Power-Efficient Networking for Balanced System Designs: Early Experiences with PCIe

John Byrne, Jichuan Chang, Kevin T. Lim, Laura Ramirez, Parthasarathy Ranganathan
Hewlett Packard Labs, Palo Alto

ABSTRACT

Recent proposals using low-power processors and Flash-based storage can dramatically improve the energy-efficiency of compute and storage subsystems in data-centric computing. However, in a balanced system design, these changes call for matching improvement in the network subsystem as well. Conventional Ethernet-based networks are a potential energy-efficiency bottleneck due to the limited performance of gigabit Ethernet and the high power overhead of 10-gigbit Ethernet. In this paper, we evaluate the benefits of using an alternative, high-bandwidth, low-power, interconnect—PCIe—for power-efficient networking. Our experiments using PCIe's Non-Transparent Bridging for data transfer demonstrate significant performance gains at lower power, leading to 60-124% better energy efficiency. Early experiences with PCIe clustering also point to several challenges of PCIe-based networks and new opportunities for low-latency power-efficient datacenter networking.

1. INTRODUCTION

Achieving power efficiency is a key challenge in data-centric computing, as the costs of power and cooling become a major component of the total costs of ownership. Critically, the I/O intensity of these workloads (which leads to new and often reduced compute-to-IO ratios) and the data communication needed in large-scale systems necessitate a balanced system design approach to improve energy efficiency.

Recent proposals [1][2][3] combine low-power, embedded-class processors with Flash-based storage, designing "microservers" that achieve 2-100 times better energy efficiency by rebalancing compute versus I/O. These proposals exploit processors optimized for performance/watt and energy proportionality [4], reducing the peak and idle power of the compute subsystem. In parallel, they improve the storage system's energy efficiency by adopting hardware and software optimizations for solid state storage.

The rapid changes in compute and storage, however, expose the network as a potential performance and energy-efficiency bottleneck. However, prior microserver proposals focus primarily on Ethernet and socket programming as the default hardware/software combination for the network. This decision is partly because there are few alternatives beside Ethernet, but also acknowledges the business advantages of traditional networks. Ethernet has the cost benefits of commodity hardware, and socket programming is widely used in data-centric software. For similar reasons, recent work in improving the scalability and energy-proportionality of data center networks [5][6][7] also assume Ethernet.

Ethernet, however, has high performance and power overheads due to the need to drive longer-distance cable and its high packet processing requirements. On compute/storage-optimized platforms in particular, the performance and power overheads of gigabit Ethernet (1GbE) can become a limiting factor. 10-gigabit Ethernet (10GbE) may alleviate the performance issue but consumes more power.

In this paper, we examine the opportunities and challenges of using an alternative, non-Ethernet interconnect that has higher performance and lower power. To match the cost benefits of Ethernet, we choose a commodity interconnect—PCIe—and demonstrate its performance and power benefits in small clusters. Our goals are to understand the benefits of PCIe as a "local" network and the challenges in making it part of "global" datacenter-scale networks.

We make two main contributions. (1) Using workloads with varied compute, storage and network requirements, we examine the performance and power impacts of different networks on the overall system. Our experiments identify today's Ethernet as an important performance and energy-efficiency bottleneck, especially for data-centric systems with power-optimized compute and storage subsystems. (2) We demonstrate the performance and power benefits of PCIe-based networks with a prototype system. Performance measurements show that the PCIe-based network provides more than 80% speedup over 1GbE, at even lower power. With non-intrusive modifications to the application software, Hadoop/sort runs 20% faster. Overall, the PCIe network enables 60-124% better energy efficiency.

2. PRIOR AND PROTOTYPED DESIGNS

To illustrate the energy efficiency improvement trends, Table 1 lists examples of today's compute, storage and network components, and their characteristics in power and performance (for example, in raw bandwidths).

Typical clusters use server processors such as Xeon X5670 and multiple rotational hard drives, both of which are power hungry. Such combinations are a mismatch for data-centric workloads which often require higher I/O performance but less compute power. This observation has motivated recent proposals (e.g., [1][2][3]) to strike a new compute-to-I/O balance, using embedded-class processors with fast, efficient, Flash-based storage as an energy-efficient data-centric computing platform.

For example, the bottom half of Table 1 shows a Gordon node [3]. This design packages an Atom processor with DDR2 memory, optimized Flash storage, chipset, and gigabit

Table 1: Component performance and power

	Part	GHz	Max Power (W)	# Cores
CPU	Xeon X5670	2.9	95	6-core
	Atom N570	1.66	8.5	2-core
Memory and Storage	**Part**	**Capacity**	**Max Power (W)**	**R/W BW (GB/s)**
	Hard drive	1.5TB	15	0.125
	DRAM	4 GB	6 (DDR3)	12.8
	Intel X25-M	120 GB	2	0.25/0.1
	Fusion-IO	160 GB	11	0.75/0.7
Network	**Part**	**# Ports**	**Max Power (W)**	**Raw BW (GB/s)**
	1Gb/s Ethernet	48-port switch	3.9 = 1.9[a]+2[b] (NIC/switch)	0.125
	10Gb/s Ethernet	24-port switch	17.3 =10[c]+7.3[d] (NIC/switch)	1.25
	InfiniBand QDR	36-port switch	21.1=15.5[e]+5.6[f] (Card/cable/switch)	5
	PCIe Gen2 with switch	24 x4 links	< 1[g] (per x4 port)	2
Recently Proposed Energy-Efficient Configuration [3]				
CPU	Atom Z540	1.9 GHz	2.4 W	1-core
Store	DDR2	2 GB	5.3 W	< 10 GB/s
Store	Flash	256 GB	6W (w/ controller)	2.2/1.1
Net	1GbE	48-port	3.9W	0.125

Sources: (a) Intel Gigabit CT Desktop Adapter; (b) HP ProCurve 2650; (c) Intel NE020 SFP+; (d) Brocade TurboIron 24X Switch; (e) Mellanox ConnectX MHQH19 QDR adapter; (f) Mellanox InfiniScale IV IS5024 switch; (g) PLX PEX 8696 switch [8].

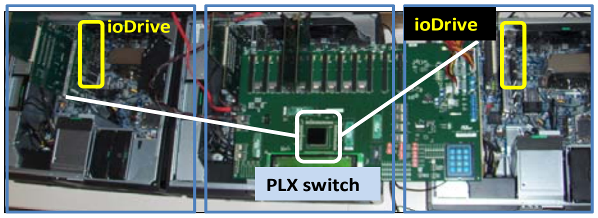

Figure 1: Testbed cluster (3 nodes with PCIe)

Ethernet in an add-on card. Despite their efficiency improvements, such proposed systems typically use gigabit Ethernet and do not investigate other network options.

To explore the balance between compute, storage and network subsystems, we build a small testbed cluster to measure the performance and power impact of different subsystems (Figure 1) and study the following options:

- **Compute**: Our systems use the Xeon X5670 processor running at 1.6GHz in low-power mode. The processor uses 9 watts per core peak power, and is configured to use 1, 2, or 4 cores.
- **Memory and Storage**: All the systems use 4GB DDR3 memory (6 watt). For storage, we study two configurations: (i) a Fusion-IO ioDrive SLC 160GB (`FIO`) with power modeled after [3] and (ii) a RAM-based file system (`RAMFS`) to model future NVM-based storage.
- **Network**: We study two main networks: (i) a baseline gigabit Ethernet (`1GbE`) with NIC and switch power values as in Table 1 and (ii) a PCIe network using the PLX PEX 8696 switch with 2GB/s x4 lanes (`PCIe`) [8].

The devices above have small form factors and low power requirements. Although currently hosted in workstations, they can be placed on small PCBs connected to a PCIe-switched midplane. Such compact designs allow the use of PCIe in a medium-size cluster (e.g., 48-node).

3. PCIE-BASED NETWORK

PCI-express (PCIe) is a commodity IO interconnect widely used in motherboards and backplanes. PCIe as a network fabric also appears in emerging system proposals (e.g., SeaMicro [9]). With appropriate provisioning for IO consolidation, PCIe can be cost-competitive with Ethernet. For example, Leigh et al. [10] show the cost/bandwidth advantage of PCIe over Ethernet. Each link between PCIe devices can have 1 to 32 lanes with increasing bandwidths. In our testbed, we use 4 lanes (x4) of Gen2 PCIe per link, each with 2GB/s raw bandwidth. Due to the short link distance and simple protocol, per-node power for PCIe network is below 1 watt, much lower than Ethernet (Table 1).

To support blade servers and I/O virtualization, PCIe vendors now provide inter-system switching capabilities that support large port counts and high bandwidth. Instead of PCIe Transparent Bridging which manages all PCIe devices in the cluster under one operating system, we use PCIe Non-Transparent Bridging (NTB [11]) which allows independent nodes in the cluster to each manage their own devices and use the communication path between them for DMA data transfers. For control and synchronization, NTB also provides doorbell and mailbox registers. We choose one node in the cluster as the primary root host and the remaining nodes as secondary leaves.

Each node in our testbed has a PEX8609 1-port adapter connected to a central PEX8696 switch, forming a star topology. The switch provides two PCIe BAR registers to map a region of the address space (called "window") on the source node to a region on the destination node.

In our software implementation, we measure actual achieved latency and bandwidth. A 4-byte CPU read through the window from remote memory takes ~2 microseconds. DMA writes can be sustained at 1.56GB/s and reads at 1.06GB/s. DMA reads have lower bandwidth because they require a round-trip while the writes are unidirectional. Although PCIe links are point-to-point, PEX8696 can forward requests by overlaying one link's destination window to a second link's source window, without incurring extra latency.

Multiple software interfaces can be implemented on top of the PCIe adapter/switch, with different tradeoffs between ease-of-porting and performance/efficiency. For example, socket and remote procedure call (RPC) interfaces preserve familiar APIs for the programmers, but add complexity in the protocol stack and hence software overhead. Encapsulating the hardware with a RDMA/DMA interface has high performance and efficiency, but requires porting existing applications. Based on our observation of the Hadoop framework, the bulk of network traffic in today's data-centric processing systems usually involves file or block transfers between nodes. We have thus implemented a library for PCIe-based data transfer between nodes, and ported applications by replacing their existing socket-based data transfer code with function calls into this library.

The PCIe data transfer library does not provide full-fledged network management functions, except for naming support. Nodes and devices in the cluster are initially discovered and configured. The name-to-device mappings are stored in a table, and consulted by the library implementation at runtime.

4. EVALUATION RESULTS

With the testbed setup, we evaluate the performance and power of different system configurations. We augment the power model of Gordon [3] with devices listed in Table 1 and calculate the system-level power consumption including switches and system overhead power. Because our current testbed does not have 10GbE, we indirectly estimate impact of future 10GbE Ethernet by using (1) the 10GbE power numbers from Table 1 and (2) the performance of PCIe. Since 10GbE has lower raw bandwidth and higher CPU overhead for packet processing [12], such estimation only makes our discussion of PCIe benefits more conservative.

4.1. Workloads with varying requirements

(1) **Distributed `sort`**. Our benchmark models a cluster JouleSort [13] with a data shuffle phase that transfers the key/value pairs to their destination nodes, and a local sort phase (using nsort [14]).

(2) **Distributed `grep`**. Our benchmark uses the grep dataset from Brown University [15] with a local grep phase (using the Unix `grep` utility) and a reduce phase that collects matching results to a central node. To vary the workload resource requirements, we examine two computation variants: "*LoC*" for low compute that matches simple patterns and "*HiC*" for high compute that matches complex patterns, and two network variants: "*LoN*" for low network with 10% matching entries and "*HiN*" for high network with 50% matching entries. Together these variants cover 4 different compute-to-communication ratios.

(3) **Hadoop/sort**. Hadoop is a widely used MapReduce implementation for large-scale data processing. Our current implementation replaces the HTTP-based shuffle code with calls to our PCIe data transfer library. Although data replication allows higher speedups for PCIe, replication is not an inherent part of sort and can overemphasize the benefits of PCIe network. Therefore, HDFS block replication is not enabled here.

4.2. Distributed `sort` results

Figure 2 shows the execution time breakdown of `sort` between the local sort and shuffle phases for different system configurations. For example, the baseline "FIO/1GbE with 1c" uses Fusion-IO and 1GbE with one 1.6GHz core, and "RAMFS/PCIe with 4c" uses RAMFS-based high-speed storage with PCIe and 4 cores.

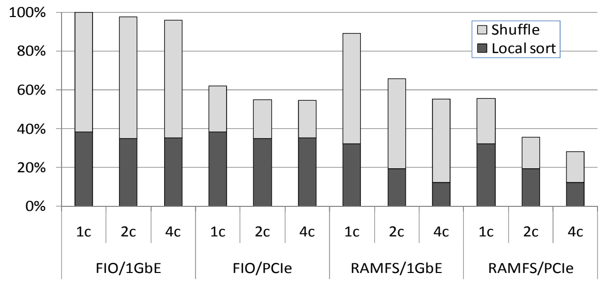

Figure 2: Relative execution time of `sort`

Shown in the left half of Figure 2, the biggest speedups (ranging from 61% to 83%) are achieved by replacing 1GbE with PCIe, demonstrating the benefits of improved network bandwidth. With PCIe, the largest improvement can be achieved by using the "RAMFS" storage device. The changes in network and storage consequently expose the performance benefits of increasing processor core count, where 4-core systems with RAMFS achieve a 3.5X speedup.

Table 2: `Sort` energy efficiency improvements

Net	1GbE		10GbE		PCIe	
Store	FIO	RAMFS	FIO	RAMFS	FIO	RAMFS
1-core	0%	12%	4%	16%	83%	105%
2-core	-25%	11%	-5%	46%	45%	124%
4-core	-51%	-14%	-31%	33%	-9%	78%

Table 2 lists the energy efficiency improvements of various configurations. Similar to the performance results, gigabit Ethernet is an efficiency bottleneck; the efficiency gains by varying all other resources (shown in the first two columns) are marginal and often negative. Today's Flash-storage also limits the benefits of upgrading to 10-gigabit Ethernet, mainly because the performance gained from 10GbE is offset by the increased network power. Combining RAMFS with10GbE does boost efficiency, by 16-46%.

PCIe network, on the other hand, can provide 83% better efficiency for today's storage, by combining its performance and power benefits. With re-balanced storage and network bandwidths, compute power can be increased (in the last column) to further improve energy efficiency.

Overall, these results show that improving network performance and efficiency has the largest impact for `sort`. Although 10GbE provides speedups, its power needs to be reduced in order to achieve net improvement in efficiency. PCIe network can both improve performance and reduce power, leading to significant energy efficiency gains.

4.3. Distributed `grep` results

In order to understand the impact of the network subsystem on a range of workloads, we next examine `grep`, with varying compute and network needs.

Figure 3 compares the runtime breakdowns of the "local grep" and "reduce" (network-heavy) phases, against the 1GbE/1-core baseline, for 4 configurations each with four compute/network requirement ratios. For brevity, we only show the results of using up to 2 cores with Fusion-IO (as "RAMFS" storage only has marginal performance benefits).

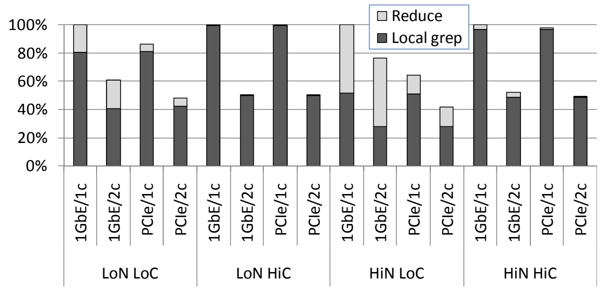

Figure 3: Relative execution time of cluster `grep`

Since `grep` is more compute intensive than `sort`, increasing the core count from 1 to 2 shows the most impact on performance (shown by the differences between the two pairs of adjacent bars in each group). Still, replacing 1GbE with PCIe achieves 16% or more speedups for the low-network, low-compute workload (LoN LoC), and 56-82% speedups for the high-network, low-compute workload (HiN LoC). For high-compute workloads, where network is by definition not the bottleneck, high-speed networks only provide less than 6% speedups.

Table 3: `Grep` energy efficiency improvements
(Normalized to 1GbE/1-core, best results highlighted in **bold**)

Net	1GbE		PCIe		
CPU	1-core	2-core	1-core	2-core	4-core
LoN LoC	0%	19%	31%	**66%**	66%
LoN HiC	0%	45%	14%	60%	**99%**
HiN LoC	0%	-4%	77%	**90%**	55%
HiN HiC	0%	39%	16%	62%	**98%**

Table 3 compares the energy efficiency results between 1GbE and PCIe for the `grep` workload. We add PCIe/4-core to further illustrate the tradeoffs between scaling network vs. compute. With 1GbE and low-compute workloads (LoC), changing from 1-core to 2-core has low impact on efficiency, while moving to PCIe can improve the efficiency significantly (by 66% and 90%). However, the benefits of moving to 4-core depends on the workload's compute intensity, as the 4-core based configurations achieve the best efficiency only with high-compute workloads (HiC).

4.3. Hadoop/sort results

Table 4 shows execution time and speedup for Hadoop/sort, with different configuration parameters. Hadoop is a complex distributed software with interrelated tuning knobs. We focus only on the impact of changing the network parameter and use the best performing compute/storage configuration as the baseline (i.e., four 2.9GHz cores running with Fusion-IO).

Due to limited number of DMA mapping windows (two in our testbed), having many small DMAs tends to increase the overhead of PCIe networking. We hence experiment with different HDFS block sizes and MapReduce task sizes (split sizes) to control the data transfer granularity.

Table 4: Hadoop/sort performance

	Block size / split size (MB)		
	256 / 256	256/550	512/550
1GbE/socket	114.8	101.2	101.2
PCIe/Data-transfer	105.2	84.2	87.0
Speedup	9%	20%	16%
Efficiency improvement	11%	24%	20%

The best runtime and speedup is achieved with a relatively large MapReduce input task size (last two columns), while HDFS block size has a second-order performance effect. PCIe network can improve Hadoop/sort execution time by up to 20%. The energy efficiency improvements are slightly higher than the speedups, mainly because the network power is relatively small in this configuration.

5. DISCUSSION

The previous section demonstrated the potential performance and power benefits of PCIe networks with the data-centric workloads that we have ported. However, there are still many open challenges regarding the appropriate use-cases for PCIe networks and using PCIe as part of the datacenter network. Below, we discuss some of the limitations and challenges of PCIe networks as well as new opportunities enabled by the high-performance energy-efficient network.

Applicability of PCIe networks

Although our results show that data-centric workloads running on power-efficient nodes benefit from PCIe, it is not a solution applicable to all scenarios. Compute-heavy or communication-light workloads cannot leverage PCIe's ample bandwidth. Additionally, high-power server clusters, where network contributes only a small fraction to the total power, are unlikely to take advantage of PCIe's low power. Therefore it is important to consider the use-case when implementing PCIe networks.

Furthermore, the scale of the network impacts how extensively PCIe can be used. PCIe is designed as a local I/O fabric to connect only tens of nodes, as compared to Ethernet, which can connect thousands of nodes. Thus while PCIe can be used alone to provide a high performance local network for small-scale clusters, at a datacenter-scale such local networks must be integrated with Ethernet or InfiniBand to provide appropriate connectivity.

Scalability

The discussion above opens up new questions regarding PCIe within scalable networks. What topologies are most appropriate for placing small, local PCIe networks within a larger scale, Ethernet connected network? Alternatively, is it possible to build a scalable network solely out of PCIe, for example, using a tree-based hierarchy?

As an evolutionary path, PCIe local network can replace the edge switches/cables in an Ethernet or InfiniBand-based global datacenter network (e.g., [5][6][7]). Such combinations take advantage of high-bandwidth low-power PCIe local networks and the scalability of Ethernet or Infiniband. This approach, however, requires the PCIe switch to interface between the two tiers of networks and appropriate abstractions of the network heterogeneity for applications. These hybrid designs will help to provide a comparison baseline for an all-PCIe network.

At a lower level, there are questions regarding how many nodes a PCIe switch can support; this capability directly affects local network size, performance and efficiency. Currently, the PLX switch's limited number of DMA windows is a scalability bottleneck, as multiplexing connections adds DMA setup overheads and reduces performance. Future PCIe switches, however, are expected to address this limit.

Manageability and programmability

Adding a heterogeneous network layer can potentially compound already complex datacenter network management. One way to encapsulate the heterogeneity is for the PCIe network to provide the same abstraction to the rest of the "global" network. Porting Open vSwitch [16] on PCIe, tunneling Ethernet over PCIe, and virtualizing the datacenter network [17] are a few example approaches to provide the Ethernet control plane while retaining PCIe's performance/power benefits in the data plane.

Another challenge is to choose the right API and protocol to expose the PCIe local networks. Figure 4 lists and compares several network stack options. On the left hand side, Ethernet with socket interface is used by a wide spectrum of applications but has high overhead. On the right hand side, our data-transfer library directly exposes PCIe with better efficiency, while InfiniBand can be exposed through either the verbs or socket interface [18]. These approaches are based on different hardware and software designs, representing potential tradeoff points along the efficiency and programmability/portability spectrum.

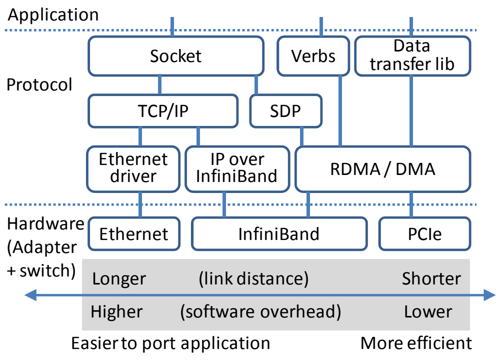

Figure 4: Network stack options (based on [18])

Fortunately, our early experiences indicate that porting modern software is quite feasible. Only minor, localized changes were made to Hadoop to exploit the benefits of PCIe network due to its highly modular code.

Similar to our study, Sur et al. [18] examine whether InfiniBand can benefit Hadoop HDFS. They demonstrate the performance advantage of InfiniBand as well as the feasibility of application porting. InfiniBand has higher costs than PCIe, but we leave the comparison as future work.

Fault tolerance

PCIe Non-Transparent Bridging allows nodes in the cluster to be in separate fault domains and support failover. Such fault-tolerance features, however, are insufficient for an enterprise-grade network. There are still open research questions regarding the fault model for PCIe network and implementing such a fault model while retaining PCIe's simplicity and performance advantages.

New opportunities

With a low-latency high-bandwidth network like PCIe, the overhead of accessing remote compute, memory, and storage resources can be much lower than what is possible today. Such improved performance provides datacenter resource managers new flexibility in pooling and using resources from different servers, or even in building disaggregated servers (e.g., [19]).

Another opportunity lies in integrating PCIe local networks with an optical datacenter network. The PCIe switch can aggregate the network traffic from its local network and share the optical cables connecting to the datacenter network, thereby amortizing the high cost of optics over all the servers within its local network.

6. CONCLUSIONS

Energy-efficient data-centric computing platforms will have to address computation, storage and communication subsystems in a holistic, balanced manner. Recent improvements in efficient compute and storage devices now leave network as a potential bottleneck, calling for solutions to simultaneously improve performance and reduce power.

In this paper, we demonstrate the benefits of power-efficient networks with PCIe for data-centric computing. With a Gen2 PCIe switch in Non-Transparent Bridging mode, each 4-lane link can achieve 1.56GB/s bandwidth using less than 1 watt, with little software overhead. When such capabilities are exposed through a simple data transfer library, they can achieve more than 80% application performance gains and enable balanced configurations that are 60-124% more energy efficient than an aggressive baseline.

Many challenges remain before a PCIe network can become part of the power-efficient datacenter network, although none of them seems unsurpassable. New opportunities also abound in exploiting low-latency, low-overhead PCIe for emerging applications.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the contribution of Steven Pelley during the early phase of the project, and discussions with Indrajit Roy, Bruce Walker, Mitch Wright, Siamak Tavellaei, Moray McLaren and Jeff Mogul as well as valuable reviewer comments.

REFERENCES

[1] K. T. Lim, P. Ranganathan, et al. Understanding and designing new server architectures for emerging warehouse-computing environments. ISCA 2008.

[2] D. Andersen, J. Franklin, et al. FAWN: a fast array of wimpy nodes. SOSP 2009.

[3] A. Caulfield, L. Grupp, and S. Swanson. Gordon: using flash memory to build fast, power-efficient clusters for data-intensive applications. ASPLOS 2009.

[4] L. A. Barroso and U. Hölzle. The case for energy-proportional computing. IEEE Computer, 40(12):33–37, 2007.

[5] R. N. Mysore, A. Pamboris, et al. PortLand: A Scalable Fault-Tolerant Layer 2 Data Center Network Fabric. SIGCOMM 2009.

[6] A. Greenberg, J. R. Hamilton, et al. VL2: A Scalable and Flexible Data Center Network. SIGCOMM 2009.

[7] D. Abts, M. R. Marty, et al. Energy Proportional Datacenter Networks. ISCA 2010.

[8] PLX. PEX 8696 96-Lane, 24-Port PCIe Gen 2 Switch. http://www.plxtech.com/products/expresslane/pex8696

[9] Anil Rao. SeaMicro Technology Overview. http://www.seamicro.com/sites/default/files/SM_TO01_64_v1%208.pdf

[10] K. Leigh, P. Ranganathan, et al. Fabric Convergence Implications on Systems Architecture. HPCA 2008.

[11] J. Regula. Using Non-Transparent Bridging in PCI Express Systems. 2004.

[12] K. K. Ram, J. R. Santos, et al. Achieving 10 Gb/s using safe and transparent network interface virtualization. VEE 2009.

[13] S. Rivoire, M. A. Shah, et al. JouleSort: a balanced energy-efficiency benchmark. SIGMOD 2007.

[14] Nsort. http://www.ordinal.com/

[15] A. Pavlo, E. Paulson, et al. A Comparison of Approaches to Large-Scale Data Analysis. SIGMOD 2009.

[16] Open vSwitch. http://openvswitch.org/

[17] J. Mudigonda, P. Yalagandula, et al. NetLord: A Scalable Multi-Tenant Network Architecture for Virtualized Datacenters. SIGCOMM 2011.

[18] S. Sur, H. Wang, et al. Can High-Performance Interconnect Benefit Hadoop Distributed File System? MASVDC 2010.

[19] K. T. Lim, J. Chang, et al. Disaggregated Memory for Expansion and Sharing in Blade Servers. ISCA 2009.