each server model has one ToR switch model attached to the same DRAM controller.

Component Name	*LUT*	*Register*	*BRAM*	*LUTRAM*
Server Models	28,445	37,463	96	6,584
NIC Models	9,467	4,785	10	752
Rack Switch Models	4,511	3,482	52	345
Miscellaneous	3,395	16,052	31	5,058
Total	45,818	62,811	189	12,739

Table 2. *Rack FPGA resource utilization on Xilinx Virtex-5 LX155T after place and route using Xilinx ISE 14.3.*

Table 2 shows the overall resource utilization of the Rack FPGA on the BEE3 board. Including the resources dedicated for FPGA routing, the device is almost fully utilized with 95% of logic slices occupied at a 90 MHz host clock rate. The chip resources are dominated by structures to simulate server computation. On each FPGA, we equally divide the 16 GB physical DRAM into 128×128 MB partitions. Each simulated server uses one partition for target memory storage. For each server pipeline, we use 31 threads out of the 32 available threads and save the DRAM storage of the remaining thread for simulating packet buffers on the ToR switch. Each DIABLO model has a dedicated on-chip host cache to optimize simulator DRAM accesses.

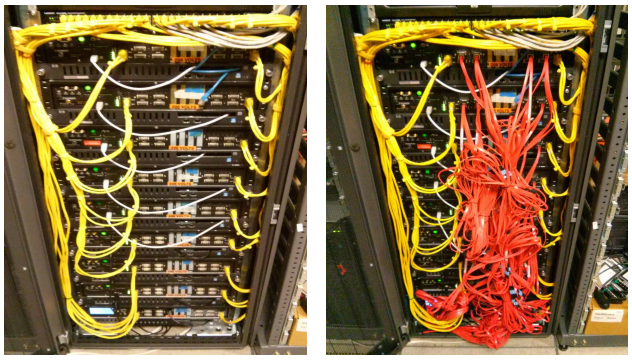
All DIABLO models on the same FPGA share a single Gigabit Ethernet connection as the frontend connection for bootstrapping, console output and functional I/O, such as keyboard and disk access.

The Switch FPGA is just a cut-down version of the Rack FPGA containing only switch models and fewer server model pipelines. A single server functional model pipeline, without a timing model, is used to run functional configuration for our switch models. The server functional model can also act as a control-plane processor for a simulated switch.

We emphasized reliability over performance in physical design. We protect all physical links, such as Gigabit Ethernet and serial links, with hardware checksums and retries to handle occasional soft errors, which we've observed occur a few times per day in our prototype. We also use ECC to protect all memory data paths that interact with the ECC memory DIMMs on our FPGA board.

A 3,000-node DIABLO system

We used 9 BEE3 boards totalling 36 FPGAs to build a sizable prototype, as Figure 5 shows. We use off-the-shelf CX4 breakout cables to connect arbitrary transceiver pairs on different FPGAs from the front panel following the target network topology. We populate six boards with the Rack-FPGA design simulating 2,976 servers with 96 rack switches. We load the remaining three boards with the Switch-FPGA design to simulate 6 array switches and one datacenter switch. Using an additional 13 boards, we could scale the existing system to build an emulated large WSC array with 11,904 servers and 385 rack switches.



(a) Without inter-board connections (b) Fully-connected with high-speed cables

Figure 5. DIABLO cluster prototype

The whole FPGA cluster has a total memory capacity of 576 GB in 72 independent DRAM channels with an aggregated bandwidth of 268 GB/s. All FPGA boards fit into a standard server rack, and the prototype consumes about 1.5 kW when active. Each FPGA has a Gigabit frontend control link connected to a shared 48-port gigabit switch. We use two 12-core Intel Xeon X5680 servers to drive all nine FPGA boards using our custom protocol over Ethernet. Each server has four 1 Gbps interface card connected to the 48-port switch to balance the control traffic to different boards. We store boot disk images and console I/O logs of all 2,976 servers on a striped RAID-0 partition on each front-end server. This setup is used to optimize the simulated Linux booting process, where all simulated servers access disk images over 8×1 Gbps links simultaneously.

The BEE3 boards used have 2007-era FPGAs and older DRAM DIMMs, and the BEE3 board was not designed with low-cost simulations in mind. Each BEE3 cost $15K, and the total cost of a 9-board system was about $140K. Although the BEE3 hardware is now somewhat dated, the system was sufficient to develop and evaluate the approach and gather experience for the next system. Because DIABLO uses a modular design and uses only the serial links between FPGAs, we can readily port DIABLO to lower-cost single-FPGA boards. Using the latest 20nm FPGAs in 2015 and with a redesigned board, we estimate we could now potentially build a 32,000-node DIABLO system using just 32 FPGAs and an overall cost of $150K including DRAM.

4. Case Studies

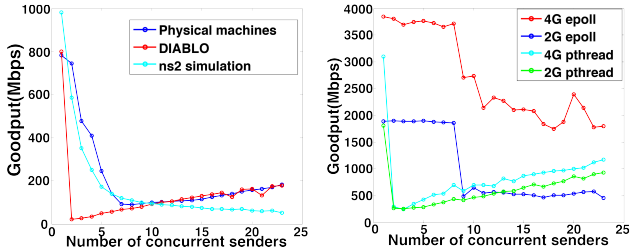
In this section, we demonstrate the capabilities of the DIABLO prototype and validate its performance models using case studies drawn from classic networking problems run at scale. We show the effects of changing both hardware and software parameters of the system, including switch band-

width, latency, buffer configuration, target server CPU performance, operating systems, and application logic.

4.1 Reproducing TCP Incast

Incast is a classic many-to-one communication pattern commonly found in many WSCs implementing scale-out distributed storage and computing frameworks, such as Hadoop and Map-Reduce. In particular, the TCP Incast problem [53, 60] refers to the application-level throughput collapse that occurs as the number of servers returning data to a client increases past the ability of an Ethernet switch to buffer packets. Previous work [28, 60] has focused on studying the interplay between the TCP retransmission timeout (RTO) value and the limited capacity of buffers in low-cost ToR switches. However, as pointed out in [53], TCP Incast is only obvious under specific hardware and software setups, for instance with switches that have very small shared packet buffers.

Our first experiment uses DIABLO to reproduce the application throughput collapse observed with shallow-buffer Gigabit switches, as used in many previous studies. We configured the DIABLO packet-switching model for 1 Gbps links, switch port-to-port delay of 1 μs, and 4 KB packet buffers per port, as found in a Nortel 5500 switch [29]. Compared to existing *ns2* [4] and analytical switch models, DIABLO models a more advanced packet buffer architecture that supports virtual queues to prevent head-of-line blocking, as usually used in high-end Cisco [11] and Fulcrum switches [31]. The test program [3] we ran has been used to test 10 Gbps production datacenter switches from Arista and Cisco [9]. We selected a typical request block size of 256 KB for the client application. We configured our server models to simulate a single-core 4 GHz CPU and ran Linux 2.6.39.3 on our simulated servers. We used up to 24 ports of the simulated switch and ran the network transaction for 40 iterations.



(a) 1 Gbps shallow-buffer switch (b) 10 Gbps simulated switch

Figure 6. Reproducing the *goodput* of TCP Incast under different configurations

We compare DIABLO results against an *ns2* simulation configured to match these parameters, and validate our results against a real hardware cluster containing 16 Intel Xeon D7950 servers and a 16-port Asante IntraCore 35516-T switch, where the commercial switches have a shared packet buffer architecture. Figure 6(a) compares the average *goodput* of the three systems. With its abstract virtual-output-queue switch model, DIABLO has a faster application throughput collapse than measured on the hardware with a shared-buffer switch. However, DIABLO captures the throughput recovery trend after the collapse better than *ns2*. The simulated throughput before collapse also matches that on the real system at around 800 Mbps. Note that many details are different between the DIABLO system and the real hardware, including instruction set, processor performance, and NIC and switch architecture, yet DIABLO captures the overall shape of the performance curve at the larger scale.

Traditional network simulators like *ns2* focus on network protocols but not the implementation of the OS network stack and application interface, which become more important for high-speed networks. To investigate the impact of target OS and application interface, we modified the TCP Incast benchmark to use the *epoll* syscall instead of the original *pthread* with blocking socket syscalls. The *epoll* syscall has been used by many WSC applications, such as *memcached*, to efficiently handle many client network connections. Applications using *epoll* proactively poll the kernel for available data. This behavior is different from that of using blocking syscalls to wait for OS notifications from multiple user threads. We also configured the DIABLO server timing model to simulate target computations of a 2 GHz CPU versus a 4 GHz CPU.

Figure 6(b) plots *goodput* curves of experiments under different server hardware and software configurations using a 10 Gbps network. With the same switch and TCP configuration, the results show that CPU speed and choice of OS syscalls significantly affects the application throughput. As pointed out in early work on 10 Gbps TCP performance at the Fermi High Energy Physics Lab [62], there are many other factors in the Linux kernel that could contribute to packet losses, such as different queues for slow and fast path handling of received packets. With 10 Gbps links, it is challenging for applications to sustain wire speed without significant tuning. Even when using scatter/gather zero-copy features in the NIC, the simulated single-core 2 GHz CPU using *pthread* could only achieve 1.8 Gbps throughput when there is no throughput collapse. Using *epoll* significantly delays the onset of throughput collapse. In addition, we only observed a moderate throughput collapse to 2.7 Gbps starting from 9 servers and 1.8 Gbps with 23 servers using 4 GHz CPUs. Using 2 GHz CPUs, the collapse is much more significant at 400–500 Mbps, which is less than a third of the throughput with 4 GHz CPUs. The collapsed throughput with the *epoll* client is only half of that of the original *pthread* client. However, using the *pthread* version, the throughput collapses quickly even with a faster CPU. The throughput recovers to only 10% of the link capacity, which again matches the observations from measurements and simulations in [60]. Moreover, the absolute throughput numbers do not seem to be correlated with the target processor performance when collapse happens.

In conclusion, switch buffers are not always the one and only limiting factor for TCP Incast. The long TCP retransmission timeout is just a consequence of an imbalanced system. Simple analytical or network simulation models might allow parameters to be set to match results for a small number of faster machines connected by a slower network, but then the models will not correctly predict behavior of a scaled system.

4.2 Studying memcached Latency Long Tail at Scale

One of the greatest challenges in building responsive large-scale computer systems is the nature of request service variations. For example, Google researchers found long-tail distributions for web query latencies, where some request latencies are multiple orders of magnitude longer than the median [24, 33]. Moreover, more requests fall into the tail as the system scale increases. There are multiple complex underlying causes of the long tail, including queuing in both software and hardware, kernel schedulers, shared hardware resources and many others [33]. Hence, reproducing the long tail and understand the contributing factors requires analyzing hardware along with the full software stack at large scale. In this section, we use DIABLO to illustrate the impact on tail latency of many factors including: implementation of the application, versions of the OS kernel, network transport protocols, as well as switch design.

Experiment setup

To demonstrate running production WSC software, we ran the popular *memcached* [2, 5, 17] distributed key-value store application on DIABLO. We used unmodified *memcached* source code versions 1.4.15 and 1.4.17.

In a real production environment, the client workloads of *memcached* servers are generated by the front-end web tier. Although the API is simple, it is difficult to accurately represent a *memcached* workload. In particular, previous studies show that the object size distribution has a large impact on the system behavior [44]. Simple microbenchmark tools like *memslap* do not attempt to reproduce the statistical characteristics of real traffic. To provide a more realistic workload, we built our own client based on recently published Facebook live traffic statistics [23]. At Facebook, *memcached* servers are partitioned based on the concept of *pools*. We focused on one of the pools that is the most representative, and validated our workload generator against the Facebook data [23].

Figure 7 shows the simulated target topology. Each rack contains 31 servers with one Top-of-Rack switch. We used the 32nd port on the rack switch to connect to an array switch, creating a bandwidth over-subscription ratio of 31 to 1. Each array switch supports up to 16 inward-facing links and one uplink to the datacenter switch, thus having a bandwidth over-subscription ratio of 16-to-1. With the runtime configurable switch timing models, we are able to simulate a 1 Gbps interconnect with 1 μs port-to-port switch latency as well as a 10 Gbps interconnect with 100 ns port-to-port switch latency. We used the same simulated switch buffer configuration as in section 4.1, and used the simple 4 GHz fixed-CPI timing model for server CPUs. This simulation required six BEE3 boards: four boards to simulate 64 server racks of 1,984 servers, one board to simulate all array switches, and the remaining board for the datacenter switch simulation.

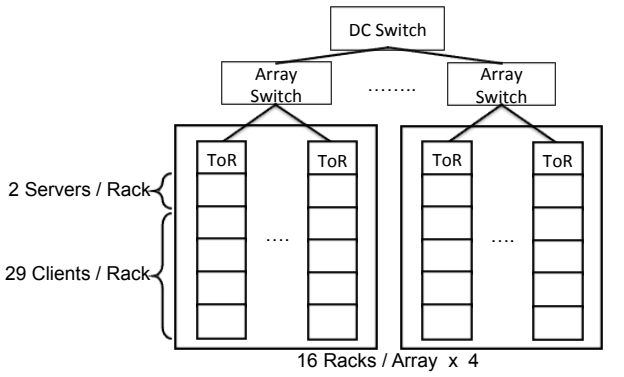


Figure 7. Memcached experimental setup.

We distributed 128 *memcached* servers evenly across all 64 racks to minimize potential hot spots in the network, and used the remaining machines as clients. This creates a configuration of two *memcached* servers and 29 clients in a single rack. Each client sends 30K requests to a randomly selected server from all 128 servers, each configured with a 64 MB memory pool.

We performed experiments at several scales: 496-node, 992-node, and 1984-node. For convenience when presenting results, we round up the exact number of nodes to 500, 1000 and 2000 respectively in the rest of this paper. We scale down the number of servers when running a smaller configuration, maintaining a constant server-to-client ratio. The 500-node setup uses only one 16-port array switch without a datacenter switch, while both 1000-node and 2000-node experiments exercise all three levels of switches. We also performed our load tests using both TCP and UDP protocols. CPU utilization in all servers is moderate, at under 50%. There is also no packet retransmission due to switch buffer overruns.

Validating *memcached* on real clusters and Reproducing the latency long tail on DIABLO

First, to validate our DIABLO models at an understandable scale, we deployed a set of *memcached* experiments at the single-rack scale using a real cluster. Our single-rack 16-node testbed includes 3 GHz Intel Xeon D7950 servers running Linux 2.6.34 connected with a 16-port Asante IntraCore 35516-T Gigabit switch. We used two machines as *memcached* servers with the rest as clients. We let each client thread send 30,000 requests till completion. We also tried 100,000 requests with up to 256 MB server memory pool, and the steady-state performance numbers were similar. We configured each server with several parameter com-

binations, for example 4 or 8 worker threads using TCP or UDP connections.

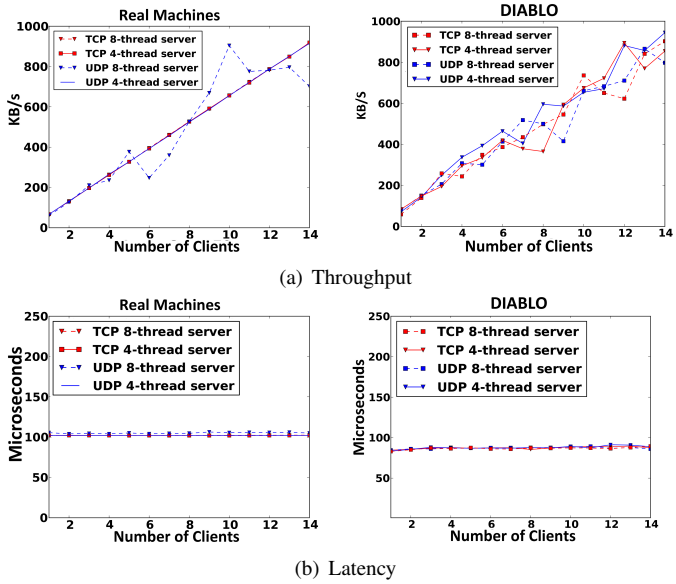


Figure 8. Real machines vs. simulated memcached servers.

We compared results from the perspective of both server and client. Figure 8(a) shows the throughput of a measured server versus a simulated server under different number of clients. We only show results from one server, as throughput curves of other servers are similar. Figure 8(b) illustrates the average request latencies measured at every client. For all configurations we tested at this small scale, DIABLO successfully reproduced the shapes of performance curves. The client latency stays low and scales linearly with a small number of clients, while more clients saturate the server. We expect absolute performance differences between real and simulated clusters, as DIABLO uses a different simulated hardware specification. However, the goal is to reproduce the trend at scale.

As with prior studies using real physical testbeds, our validation efforts are limited by available networking and computing hardware. To perform scale-out validations, we ran the same experiments on an 8-rack 120-node Intel Xeon E5-2620 v2 cluster with 10 Gbps rack switches and a 40 Gbps aggregate switch [6]. As we could not obtain dedicated access to this large shared cluster, we had to run the same experiments many times and choose 10 runs without a significant background workload in the cluster. Figure 9 compares the client request latency tail measured from the large physical cluster with the simulation results from DIABLO. To show the robustness of our data from the shared physical cluster, we plotted the error bars in Figure 9. At the scale of 120 machines, we do see $< 0.1\%$ of total requests finish orders of magnitude slower than regular requests. The newer memcached 1.14.17 has a slightly better tail performance than the older 1.14.15 on both simulated and real cluster. The simulated 120-node setup is a more ideal environment with less software services running in the background. Therefore, there are fewer requests falling into the tail compared to a real system. Similarly, the difference between the two versions of memcached becomes less obvious.

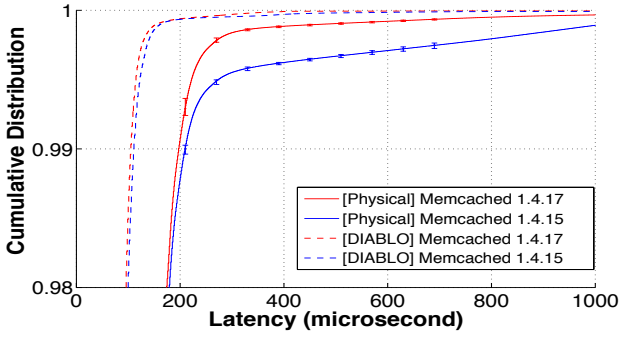


Figure 9. Client latency CDF on a 120-node real cluster vs. DIABLO

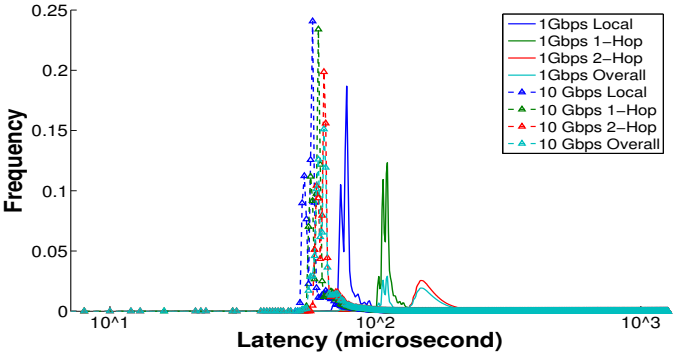


Figure 10. PMF of client request latency at 2000-node on DIABLO using UDP.

Figure 10 plots the Probability Mass Function (PMF) of all client queries for the 2k-node DIABLO simulation running UDP over different interconnects. The shape of PMF is similar at the scale of 500-node and 1k-node. We also observed similar latency long tails using TCP. We found the majority of requests finished in less than $100\mu s$, but there are a small number of requests that finished more than two orders of magnitude slower than the average, reproducing the long tail distribution. To understand the long tail better, we classified all queries into three different categories based on the number of physical switches they traverse, with *local* indicating requests hit the same rack. From Figure 10, we know that all three types of requests exhibit a long tail distribution. The more switches the query traverses, the greater the latency variation. Moreover, 2-hop requests dominate the overall latency distribution at large scale, as the majority of requests travel to a remote rack.

From Figure 11, we can also tell the impact of system scale on the latency tail. For example, the 99-percentile latency of a 2K-node system is more than an order-of-magnitude worse than that of a 500-node system. These observations from DIABLO match those reported in a recent Google publication [33].

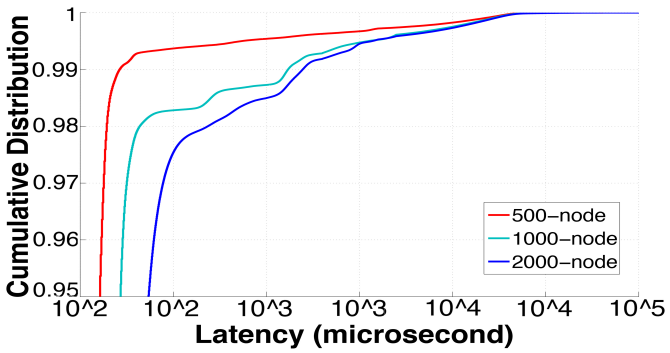


Figure 11. $95^{th} - 100^{th}$ percentile CDF of client latency at different scales on a 1 Gbps interconnect running UDP.

Impact of network hardware

One straightforward solution to reduce request latency is to upgrade the network hardware, and we next explore the potential performance gain via DIABLO simulations. We simulate a low-latency 10 Gbps switch with 10× bandwidth and 10× shorter latency compared to the simulated 1 Gbps network. From our results, the improved cut-through switch does help to reduce request latencies, but the improvement is no more than 2×. This indicates that the full OS networking stack dominates the request latency. Google showed similar results from their large-scale testbed using O(100 ns) low-latency switches [24].

Further, to evaluate the robustness of our simulation results, we simulated the client latency tail with an additional 50 ns and 100 ns port-to-port latency at all switch levels, with results in Figure 12. The error bars on the figure are very small showing that the cycle-level DIABLO simulator is quite stable when tweaking simulated hardware configurations. We found that the extra switch latency does not affect the shape of the tail curves, although the 99-percentile latency increases to 364 μs from 253 μs. In addition, the extra switch latency does not impose a significant tax on regular non-tail request latencies.

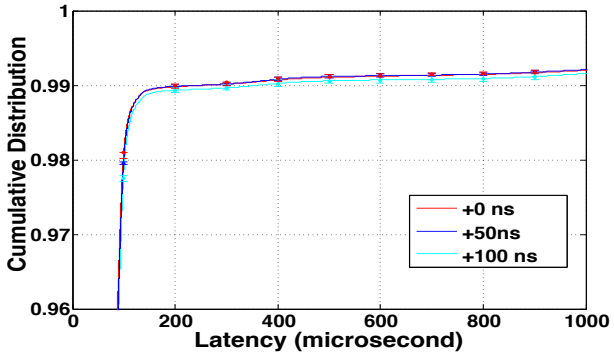


Figure 12. Client latency tail with different switch latencies.

Impact of the network transport protocol at scale

As mentioned earlier, current evaluation technologies only allow most networking researchers to conduct full-system experiments at scales of O(10) to O(100). Often the most intuitive solution to a networking issue is to change the network transport protocol. However, one big question is whether we can generalize O(10)–O(100) node results to a larger scale at O(1000)–O(10,000). In this section, we conduct a simple experiment using DIABLO to quantitatively analyze which network protocol (TCP or UDP) is better at reducing the long-tail at scale, by sweeping through several system parameters including interconnect performance and system scale.

Figure 13 shows the cumulative tail distribution of using different protocols. For the 1 Gbps interconnect, at the 500-node scale the UDP protocol is a clear winner compared to TCP. However, UDP's advantage disappears when moving to 1000 nodes, as TCP slightly outperforms UDP. When we move to the 2000-node scale, TCP is better. The conclusion at 2000-node scale is completely reversed to that at 500 nodes. On the other hand, upgrading the interconnect to 10 Gbps shows a very different result. Now, there is much less difference between UDP and TCP.

The 500-node setup represents a small cluster in a full-scale datacenter, with 13–14 fully populated standard server racks connected to a single aggregate array switch. For the given topology we simulated, both 1000-node and 2000-node setups require an extra aggregate switch. The extra aggregate switch contributes to the tail latency more than using more ports (2000 vs. 1000) on the same extra switch.

Impact of target operating system

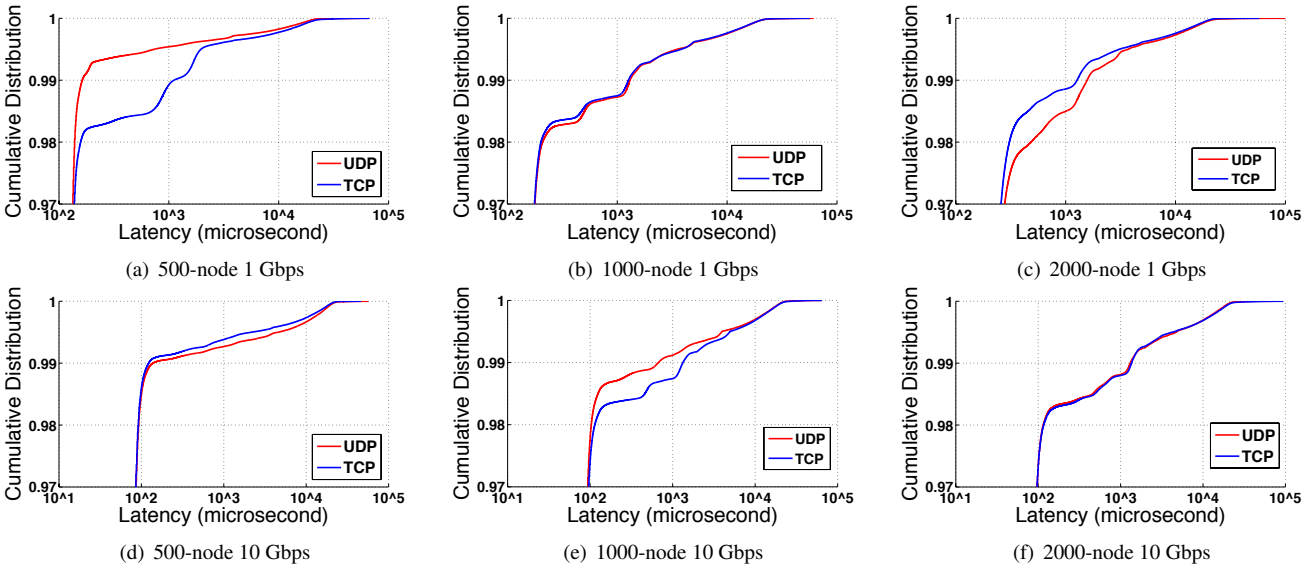
DIABLO is different from prior simulators in that it can run 1,000s of full OS instances while modeling time accurately. This enables experiments that are difficult even in a real WSC, e.g., deploying an experimental kernel on thousands of machines.

We ported a more recent Linux 3.5.7 kernel and compared its performance running our large-scale *memcached* experiments against those from the one-year-older 2.6.39.3 kernel used earlier in this section. We compare them at 2K nodes with the same 10 Gbps interconnect and server hardware. Figure 14 plots the cumulative distribution of 95-percentile client request latencies. We can see that there are significant improvements in terms of request responsiveness in the new 3.5.7 kernel. The average request latency is almost halved, and the better kernel scheduler and more efficient networking stack also helps to alleviate the latency long-tail issue.

This result shows that OS optimizations play a critical role in the performance of distributed applications, and OS behaviour should be modeled when evaluating any new proposed new network designs.

Impact of application implementation

In addition to OS network stack design, another factor that affects client request latency is the application's implementation and choice of system calls. We ran two versions of memcached, 1.4.17 and 1.4.15 on both the physical testbed



(a) 500-node 1 Gbps (b) 1000-node 1 Gbps (c) 2000-node 1 Gbps

(d) 500-node 10 Gbps (e) 1000-node 10 Gbps (f) 2000-node 10 Gbps

Figure 13. Comparing TCP vs UDP on CDFs of client request latency at different scale with different interconnect.

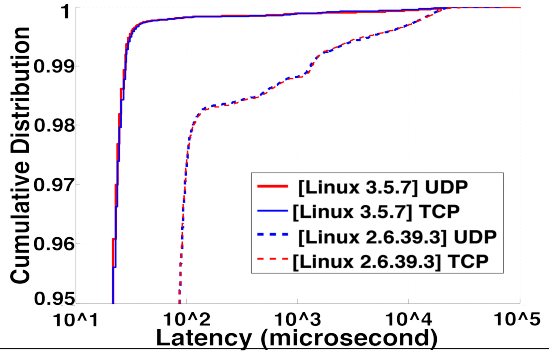


Figure 14. Impact of OS kernel versions on the 2,000-node system.

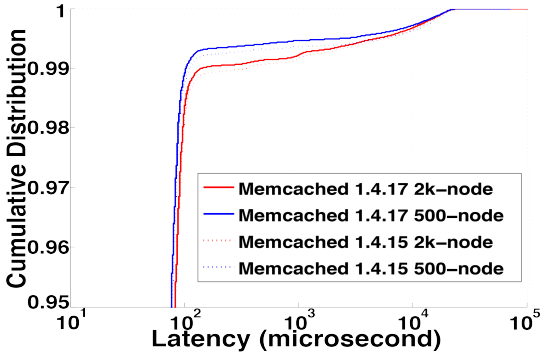


Figure 15. Impact of memcachd versions on the latency CDF. 1.4.17 has a slightly better tail performance similar to our validation results in Figure 9.

and DIABLO. One of the major differences for the slight newer 1.4.17 is the support of the *accept4* syscall, which eliminates one extra syscall for each new TCP connection [22].

For both versions, there is no significant difference between the client request latencies on the real cluster at the scale of 120 nodes. The 1.4.17 memcached has a marginal better 99 percentile latency of 197 μs, while the older 1.4.15 is at 210 μs.

To investigate the effectiveness of the *accept4* syscall at a larger scale, we simulate a configuration with 500 nodes and another configuration with 2,000 nodes on DIABLO. Figure 15 shows the CDF of client latency of both configurations. At the 500-node scale, both versions of memcached on the simulated cluster behave similarly to the 120-node real cluster. The 99-percentile client request latency of 1.4.17 memcached is only 8 μs faster on the simulated hardware. Note that the absolute latency difference between the simulated hardware and real cluster is due to the fact that we configure our abstract model with a slightly faster switch.

On the other hand, when scaling to the 2,000-node scale with one more aggregate switch in the network hierarchy, the benefit of fewer syscalls in the newer version of memcached becomes more apparent. The 99-percentile client request latency of version 1.4.17 is 145 μs versus 345 μs of version 1.4.15. This again shows how the scale of the system amplifies the latency tail effect.

To summarize, in our case studies, the impact of software implementation of OS and application interface are far more important than all the hardware and network transport protocol changes we have studied. The detailed execution-driven DIABLO models show that there are many more aspects in both HW and SW stacks that could significantly affect the application tail latency at scale.

5. Discussion and Future Work

When simulating 4 GHz servers with a 10 Gbps interconnect, around 50 minutes of simulation wall-clock time are required for one second of target time; software simulation would take almost two weeks. Moreover, DIABLO's simulation performance scales perfectly. We observed no performance drop from simulating 500 nodes with two boards, to 2,000 nodes using six boards. The simulation performance is dominated by simulating the server computation using RAMP Gold, which can be improved by reducing the number of model threads per host hardware pipeline at the cost of reduced system scale or increased hardware cost [54].

The FAME-7 modeling technique gives us the capability of emulating larger target systems with fewer virtualized FPGA resources. However, the target memory capacity is very hard to virtualize with limited physical DRAM storage. We should note that this is an open topic for any work that is trying to emulate a WSC with limited hardware resources. Another limitation is that we have only simulated fixed-CPI single-CPU servers with one hardware thread per server model. A more complex timing model supporting multi-core CPUs is planned for DIABLO-2.

Neither limitation is intrinsic to the methodology, but instead reflects our use of a 5-year-old FPGA board. We are developing a new FPGA board using upcoming 20 nm FPGAs that should support a quad-core 64-bit server model with 4 GB/node memory capacity. Additional simulated DRAM storage will be provided through PCIe-attached NAND FLASH storage.

Our current prototype used FAME-7 style NIC/switch models, which run far faster than necessary for the simulation while consuming considerable FPGA resources. We will replace these hardware models with a software-programmable microcoded model, to provide additional flexibility as well as reduced resource usage.

6. Conclusion

We believe the research community needs a boost in evaluation technology to look at systems with scale of O(1,000) to O(10,000), as behavior is drastically different compared to O(10) to O(100) systems. The working DIABLO prototype enables networking researchers to conduct controllable, reproducible, full-stack WSC simulations at much larger scales than previously possible. We show that considering end-to-end server computation, including full OS and application code, is essential to understanding network system performance, and has much greater impact than networking hardware and protocols.

Acknowledgments

This research is supported in part by gifts from Oracle, 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), AFOSR under MURI award FA9550-09-1-0539 (Air Force Office of Sponsored Research in MURI program), and the University of California Industry/University Cooperative Research Program (UC Discovery) grant COM07-10240. The testbed for validation work was supported in part by the by the National Basic Research Program of China Grant 2011CBA00300, 2011CBA00301, the National Natural Science Foundation of China Grant 61033001, 61361136003. We thank the anonymous reviewers for their many insightful comments and suggestions.

References

[1] XGS Core Switch Series - BCM88030 Series. http://www.broadcom.com/products/Switching/Carrier-and-Service-Provider/BCM88030-Series.

[2] Facebook Memcached. https://github.com/amanuel/facebook-memcached.

[3] R2D2: RAPID AND RELIABLE DATA DELIVERY IN DATA CENTERS. http://www.stanford.edu/~atikoglu/r2d2/.

[4] Network simulator, ns-2 : http://www.isi.edu/nsnam/ns/.

[5] Twemcache. https://twitter.com/twemcache.

[6] Tsinghua IIIS test cluster. http://wiki.iiis.systems/w/index.php/Cluster.

[7] Yahoo! Reaches for the Stars with M45 Supercomputing Project. http://research.yahoo.com/node/1884, 2007.

[8] Glen Anderson, private communications, 2009.

[9] Switching Architectures for Cloud Network Designs. http://www.aristanetworks.com/en/SwitchingArchitecture_wp.pdf, 2009.

[10] Sun Datacenter InfiniBand Switch 648. http://www.sun.com/products/networking/infiniband.jsp, 2009.

[11] Cisco Nexus 5000 Series Architecture: The Building Blocks of the Unified Fabric. http://www.cisco.com/en/US/prod/collateral/switches/ps9441/ps9670/white_paper_c11-462176.html, 2010.

[12] Force10 S60 High-Performance 1/10 GbE Access Switch. http://www.force10networks.com/products/s60.asp, 2010.

[13] Hadoop. http://hadoop.apache.org/, 2010.

[14] Xilinx Virtex 7 Series FPGAs. http://www.xilinx.com/technology/roadmap/7-series-fpgas.htm, 2010.

[15] Google G-Scale Network. http://www.eetimes.com/electronics-news/4371179/Google-describes-its-OpenFlow-network, 2012.

[16] Microsoft Hyper-V virtualization platform. http://www.microsoft.com/en-us/server-cloud/windows-server/server-virtualization.aspx, 2012.

[17] memcached: a distributed memory objec caching system. http://memcached.org, 2012.

[18] Open Compute Project. http://opencompute.org, 2012.

[19] QEMU open source processor emulator. http://wiki.qemu.org, 2012.

[20] Oracle Virtualbox VM. http://www.virtualbox.org/, 2012.

[21] VMware Virtual Server. http://www.vmware.com, 2012.

[22] Memcache 1.14.17 release notes. https://code.google.com/p/memcached/wiki/ReleaseNotes1417, 2013.

[23] B. Atikoglu, Y. Xu, E. Frachtenberg, S. Jiang, and M. Paleczny. Workload analysis of a large-scale key-value store. In *Proceedings of the 12th ACM SIGMETRICS/PERFORMANCE joint international conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '12, pages 53–64, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1097-0. . URL http://doi.acm.org/10.1145/2254756.2254766.

[24] L. A. Barroso. Warehouse-scale computing: Entering the teenage decade. In *Proceedings of the 38th annual international symposium on Computer architecture*, ISCA '11, pages –, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0472-6.

[25] L. A. Barroso and U. Hölzle. *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*. Synthesis Lectures on Computer Architecture. Morgan & Claypool Publishers, 2009.

[26] A. Bechtolsheim. Moore's Law and Networking. In *The Linley Group Processor Conference*, San Jose, CA, USA, 2012.

[27] 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. ISSN 0272-1732.

[28] 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. ISBN 978-1-60558-443-0.

[29] Y. Chen, R. Griffit, D. Zats, and R. H. Katz. Understanding tcp incast and its implications for big data workloads. Technical Report UCB/EECS-2012-40, EECS Department, University of California, Berkeley, Apr 2012. URL http://www.eecs.berkeley.edu/Pubs/TechRpts/2012/EECS-2012-40.html.

[30] M. Chowdhury, M. Zaharia, J. Ma, M. I. Jordan, and I. Stoica. Managing data transfers in computer clusters with orchestra. In *SIGCOMM*, pages 98–109, 2011.

[31] U. Cummings, D. Daly, R. Collins, V. Agarwal, F. Petrini, M. Perrone, and D. Pasetto. Fulcrum's FocalPoint FM4000: A Scalable, Low-Latency 10GigE Switch for High-Performance Data Centers. In *Proceedings of the 2009 17th IEEE Symposium on High Performance Interconnects*, pages 42–51, Washington, DC, USA, 2009. IEEE Computer Society. ISBN 978-0-7695-3847-1. . URL http://portal.acm.org/citation.cfm?id=1633800.1634467.

[32] J. Davis, C. Thacker, and C. Chang. BEE3: Revitalizing Computer Architecture Research. Technical Report MSR-TR-2009-45, Microsoft Research, Apr 2009. URL http://research.microsoft.com/apps/pubs/default.aspx?id=80369.

[33] J. Dean and L. A. Barroso. The tail at scale. *Communications of the ACM*, 56(2):74–80, 2013.

[34] A. Ganesan, D. Lee, A. Leinwand, A. Shaikh, and M. Shaw. What is the impact of cloud computing on the data center interconnect? In *Hot Interconnects*, 2011.

[35] 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. ISSN 0146-4833.

[36] 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*, pages 51–62, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-594-9.

[37] 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*, pages 63–74, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-594-9.

[38] J. L. Hennessy and D. A. Patterson. *Computer architecture: a quantitative approach*. Elsevier, 2012.

[39] L. R. Hsu, A. G. Saidi, N. L. Binkert, and S. K. Reinhardt. Sampling and stability in tcp/ip workloads. In *Proceedings of the First Annual Workshop on Modeling, Benchmarking, and Simulation*, MoBS '05, pages 68–77, 2005.

[40] D. A. Joseph, A. Tavakoli, and I. Stoica. A policy-aware switching layer for data centers. In *SIGCOMM '08*, pages 51–62, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-175-0.

[41] R. Katz. Tech titans building boom: The architecture of internet datacenters. *IEEE Spectrum*, February 2009.

[42] B. Kwan, P. Agarwal, and L. Ashvin. Flexible buffer allocation entities for traffic aggregate containment. US Patent 20090207848, August 2009.

[43] J. Li, N. K. Sharma, D. R. K. Ports, and S. D. Gribble. Tales of the tail: Hardware, OS, and application-level sources of tail latency. In *Proceedings of the ACM Symposium on Cloud Computing (SoCC)*, Seattle, WA, USA, 11 2014. ACM. URL papers/latency-socc14.pdf.

[44] K. Lim, D. Meisner, A. Saidi, P. Ranganathan, and T. F. Wenisch. Thin Servers with Smart Pipes: Designing SoC Accelerators for Memcached. In *Proceedings of the 40th annual international symposium on Computer architecture*, ISCA '13, 2013.

[45] R. Liu et al. Tessellation: Space-Time partitioning in a manycore client OS. In *HotPar09*, Berkeley, CA, 03/2009 2009. URL http://www.usenix.org/event/hotpar09/tech/.

[46] P. S. Magnusson et al. Simics: A Full System Simulation Platform. *IEEE Computer*, 35, 2002.

[47] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74, Mar. 2008. ISSN 0146-4833. . URL `http://doi.acm.org/10.1145/1355734.1355746`.

[48] D. Meisner, J. Wu, and T. F. Wenisch. BigHouse: A Simulation Infrastructure for Data Center Systems. *ISPASS '12: International Symposium on Performance Analysis of Systems and Software*, April 2012.

[49] M. Mitzenmacher and A. Broder. Using multiple hash functions to improve ip lookups. In *In Proceedings of IEEE INFOCOM*, pages 1454–1463, 2000.

[50] 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*, pages 39–50, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-594-9.

[51] D. Ongaro, S. M. Rumble, R. Stutsman, J. K. Ousterhout, and M. Rosenblum. Fast crash recovery in ramcloud. In *SOSP*, pages 29–41, 2011.

[52] J. H. Salim, R. Olsson, and A. Kuznetsov. Beyond softnet. In *Proceedings of the 5th annual Linux Showcase & Conference - Volume 5*, ALS '01, pages 18–18, Berkeley, CA, USA, 2001. USENIX Association. URL `http://dl.acm.org/citation.cfm?id=1268488.1268506`.

[53] H. Shah. *Solving TCP Incast in Cluster Storage Systems*. Technical report (Information Networking Institute). Carnegie Mellon University. Information Networking Institute, 2009. URL `http://books.google.com/books?id=Rl2jYgEACAAJ`.

[54] Z. Tan, A. Waterman, R. Avizienis, Y. Lee, H. Cook, D. Patterson, and K. Asanović and. RAMP gold: An FPGA-based architecture simulator for multiprocessors. In *Design Automation Conference (DAC), 2010 47th ACM/IEEE*, pages 463 – 468, June 2010.

[55] Z. Tan, A. Waterman, H. Cook, S. Bird, K. Asanović, and D. Patterson. A case for FAME: FPGA architecture model execution. In *Proceedings of the 37th annual international symposium on Computer architecture*, ISCA '10, pages 290–301, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0053-7.

[56] Z. Tan, K. Asanovic, and D. Patterson. Datacenter-scale network research on fpgas. In *Proc. Workshop on Exascale Evaluation and Research Techniques*, 2011.

[57] A. Tavakoli, M. Casado, T. Koponen, and S. Shenker. Applying NOX to the datacenter. In *HotNets*, 2009.

[58] C. Thacker. Rethinking data centers. October 2007.

[59] C. Thacker. A data center network using FPGAs, May 2010.

[60] 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*, pages 303–314, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-594-9.

[61] G. Wang, D. G. Andersen, M. Kaminsky, K. Papagiannaki, T. S. E. Ng, M. Kozuch, and M. P. Ryan. c-through: part-time optics in data centers. In *SIGCOMM*, pages 327–338, 2010.

[62] W. Wu and M. Crawford. Potential performance bottleneck in linux tcp. *Int. J. Commun. Syst.*, 20(11):1263–1283, Nov. 2007. ISSN 1074-5351. . URL `http://dx.doi.org/10.1002/dac.v20:11`.

[63] M. Zaharia, D. Borthakur, J. S. Sarma, K. Elmeleegy, S. Shenker, and I. Stoica. Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling. In *EuroSys*, pages 265–278, 2010.