Big Data Infrastructure

CS 489/698 Big Data Infrastructure (Winter 2016)

Week 1: Introduction (2/2)

January 7, 2016

Jimmy Lin
David R. Cheriton School of Computer Science
University of Waterloo

These slides are available at http://lintool.github.io/bigdata-2016w/

Why big data?

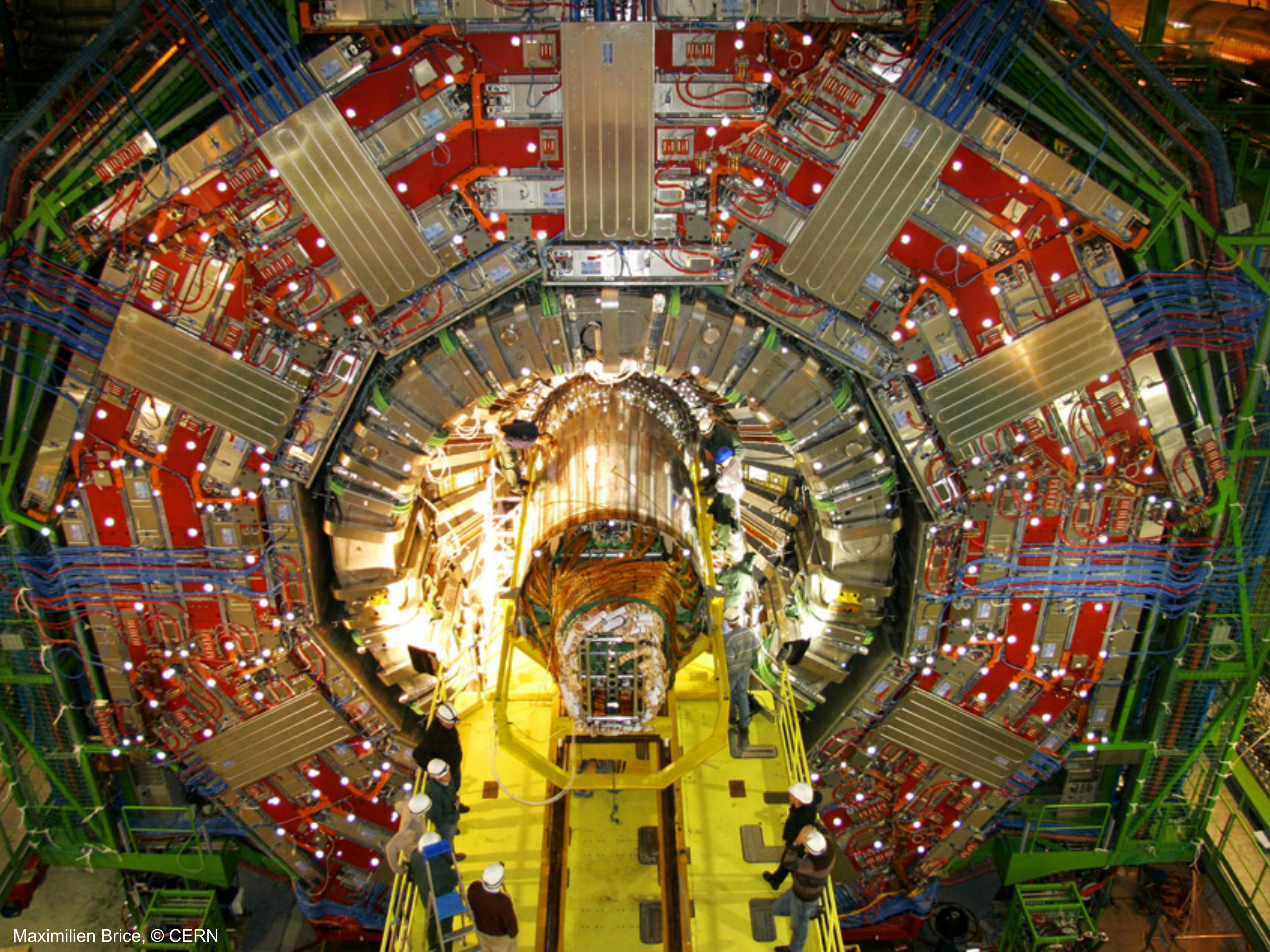
Science
Engineering
Commerce
Society

Source: Wikipedia (Everest)

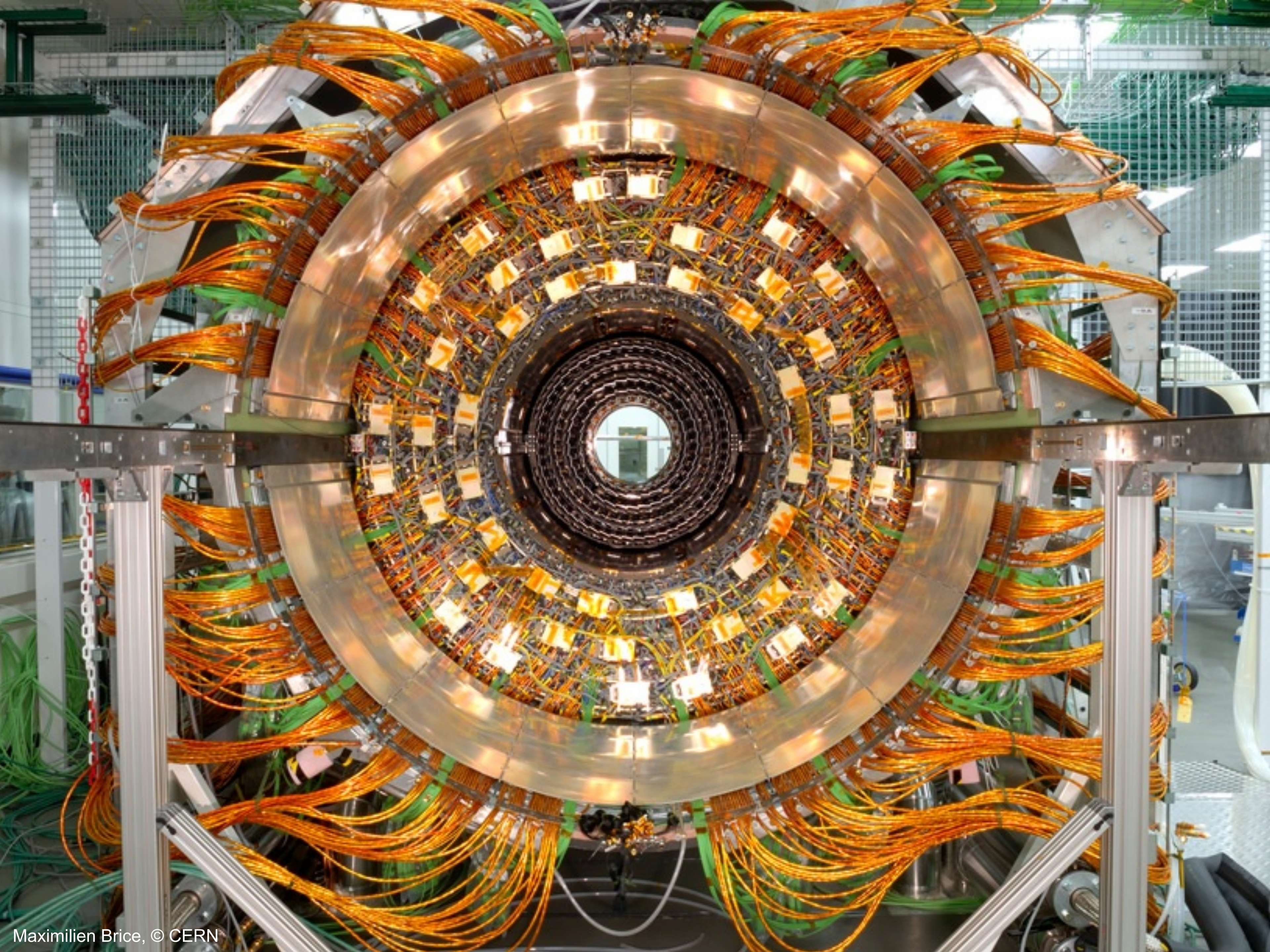
Science

Emergence of the 4^{th} Paradigm

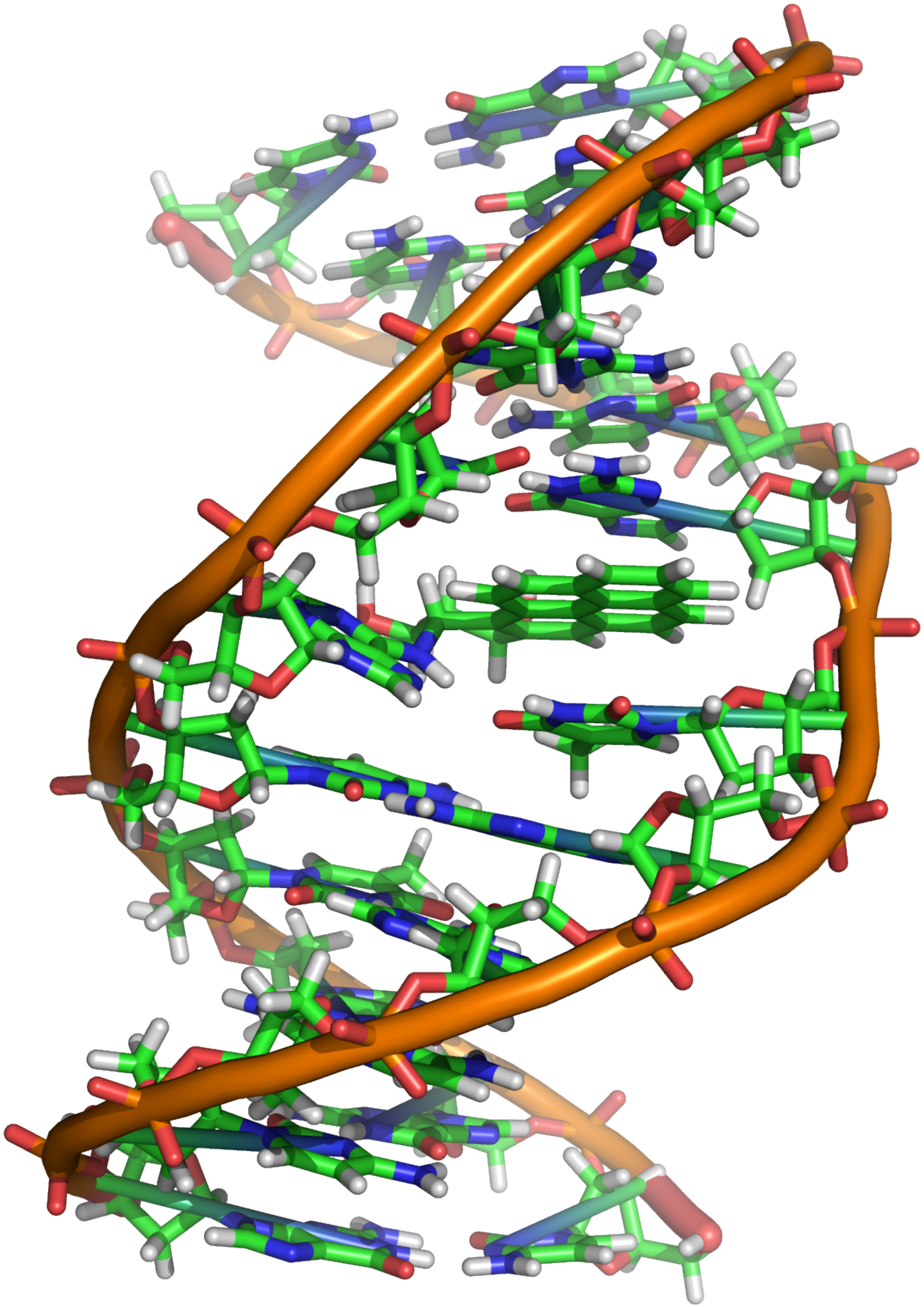
Data-intensive e-Science



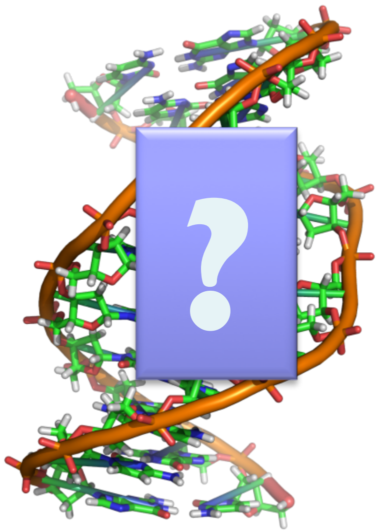
Maximilien Brice, © CERN



Source: Wikipedia (Galaxy)



Source: Wikipedia (DNA)



Subject genome

Sequencer

GATGCTTACTATGCGGGCCCC
CGGTCTAATGCTTACTATGC
GCTTACTATGCGGGCCCCTT
AATGCTTACTATGCGGGCCCCTT
TAATGCTTACTATGC
AATGCTTAGCTATGCGGGC
AATGCTTACTATGCGGGCCCCTT
AATGCTTACTATGCGGGCCCCTT
CGGTCTAGATGCTTACTATGC
AATGCTTACTATGCGGGCCCCTT
CGGTCTAATGCTTAGCTATGC
ATGCTTACTATGCGGGCCCCTT

Reads

Human genome: 3 gbp
A few billion short reads
(~100 GB compressed data)

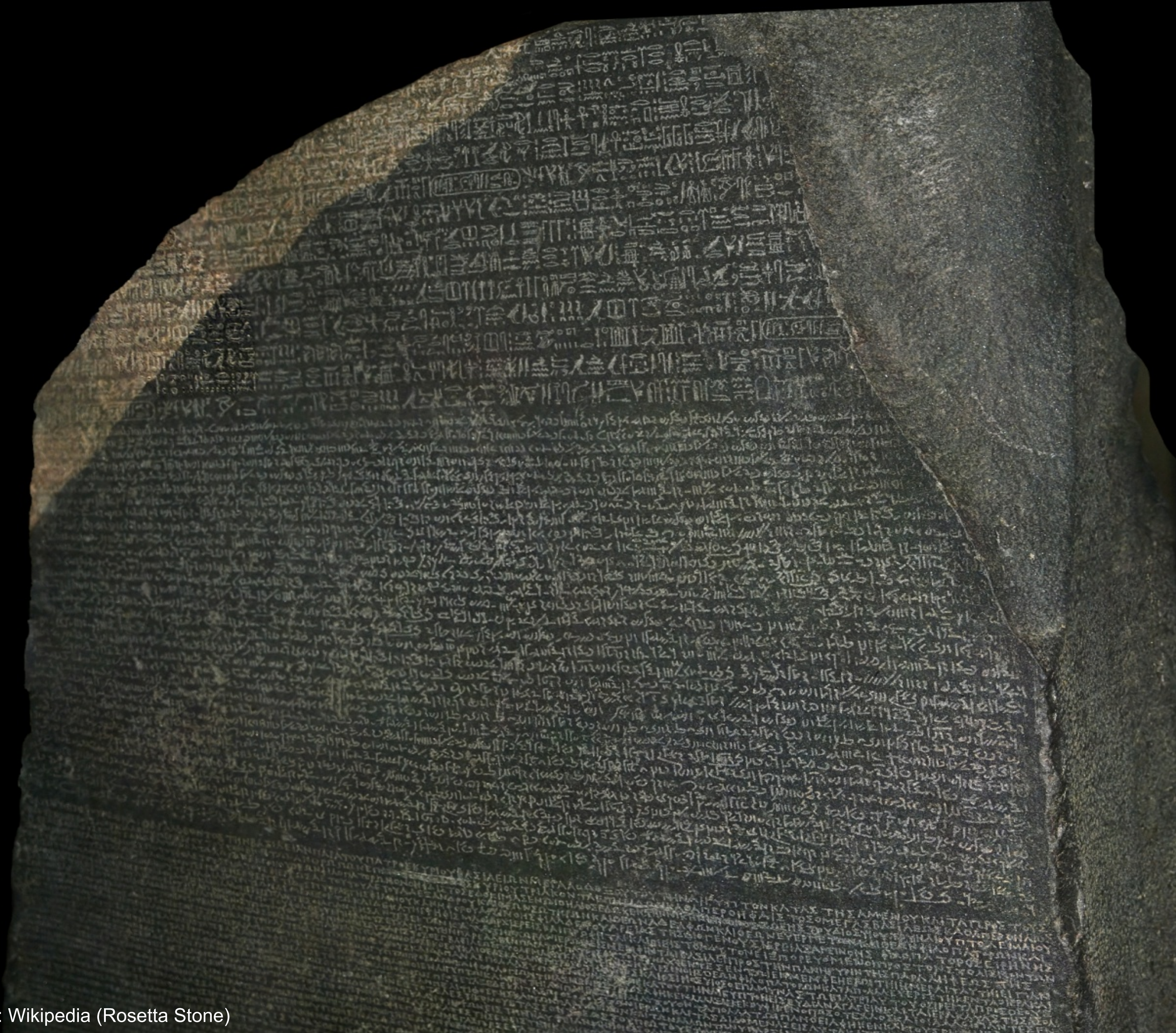
Engineering

The unreasonable effectiveness of data

Search, recommendation, prediction, …

Source: Wikipedia (Three Gorges Dam)

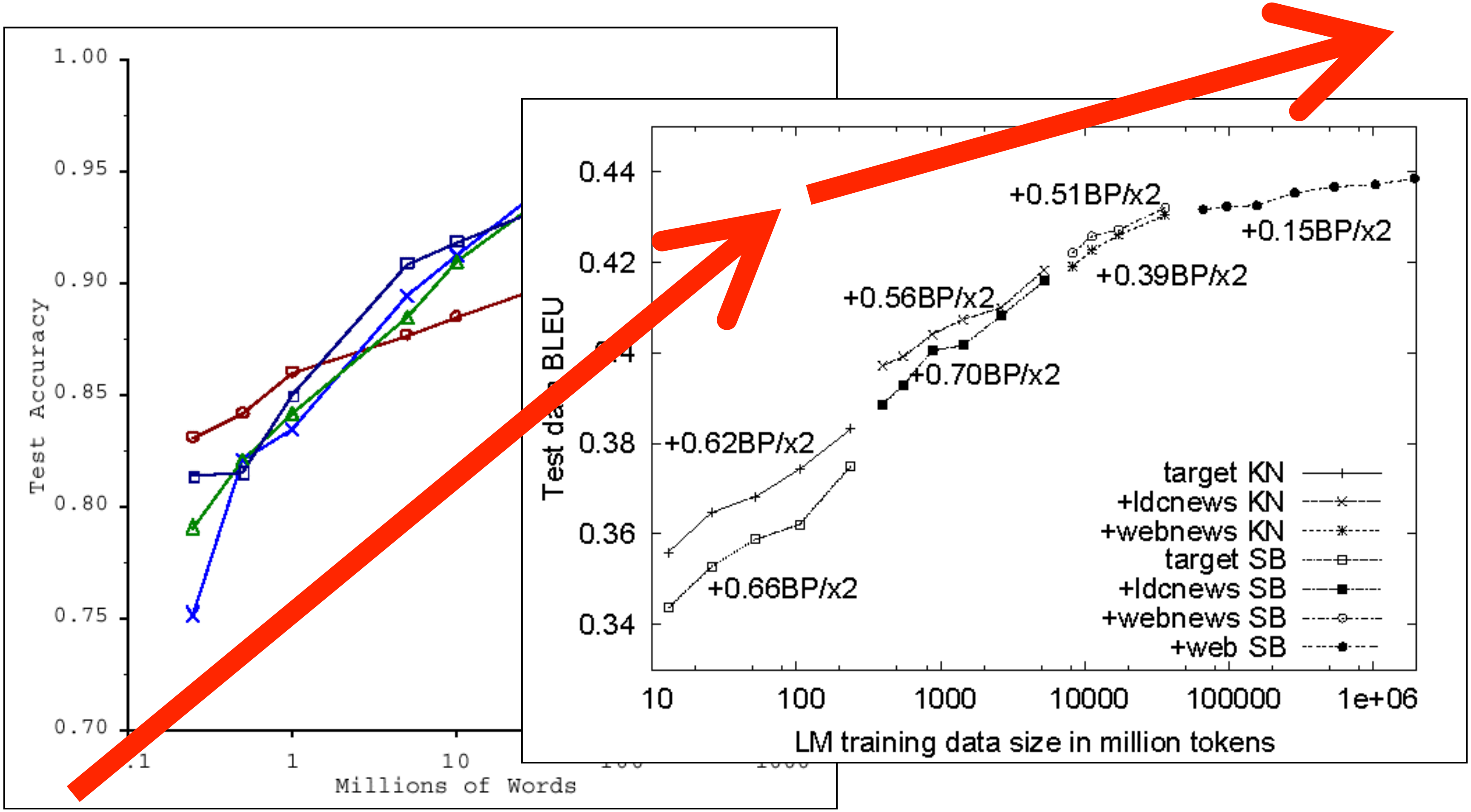

Translate
English
Spanish
French
English - detected
How does Google's translation system work?
如何谷歌的翻译系统的工作?
Wrong?
Rúhé gǔgē de fānyì xìtǒng de gōngzuò?



Source: Wikipedia (Rosetta Stone)

No data like more data!

s/knowledge/data/g;



(Banko and Brill, ACL 2001)
(Brants et al., EMNLP 2007)

Know thy customers

Data → Insights → Competitive advantages

Commerce

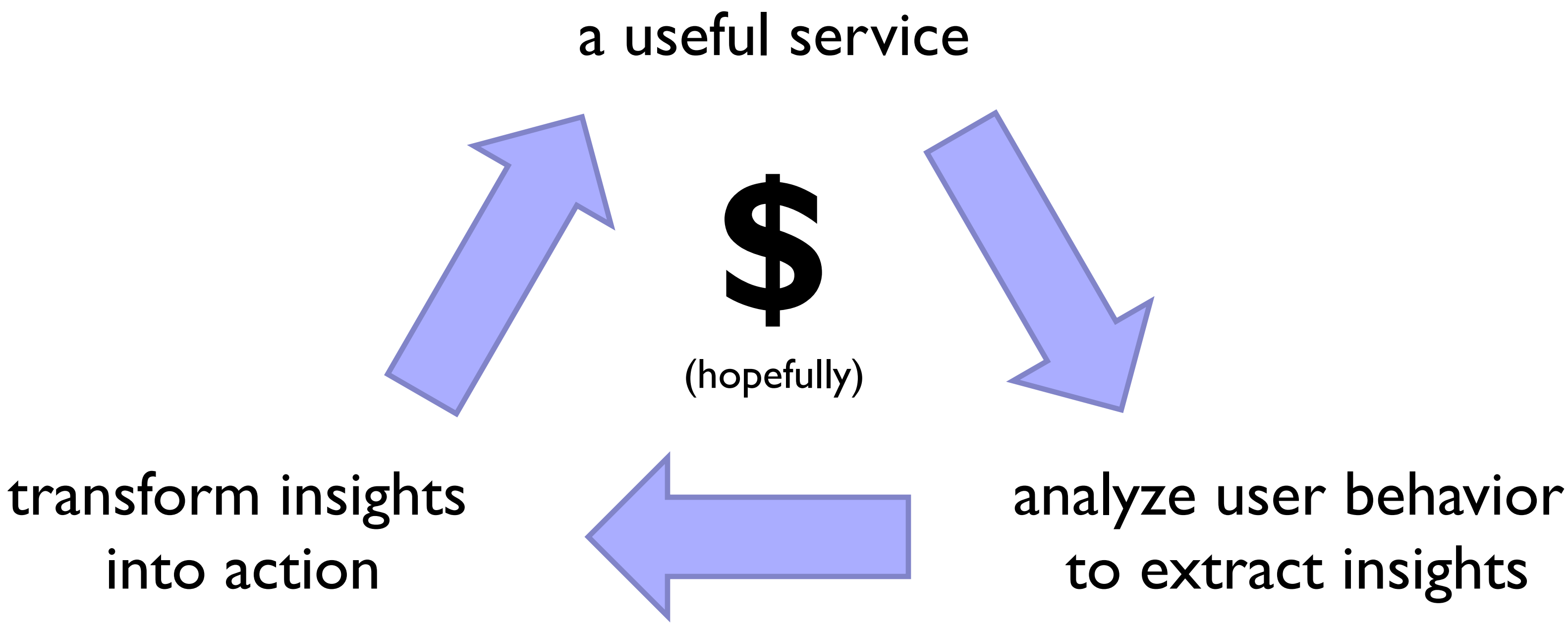
Source: Wikiedia (Shinjuku, Tokyo)

Business Intelligence

An organization should retain data that result from carrying out its mission and exploit those data to generate insights that benefit the organization, for example, market analysis, strategic planning, decision making, etc.

Duh!?

Virtuous Product Cycle



Google. Facebook. Twitter. Amazon. Uber.

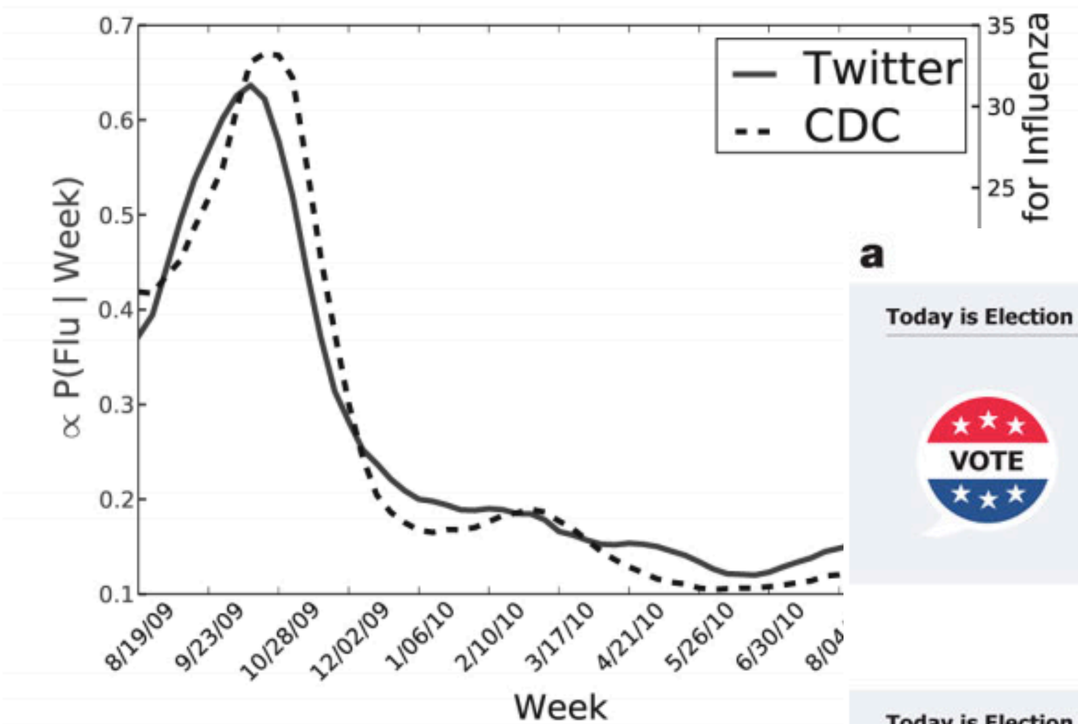
data products **data science**

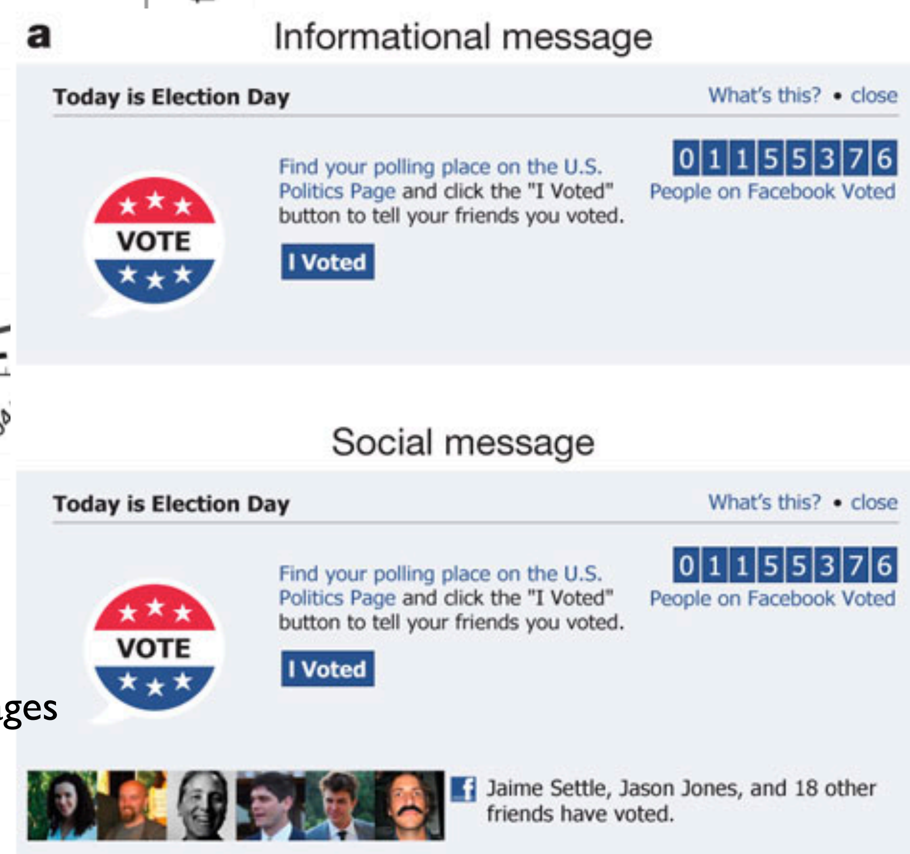
Society

- Humans as social sensors
- Computational social science

Source: Wikiedia (Guardian)

Predicting *X* with Twitter





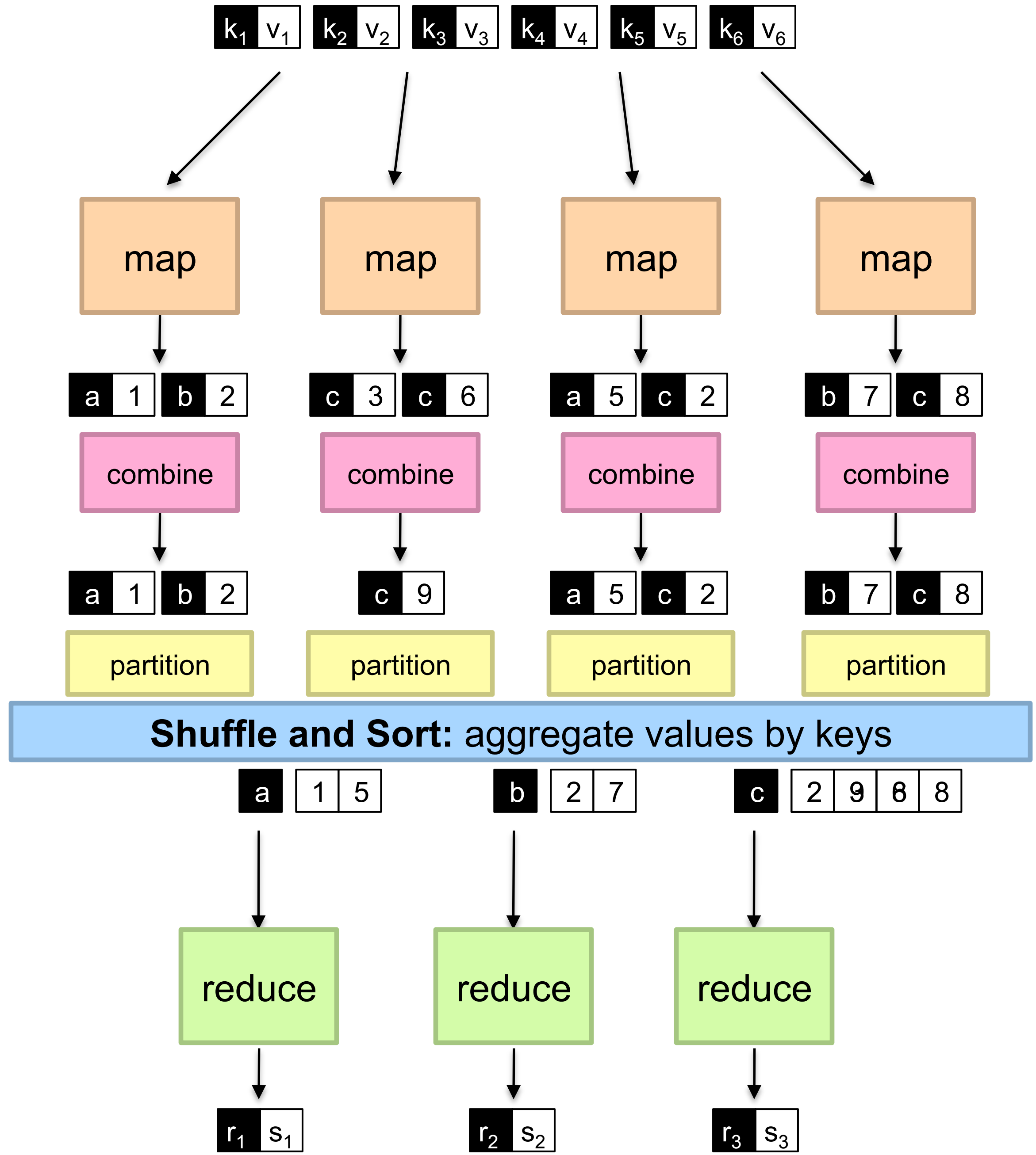
2010 US Midterm Elections:
60m users shown "I Voted" Messages

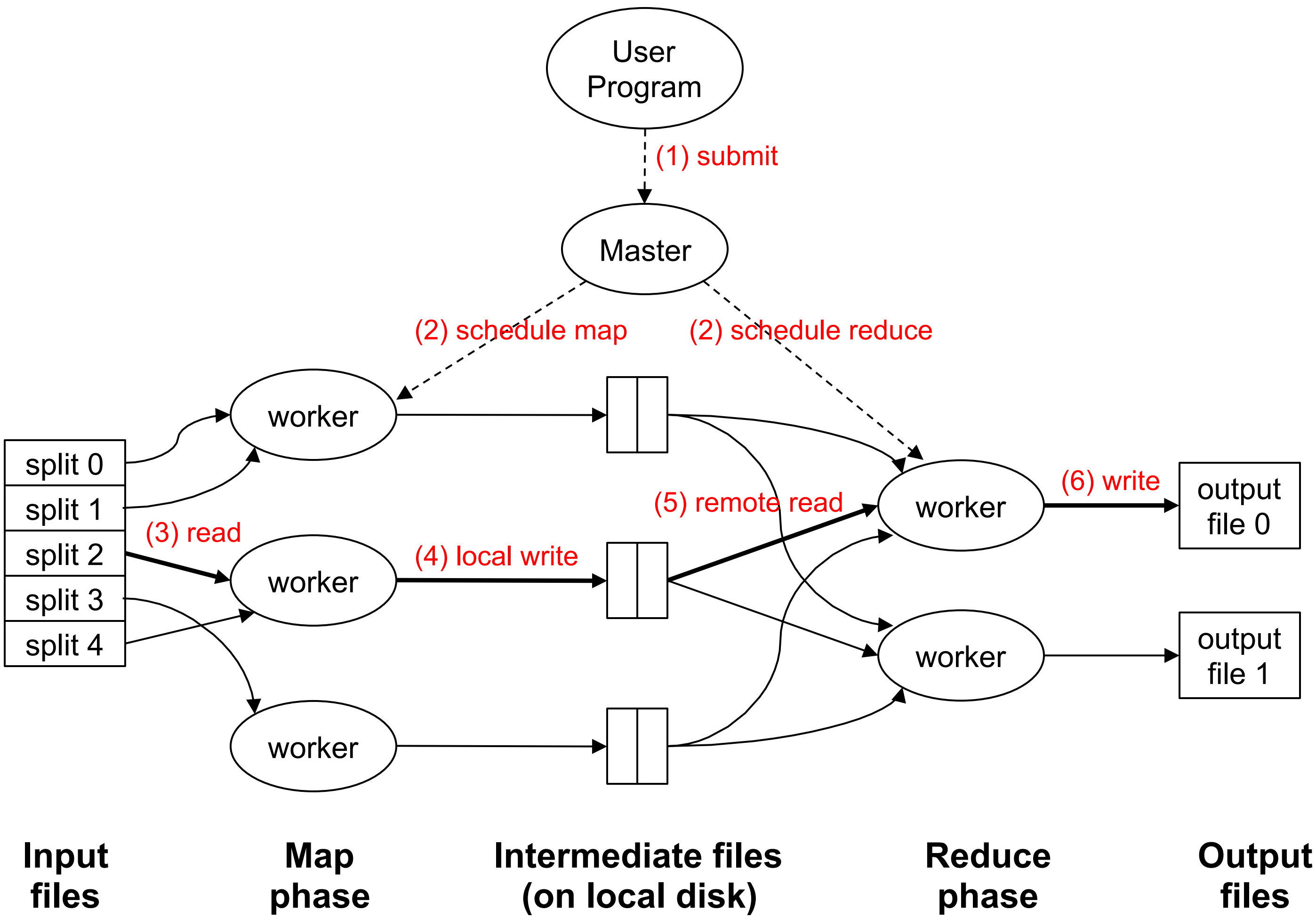
Summary: increased turnout by
60k directly and 280k indirectly

Political Mobilization on Facebook

(Paul and Dredze, ICWSM 2011; Bond et al., Nature 2011)


Tackling Big Data
Source: Google


k1 v1
k2 v2
k3 v3
k4 v4
k5 v5
k6 v6
map
map
map
map
a 1 b 2
c 3 c 6
a 5 c 2
b 7 c 8
combine
combine
combine
combine
a 1 b 2
c 9
a 5 c 2
b 7 c 8
partition
partition
partition
partition
Shuffle and Sort: aggregate values by keys
a 1 5
b 2 7
c 2 9 8 8
reduce
reduce
reduce
r1 s1
r2 s2
r3 s3



Adapted from (Dean and Ghemawat, OSDI 2004)

The datacenter *is* the computer

- It's all about the right level of abstraction
  - Moving beyond the von Neumann architecture
  - What's the "instruction set" of the datacenter computer?
- Hide system-level details from the developers
  - No more race conditions, lock contention, etc.
  - No need to explicitly worry about reliability, fault tolerance, etc.
- Separating the *what* from the *how*
  - Developer specifies the computation that needs to be performed
  - Execution framework ("runtime") handles actual execution


The datacenter *is* the computer!
Source: Google

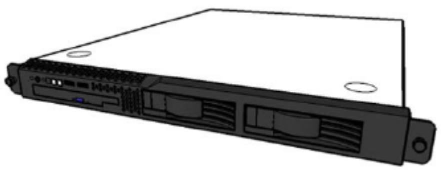
Source: Wikipedia (The Dalles, Oregon)

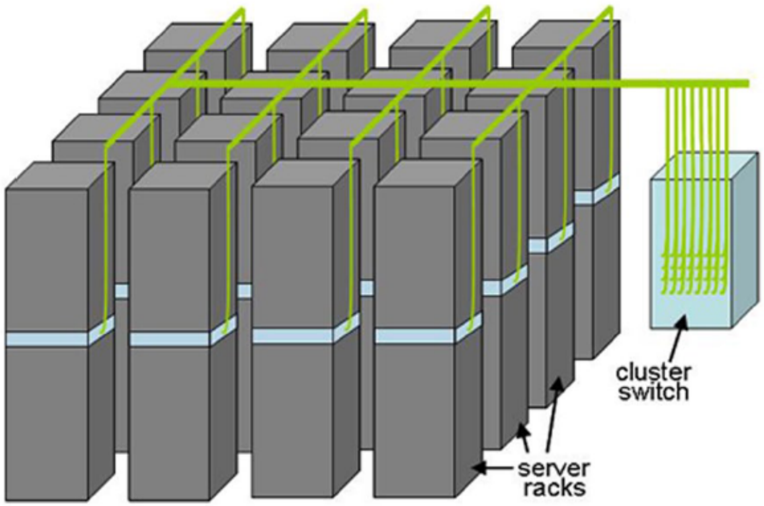

Source: Google

Source: Google

Source: Bonneville Power Administration

Building Blocks





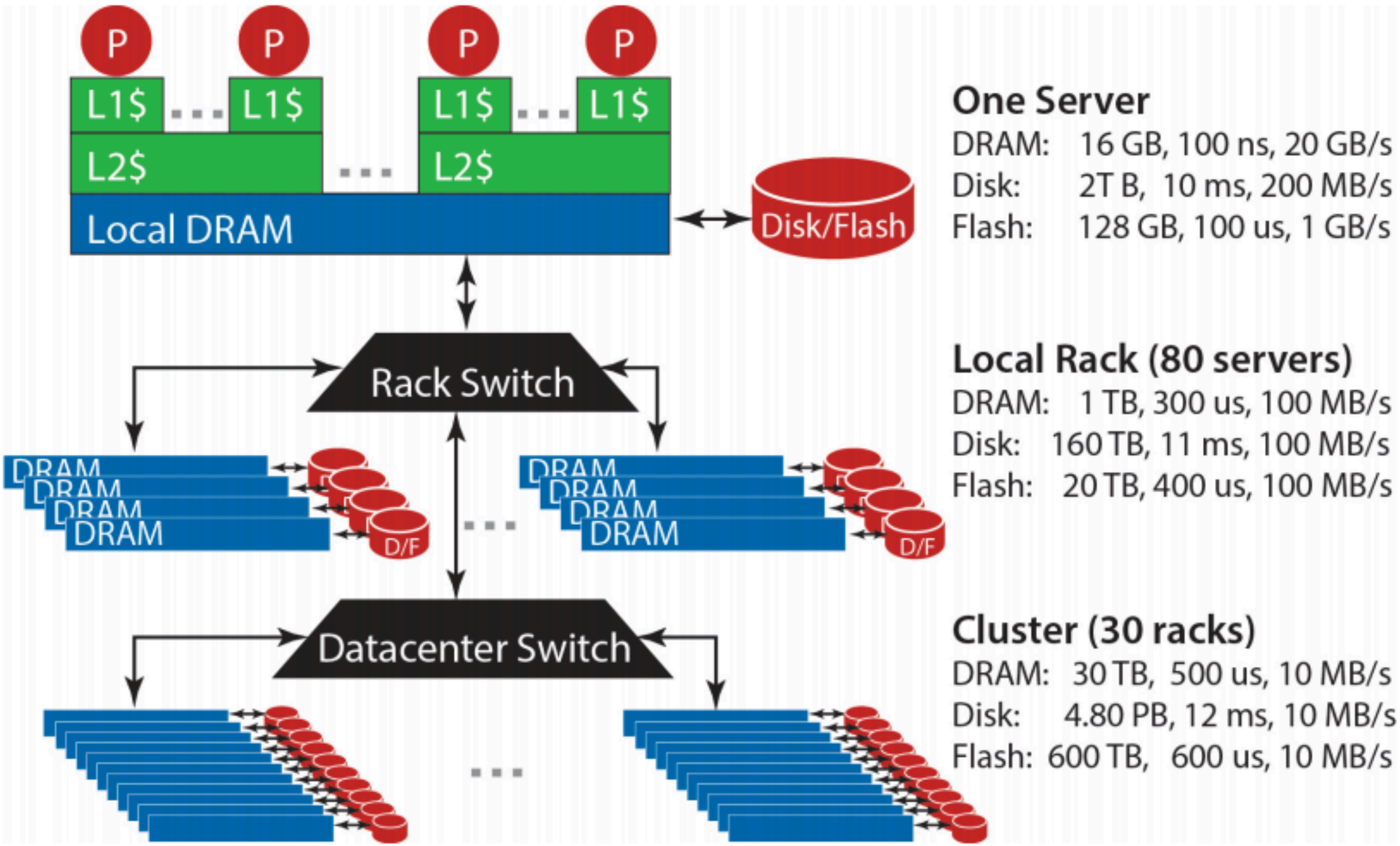
Source: Barroso and Urs Hölzle (2009)


Source: Google

Source: Google

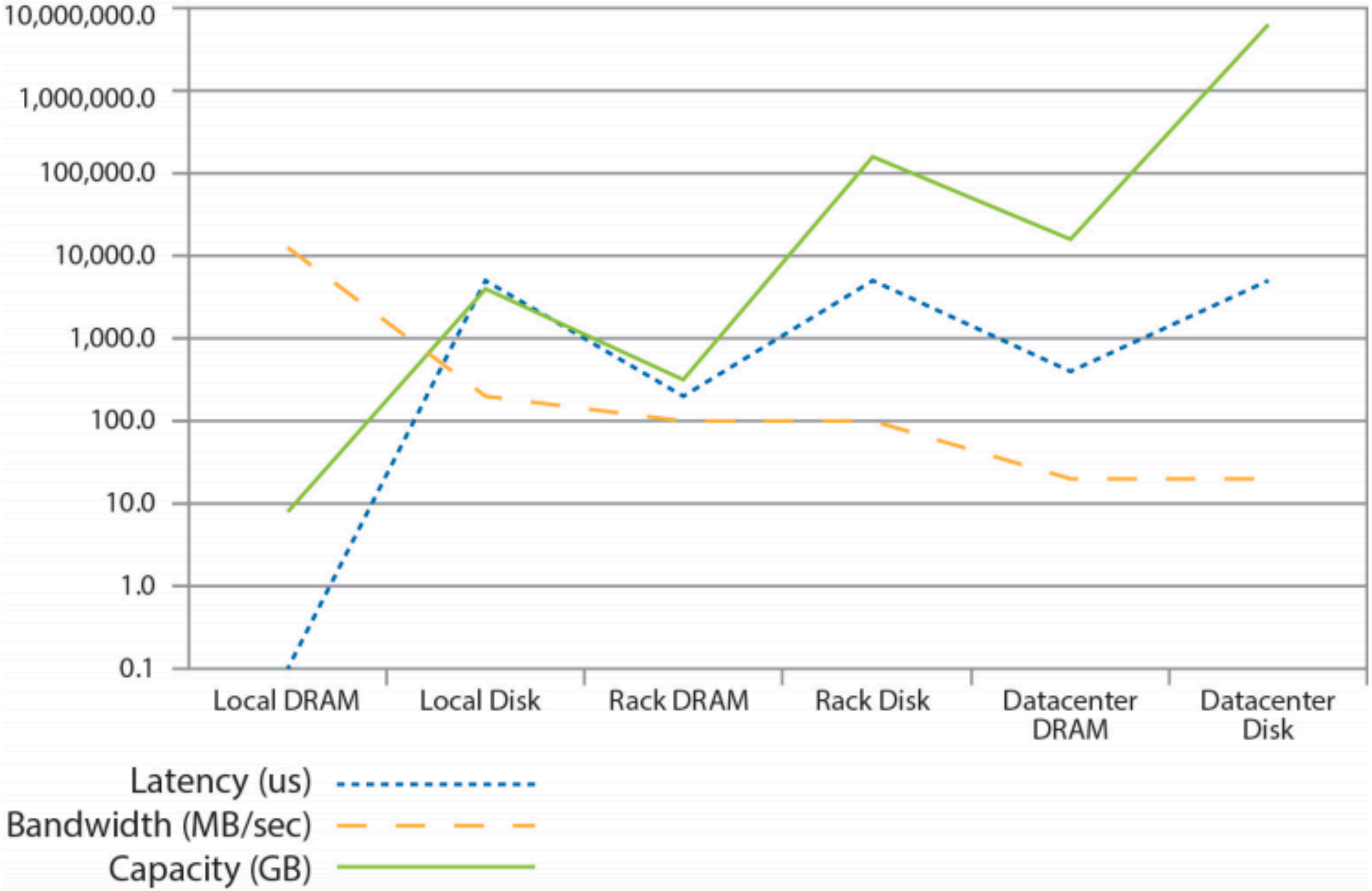

Source: Facebook

Storage Hierarchy



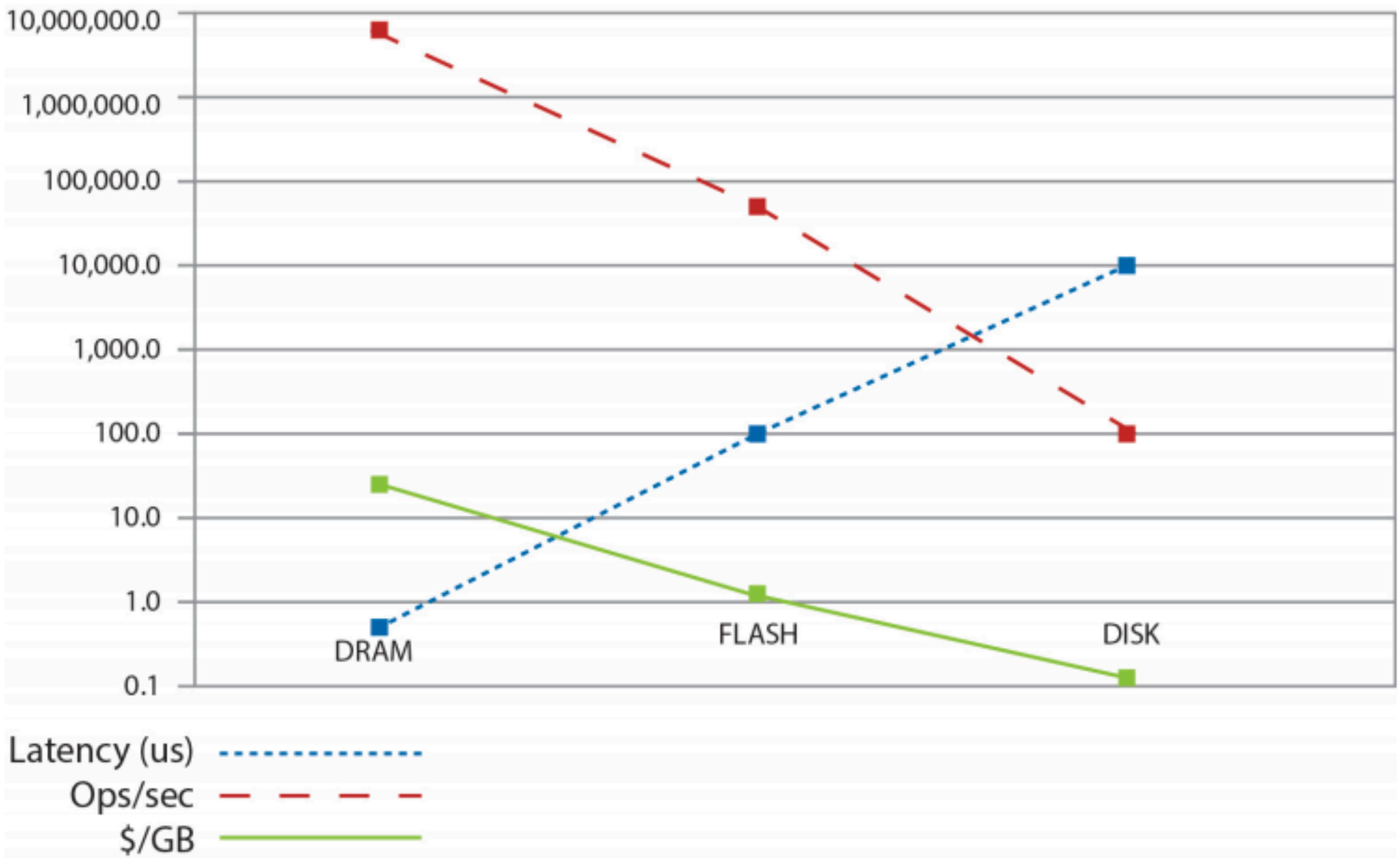
Source: Barroso and Urs Hölzle (2013)

Storage Hierarchy



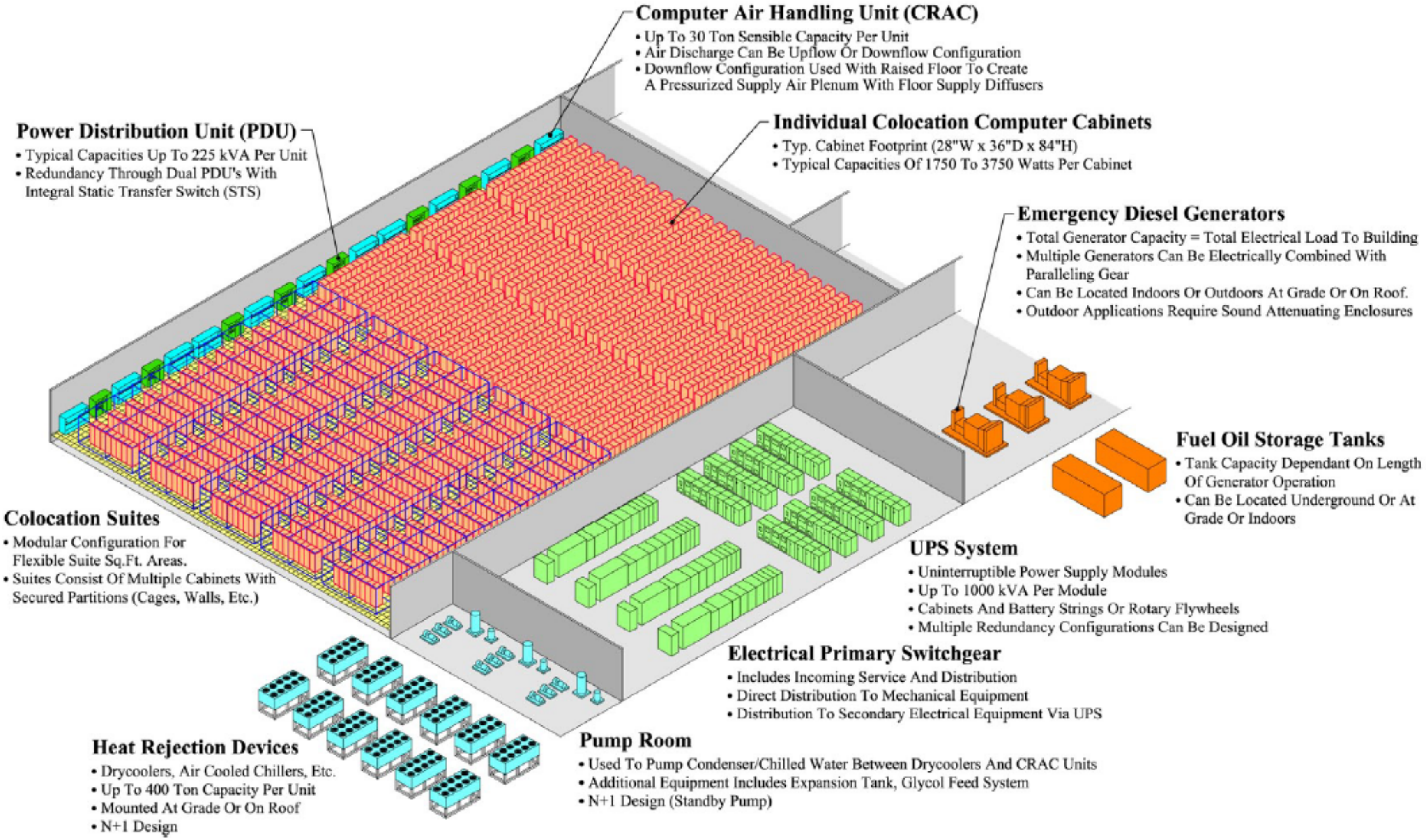
Source: Barroso and Urs Hölzle (2013)

Storage Hierarchy



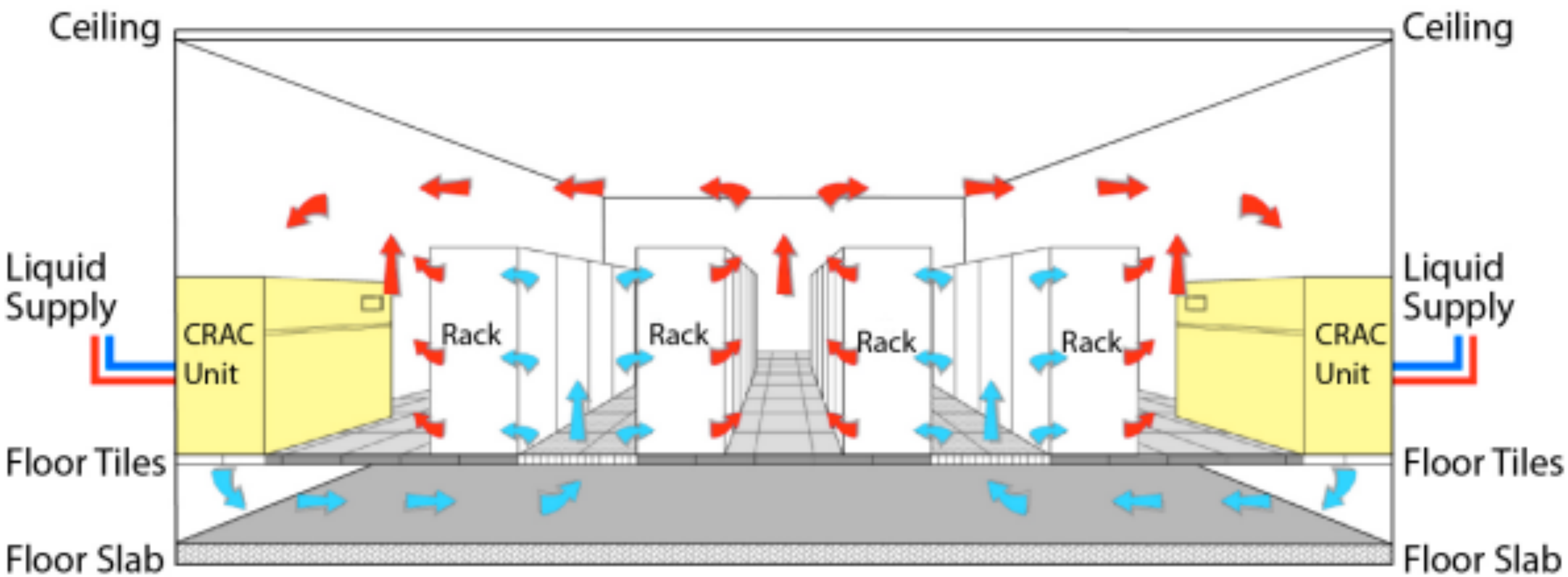
Source: Barroso and Urs Hölzle (2013)

Anatomy of a Datacenter



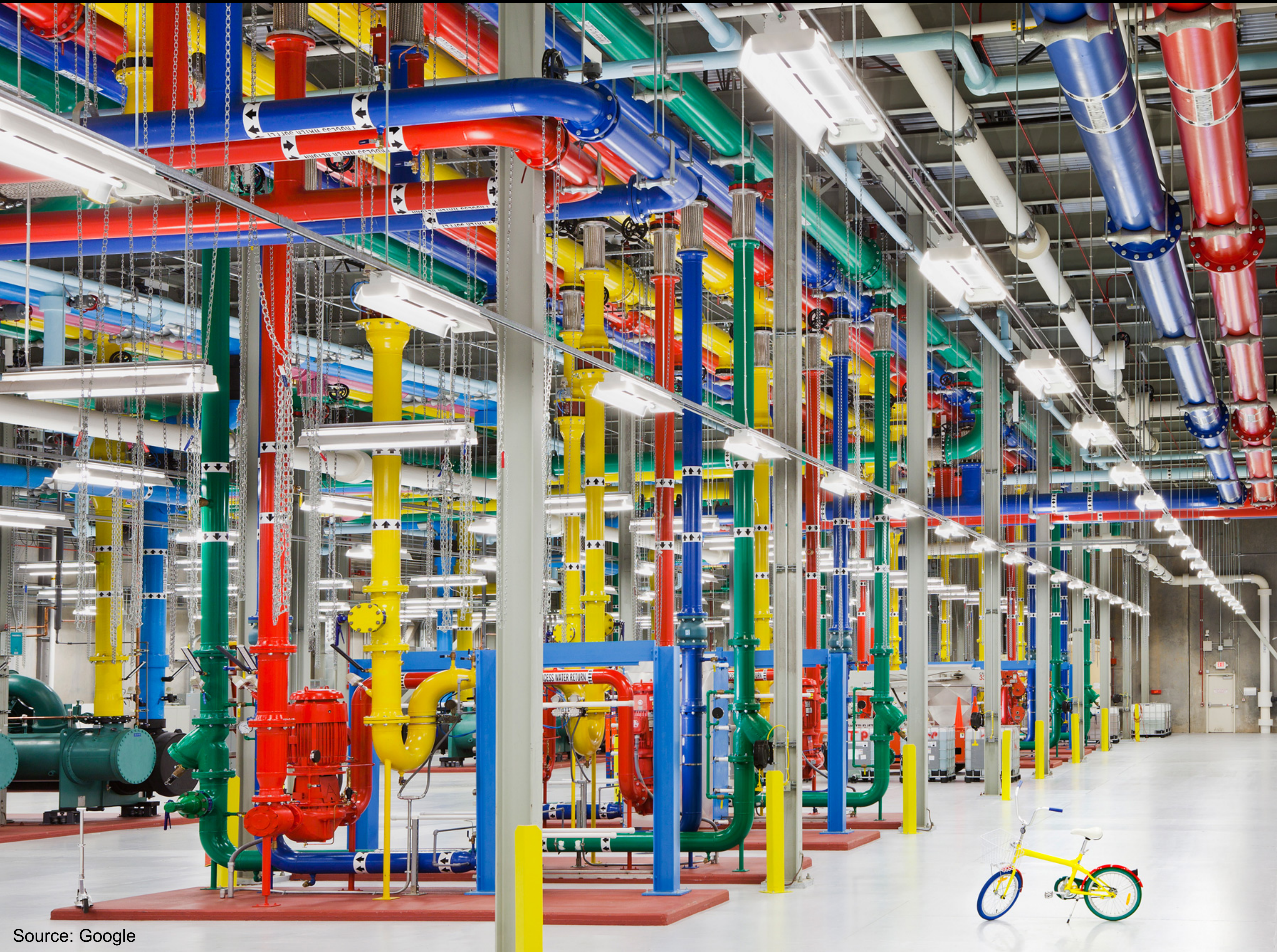
Source: Barroso and Urs Hölzle (2013)

Anatomy of a Datacenter

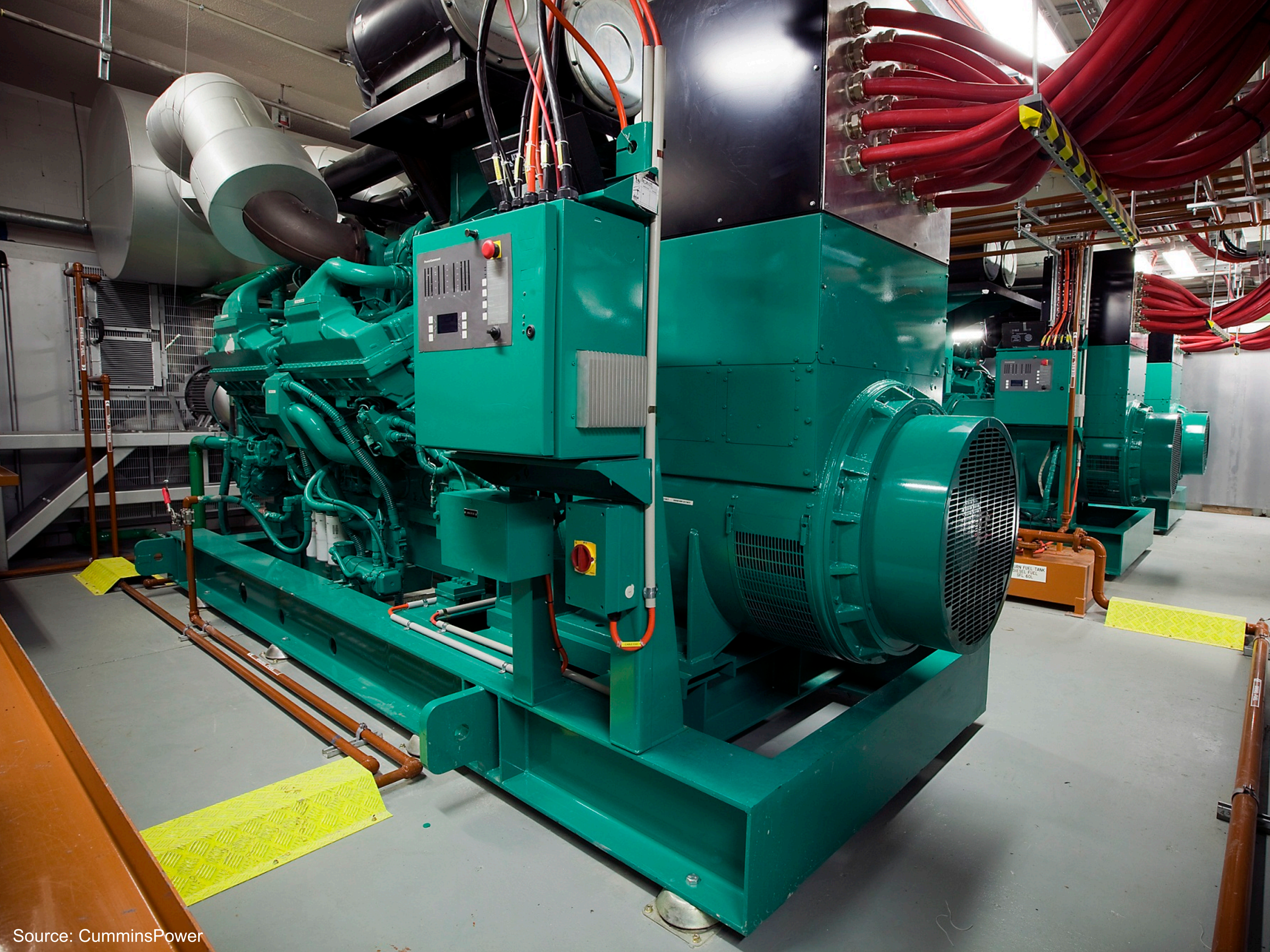


Source: Barroso and Urs Hölzle (2013)


Source: Google



Source: Google



Source: CumminsPower

Source: Google

Source: Google

The datacenter *is* the computer

- It's all about the right level of abstraction
  - Moving beyond the von Neumann architecture
  - What's the "instruction set" of the datacenter computer?
- Hide system-level details from the developers
  - No more race conditions, lock contention, etc.
  - No need to explicitly worry about reliability, fault tolerance, etc.
- Separating the *what* from the *how*
  - Developer specifies the computation that needs to be performed
  - Execution framework ("runtime") handles actual execution

"Big Ideas"

- Scale "out", not "up"
  - Limits of SMP and large shared-memory machines
- Move processing to the data
  - Cluster have limited bandwidth
- Process data sequentially, avoid random access
  - Seeks are expensive, disk throughput is reasonable
- Seamless scalability
  - From the mythical man-month to the tradable machine-hour

Scaling "up" vs. "out"

- No single machine is large enough
  - Smaller cluster of large SMP machines vs. larger cluster of commodity machines (e.g., 16 128-core machines vs. 128 16-core machines)
- Nodes need to talk to each other!
  - Intra-node latencies: ~100 ns
  - Inter-node latencies: ~100 μs
- Let's model communication overhead…

Source: analysis on this an subsequent slides from Barroso and Urs Hölzle (2009)

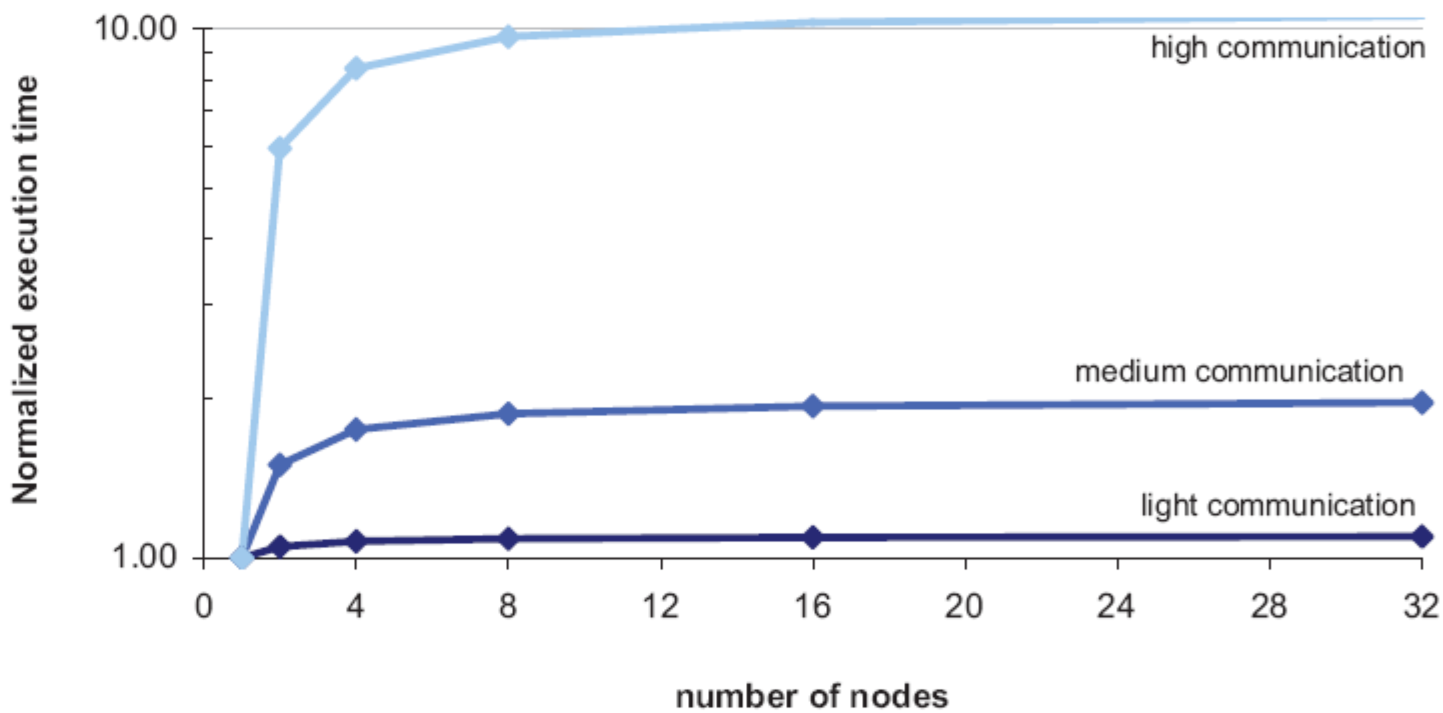
Modeling Communication Costs

- Simple execution cost model:
  - Total cost = cost of computation + cost to access global data
  - Fraction of local access inversely proportional to size of cluster
  - *n* nodes (ignore cores for now)

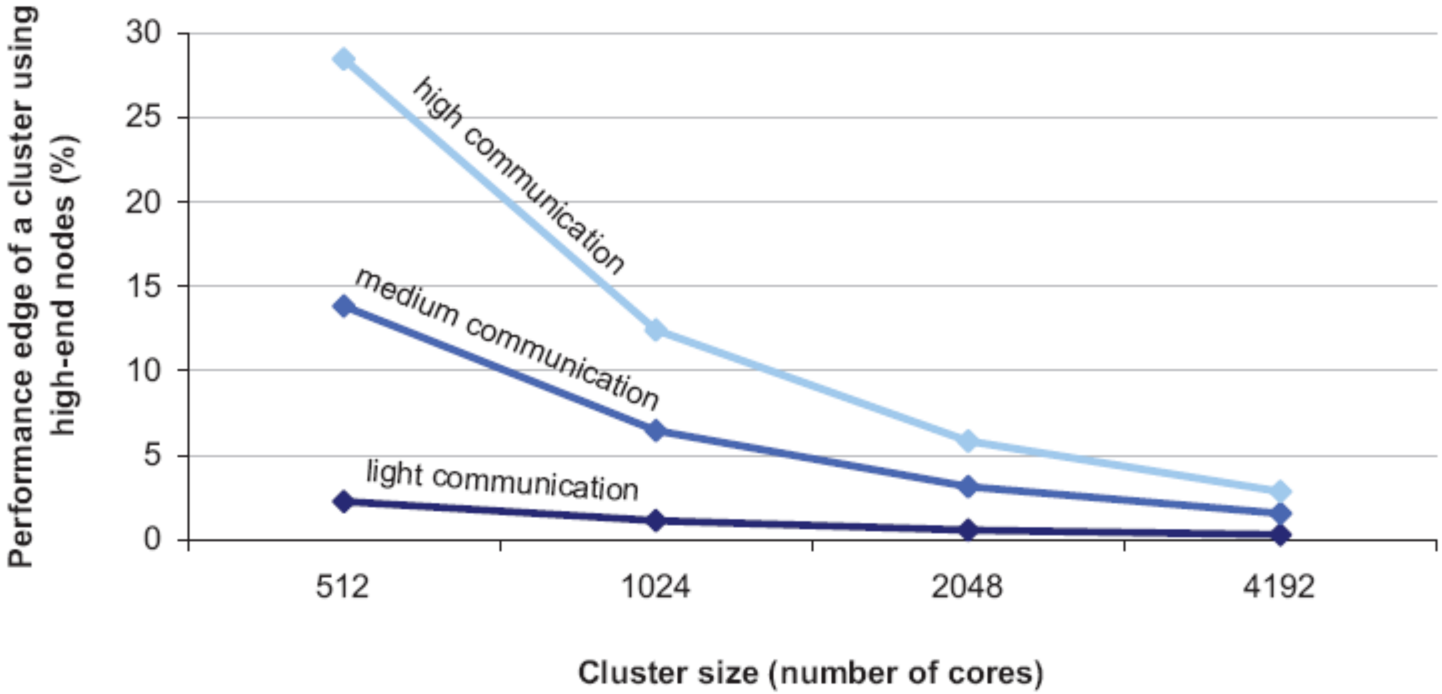
$$1 \text{ ms} + f \times [100 \text{ ns} \times (1/n) + 100\ \mu\text{s} \times (1 - 1/n)]$$

  - Light communication: $f = 1$
  - Medium communication: $f = 10$
  - Heavy communication: $f = 100$
- What are the costs in parallelization?

Cost of Parallelization



Advantages of scaling “up”



So why not?

Why does commodity beat exotic?

Counterpoint: Scaling up?

- No single machine is large enough
  - Smaller cluster of large SMP machines vs. larger cluster of commodity machines (e.g., 16 128-core machines vs. 128 16-core machines)
- Is this really true? Modern "commodity" machine:
  - Four 18-core processors: 72 cores total
  - 3TB RAM

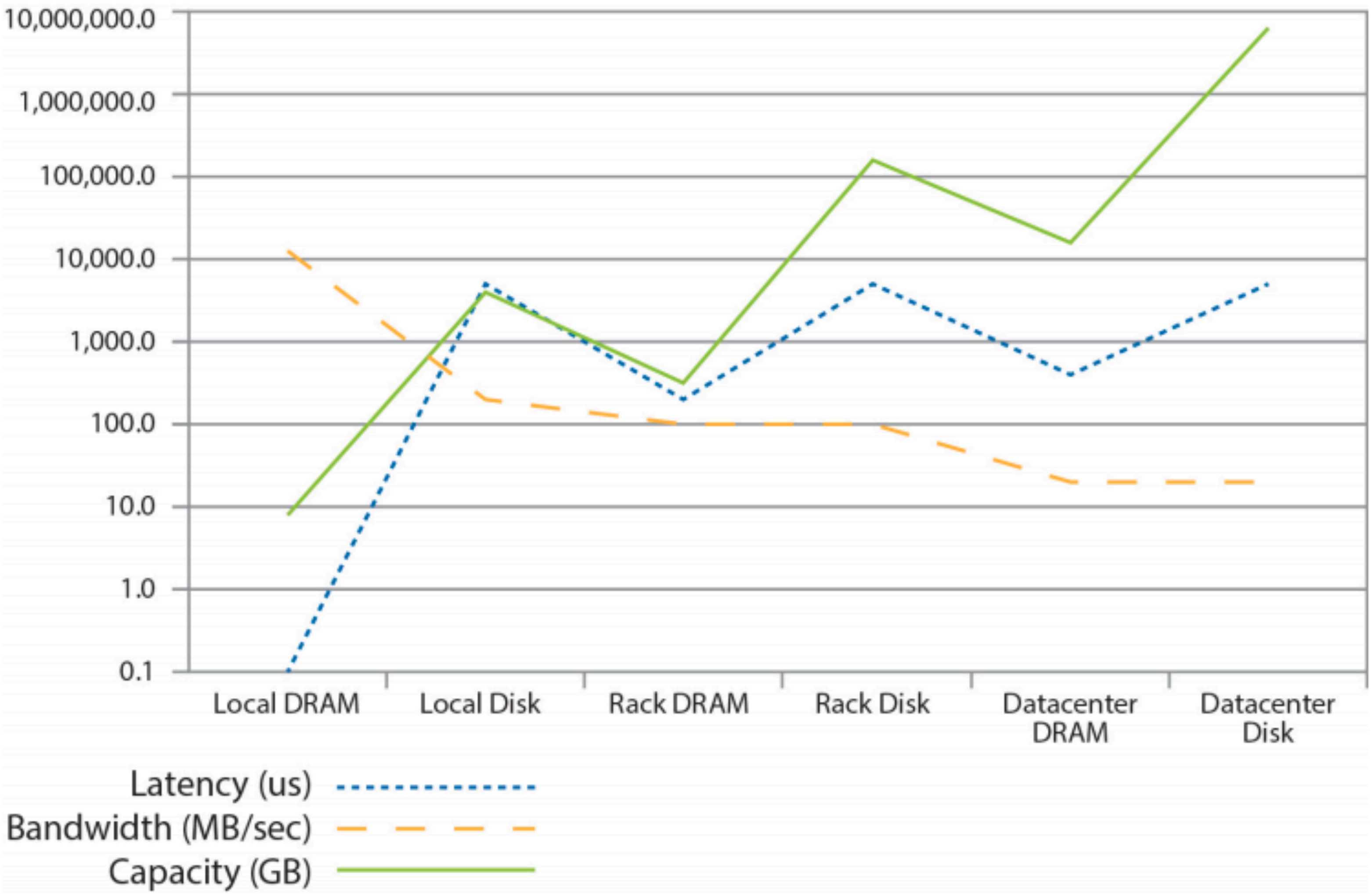
Who really has big data problems?

Numbers Everyone Should Know

According to Jeff Dean

L1 cache reference	0.5 ns
Branch mispredict	5 ns
L2 cache reference	7 ns
Mutex lock/unlock	100 ns
Main memory reference	100 ns
Compress 1K bytes with Zippy	10,000 ns
Send 2K bytes over 1 Gbps network	20,000 ns
Read 1 MB sequentially from memory	250,000 ns
Round trip within same datacenter	500,000 ns
Disk seek	10,000,000 ns
Read 1 MB sequentially from network	10,000,000 ns
Read 1 MB sequentially from disk	30,000,000 ns
Send packet CA->Netherlands->CA	150,000,000 ns

Moving Data Around



Source: Barroso and Urs Hölzle (2013)

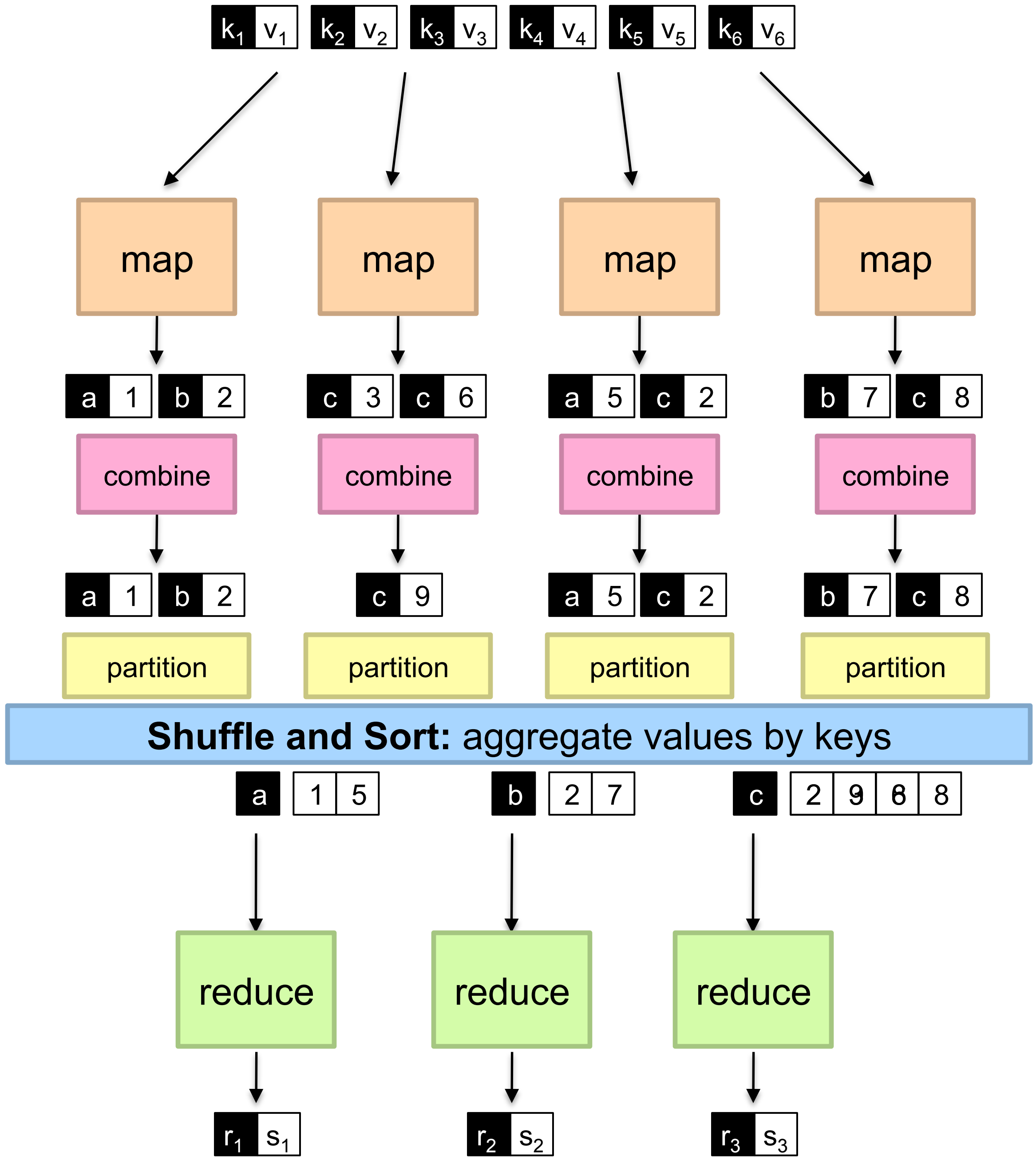
Seeks vs. Scans

- Consider a 1 TB database with 100 byte records
  - We want to update 1 percent of the records
- Scenario 1: random access
  - Each update takes ~30 ms (seek, read, write)
  - 10^8 updates = ~35 days
- Scenario 2: rewrite all records
  - Assume 100 MB/s throughput
  - Time = 5.6 hours(!)
- Lesson: avoid random seeks!

Source: Ted Dunning, on Hadoop mailing list

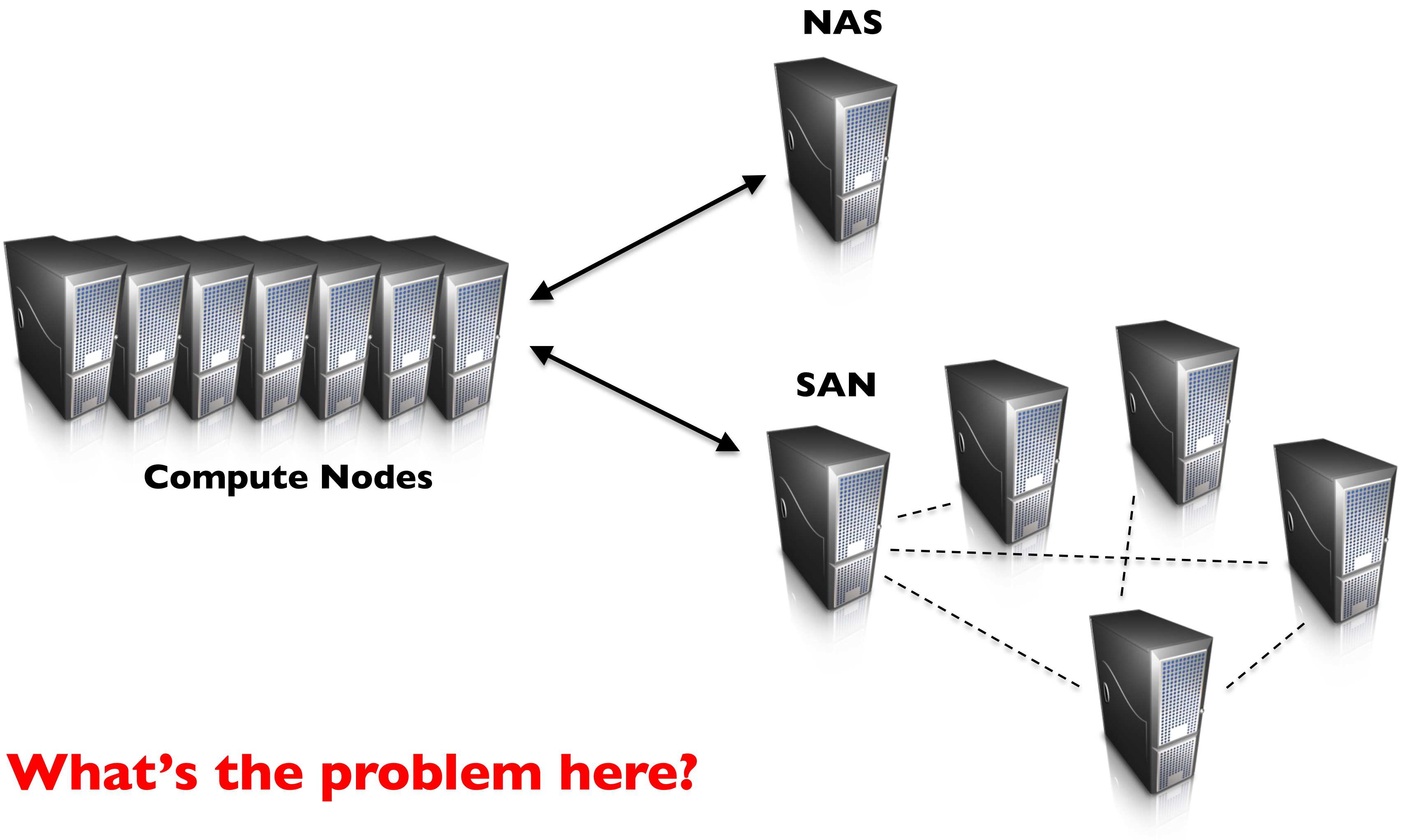
Justifying the “Big Ideas”

- Scale “out”, not “up”
  - Limits of SMP and large shared-memory machines
- Move processing to the data
  - Cluster have limited bandwidth
- Process data sequentially, avoid random access
  - Seeks are expensive, disk throughput is reasonable
- Seamless scalability
  - From the mythical man-month to the tradable machine-hour



k1 v1
k2 v2
k3 v3
k4 v4
k5 v5
k6 v6
map
map
map
map
a 1 b 2
c 3 c 6
a 5 c 2
b 7 c 8
combine
combine
combine
combine
a 1 b 2
c 9
a 5 c 2
b 7 c 8
partition
partition
partition
partition
Shuffle and Sort: aggregate values by keys
a 1 5
b 2 7
c 2 9 8 8
reduce
reduce
reduce
r1 s1
r2 s2
r3 s3

How do we get data to the workers?



What's the problem here?

Distributed File System

- Don't move data to workers… move workers to the data!
  - Store data on the local disks of nodes in the cluster
  - Start up the workers on the node that has the data local
- Why?
  - (Perhaps) not enough RAM to hold all the data in memory
  - Disk access is slow, but disk throughput is reasonable
- A distributed file system is the answer
  - GFS (Google File System) for Google's MapReduce
  - HDFS (Hadoop Distributed File System) for Hadoop

GFS: Assumptions

- Commodity hardware over "exotic" hardware
  - Scale "out", not "up"
- High component failure rates
  - Inexpensive commodity components fail all the time
- "Modest" number of huge files
  - Multi-gigabyte files are common, if not encouraged
- Files are write-once, mostly appended to
  - Perhaps concurrently
- Large streaming reads over random access
  - High sustained throughput over low latency

GFS slides adapted from material by (Ghemawat et al., SOSP 2003)

GFS: Design Decisions

- Files stored as chunks
  - Fixed size (64MB)
- Reliability through replication
  - Each chunk replicated across 3+ chunkservers
- Single master to coordinate access, keep metadata
  - Simple centralized management
- No data caching
  - Little benefit due to large datasets, streaming reads
- Simplify the API
  - Push some of the issues onto the client (e.g., data layout)

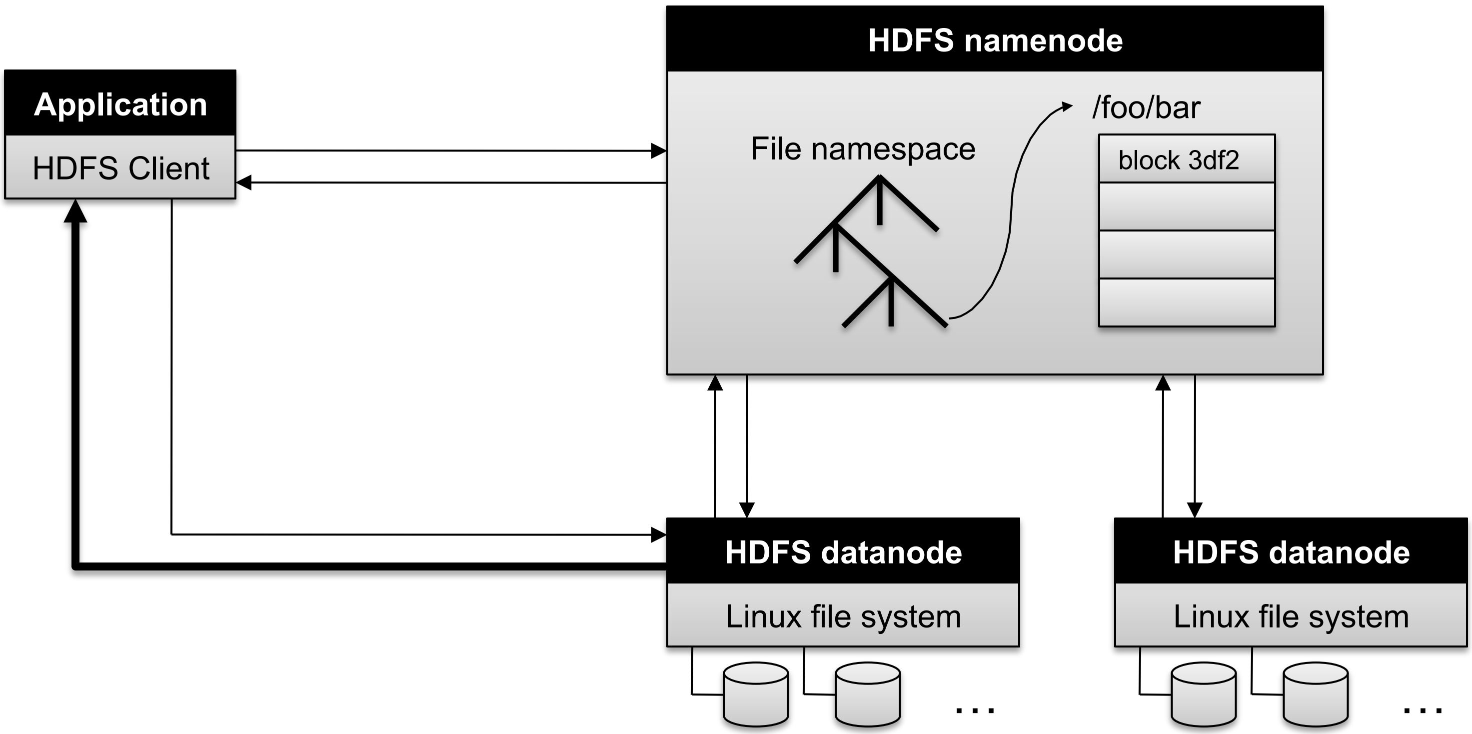
HDFS = GFS clone (same basic ideas)

From GFS to HDFS

- Terminology differences:
  - GFS master = Hadoop namenode
  - GFS chunkservers = Hadoop datanodes
- Differences:
  - Different consistency model for file appends
  - Implementation
  - Performance

For the most part, we'll use Hadoop terminology…

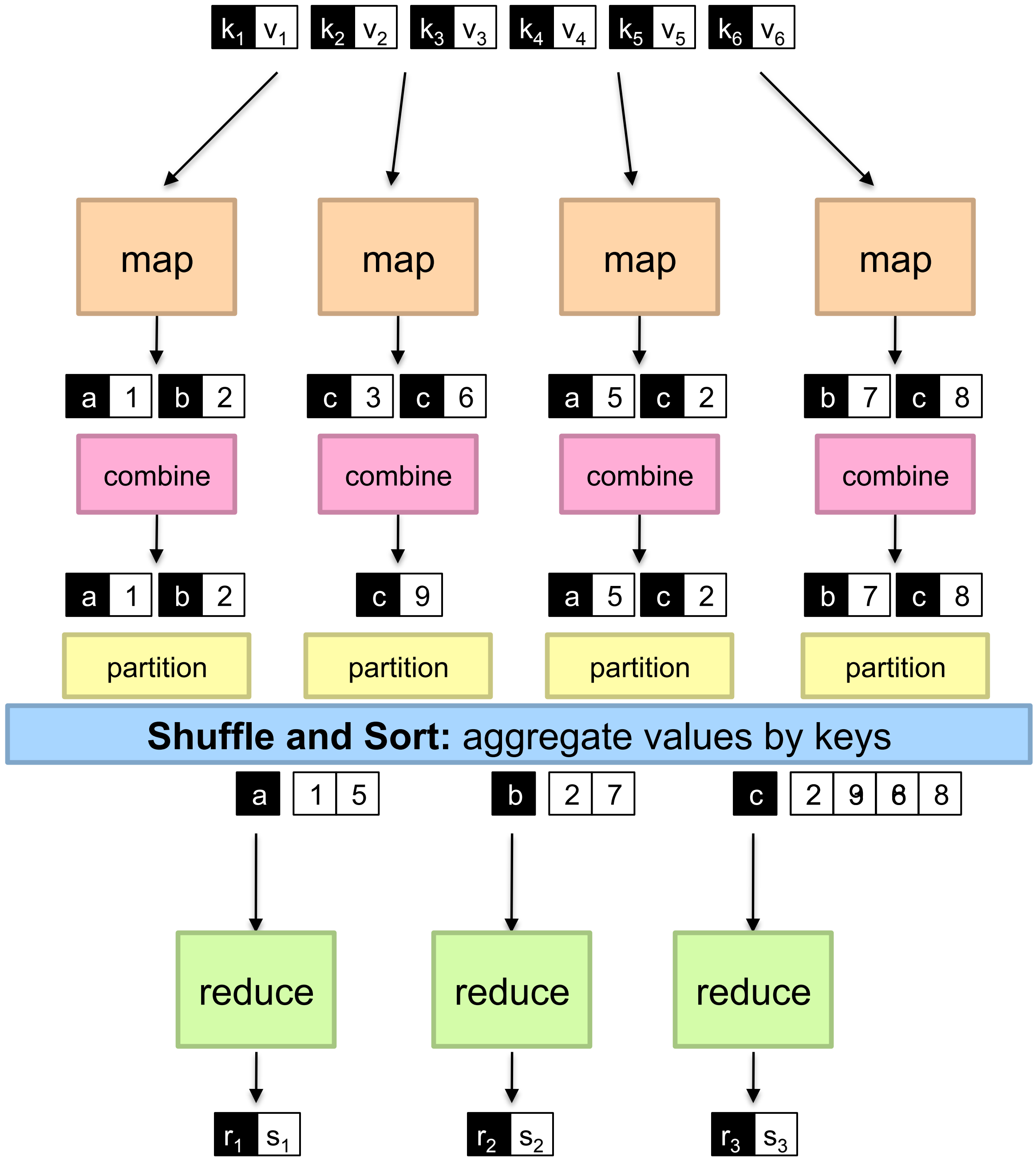
HDFS Architecture



Adapted from (Ghemawat et al., SOSP 2003)

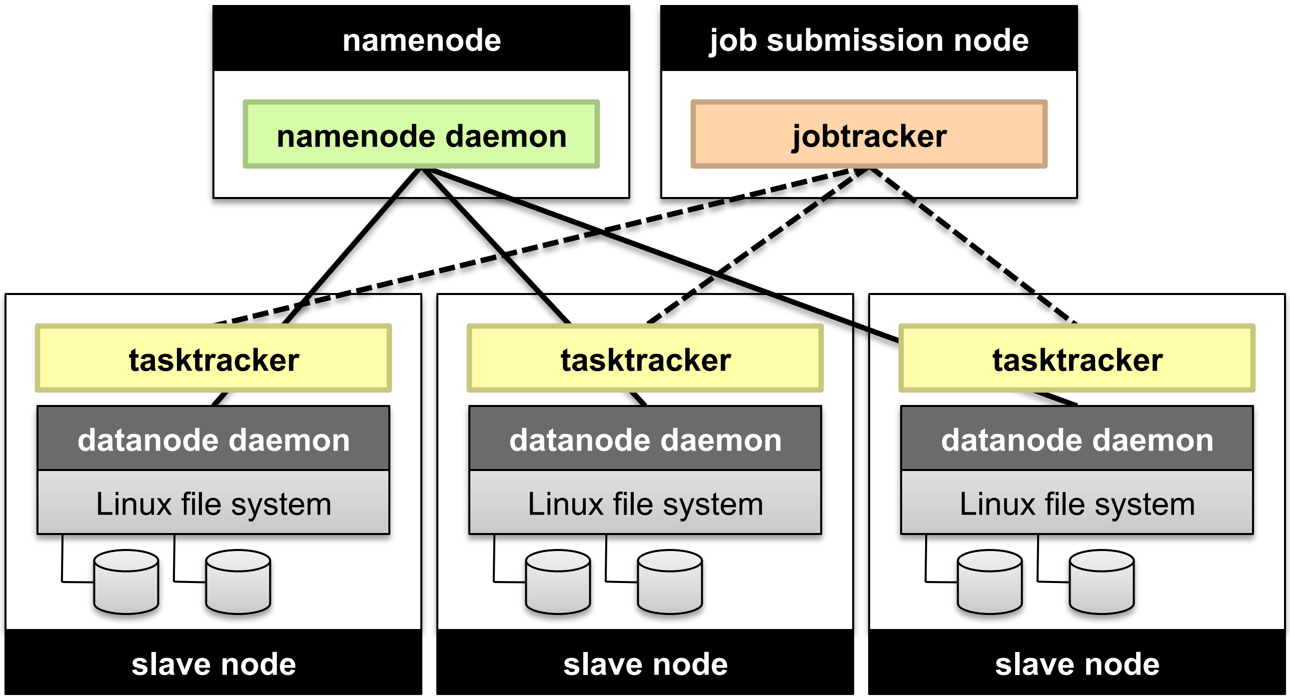
Namenode Responsibilities

- Managing the file system namespace:
  - Holds file/directory structure, metadata, file-to-block mapping, access permissions, etc.
- Coordinating file operations:
  - Directs clients to datanodes for reads and writes
  - No data is moved through the namenode
- Maintaining overall health:
  - Periodic communication with the datanodes
  - Block re-replication and rebalancing
  - Garbage collection



k_1 v_1 k_2 v_2 k_3 v_3 k_4 v_4 k_5 v_5 k_6 v_6
map
map
map
map
a 1 b 2
c 3 c 6
a 5 c 2
b 7 c 8
combine
combine
combine
combine
a 1 b 2
c 9
a 5 c 2
b 7 c 8
partition
partition
partition
partition
Shuffle and Sort: aggregate values by keys
a 1 5
b 2 7
c 2
8
reduce
reduce
reduce
r_1 s_1
r_2 s_2
r_3 s_3

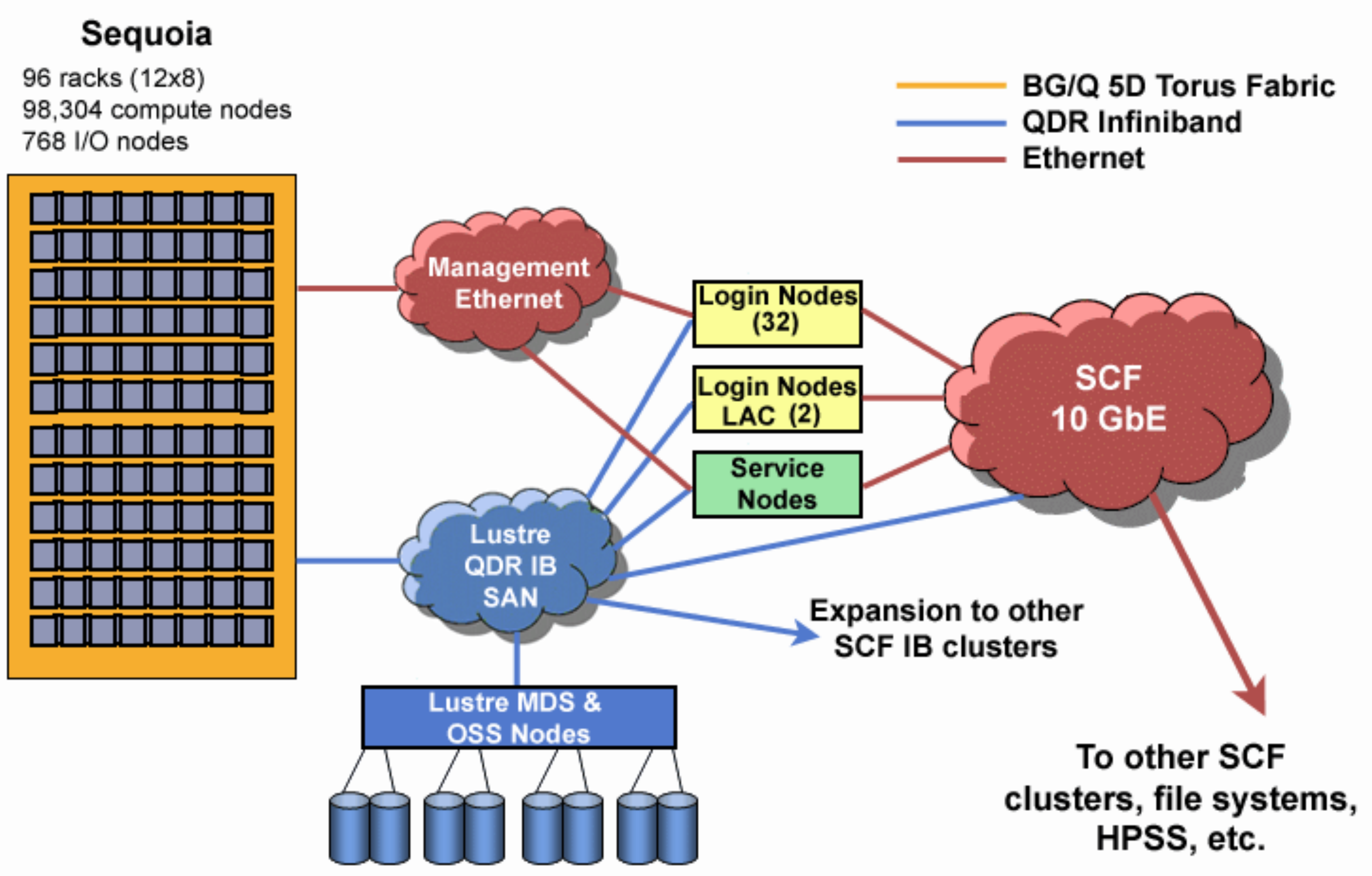
Putting everything together…



(Not Quite… We'll come back to YARN later)

Sequoia

16.32 PFLOPS
98,304 nodes with 1,572,864 million cores
1.6 petabytes of memory
7.9 MWatts total power


Sequoia
96 racks (12x8)
98,304 compute nodes
768 I/O nodes
BG/Q 5D Torus Fabric
QDR Infiniband
Ethernet
Management Ethernet
Login Nodes (32)
Login Nodes LAC (2)
Service Nodes
SCF 10 GbE
Lustre QDR IB SAN
Expansion to other SCF IB clusters
Lustre MDS & OSS Nodes
To other SCF clusters, file systems, HPSS, etc.

Source: LLNL

Aside: Cloud Computing

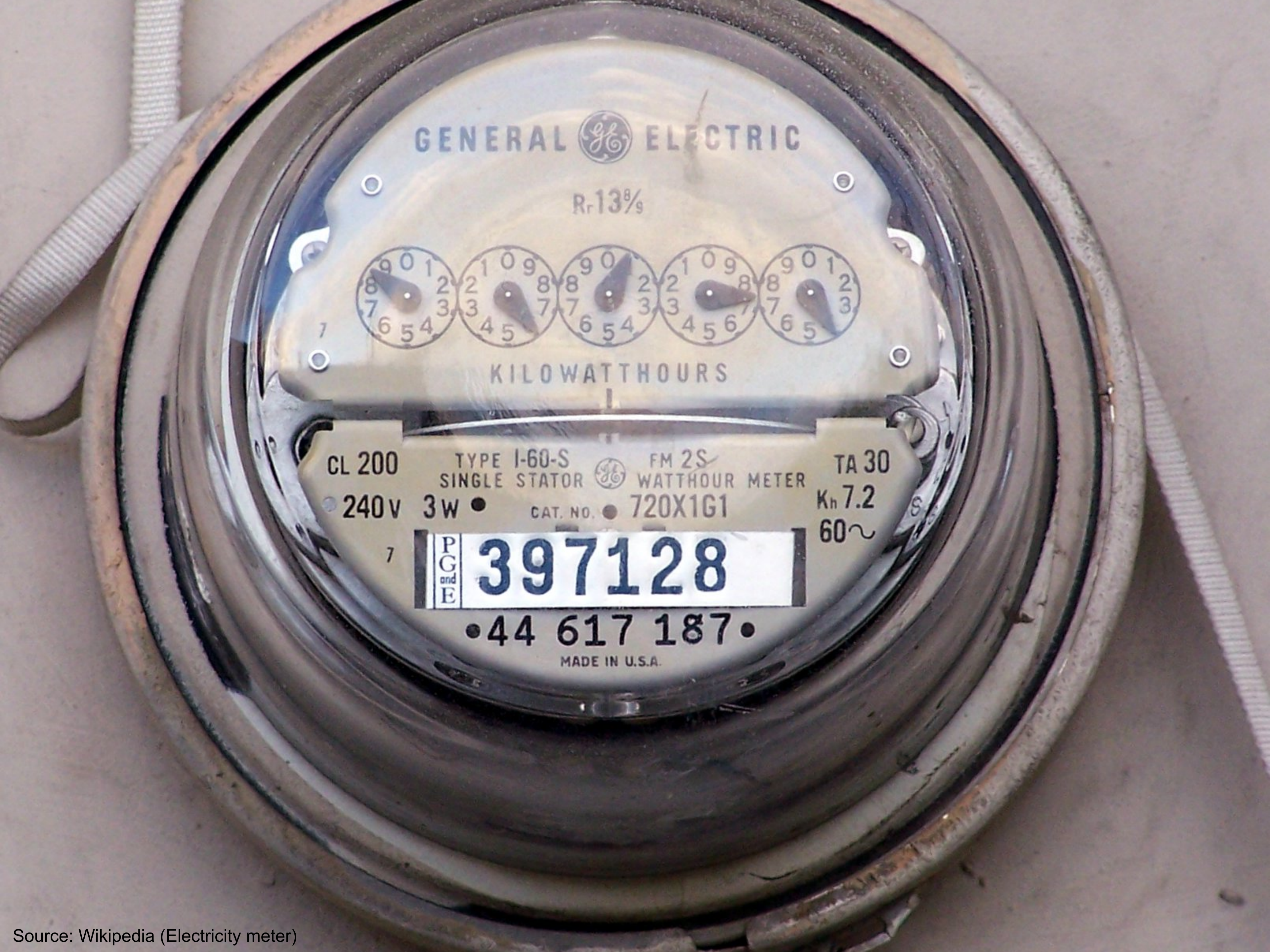
Source: Wikipedia (Clouds)

The best thing since sliced bread?

- Before clouds…
  - Grids
  - Connection machine
  - Vector supercomputers
  - …
- Cloud computing means many different things:
  - Big data
  - Rebranding of web 2.0
  - Utility computing
  - Everything as a service

Rebranding of web 2.0

- Rich, interactive web applications
  - Clouds refer to the servers that run them
  - AJAX as the de facto standard (for better or worse)
  - Examples: Facebook, YouTube, Gmail, …
- "The network is the computer": take two
  - User data is stored "in the clouds"
  - Rise of the netbook, smartphones, etc.
  - Browser *is* the OS

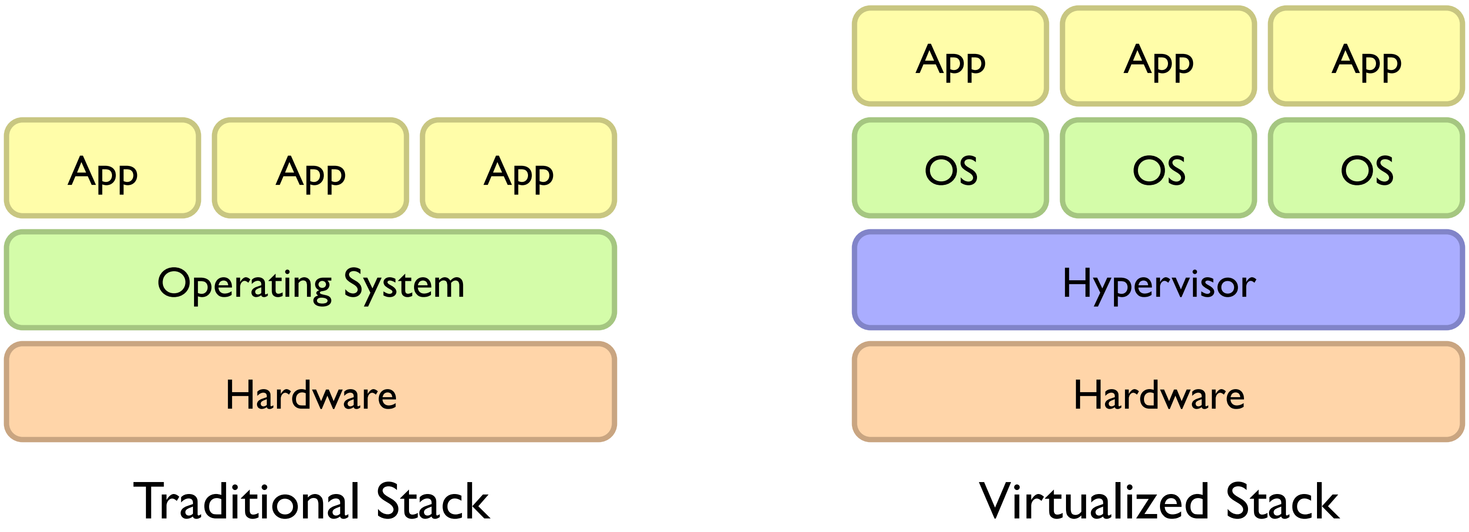


Source: Wikipedia (Electricity meter)

Utility Computing

- What?
  - Computing resources as a metered service ("pay as you go")
  - Ability to dynamically provision virtual machines
- Why?
  - Cost: capital vs. operating expenses
  - Scalability: "infinite" capacity
  - Elasticity: scale up or down on demand
- Does it make sense?
  - Benefits to cloud users
  - Business case for cloud providers

Enabling Technology: Virtualization

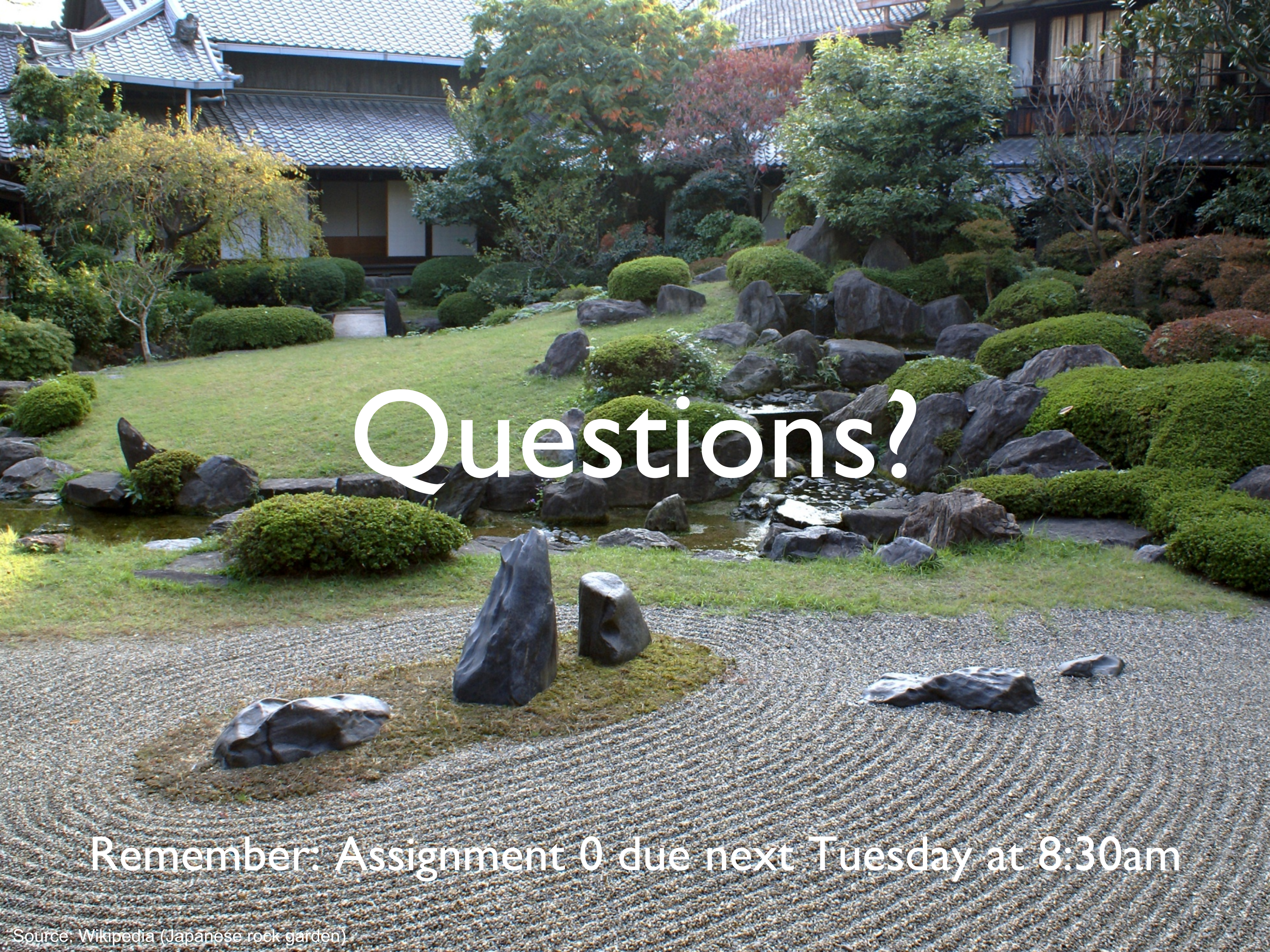


Everything as a Service

- Utility computing = Infrastructure as a Service (IaaS)
  - Why buy machines when you can rent cycles?
  - Examples: Amazon's EC2, Rackspace
- Platform as a Service (PaaS)
  - Give me nice API and take care of the maintenance, upgrades, …
  - Example: Google App Engine
- Software as a Service (SaaS)
  - Just run it for me!
  - Example: Gmail, Salesforce

Who cares?

- A source of problems…
  - Cloud-based services *generate* big data
  - Clouds make it easier to start companies that *generate* big data
- As well as a solution…
  - Ability to provision analytics clusters on-demand in the cloud
  - Commoditization and democratization of big data capabilities



Source: Wikipedia (Japanese rock garden)