



# Data-Intensive Distributed Computing

## CS 431/631 451/651 (Winter 2019)

Part 1: MapReduce Algorithm Design (3/4)  
January 15, 2019

Adam Roegiest  
Kira Systems

These slides are available at <http://lintool.github.io/bigdata-2018f/>

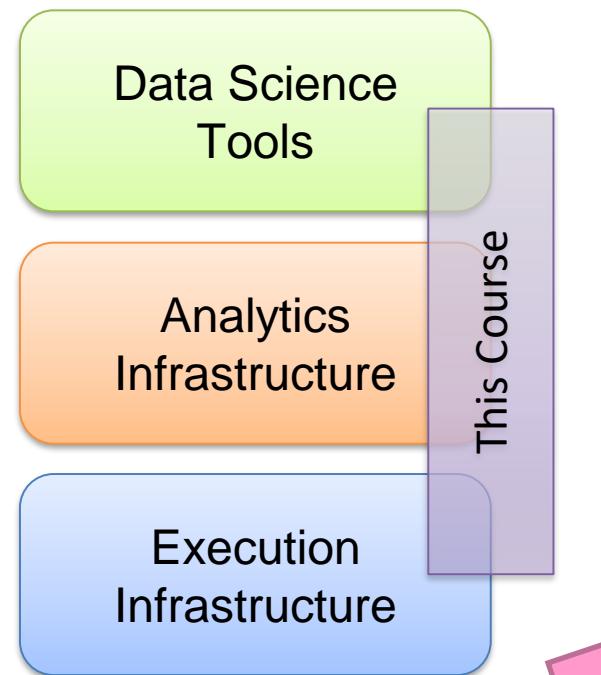


This work is licensed under a Creative Commons Attribution-Noncommercial-Share Alike 3.0 United States  
See <http://creativecommons.org/licenses/by-nc-sa/3.0/us/> for details

# Agenda for Today

Cloud computing  
Datacenter architectures  
Hadoop cluster architecture  
MapReduce physical execution

# Today



“big data stack”

An aerial photograph showing a vast expanse of white and grey cumulus clouds against a clear blue sky. The clouds are dense and layered, creating a textured pattern across the frame. In the lower right quadrant, a darker, more solid cloud formation is visible, possibly indicating a front or a different atmospheric layer.

## Aside: Cloud Computing

# 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

Javascript! (ugh)

Examples: Facebook, YouTube, Gmail, ...

“The network is the computer”: take two

User data is stored “in the clouds”

Rise of the tablets, smartphones, etc. (“thin clients”)

Browser is the OS

GENERAL  ELECTRIC

R 13%

8 9 0 1 2 2 1 0 9 8 8 9 0 2 1 0 9 8 8 9 0 1 2  
7 6 5 4 3 3 4 5 7 7 6 5 4 3 3 4 5 6 7 7 6 5 3  
K I L O W A T T H O U R S

CL 200

TYPE I-60-S  
SINGLE STATOR  FM 2S  
WATTHOUR METER

240V

3W

CAT. NO.

720X1G1

TA 30

Kh 7.2

60~

397128

• 44 617 187 •

MADE IN U.S.A.

P  
G  
E

# Utility Computing

What?

Computing resources as a metered service (“pay as you go”)

Why?

Cost: capital vs. operating expenses

Scalability: “infinite” capacity

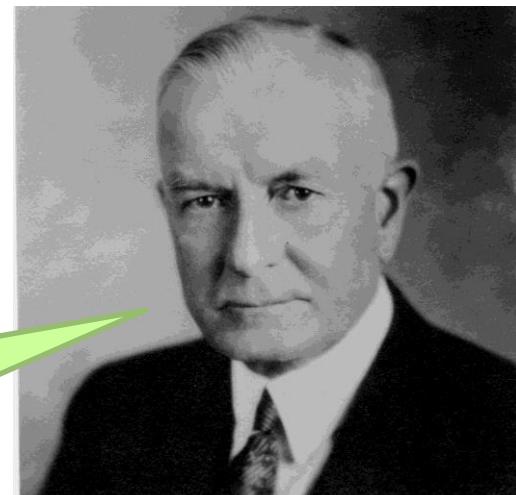
Elasticity: scale up or down on demand

Does it make sense?

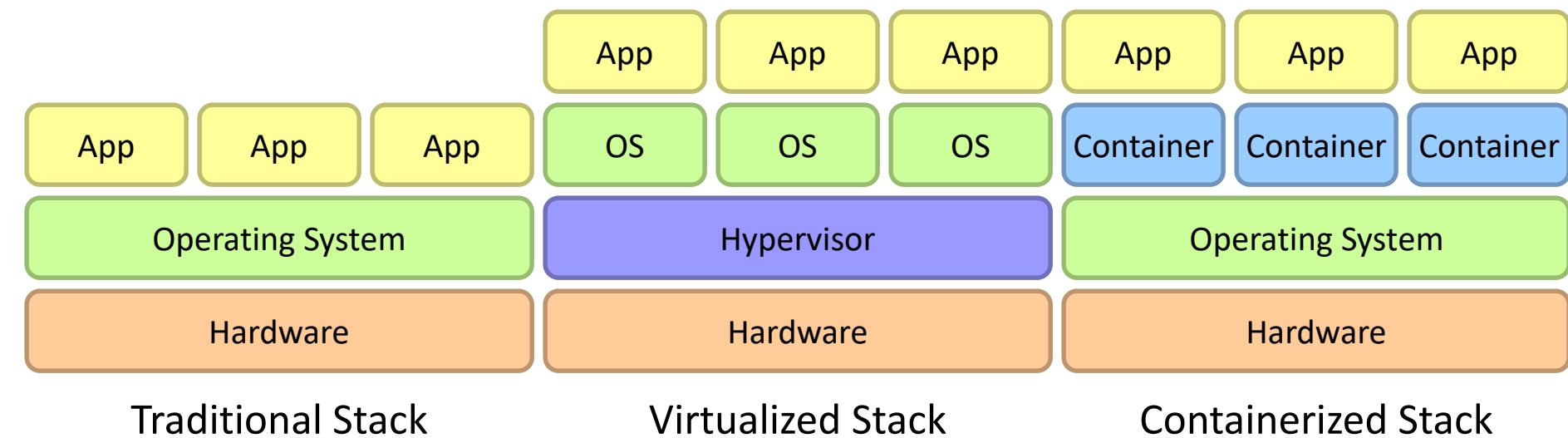
Benefits to cloud users

Business case for cloud providers

I think there is a world market for about five computers.



# Evolution of the Stack



# Everything as a Service

## Infrastructure as a Service (IaaS)

Why buy machines when you can rent them instead?

Examples: Amazon EC2, Microsoft Azure, Google Compute

## Platform as a Service (PaaS)

Give me a nice platform and take care of maintenance, upgrades, ...

Example: Google App Engine

## Software as a Service (SaaS)

Just run the application for me!

Example: Gmail, Salesforce

# Everything as a Service

## Database as a Service

Run a database for me

Examples: Amazon RDS, Microsoft Azure SQL, Google Cloud BigTable

## Search as a Service

Run a search engine for me

Example: Amazon Elasticsearch Service

## Function as a Service

Run this function for me

Example: Amazon Lambda, Google Cloud Functions

# 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 clusters on-demand in the cloud

Commoditization and democratization of big data capabilities

An aerial photograph showing a vast expanse of white and grey cumulus clouds against a clear blue sky. The clouds are dense and layered, creating a textured pattern across the frame. In the lower right quadrant, a darker, more solid cloud formation is visible, possibly indicating a front or a different atmospheric layer.

So, what *is* the cloud?

# What is the Matrix?

The Matrix is a science fiction film series directed by the Wachowskis. It consists of three main films: "The Matrix" (1999), "The Matrix Reloaded" (2003), and "The Matrix Revolutions" (2003). The story follows Neo, a computer hacker who discovers that he is actually a slave in a simulated reality called the Matrix. He is recruited by Trinity and Neo to join the resistance against the machines that control the world. The Matrix is a complex system of simulated reality that the machines use to keep humans complacent and provide them with energy.





Source: Wikipedia (The Dalles, Oregon)

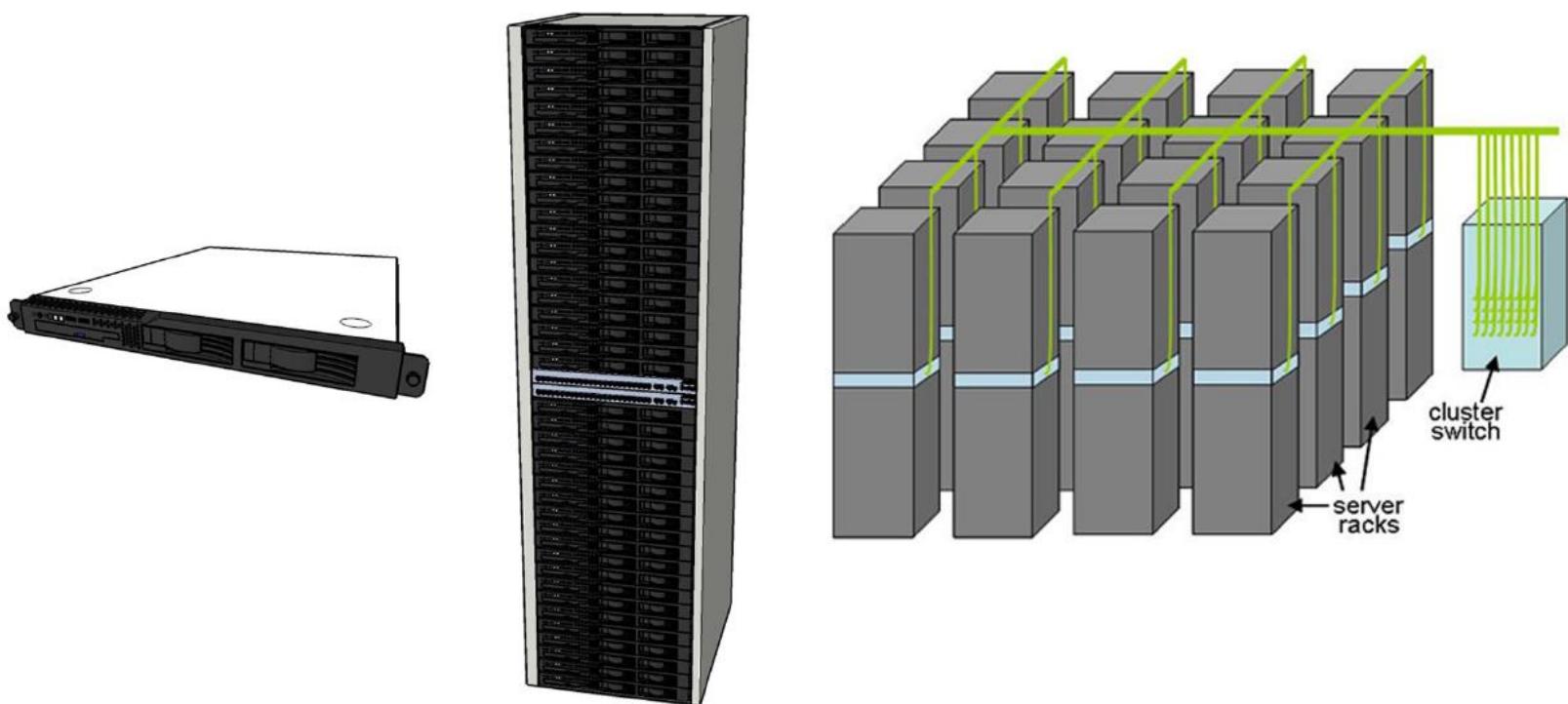


Source: Bonneville Power Administration





# Building Blocks

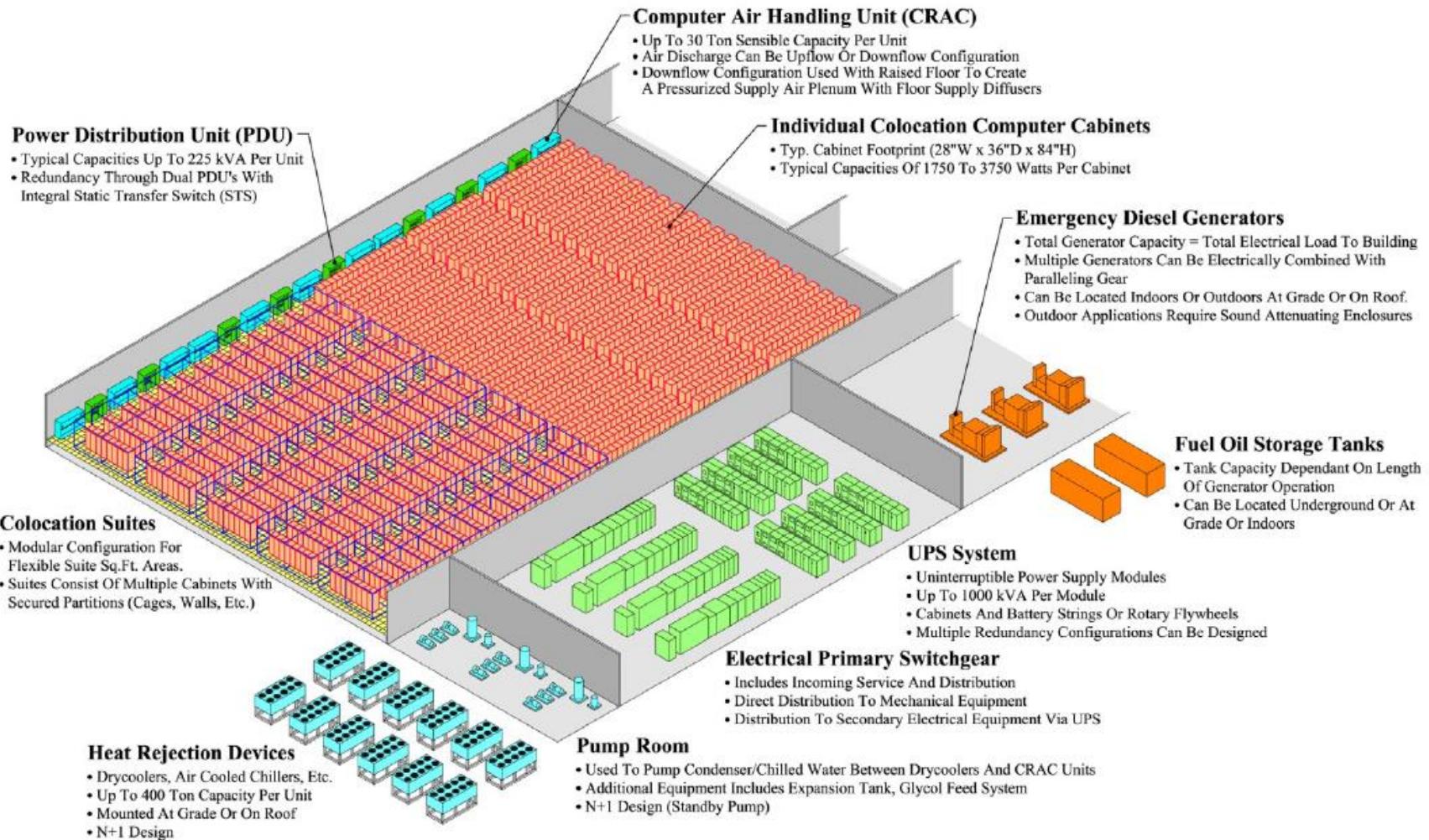






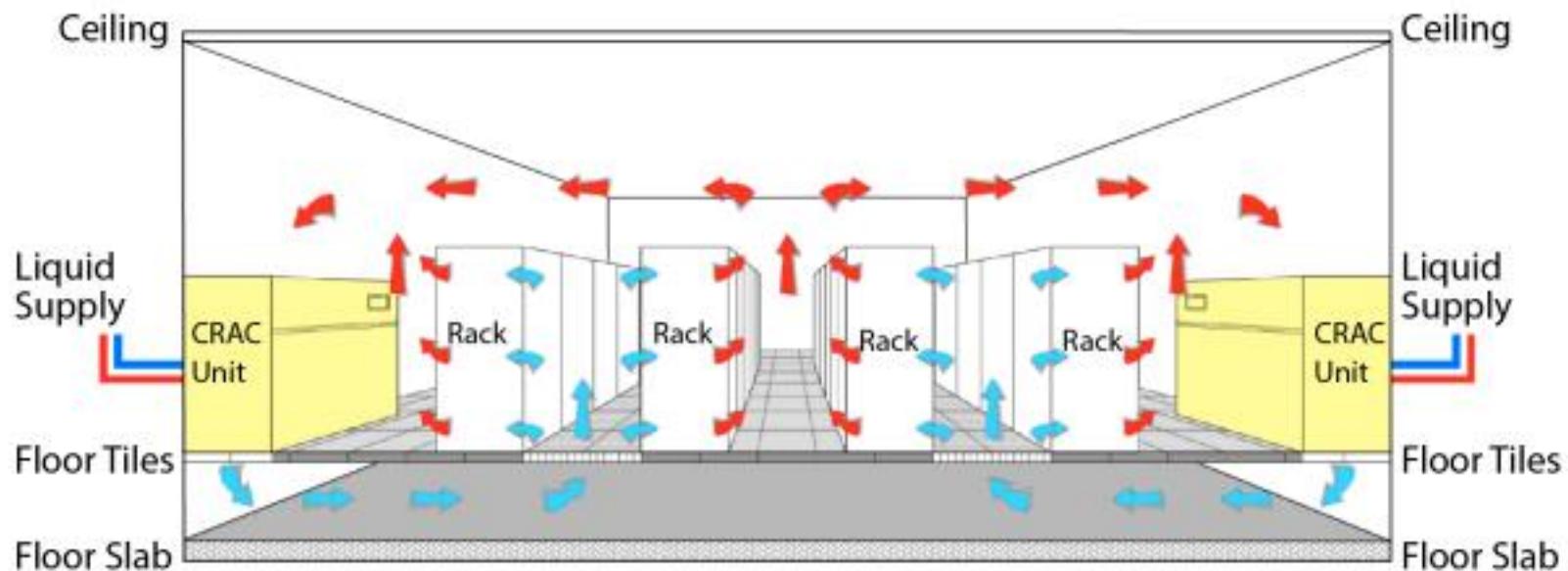


# Anatomy of a Datacenter

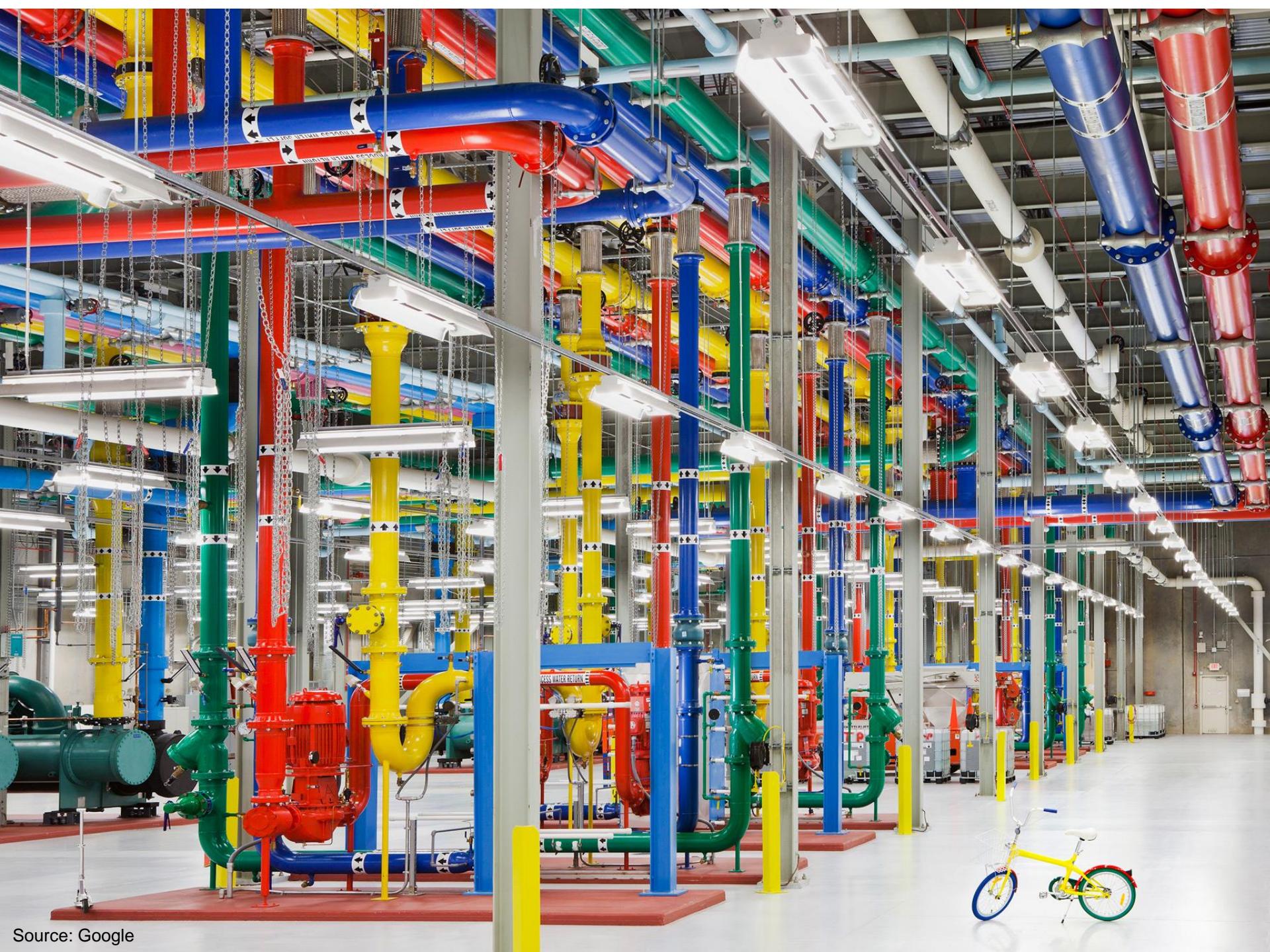


# Datacenter cooling

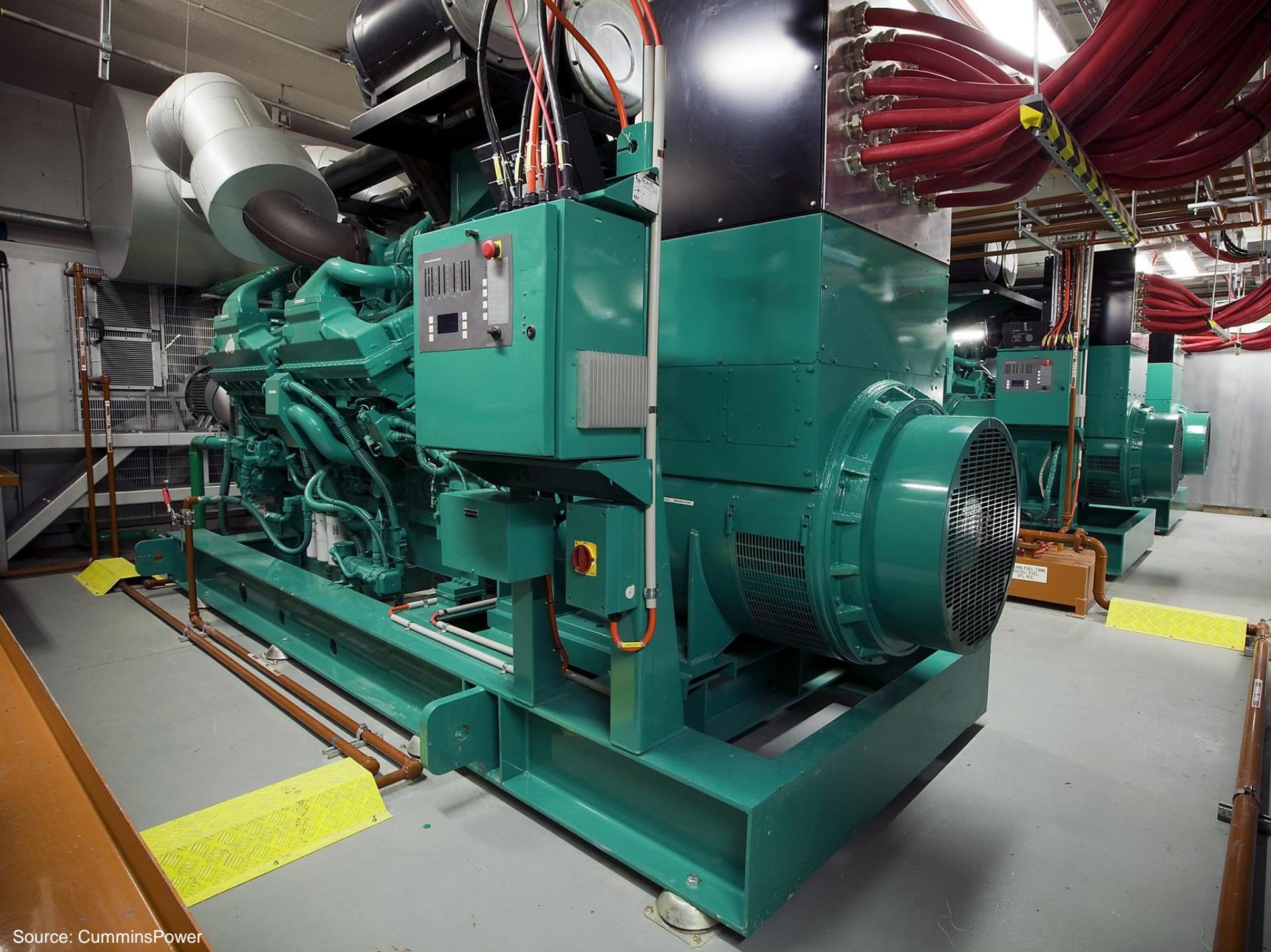
## What's a computer?



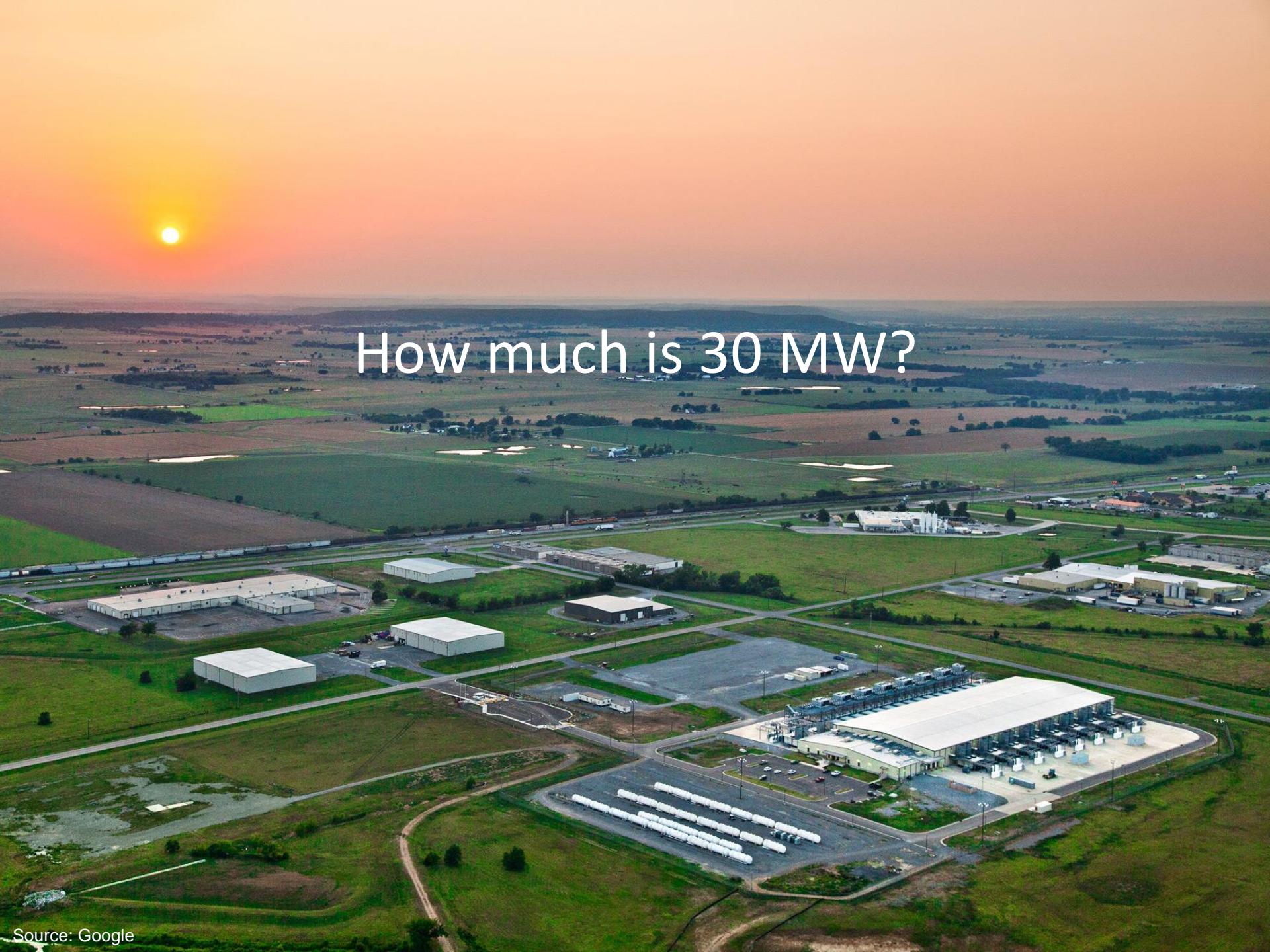




Source: Google

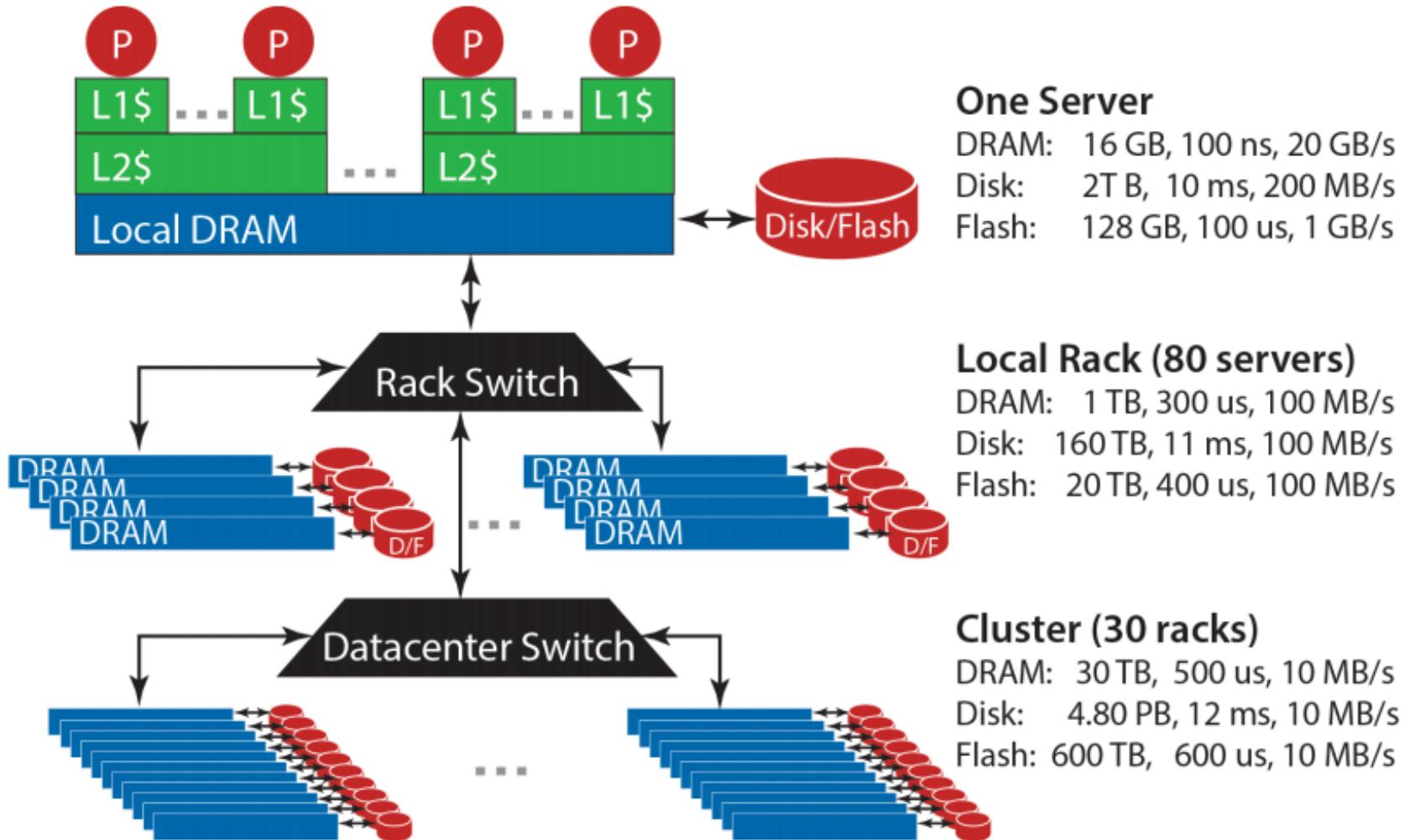




An aerial photograph of a large industrial facility during sunset. The sky is a vibrant orange and yellow. In the foreground, there's a mix of green fields and industrial buildings, including several large white structures and smaller utility buildings. A road or highway cuts through the area. In the background, more fields extend to a distant horizon under the setting sun.

# How much is 30 MW?

# Datacenter Organization



# 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

Wait, why do we care?

# Mechanical Sympathy

“You don’t have to be an engineer to be  
be a racing driver, but you do have to  
have mechanical sympathy”

– Formula One driver Jackie Stewart



Data Science  
Tools

Analytics  
Infrastructure

Execution  
Infrastructure

This Course

“big data stack”

# Intuitions of time and space

How long does it take to read 100 TBs from 100 hard drives?

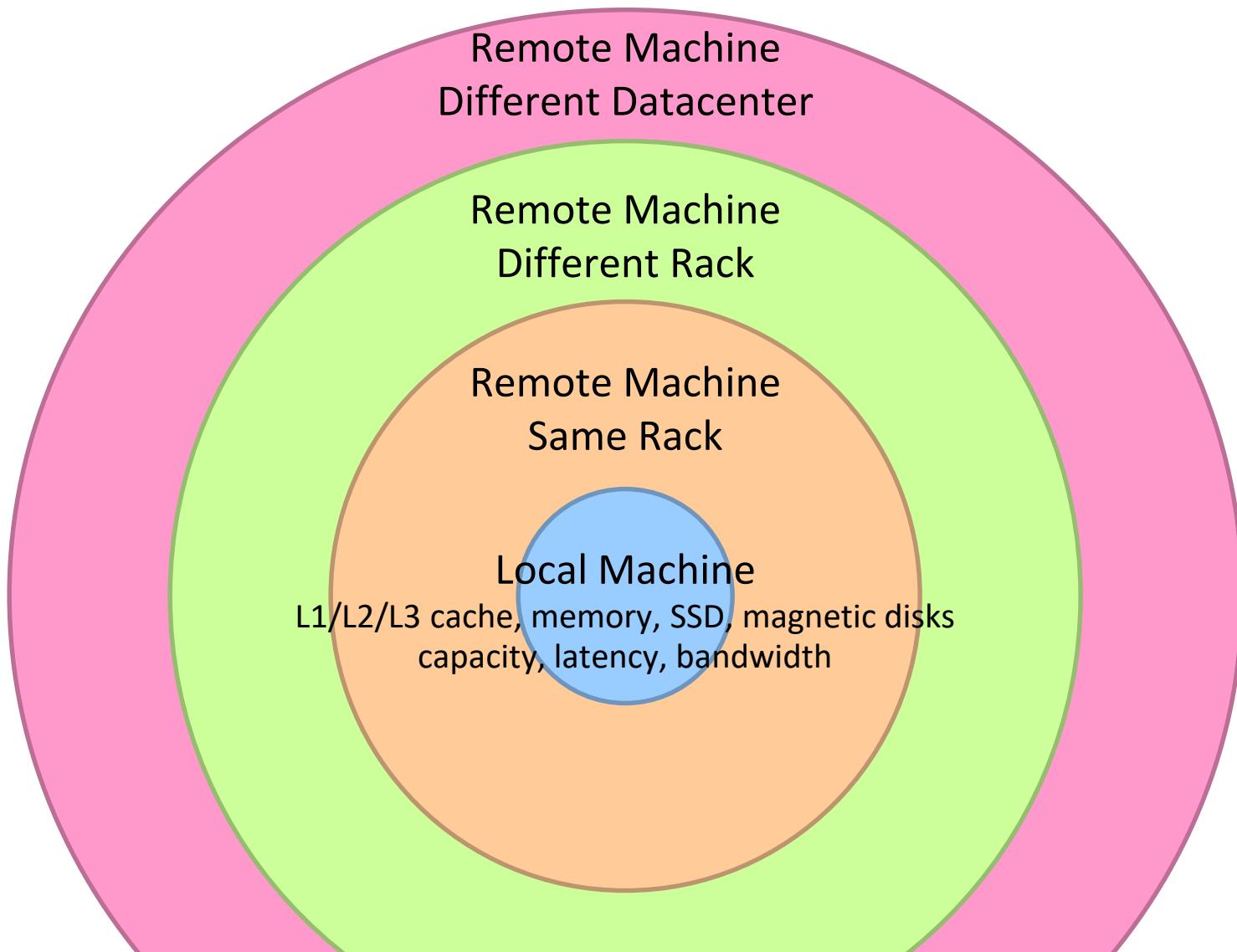
Now, what about SSDs?

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 |

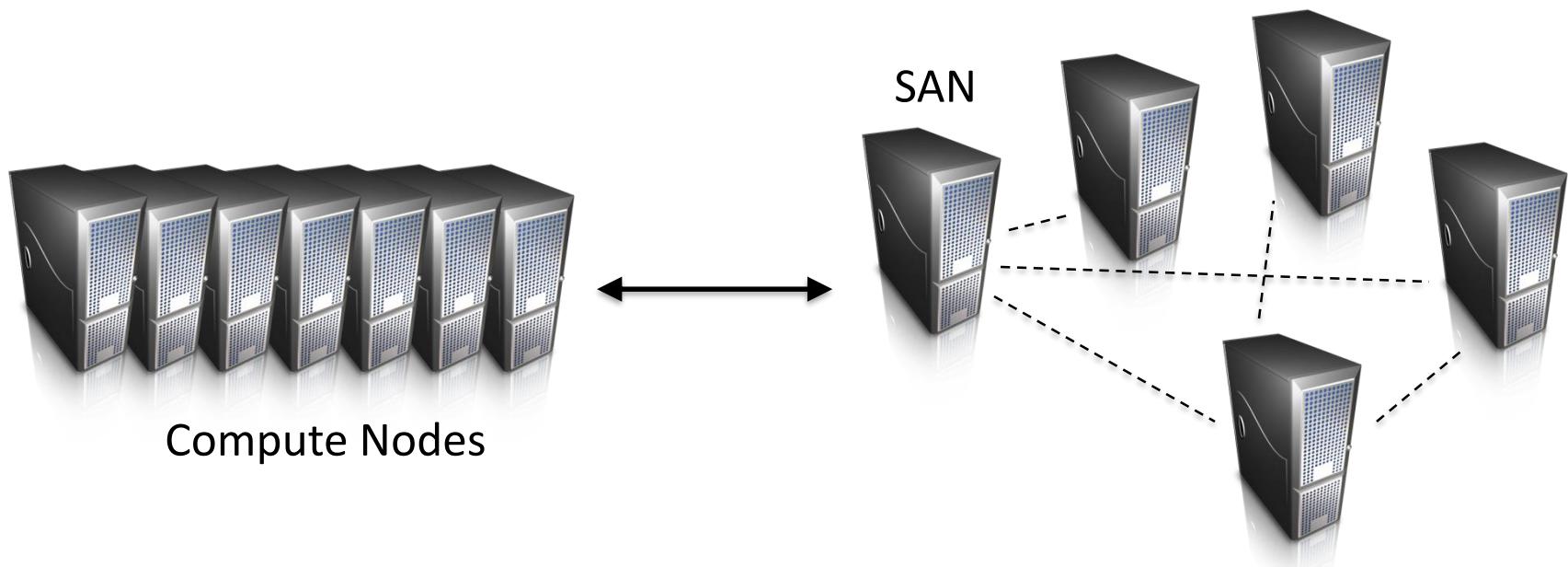


# Hadoop Cluster Architecture



# How do we get data to the workers?

Let's consider a typical supercomputer...



Sequoia will enable simulations that explore phenomena at a level of detail never before possible. Sequoia is dedicated to NNSA's Advanced Simulation and Computing (ASC) program for stewardship of the nation's nuclear weapons stockpile, a joint effort from LLNL, Los Alamos National Laboratory and Sandia National Laboratories.



# Sequoia

16.32 PFLOPS

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

1.6 petabytes of memory

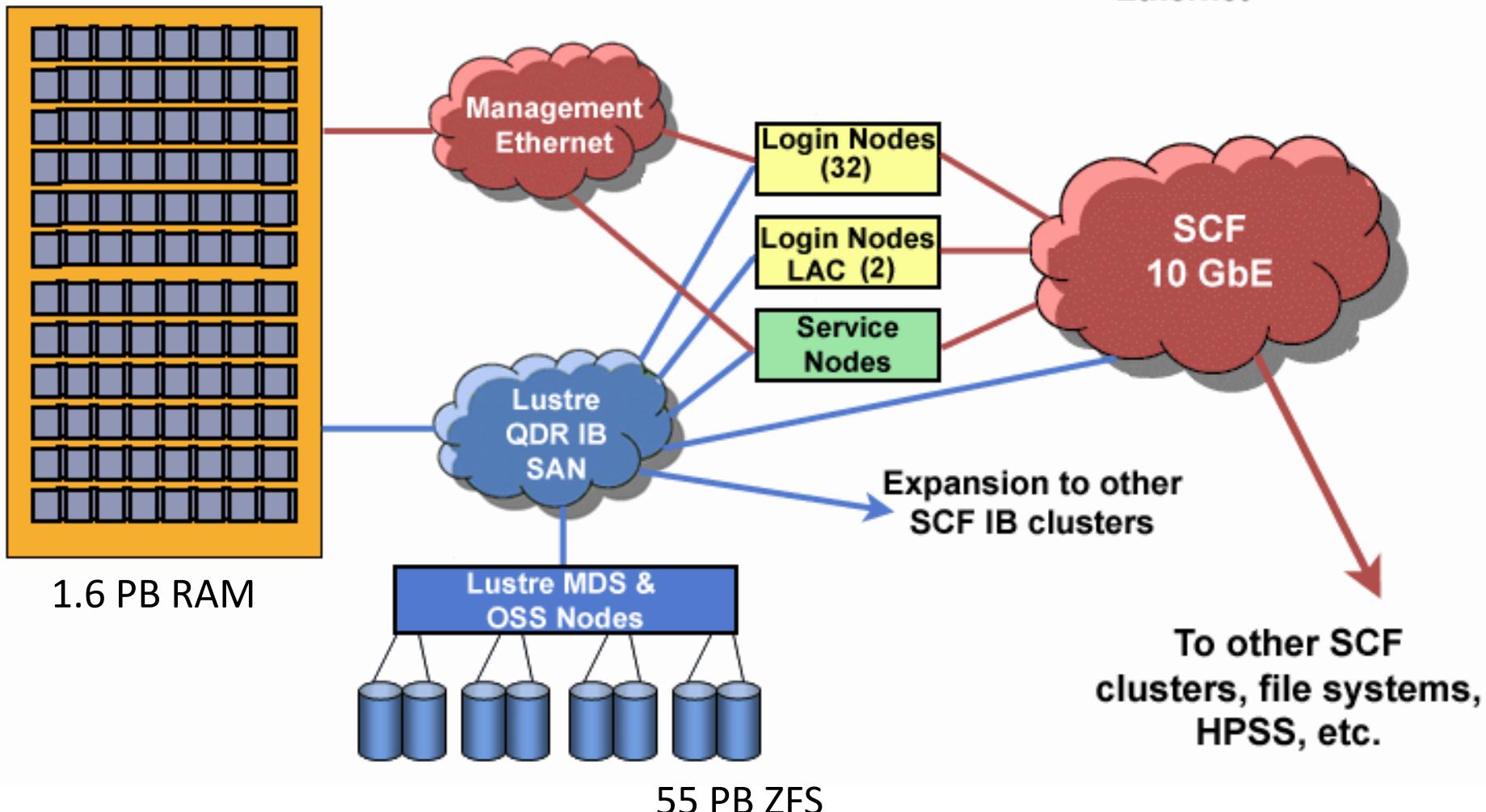
7.9 MWatts total power

Deployed in 2012, still #8 in TOP500 List (June 2018)

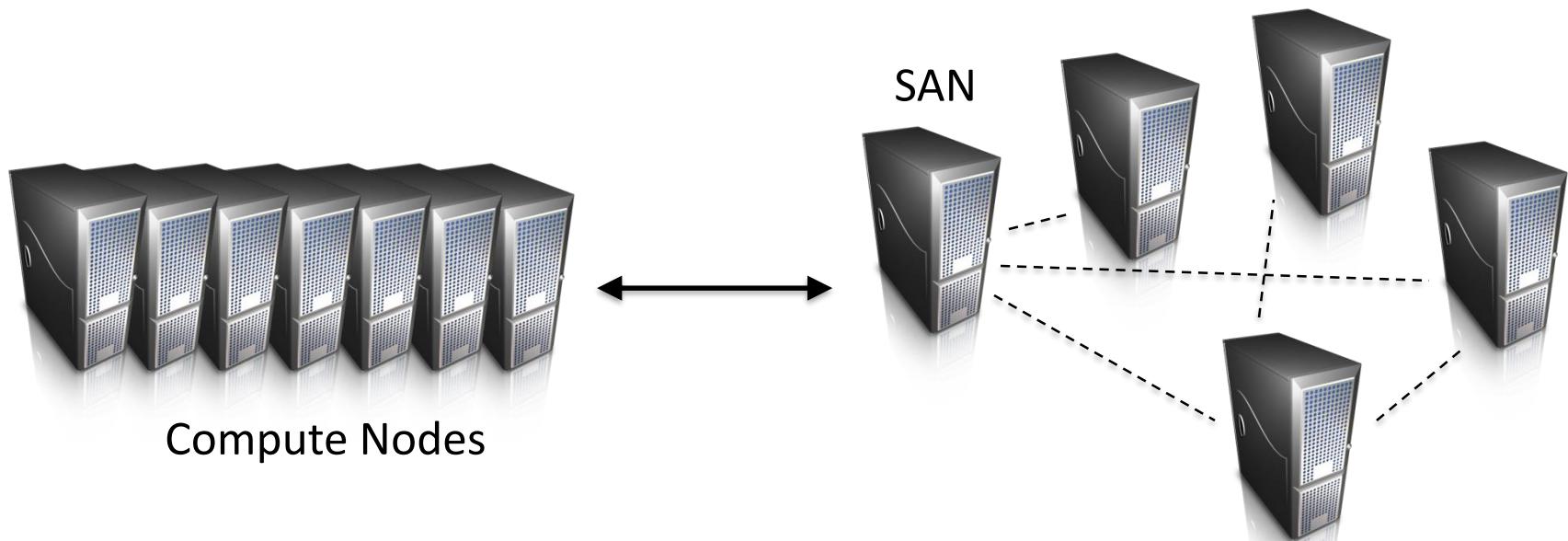
# Sequoia

96 racks (12x8)  
98,304 compute nodes  
768 I/O nodes

- BG/Q 5D Torus Fabric
- QDR Infiniband
- Ethernet



# Compute-Intensive vs. Data-Intensive



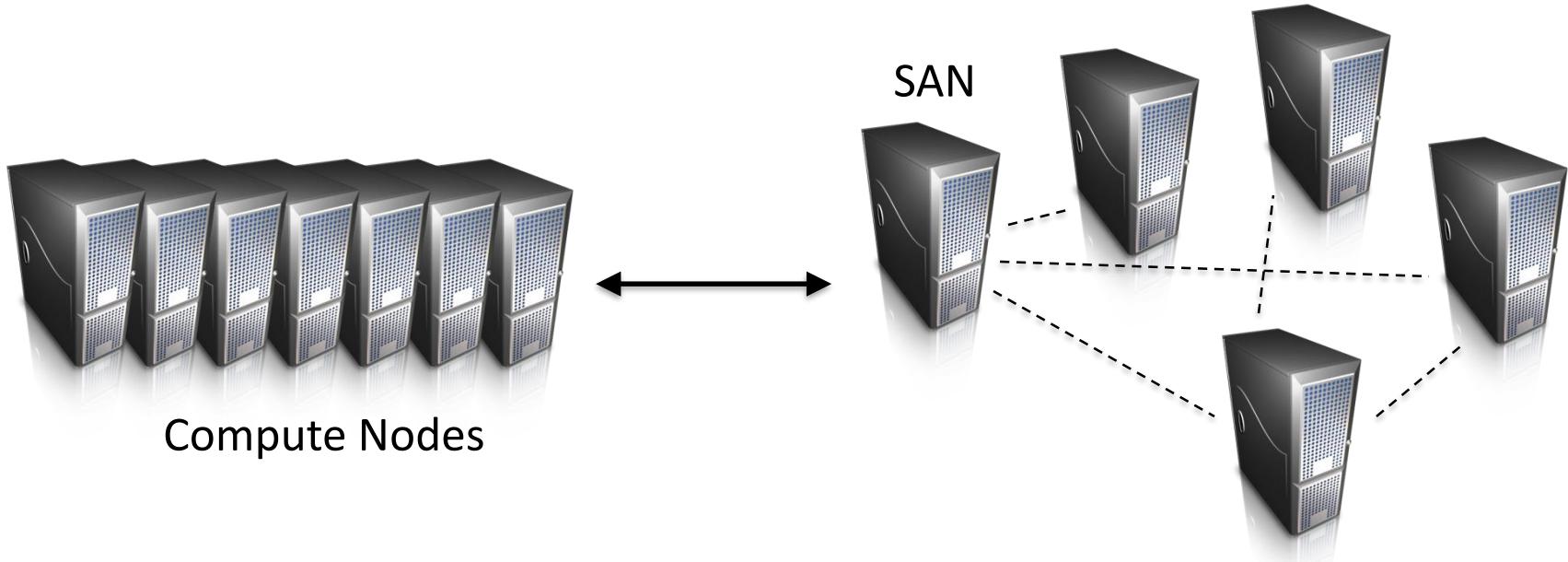
Why does this make sense for compute-intensive tasks?  
What's the issue for data-intensive tasks?

# 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: 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:

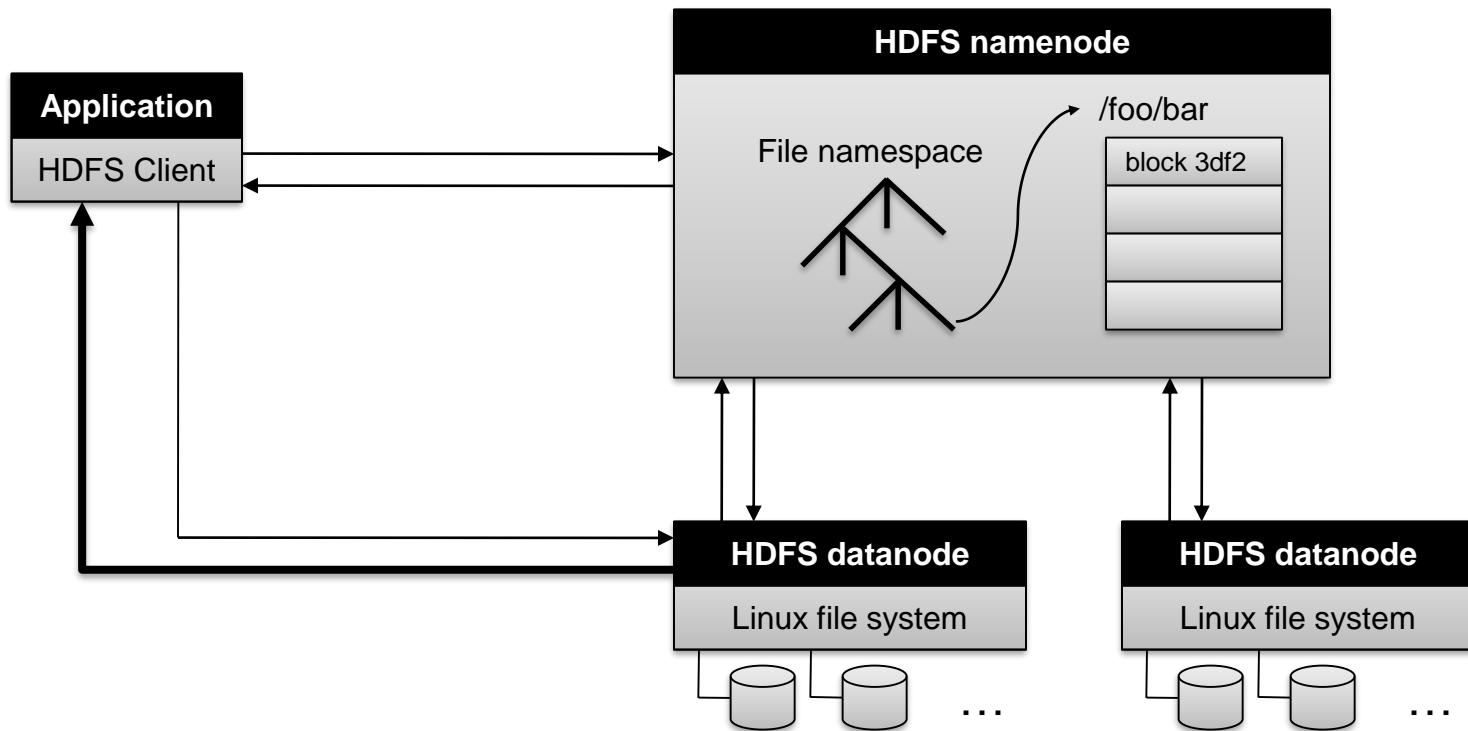
Different consistency model for file appends

Implementation language

Performance

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

# HDFS Architecture



# Namenode Responsibilities

Managing the file system namespace

Holds file/directory structure, file-to-block mapping,  
metadata (ownership, access permissions, etc.)

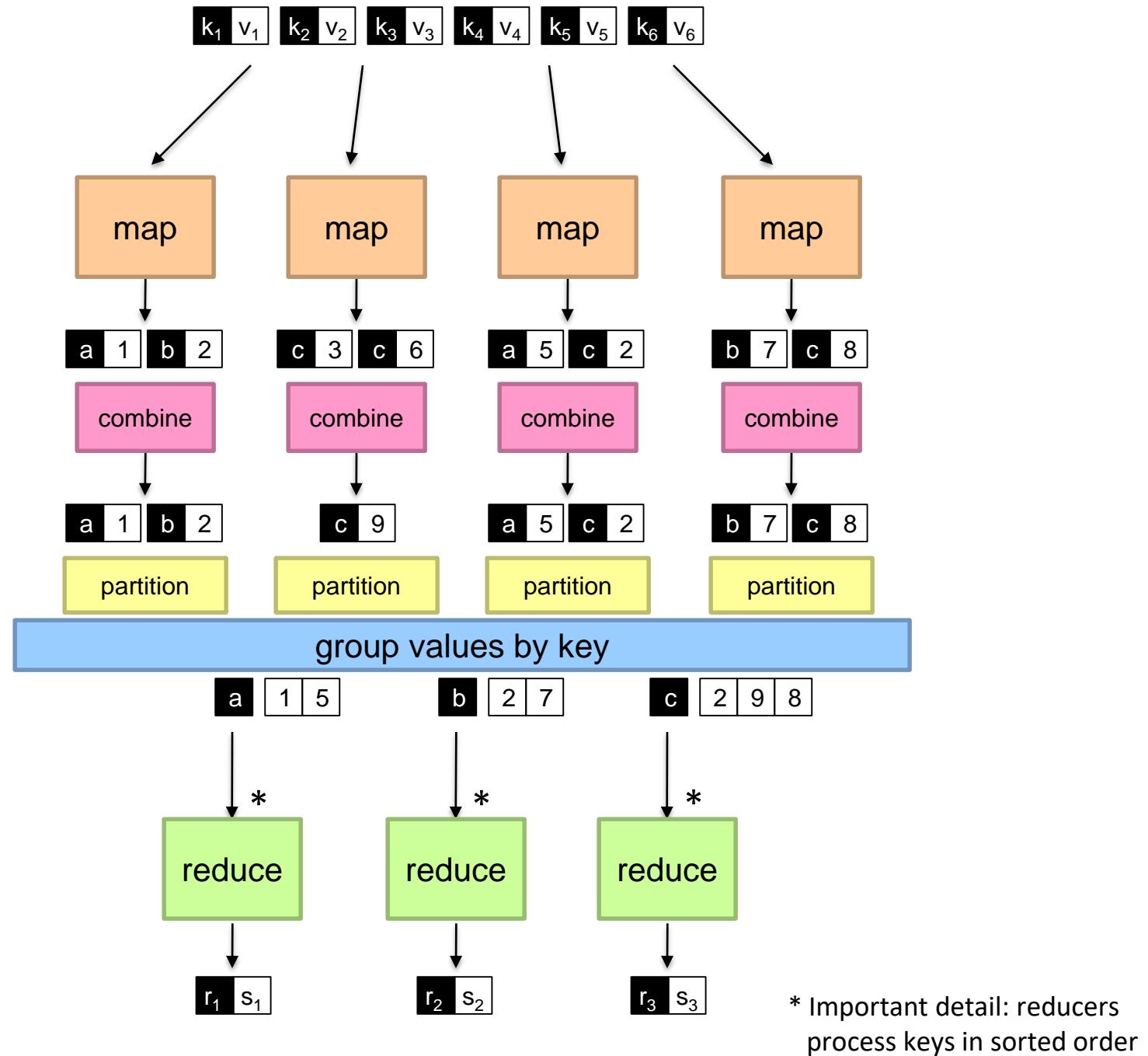
Coordinating file operations

Directs clients to datanodes for reads and writes  
No data is moved through the namenode

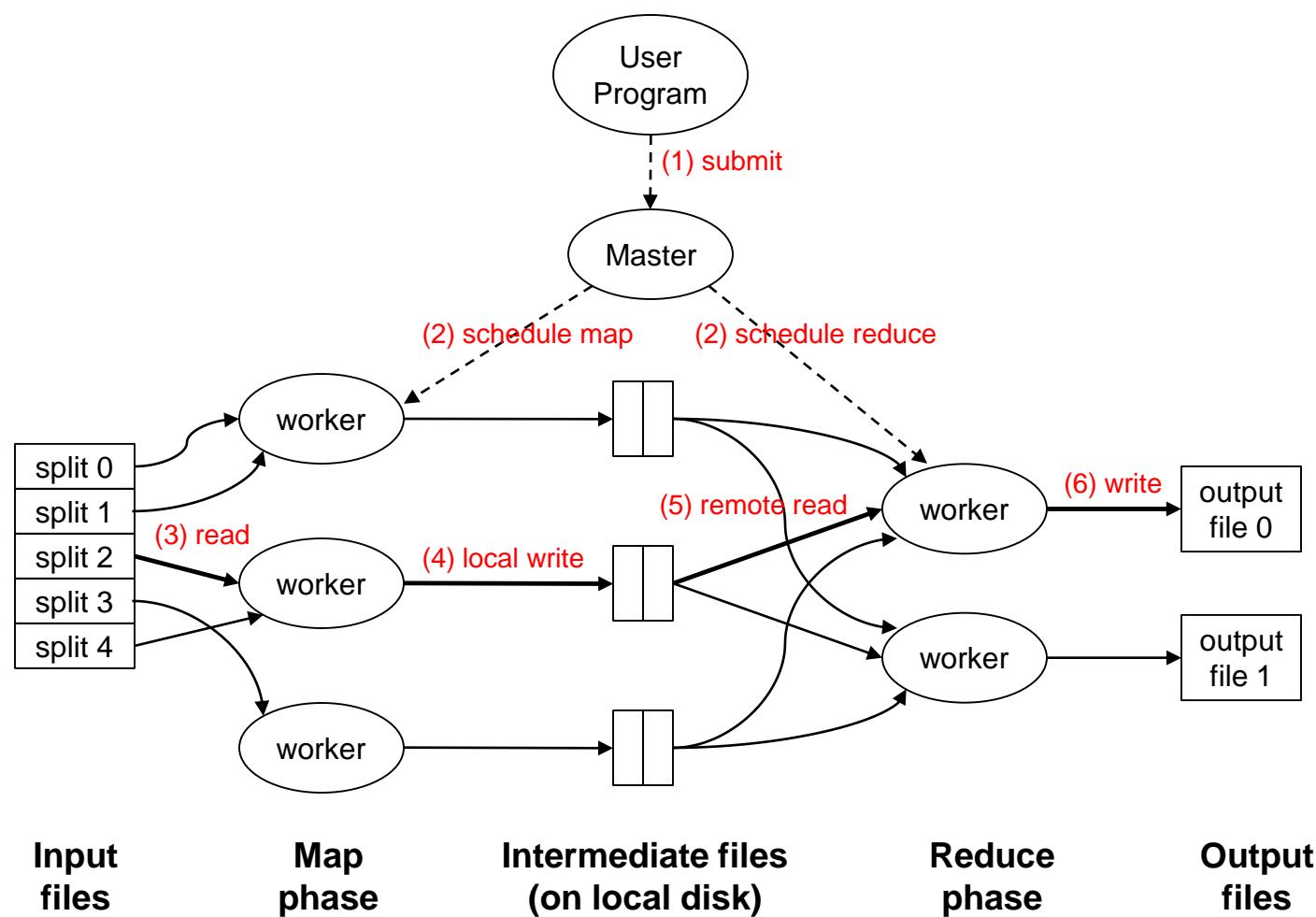
Maintaining overall health

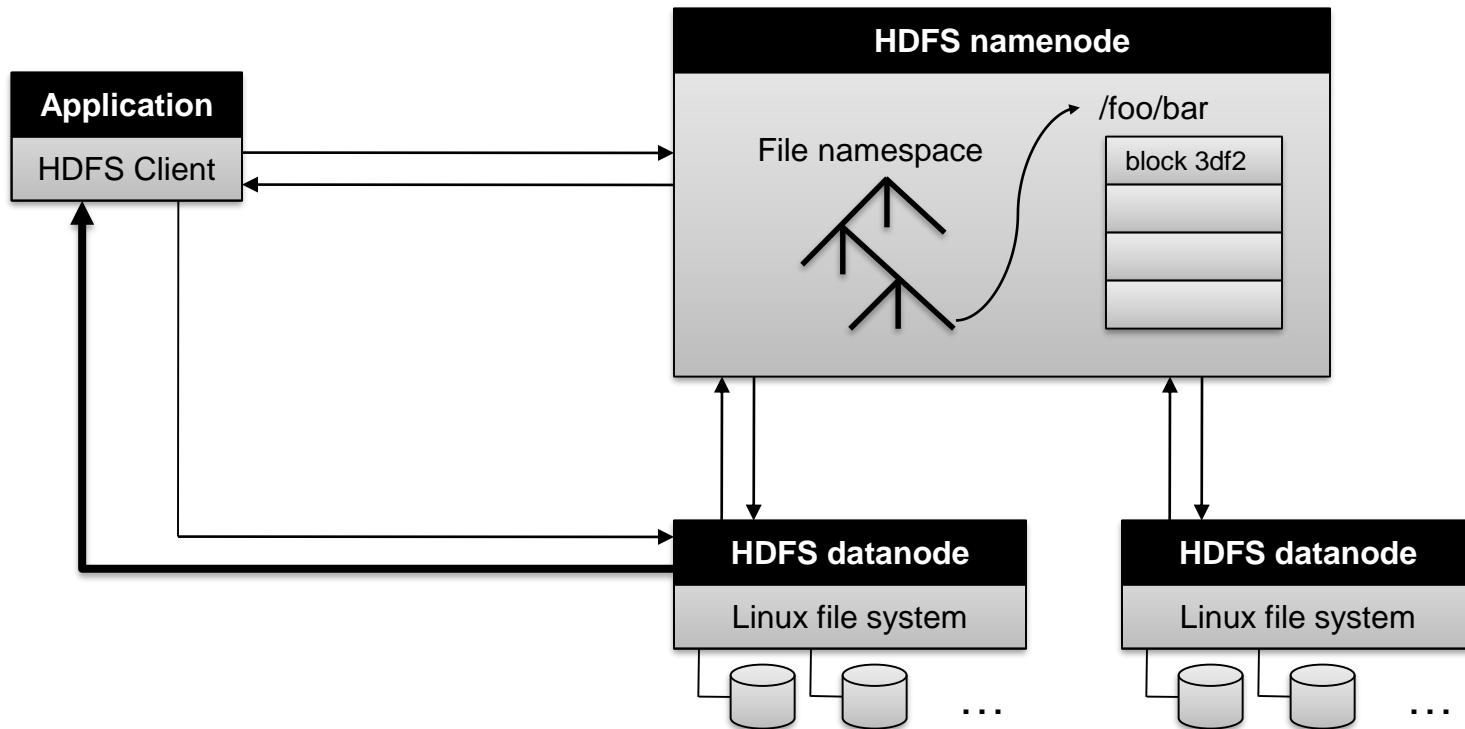
Periodic communication with the datanodes  
Block re-replication and rebalancing  
Garbage collection

# Logical View

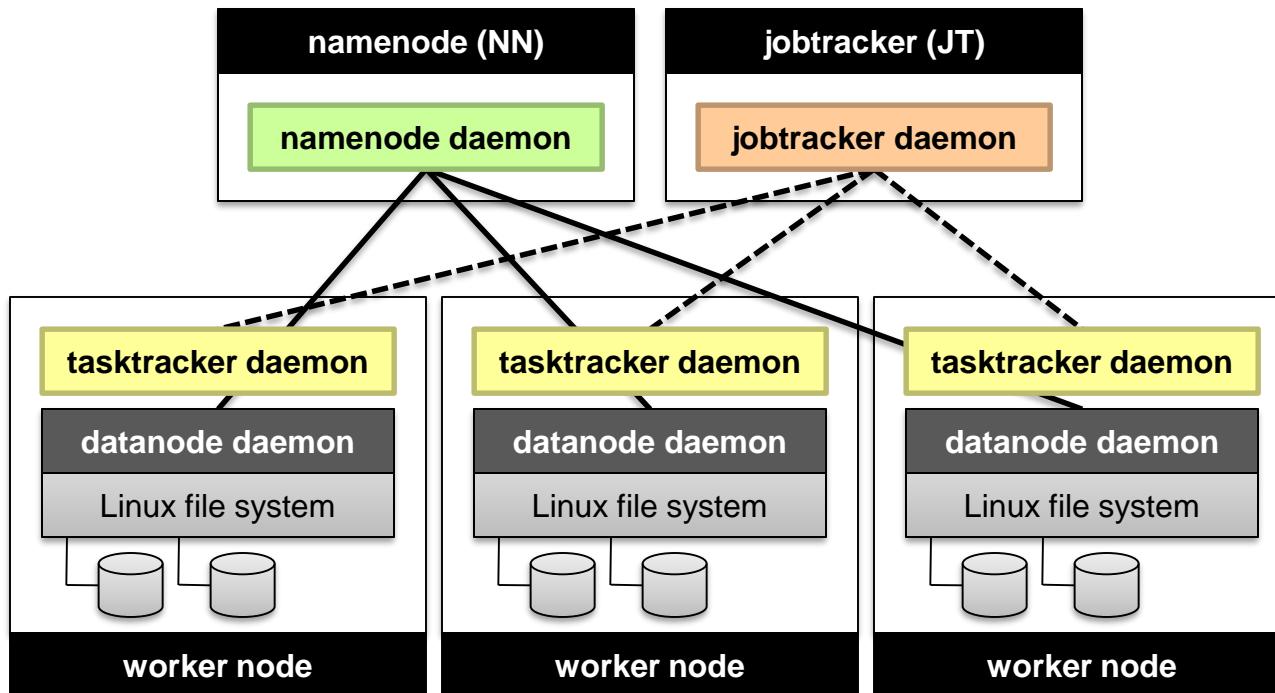


# Physical View





# Putting everything together...



# Basic Cluster Components\*

Namenode (NN)

Master for HDFS

Jobtracker (JT)

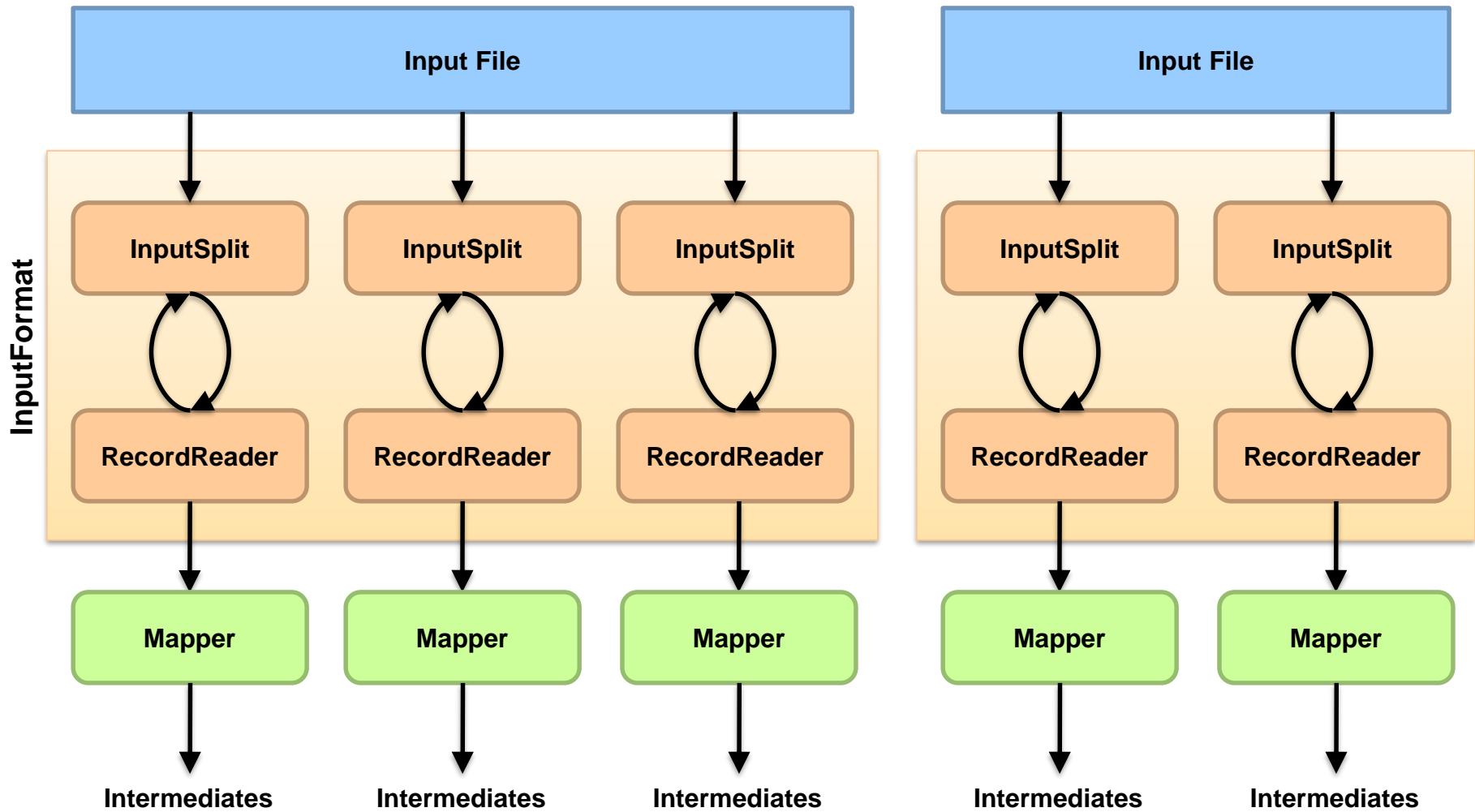
Coordinator for MapReduce jobs

On *each* of the worker machines:

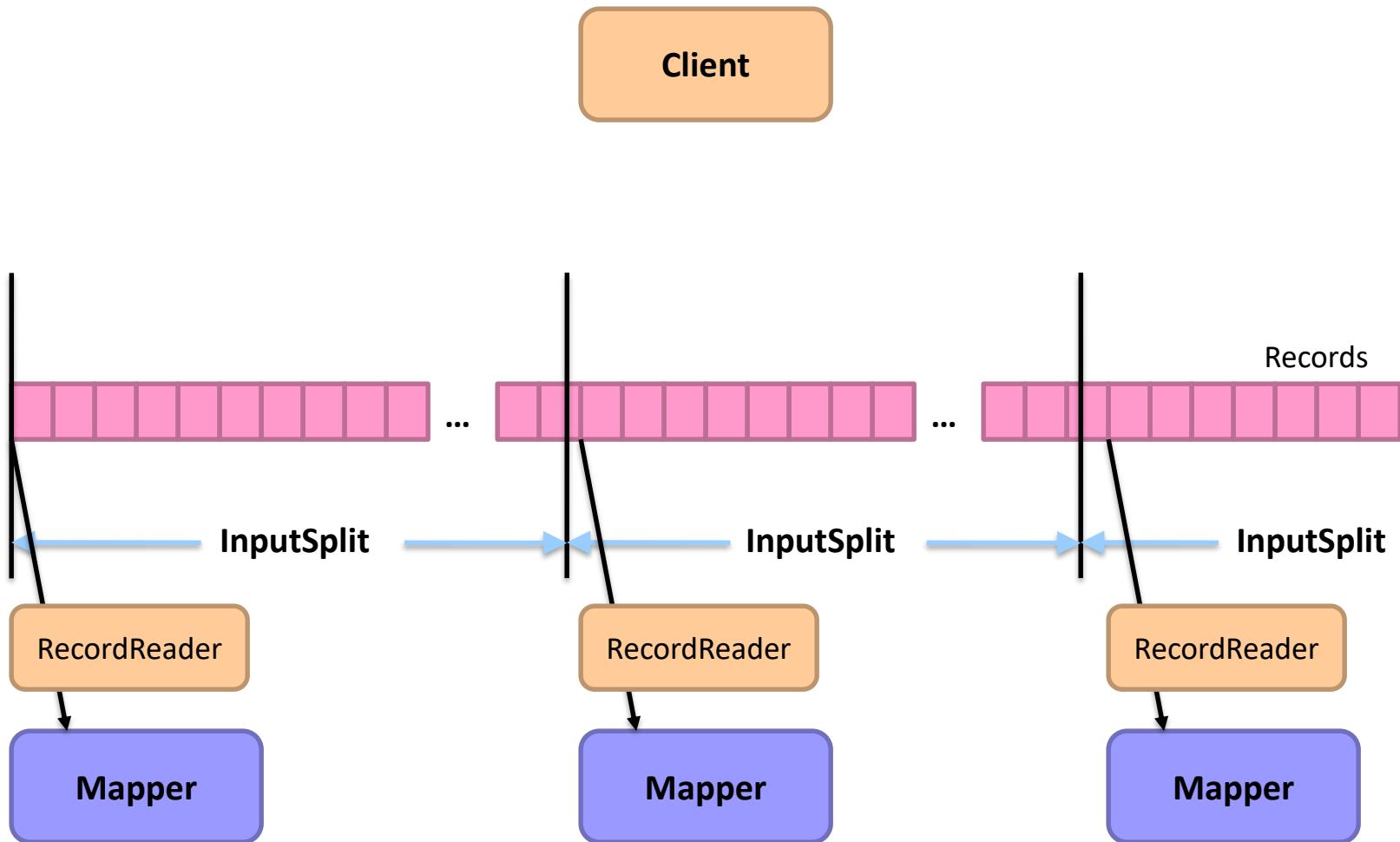
Tasktracker (TT): contains multiple task slots

Datanode (DN): serves HDFS data blocks

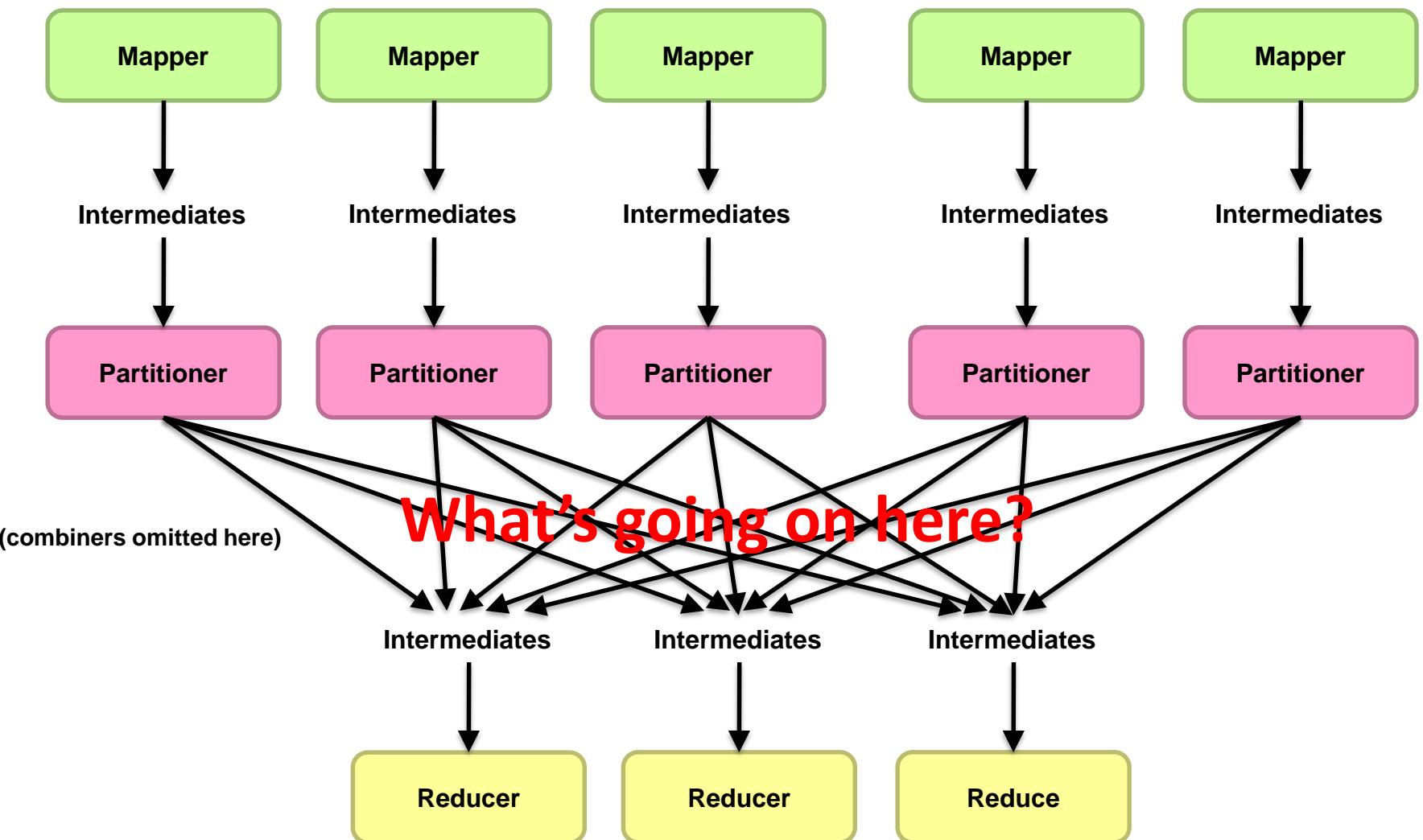
\* Not quite... leaving aside YARN for now



What are these input split?



What are these input split?



# Distributed Group By in MapReduce

## Map side

Map outputs are buffered in memory in a circular buffer

When buffer reaches threshold, contents are “spilled” to disk

Spills are merged into a single, partitioned file (sorted within each partition)

Combiner runs during the merges

## Reduce side

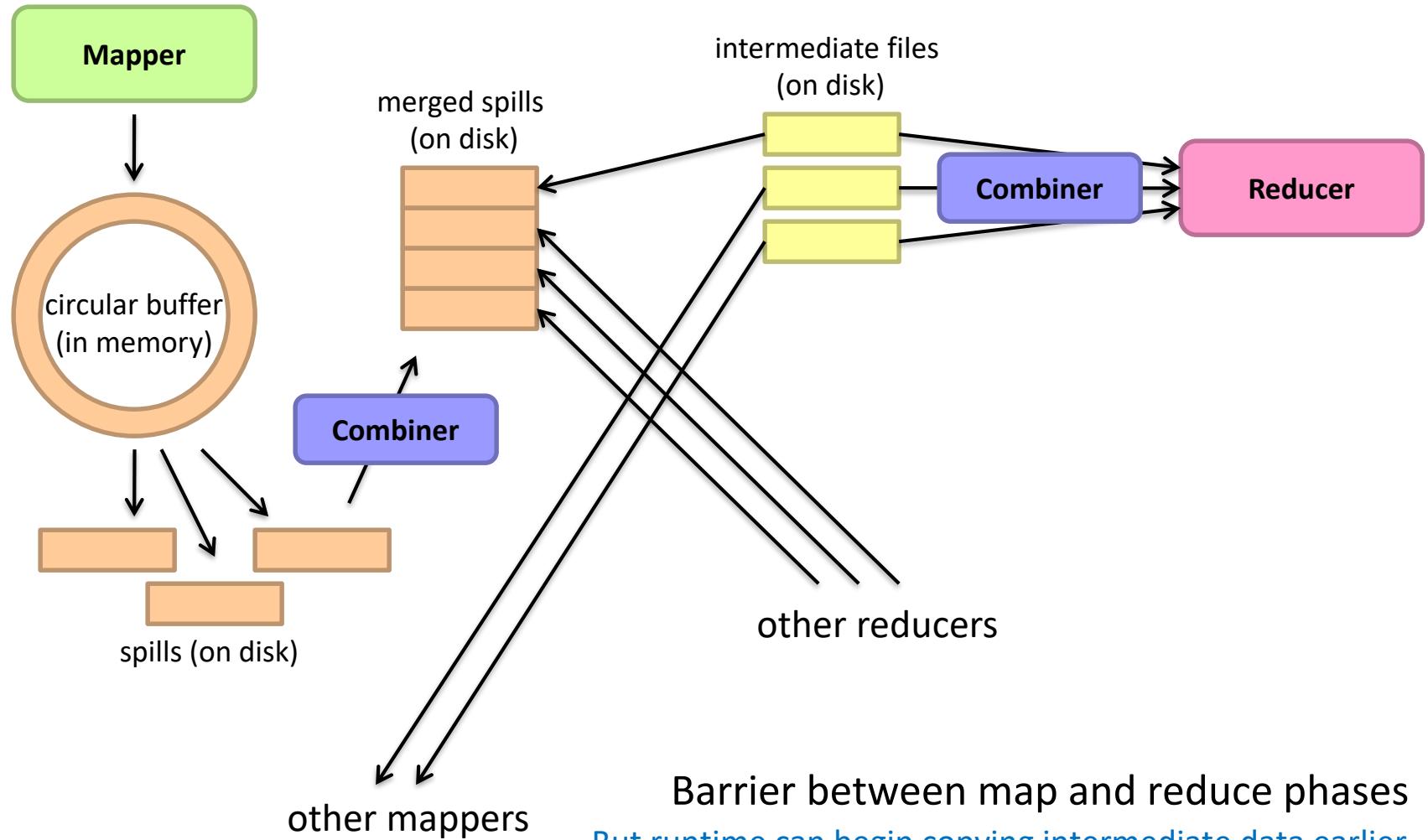
First, map outputs are copied over to reducer machine

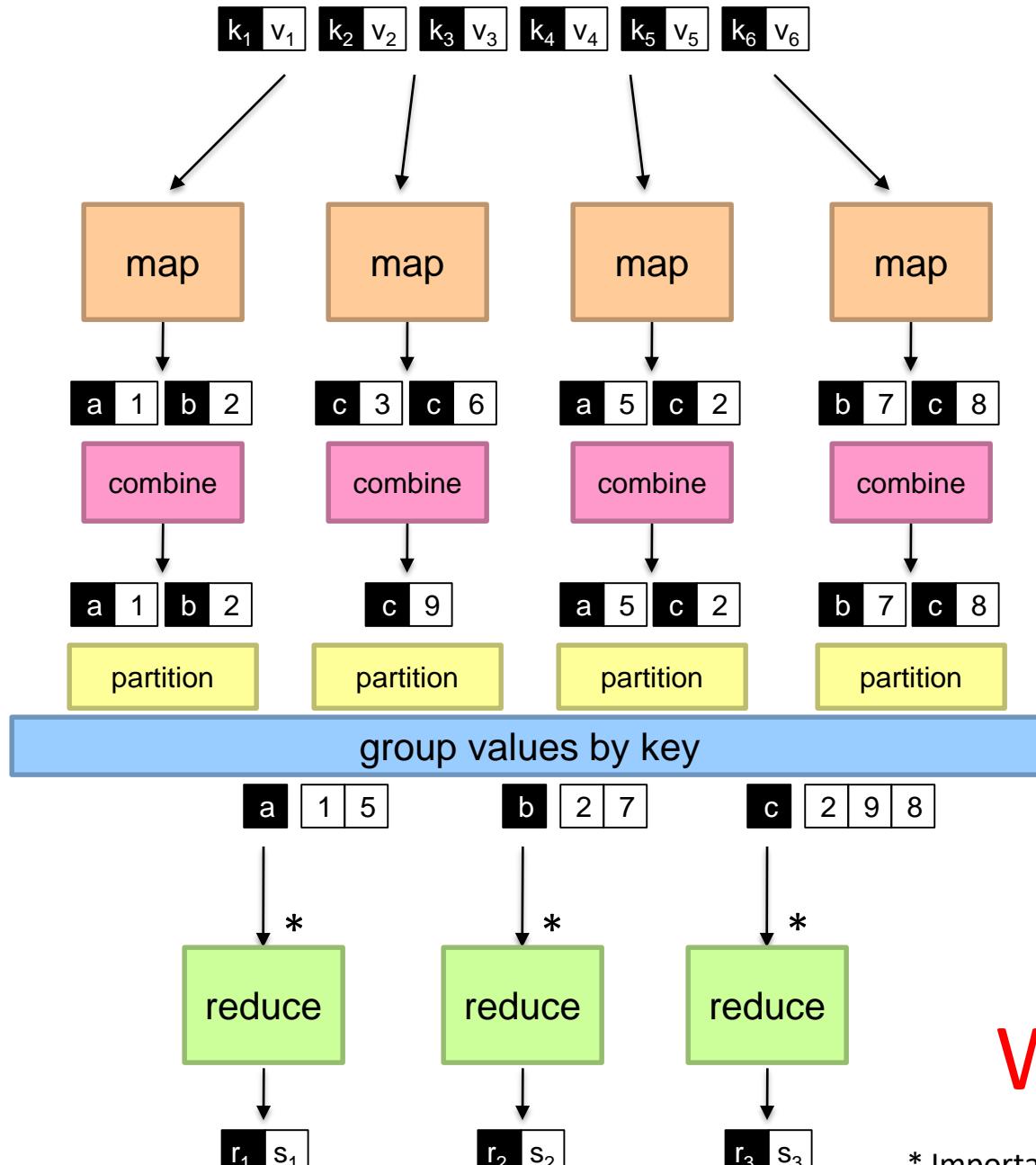
“Sort” is a multi-pass merge of map outputs (happens in memory and on disk)

Combiner runs during the merges

Final merge pass goes directly into reducer

# Distributed Group By in MapReduce





Why?

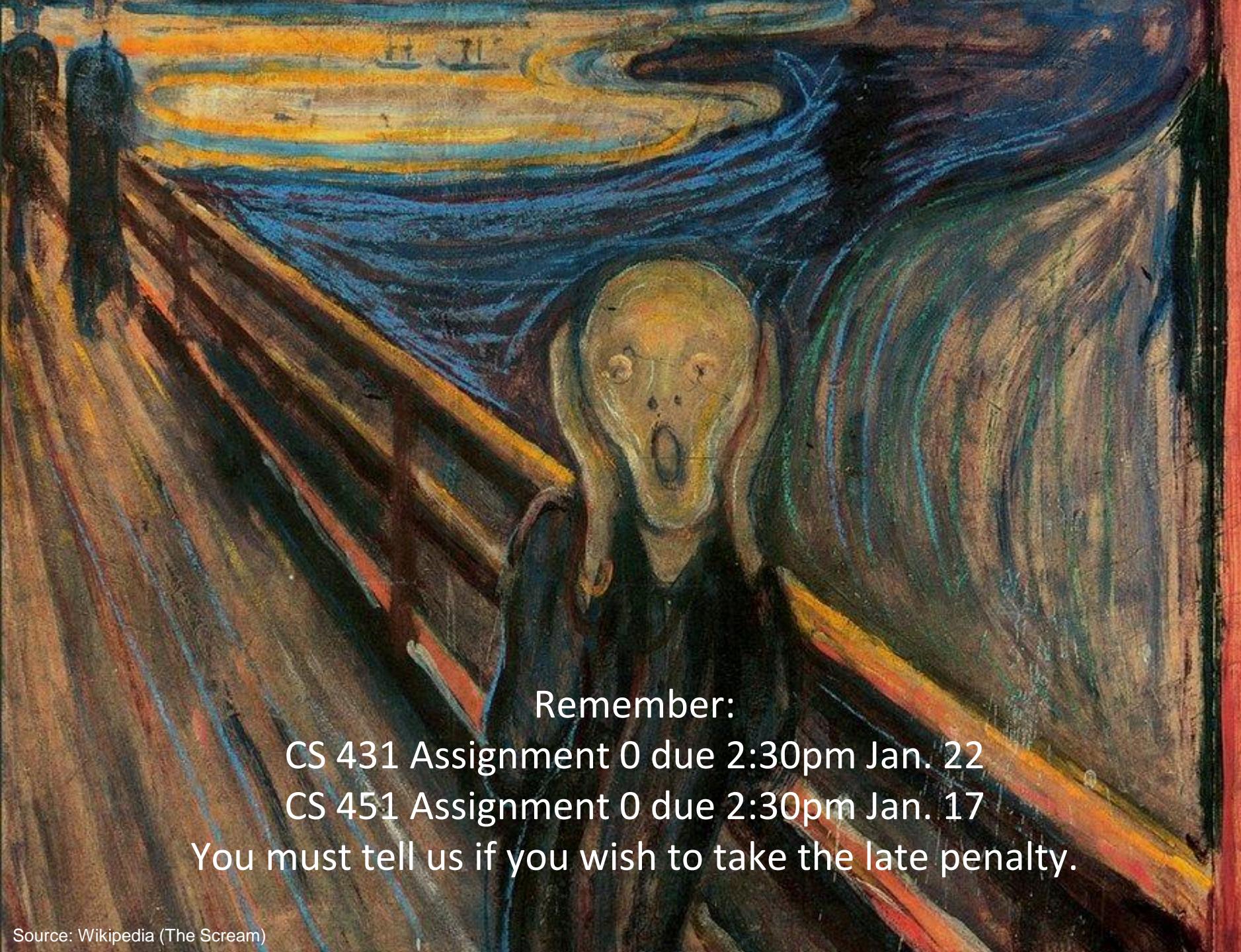
\* Important detail: reducers process keys in sorted order

# Law of Leaky Abstractions

All non-trivial abstractions, to some degree, are leaky.

Joel Spolsky

Remember logical vs. physical?



Remember:

CS 431 Assignment 0 due 2:30pm Jan. 22

CS 451 Assignment 0 due 2:30pm Jan. 17

You must tell us if you wish to take the late penalty.