UNIVERSITY OF WATERLOO

Big Data Infrastructure

CS 489/698 Big Data Infrastructure (Winter 2017)

Week 1: Introduction (2/2)

January 5, 2017

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

These slides are available at http://lintool.github.io/bigdata-2017w/

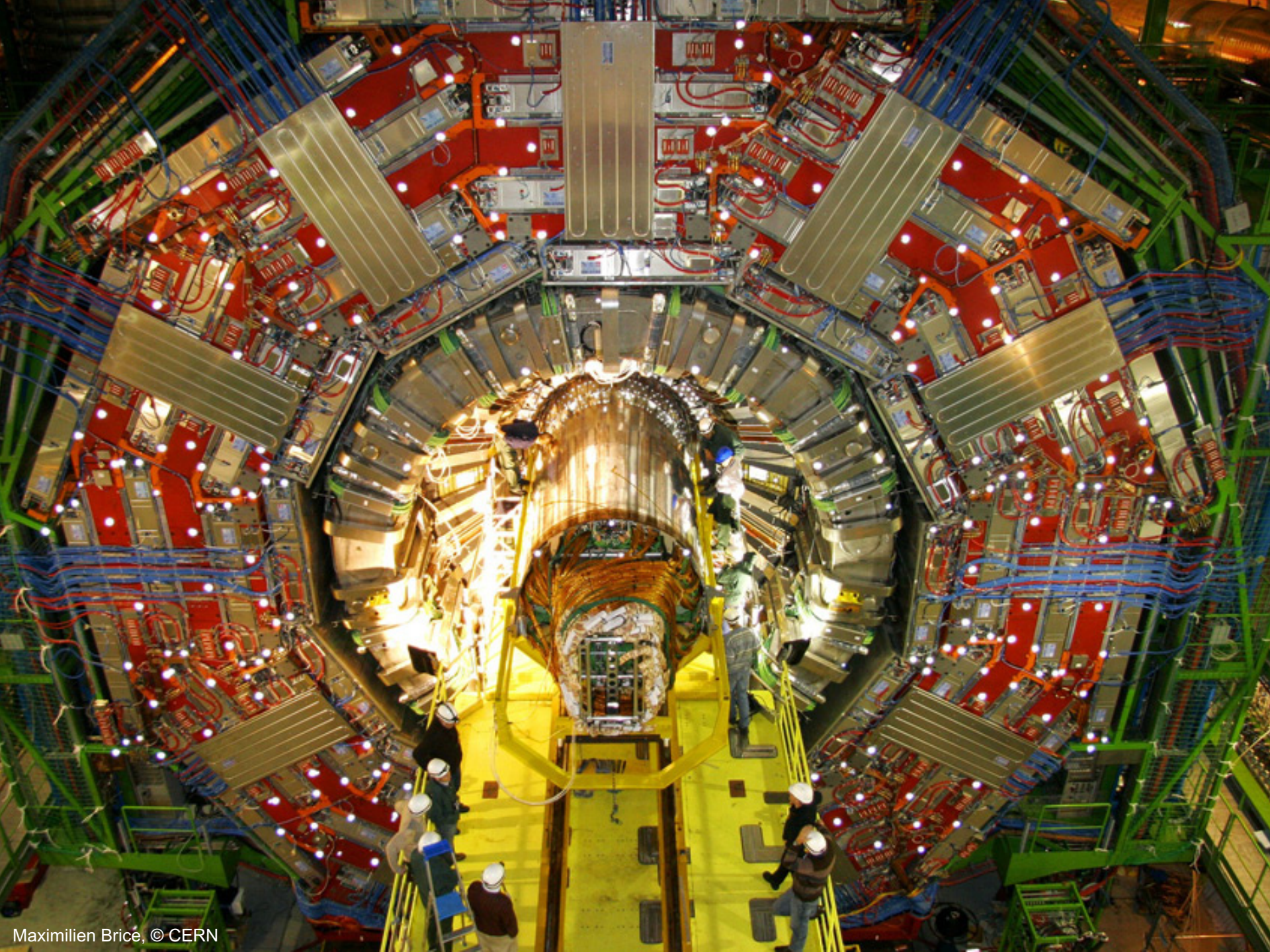

Why big data?
Science
Engineering
Commerce
Society
Source: Wikipedia (Everest)

Science

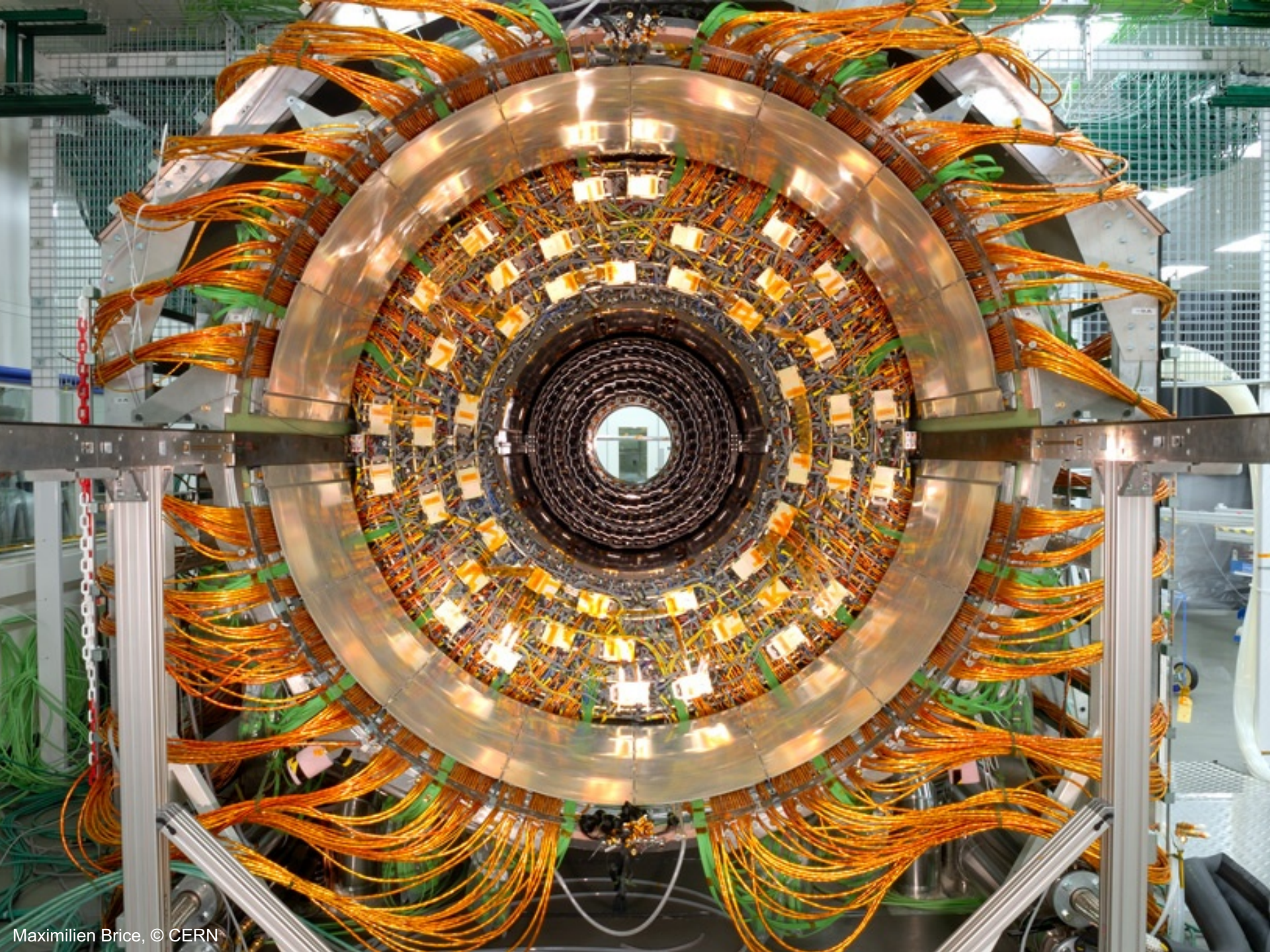
Emergence of the 4th Paradigm

Data-intensive e-Science

Maximilien Brice, © CERN

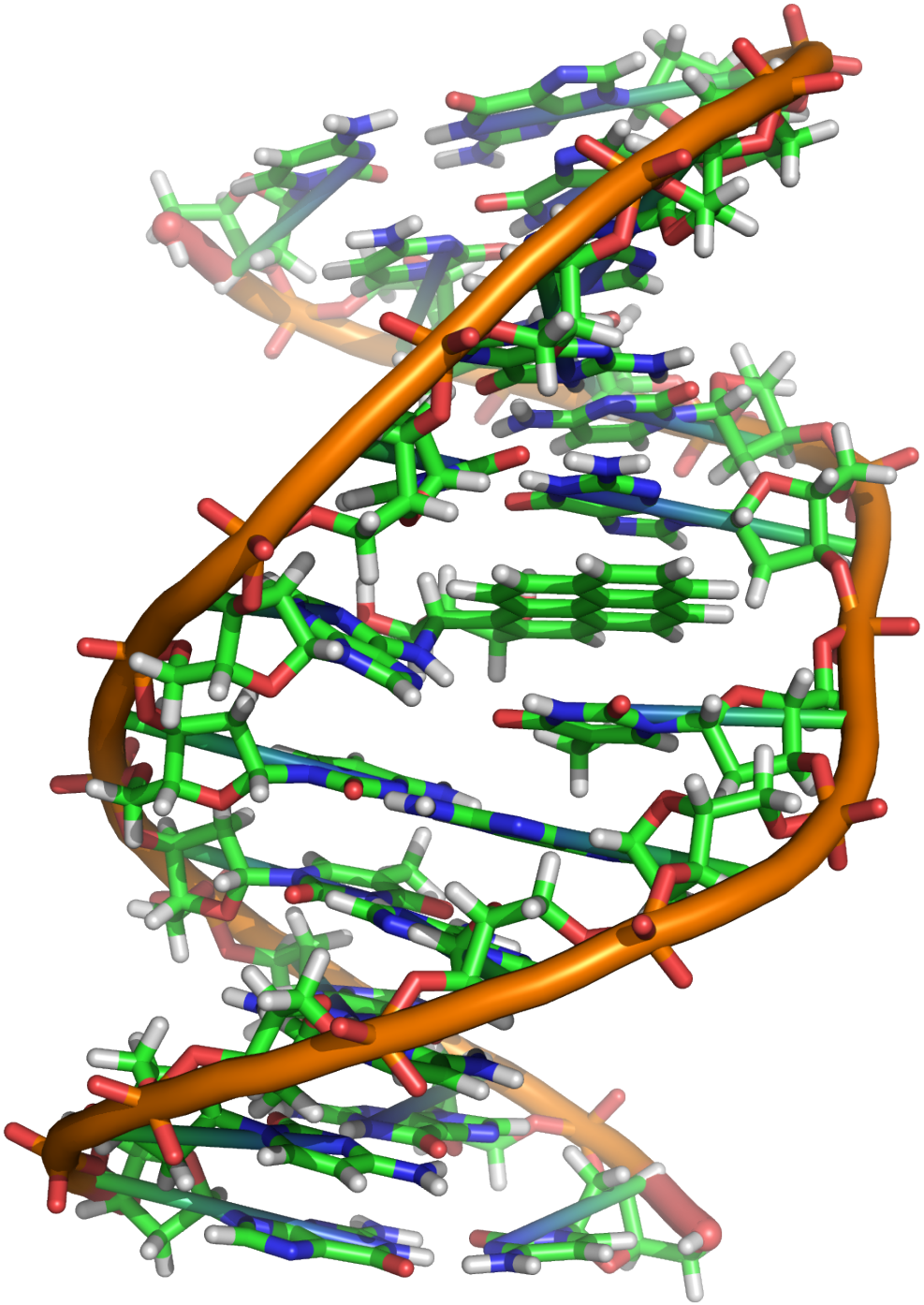


Maximilien Brice, © CERN



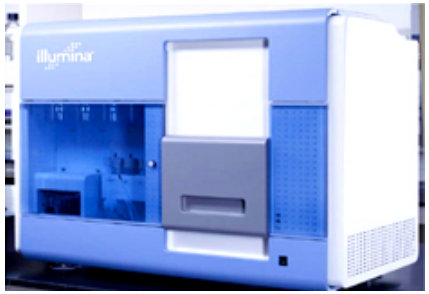
Maximilien Brice, © CERN

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

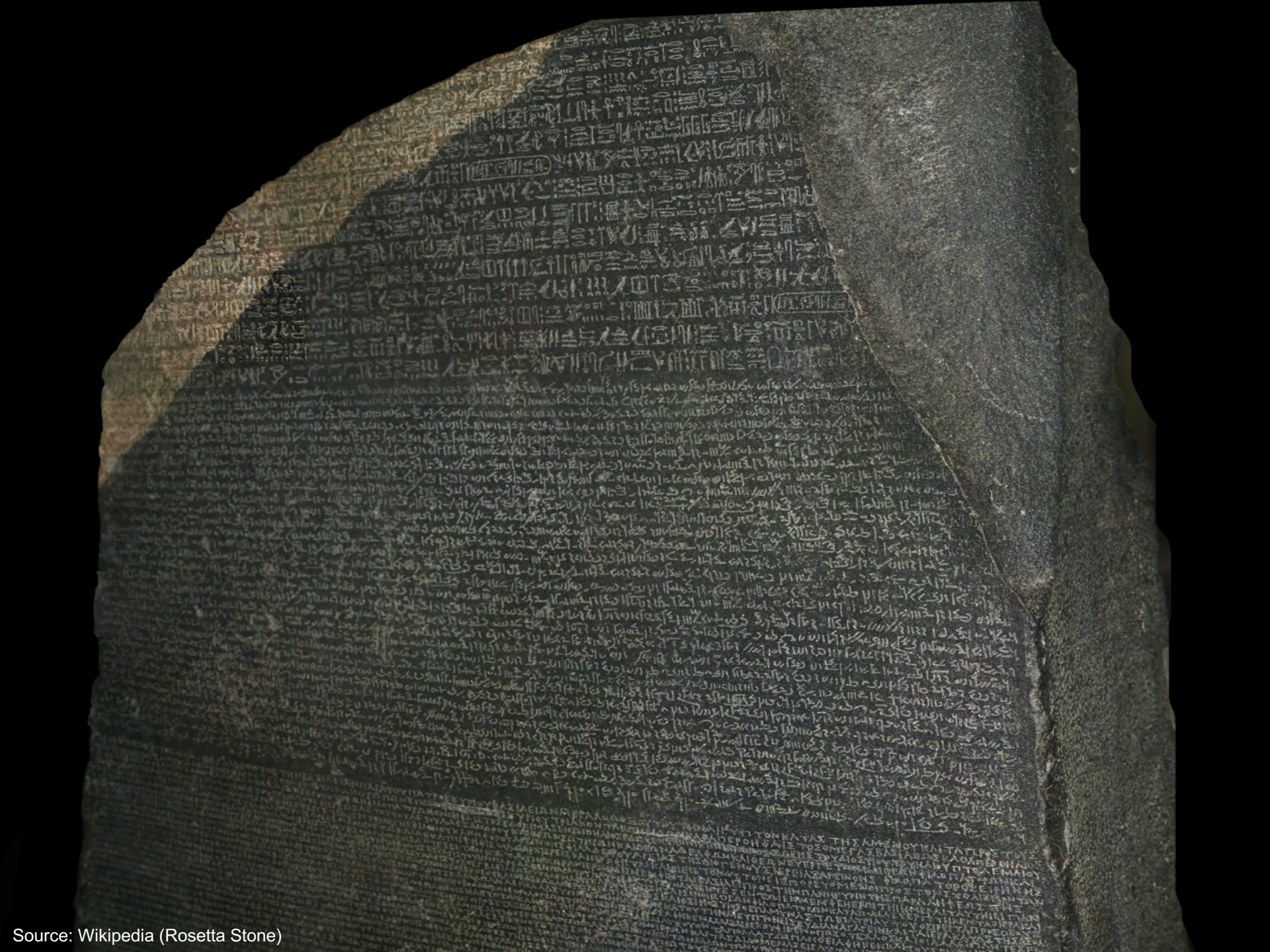
English | Spanish | French | English - detected

How does Google's translation system work?

如何谷歌的翻译系统的工作?

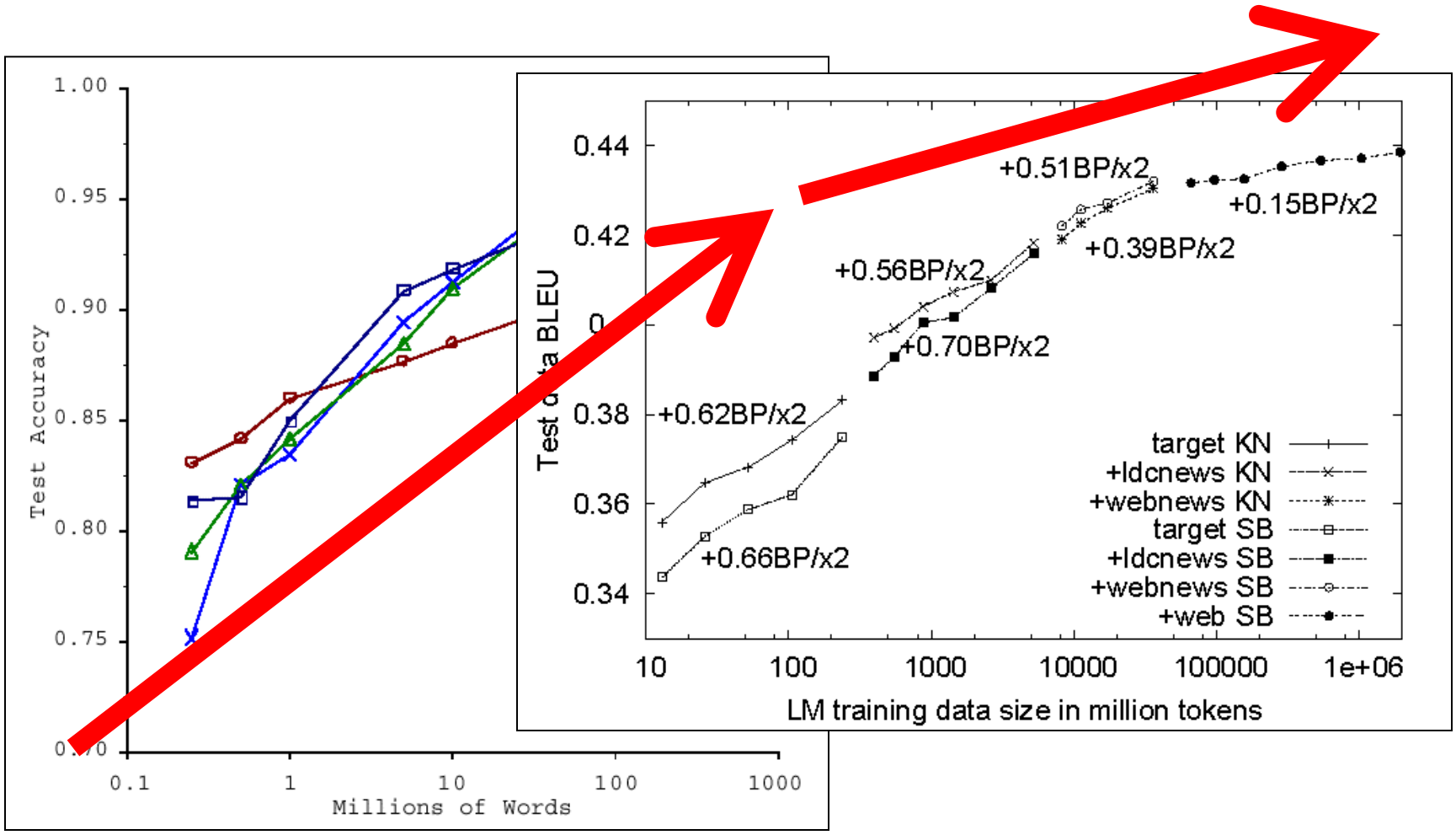
Wrong?

Rúhé gǔgē de fānyì xìtǒng de gōngzuò?



Source: Wikipedia (Rosetta Stone)

No data like more data!



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

Commerce

Know thy customers

Data → Insights → Profit!

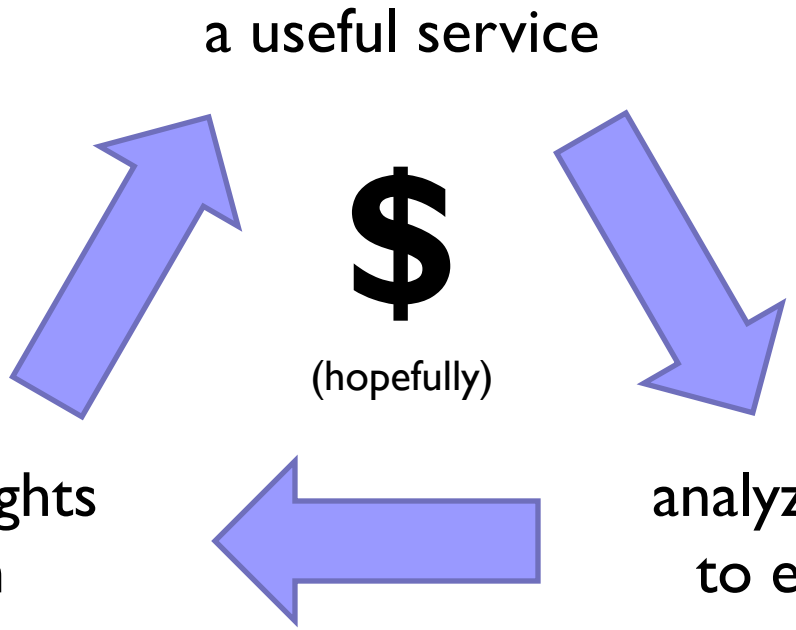
Source: Wikiedia (Shinjuku, Tokyo)

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**

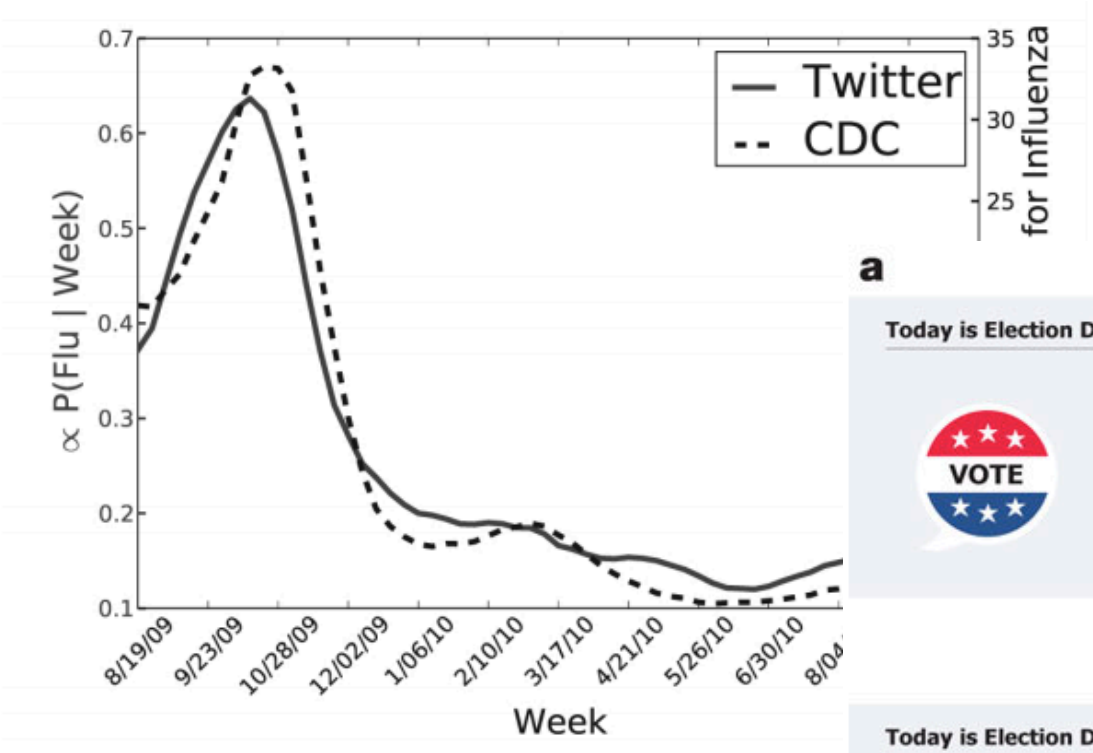
Society

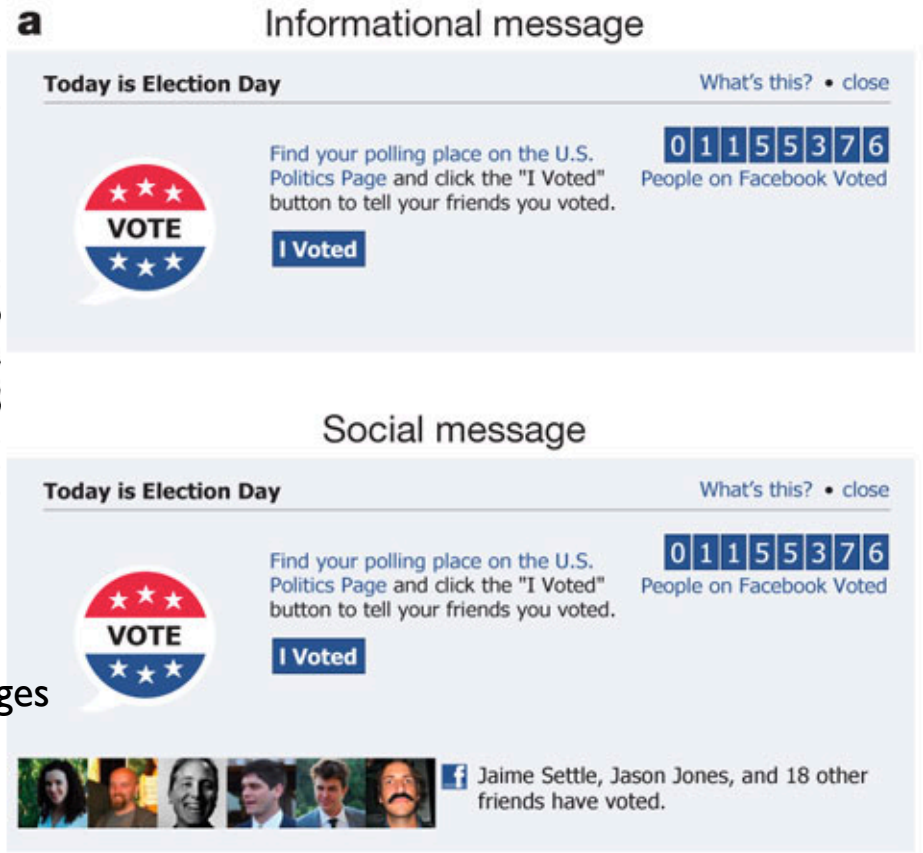
Humans as social sensors

Computational social science

Source: 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)

The Perils of Big Data

The end of privacy

Facebook's terms of service, large-scale government surveillance

The echo chamber

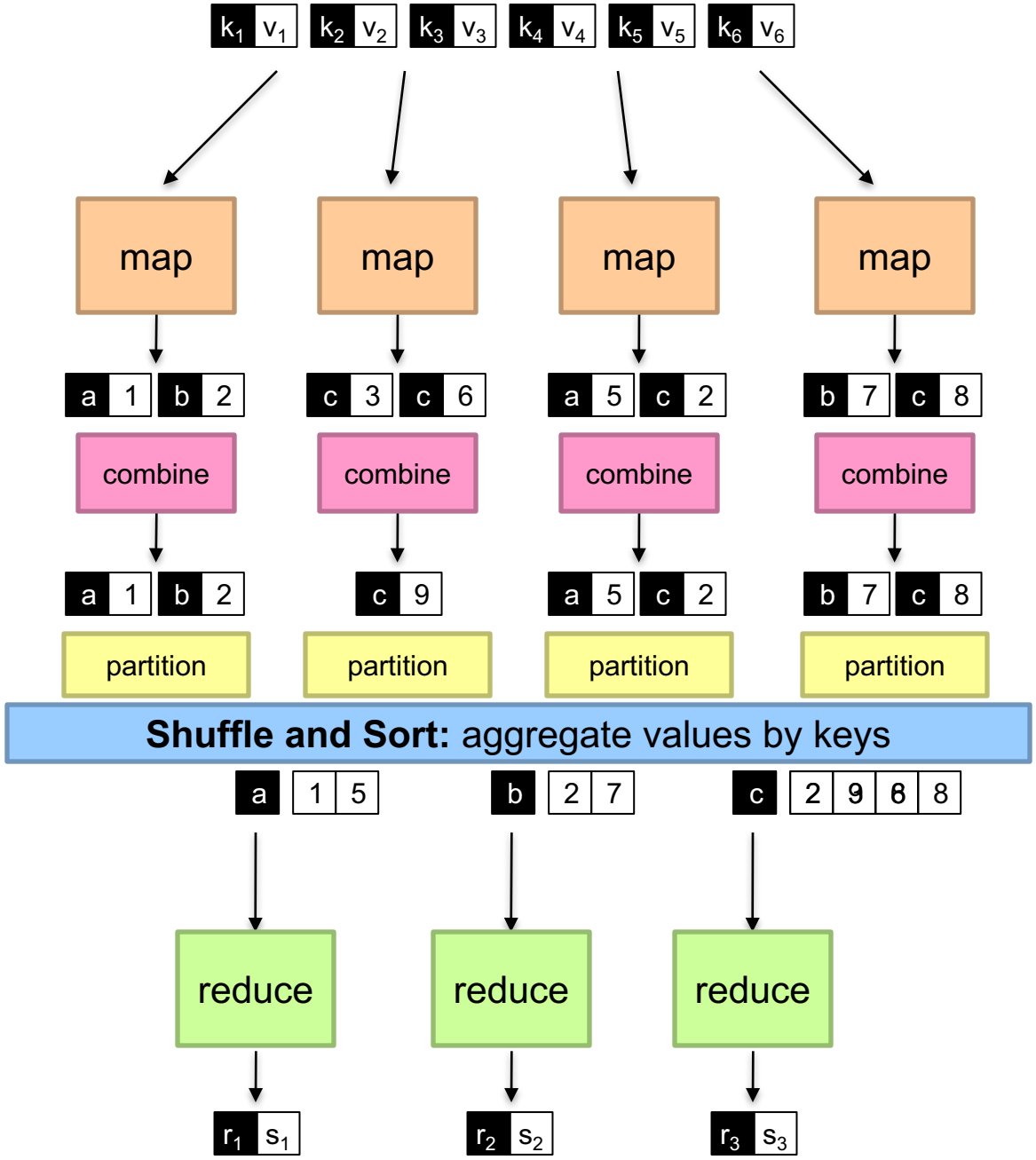
The polarization of politics and erosion of civility

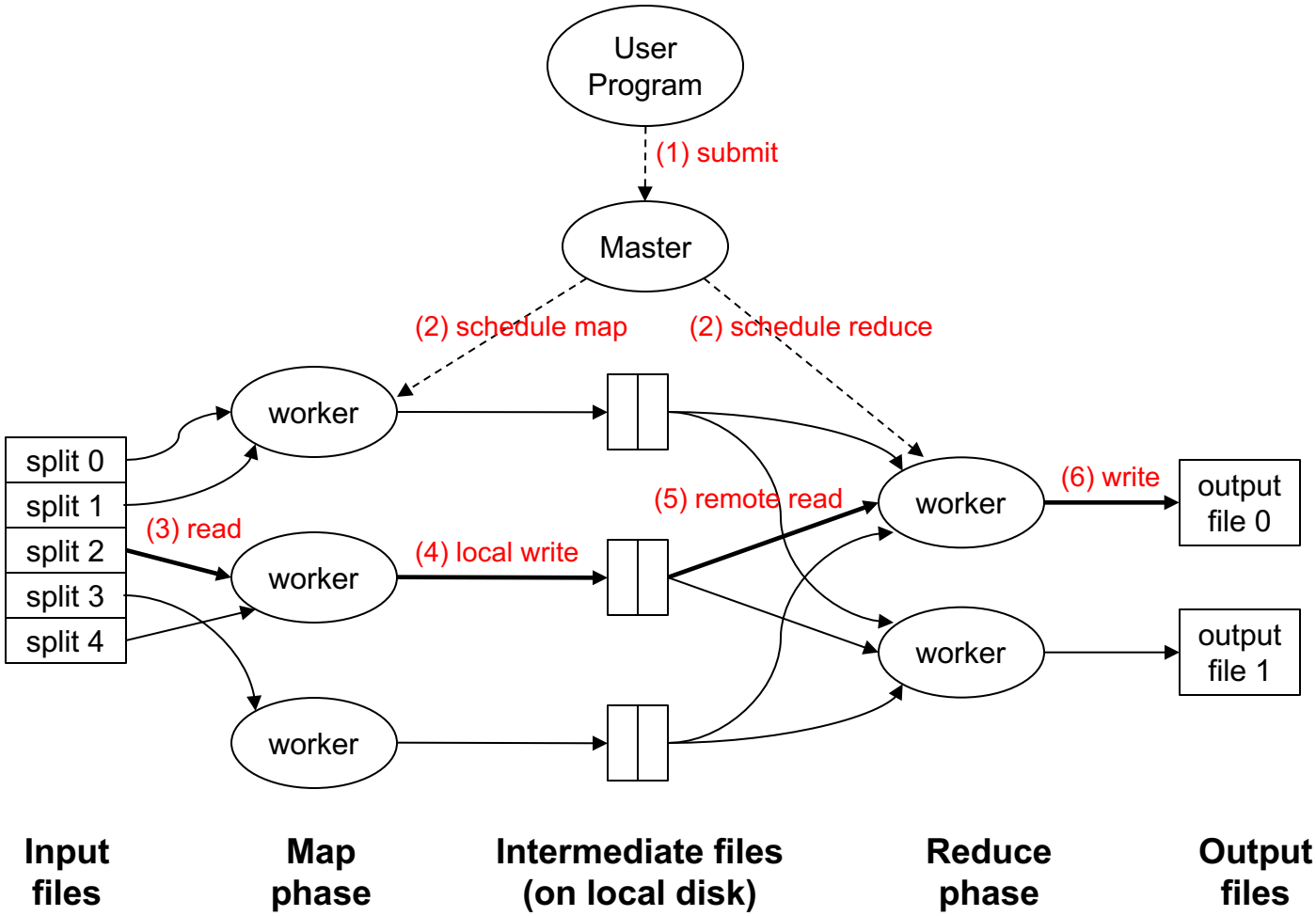
The racist algorithm

Algorithms aren't racist, people are?

We *desperately* need a new "information ethics" to go with 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 6 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

MapReduce is the first instantiation of this idea… but not the last!

Source: Google

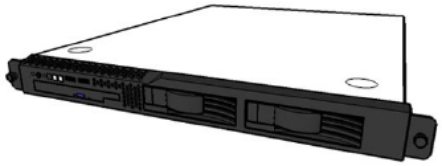
Source: Wikipedia (The Dalles, Oregon)

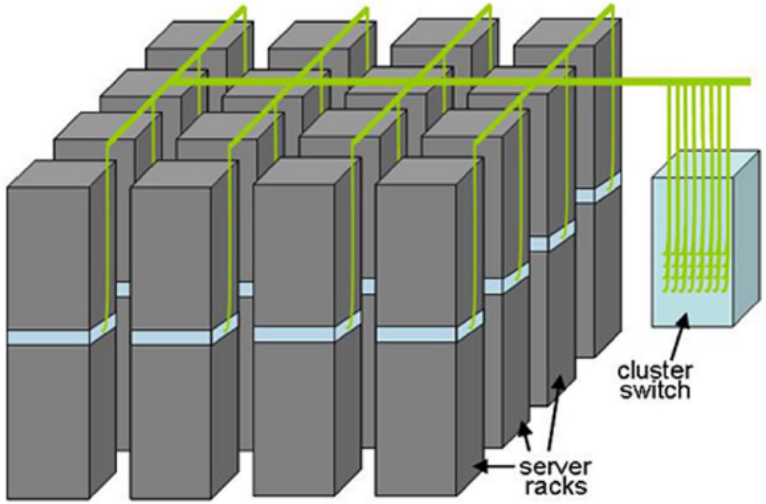
Source: Google

Source: Google

Source: Bonneville Power Administration

Building Blocks





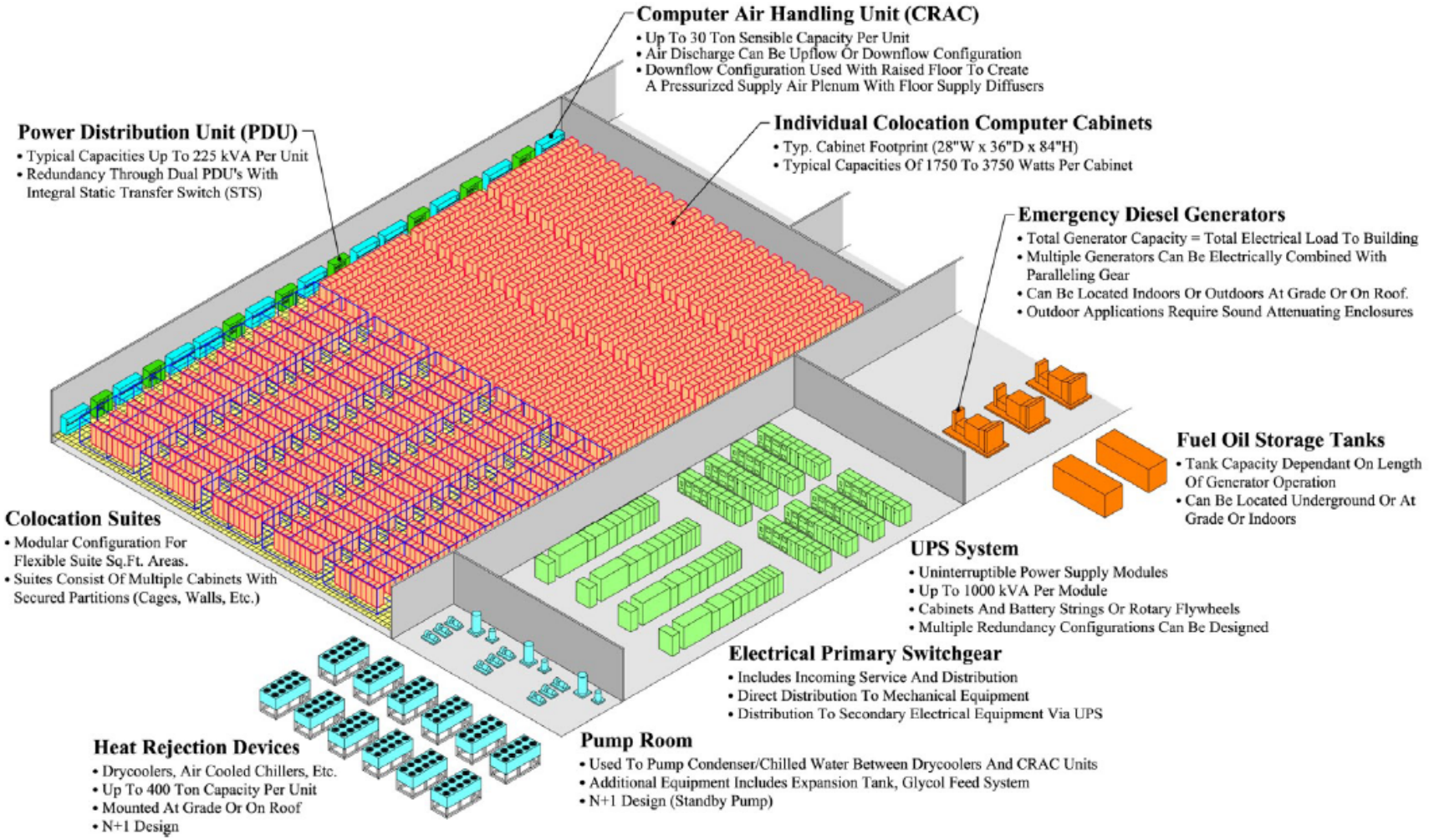
Source: Barroso and Urs Hölzle (2009)

Source: Google

Source: Google

Source: Facebook

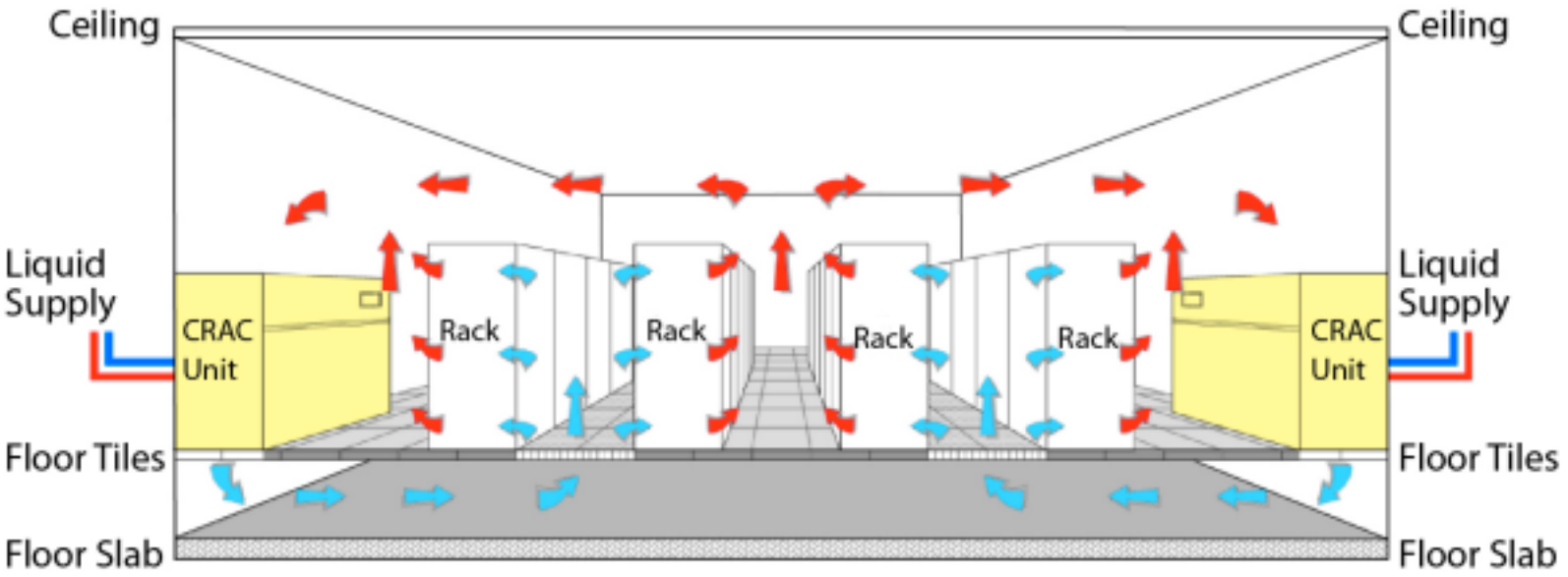
Anatomy of a Datacenter



Source: Barroso and Urs Hölzle (2013)

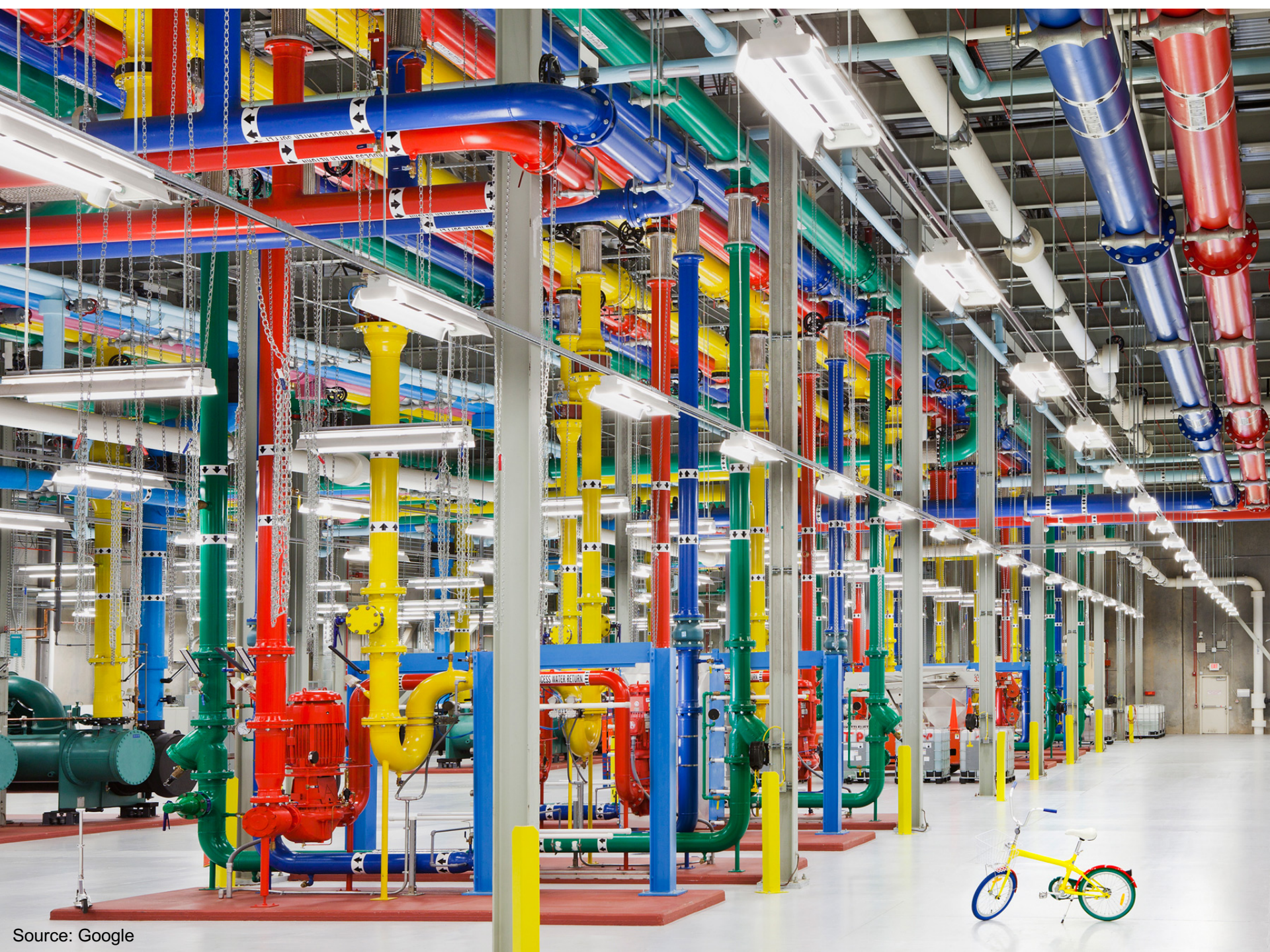
Datacenter cooling

What's a computer?

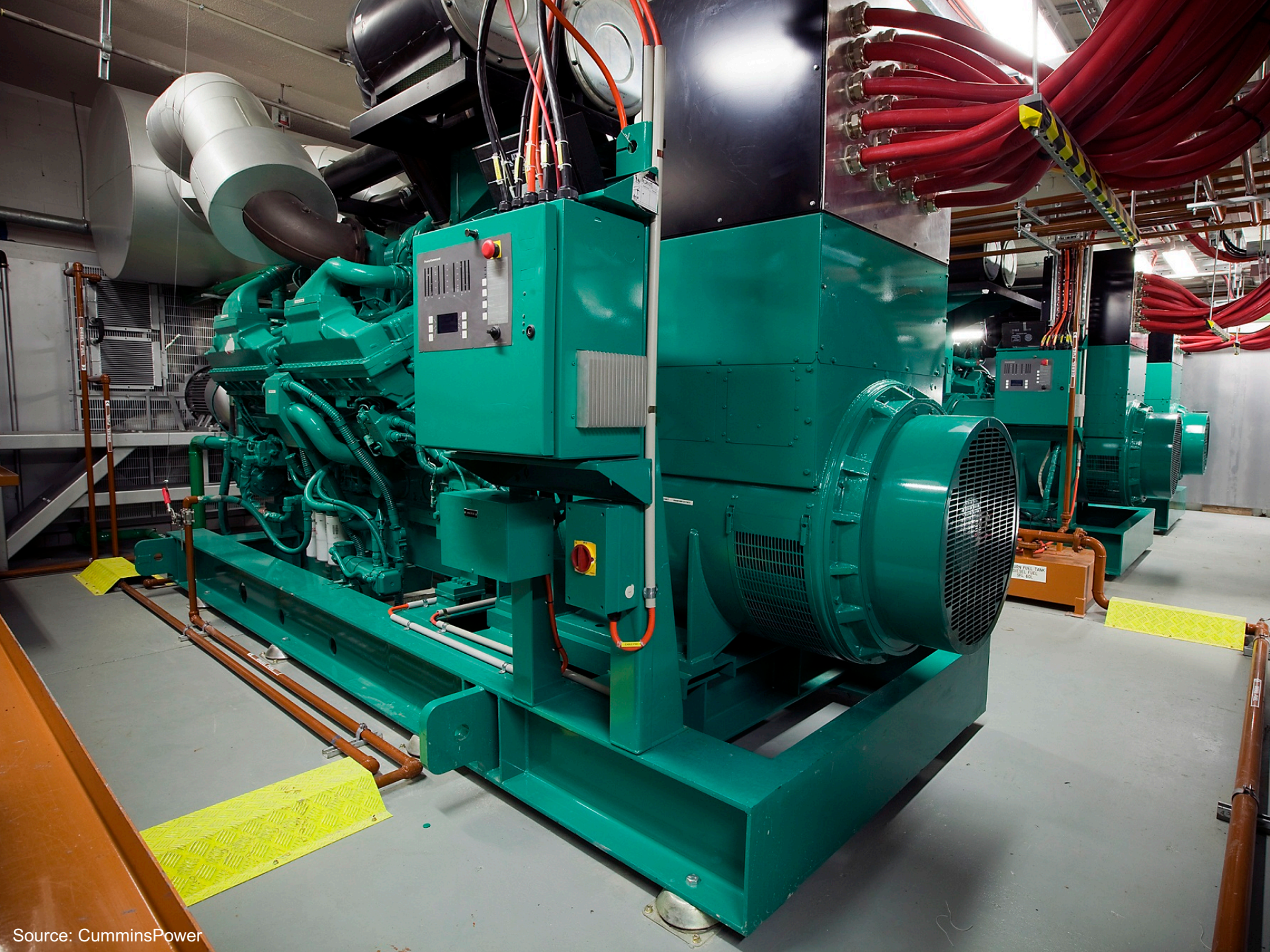


Source: Barroso and Urs Hölzle (2013)

Source: Google

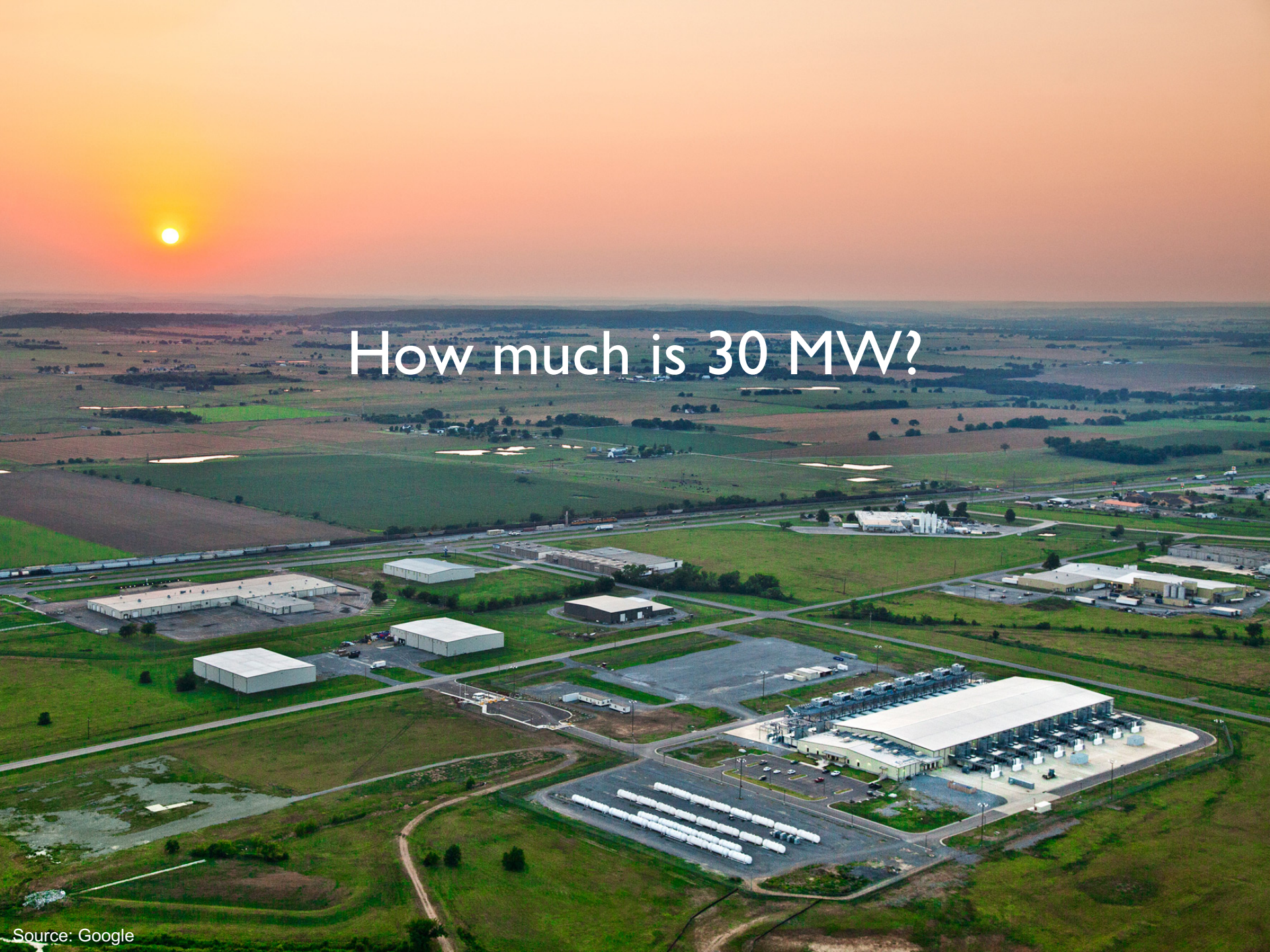


Source: Google



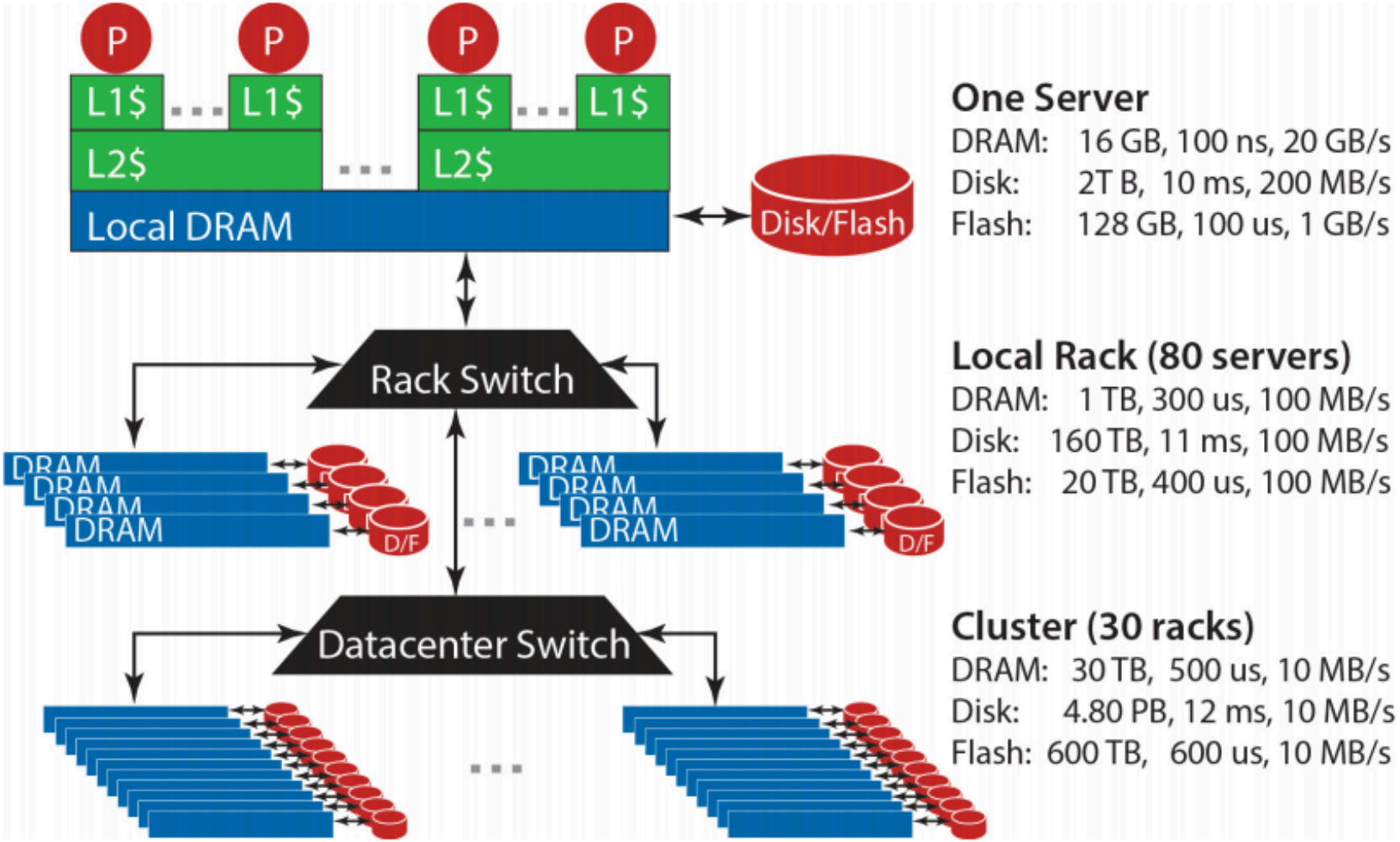
Source: CumminsPower

Source: Google



Source: Google

Datacenter Organization



Source: Barroso and Urs Hölzle (2013)

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

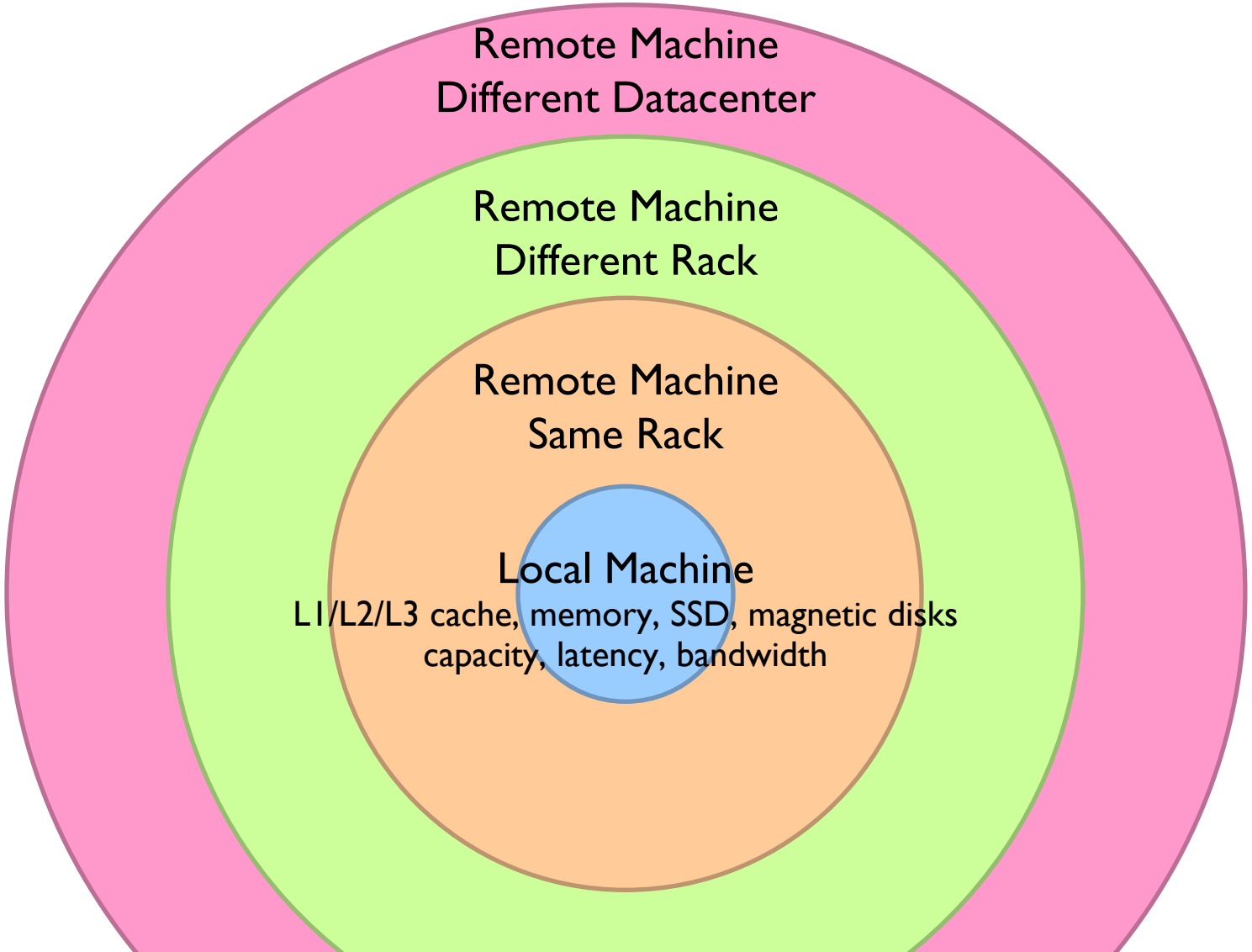
But wait, why do I care?

Intuitions of time and space

How long does it take to read 100 TBs from 100 hard drives?
Now, what about SSDs?

How long will it take to exchange 1b key-value pairs:
Between machines on the same rack?
Between datacenters across the Atlantic?

Storage Hierarchy



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

Google

The datacenter *is* the computer!
"Big ideas"

Scale "out", not "up"

Limits of SMP and large shared-memory machines

Assume that components will break

Engineer software around hardware failures

Move processing to the data

Cluster have limited bandwidth, code is a lot smaller

Process data sequentially, avoid random access

Seeks are expensive, disk throughput is good

Seek vs. Scans

Consider a 1 TB database with 100 byte records

We want to update 1 percent of the records

Scenario 1: Mutate each record

Each update takes ~30 ms (seek, read, write)

108 updates = ~35 days

Scenario 2: Rewrite all records

Assume 100 MB/s throughput

Time = 5.6 hours(!)

Lesson? Random access is expensive!

Source: Ted Dunning, on Hadoop mailing list

The datacenter *is* the computer!
"Big ideas"

Scale "out", not "up"

Limits of SMP and large shared-memory machines

Assume that components will break

Engineer software around hardware failures

Move processing to the data

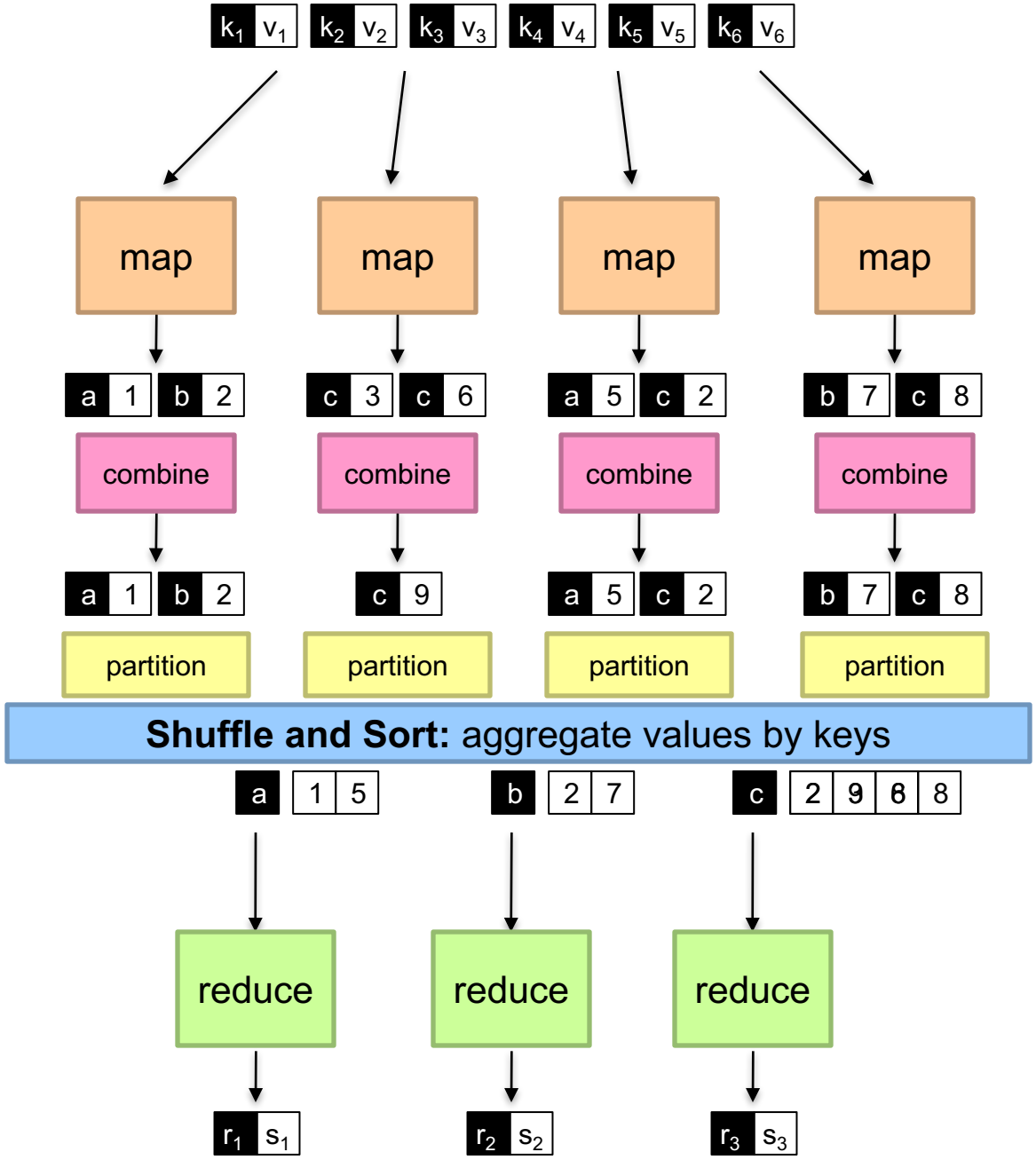
Cluster have limited bandwidth, code is a lot smaller

Process data sequentially, avoid random access

Seeks are expensive, disk throughput is good

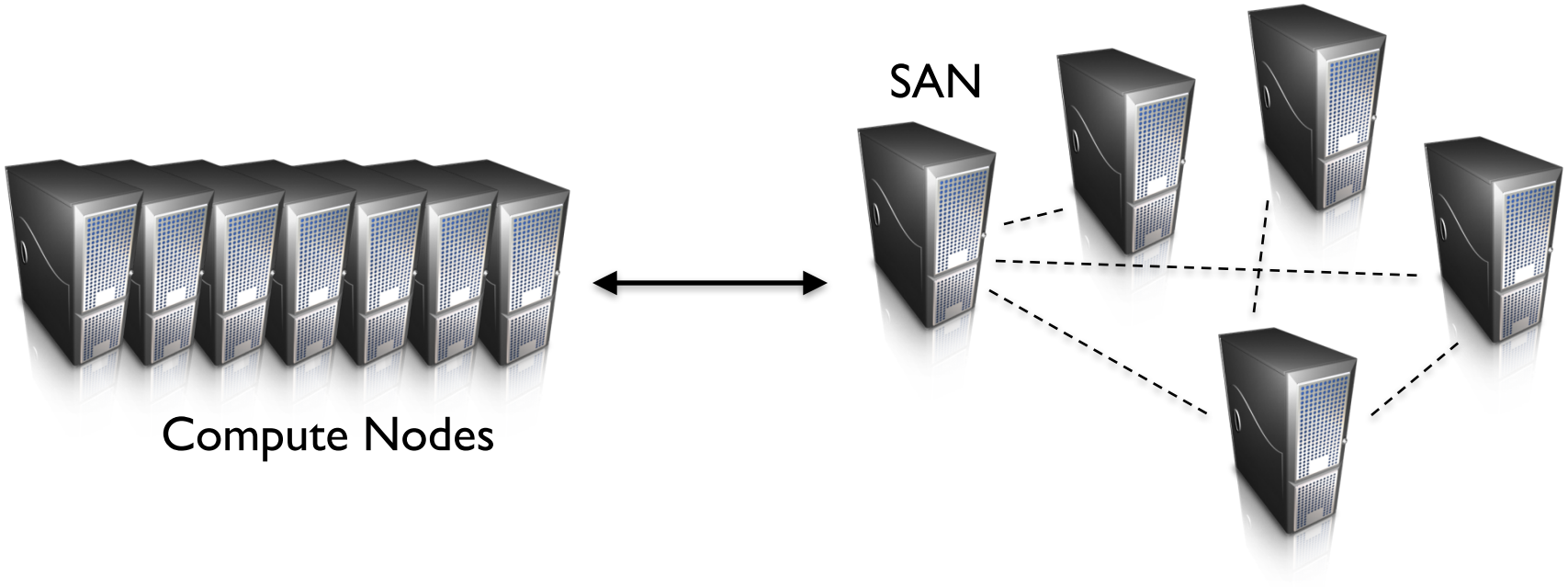
Seamless scalability

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 6 8
reduce
reduce
reduce
r1 s1
r2 s2
r3 s3

How do we get data to the workers?

Let's consider a typical supercomputer…



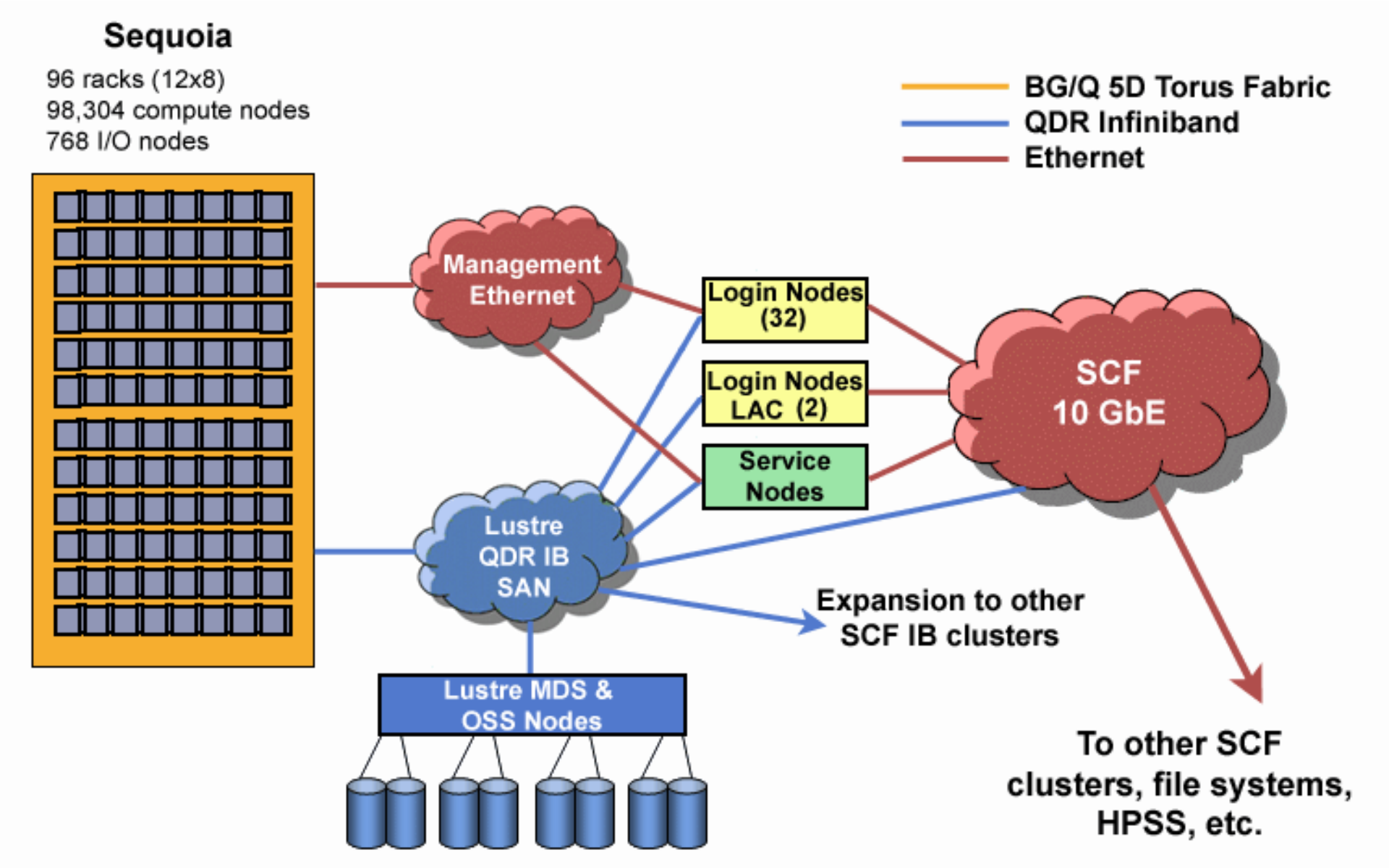
Sequoia

16.32 PFLOPS

98,304 nodes with 1,572,864 million cores

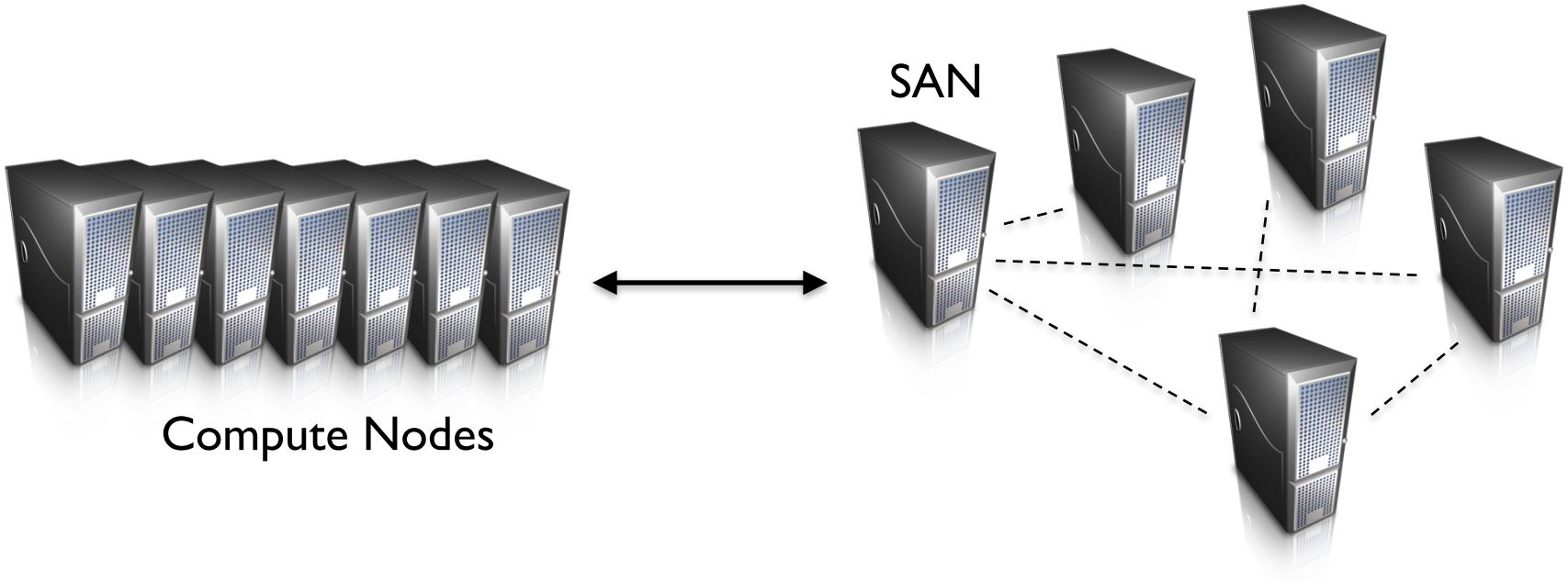
1.6 petabytes of memory

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

How do we get "big data" to the workers?



Why does this make sense for "traditional HPC"?

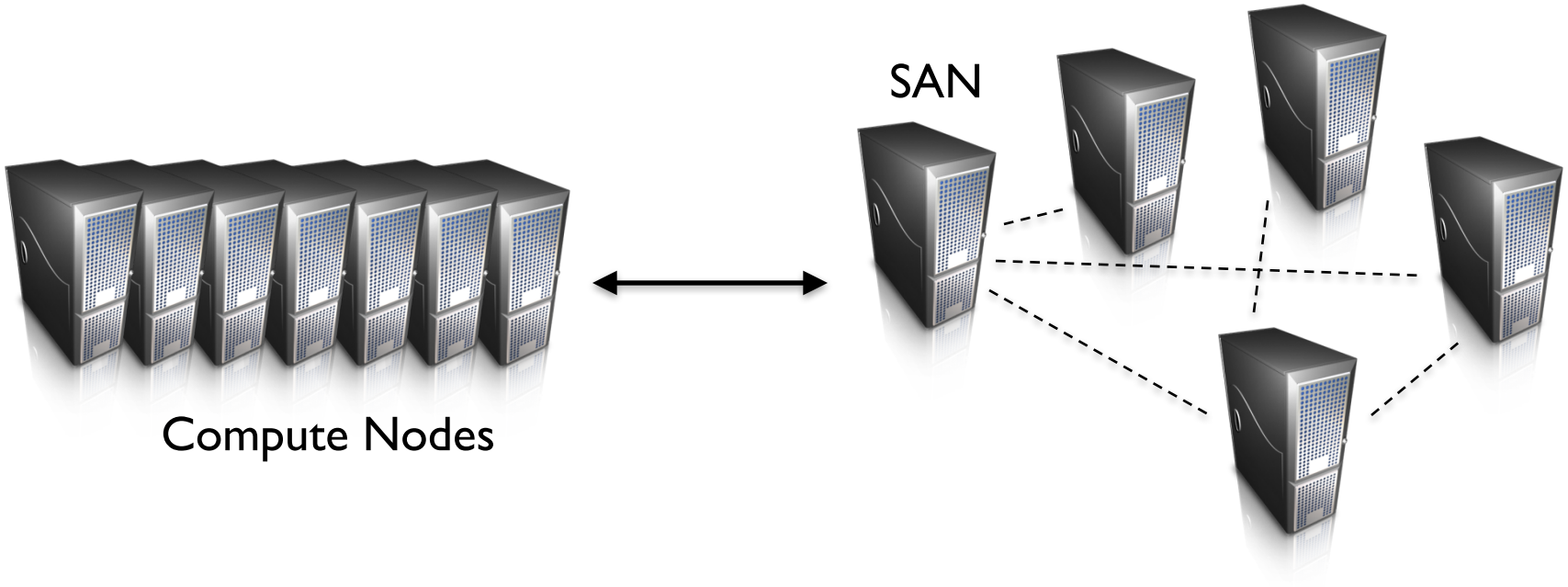
What's the issue for "big data processing"?

What's the solution?

Don't move data to workers… move workers to the data!

Key idea: co-locate storage and compute

Start up worker on nodes that hold the data



What's the solution?

Don't move data to workers… move workers to the data!

Key idea: co-locate storage and compute

Start up worker on nodes that hold the data

We need a distributed file system for managing this

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

Logs are a common case

Large streaming reads over random access

Design for 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 and hold metadata

Simple centralized management

No data caching

Little benefit for streaming reads over large datasets

Simplify the API: not POSIX!

Push many 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

Implementation differences:

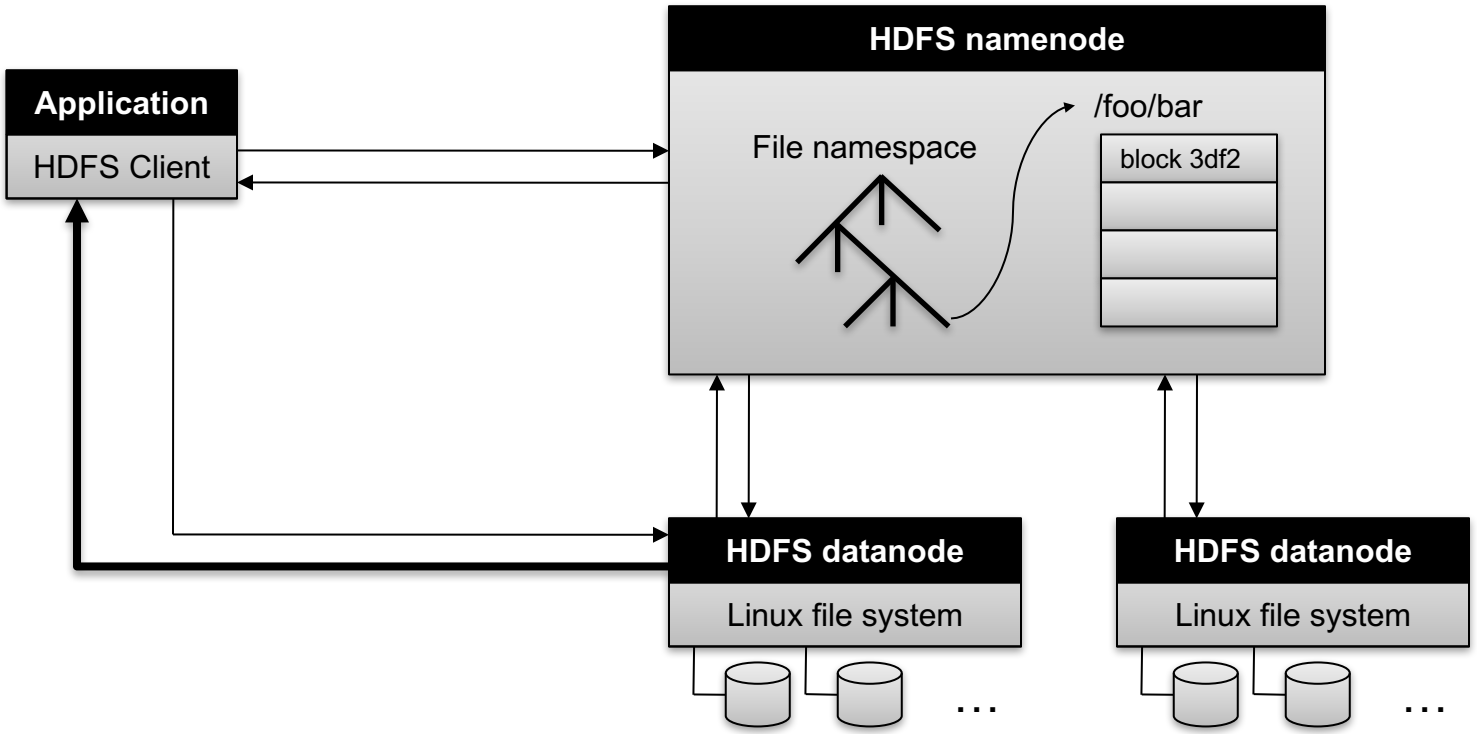
Different consistency model for file appends

Implementation language

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, file-to-block mapping, metadata (ownership, access permissions, etc.)

Coordinating file operations

Directs clients to datanodes for reads and writes

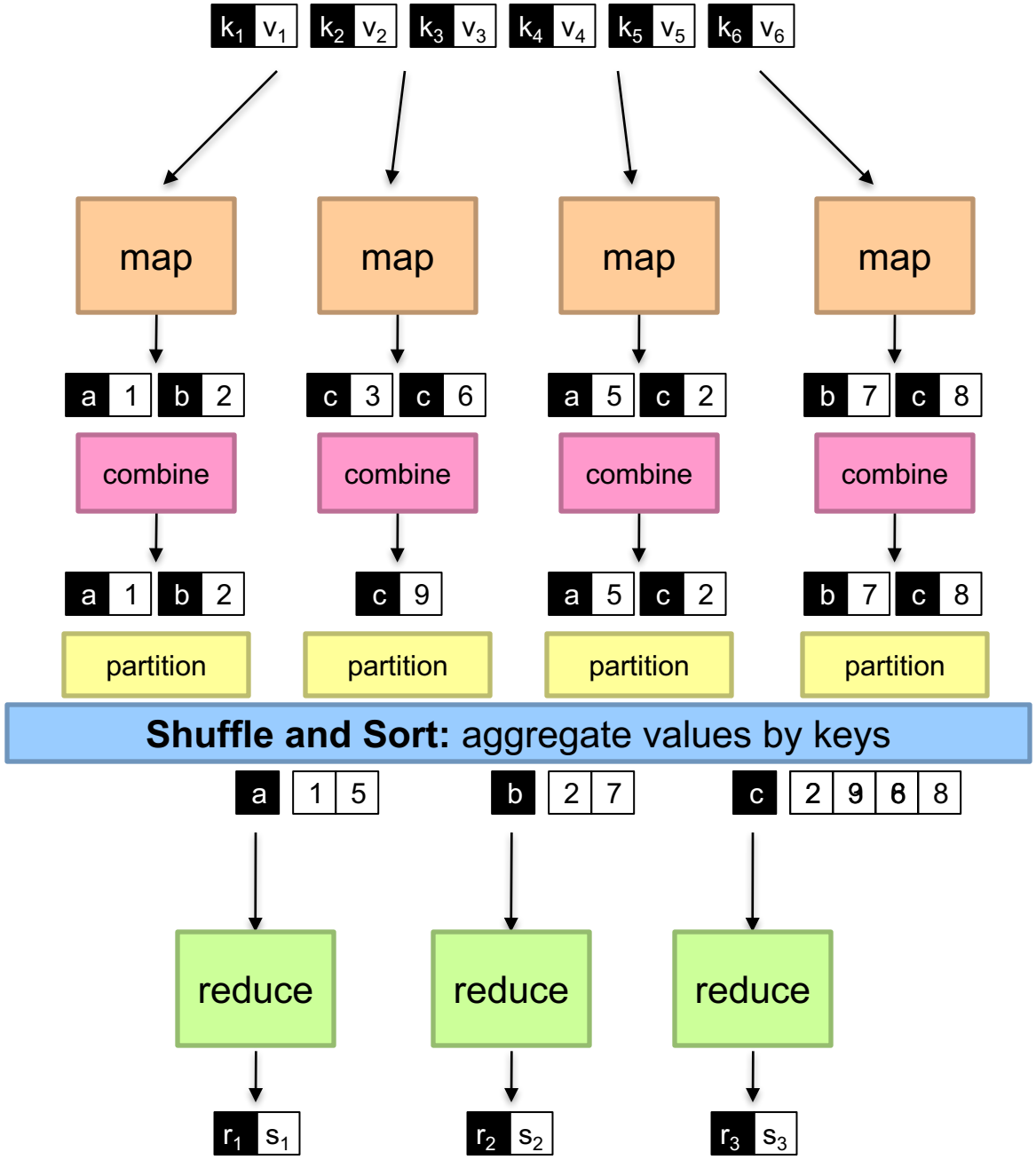
No data is moved through the namenode

Maintaining overall health

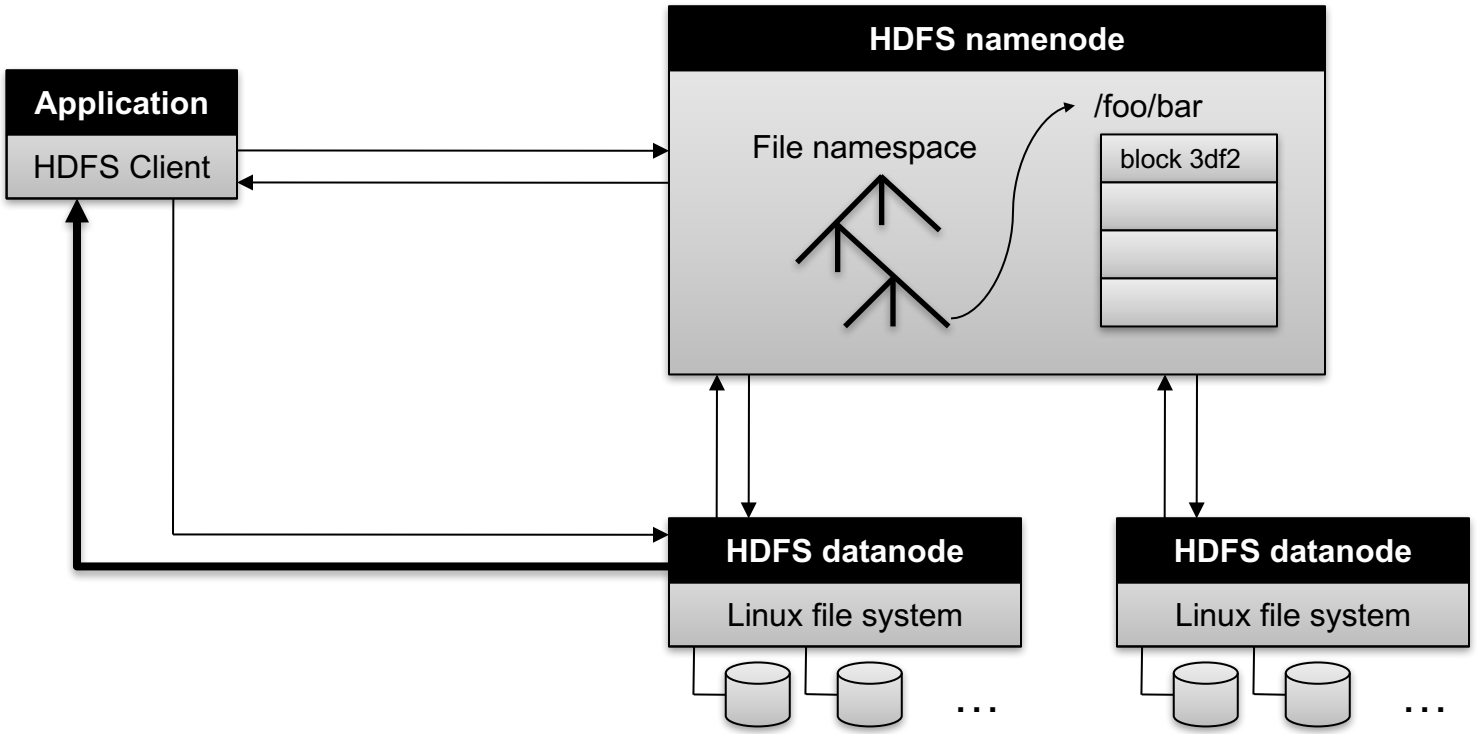
Periodic communication with the datanodes

Block re-replication and rebalancing

Garbage collection


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 6 8
reduce
reduce
reduce
r1 s1
r2 s2
r3 s3

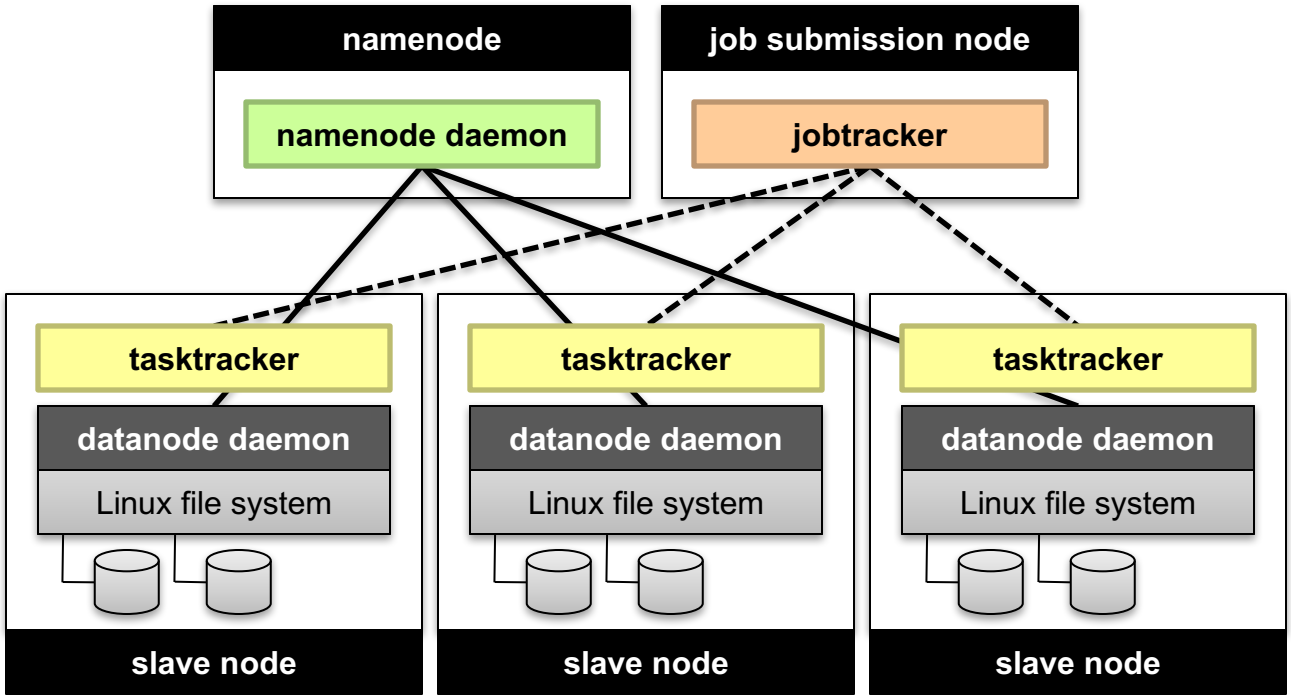
HDFS Architecture



It's a different file system – different set of commands!

Adapted from (Ghemawat et al., SOSP 2003)

Putting everything together…



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


If you’re not (yet) registered…
Is there still hope?

Source: Wikipedia (Japanese rock garden)