COMSM0010 Cloud Computing

Lecture 03

Economics of Cloud

Dave Cliff

Department of Computer Science
University of Bristol

csdtc@bristol.ac.uk

Le menu du jour

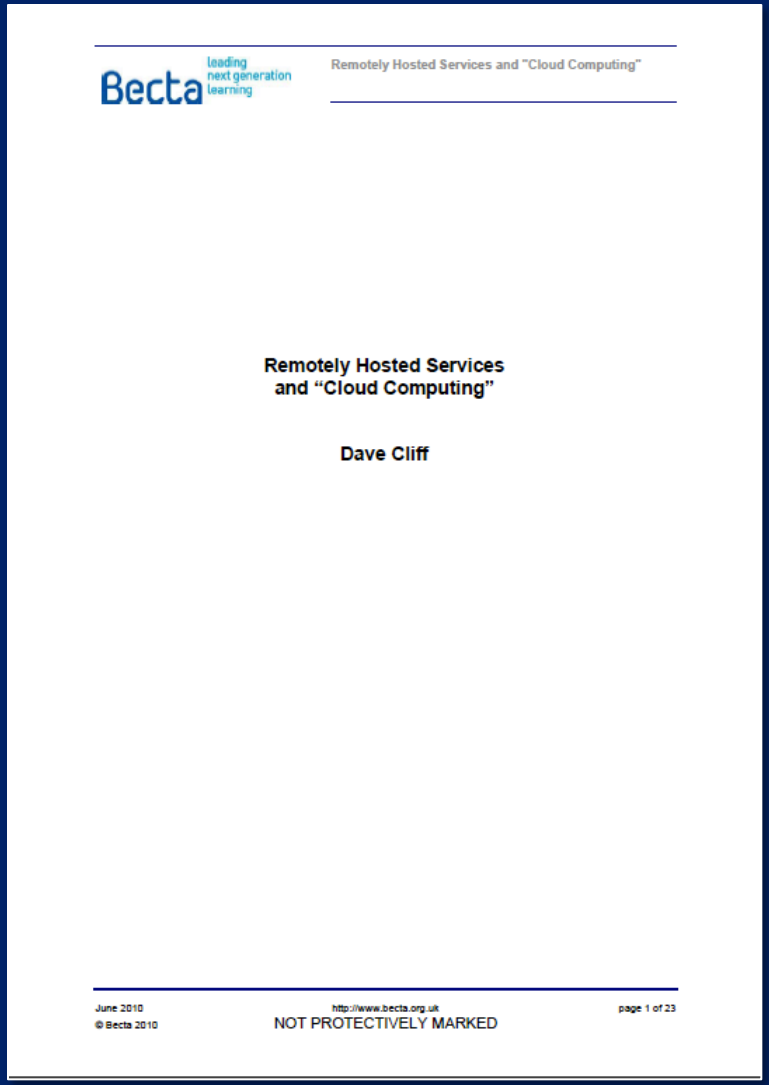
50 minute tour through elements/elementary economics of cloud

From L01: let me tell you a story…

http://www.economist.com/displaystory.cfm?story_id=2352183

For a non-technical introduction…

http://dera.ioe.ac.uk/1514/1/becta_2010_emergingtechnologies_cloudcomputing_report.pdf



Becta leading next generation learning

Remotely Hosted Services and "Cloud Computing"

Remotely Hosted Services and "Cloud Computing"

Dave Cliff

June 2010
© Becta 2010

http://www.becta.org.uk
NOT PROTECTIVELY MARKED

page 1 of 23

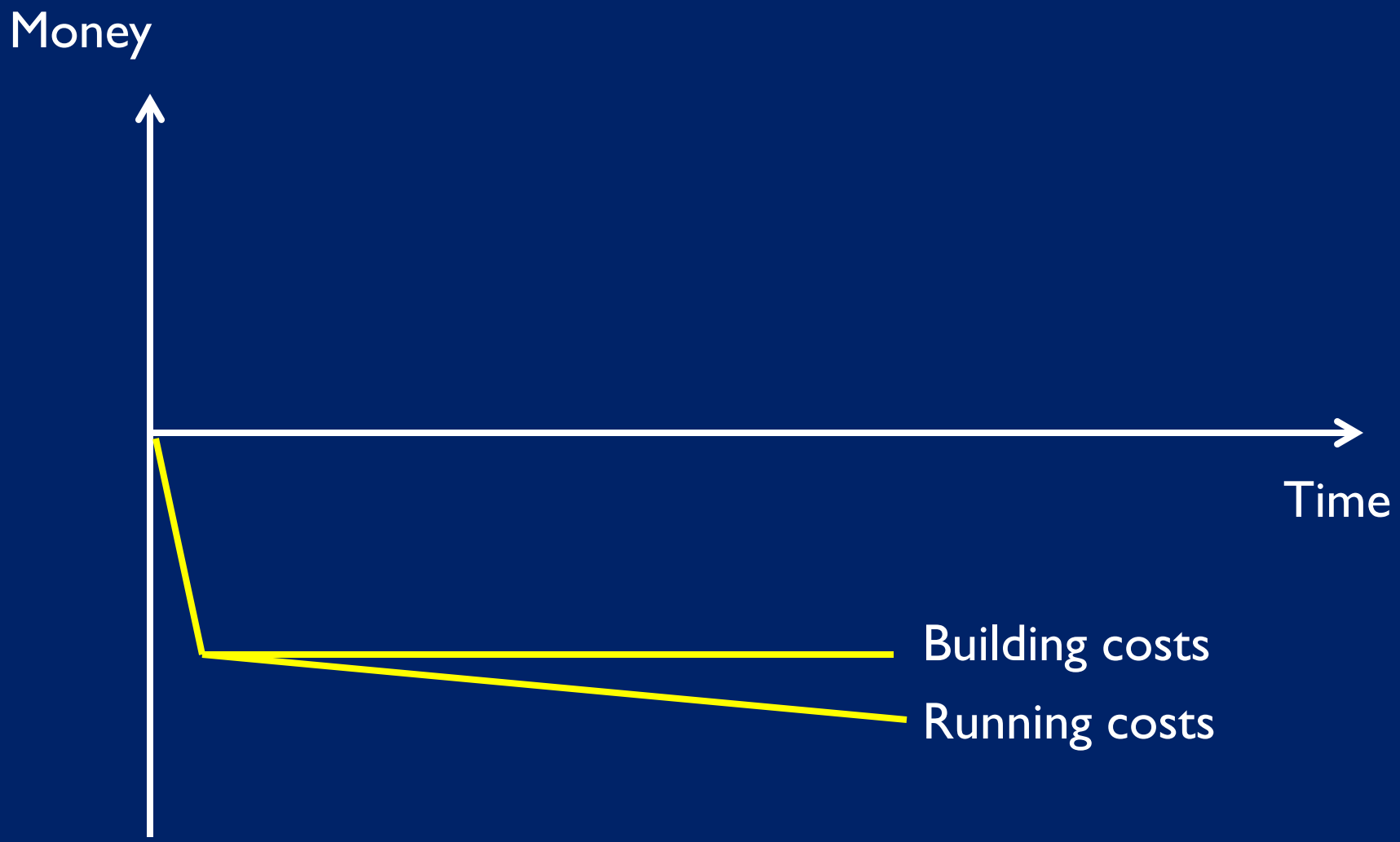
The basic economics…

The Basic Economics…

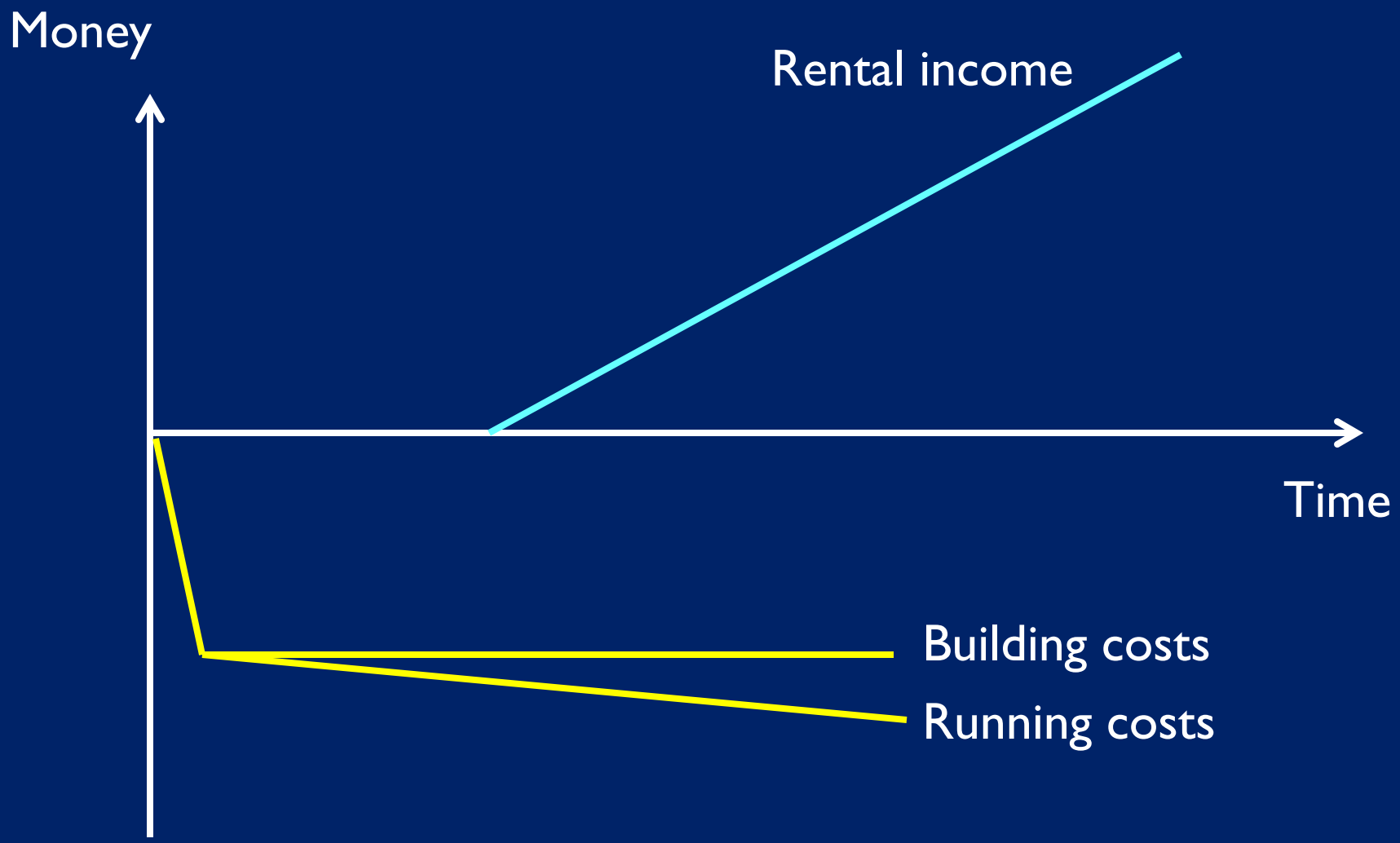
The Basic Economics…

The Basic Economics…

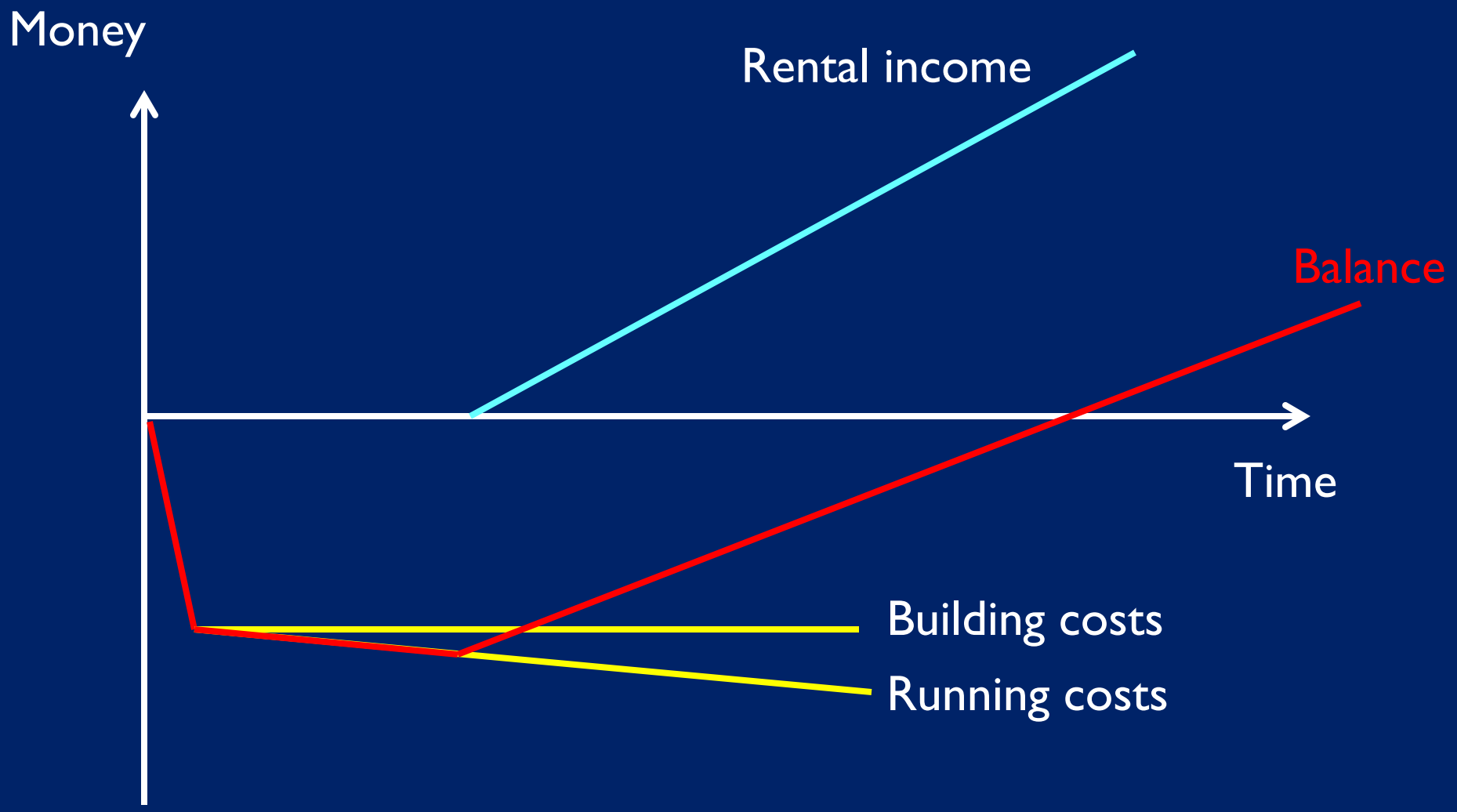
The Basic Economics…



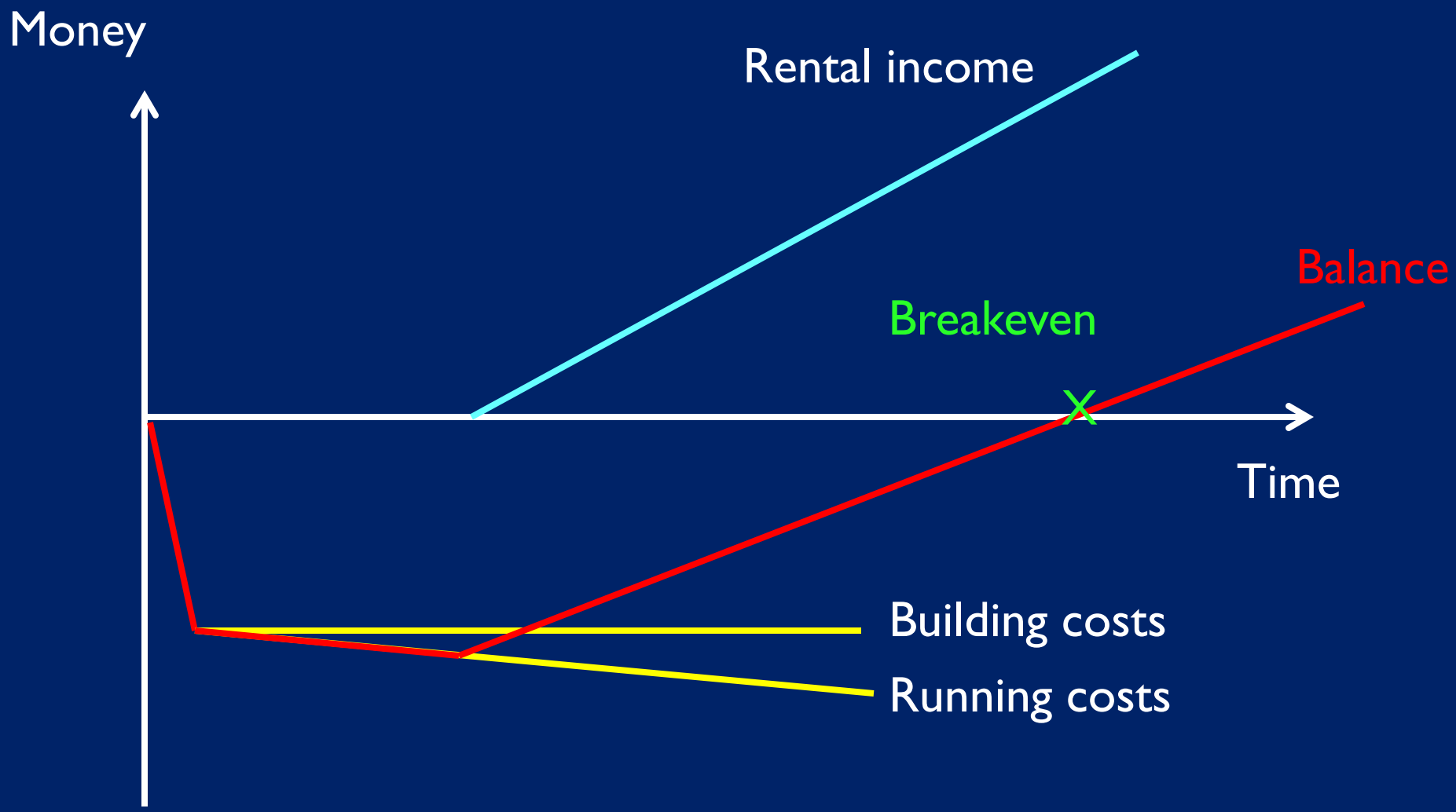
The Basic Economics…



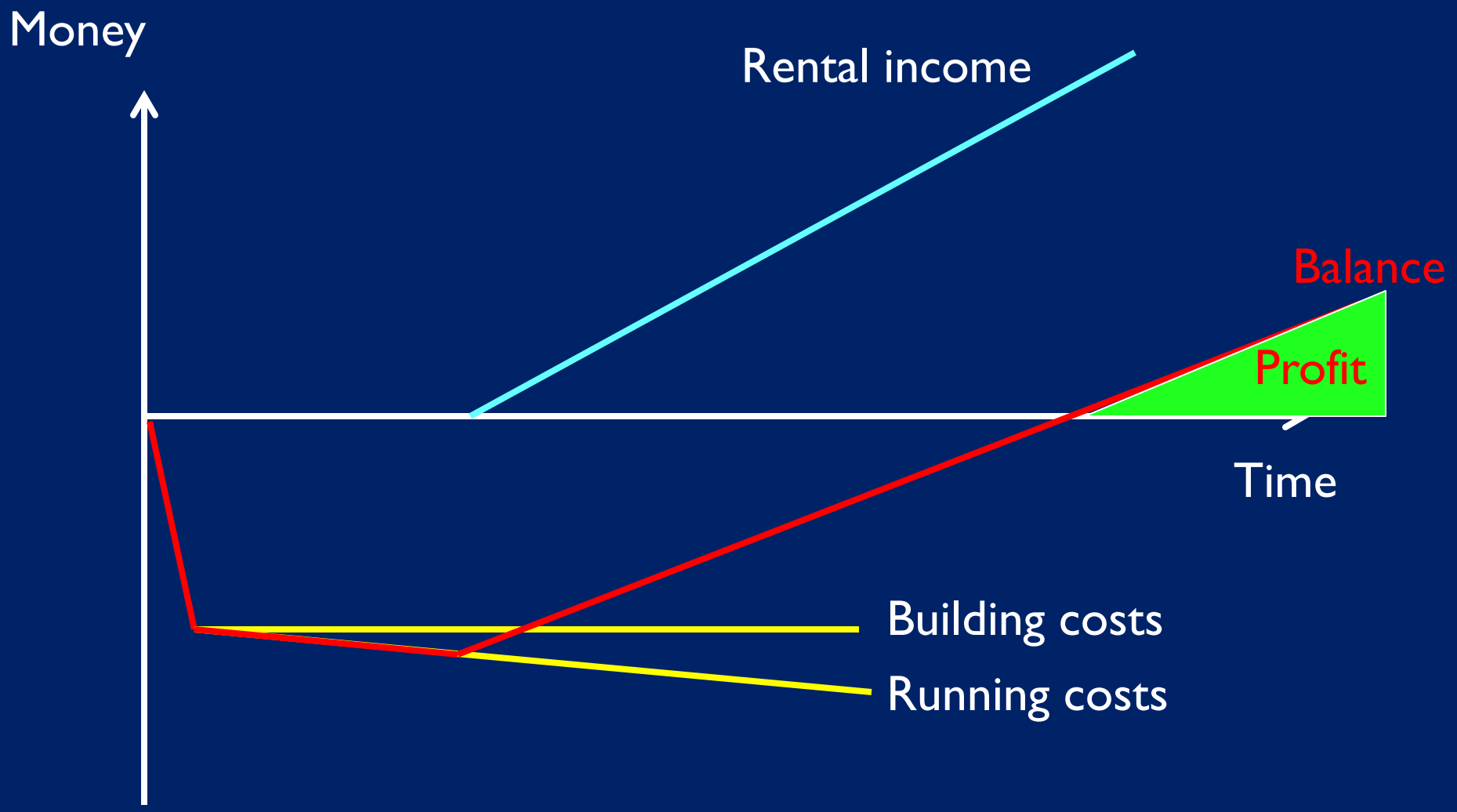
The Basic Economics…



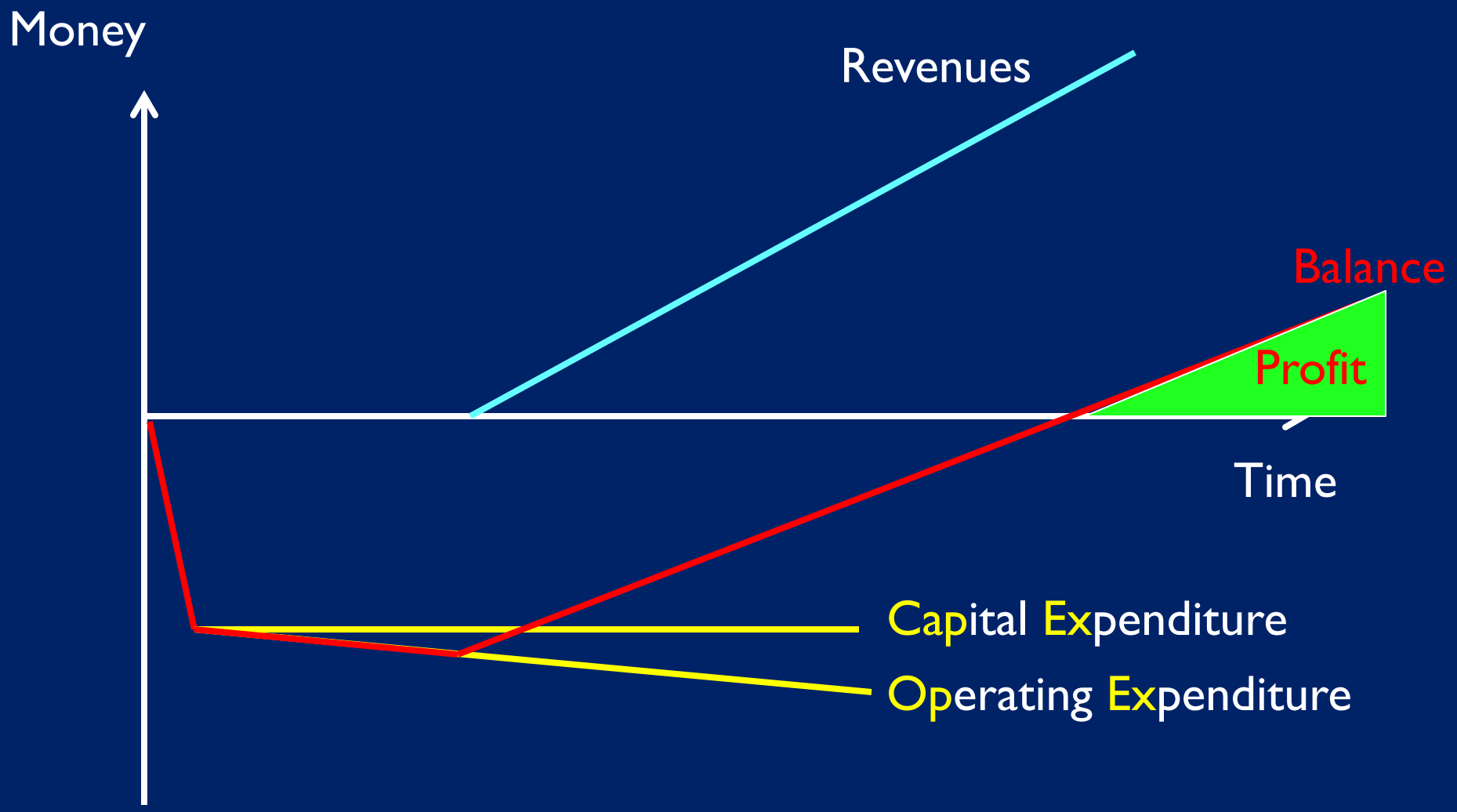
The Basic Economics…



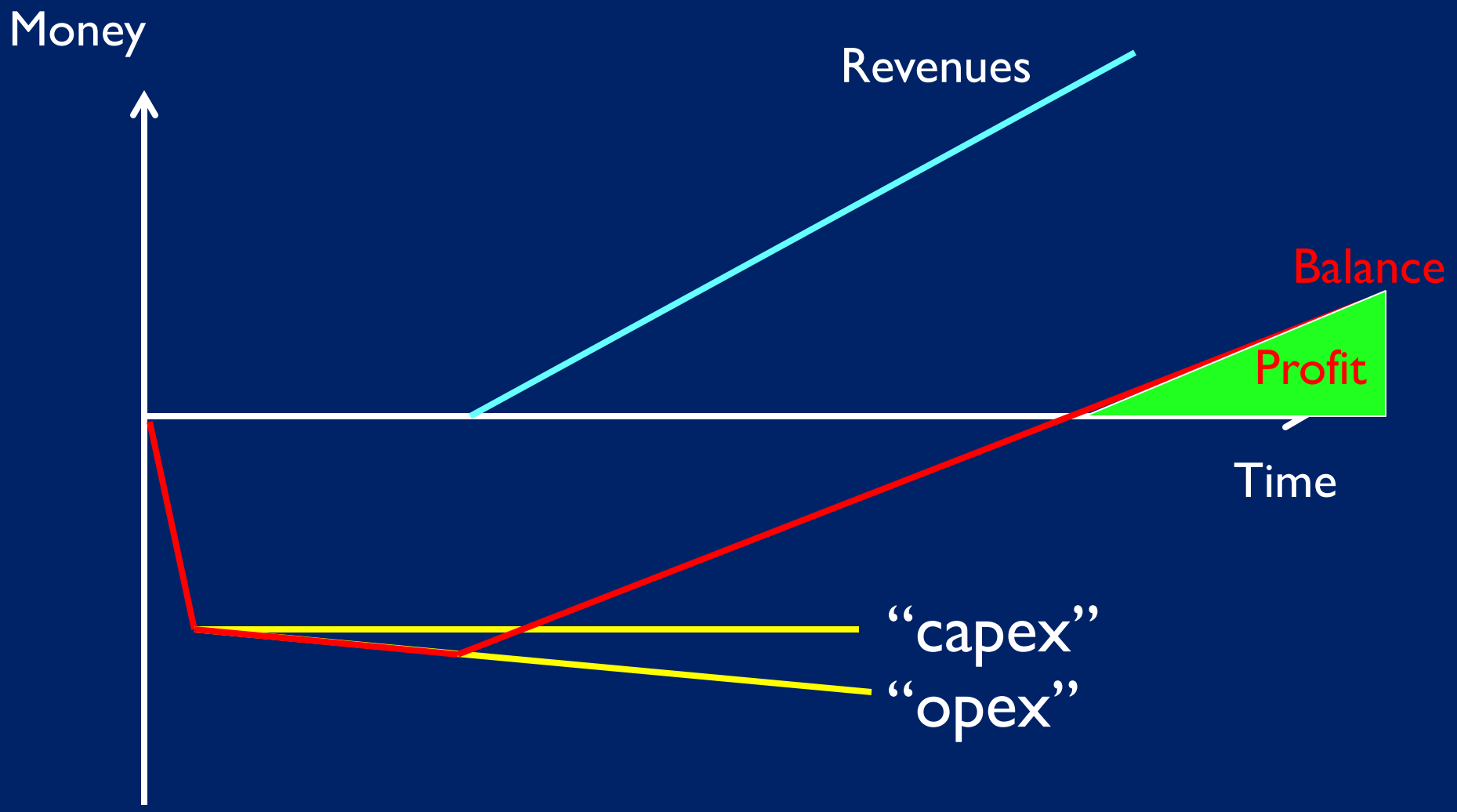
The Basic Economics…



The Basic Economics…



The Basic Economics…



The Basic Economics…

That explained the economics of provision
What about the consumers? Why buy cloud services?

It's again down to capex vs opex…

"Why buy a cow if all you need is milk?"

Only pay for the capacity you need, when you need it
1 CPU for n hours vs n CPUs for 1 hour

Let's look at some real data, from AWS

First, some background on Amazon Web Services (AWS)

AWS instance types – c.2012 (now there are lots more)

(NB: one EC2 Compute Unit (ECU) approx the CPU capacity of a 1.0-1.2GHz 2007Opteron/Xeon)

Standard Instances (well suited for most applications)

- Small Instance (Default): 1.7 GB memory, 1 EC2 Compute Unit (1 virtual core with 1 EC2 Compute Unit), 160 GB instance storage, 32-bit platform
- Large Instance: 7.5 GB of memory, 4 EC2 Compute Units (2 virtual cores with 2 EC2 Compute Units each), 850 GB of instance storage, 64-bit platform
- Extra Large Instance: 15 GB of memory, 8 EC2 Compute Units (4 virtual cores with 2 EC2 Compute Units each), 1690 GB of instance storage, 64-bit platform

High-Memory Instances (for high throughput applications, including database apps)

- High-Memory Double Extra Large Instance: 34.2 GB of memory, 13 EC2 Compute Units (4 virtual cores with 3.25 EC2 Compute Units each), 850 GB of instance storage, 64-bit platform
- High-Memory Quadruple Extra Large Instance: 68.4 GB of memory, 26 EC2 Compute Units (8 virtual cores with 3.25 EC2 Compute Units each), 1690 GB of instance storage, 64-bit platform

High-CPU Instances (for compute-intensive applications)

- High-CPU Medium Instance: 1.7 GB of memory, 5 EC2 Compute Units (2 virtual cores with 2.5 EC2 Compute Units each), 350 GB of instance storage, 32-bit platform
- High-CPU Extra Large Instance: 7 GB of memory, 20 EC2 Compute Units (8 virtual cores with 2.5 EC2 Compute Units each), 1690 GB of instance storage, 64-bit platform

What It All Costs: European prices, Oct 2012

Standard On-Demand Instances	Linux/UNIX Usage	Windows Usage
Small (Default)	$0.11 per hour *($0.095 on Nov 1)*	$0.135 per hour *($0.13 on Nov 1)*
Large	$0.44 per hour *($0.38 on Nov 1)*	$0.54 per hour *($0.52 on Nov 1)*
Extra Large	$0.88 per hour *($0.76 on Nov 1)*	$1.08 per hour *($1.04 on Nov 1)*
High-Memory On-Demand Instances	**Linux/UNIX Usage**	**Windows Usage**
Double Extra Large	$1.34 per hour	$1.58 per hour
Quadruple Extra Large	$2.68 per hour	$3.16 per hour
High-CPU On-Demand Instances	**Linux/UNIX Usage**	**Windows Usage**
Medium	$0.22 per hour *($0.19 on Nov 1)*	$0.32 per hour *($0.31 on Nov 1)*
Extra Large	$0.88 per hour *($0.76 on Nov 1)*	$1.28 per hour *($1.24 on Nov 1)*

What It All Costs: Oct 2013 (average -33% YoY)

Standard On-Demand Instances	Linux/UNIX Usage	Windows Usage
Small (Default)	$0.065 per hour *(-40%)*	$0.091 per hour *(-33%)*
Large	$0.26 per hour *(-40%)*	$0.364 per hour *(-33%)*
Extra Large	$0.52 per hour *(-40%)*	$0.728 per hour *(-33%)*
High-Memory On-Demand Instances	**Linux/UNIX Usage**	**Windows Usage**
Double Extra Large	$0.92 per hour (-31%)	$1.02 per hour (-35%)
Quadruple Extra Large	$1.84 per hour (-31%)	$2.04 per hour (-35%)
High-CPU On-Demand Instances	**Linux/UNIX Usage**	**Windows Usage**
Medium	$0.165 per hour *(-25%)*	$0.225 per hour *(-30%)*
Extra Large	$0.66 per hour *(-25%)*	$0.90 per hour *(-30%)*

What It All Costs: Sep 2014 (average -44% YoY)

Standard On-Demand Instances	Linux/UNIX Usage	Windows Usage
Small (Default)	$0.028 p.h. *(t2.small -57%)*	$0.038 p.h. *(-58%)*
Large	$0.154 p.h. *(m3.large -41%)*	$0.266 p.h. *(-27%)*
Extra Large	$0.308 p.h. *(m3. xlarge -41%)*	$0.532 p.h. *(-27%)*
High-Memory On-Demand Instances	**Linux/UNIX Usage**	**Windows Usage**
Double Extra Large	$0.390 p.h. *(r3.xlarge -58%)*	$0.600 p.h. (-41%)
Quadruple Extra Large	$0.780 p.h. *(r3.2xlarge -58%)*	$1.080 p.h. (-47%)
High-CPU On-Demand Instances	**Linux/UNIX Usage**	**Windows Usage**
Medium	$0.120 p.h. *(c3.large -27%)*	$0.188 p.h. *(-16%)*
Extra Large	$0.239 p.h. *(c3.xlarge -64%)*	$0.376 p.h. *(-58%)*

Price drop Oct 2012 – Sep 2014 (2yr average: -63%)

Standard On-Demand Instances	Linux/UNIX Usage	Windows Usage
Small (Default)	$0.028 p.h. *(t2.small -75%)*	$0.038 p.h. *(-72%)*
Large	$0.154 p.h. *(m3.large -65%)*	$0.266 p.h. *(-51%)*
Extra Large	$0.308 p.h. *(m3. xlarge -65%)*	$0.532 p.h. *(-51%)*
High-Memory On-Demand Instances	**Linux/UNIX Usage**	**Windows Usage**
Double Extra Large	$0.390 p.h. *(r3.xlarge -71%)*	$0.600 p.h. (-62%)
Quadruple Extra Large	$0.780 p.h. *(r3.2xlarge -71%)*	$1.080 p.h. (-66%)
High-CPU On-Demand Instances	**Linux/UNIX Usage**	**Windows Usage**
Medium	$0.120 p.h. *(c3.large -45%)*	$0.188 p.h. *(-41%)*
Extra Large	$0.239 p.h. *(c3.xlarge -73%)*	$0.376 p.h. *(-71%)*

What It All Costs: Oct 2015 (average -6% YoY)

Standard On-Demand Instances	Linux/UNIX Usage	Windows Usage
Small (Default)	$0.026 p.h. *(t2.small -7%)*	$0.036 p.h. *(-5%)*
Large	$0.133 p.h. *(m3.large -14%)*	$0.259 p.h. *(-3%)*
Extra Large	$0.266 p.h. *(m3. xlarge -14%)*	$0.518 p.h. *(-3%)*
High-Memory On-Demand Instances	**Linux/UNIX Usage**	**Windows Usage**
Double Extra Large	$0.350 p.h. *(r3.xlarge -10%)*	$0.600 p.h. (0%)
Quadruple Extra Large	$0.700 p.h. *(r3.2xlarge -10%)*	$1.080 p.h. (0%)
High-CPU On-Demand Instances	**Linux/UNIX Usage**	**Windows Usage**
Medium	$0.105 p.h. *(c3.large -13%)*	$0.188 p.h. *(0%)*
Extra Large	$0.210 p.h. *(c3.xlarge -12%)*	$0.376 p.h. *(0%)*

What It All Costs: Oct 2016 (average +5% YoY)

Standard On-Demand Instances	Linux/UNIX Usage	Windows Usage
Small (Default)	$0.028 p.h. *(t2.small +8%)*	$0.038 p.h. *(+6%)*
Large	$0.146 p.h. *(m3.large +10%)*	$0.258 p.h. *(0%)*
Extra Large	$0.293 p.h. *(m3. xlarge +10%)*	$0.517 p.h. *(0%)*
High-Memory On-Demand Instances	**Linux/UNIX Usage**	**Windows Usage**
Double Extra Large	$0.371 p.h. *(r3.xlarge +6%)*	$0.581 p.h. (-3%)
Quadruple Extra Large	$0.741 p.h. *(r3.2xlarge +6%)*	$1.080 p.h. (0%)
High-CPU On-Demand Instances	**Linux/UNIX Usage**	**Windows Usage**
Medium	$0.12 p.h. *(c3.large +14%)*	$0.188 p.h. *(0%)*
Extra Large	$0.239 p.h. *(c3.xlarge +142%)*	$0.376 p.h. *(0%)*

What It All Costs: Oct 2018 (average -23% wrt 2016)

Standard On-Demand Instances	**Linux/UNIX Usage**	**Windows Usage**
Small (Default)	$0.023 p.h. *(t2.small -18%)*	$0.032 p.h. *(-16%)*
Large	$0.096 p.h. *(m5.large +-4%)*	$0.188 p.h. *(-27%)*
Extra Large	$0.192 p.h. *(m5.xlarge -34%)*	$0.376 p.h. *(-27%)*
High-Memory On-Demand Instances	**Linux/UNIX Usage**	**Windows Usage**
Double Extra Large	$0.252 p.h. *(r5.xlarge -32%)*	$0.436 p.h. (-25%)
Quadruple Extra Large	$0.504 p.h. *(r5.2xlarge -32%)*	$0.872 p.h. (-19%)
High-CPU On-Demand Instances	**Linux/UNIX Usage**	**Windows Usage**
Medium	$0.085 p.h. *(c5.large -29%)*	$0.188 p.h. *(0%)*
Extra Large	$0.17 p.h. *(c5.xlarge -29%)*	$0.376 p.h. *(0%)*

The complete 2014 AMI product range

Source http://aws.amazon.com/ec2/pricing/

General Purpose: t2.micro; t2.small; t2.medium; m3.medium; m3.large; m3.xlarge; m3.2xlarge

Compute-Optimized: c3.large; c3.xlarge; c3.2xlarge; c3.4xlarge; c3.8xlarge

GPU: g2.2xlarge

Memory-Optimized: r3.large; r3.xlarge; r3.2xlarge; r3.4xlarge; r3.8xlarge

Storage-Optimized: i2.xlarge; i2.2xlarge; i2.4xlarge; i2.8xlarge; hs1.8xlarge

NB all except t2 range now use SSD storage rather than EBS.

Range of OS's now: Linux, RedHat Enterprise Linux; Suse Linux Enterprise Server; Windows; Windows with SQL Standard; Windows with SQL Web

So, 6 OS options and 23 AMI types: 138 different combinations, different prices.

The complete 2015 AMI product range

Source http://aws.amazon.com/ec2/pricing/

General Purpose: t2.micro; t2.small; t2.medium; t2.large; m3.medium; m3.large; m3.xlarge; m3.2xlarge; m4.large; m4.xlarge; m4.2xlarge; m4.4xlarge; m4.10xlarge.

Compute-Optimized: c3.large; c3.xlarge; c3.2xlarge; c3.4xlarge; c3.8xlarge; c4.large; c4.2xlarge; c4.4xlarge; c4.8xlarge.

GPU: g2.2xlarge; g2.8xlarge.

Memory-Optimized: r3.large; r3.xlarge; r3.2xlarge; r3.4xlarge; r3.8xlarge

Storage-Optimized: i2.xlarge; i2.2xlarge; i2.4xlarge; i2.8xlarge; d2.xlarge; d2.2xlarge; d2.4xlarge; d2.8xlarge

Range of OS's now: Linux, RedHat Enterprise Linux; Suse Linux Enterprise Server; Windows; Windows with SQL Standard; Windows with SQL Web; Windows with SQL Enterprise

So, 7 OS options and 37 AMI types: 259 different combinations, different prices.

The complete 2016 AMI product range

Source http://aws.amazon.com/ec2/pricing/

General Purpose: t2.nano; t2.micro; t2.small; t2.medium; t2.large; m3.medium; m3.large; m3.xlarge; m3.2xlarge; m4.large; m4.xlarge; m4.2xlarge; m4.4xlarge; m4.10xlarge; m4.16xlarge.

Compute-Optimized: c3.large; c3.xlarge; c3.2xlarge; c3.4xlarge; c3.8xlarge; c4.large; c4.2xlarge; c4.4xlarge; c4.8xlarge.

GPU: g2.2xlarge; g2.8xlarge.

Memory-Optimized: r3.large; r3.xlarge; r3.2xlarge; r3.4xlarge; r3.8xlarge; x1.32xlarge

Storage-Optimized: i2.xlarge; i2.2xlarge; i2.4xlarge; i2.8xlarge; d2.xlarge; d2.2xlarge; d2.4xlarge; d2.8xlarge

Same set of 7 OS options.

So, 7 OS options and 40 AMI types: 280 different combinations, different prices.

The complete 2018 AMI product range

Source http://aws.amazon.com/ec2/pricing/

General Purpose: t2.nano; t2.micro; t2.small; t2.medium; t2.large; t2.xlarge; t2.2xlarge t3.nano; t3.micro; t3.small; t3.medium; t3.large; t3.xlarge; t3.2xlarge; m4.large; m4.xlarge; m4.2xlarge; m4.4xlarge; m4.10xlarge; m4.16m5.large; m5.xlarge; m5.2xlarge; m5.4xlarge; m5.12xlarge; m5.24xlarge; m5d.large; m5d,xlarge; m5d.2xlarge; m5d.12xlarge; m5d.24xlarge.

Compute-Optimized: c4.large; c4.2xlarge; c4.4xlarge; c4.8xlarge; c5.large; c5.xlarge; c5.2xlarge; c5.4xlarge; c5.9xlarge; c5.19xlarge; c5d.large; c5d.xlarge; c5d.2xlarge; c5d.4xlarge; c5d.9xlarge; c5d.18xlarge.

GPU: p3.2xlarge; p3.8xlarge; p3.16xlarge; p2.xlarge; p2.8xlarge; p2.16xlarge; g3.4xlarge; g3.8xlarge; g3.16xlarge.

Memory-Optimized: x1.16xlarge; x1.32xlarge; r5.large; r5.2xlarge; r5.4xlarge; r5.12xlarge; r5.24xlarge; r5dlarge; r5d.xlarge; r5d.2xlarge; rd5.4xlarge; r5d.12xlarge; r5d.24xlarge; r4.large; r4.xlarge; r4.2xlarge; r4.4xlarge; r4.8xlarge; r4.16xlarge.

Storage-Optimized: i3.large; i3.xlarge; i3.sxlarge; i3.4xlarge; i3.8xlarge; i3.16xlarge; i3.metal; h1.2xlarge; h1.4xlarge; h1.8xlarge; h1.16xlarge; d2.xlarge; d2.2xlarge; d2.4xlarge; d2.8xlarge.

Now 10 OS options.

So, 10 OS options and 89 AMI types: 890 different combinations, different prices.

Financial Times, 4th Nov 2015: Andy Jassy, AWS CEO

FINANCIAL TIMES

You are signed in | Search for...

ft.com/work&careers

Home | UK | World | Companies | Markets | Global Economy | Lex | Comment | Work & Careers | Personal Finance | Life & Arts

Business Education | Entrepreneurship | Business Books | Recruitment | The Connected Business | Lucy Kellaway | Tools

November 4, 2015 12:57 pm

The Amazon boss who conquered the cloud

Leslie Hook

Share | Author alerts | Print | Clip | Gift Article | Comments

Rivals strive to catch up with Andy Jassy's web services unit

Tighter and tighter: Andy Jassy went through 31 drafts of the six-page memo that led to Amazon Web Services

Selling porcelain figurines and starting a cloud computing company may seem like rather different skills. But Andy Jassy, who founded and now leads Amazon Web Services, says one experience deeply informed the other.

EDITOR'S CHOICE

DEAR LUCY

Next problem: Will Brexit hurt my legal career?

TIM HARFORD

Why everyone should give a TED talk and how to do it

COLUMNISTS

1. Lucy Kellaway on the Office
2. Andrew Hill on Management

Financial Times, 4th Nov 2015

Interviewees were told they would be working on expanding a programming interface for another Amazon business. The strategy worked — even better than Mr Jassy had expected — and it took Amazon's competitors years to cotton on to the potential of cloud computing services.

"In my wildest dreams I don't think anyone believed at the time we would have the seven-year headstart we had," Mr Jassy says.

Today the competitors have caught on in a big way. AWS has come into the limelight at a time when pretty much every large IT company — from IBM to Oracle to Google — would be happy to steal some of its business.

AWS has stayed ahead of the pack: Gartner, the research group, estimates that it has 10 times the computing capacity of its 14 nearest competitors combined. But Microsoft in particular has been catching up. Last month its share price rose to a 15-year high, helped by the news that revenues from Azure, its cloud computing platform, had doubled from the previous year.

"AWS has competition that it has to think about at

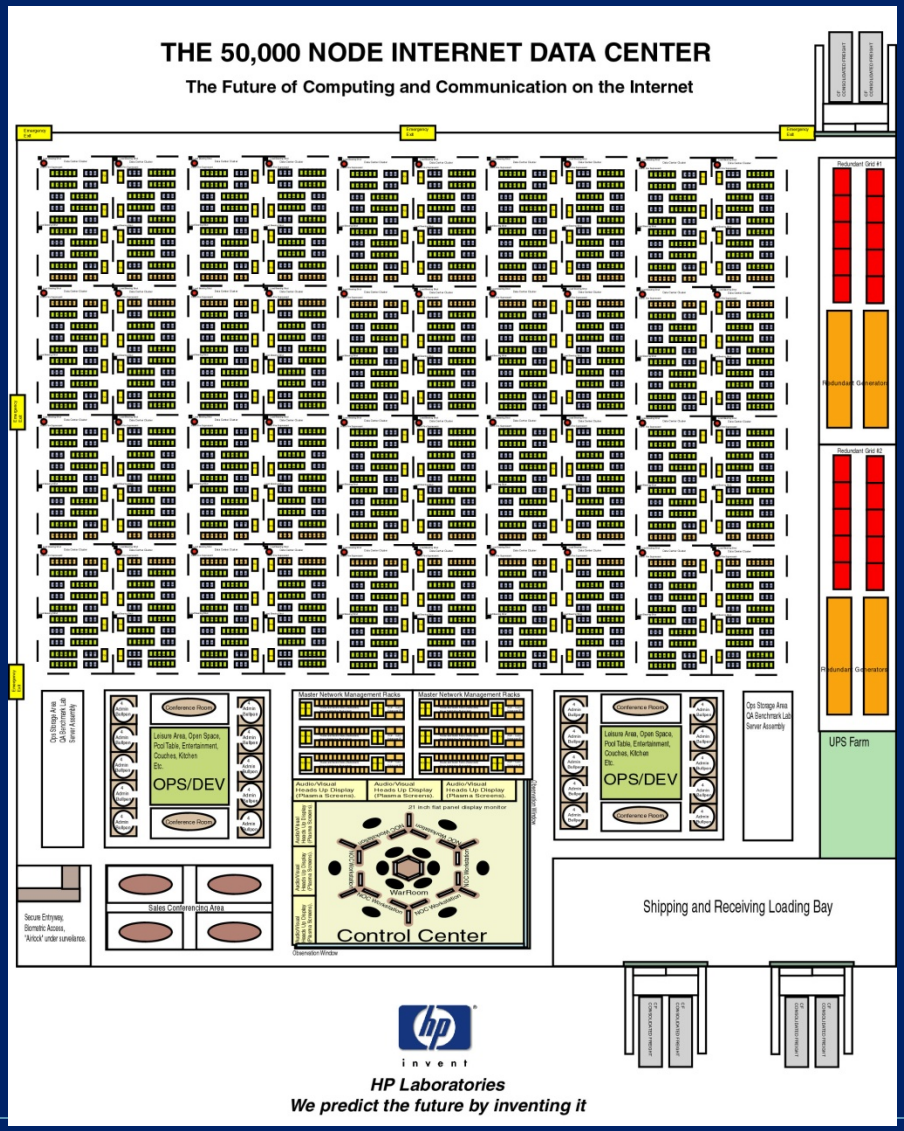
Frugality under scrutiny

Amazon employs more than 200,000 people, ranging from

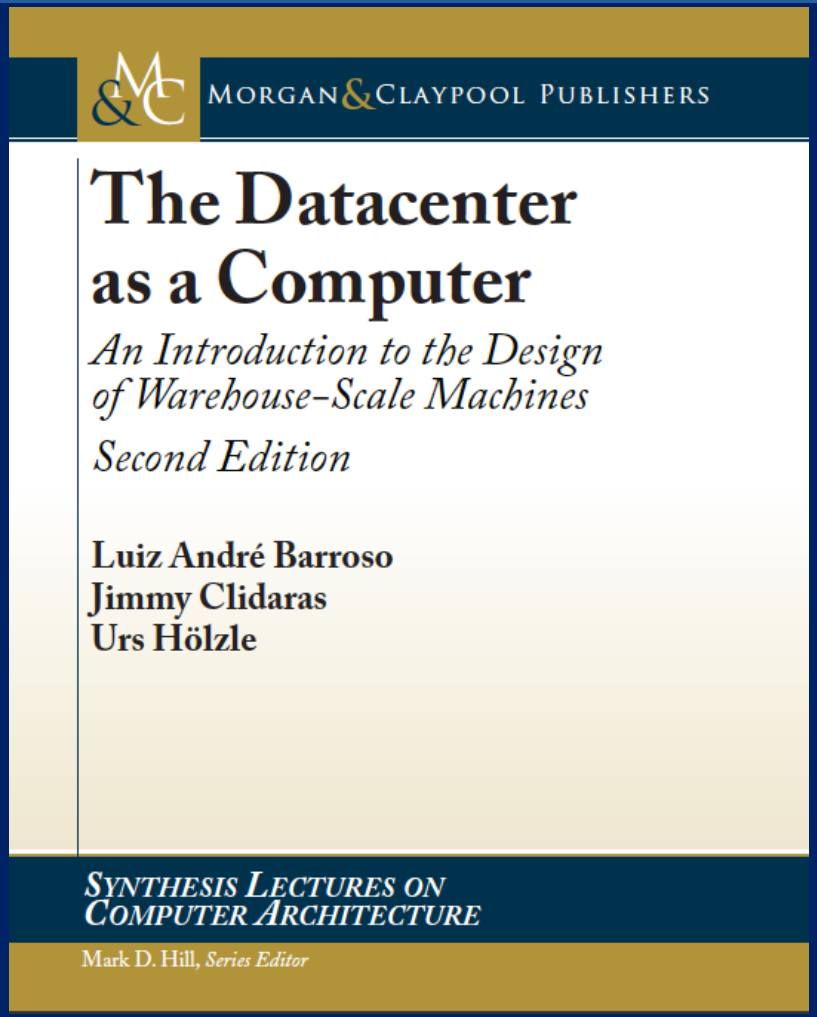
rather different skills. But Andy Jassy, who founded and now leads Amazon Web Services, says one experience deeply informed the other.

1. Lucy Kellaway on the Office
2. Andrew Hill on Management

From Lecture 1 (HP Datacenter Floorplan, c.1999)

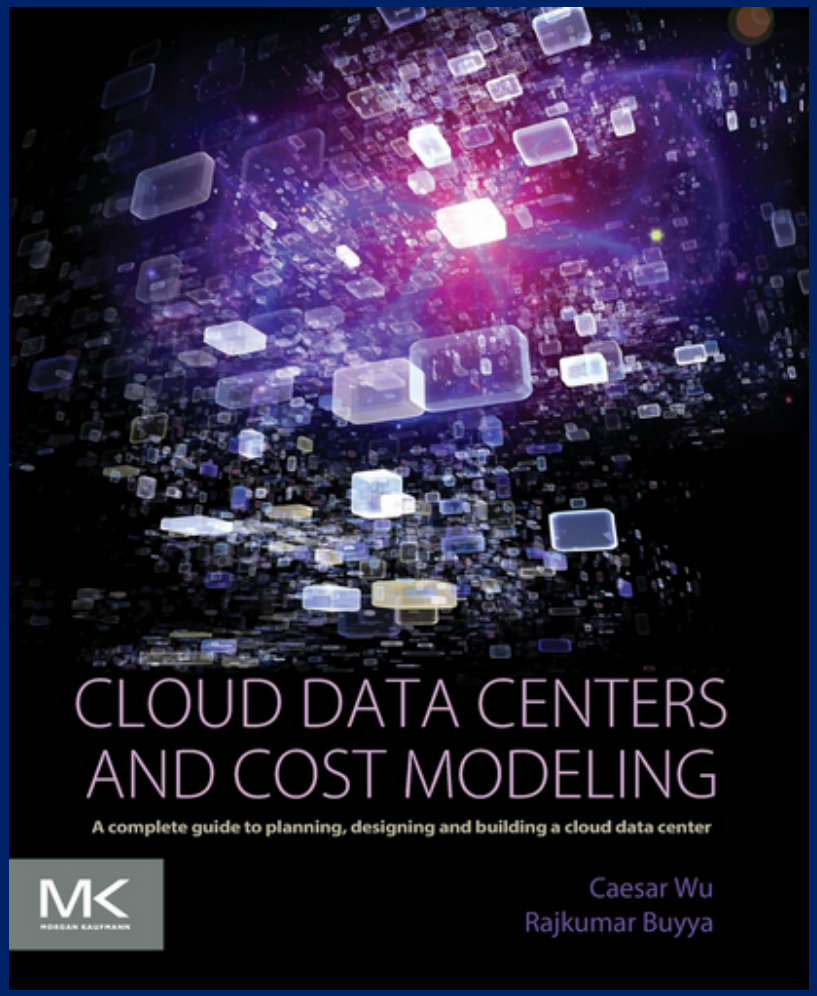


The Datacenter as a Computer ("the BCH book")

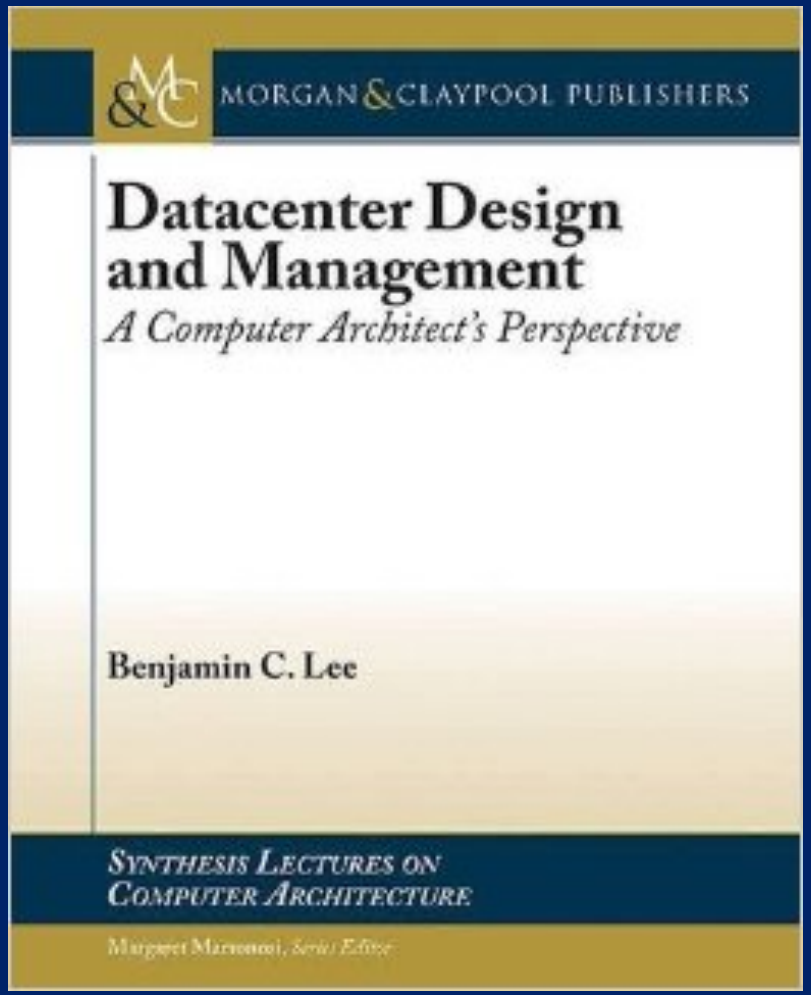


1/e 2009; 2/e 2013

Other books are available (but not used here)



2015

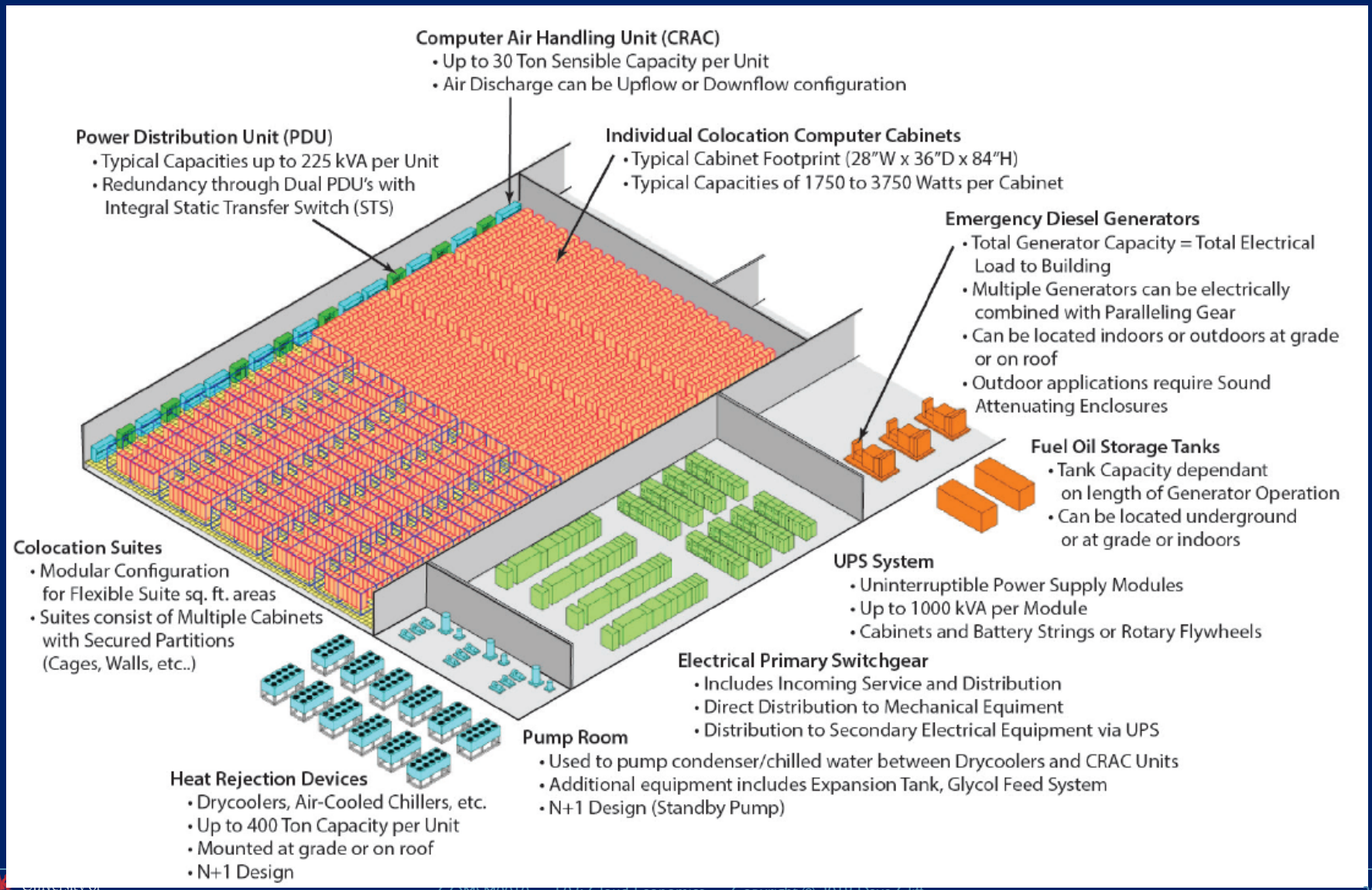


2016

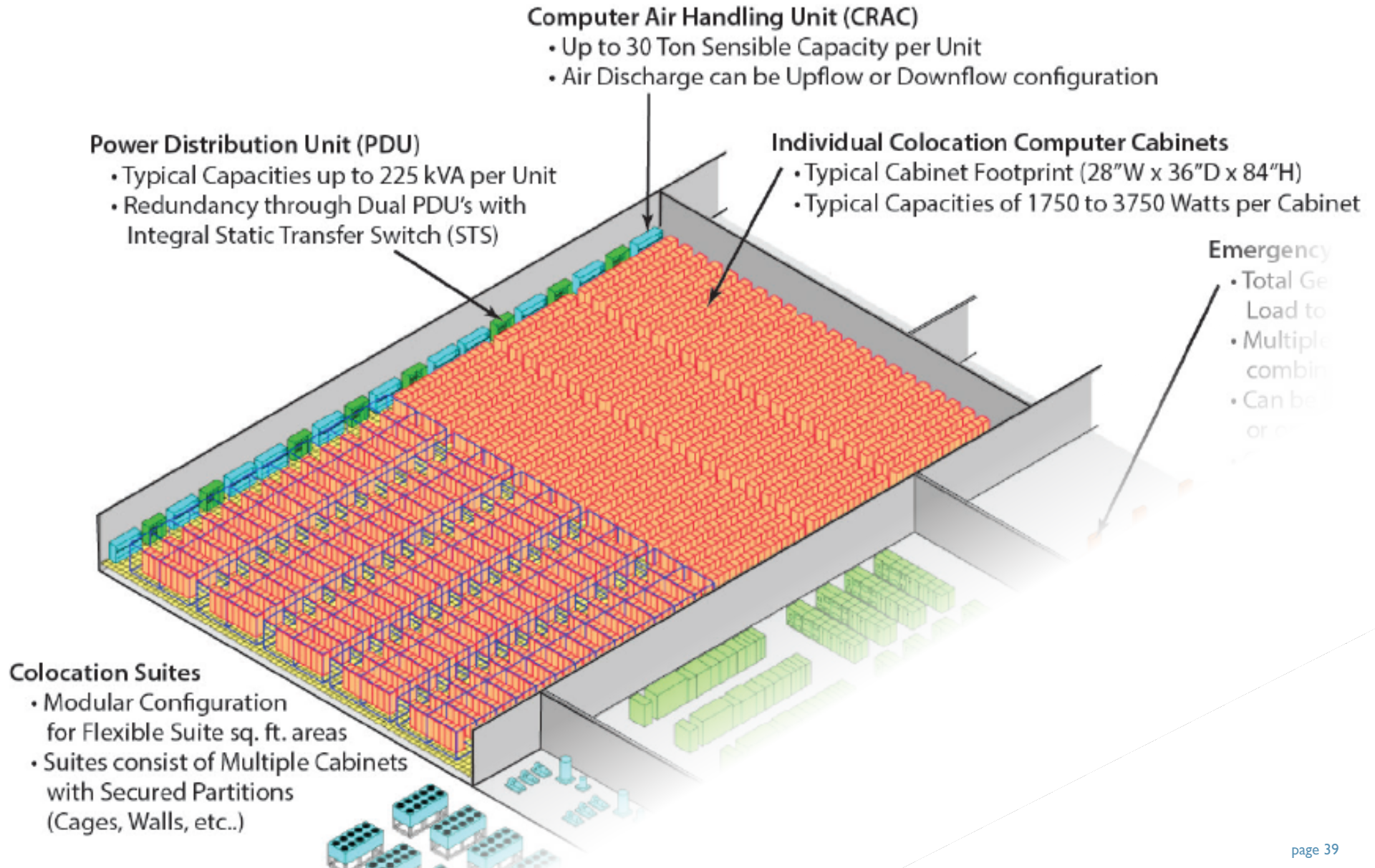
BCH gives details of the whole story, including power/cooling…

A typical Warehouse-Scale Computer (WSC)

Source: BCH (2013) Ch4.

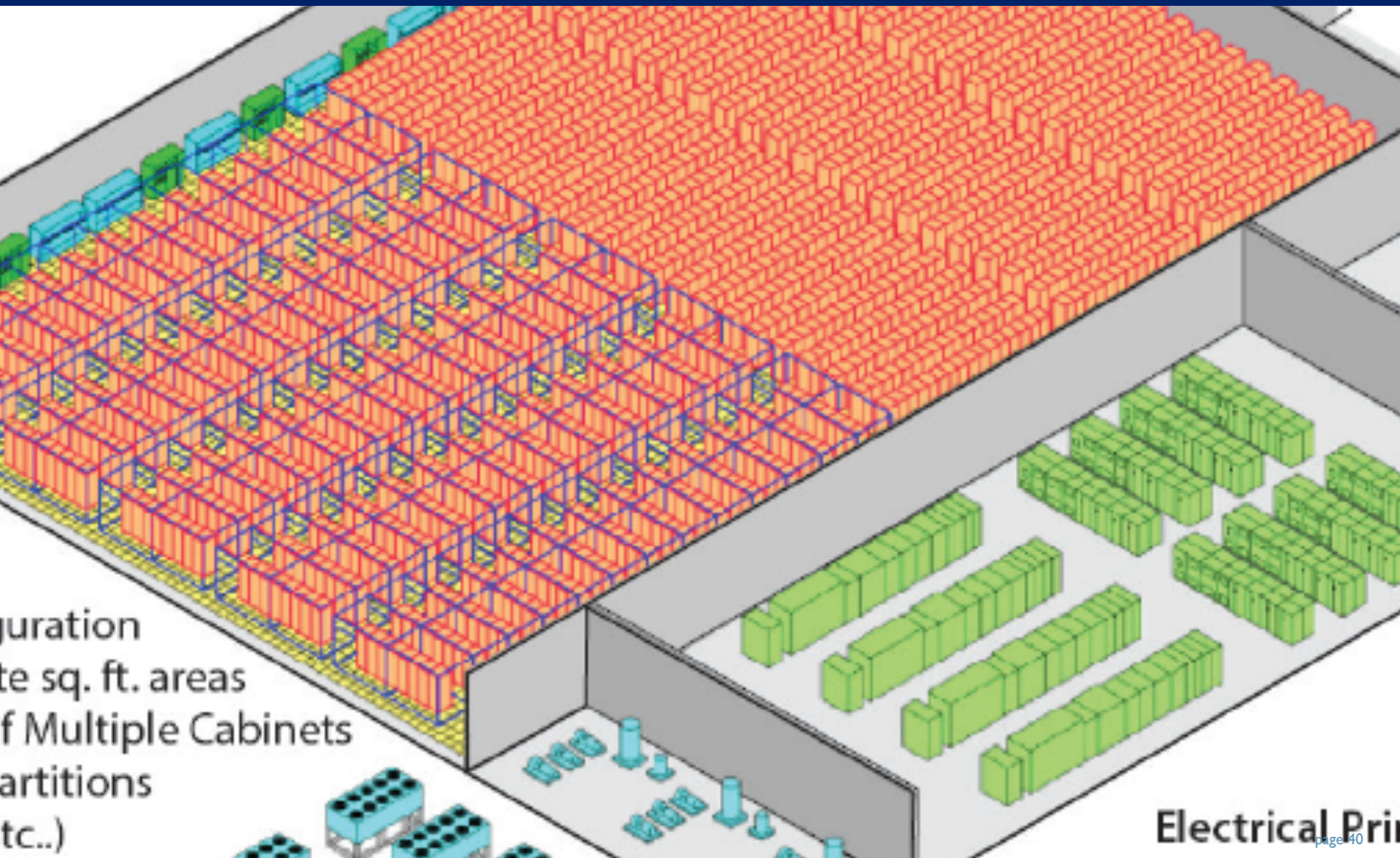


A typical Warehouse-Scale Computer (WSC)



A typical Warehouse-Scale Computer (WSC)

Source: BCH (2013) Ch4.



A typical Warehouse-Scale Computer (WSC)

Source: BCH (2013) Ch1.

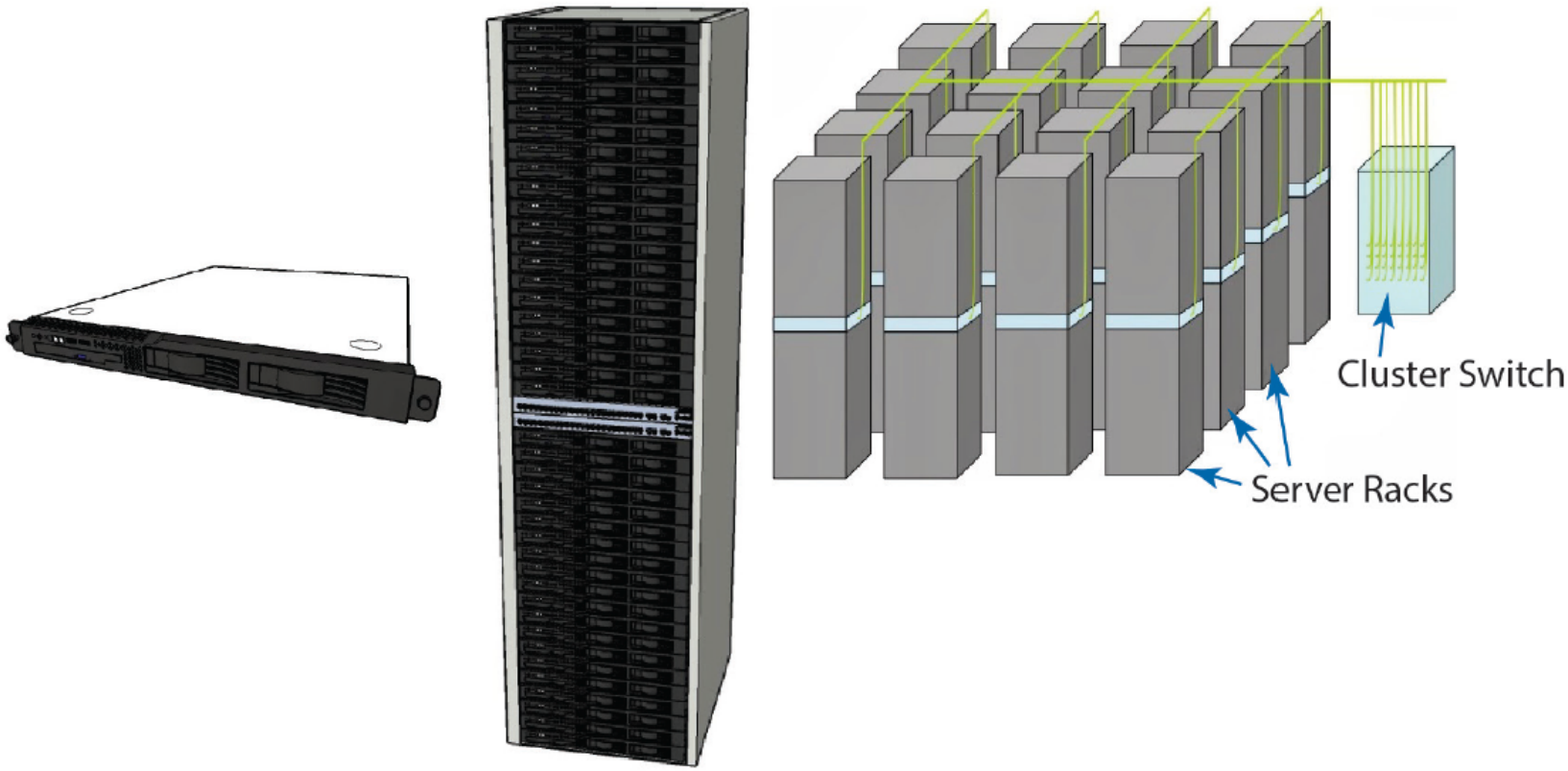


Figure 1.1: Sketch of the typical elements in warehouse-scale systems: 1U server (left), 7' rack with Ethernet switch (middle), and diagram of a small cluster with a cluster-level Ethernet switch/ router (right).

A typical Warehouse-Scale Computer (WSC)

Source: BCH (2013) Ch1.

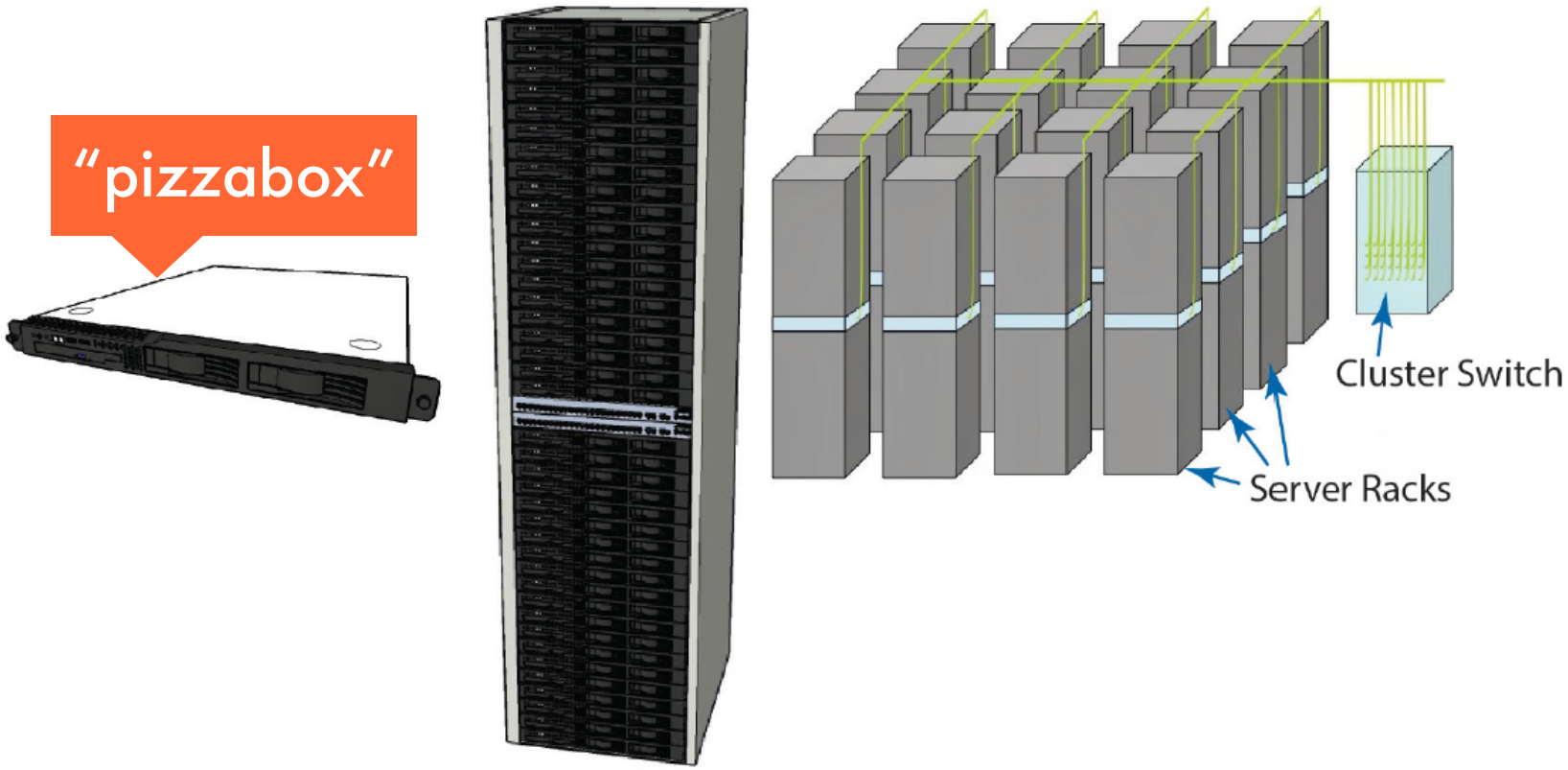


Figure 1.1: Sketch of the typical elements in warehouse-scale systems: 1U server (left), 7' rack with Ethernet switch (middle), and diagram of a small cluster with a cluster-level Ethernet switch/ router (right).

A typical Warehouse-Scale Computer (WSC)

Source: BCH (2013) Ch1.

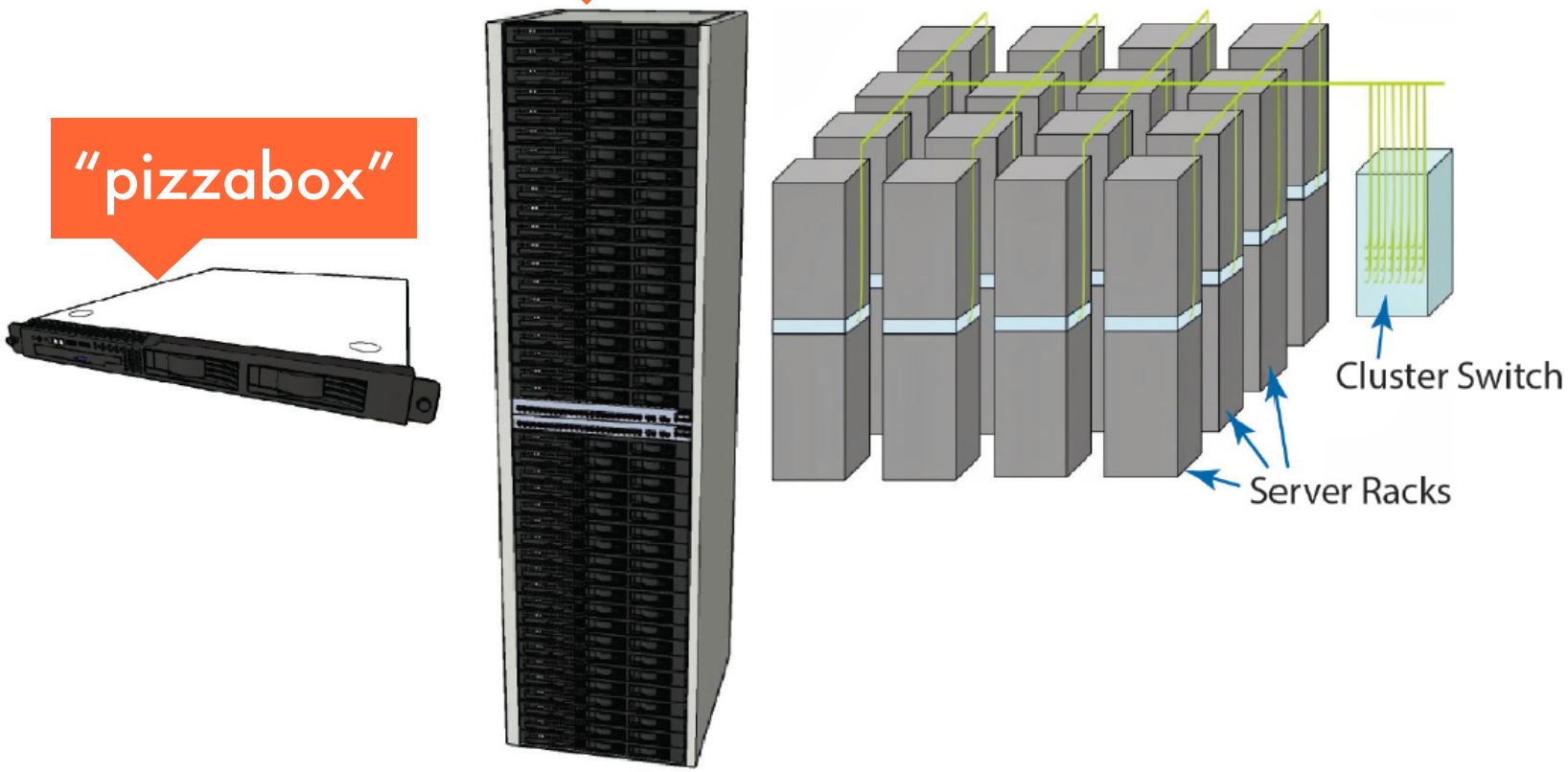


Figure 1.1: Sketch of the typical elements in warehouse-scale systems: 1U server (left), 7' rack with Ethernet switch (middle), and diagram of a small cluster with a cluster-level Ethernet switch/ router (right).

A typical Warehouse-Scale Computer (WSC)

Source: BCH (2013) Ch1.

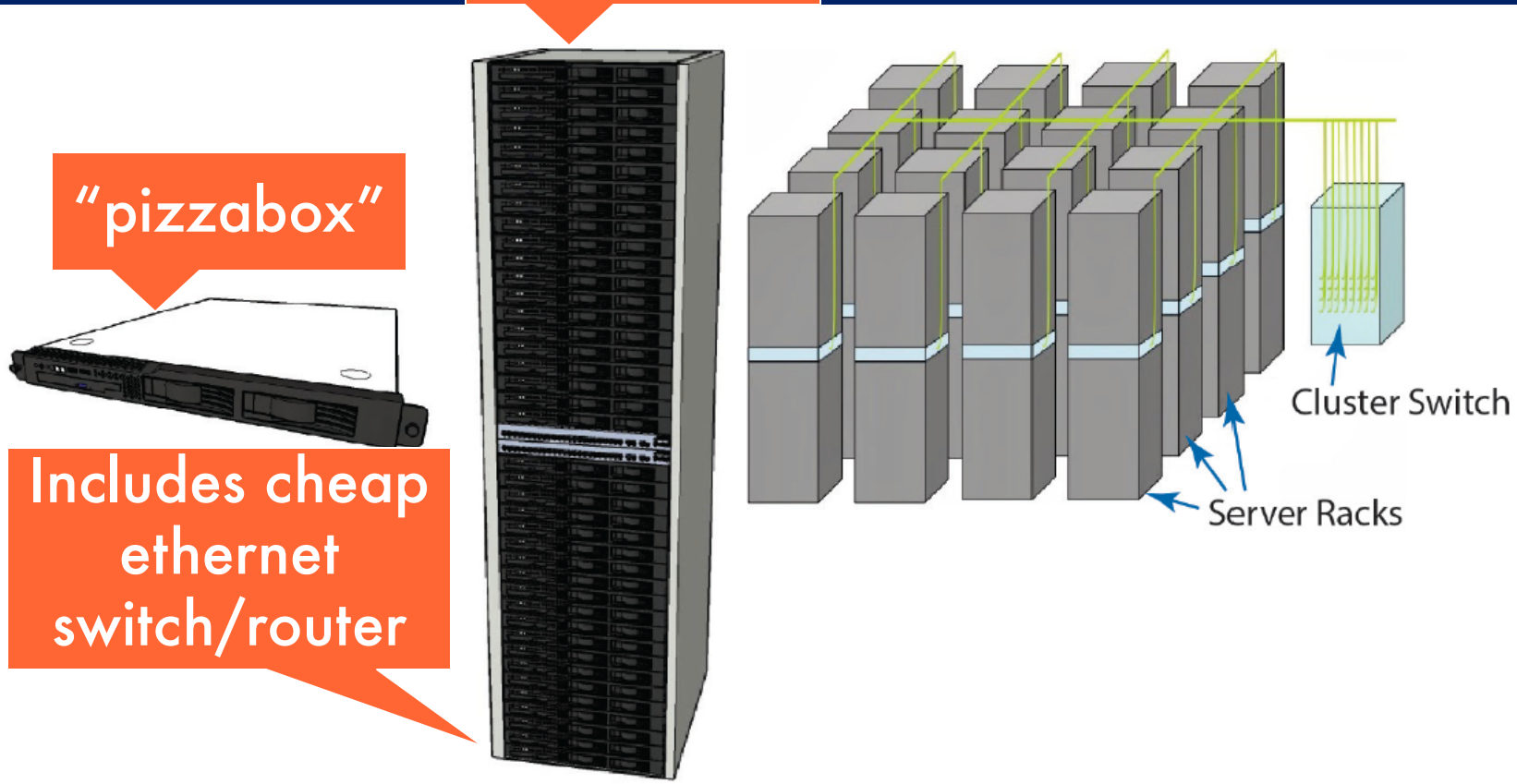


Figure 1.1: Sketch of the typical elements in warehouse-scale systems: 1U server (left), 7' rack with Ethernet switch (middle), and diagram of a small cluster with a cluster-level Ethernet switch/ router (right).

A typical Warehouse-Scale Computer (WSC)

Source: BCH (2013) Ch1.

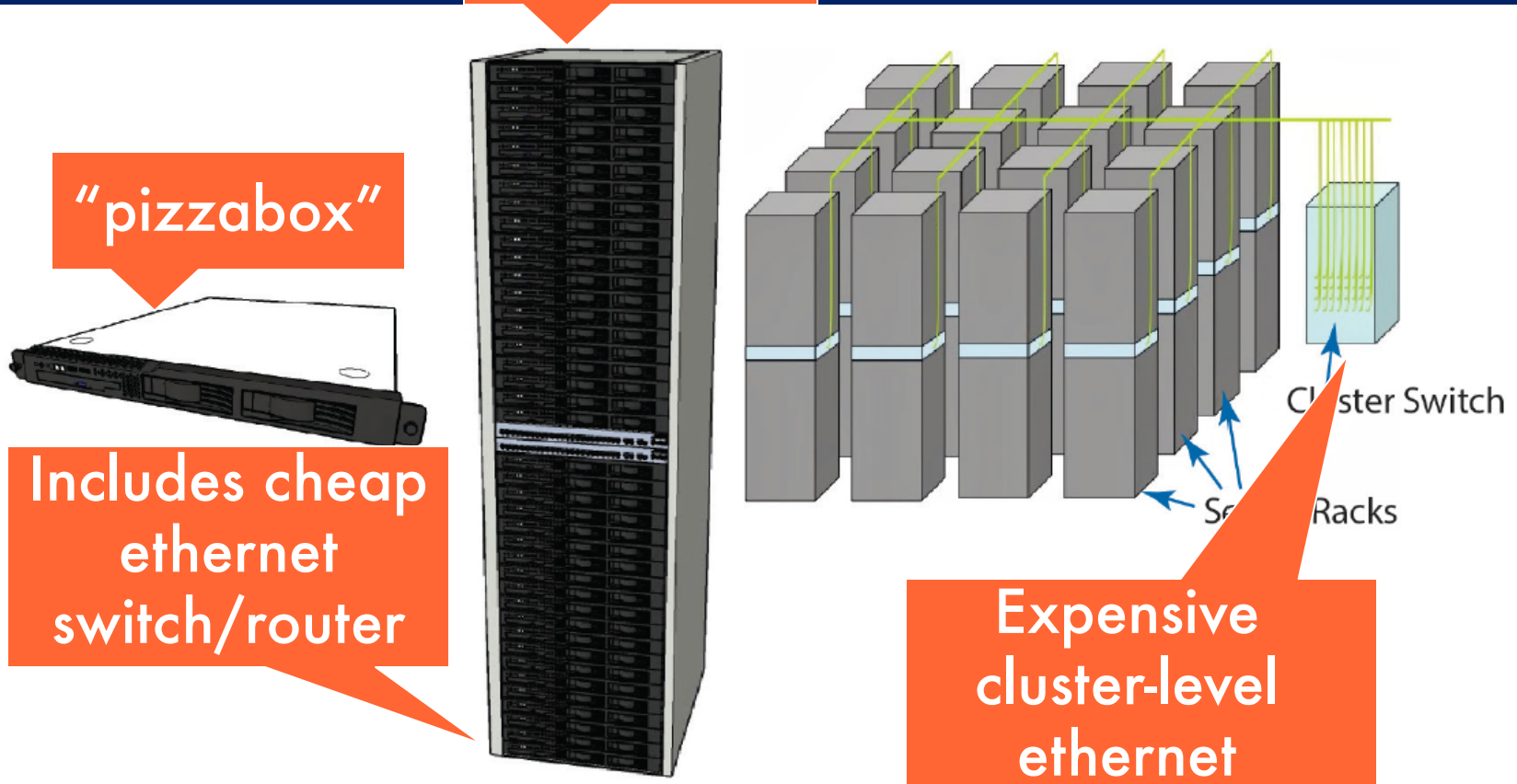


Figure 1.1: Sketch of the typical elements in warehouse-sca rack with Ethernet switch (middle), and diagram of a small cluster with a cluster-level Ethernet switch/ router (right).

A typical Warehouse-Scale Computer (WSC)

Source: BCH (2013) Ch1.

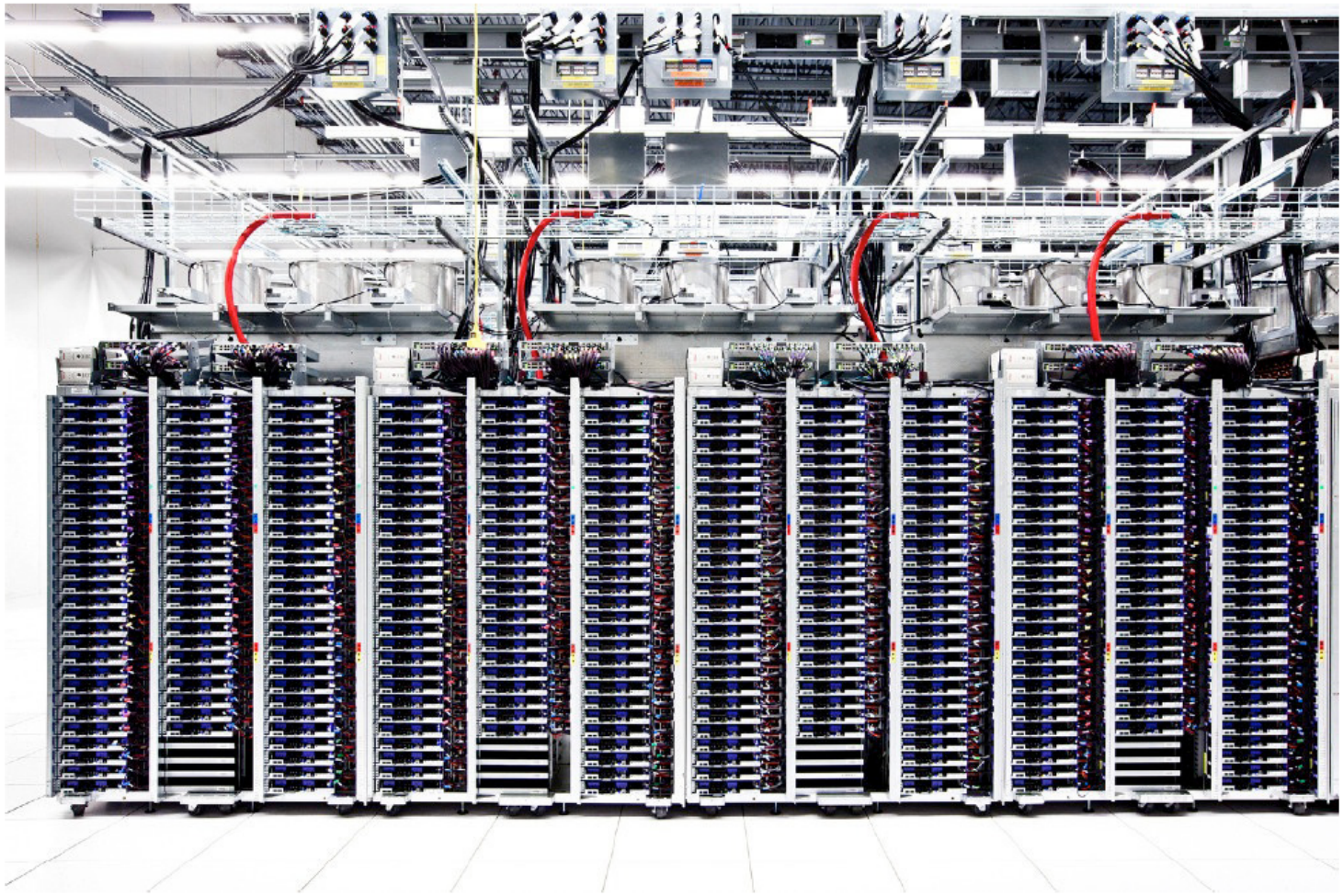
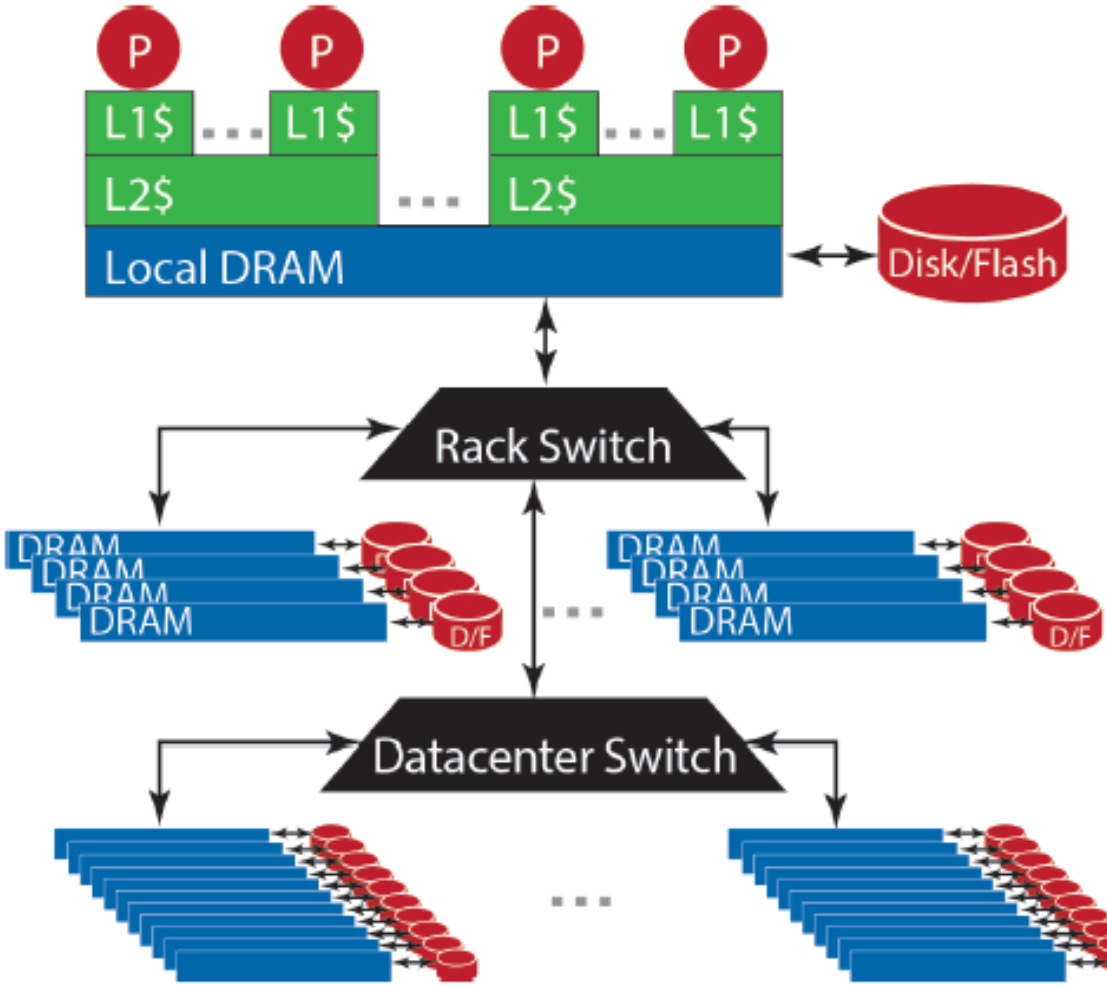


Figure 1.2: Picture of a row of servers in a Google WSC, 2012.

WSC Storage Hierarchy

Source: BCH (2013) Ch1.



One Server
DRAM: 16 GB, 100 ns, 20 GB/s
Disk: 2T B, 10 ms, 200 MB/s
Flash: 128 GB, 100 us, 1 GB/s

Local Rack (80 servers)
DRAM: 1 TB, 300 us, 100 MB/s
Disk: 160 TB, 11 ms, 100 MB/s
Flash: 20 TB, 400 us, 100 MB/s

Cluster (30 racks)
DRAM: 30 TB, 500 us, 10 MB/s
Disk: 4.80 PB, 12 ms, 10 MB/s
Flash: 600 TB, 600 us, 10 MB/s

Figure 1.3: Storage hierarchy of a WSC.

WSC Storage Hierarchy

Source: BCH (2013) Ch1.

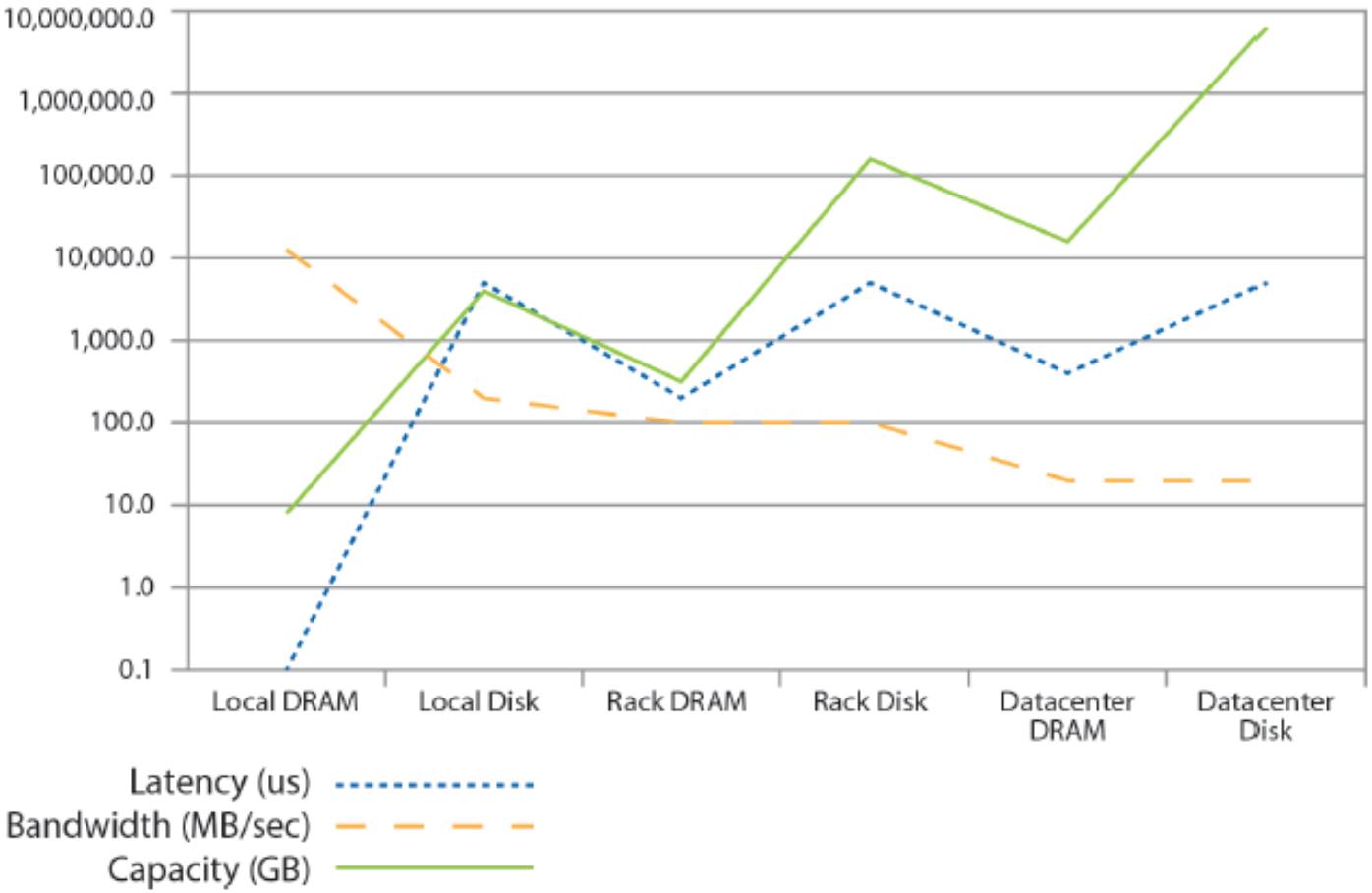


Figure 1.4: Latency, bandwidth, and capacity of a WSC.

WSC Storage Hierarchy

Source: BCH (2013) Ch1.

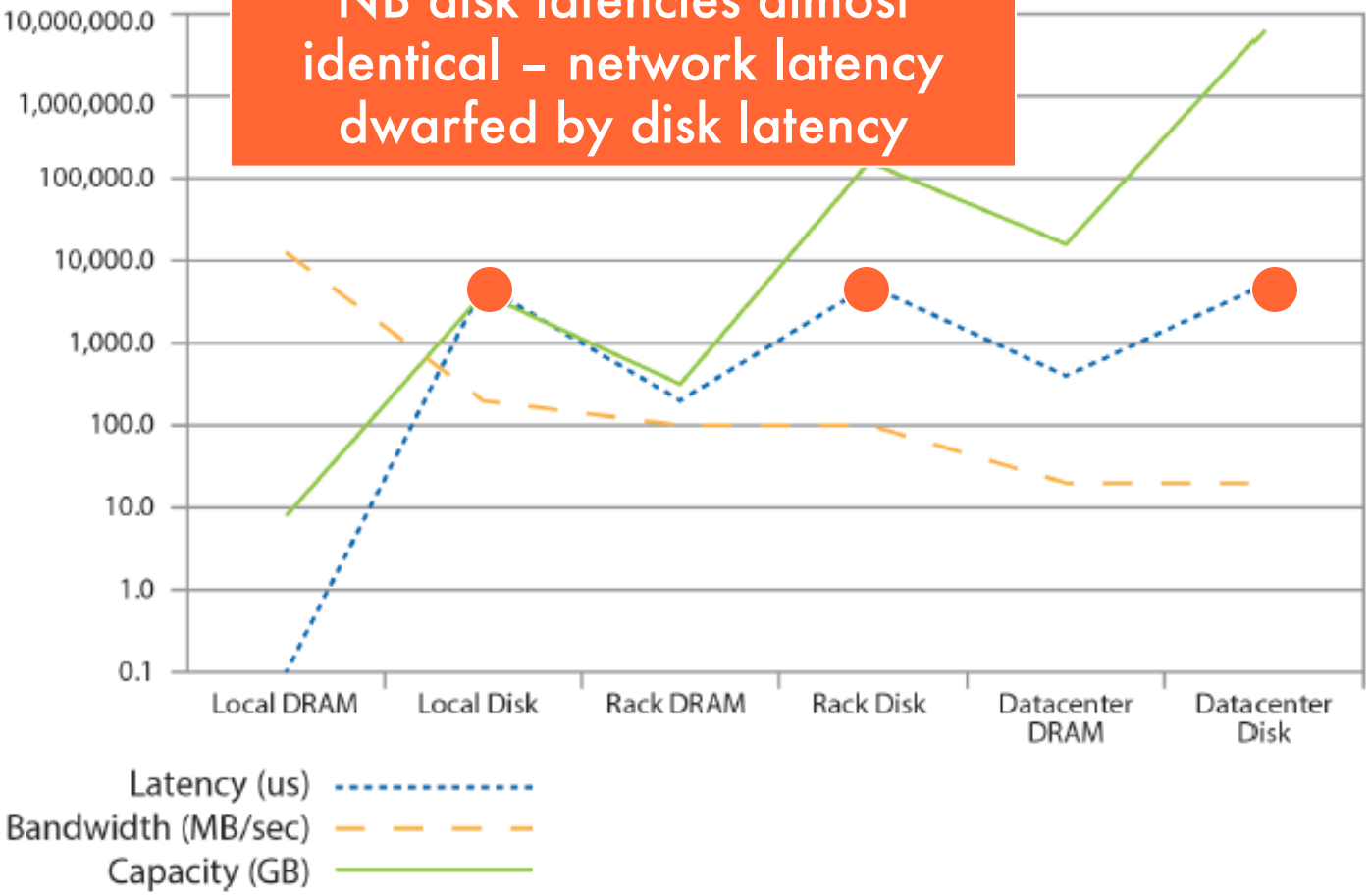


Figure 1.4: Latency, bandwidth, and capacity of a WSC.

WSC Storage Hierarchy

Source: BCH (2013) Ch1.

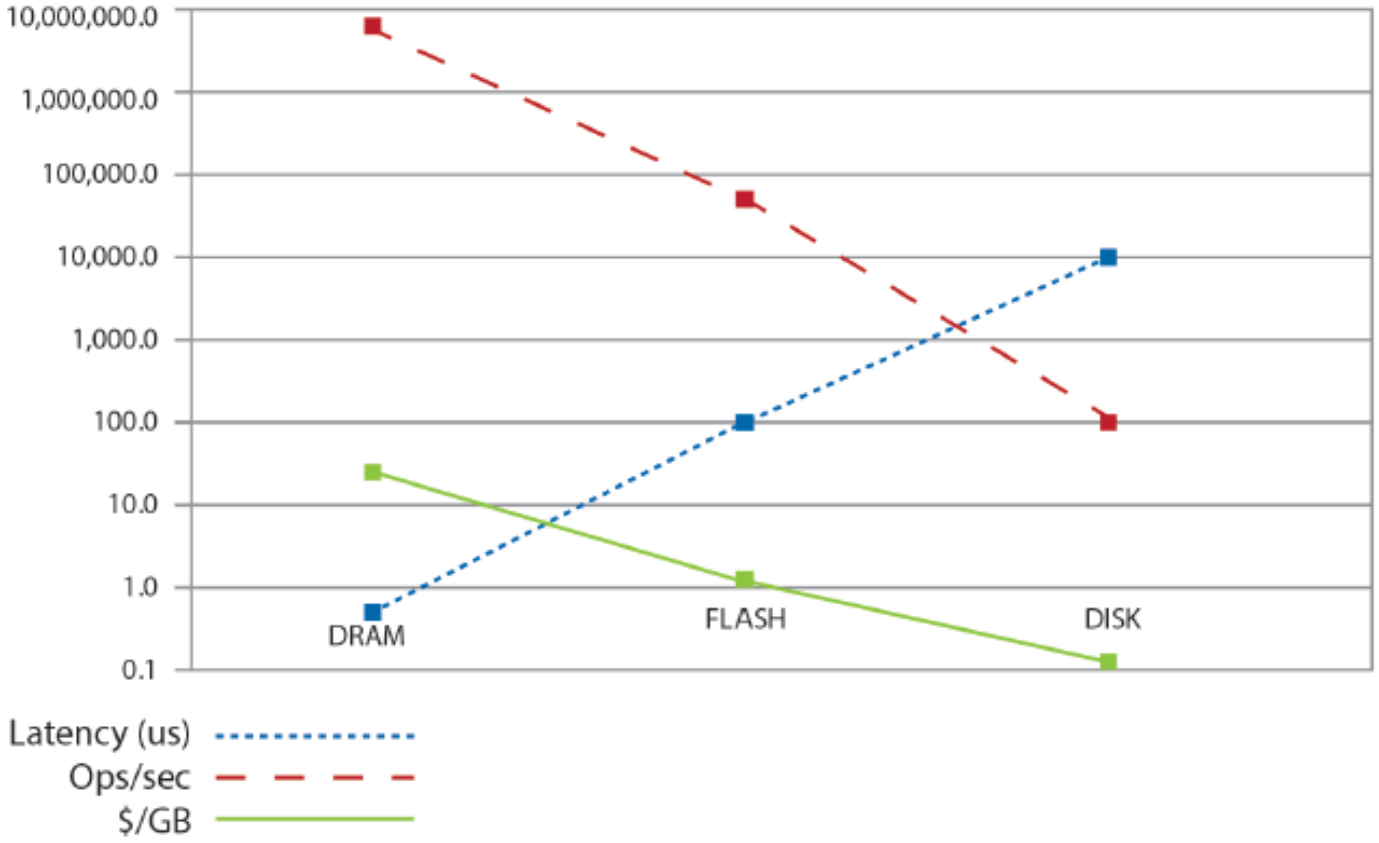
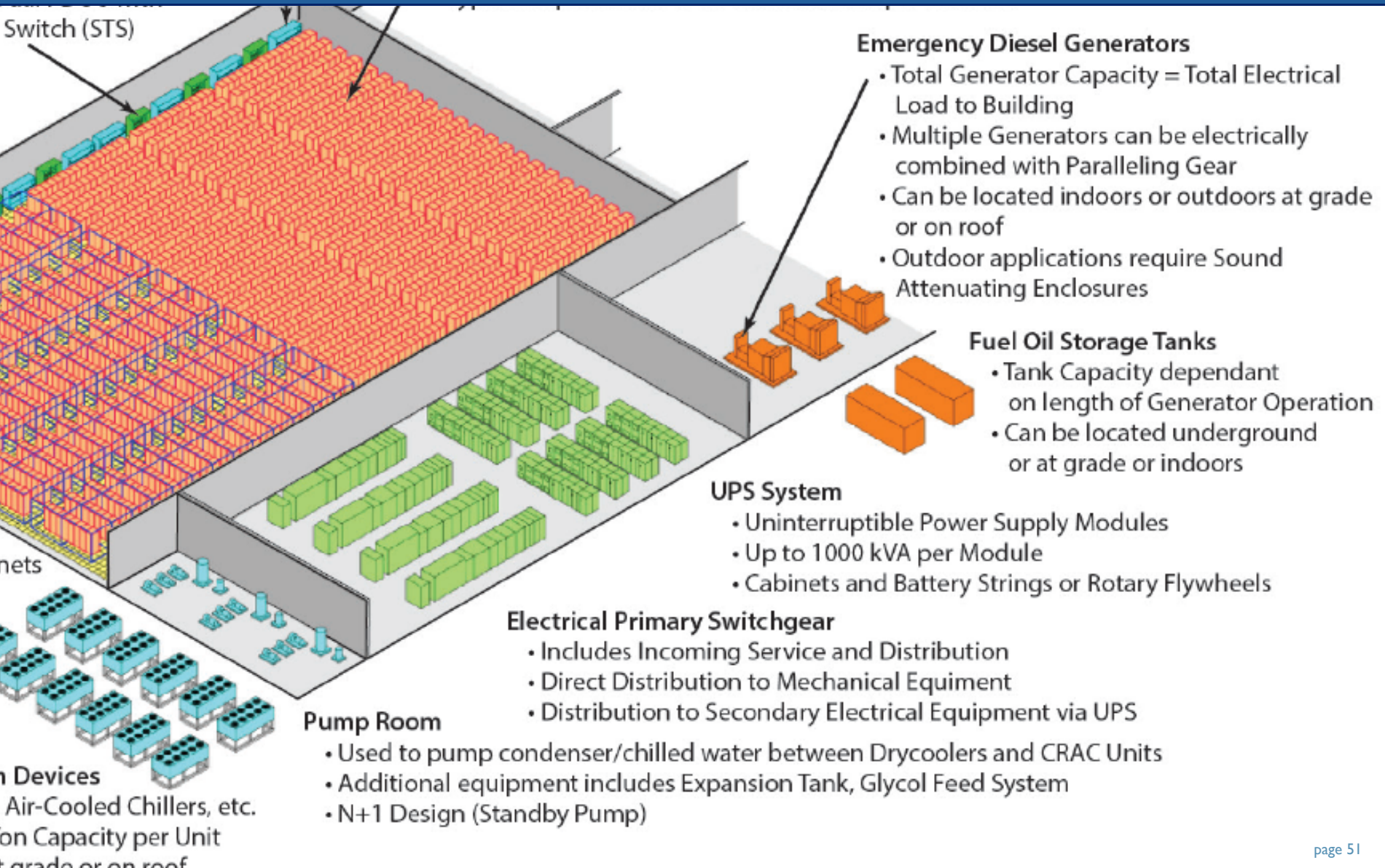


Figure 1.5: Performance and cost of NAND Flash with respect to DRAM and Disks (latency and Ops/sec assume a 4 KB data transfer).

A typical Warehouse-Scale Computer (WSC)

Source: BCH (2013) Ch4.



Keeping it cool…

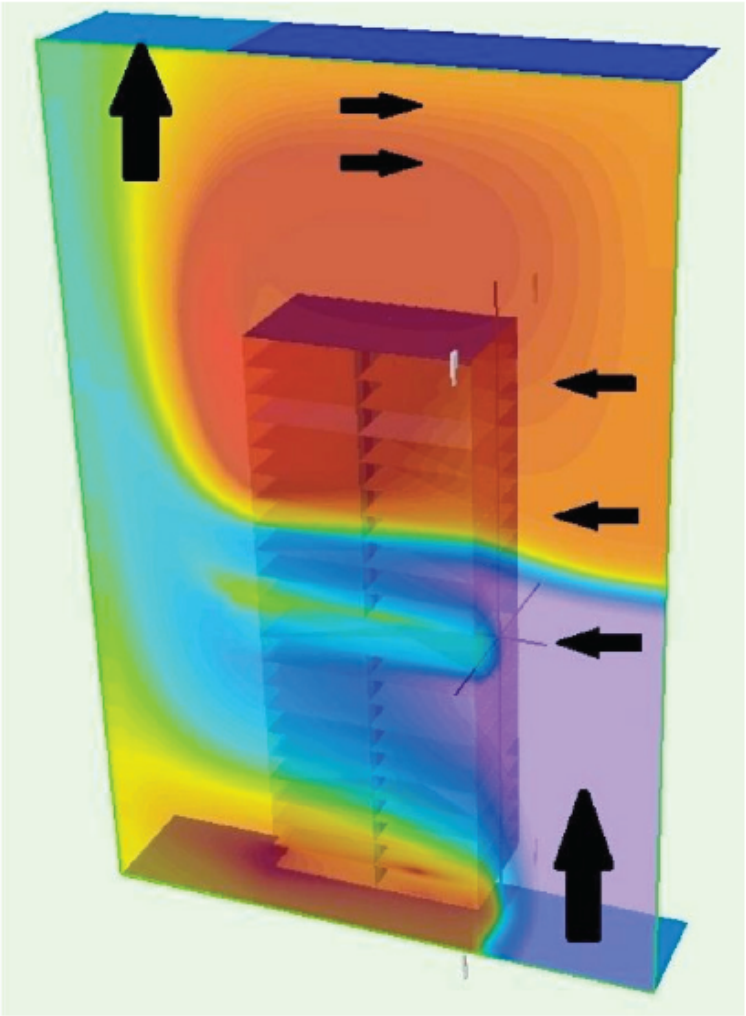


Figure 4.10: CFD model showing recirculation paths and temperature stratification for a rack with under-provisioned airflow.

Keeping it cool #1: "open the windows"

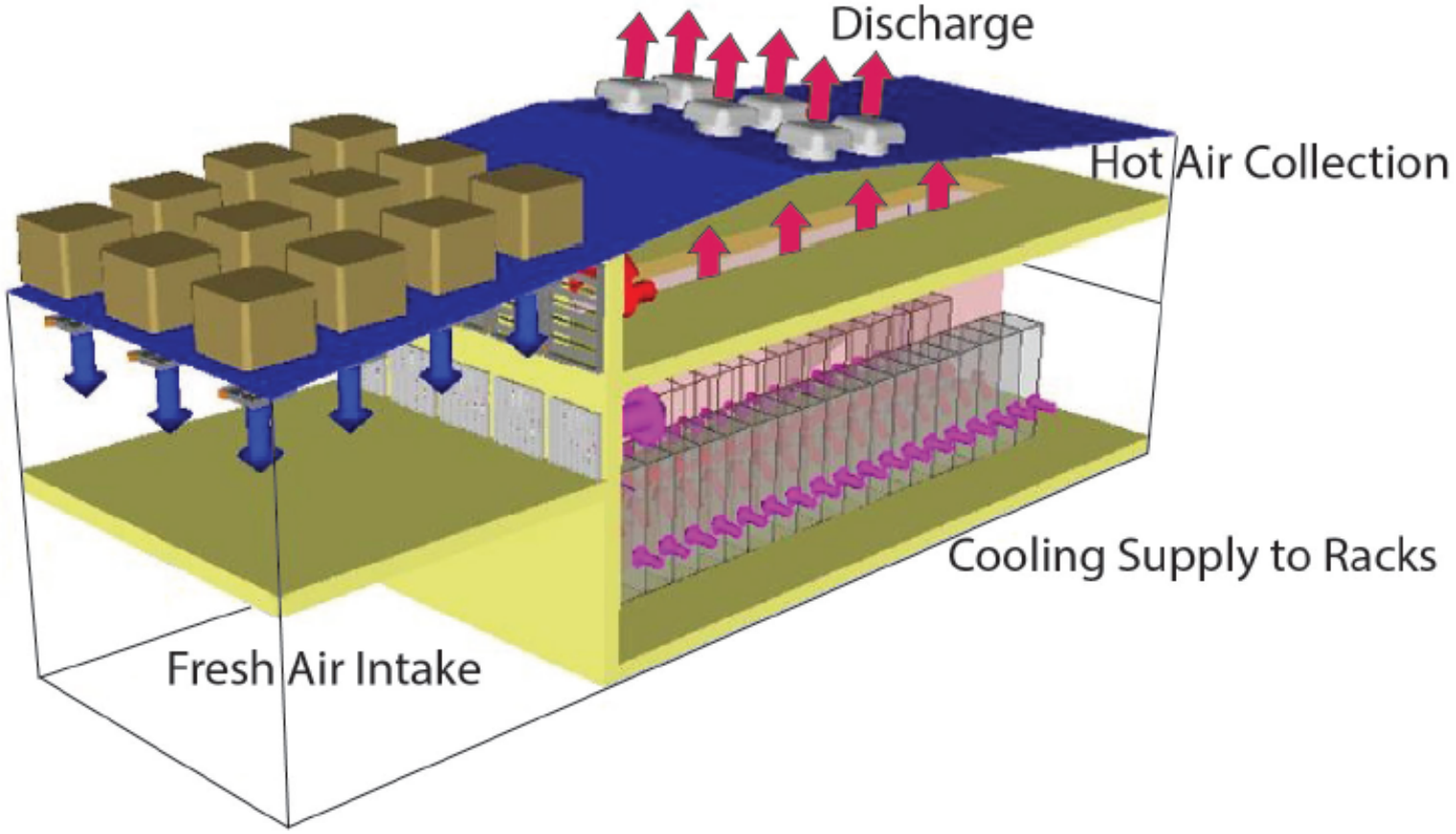


Figure 4.5: Airflow schematic of an air-economized datacenter (source: Perspectives, James Hamilton's blog).

Keeping it cool #2: Two-loop cooling

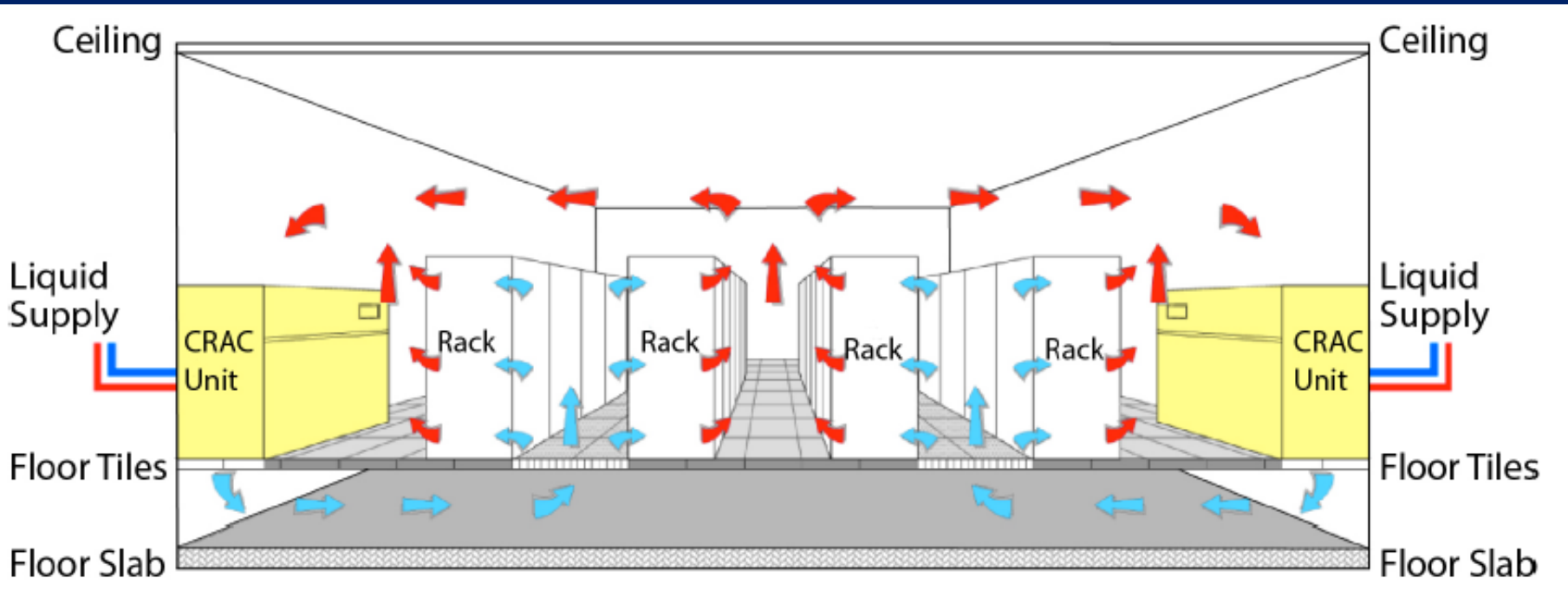


Figure 4.6: Datacenter raised floor with hot–cold aisle setup (image courtesy of DLB Associates [45]).

Keeping it cool #2: Two-loop cooling – Loop 1

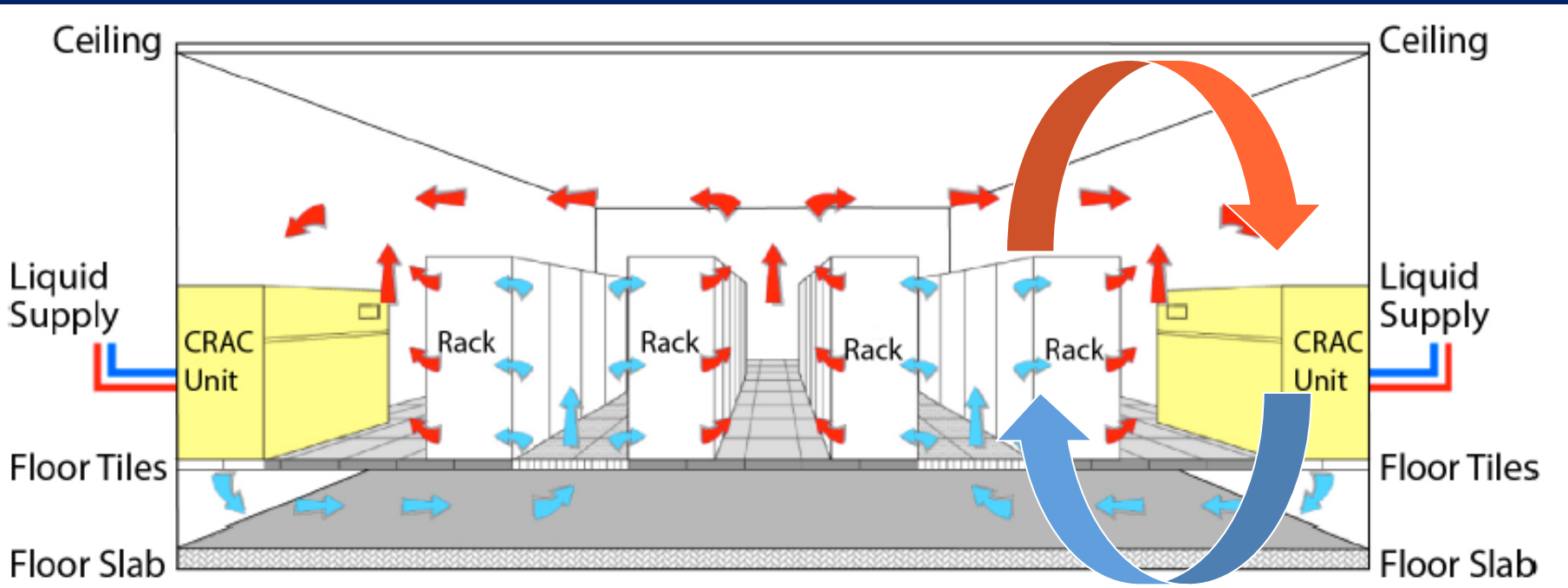


Figure 4.6: Datacenter raised floor with hot–cold aisle setup (image courtesy of DLB Associates [45]).

Keeping it cool #2: Two-loop cooling – Loop 2

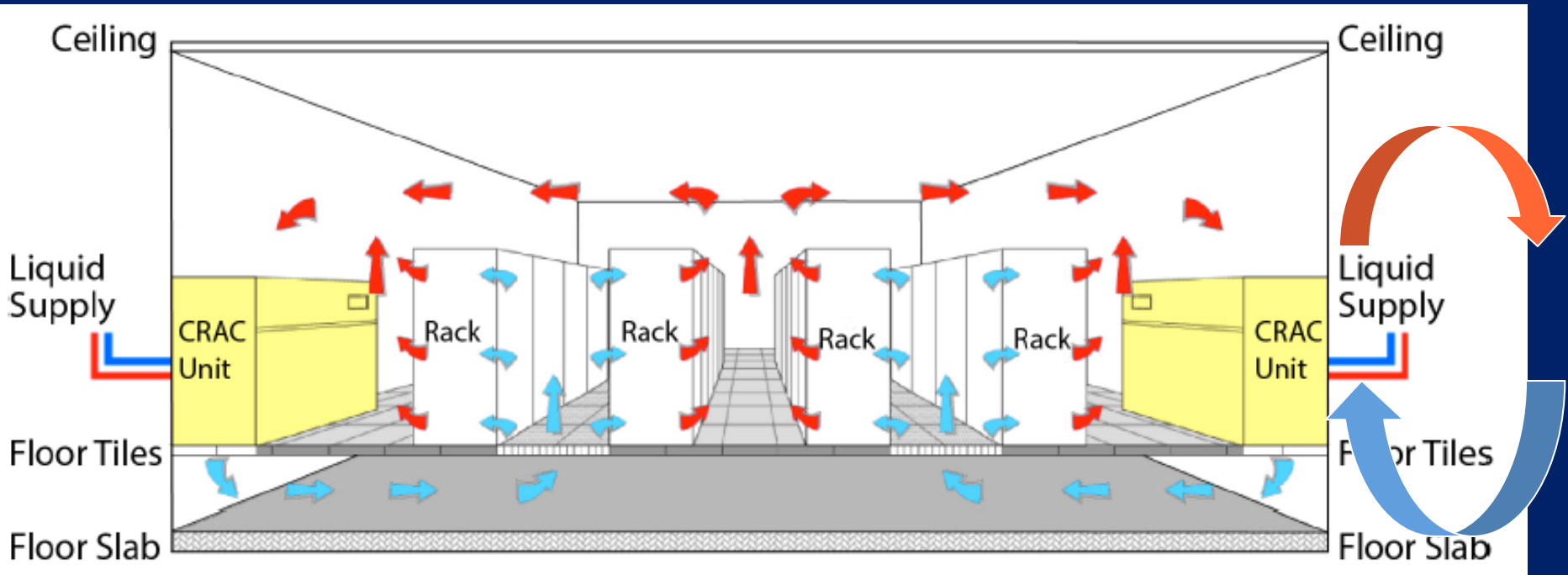


Figure 4.6: Datacenter raised floor with hot–cold aisle setup (image courtesy of DLB Associates [45]).

Keeping it cool #3: Three-loop cooling

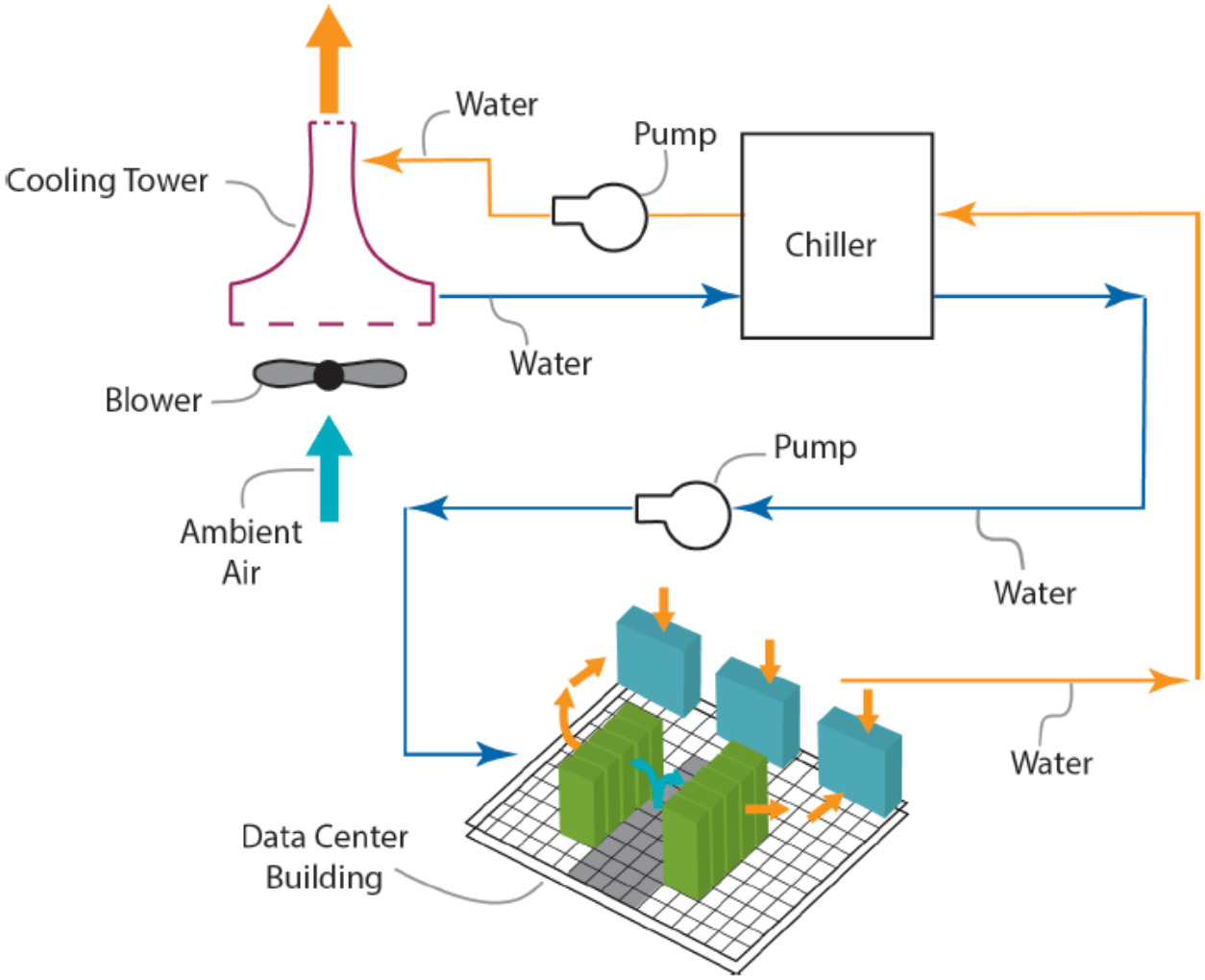


Figure 4.7: Three-loop datacenter cooling system (source: Iyengar, Schmidt, Energy Consumption of Information Technology Data Centers).

Keeping it cool #3: Three-loop cooling – Loop 1

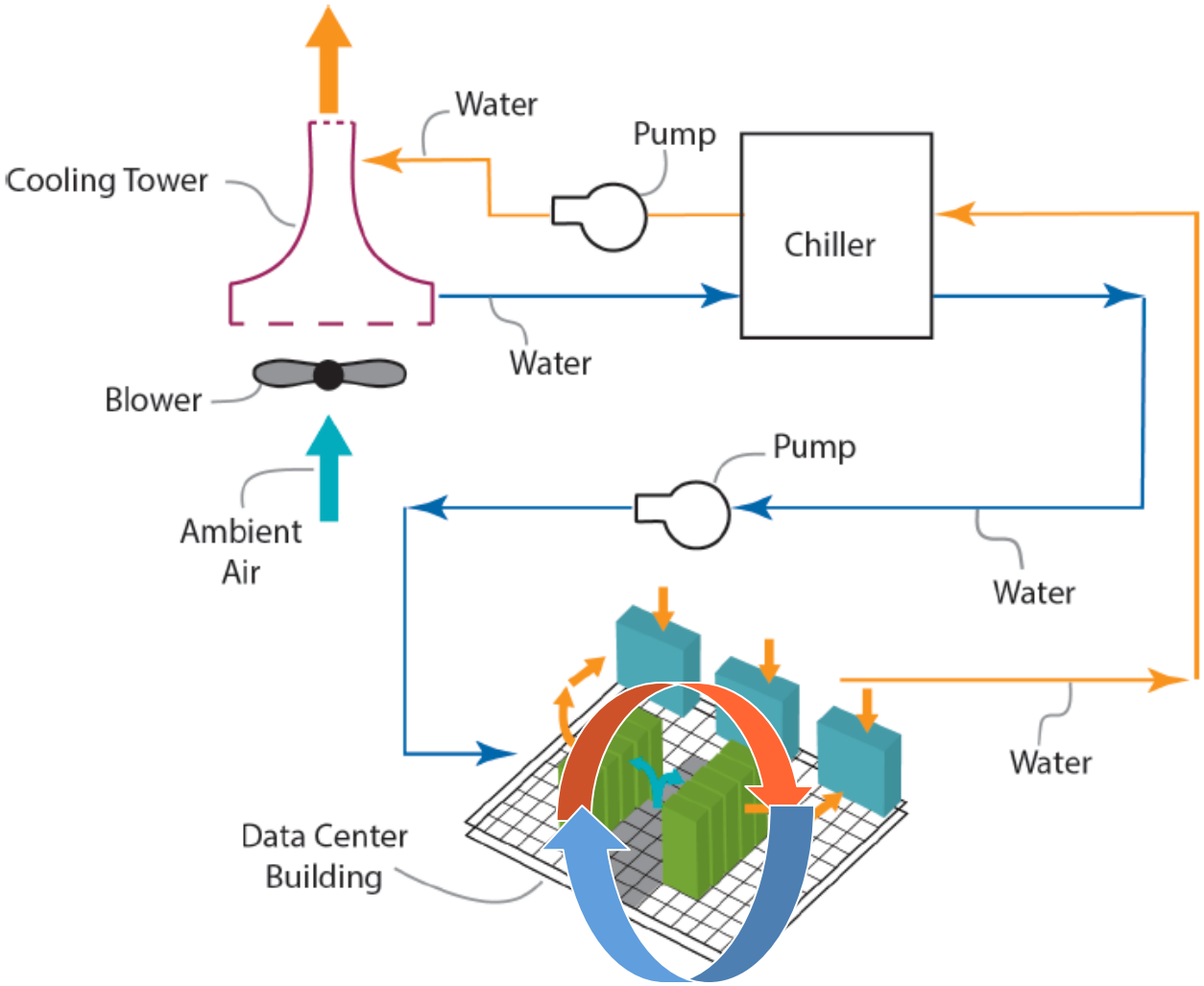


Figure 4.7: Three-loop datacenter cooling system (source: Iyengar, Schmidt, Energy Consumption of Information Technology Data Centers).

Keeping it cool #3: Three-loop cooling – Loop 2

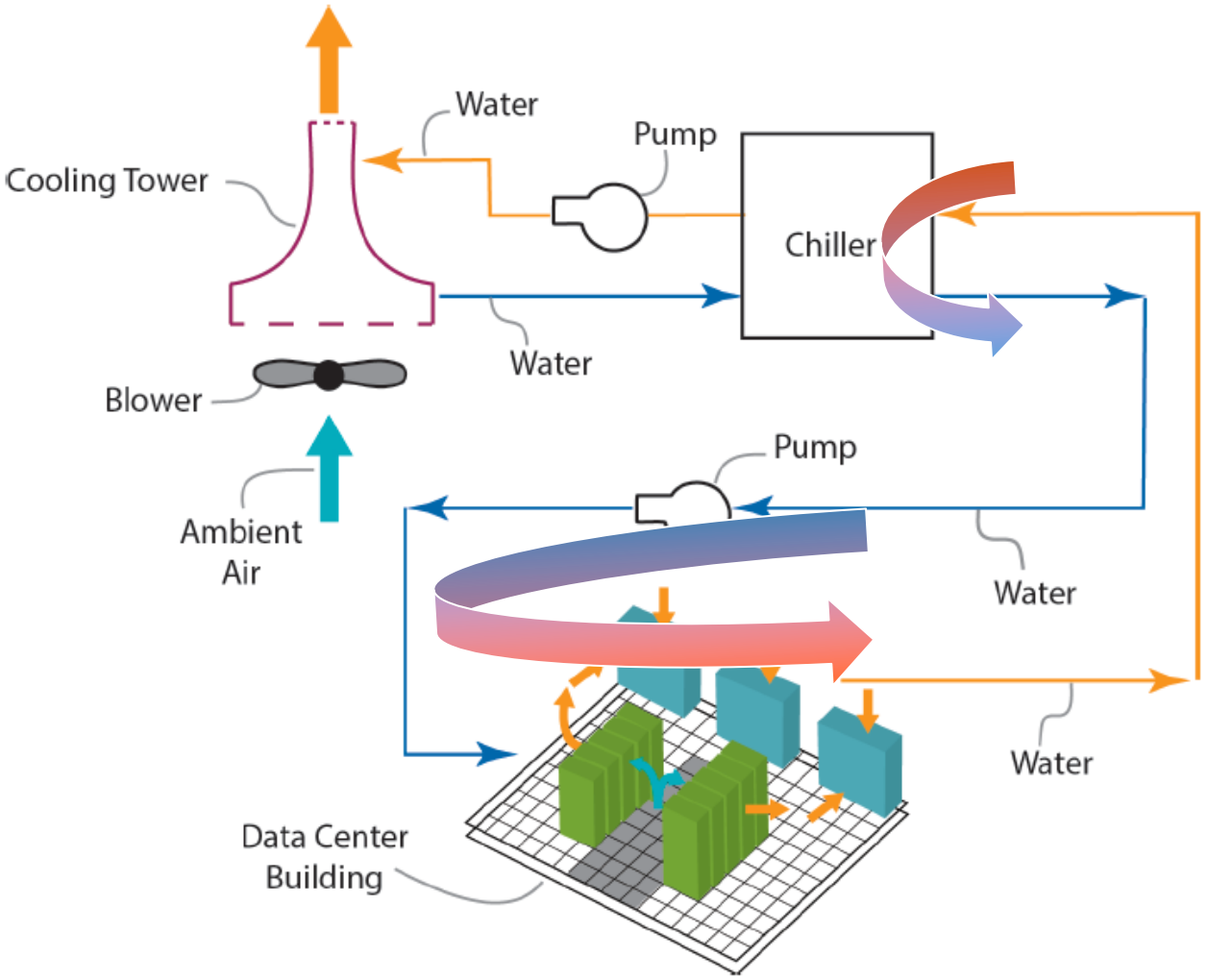


Figure 4.7: Three-loop datacenter cooling system (source: Iyengar, Schmidt, Energy Consumption of Information Technology Data Centers).

Keeping it cool #3: Three-loop cooling – Loop 3

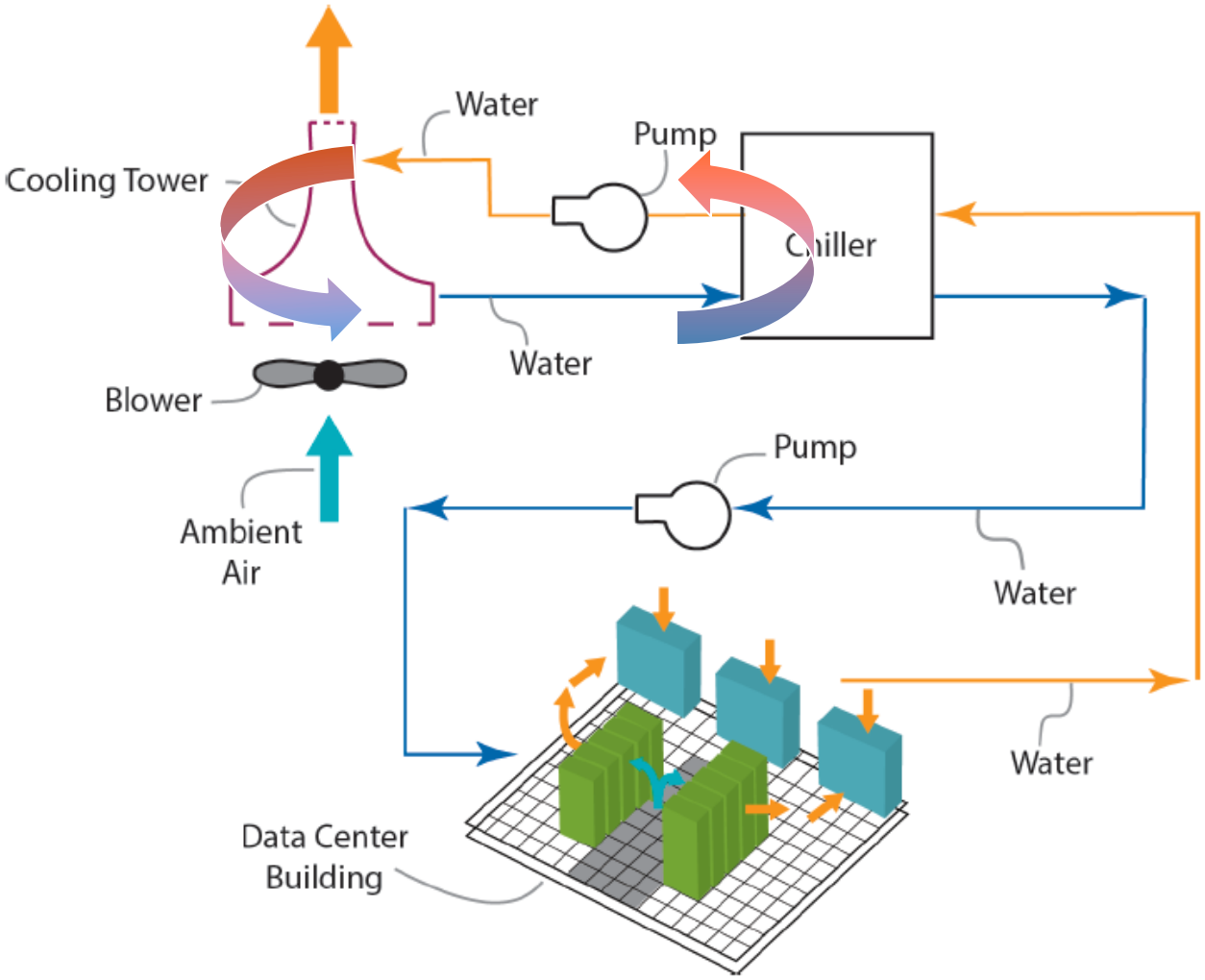


Figure 4.7: Three-loop datacenter cooling system (source: Iyengar, Schmidt, Energy Consumption of Information Technology Data Centers).

Keeping it cool: the bottom line

Source: BCH (2013) p.65

"Datacenters power the servers they contain, and remove the heat being generated. Historically, data centers have consumed twice as much energy as is needed just to power the servers, but with good engineering this overhead shrinks to 10–20%. Key energy saving techniques include free cooling (further boosted by raising the target inlet temperature of servers), well-managed air flow, and high-efficiency power conversions and UPSes."

Power Usage Proportions

Source: BCH (2013) Ch.1

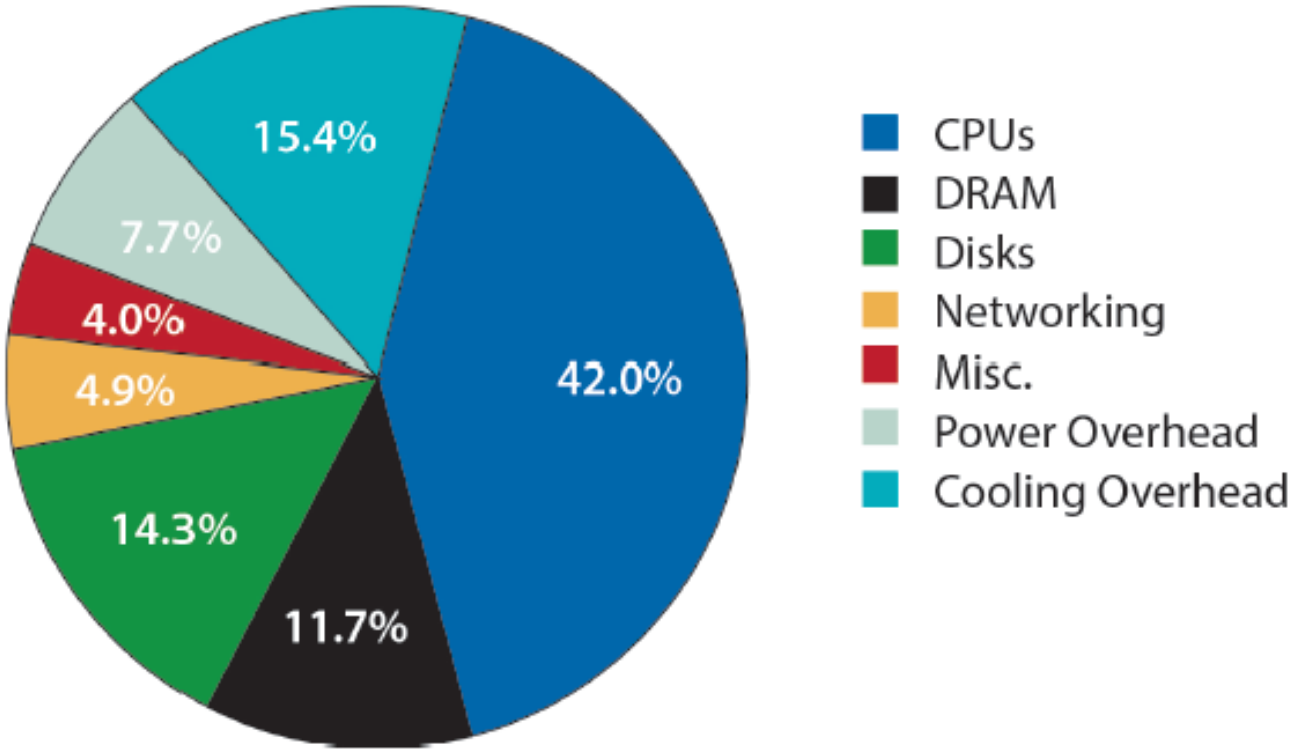


Figure 1.6: Approximate distribution of peak power usage by hardware subsystem in a modern data-center using late 2012 generation servers and a facility energy overhead of 30%. The figure assumes two-socket x86 servers, 16 DIMMs and 8 disk drives per server, and an average utilization of 80%.

Energy & Power Efficiency

Source: BCH (2013) Ch.5

"Power usage effectiveness (PUE) reflects the quality of the datacenter building infrastructure itself, and captures the ratio of total building power to IT power (the power consumed by the actual computing and network equipment, etc.)."

PUE = (Facility power) / (IT Equipment power)

"Very large operators (usually consumer Internet companies like Google, Microsoft, Yahoo!, Facebook, and eBay) have reported excellent PUE results over the past few years, typically below 1.2, though only Google has provided regular updates of their entire fleet based on a clearly defined metric … At scale, it is easy to justify special attention to efficiency; for example, Google reported having saved over one billion dollars to date from energy efficiency measures."

Energy efficiency is a key cost driver for WSCs, and we expect energy usage to become an increasingly important factor of WSC design. The current state of the industry is poor … mostly because efficiency has historically been neglected and has taken a backseat relative to reliability, performance, and capital expenditures. As a result, the average WSC wastes two thirds or more of its energy.

All of the stuff so far is more mech/elec/aero eng than CS

Let's concentrate on the computational aspects…

WSC: Hardware Building Blocks

Source: BCH (2013) Ch3.

- The architecture of a warehouse-scale computer (WSC) is largely defined by the choice of its building blocks. Cf choosing chips for a server.
- In WSC, the main building blocks are server hardware, networking fabric, and storage hierarchy components.
- WSCs typically built from clusters of low-end servers.
- Primary reason for this is the cost-efficiency of low-end servers when compared with the high-end shared-memory SMP (symmetric multi-processing) systems that had earlier been the preferred building blocks for HPC and technical computing. (High-end SMP servers are a dying market)
- Low-end server platforms share many key components with the very high-volume personal computing market, and therefore benefit more substantially from (huge) economies of scale.
- BCH observed in first edition of their book (2009) that the economics of the server space made hi-end SMP servers nearly obsolete in today's marketplace.
- The more interesting discussions today are between low-end server nodes and extremely low-end (so-called "Wimpy") servers

WSC: Impact of Communication Efficiency

Source: BCH (2013) Ch3.

- Simple processor-centric cost-efficiency analyses do not account for the fact that large SMPs benefit from drastically superior intercommunication performance wrt clusters of low-end servers connected by commodity network fabrics.
- Nodes in a large SMP may communicate at latencies on the order of 100ns, whereas the LAN-based networks usually deployed in clusters of servers will experience latencies at or above 100μs (i.e, 1000 times slower).
- For parallel applications that fit within a single large SMP the efficient comms can translate into big performance gains. However, WSC workloads are unlikely to fit within an SMP, so it is important to understand what is the relative performance of clusters of large SMPs with respect to clusters of low-end servers.
- Simple model:

  Assume fixed local computation time of 1ms (0.001sec), plus latency of access to data;

  Data access from RAM is c. 100ns; data access across LAN is c. 100μs;

  f is # of data accesses per 1ms of compute time (i.e., heaviness of communication);

  N is # of nodes; fraction of data accesses that map to local node is inv.prop. to N…

Execution time = 1ms + f * [100ns / N + 100μs * (1 − 1/N)]

WSC: Impact of Communication Efficiency

Source: BCH (2013) Ch3.

- Simple processor-centric cost-efficiency analyses do not account for the fact that large SMPs benefit from drastically superior intercommunication performance wrt clusters of low-end servers connected by commodity network fabrics.
- Nodes in a large SMP may communicate at latencies on the order of 100ns, whereas the LAN-based networks usually deployed in clusters of servers will experience latencies at or above 100μs (i.e, 1000 times slower).
- For parallel applications that fit within a single large SMP the efficient comms can translate into big performance gains. However, WSC workloads are unlikely to fit within an SMP, so it is important to understand what is the relative performance of clusters of large SMPs with respect to clusters of low-end servers.
- Simple model:
  - Assume fixed , plus latency of access to data;
  - Data access f LAN is c. 100μs;
  - f is # of data eaviness of communication);
  - N is # of nodes; fracti data accesses that map to local node is inv.prop. to N…

The assumption here is that accesses to global store are uniformly distributed among all nodes. (Seems reasonable)

Execution time = 1ms + f * [100ns / N + 100μs * (1 − 1/N)]

WSC: Impact of Communication Efficiency

Source: BCH (2013) Ch3.

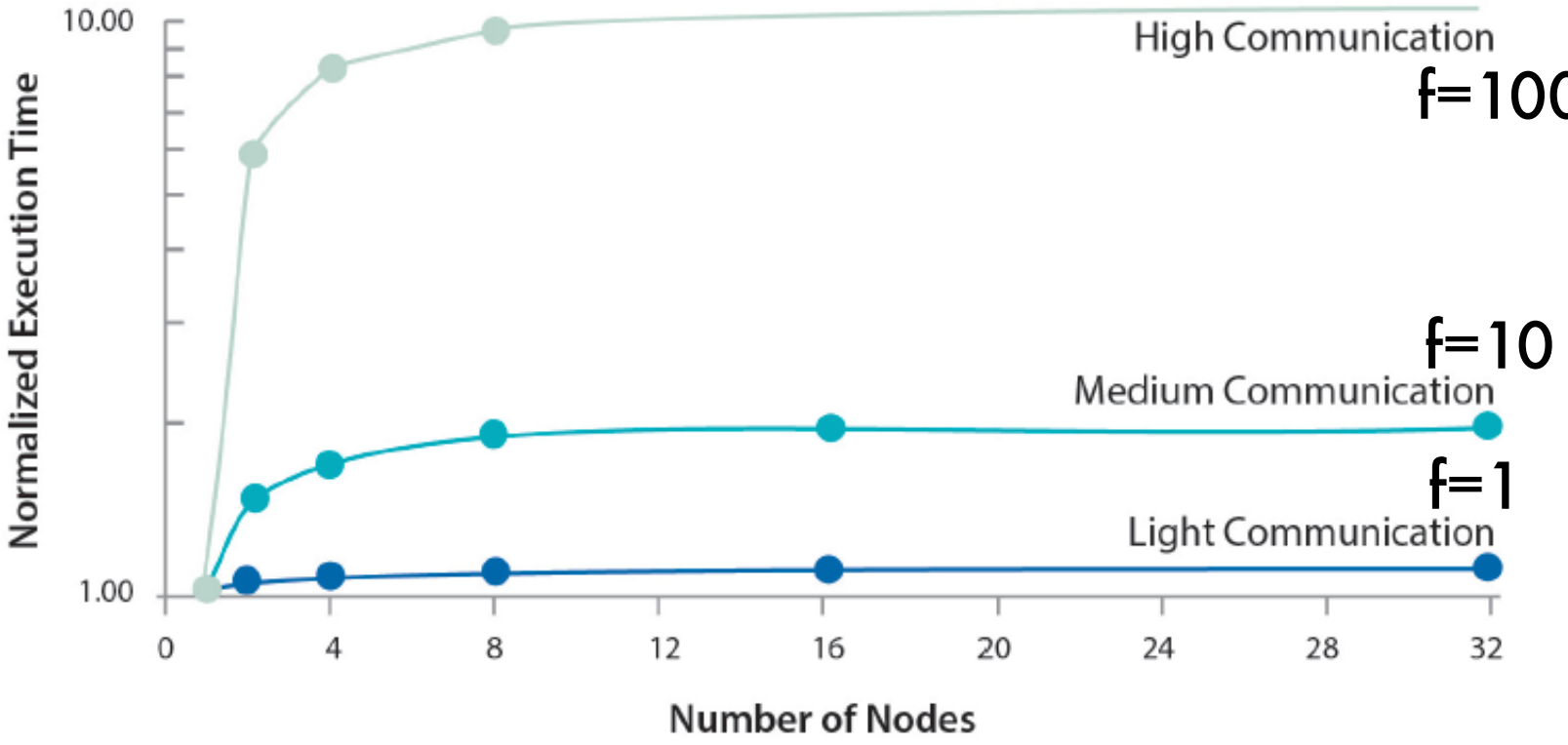


Figure 3.1: Execution time of parallel tasks as the number of SMP nodes increases for three levels of communication intensity. Execution time is normalized to the single node case and plotted in logarithmic scale.

WSC: Impact of Communication Efficiency

Source: BCH (2013) Ch3.

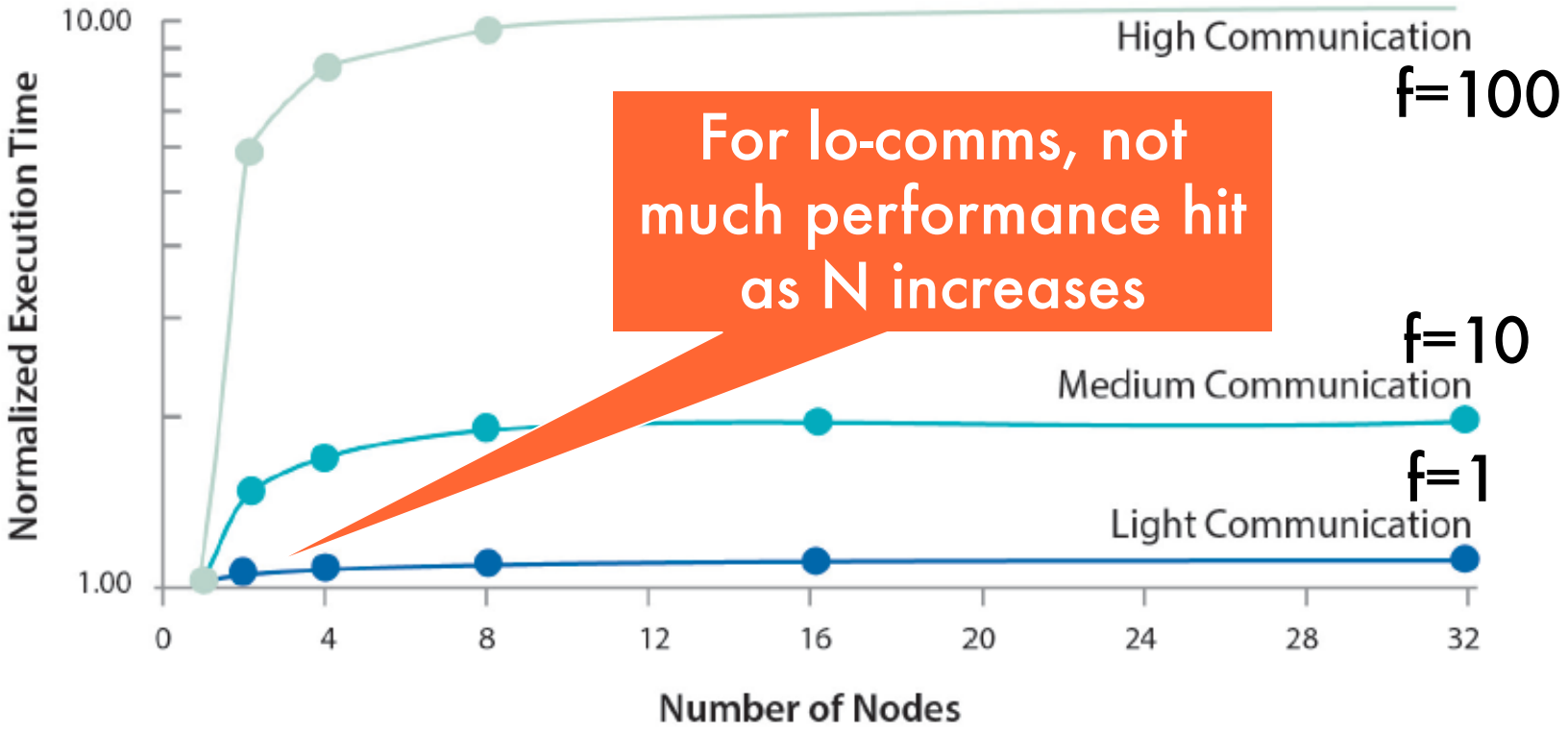


Figure 3.1: Execution time of parallel tasks as the number of SMP nodes increases for three levels of communication intensity. Execution time is normalized to the single node case and plotted in logarithmic scale.

WSC: Impact of Communication Efficiency

Source: BCH (2013) Ch3.

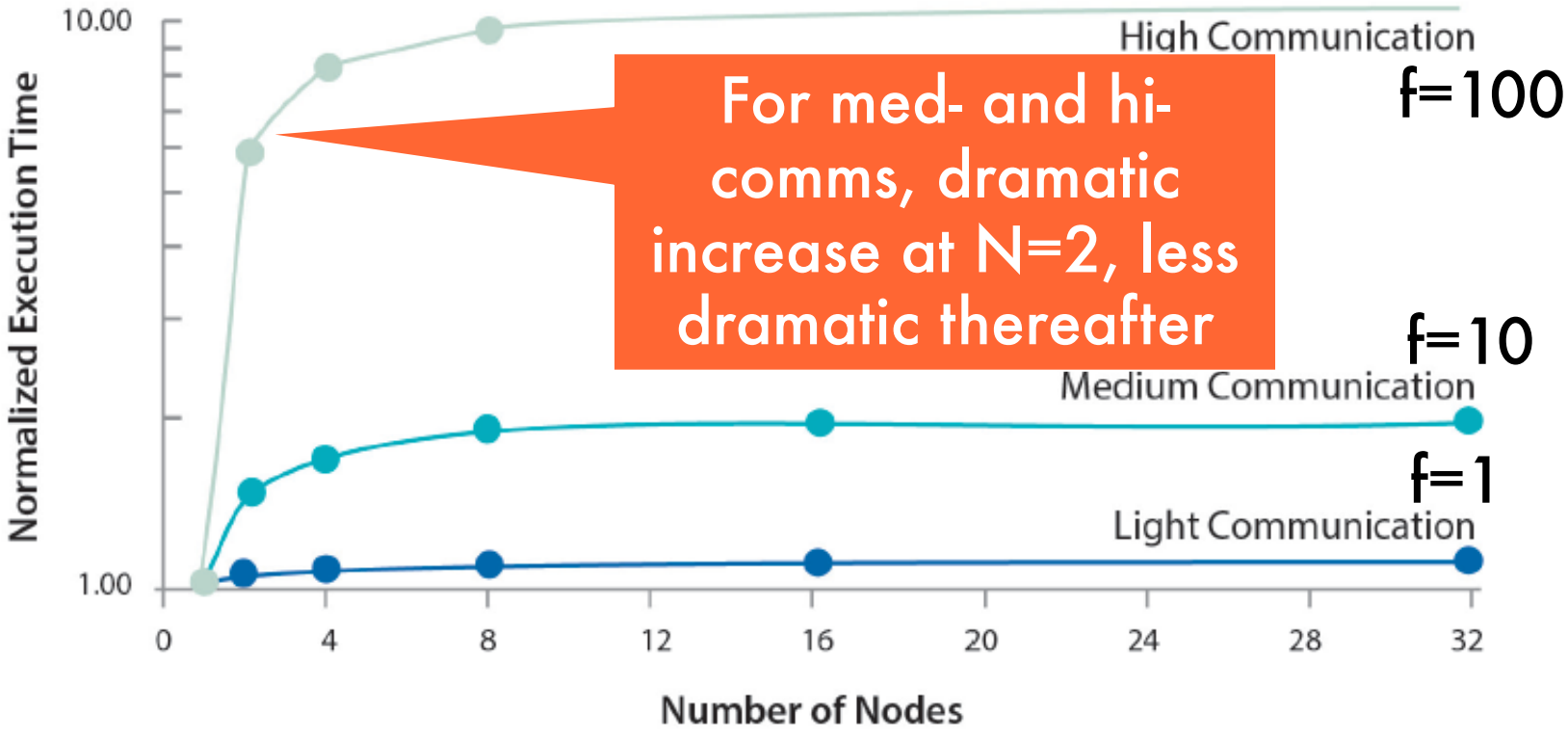


Figure 3.1: Execution time of parallel tasks as the number of SMP nodes increases for three levels of communication intensity. Execution time is normalized to the single node case and plotted in logarithmic scale.

WSC: Impact of Communication Efficiency

Source: BCH (2013) Ch3.

So this model implies that for hi-comms (e.g.) a single 128-core SMP node would be way better than using 32 x 4-core nodes.

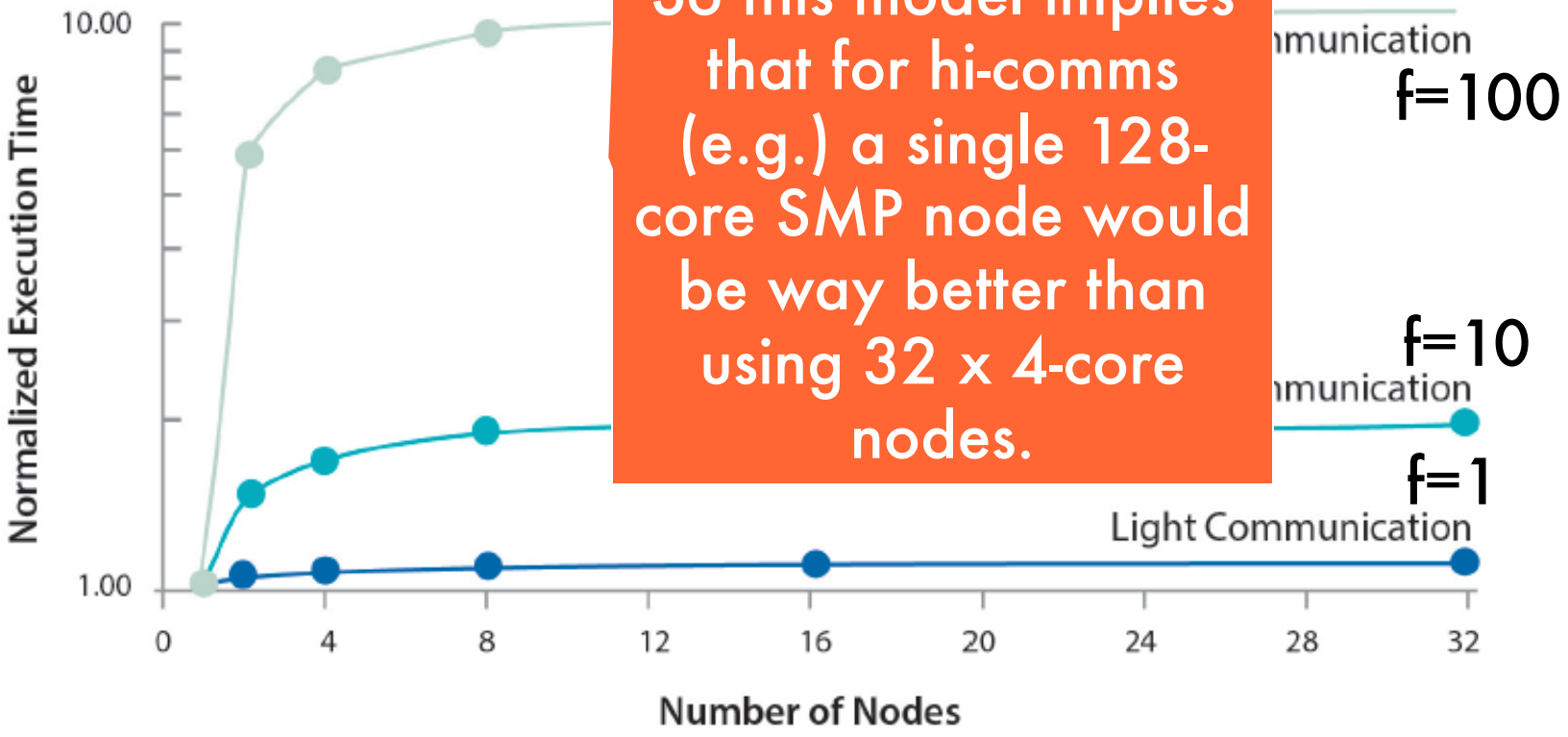


Figure 3.1: Execution time of parallel tasks as the number of SMP nodes increases for three levels of communication intensity. Execution time is normalized to the single node case and plotted in logarithmic scale.

WSC: Impact of Communication Efficiency

But, N=32 is small-scale.

Let's look at $N=2^9$, 2^{10}, 2^{11}, 2^{12}…

WSC: Impact of Communication Efficiency

Source: BCH (2013) Ch3.

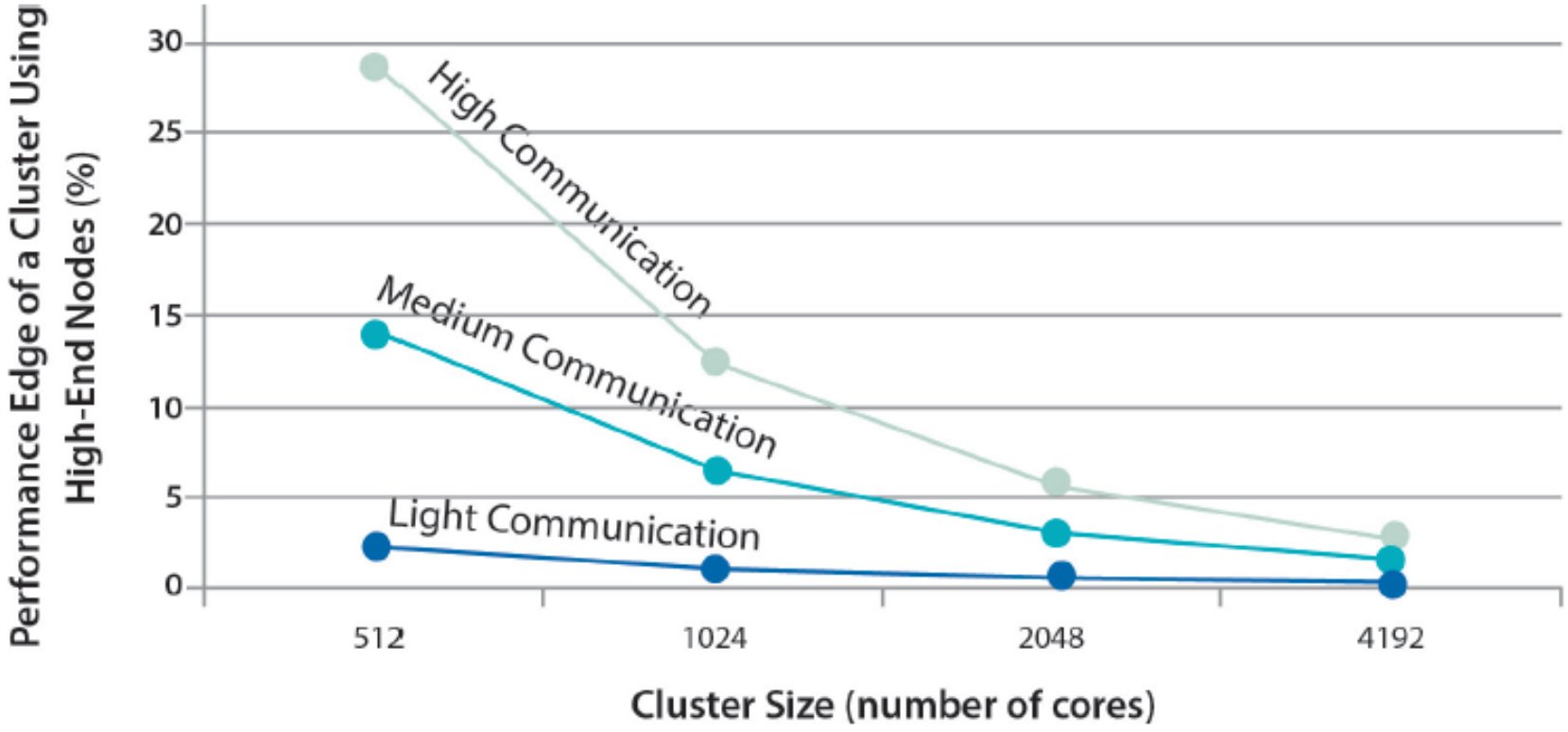
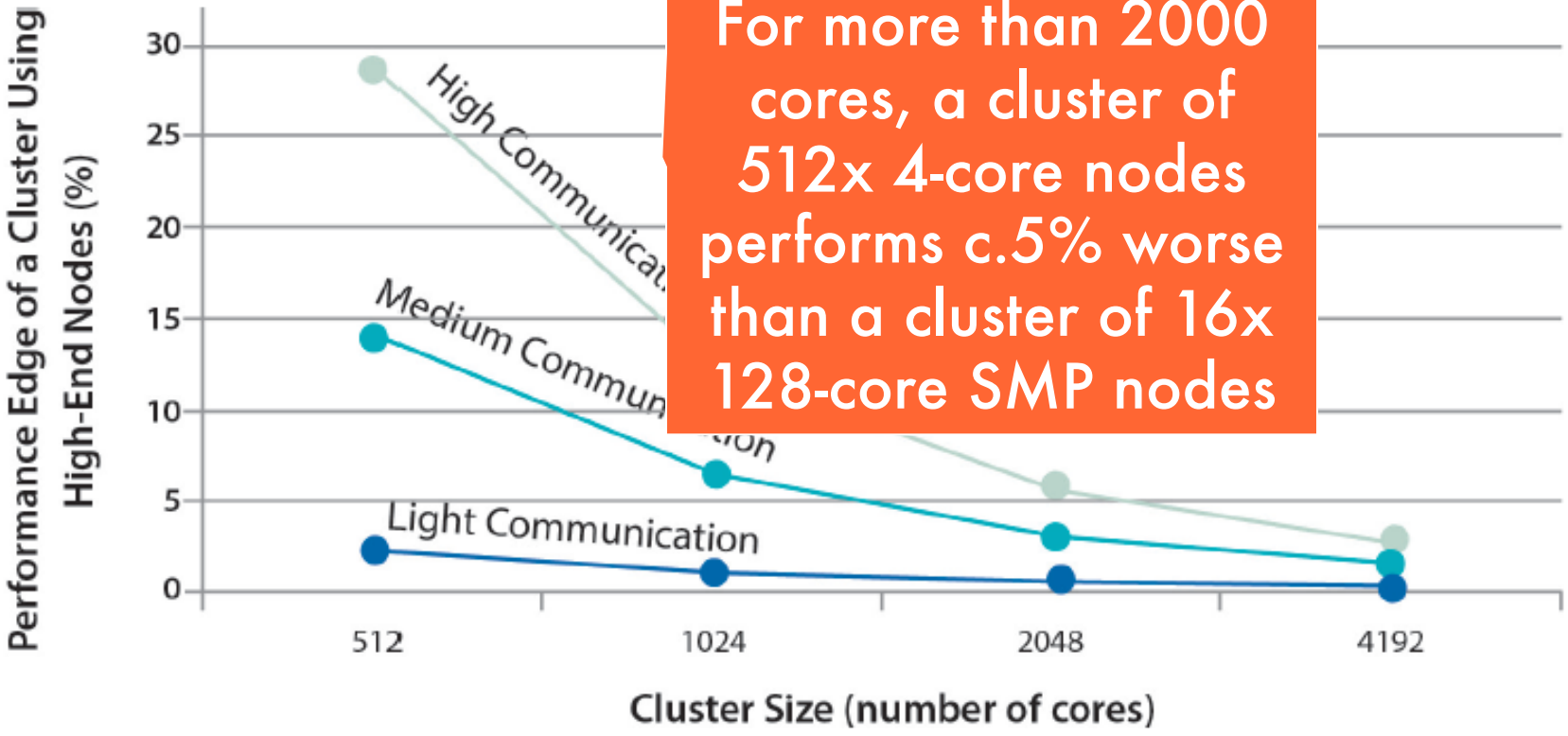


Figure 3.2: Performance advantage of a cluster built with large SMP server nodes (128-core SMP) over a cluster with the same number of processor cores built with low-end server nodes (four-core SMP), for clusters of varying size.

WSC: Impact of Communication Efficiency

Source: BCH (2013) Ch3.

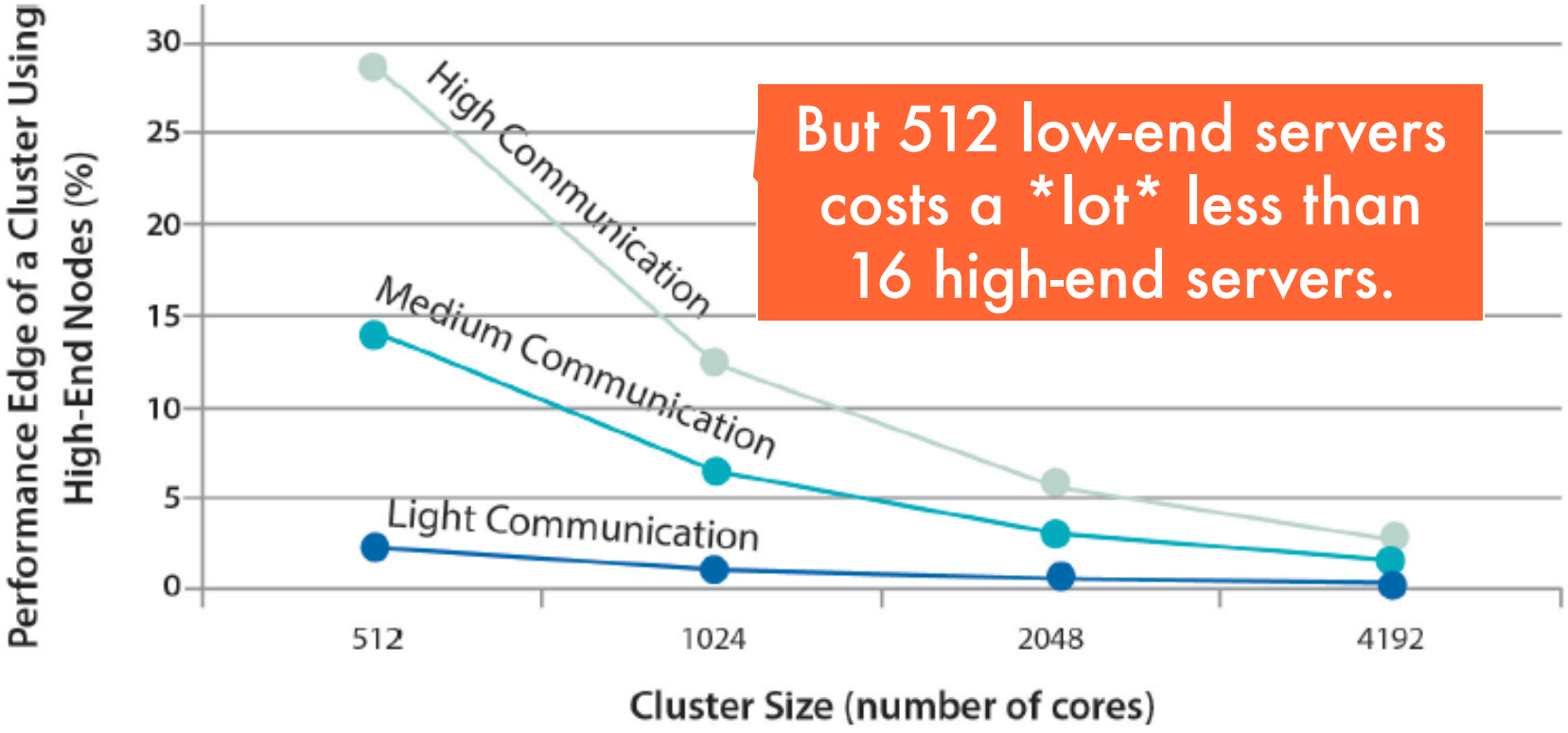


For more than 2000 cores, a cluster of 512x 4-core nodes performs c.5% worse than a cluster of 16x 128-core SMP nodes

Figure 3.2: Performance advantage of a cluster built with large SMP server nodes (128-core SMP) over a cluster with the same number of processor cores built with low-end server nodes (four-core SMP), for clusters of varying size.

WSC: Impact of Communication Efficiency

Source: BCH (2013) Ch3.



But 512 low-end servers costs a *lot* less than 16 high-end servers.

Figure 3.2: Performance advantage of a cluster built with large SMP server nodes (128-core SMP) over a cluster with the same number of processor cores built with low-end server nodes (four-core SMP), for clusters of varying size.

Fault-tolerant computing is reasonably well developed:

N-plex redundancy deals with "normal failure"

Fault-tolerant computing is reasonably well developed.

Latency-tolerant computing is a newer field…

BCH2013 Ch2 revised/reprinted in CACM 56(2) 2013

CHAPTER 2

Workloads and Software Infrastructure

The applications that run on warehouse-scale computers (WSCs) dominate many system design trade-off decisions. This chapter outlines some of the distinguishing characteristics of software that runs in large Internet services and the system software and tools needed for a complete computing platform. Here is some terminology that defines the different software layers in a typical WSC deployment.

- *Platform-level software:* the common firmware, kernel, operating system distribution, and libraries expected to be present in all individual servers to abstract the hardware of a single machine and provide basic server-level services.
- *Cluster-level infrastructure:* the collection of distributed systems software that manages resources and provides services at the cluster level; ultimately, we consider these services as an operating system for a datacenter. Examples are distributed file systems, schedulers and remote procedure call (RPC) libraries, as well as programming models that simplify the usage of resources at the scale of datacenters, such as MapReduce [35], Dryad [86], Hadoop [77], Sawzall [122], BigTable [26], Dynamo [36], Dremel [104], Spanner [32], and Chubby [22].
- *Application-level software:* software that implements a specific service. It is often useful to further divide application-level software into online services and offline computations because those tend to have different requirements. Examples of online services are Google search, Gmail, and Google Maps. Offline computations are typically used in large-scale data analysis or as part of the pipeline that generates the data used in online services; for example, building an index of the Web or processing satellite images to create map tiles for the online service.

2.1 DATACENTER VS. DESKTOP

Software development in Internet services differs from the traditional desktop/server model in many ways.

- *Ample parallelism:* Typical Internet services exhibit a large amount of parallelism stemming from both data- and request-level parallelism. Usually, the problem is not to find

DOI:10.1145/2408776.2408794

Software techniques that tolerate latency variability are vital to building responsive large-scale Web services.

BY JEFFREY DEAN AND LUIZ ANDRÉ BARROSO

The Tail at Scale

SYSTEMS THAT RESPOND to user actions quickly (within 100ms) feel more fluid and natural to users than those that take longer.[3] Improvements in Internet connectivity and the rise of warehouse-scale computing systems[2] have enabled Web services that provide fluid responsiveness while consulting multi-terabyte datasets spanning thousands of servers; for example, the Google search system updates query results interactively as the user types, predicting the most likely query based on the prefix typed so far, performing the search and showing the results within a few tens of milliseconds. Emerging augmented-reality devices (such as the Google Glass prototype[7]) will need associated Web services with even greater responsiveness in order to guarantee seamless interactivity.

It is challenging for service providers to keep the tail of latency distribution short for interactive services as the size and complexity of the system scales up or as overall use increases. Temporary high-latency episodes (unimportant in moderate-size systems) may come to dominate overall service performance at large scale. Just as fault-tolerant computing aims to create a reliable whole out of less-reliable parts, large online services need to create a predictably responsive whole out of less-predictable parts; we refer to such systems as "latency tail-tolerant," or simply "tail-tolerant." Here, we outline some common causes for high-latency episodes in large online services and describe techniques that reduce their severity or mitigate their effect on whole-system performance. In many cases, tail-tolerant techniques can take advantage of resources already deployed to achieve fault-tolerance, resulting in low additional overhead. We explore how these techniques allow system utilization to be driven higher without lengthening the latency tail, thus avoiding wasteful overprovisioning.

Why Variability Exists?

Variability of response time that leads to high tail latency in individual components of a service can arise for many reasons, including:

Shared resources. Machines might be shared by different applications contending for shared resources (such as CPU cores, processor caches, memory bandwidth, and network bandwidth), and within the same application different requests might contend for resources;

Daemons. Background daemons may use only limited resources on average but when scheduled can generate multi-millisecond hiccups;

» key insights

- Even rare performance hiccups affect a significant fraction of all requests in large-scale distributed systems.
- Eliminating all sources of latency variability in large-scale systems is impractical, especially in shared environments.
- Using an approach analogous to fault-tolerant computing, tail-tolerant software techniques form a predictable whole out of less-predictable parts.

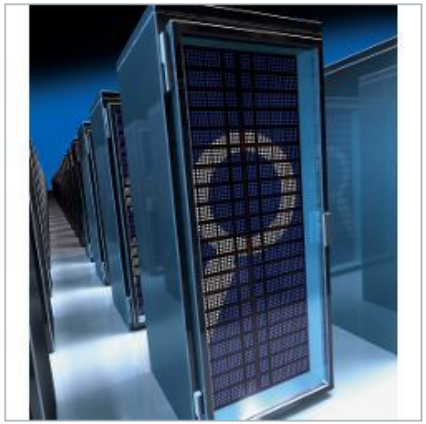
Required reading!!

http://cacm.acm.org/magazines/2013/2/160173-the-tail-at-scale/fulltext

CONTRIBUTED ARTICLES

The Tail at Scale

By Jeffrey Dean, Luiz Andr Barroso
Communications of the ACM, Vol. 56 No. 2, Pages 74-80
10.1145/2408776.2408794
Comments



Systems that respond to user actions quickly (within 100ms) feel more fluid and natural to users than those that take longer.[3] Improvements in Internet connectivity and the rise of warehouse-scale computing systems[2] have enabled Web services that provide fluid responsiveness while consulting multi-terabyte datasets spanning thousands of servers; for example, the Google search system updates query results interactively as the user types, predicting the most likely query based on the prefix typed so far, performing the search and showing the results within a few tens of milliseconds. Emerging augmented-reality devices (such as the Google Glass prototype[7]) will need associated Web services with even greater responsiveness in order to guarantee seamless interactivity.

Key Insights from *The Tail at Scale*

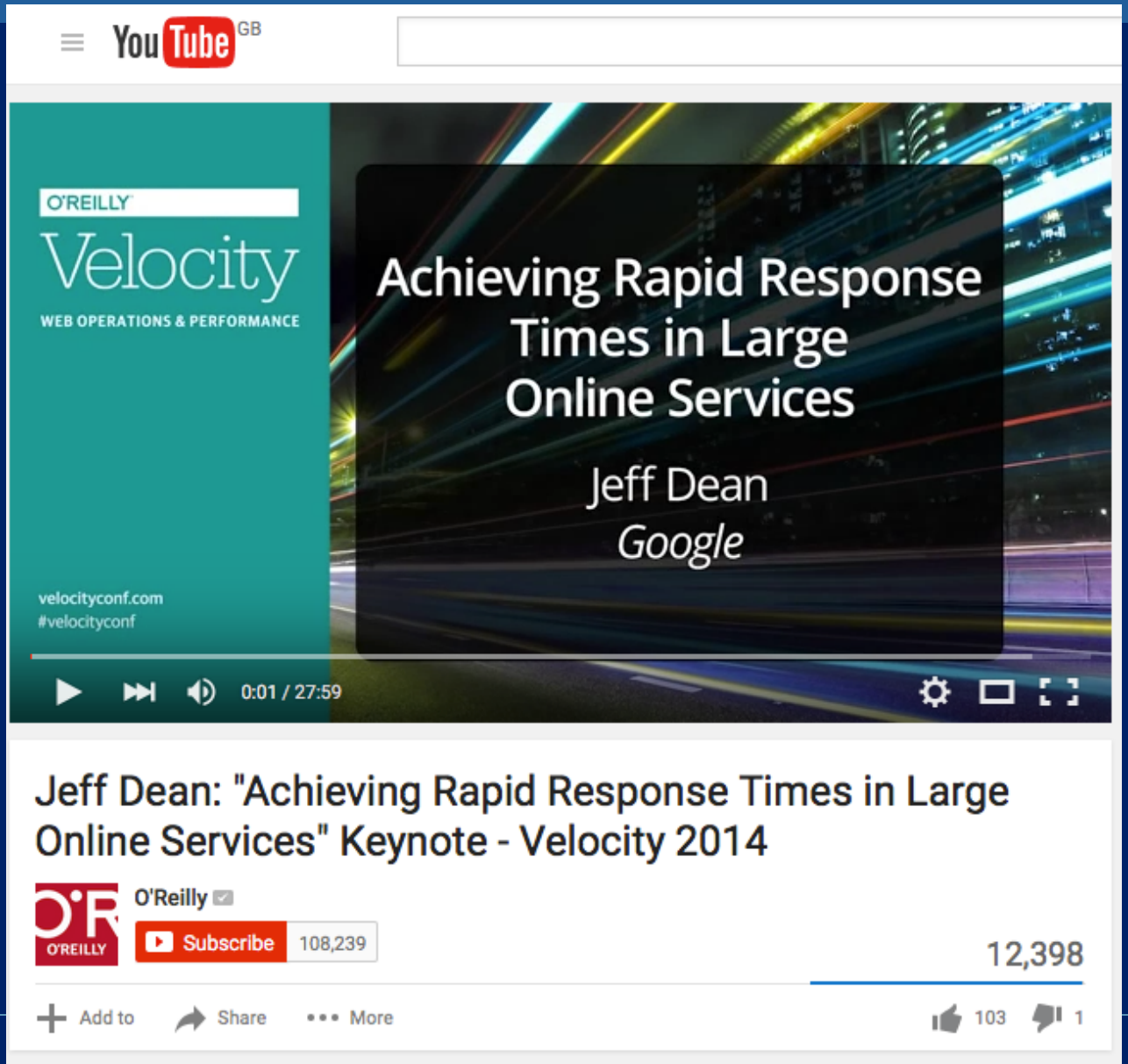
- In large-scale distributed systems, localised performance "hiccups" (sudden up-spikes in latency) can radiate out to affect a significant fraction of all requests.
  - Sources of variability: shared/scarce resources; woken daemons; maintenance activities; normal failure; queuing; GC; power limits & energy management.
  - E.g. Server response usually <10ms but Pr(>=1000ms)=0.01
    - If generating a user response requires only one server, one in 100 will be slow.
    - If response requires 100 servers working in parallel $1-0.99^{100}=63\%$ will be slow.
- In such systems, especially when they're shared, it is often not practicable (or possible) to eliminate all sources of latency variability.
- Latency tail-tolerant software techniques, an approach analogous to fault-tolerant computing, have been developed to smooth out unpredictability.

  "…form a predictable whole out of less-predictable parts"

  e.g. *Hedged requests* (cf *n*-plex redundancy) issue *n* requests, accept first response.

Jeff Dean lecture

https://www.youtube.com/watch?v=1-3Ahy7Fxsc



Summary

Economies of scale drive providers and users to WSC

There is a lot going on "under the hood" of a WSC!

Much of this involves advancing the state of the art, but prod>>dev

i.e. production environments are way, way bigger than development environments.

The leading-edge innovations and advances are being made by practitioners at Google, Amazon, MSFT, HP, Oracle etc – not by academics in university labs (a point we return to in the final lecture…)