

# Rack-scale Computing

Edited by

Babak Falsafi<sup>1</sup>, Tim Harris<sup>2</sup>, Dushyanth Narayanan<sup>3</sup>, and David A. Patterson<sup>4</sup>

1 EPFL – Lausanne, CH, babak.falsafi@epfl.ch

2 Oracle Labs – Cambridge, GB, timothy.l.harris@oracle.com

3 Microsoft Research UK – Cambridge, GB, dnarayan@microsoft.com

4 University of California – Berkeley, US

---

## Abstract

---

This report documents the program and the outcomes of Dagstuhl Seminar 15421 “Rack-scale Computing”. The seminar was successful and facilitated interaction between researchers working in a diverse set of fields, including computer architecture, parallel workloads, systems software, and programming language design. In addition to stimulating interaction during the seminar, the event led to a follow-on Workshop on Rack-Scale Computing to be organized during 2016.

**Seminar** October 11–16, 2015 – <http://www.dagstuhl.de/15421>

**1998 ACM Subject Classification** B.3 Memory Structures, C.1.4 Parallel Architectures, D.3 Programming Languages, D.4 Operating Systems

**Keywords and phrases** Rack-scale systems, Parallelism, Computer Architecture

**Digital Object Identifier** 10.4230/DagRep.5.10.35


## 1 Executive Summary

*Babak Falsafi*

*Tim Harris*

*Dushyanth Narayanan*

*Kaveh Razavi*

**License**  Creative Commons BY 3.0 Unported license  
© Babak Falsafi, Tim Harris, Dushyanth Narayanan, and Kaveh Razavi

Rack-scale computing is an emerging research area concerned with how we design and program the machines used in data centers. Typically, these data centers are built from racks of equipment, with each rack containing dozens of discrete machines connected by Ethernet or by InfiniBand. Over the last few years researchers have started to weaken the boundaries between these individual machines, leading to new “rack-scale” systems. These architectures are being driven by the need to increase density and connectivity between servers, while lowering cost and power consumption.

Initial commercial systems provide high-density processor nodes connected through an in-machine interconnect to storage devices or to external network interfaces (e.g., HPE Moonshot, or SeaMicro Fabric Compute). Many ideas are now being explored in research projects – e.g., the use of custom system-on-chip processors in place of commodity chips, the use of emerging non-volatile-memory technologies or stacked Flash in place of disks, and the use of silicon photonics and wireless links for communication within or between rack-scale systems. In addition, researchers are exploring how systems software, language runtime systems, and programming models can evolve for these new architectures.



Except where otherwise noted, content of this report is licensed under a Creative Commons BY 3.0 Unported license

Rack-scale Computing, *Dagstuhl Reports*, Vol. 5, Issue 10, pp. 35–49

Editors: Babak Falsafi, Tim Harris, Dushyanth Narayanan, and David A. Patterson



Dagstuhl Reports

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

This seminar sought to bring together researchers working on different parts of these problems. We structured the seminar around a small number of invited introductory talks (Section 4) accompanied by break-out sessions (Section 5) and a series of four poster sessions. The poster sessions permitted everyone to have an opportunity to present their own work (if they wished to), and enabled many parallel discussions to continue at the same time around different posters.

## 2 Table of Contents

<b>Executive Summary</b>	
<i>Babak Falsafi, Tim Harris, Dushyanth Narayanan, and Kaveh Razavi</i> . . . . .	35
<b>Overview of Talks</b> . . . . .	38
<b>Workloads and Technologies</b> . . . . .	39
SpiNNaker	
<i>Steve Furber</i> . . . . .	39
Data management at rack-scale	
<i>Gustavo Alonso</i> . . . . .	39
Building Hardware for Microsoft's Datacenters	
<i>Leendert van Doorn</i> . . . . .	40
Memory: Past, Present and Future Trends	
<i>Paolo Faraboschi</i> . . . . .	40
Chips in Racks: What are the Trends?	
<i>Boris Grot</i> . . . . .	41
Rack-Scale Storage: Business as Usual or Something Different?	
<i>Ant Rowstron</i> . . . . .	42
<b>Breakout Sessions</b> . . . . .	43
Research Challenges . . . . .	44
Research Tools for Rack-Scale Computing . . . . .	45
Rack-Scale Networking . . . . .	46
Architectural Support . . . . .	46
Operating Systems . . . . .	47
High Performance Computing . . . . .	47
<b>Participants</b> . . . . .	49

### 3 Overview of Talks

In the near future we expect to see rack-scale computers with 1000s of cores, terabytes of memory, high-bandwidth and low-latency internal fabrics. These new architectures raise several interesting research questions. What are the trade-offs between the different approaches being explored? How should the interconnect fabric be organized, and how should CPUs, DRAM, and storage be placed in it? Are different kinds of rack-scale systems required to handle different kinds of workloads efficiently? How should we integrate rack-scale computers into data center networks and warehouse-scale computers (WSCs)? To what extent should rack-scale machines be programmed as large shared-memory NUMA servers, or as traditional distributed systems, or a combination of the two? What are the correct communication primitives to let applications benefit from low-latency communication within the system? What are the likely failure modes and how do we achieve fault tolerance? How can researchers effectively prototype and test novel ideas?

Specifically, we sought to involve researchers and practitioners working on:

- **Systems-on-Chip.** SoCs are used both in industrial and research rack-scale systems, motivated by a drive for high density and high performance-per-Watt. For instance, Intel Atom C2000 processors (used in HPE Moonshot servers) provide up to eight 64-bit cores, a dual-channel memory controller, integrated PCIe, four GbE ports, along with SATA and USB. The UC Berkeley FireBox is a 50kW WSC building block containing a thousand compute sockets, each containing a SoC with around 100 cores connected to high-bandwidth on-package DRAM, and fast SoC network interfaces.
- **Interconnects.** A motivation for many rack-scale systems is to support workloads that benefit from more fine-grained communication than is possible between machines in traditional data centers. This includes developments at the physical level (for instance, silicon photonics). In addition, it includes development in networking, and in how the interconnect is exposed to software to enable high bandwidth communication or high message rates. For instance, the Scale-Out NUMA architecture exposes the interconnect via a remote memory controller which is mapped into a node's local cache-coherent address space. This removes the latency of crossing interfaces such as PCIe when communicating across a rack-scale system.
- **Storage systems.** There are many different technologies emerging for non-volatile random-access memory (NV-RAM), some promising high-capacity non-volatile storage coupled with read performance comparable with DRAM. At the same time, other researchers are exploring techniques for 3D-stacking FLASH and DRAM.
- **Systems software and language runtime systems.** How should operating systems handle rack-scale systems, and schedule work on them? Should a single operating system run over a complete system, as in today's large NUMA machines, or should separate operating systems run on each core, or each socket, or at some other granularity. How do we expose locality concepts, and what problems do we solve in hardware versus software. Experience with multi-core research operating systems, such as Barrelfish, fos, and Tessellation is relevant here. At a higher level, to what extent should rack-scale systems be programmed using the abstractions developed for distributed computing (such as replicated objects, actors, or message-passing), or via new implementations of the abstractions used for shared-memory (such as task-parallel programming models, or transactional memory). How are hardware failures, and tolerance of tail-latency accommodated in programming models?

The goal of this Dagstuhl research seminar was to bring together leading international researchers from both academia and industry working on different aspects of rack-scale systems. Effective solutions will require renewed collaboration across architecture, systems, and programming language communities. In addition, we sought to involve participants with practical experience of workloads, and of running industrial warehouse-scale systems.

## 4 Workloads and Technologies

In this report, we briefly summarize the seminar's introductory talks on some of the workloads and technologies relevant to rack-scale computing.

### 4.1 SpiNNaker

*Steve Furber (University of Manchester, GB)*

License  Creative Commons BY 3.0 Unported license  
© Steve Furber


The SpiNNaker project aims to implement a massively-parallel system incorporating a million ARM processor cores for modeling large-scale systems of spiking neurons in biological real time. The 2D toroidal mesh must be folded into 10 racks with 3,600 high-speed interconnect cables, with no single cable longer than 1 meter. A purpose-built machine room with 100kW cooling capacity has been built to house the machine. This talk described the engineering challenges of constructing the machine to a modest budget.

#### Discussion notes

- Verifying that the hardware is doing what the spec says is a challenge (the hardware is non-deterministic).
- Power density and heat removal are also major challenges.
- Analogies between neuromorphic computing and a biological brain from a capability perspective are difficult. At best one can compare complexity/density of computational elements.

### 4.2 Data management at rack-scale

*Gustavo Alonso (ETH Zürich, CH)*

License  Creative Commons BY 3.0 Unported license  
© Gustavo Alonso

The killer application these days is big data, which translated to something practical means processing and managing large data collections of a variety of types (graphs, records, streams, tables, files, images, etc.). The platform of choice these days is a rack, either as a single unit or combined into a larger cluster. Racks are needed in the context of big data because of I/O and/or because of the need to support complex, concurrent workloads.

This talk focussed on these two aspects: I/O and workloads. Often forgotten in research papers, these are typically more important than CPU capacity and pose the real challenge in designing rack-scale data management systems.

### 4.3 Building Hardware for Microsoft's Datacenters

*Leendert van Doorn (Microsoft Corporation – Redmond, US)*

License  Creative Commons BY 3.0 Unported license  
© Leendert van Doorn


This talk briefly described some key workload criteria and design constraints for Microsoft's internal and external datacenters and how those shape our hardware designs. A key element of this is how to control the server cost. The talk specifically covered a set of new silicon technologies that are on the 3-6 year horizon that can help datacenters but also pose new questions. The talk ended with a case for disaggregated resource/rack scale designs and their challenges.

#### Discussion notes

- The design cycle for cloud infrastructure is roughly six months.
- There are two workloads of interest:
  1. First-party data analytics which leads to high utilization on bare metal.
  2. “Lift and shift” (virtualization) exhibiting low utilization but with QoS guarantees.
- Regarding supply chain, a key question is whether one size fit all in terms of compute/network/storage? Mostly yes, but premium hardware is useful for special workloads.
- Power is the limiting resource, not the compute density as per conventional wisdom. Racks are sometimes power-limited and not fully populated. Rack height is standardized (e.g., 48Us).
- The unit of failure is a rack. There is replication across clusters and geo-locations.
- How does the cost break down? 30% CPU, 70% the rest of the system (memory, SSD/HDD, networking, ...). DRAM price margins are so low that DRAM prices can not be publicized.
- Future: every SoC will include everything: cores, lots of memory, fast NICs, switches, and accelerators. There will be memory stacking and high-bandwidth serial links. How should one best take advantage of emerging memory technologies? How should we organize cache hierarchies and to what extent do we need NUMA?

### 4.4 Memory: Past, Present and Future Trends

*Paolo Faraboschi (HP Labs – Palo Alto, US)*

License  Creative Commons BY 3.0 Unported license  
© Paolo Faraboschi

The talk surveyed the trend in memory technology and architecture, starting from the historical importance of DRAM in today's computing systems, all the way to future trends. It covers technology and business aspects that highlight the limitations and bottlenecks of the memory architecture we build in computing systems. The talk also covered the transition

to non-volatile memory that is expected to happen in the near future, and the implication on the memory architecture organization.

### Discussion notes

- Intel Developer Forum keynote 2000 – Quiz: Who was the guest who asked for more memory capacity to fit the whole web in DRAM? Answer: Larry Page.
- DRAM is the main medium for storing data. And over time the cost of memory has been going down constantly and reduced significantly. Zooming in the last few years (2000–today) we observe complete flattening of the cost of DRAM (\$6–\$10 per GB) and NAND (\$0.3–\$0.5 per GB).
- Memory is currently enslaved to the CPU memory channels; should we re-architect the way we place the memory? We need more system capacity, more chip density, bandwidth, byte addressability, more near-data intelligence, etc. Let’s see some of the trends out there:
  - More bandwidth (beyond DDRx): serially attached Memory (Micron HMC). Two-tier applicability: (1) performance and (2) capacity. But currently the cost of HMC per GB is huge and therefore it is impractical to integrate it at scale.
  - Another alternative for getting more bandwidth is the 2.5D-3D integration. 2.5D packaging on Si interposer, 3D die stacking, or you can also go for system-in-package: combination of 3D die stacking and 2.5D interposer integration. Is this the solution? Well, yes, maybe. But stacking things like this (8x93%) results in a yield of 55%.
- There are three technology candidates (all NV): PC-RAM, STT-RAM, R-RAM and they all have different properties. The most important property is the density. If you don’t get that one right, the cost is going to be huge and nobody is going to use it. And all these technologies are covering a wide-range of reading/writing latencies, and no energy cost for maintaining state (unlike DRAM). And of course these technologies have different maturity levels but we are not going to focus on that for now.
- There is another way to put it together: maybe I can put some memory in a fabric. Emancipated memory (outside of the control of the CPU), you have it in another fault domain and there are many other things you can do with it... But how much can one scale the memory fabric? What is the cost of the fabric? What about the latency?

## 4.5 Chips in Racks: What are the Trends?

*Boris Grot (University of Edinburgh, GB)*

License  Creative Commons BY 3.0 Unported license  
© Boris Grot

The talk covered emerging trends in hardware for rack-scale computing. These trends include emergence of specialized many-core datacenter processors, pervasive accelerators, high-bandwidth memory technologies, and low-latency rack-scale fabrics. While individually, these technologies mitigate specific performance and energy-efficiency bottlenecks of today’s systems, effectively leveraging the entire ensemble in a rack-scale deployment calls for potentially significant modifications to the system and application software stack.

**Discussion notes**

- Data is outgrowing compute power, 2% of world power is being spent in data centers.
- What is in a Xeon processors? Large LLC, few fat cores. LLC is not helpful for many workloads (100W).

**Data center trends**

1. Server processors.
  - Q: Why have you married hw/sw w.r.t. coherence? Why not let the OS handle this?
  - A: OS can if it wants; scale-out applications do not really need coherence
  - Q: I wouldn't agree with the power argument (w.r.t. LLC). There is new technology that can replace LLC.
  - A: These technology are for out of CPU.
2. Accelerators (GPU vs. FPGA) will help improve efficiency with the slowdown in Dennard Scaling.
3. Goodbye bandwidth wall – HMC is 10x faster than DDR3.
  - Q: But HMC is small?
  - A: That is not fundamental. we can have higher density in the future.
  - Q: SerDes links may provide high bandwidth but the links need to be always on, and dissipate much power?
  - A: Yes
4. Very fast intra-rack network – low energy.
  - Q: But these photonics need transceivers (high energy)?
  - A: No, laser on chip, so no need for transceivers. The audience was not unanimous about this.
5. Low-latency communication (integrated fabrics, Scale-out NUMA).
  - Q: Point-to-point communication is not always the best for all workloads (e.g., graph processing). You want a lower hop-count due to the random nature of the workload.
  - A: I agree, but these provide ample benefits for load-balancing.
  - Q: I think having more hardware resources makes the problem on the application side harder. For example, it is really hard to write an application that can saturate 400 GB/s of bandwidth.
  - A: Our point is that, changing the software to scale it with the number of cores is hard. Instead we argue that if you application runs fine with one pod of cores, by scaling the number of pods (i.e., partitioning the cores), the applications can scale easier without modification.
  - Q: It is really hard to increase the memory bandwidth.
  - A: We actually agree with that. That is why we argue for partitioning the bandwidth.

**4.6 Rack-Scale Storage: Business as Usual or Something Different?**

*Ant Rowstron (Microsoft Research UK – Cambridge, GB)*

License  Creative Commons BY 3.0 Unported license  
© Ant Rowstron

This talk shed some light on what rack-scale computing means in general and what this means for storage specifically. The key is a converged rack-scale design – where all resources are



carefully provisioned and managed to achieve a specific performance or price point. The talk used public cloud storage as the motivation, and MSR's experiences building out rack-scale storage for the cloud. The talk highlighted some ways forward and also pointed out some of the many pitfalls that the designers experienced.

### Discussion notes

- Currently the unit of deployment is 1U-4U.
- Key principle: reduce \$/GB given a certain performance – e.g., BlackBlaze backup: 4U, 45 HDD in JBOD.
- Latency (ms. . .hours) vs. Access pattern (hot – nearline – cold) vs. cost.
- Capacity vs. IOPS vs. latency vs. cost.
- Pelican: capacity for cold data (seconds latency) at a price point of tape. 8% of drives active at a time, power and cooling provisioned for those drives. Software enforces constraints: hard (power/cooling/vibration) and soft (bandwidth).
  - Q: What is the difference between temperature between top to bottom of the rack?
  - A: About 20 degrees, air comes at about 20C and goes out at 40C.
  - Q: Effect of vibration due to fans?
  - A: The way we do the layout we reduce the number of fan.
  - Q: But there is also ambient noise?
  - A: We have not seen any of that.
  - Q: Is this a block interface for the rest of system?
  - A: Think of it as blob service.
  - Q: Rack is a unit of failure, this is not true here. . .
  - A: We can fail over between servers and drives.
  - Q: What percentage of storage would benefit from this?
  - A: I think this is a significant fraction. Things like OneDrive, etc.
- Taming hardware was hard. ASIC designers like to add feature. We had to fix bugs in silicon that can take up to 6 months to fix. So we have to push complexity to software as much as possible. Hence we need tools for doing this.
  - Q: Do you provide replication, availability inside the rack?
  - A: Yes, we do some. But we have to think cross rack replication as well.
  - Q: This is archival. So most of your bandwidth is going to serve writes
  - A: We see usually multiple reads at a time.
  - Q: Maybe you should do scheduling at multiple rack levels to reduce latency?
  - A: Yes, that is a possibility.
  - Q: What is the role of tiering in the rack-scale system? What is the proper way of moving things between tiers?
  - A: We have to see whether we can have racks that can work in different tiers. If not, we have to design them based on a certain throughput.

## 5 Breakout Sessions

Much of the time during the seminar was devoted to breakout sessions. We briefly summarize the main topics of discussion which arose.

## 5.1 Research Challenges

Below, we describe rack-scale research challenges identified in different domains. This section is formed mostly in terms of open-ended questions or proposals.

### Network Topology

- Should the interconnect be electrical, optical, or hybrid?
- It depends on the maturity of the technologies. How fast are these technologies going to mature?
- Which topologies should we use? What are the trade-offs that we want to make (latency vs. bandwidth)? Does the topology even matter?
- We need reasonable benchmarks for evaluating rack-scale networks.

### Rack-Scale Memory

- What is the impact of NVM on the OS?
- Is it going to be more like a cluster or NUMA?
- We need to be able to control: the movement of data between volatile and non-volatile state and ordering of memory operations. Sync vs. async.
- Alternate/relaxed consistency – entry, release?
- Should systems use explicit messaging or implicit read/write?

### Power

- Tradeoffs are between the cooling of rack and what goes inside the rack.
- Up to 20 KW per rack including cooling is reasonable today.

### Rack Organization

- What is disaggregation?
  - Is memory close to processing?
  - Rack-podded memory (“emancipated”) is attractive for large working sets. Is good latency/bandwidth possible?
  - Disaggregation  $\Rightarrow$  virtualization  $\Rightarrow$  Cost savings.
  - Trade-offs:
    - \* Cost of “infinitely fast” interconnects.
    - \* Cost of porting or reimplementing legacy apps.
- What is a server that I can buy?
  - Is it a node or a collection of nodes?
  - A node is a heterogeneous set of CPU/RAM/Storage/Network.
- Active memory components may help the disaggregation (e.g., RDMA).

### Availability/Reliability

- What is the failure domain in a disaggregated world?
- MTBF for virtualized resources  $\neq$  for physical resources.
- What about bit rot in DRAM/NVM?
- Byte-addressable NVM programming models remove the middleman  $\Rightarrow$  faulty programs.
- Replication is required for storage class reliability which increases the requirements on the interconnect bandwidth.

- Local versus global versus remote  $\Rightarrow$  Need remote copy to survive local failure.
- What is the role of OS? Authentication and authorization, setting up mappings, sharing, etc.

### Security

- We need enclave-like approaches to encrypt everything but the CPU caches.
- The security of new operating systems, file systems, etc. needs to be verified.
- The identity of (disaggregated) resources needs to be verified to avoid spoofing.
- Fine-grained protections versus coarse-grained address spaces. Lessons learned with CHERI using variable-length segments protected through capabilities are applicable here.
- What are the OS integration issues with capabilities?
- End-to-end security for rack-scale.
  - Compute, network, storage.
  - Key management.
  - Multi-tenancy.

## 5.2 Research Tools for Rack-Scale Computing

### Benchmarks

We need new benchmarks and evaluation methodologies for rack-scale computers. The evaluation requires workloads over small and large datasets.

### Debugging

For debugging, we will likely still rely on logging. On top of that, support for tracing and trace inspection will be instrumental. In addition, we require support from hardware for synchronized clocks and to be able to inspect remote memory, tap into the network, and replay. We also need to build tools to identify coherence errors across nodes, deadlocks, live locks, lost packets, and messages.

### Performance

We require additional information on top of what commodity hardware and networks provide. For example, Intel PT-like tracing can allow us to understand performance behavior of rack-scale applications. We need to be able to identify the number of hops a packet traverses, traffic information between A and B, and in general be able to query for locality information.

### Simulation

Rack-scale computers can be simulated at the different levels. There are possibilities for architecture simulation as well as network simulation. FPGAs are a good test-bed for running these simulations that involve a large number of events. Further, rack-scale computers can be emulated. Emulation of slowness is possible by adding delays, and emulation of fastness is possible by making everything else in the system slower. We require test-bed prototypes for rack-scale research. We either need to build proofs or use small machines and project the results to larger machines.

### 5.3 Rack-Scale Networking

There are certain advantages to rack-scale networks. They allow for significant cost savings by allowing resource disaggregation. Further, they make it easy to ship new technology inside the rack and hiding it. How is the rack seen from the rest of the world? Is it going to be an open-box or a black-box, a single computer or a collection of computers? The rack is already a shipping unit. Note that the answer to these questions has implication for naming and failure domains.

Do we need to have two networks (one for management tasks) or is one enough? Also, another important aspect is the choice low-radix versus high-radix switches. Low radix switches (potentially embedded in SoCs) are better in terms of economy of scale but have higher latencies.

### 5.4 Architectural Support

#### Synchronization

We require atomically ordered multicast for implementing locks, transactions, etc. It can be implemented as a separate bus in a rack-scale computer. Spinning should be banned, but we need a high-performance wakeup.

#### Communication

The rack network needs to support multicast, provides high packet injection rate with low latency and high throughput. Connection-less RDMA or migratable RDMA connections to scale RDMA communication is desired. We need to develop efficient routing and congestion control inside the rack. The network needs to support virtualization for handling multiple users and migration. It further needs to provide programmable wake-ups on various events.

#### Fault tolerance

The rack needs to forward error correction on communication channels. It further needs to provide support for observing error rate counters and allow for isolation of components. We need support for replication and migration of memory in face of failures. Failure detection and/or prediction can be implementation as a stand-alone service.

#### Caching

Software needs control over local caches. Rack nodes need to provide support for cache specific load instructions to potentially more than one L1 cache (similar to scratch-pads). Variable-sized cache-lines and programmable prefetch engines are also desired in a rack-scale computer.

#### Accelerators

Accelerators allow for heterogeneous processing that allows for power efficiency. These accelerators need to support multiple processes. Accelerators that reduce data movement (i.e., near memory processing) and allow for programmable I/O are desired. Integrated FPGAs on the sockets can bring flexibility for processors in a rack.

## QoS

The rack should provide strong QoS throughout the rack in order to meet SLAs. Introspection support will allow finding hot-spots using performance counters.

## 5.5 Operating Systems

The OS should provide the following set of services for a rack-scale computer:

- Failure detection.
- Location-transparent communication.
- Resource allocation/QoS.
- State machine replication as a service.

The hardware can provide these services to make it easier for the OS mechanisms:

- Warning before failure.
- Clearly defined “fault containers”: memory, threads, etc. but also at the language level.
- Are these just transactions?
- Turn suspected failure into fail stop.

Resource allocation: Disaggregation gives great opportunities for scheduling/migrating/consolidation/dealing with failures/shared resources. But it adds complexity. Can we have a single level of scheduling? Is there anything we can do to make it simpler? Hierarchical versus “simple” scheduling. Can we do traditional OS tasks centralized or not? This generalizes to all the resources and sharing of all resources.

Can we do sharing via exclusive ownership of state? Partition and migrate the state and the ownership. The challenge is how to quickly and correctly change ownership. Shared/global abstractions with different implementations depending on the hardware? If IaaS, then OS layer can be very simple. If PaaS then containers? Either way, performance isolation and QoS are key. Memory bandwidth will be a new challenge to provide QoS when it is dis-aggregated. Scale will also make it harder.

Communities need to work together: we are talking to each other. Should we replicate functionality? Observation: we do already replicate functionality, can we get rid of this (VMM, OS, JVM). Why do you need to virtualize at all? Should we just do containers? A lot of people liked the idea of exporting info between DB and OS and doing affinity scheduling.

Resource monitoring on a large scale. How do you debug it? It’s a distributed system. Not just globally but within each application.

## 5.6 High Performance Computing

### HPC versus general-purpose computing

HPC systems resemble rack-scale computers to some extent. 99% of rack-scale computers are built to run multi-purpose applications and their scale is often a few large machines. HPC systems are rarely a success except for science for which they are built for, but the technology of the ones that are successful are transferred into general-purpose computing. Examples include GPUs, RDMA, and fat-tree topologies.

**Programming model**

MPI is often used as a library to program communicating applications on a HPC system. MPI is not a programming model, but provides a communication abstraction analogous to sockets. It is similar to assembly in which many programming models have been built on top of it. These models, however, are slow compared to native MPI, hence everyone instead uses MPI natively.

MPI is slowly moving away from message passing. One-sided put/get operations combined with fences allow for efficient transfer of data. You can think of it as a new ISA for programming HPC systems. There exists an axiomatic formal memory model that provides a very loose consistency model.

**Reliability/Faults**

In the HPC world, when faults happen the application is either started from the beginning or resumed from a checkpoint if the system supports checkpoints. Instead, in the data center world partial failures in stateful applications happen. ACID, Paxos, and alike are developed to deal with this. It would be interesting to investigate what happens with libfabric (a commonly used HPC library for communication) in face of partial failures.

**Topologies**

HPC networks focus on cheaper topologies. The main metric is the total global throughput on random uniform traffic. Low diameter topologies such as Dragonfly where groups of nodes are fully connected with full connection between groups perform well with this metric. These topologies however are not very modular: we cannot connect extra nodes with local changes, so the data center model of “wheel in an extra container and plug in electricity/water/power” does not directly apply.

## Participants

- Gustavo Alonso  
ETH Zürich, CH
- Yungang Bao  
Chinese Academy of Sciences –  
Beijing, CN
- Angelos Bilas  
FORTH – Heraklion, GR
- Peter Corbett  
NetApp – Sunnyvale, US
- Paolo Costa  
Microsoft Research UK –  
Cambridge, GB
- Christina Delimitrou  
Stanford University, US
- Felix Eberhardt  
Hasso-Plattner-Institut –  
Potsdam, DE
- Lars Eggert  
NetApp Deutschland GmbH –  
Kirchheim, DE
- Babak Falsafi  
EPFL – Lausanne, CH
- Paolo Faraboschi  
HP Labs – Palo Alto, US
- Christof Fetzer  
TU Dresden, DE
- Steve Furber  
University of Manchester, GB
- Jana Giceva  
ETH Zürich, CH
- Matthew P. Grosvenor  
University of Cambridge, GB
- Boris Grot  
University of Edinburgh, GB
- Hermann Härtig  
TU Dresden, DE
- Tim Harris  
Oracle Labs – Cambridge, GB
- Maurice Herlihy  
Brown Univ. – Providence, US
- Matthias Hille  
TU Dresden, DE
- Torsten Hoeffler  
ETH Zürich, CH
- Konstantinos Katrinis  
IBM Research – Dublin, IE
- Kimberly Keeton  
HP Labs – Palo Alto, US
- John Kim  
KAIST – Daejeon, KR
- Christoph M. Kirsch  
Universität Salzburg, AT
- Sergey Legtchenko  
Microsoft Research UK –  
Cambridge, GB
- Martin Maas  
University of California –  
Berkeley, US
- Sue Moon  
KAIST – Daejeon, KR
- Andrew W. Moore  
University of Cambridge, GB
- Dushyanth Narayanan  
Microsoft Research UK –  
Cambridge, GB
- Jörg Nolte  
BTU Cottbus, DE
- Mark H. Oskin  
University of Washington –  
Seattle, US
- Simon Peter  
University of Washington –  
Seattle, US
- Andreas Polze  
Hasso-Plattner-Institut –  
Potsdam, DE
- Danica Porobic  
EPFL – Lausanne, CH
- Zoran Radovic  
Oracle – Stockholm, SE
- Kaveh Razavi  
VU University Amsterdam, NL
- Randolph Rotta  
BTU Cottbus, DE
- Ant Rowstrom  
Microsoft Research UK –  
Cambridge, GB
- Stefan Schmid  
TU Berlin, DE
- Bernhard Schröder  
Fujitsu Technology Solutions  
GmbH – Paderborn, DE
- Malte Schwarzkopf  
MIT – Cambridge, US
- Liuba Shrira  
Brandeis Univ. – Waltham, US
- Jens Teubner  
TU Dortmund, DE
- Gael Thomas  
Télécom & Management  
SudParis – Evry, FR
- Jana Traue  
BTU Cottbus, DE
- Leendert van Doorn  
Microsoft Corporation –  
Redmond, US
- Haris Volos  
HP Labs – Palo Alto, US
- Bernard Wong  
University of Waterloo, CA
- Noa Zilberman  
University of Cambridge, GB
- Ferad Zylkyarov  
Barcelona Supercomputing  
Center, ES

