Network Support for Resource Disaggregation in Next-Generation Datacenters

Sangjin Han[+], Norbert Egi[†], Aurojit Panda[+], Sylvia Ratnasamy[+], Guangyu Shi[†], and Scott Shenker[+*]

[+]University of California, Berkeley, CA [†]Futurewei Technologies, Santa Clara, CA [*]ICSI, Berkeley, CA

{sangjin, apanda, sylvia}@cs.berkeley.edu {norbert, shiguangyu}@huawei.com shenker@icsi.berkeley.edu

ABSTRACT

Datacenters have traditionally been architected as a collection of servers wherein each server aggregates a fixed amount of computing, memory, storage, and communication resources. In this paper, we advocate an alternative construction in which the resources within a server are *disaggregated* and the datacenter is instead architected as a collection of standalone resources.

Disaggregation brings greater modularity to datacenter infrastructure, allowing operators to optimize their deployments for improved efficiency and performance. However, the key enabling or blocking factor for disaggregation will be the network since communication that was previously contained within a single server now traverses the datacenter fabric. This paper thus explores the question of whether we can build networks that enable disaggregation at datacenter scales.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design

General Terms

Design, Performance

1 Introduction

From the dawn of the PC era to modern day datacenters, the computer has been the cornerstone of computing infrastructure. At a high level, a computer is built by tightly integrating a small amount of the various resources needed for a computing task – processors, memory, networking, and storage – onto a single motherboard. While there is no denying the tremendous success of these computer- or server-centric architectures, we believe modern datacenters would be better served by an alternate construction in which the internal resources of a server are *disaggregated*, by which we mean that each resource type is built as a standalone "resource blade" and a datacenter-wide network directly interconnects all resource blades. In such a datacenter, the aggregation of resources needed by a job is then logical (allocated by a software scheduler) rather than physical (dictated by hardware).

Disaggregation makes it easier to optimize for efficiency, an issue of mounting importance given the massive scale of today's datacenters. By physically decoupling different resources, operators can more easily adopt the state-of-the-art in any particular technology and/or customize their infrastructure to maximize "performance per dollar" for their target workloads. For example, an ongoing challenge for hardware architects is that technologies for CPUs, memory, storage, etc. have very different cost, performance, and power scaling trends and this constrains their integration. For instance, recent work [16, 17] warns of an impending "memory capacity wall" due to the growing imbalance in the peak compute-to-memory-capacity ratio and argues that traditional compute-memory co-location on a single server will not be sustainable. Similarly, upgrading to a new technology (e.g., NVRAM, memristors, or silicon photonics) or incorporating specialized hardware (e.g., GPUs or accelerators for encryption, coding, or regular-expression matching) can be burdensome since it requires reworking the integration process, server form factor planning, and motherboard designs. Disaggregation instead allows each technology to evolve independently and gives operators finer-grained control over how they select, provision, and upgrade individual resources.

Disaggregation also enables more efficient use of the resources already in a datacenter deployment. For ex-

Hotnets '13, November 21–22, 2013, College Park, MD, USA.

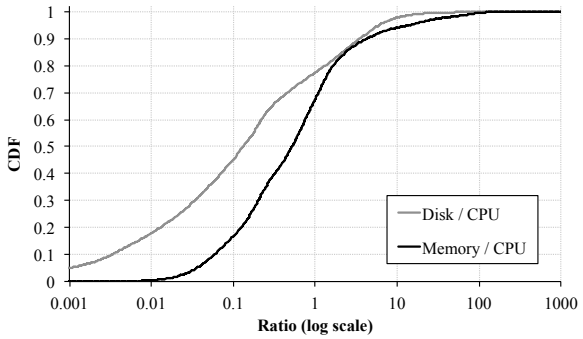


Figure 1: Distribution of relative disk/memory capacity demand to CPU usage for tasks in Google's datacenter.

ample, consider an application that requires 8 cores and 1 GB of memory in a datacenter where the typical server is provisioning with 8 cores and 32 GB. While a server-centric architecture would waste 31 GB memory, but in a disaggregated architecture, the memory is not rigidly coupled to any particular CPU and hence could be allocated to other applications. Such inefficiency is hard to avoid in server-centric architectures since applications vary greatly in their resource needs and hence there is no easy one-size-fits-all answer to provisioning server resources. Figure 1, which plots the ratio of disk-to-CPU and memory-to-CPU consumption for tasks in Google's datacenter [23], shows that the resource requirements of tasks is spread over more than three orders of magnitude! Disaggregation thus can allow statistical multiplexing to occur at a much larger scale and hence naturally enables higher efficiency. In addition, incremental resource upgrade is straightforward and non-disruptive, when a disaggregated datacenter as a whole faces more resource demands.

In short: disaggregation brings greater *modularity* to computing hardware, and with it, the flexibility that enables greater optimization and customization.

Several industry and research efforts can be viewed as already on the path to disaggregation – e.g., SeaMicro's server architecture [6] uses a looser coupling of components *within* a single server, while Intel's Rack Scale Architecture (RSA) [15] extends this approach to rack scales. And, as mentioned above, the authors in [16, 17] argue the inevitability of disaggregating memory and CPUs and even prototype a disaggregated memory blade. What is not clear however is how widely one might apply disaggregation and hence how general the benefits of modularity would be. It may be possible that mid-scale disaggregation (resource disaggregation at the rack- or pod-level) would provide enough efficiency and flexibility. However, we speculate that having a flat, unified interconnect across a datacenter can be much simpler than the two levels of network structure, in terms of the design and scheduling complexity. As a straw man, in this paper, we rather ask whether one can disaggregate resources at datacenter scale. We hope that exploring this extreme design point will shed light on the extent to which we can and should disaggregate.

What is the barrier to scaling disaggregation? The greatest burden of disaggregation falls on the network. Communication that was previously contained within a server now hits the datacenter-wide fabric. This both increases the load on the network and makes the need for low latency communication critical. The network will be the key enabling or blocking factor to disaggregation.

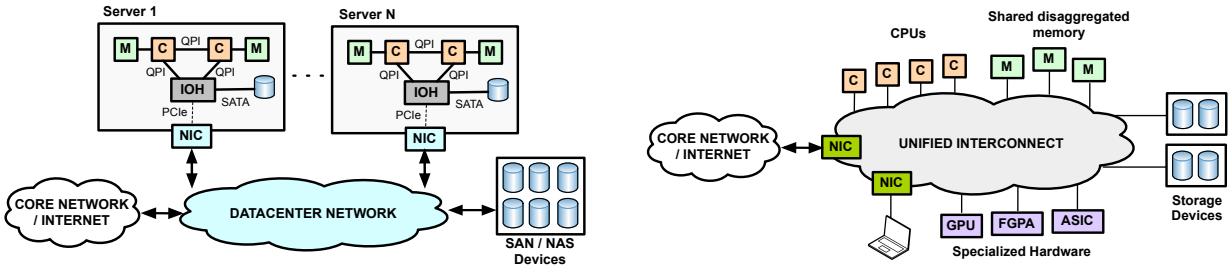
Thus, in this paper, we explore questions such as: what will the bandwidth and latency demands due to disaggregation look like? What are the key application and hardware parameters that affect these demands? How might we meet these demands? Although our discussion is neither conclusive nor comprehensive, we hope it triggers discussion on network support for disaggregation and disaggregated datacenter architectures more broadly.

While we focus primarily on questions of network performance, we recognize that disaggregation has broader implications for how we build datacenter systems that we do not begin to address (e.g., hardware design of resource blades, new unified scheduler designs, programming models that better exploit disaggregation, etc.). We touch on these briefly at the end of this paper but leave an in-depth discussion to future work.

2 Disaggregated Datacenters

We elaborate on what we mean by a disaggregated datacenter and highlight some important assumptions we make regarding how disaggregation will be implemented. As illustrated in Figure 2, the high-level idea behind disaggregation is to develop standalone hardware "blades" for each resource type including CPUs, memory, storage, and network interfaces as well as specialized components (GPUs, various ASIC accelerators, etc.). Those resource blades are interconnected by a datacenter-wide network fabric. Understanding the specifications and nature of this network fabric is our focus in this paper.

With some simplification, we can view each blade as comprised of the resource component in question (CPU, memory, I/O devices) with a direct interface to the datacenter network fabric. In the case of memory, each blade may also require a controller ASIC (or a lightweight processor) that implements local resource management and address translation between the remote CPU's view of its address space and the local addressing used internally within a blade. Many I/O device controllers are beginning to support virtualization via the PCIe SR-IOV or MR-IOV features, which can be also leveraged in the disaggregated datacenter network in a similar fashion. We note that while the implementation of such blades may require some additional new hardware, it requires no change to existing components such as CPUs, mem-



(a) Current datacenter

(b) Disaggregated datacenter

Figure 2: Architectural differences between server-centric and resource-centric datacenters

ory modules, or storage devices themselves. We refer the reader to [16–18] for an in-depth discussion of hardware designs for disaggregated blades.

The scheduler allocates resources for computation needs; such allocation involves selecting the location of assigned resources, configuring the resultant resource and address space assignments at the different resource blades, and (as required) configuring the network that interconnects these blades. Each rack may contain different types of resource blades and (as today) the scheduler may optimize for locality when it allocates resources to a job. The best layout of resource blades and the corresponding scheduler optimizations is a topic for future exploration.

The usage model, implementation, and feasibility of resource disaggregation are simultaneously interdependent in the design of a disaggregated datacenter. For reasons of performance, scalability, and applicability, we make a few important assumptions as follows.

Assumption 1: VM as a computational unit
The current datacenter usage model is heavily based on the server-centric architecture. While physical servers in datacenters have evolved to server virtualization [9] or other comparable technologies [26], they are still all centered around the concept of "server", which aggregates slices of hardware resources within a server. The operators/schedulers plan virtual machines (VMs) to meet the computational demands and place jobs across the VMs.

In contrast, the usage model of a disaggregated datacenter does not necessarily follow the same approach; since computation, storage, and I/O functions can be completely disseminated across the datacenter, we do not need to restrict our usage model within the VM-oriented architecture. However, we note that the VM model can be still useful, as in this way we can leverage the existing software infrastructure, such as hypervisors, operating systems, datacenter middleware, and applications with little or no modification. Thus in this paper we assume that computational resources are still utilized by aggregating them to form VMs, while each resource is now physically disaggregated across the datacenter.

Assumption 2: local/remote memory
While disaggregation of I/O devices is relatively straightforward as briefly discussed above, memory disaggregation brings a set of new challenges in terms of performance. Since memory access from CPUs must run at very high speed (we discuss this in more detail in §3), similarly to prior work [16, 17], we assume that each CPU blade retains some amount of local memory that acts as a cache for remote memory; thus disaggregating memory can be viewed as expanding the memory hierarchy to include a remote level. While remote memory may be allocated to any CPUs in the datacenter, local memory is dedicated to its co-located CPU. As we shall see, the assumption of local memory is necessary to ensure reasonable performance given the increased latency to access remote memory.

Assumption 3: page-level remote memory access
We assume that CPU blades access remote memory at the page-granularity (4 KB in x86) over the fabric. While typical memory access between CPU and DRAM in traditional servers occurs in the unit of cache-line size (64 B in x86), it is known that page-level access better exploits spatial locality in common memory access patterns and amortizes the round-trip latency more effectively [16]. In addition, page-level access requires little or no modification to the virtual memory subsystem of hypervisor or operating system, and it is completely transparent to user-level applications. We further assume that those remotely accessed pages are not shared by multiple VMs at a given time, in order to not introduce cache coherence traffic across the network.

Prior work
As mentioned earlier, there is growing interest in disaggregation in both industry and research [2, 6, 15–17]. We build on these efforts – in particular prior work on understanding how memory might be disaggregated [16, 17] as a starting point. To date, these efforts consider disaggregation at server or rack scale and, as such, they use specialized interconnects designed for their specific context and do not discuss network support for disaggregation more generally nor consider the possibility of leveraging known datacenter network technologies to

Communication type	Latency (ns)	Bandwidth (Gbps)
CPU - CPU	10	200
CPU - Memory	20	300
CPU - 10G NIC	$> 10^3$	10
CPU - Disk (SSD)	$> 10^4$	5
CPU - Disk (HDD)	$> 10^6$	1

Table 1: Typical latency and peak bandwidth requirements within a traditional server. Numbers vary between hardware.

enable disaggregation; e.g., the network in SeaMicro's architecture implements a 3D torus interconnect, which only disaggregate I/O and does not scale beyond the rack.

3 Latency/Bandwidth Requirements

In a disaggregated datacenter, traffic between resources that was contained within a server is now carried on the "external" network. As other types of interconnects, the key requirements will be low latency and high throughput to enable this disaggregation. We review the communication types between resources within a server in Table 1, to examine the feasibility and limitation of such a network. For I/O traffic, such as network interfaces and disks, the required latency and bandwidth level is sufficiently low to consolidate them within the unified network.

Besides I/O, we expect that we cannot fully embrace CPU-to-CPU (cache coherence) and CPU-to-memory traffic into the external network, due to their high bandwidth and extremely low latency requirements. We can avoid or reduce those traffic in the network by making two decisions. First, we can keep each VM from spanning multiple CPU blades, to eliminate CPU-to-CPU traffic. Since latest server-class x86 CPUs already have 8-16 cores and Moore's law is keeping pace, we posit that this single-CPU restriction is reasonable[1]. Second, as mentioned earlier, instead of fully disaggregating memory, we envisage that each CPU has a small amount of private, directly connected local memory. In order to meet elastic memory capacity demands beyond local memory, VMs may access remote memory. While the basic idea was previously proposed [16], we extend this idea further by examining if we can push remote memory access to the datacenter scale, rather than within servers/racks over dedicated links.

As briefly discussed, we assume that the remote memory is managed at the page granularity, in conjunction with virtual memory page replacement algorithms of either the hypervisor or the operating system [17]. We take this approach as it not only guarantees frequently-accessed remote pages to be cached locally, but also transparently supports unmodified applications. For each

[1]Or alternatively, a single CPU blade can still house "multiple" CPU sockets just as servers do today, to contain low-latency, high-bandwidth CPU-to-CPU traffic within the blade. In this case, the scheduler will need to take it into account when making scheduling decisions.

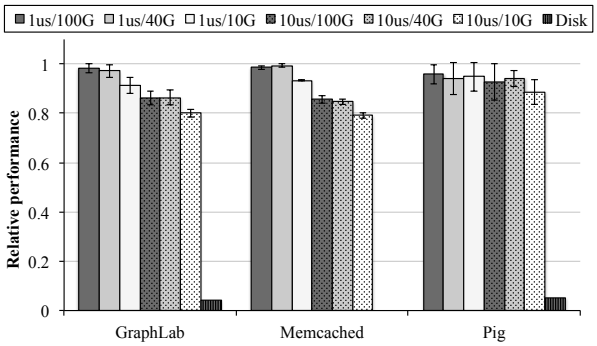


Figure 3: Application-level performance degradation with disaggregated memory, over various network configurations. 75% of the working set size was configured as remote memory. Memcached with disk-based swap performed too slow to get the benchmark result.

paging operation there are two main sources of performance penalty: *i)* software overhead for trap and page eviction and *ii)* page transfer time over the network. In this section, we focus on the latter one, by looking into the performance requirements of the network to support remote memory access, without introducing significant performance penalty.

We conduct a series of experiments to examine how network latency and bandwidth affect application performance with remote memory access. For the experiment, we consider three popular datacenter workloads: GraphLab [19], a machine learning toolkit (with the collaborative filtering example provided in the package); Memcached [3], an in-memory, key-value store (with the YCSB cloud benchmark tool [11]); and Pig [20], a data-analysis platform based on Hadoop [1] (with the PigMix benchmark suite [5]). We emulated remote memory access by implementing a special swap device (backed by physical memory rather than a disk) and injecting artificial delays to emulate network round-trip latency and bandwidth for each paging operation. We measure relative performance on the basis of throughput or completion time as compared to the zero-delay case. Note that the experiment results do not account for the delay caused by software overhead[2] for page operations. The result should be interpreted as relative performance degradation over different network configurations, not the absolute performance of disaggregation.

Figure 3 depicts the results with six latency/bandwidth combinations, given 25% of local memory capacity of the measured memory footprint for each workload. We observe two interesting points from the result. First, use

[2]The per-page overhead was reported to be around 2-6 µs in the research prototype implemented in Xen, depending on the page replacement algorithm [17]. We optimistically expect that it can be further reduced to sub-micro seconds with faster CPUs and software optimization.

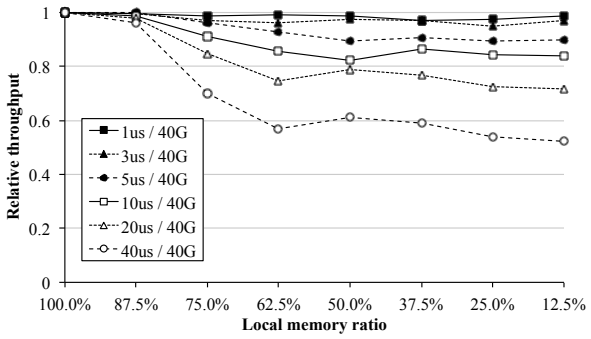


Figure 4: Memcached performance with varying memory ratio and round-trip latency

of remote memory can drastically improve application performance when the working set size is bigger than physical memory, as compared to traditional disk-based swap. Since the working set size is hard to predict in advance, memory tends to be highly over-provisioned in datacenter servers to prevent thrashing. Disaggregated remote memory can reduce this waste by providing an elastic memory capacity pooled at the datacenter scale. Second, low latency is more important than high bandwidth. The 100 Gbps bandwidth did not provide any significant improvement over the 40 Gbps link. In contrast, 10 µs round-trip latency causes noticeable performance degradation, as compared to the 1 µs case.

To examine the role of latency in memory disaggregation, we take a closer look at how memcached performance overhead varies along the round-trip latency configurations. For the experiment, we fixed the bandwidth at 40 Gbps and varied the amount of local memory from 1 GB to 8 GB, out of the total 8 GB working set size. Figure 4 again confirms that low latency will be crucial in the implementation of resource disaggregation. The low latency (≤ 10 µs) cases show fairly constant performance over any local memory ratio, while the performance of high latency (≥ 20 µs) cases quickly degrades as we rely more on remote memory.

The desired 40 Gbps bandwidth is already within reach, as the latest interconnect standards, such as PCIe, Ethernet, and InfiniBand, already provide sufficient capacity. Also, we found that the *average* link utilization for remote memory access was pretty low (< 5 Gbps), regardless of workloads and network configurations, implying that the aggregated bandwidth at the network core can be quite low. These facts strongly indicate that we should more focus on latency than bandwidth, for the design of the unified network. Although the very low (≤ 10 µs) end-to-end latency sounds challenging, we expect it to be feasible in the foreseeable future, as we discuss in the following section.

We admit that our simple experiments presented here leave many questions unanswered. For example, network congestion may cause spikes in latency, adversely affecting application performance. Also, besides latency and bandwidth issues, the network must be able to scale up to millions of disaggregated resources. We discuss some of these issues in the following section but leave an in-depth exploration to future work.

4 Research Directions

In this section, we briefly discuss some of the research questions that disaggregation raises on three fronts: *i)* approaches to building low-latency networks, *ii)* network architecture, and *iii)* systems architecture. Each of these merits a paper in itself; as such, what follows is more an enumeration than an in-depth discussion of potential issues.

4.1 Realizing Low Latency Networks

As demonstrated in §3, building low latency networks—with round-trip times under of 10 µs—will be critical for large scale disaggregation. Fortunately, this is a topic of that has been receiving a great deal of attention in recent research and, coincidentally, a recent paper [24] argues the feasibility of such low latency in datacenter networks in the near future. The authors cite the growing prevalence of cut-through switches and vendor plans for tighter integration of IO capabilities into the CPU as key enabling factors.

In addition to the hardware trends, we believe there are many opportunities to further reduce network latency including improved protocol designs [7] and all-optical switches with no buffering. The effectiveness and suitability of these approaches for disaggregation is a topic for future work.

An orthogonal approach to reducing latency is to try and reduce the network distance between the resources allocated to a job, in much the same way that map-reduce schedulers today aim for data locality in scheduling tasks. Future research should study how to best distribute resource blades across racks and the design of scheduler optimizations for low latency.

Finally, an important goal should be to achieve latencies that are not just low but also *deterministic* since high variability will lead to unpredictable application performance. An intriguing possibility here is the use of TDMA-based network architectures as proposed in recent work by Vattikonda et al. [27]

4.2 Network Architecture

We can probably build networks for disaggregated datacenters using existing networking technologies such as Ethernet, InfiniBand, or PCIe. An interesting research question – if only to understand what change might be desirable – is to ask what the ideal network architecture in support of disaggregation might look like.

It is worth noting that disaggregation effectively blurs the lines between what used to be separate intra- and inter-server networks. E.g., in Figure 2(a) we see that today's server architecture includes networks for communication between CPUs (e.g., Intel QuickPatch Interconnect or AMD HyperTransport protocols), between CPUs and memory (DDR3) and to peripheral devices (e.g., based on the PCIe protocol). Traditionally, these intra- and inter-server technologies have evolved very differently. Basic concepts such as variable-sized packets and best-effort service are common in inter-server networks but not so in intra-server links/networks. The network in a disaggregated datacenter combines aspects of both – e.g., it is resource-centric (like intra-server networks today) but is less tightly integrated with the endpoints and must operate at scale (like existing inter-server networks) – and hence picking new network abstractions should be done carefully.

A starting point might be to ask whether *packets* are the right abstraction. Since both existing intra- (except for CPU-to-memory DDR3) and inter-server link protocols today use packet-like switched technologies, we believe packets remain the right abstraction. An open question however is whether we would be better served with solutions that allow us to amortize per-packet processing overheads (for reduced latency) such as larger MTUs or a "packet bursts" abstraction.

A second question regards communication *reliability*. Clearly, the resource endpoints must see an end-to-end abstraction of reliable communication, however, it is not clear whether we need reliability at the level of individual network links (as found in intra-server link technologies and some inter-server links such as InfiniBand) or whether end-to-end retransmissions (as used with Ethernet networks) will suffice. Our conjecture is that end-to-end retransmissions should be adequate given the low RTTs we envisage, however this is an important question that warrants more rigorous exploration.

Another related question is whether we need support for bandwidth reservations, or fair resource sharing mechanisms, or whether pure statistical multiplexing with end-to-end congestion control will suffice. There are many calls for reservations and fairness [21, 22, 25] even in existing datacenter networks – if the case for these mechanisms in existing datacenters proves compelling then it is likely to be only stronger in a disaggregated datacenter (since the network's impact on application performance is only greater). We leave exploring the case for such mechanisms and the form of necessary solutions to future work.

4.3 Systems Architecture

The cost of hardware and its maintenance has been the most powerful driving force of datacenter evolution, such as migration from powerful mainframes to commodity servers [8]. We believe that a disaggregated datacenter will be cheaper than the server-oriented architecture, because *i)* the operator has finer-grained control over provisioning decisions, *ii)* disaggregated resources can simplify management complexity, and *iii)* the unified network cuts out a layer of integration (in lieu of the PCIe-Ethernet-PCIe traverse in current server-to-server communication). In some sense, disaggregation is an extreme extrapolation of the streamlining and customization efforts that have been made by the biggest datacenters [4, 10]. Although the cost reduction from disaggregation is hard to quantify at this point, we suspect that cost savings might turn out to be one of the strongest motivations for disaggregated datacenters.

In this paper, we tried to answer if we can disaggregate resources across an entire datacenter. While we are positive that disaggregation is feasible and quite likely going to happen as evidenced by our experiments and the current trends, one question still remains; what is the right scale for disaggregation? Resources can be disaggregated at many different levels, such as server, rack, pod, datacenter, or something else. The answer will depend on the level of savings due to disaggregation and the networking costs, and we will need to quantify this trade-off.

While the VM-as-a-unit assumption made in §2 is a good starting point as it can readily utilize existing software infrastructures, we speculate that disaggregation may enable a more intuitive abstraction for modern datacenter applications. Jobs can be most naturally described in terms of their resource requirements – e.g., "give me 200 CPU cores, 1 TB memory, and 100 Gbps Internet connectivity" – but today application developers and datacenter operators must map their resource demand to the granularity of servers or VMs. One can view disaggregation as changing the abstraction offered by the infrastructure from that of a "pool of servers" to that of a "pool of resources". We believe that the latter offers greater flexibility and will prove to be a more natural and powerful abstraction.

Finally, one avenue ripe for exploration is that of network management for disaggregated datacenters. Instead of a standalone network management solution, we envisage a unified resource management architecture as a combination of the centralized network controller architectures advocated by work on 4D [12] and SDN [13] and the job schedulers found in existing datacenters [14]. This tight integration of network and resource scheduling can enable greater flexibility; for example, a scheduler can seamlessly migrate resources to detour congested links (recall that disaggregation decouples resource usage from its physical location). The design of such unified schedulers is an interesting topic for future work.

5 References

[1] Apache Hadoop. http://hadoop.apache.org/.

[2] HP Moonshot System. http://goo.gl/fteii.

[3] Memcached - a distributed memory object caching system. http://memcached.org/.

[4] Open Compute Project. http://www.opencompute.org/.

[5] PigMix benchmark tool. http://cwiki.apache.org/confluence/display/PIG/PigMix.

[6] SeaMicro Technology Overview. http://seamicro.com/sites/default/files/SM_TO01_64_v2.5.pdf.

[7] M. Alizadeh, S. Yang, M. Sharif, S. Katti, N. McKeown, B. Prabhakar, and S. Shenker. pFabric: Minimal Near-Optimal Datacenter Transport. In *Proc. SIGCOMM*, 2013.

[8] T. E. Anderson, D. E. Culler, and D. Patterson. A case for NOW (networks of workstations). *Micro, IEEE*, 15(1):54–64, 1995.

[9] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the Art of Virtualization. In *Proc. SOSP*, 2003.

[10] L. A. Barroso, J. Dean, and U. Holzle. Web search for a planet: The Google cluster architecture. *Micro, IEEE*, 23(2):22–28, 2003.

[11] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears. Benchmarking cloud serving systems with YCSB. In *Proc. SoCC*, 2010.

[12] A. Greenberg, G. Hjalmtysson, D. A. Maltz, A. Myers, J. Rexford, G. Xie, H. Yan, J. Zhan, and H. Zhang. A clean slate 4D approach to network control and management. *ACM SIGCOMM Computer Communication Review*, 35(5):41–54, 2005.

[13] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker. Nox: towards an operating system for networks. *ACM SIGCOMM Computer Communication Review*, 38(3):105–110, 2008.

[14] B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A. D. Joseph, R. Katz, S. Shenker, and I. Stoica. Mesos: A platform for fine-grained resource sharing in the data center. In *Proc. NSDI*, 2011.

[15] Intel Newsroom. Intel, Facebook Collaborate on Future Data Center Rack Technologies. http://newsroom.intel.com/community/intel_newsroom/blog/2013/01/16/intel-facebook-collaborate-on-future-data-center-rack-technologies.

[16] K. Lim and J. Chang and T. Mudge and P. Ranganathan and S. K. Reinhardt and T. F. Wenisch. Disaggregated Memory for Expansion and Sharing in Blade Servers. In *Proc. ISCA*, 2009.

[17] K. Lim and Y. Turner and J. R. Santos and A. AuYoung and J. Chang and P. Ranganathan and T. F. Wenisch. System-level implications of disaggregated memory. In *Proc. HPCA*, 2012.

[18] Kshitij Sudan, Saisanthosh Balakrishnan, Sean Lie, Min Xu, Dhiraj Mallick, Gary Lauterbach, and Rajeev Balasubramonian. A Novel System Architecture for Web Scale Applications Using Lightweight CPUs and Virtualized I/O. In *Proc. HPCA*, 2013.

[19] Y. Low, J. Gonzalez, A. Kyrola, D. Bickson, C. Guestrin, and J. M. Hellerstein. Graphlab: A new framework for parallel machine learning. 2010.

[20] C. Olston, B. Reed, U. Srivastava, R. Kumar, and A. Tomkins. Pig latin: a not-so-foreign language for data processing. In *Proc. SIGMOD*, 2008.

[21] L. Popa, G. Kumar, M. Chowdhury, A. Krishnamurthy, S. Ratnasamy, and I. Stoica. FairCloud: sharing the network in cloud computing. In *Proc. SIGCOMM*, 2012.

[22] L. Popa, P. Yalagandula, S. Banerjee, J. C. Mogul, and Y. T. J. R. Santos. ElasticSwitch: Practical Work-Conserving Bandwidth Guarantees for Cloud Computing. In *Proc. SIGCOMM*, 2013.

[23] C. Reiss, J. Wilkes, and J. L. Hellerstein. Google cluster-usage traces: format + schema. Technical report, Google Inc., Mountain View, CA, USA, Nov. 2011. Revised 2012.03.20. Posted at URL http://code.google.com/p/googleclusterdata/wiki/TraceVersion2.

[24] S. Rumble, D. Ongaro, R. Stutsman, M. Rosenblum, and J. Ousterhout. It's time for low latency. In *Proc. HotOS*, 2011.

[25] A. Shieh, S. Kandula, A. Greenberg, and C. Kim. Seawall: performance isolation for cloud datacenter networks. In *Proc. HotCloud*, 2010.

[26] S. Soltesz, H. Pötzl, M. E. Fiuczynski, A. Bavier, and L. Peterson. Container-based operating system virtualization: a scalable, high-performance alternative to hypervisors. In *Proc. EuroSys*, 2007.

[27] B. C. Vattikonda, G. Porter, A. Vahdat, and A. C. Snoeren. Practical TDMA for datacenter Ethernet. In *Proc. EuroSys*, 2012.