L01: What's Massive Data?

ANLY 502: Massive Data Fundamentals

Simson Garfinkel & Ghaleb Abdulla

January 13, 2016

Please fill out survey at http://bit.ly/ANLY502-2016

GEORGETOWN UNIVERSITY

Welcome!

Q: What is "massive data?"

(think about it!)

(Please fill out survey at http://bit.ly/ANLY502-2016)

Overview

- ““Today's data scientists are commonly faced with huge data sets (Big Data) that may arrive at fantastic rates and in a broad variety of formats. This core course addresses the resulting challenges to data professionals. The course will introduce students to the advantages and limitations of distributed computing and to methods of assessing its impact. Techniques for parallel processing (MapReduce) and their implementation (Hadoop) will be covered, as well as techniques for accessing unstructured data and for handling streaming data. These techniques will be applied to real world examples, using clusters of computational cores and cloud computing. Prerequisite: Good command of R or Python, some knowledge of data structures. Three credits”

Spring 2016 • Mon 6:30 — 9:00 (except tonight, which is Wed.)

This is a new class!

- This is our first time teaching at Georgetown
- This class is designed to be forward-looking and research-focused

Before we get started, please fill out the class survey:

- bit.ly/ANLY502-2016

Introducing your teachers.

Simson L. Garfinkel, Ph.D.
National Institute of Standards and Technology*
https://simson.net/
simsong@acm.org

Interests: Security, Privacy, Digital Forensics
L01–L08

Ghaleb M. Abdulla, Ph.D.
Lawrence Livermore National Laboratory
http://people.llnl.gov/abdulla1
abdulla1@llnl.gov

Interests: networking, WWW caching, information organization
L09–L12

*Institutional affiliation is provided for identification purposes only.

Outline for today’s class

Introduction to ANLY 502

- Course introduction, policies and outline
- What you need to succeed in ANLY 502
- Information about labs and Amazon

Massive Data and the end of “Moore’s law”

- Where will tomorrow’s computing speed increases come from?

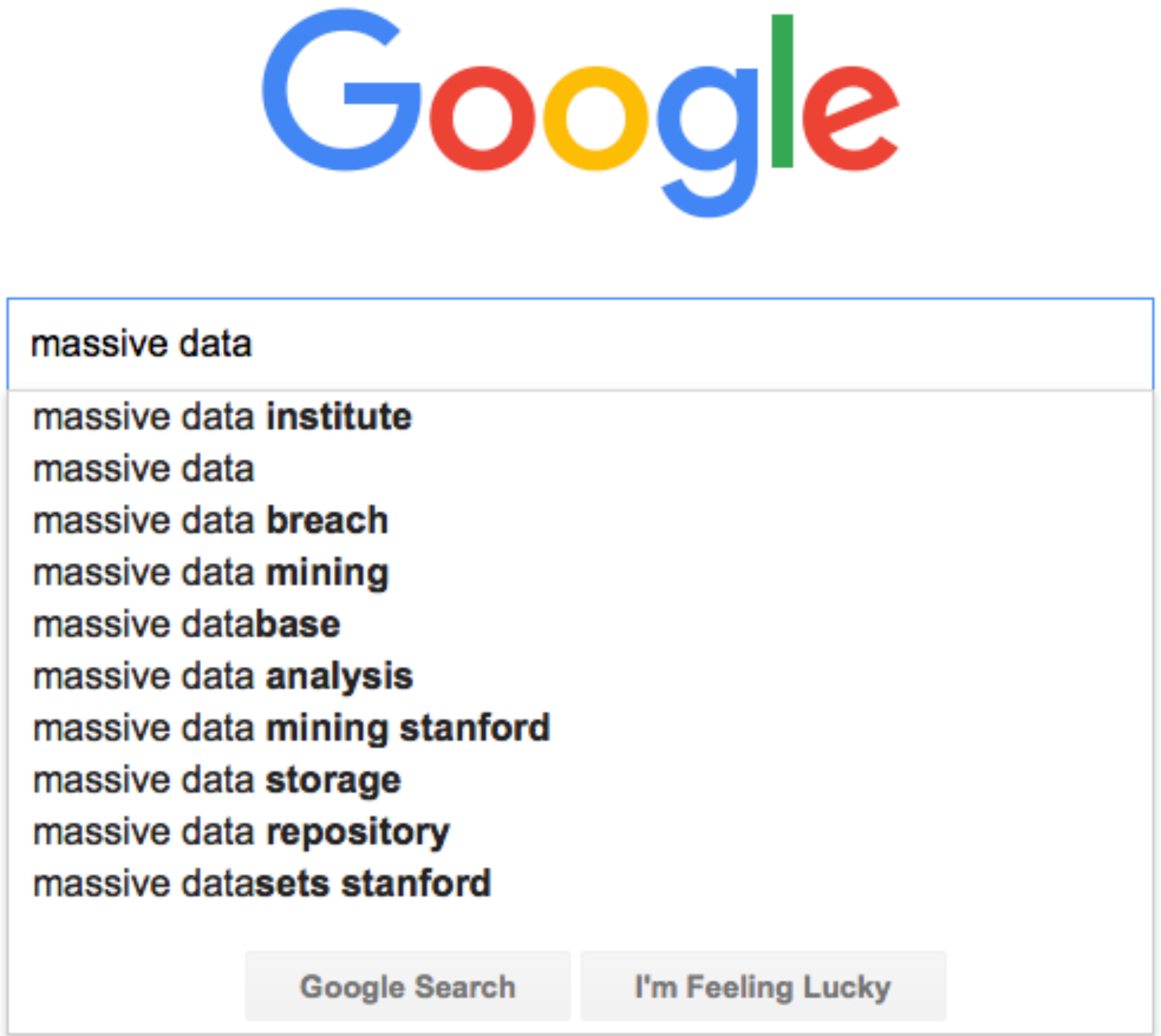
Introducing Hadoop and MapReduce

Setting up your laptop:

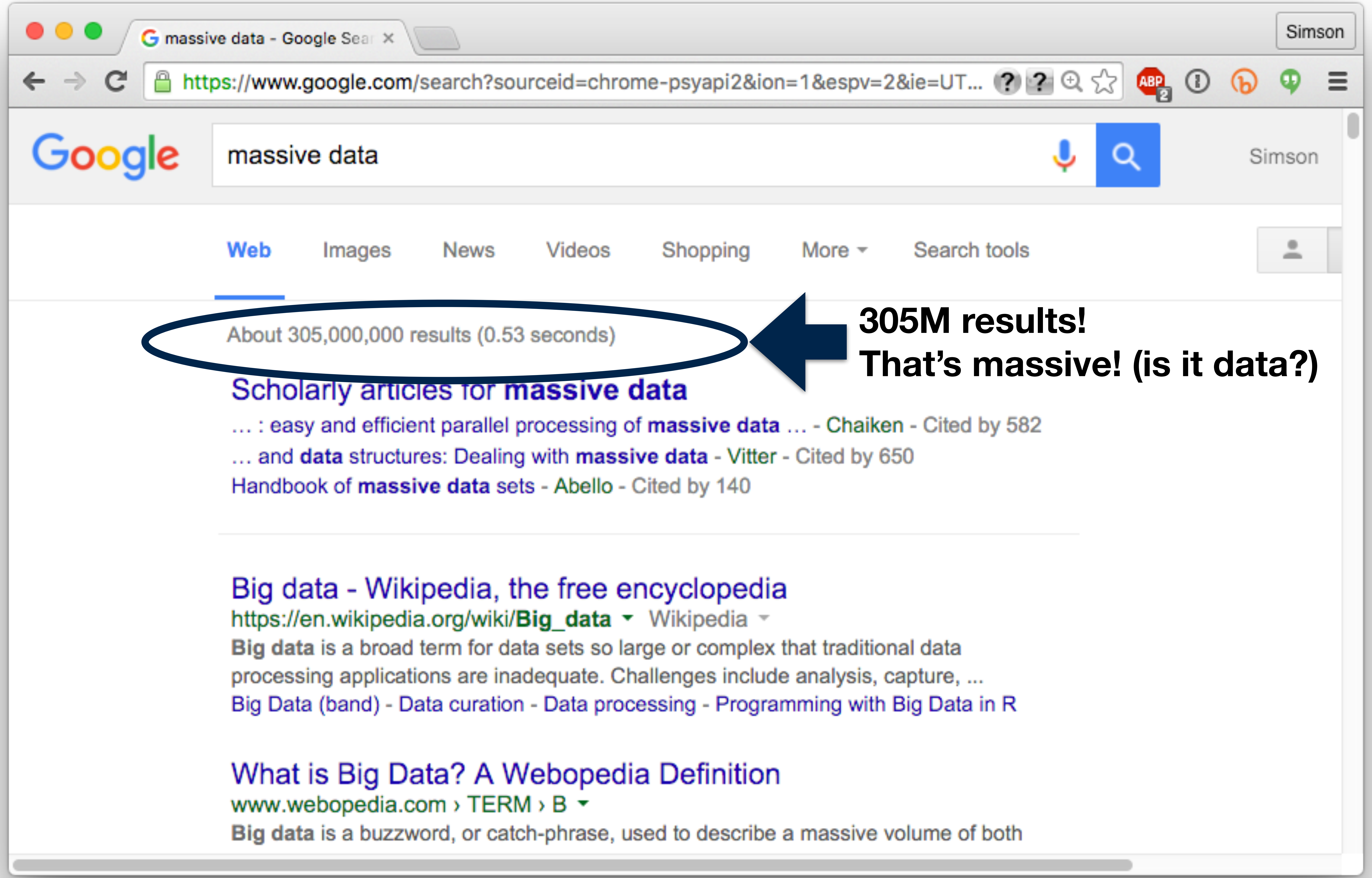
So what is “massive data?”

So what is massive data?

Let's ask Google:

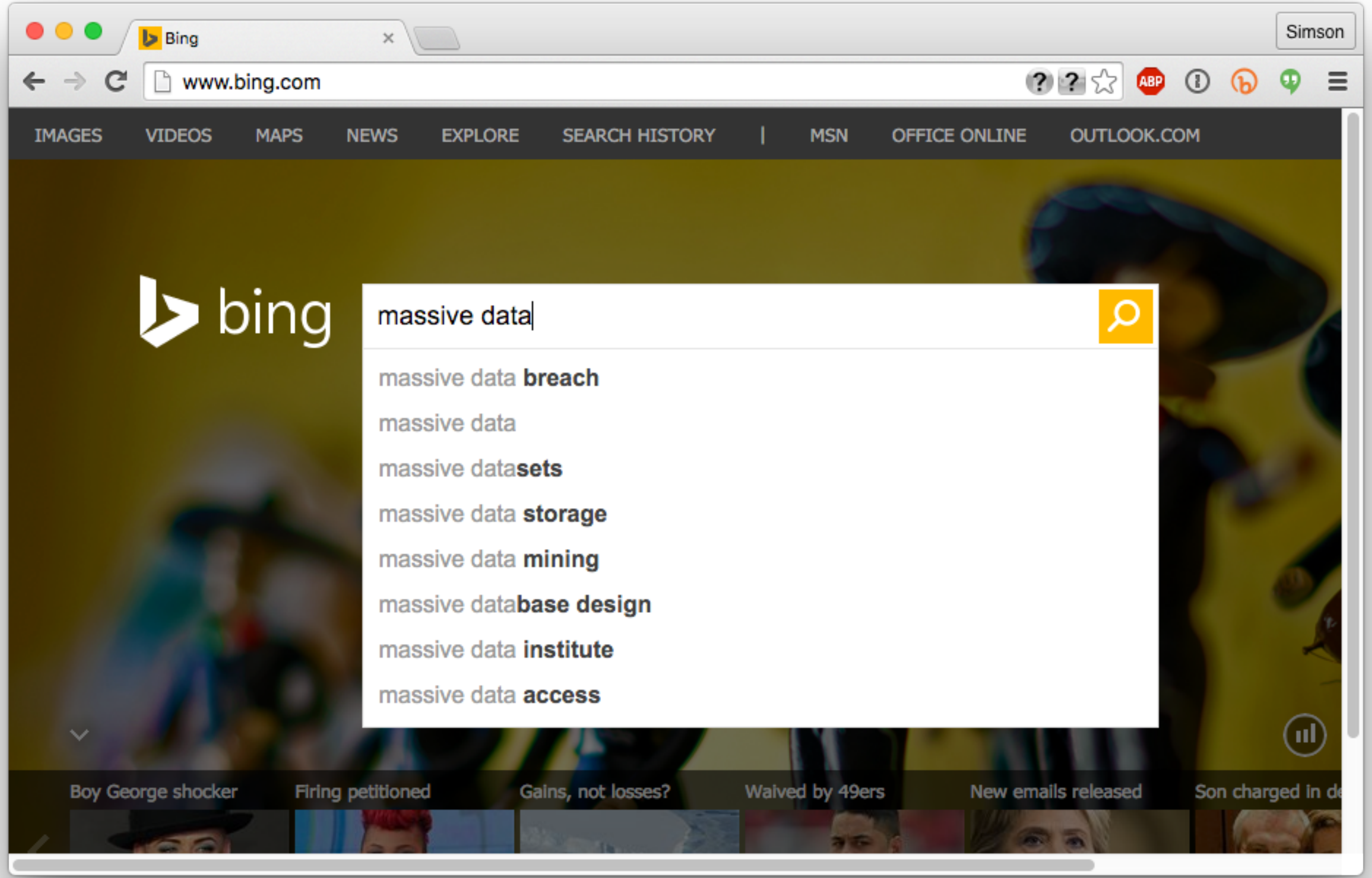


Google's view of massive data:

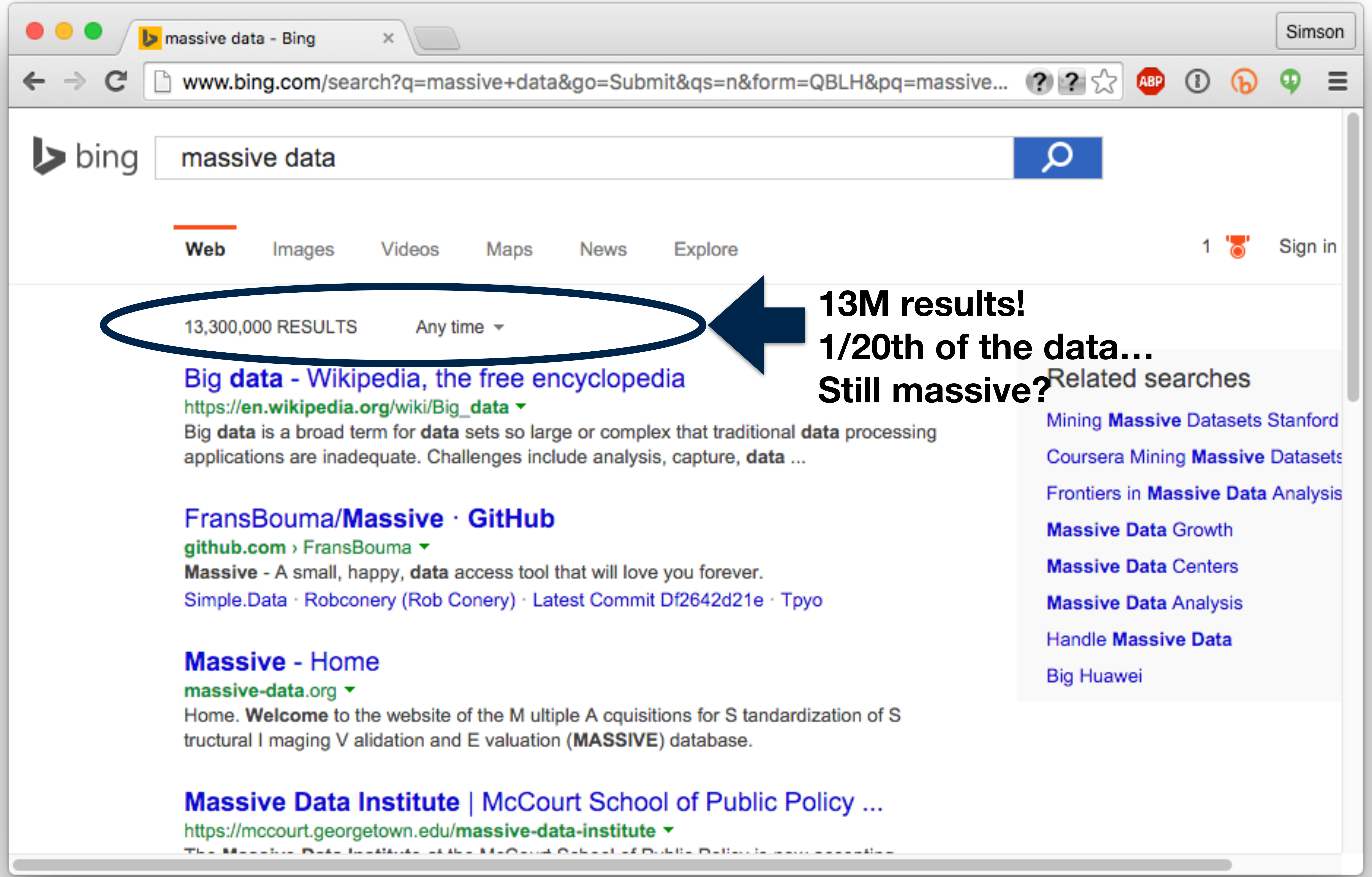


305M results!
That's massive! (is it data?)

Bing's view of massive data



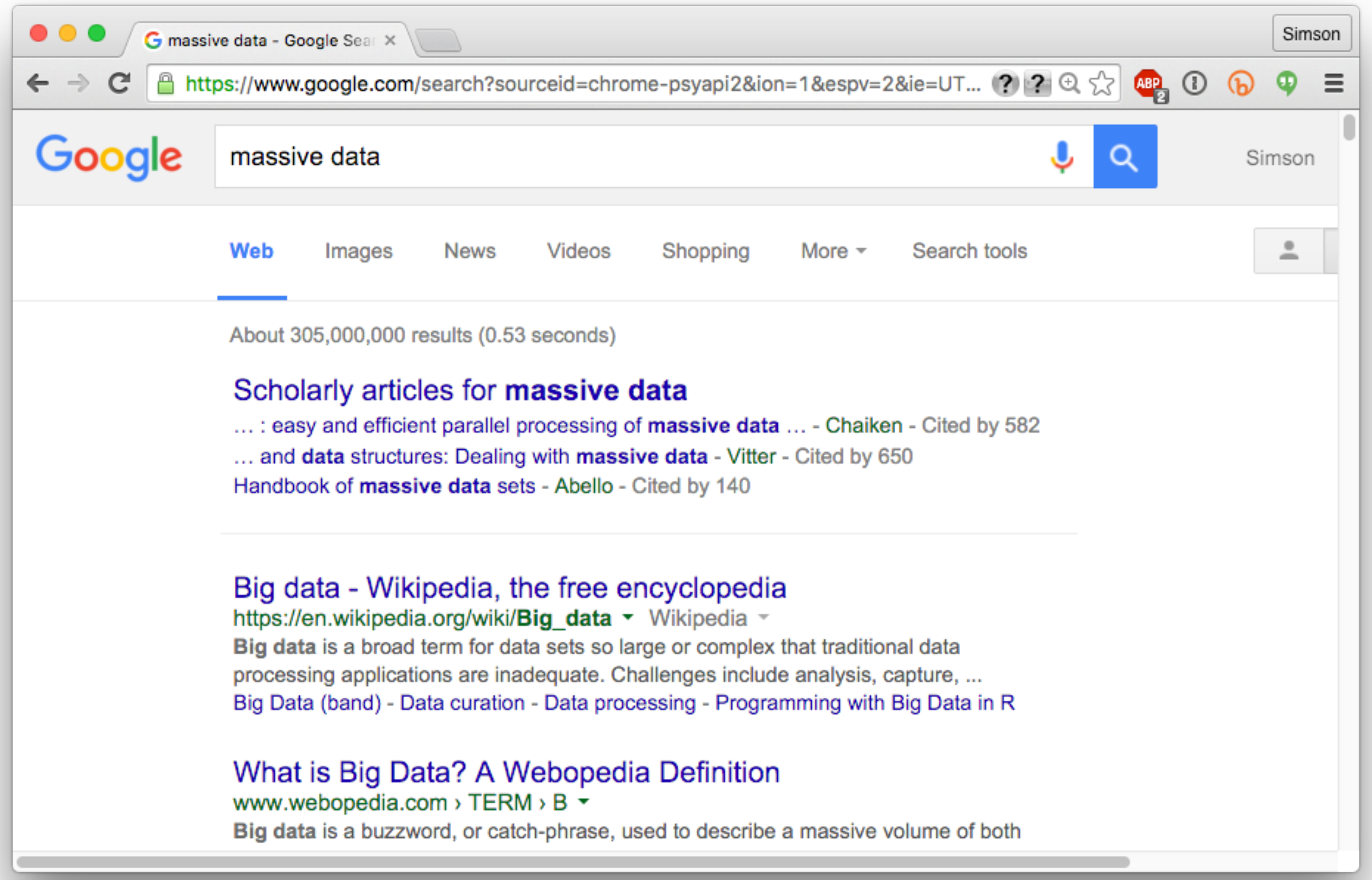
Bing: What's different?



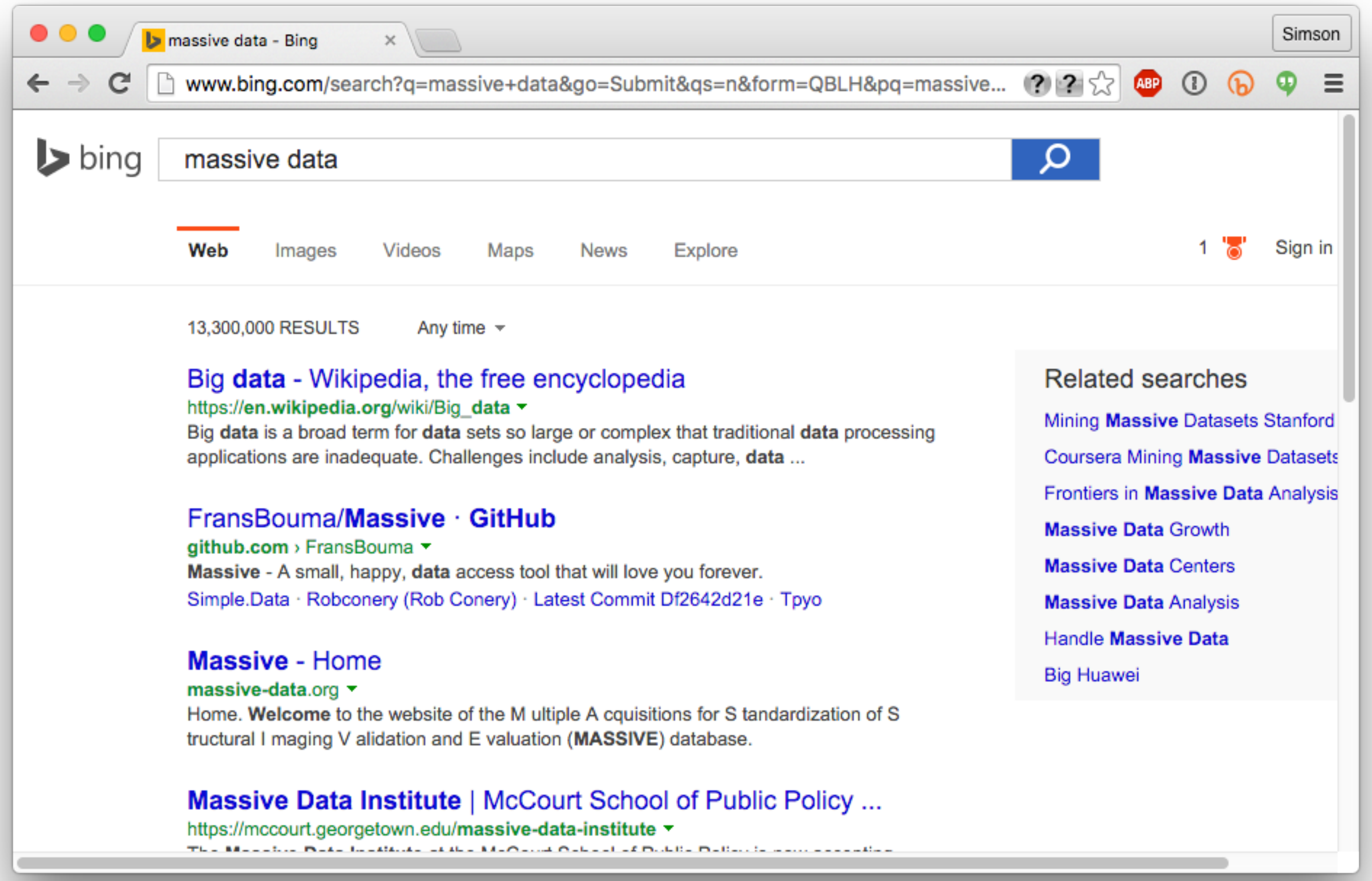
13M results!
1/20th of the data…
Still massive?

These search results depend upon massive data.

305M results



13M results



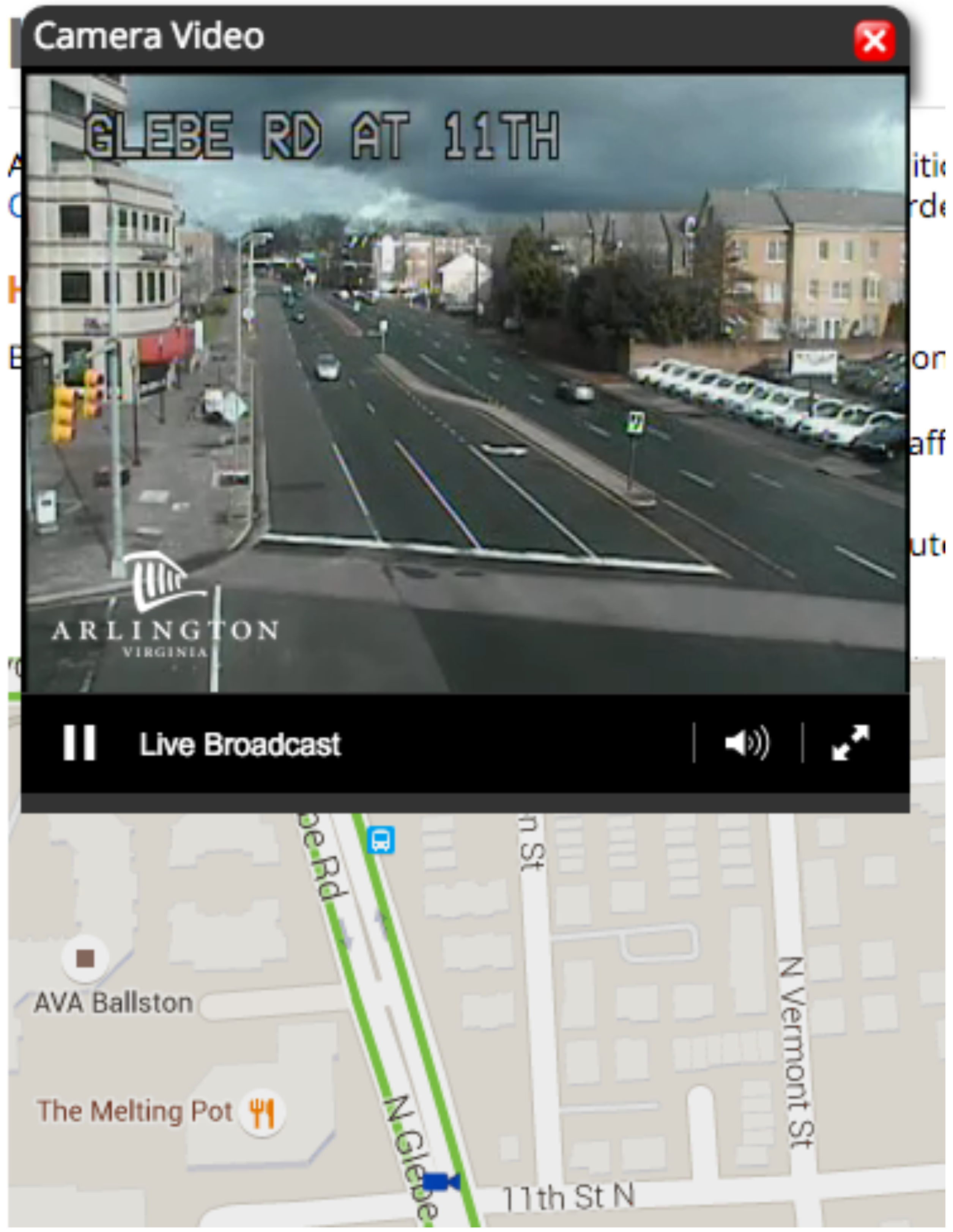
To make these results:

- Scan and index billions of web pages.
- Find all of the pages about "massive data". (What does the word "about" mean?)
- Eliminate "spam" pages.
- Group similar pages.
- Perform search of index with billions of entries in less than a second.

Q: Which results "better?"

Other examples of “massive data” — Real Time Traffic

Old approach to traffic: Traffic cameras and induction loops

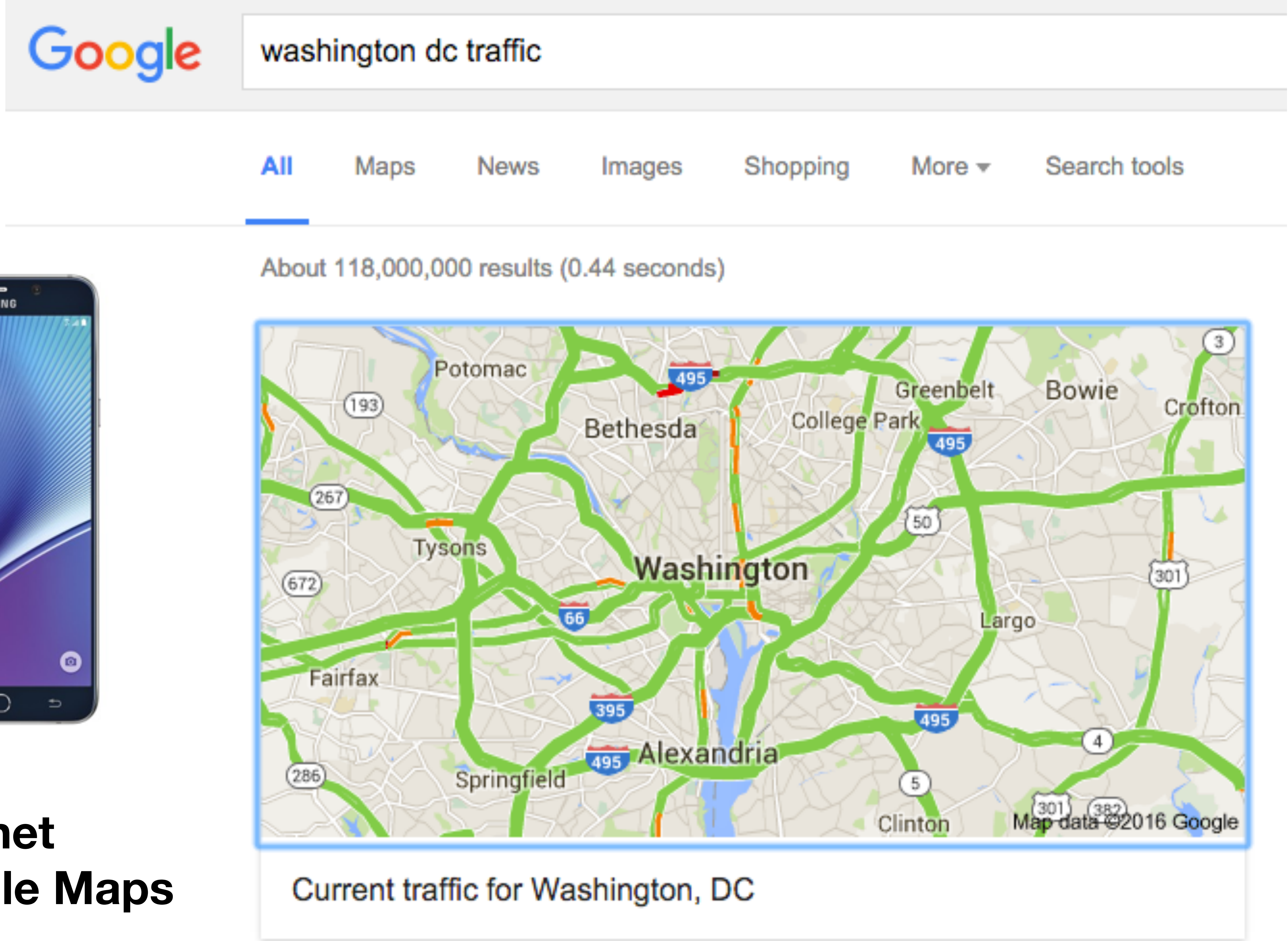


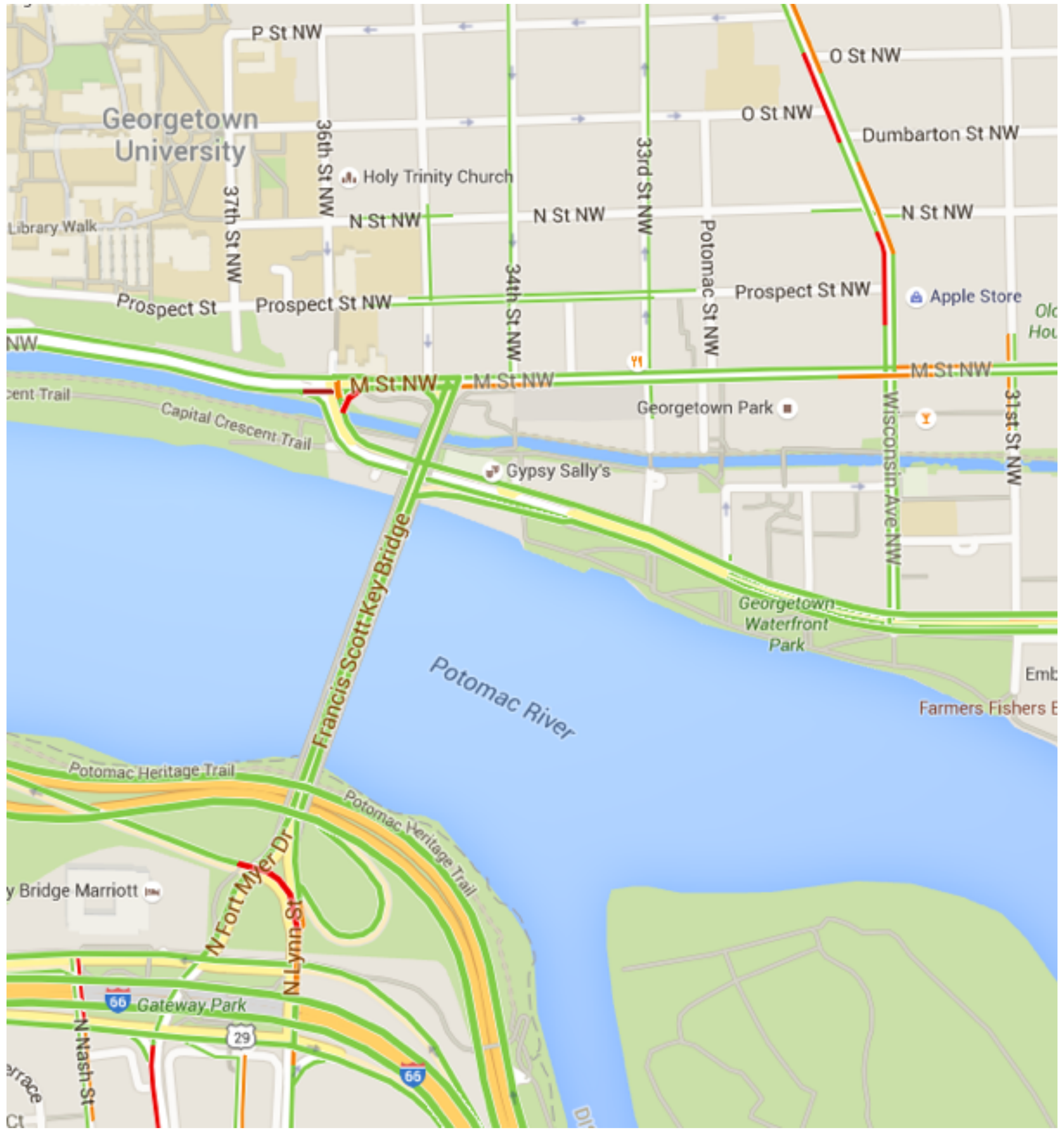
https://en.wikipedia.org/wiki/Induction_loop

Other examples of "massive data" — Real Time Traffic

New Approach: Cell Phones as Sensors

GPS
Internet
Google Maps





Street Level Detail

Massive data creates the potential for massive privacy problems

https://en.wikipedia.org/wiki/Traffic_enforcement_camera

http://www.cleveland.com/roadrant/index.ssf/2010/11/voters_oust_traffic_cameras_in.html

“Voters oust traffic cameras…”
Cleveland Plain Dealer
Sept. 7, 2010

Super-camera to catch 50 times more drivers

NEW traffic cameras which catch the most minor offences are set to be introduced across the country.

By OLLIE GILLMAN
PUBLISHED: 00:01, Mon, Sep 23, 2013

SHARE TWEET 101 16

The new cameras will detect even minor errors, such as straying into a bus lane

http://www.express.co.uk/news/uk/431426/Super-camera-to-catch-50-times-more-drivers

Google Flu Trends —

Big idea: Predict the Flu using Google search queries

- *Detecting influenza epidemics using search engine query data, Jeremy Ginsberg, Matthew H. Mohebbi, Rajan S. Patel, , Lynnette Brammer, Mark S. Smolinski, Larry Brilliant, Nature Vol 457, 19 February 2009*
- *5 Google Authors, 1 CDC Author*

- Hypothesis: People search for their symptoms when they are sick
- Claim: Model correlated with CDC-reported influenza-like illness (ILI).
  - *Prediction was 1-2 weeks earlier than CDC surveillance system*

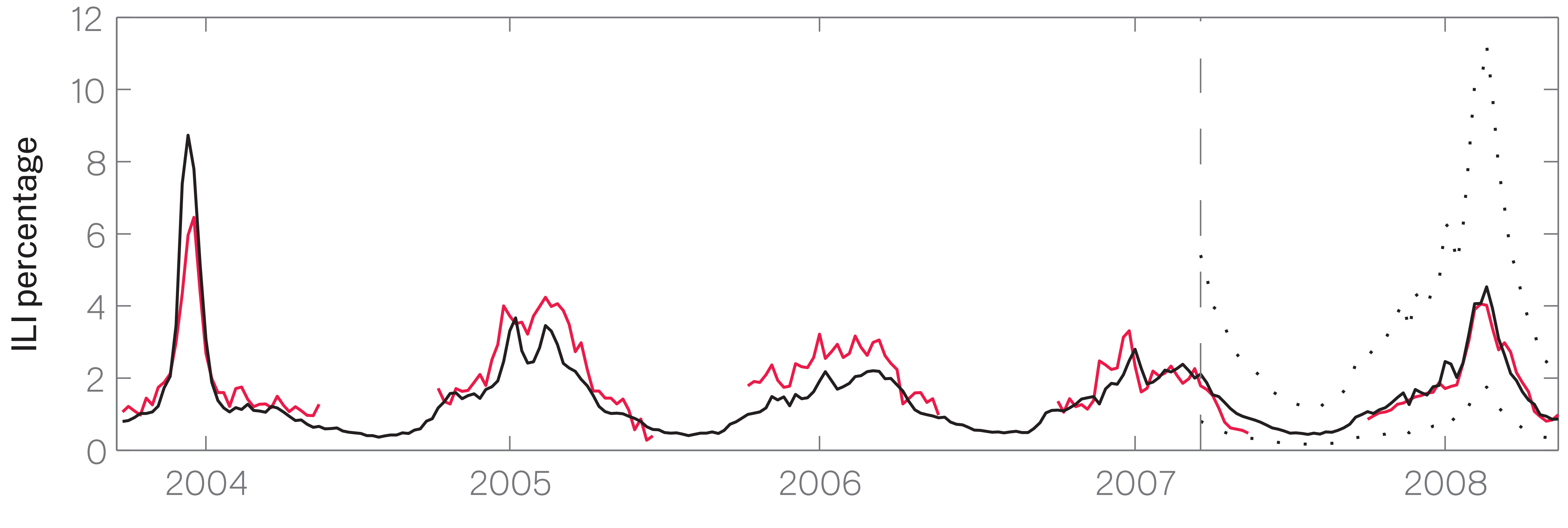


Figure 2: A comparison of model estimates for the Mid-Atlantic Region (black) against CDC-reported ILI percentages (red), including points over which the model was fit and validated. A correlation of 0.85 was obtained over 128 points from this region to which the model was fit, while a correlation of 0.96 was obtained over 42 validation points. 95% prediction intervals are indicated.

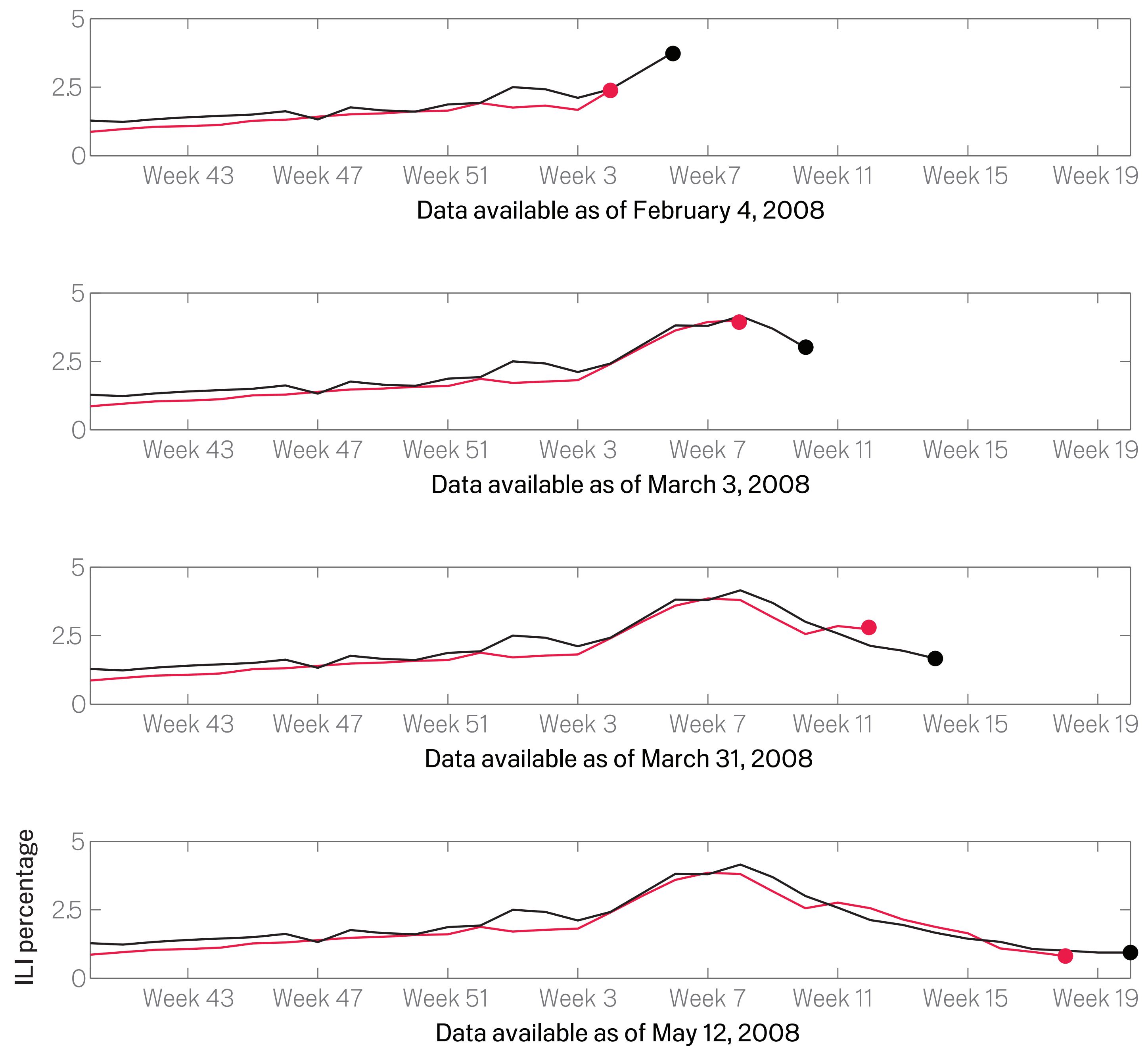


Figure 3: ILI percentages estimated by our model (black) and provided by CDC (red) in the Mid-Atlantic region, showing data available at four points in the 2007-2008 influenza season. During week 5, we detected a sharply increasing ILI percentage in the Mid-Atlantic region; similarly, on March 3, our model indicated that the peak ILI percentage had been reached during week 8, with sharp declines in weeks 9 and 10. Both results were later confirmed by CDC ILI data.

Google Flu Trends — Why this is important

Better data, sooner.

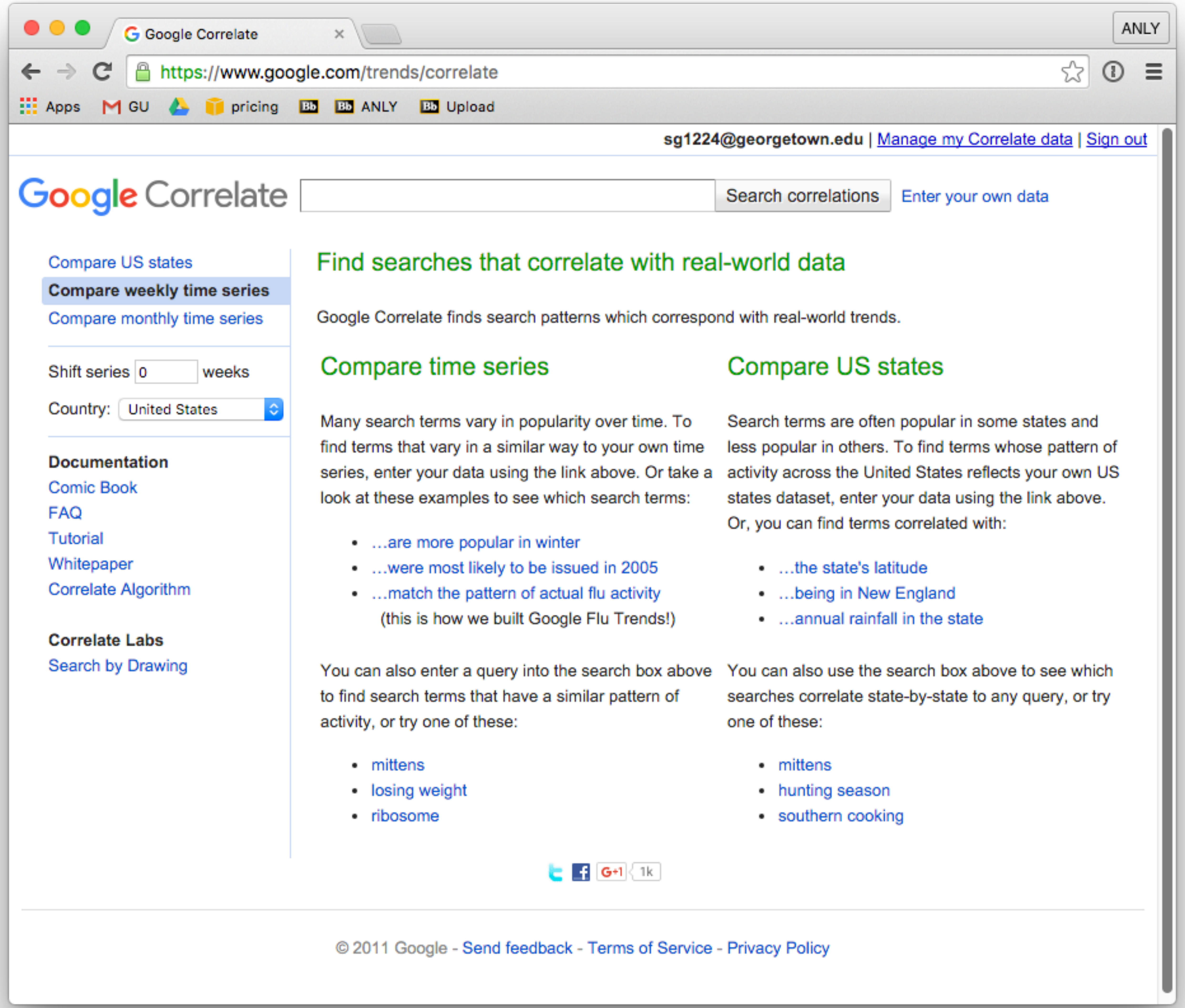
- Detect influenza outbreaks early — early intervention
- High resolution — community-specific data
- Cheaper than traditional surveillance.
- Distinguish flu from colds

How they did it:

- Input data:
  - —*CDC influenza reports (# of cases in each region)*
  - —*50 M google search queries over the same time period*

You can do it too!

- https://www.google.com/trends/correlate



https://www.google.com/trends/correlate

Google Flu Trends — "Big Data Hubris" (Lazar et al.)
David Lazar, Ryan Kennedy, Gary King, Alessandro Vespignani, Science, Vol. 343, 14 March 2014

BIG DATA

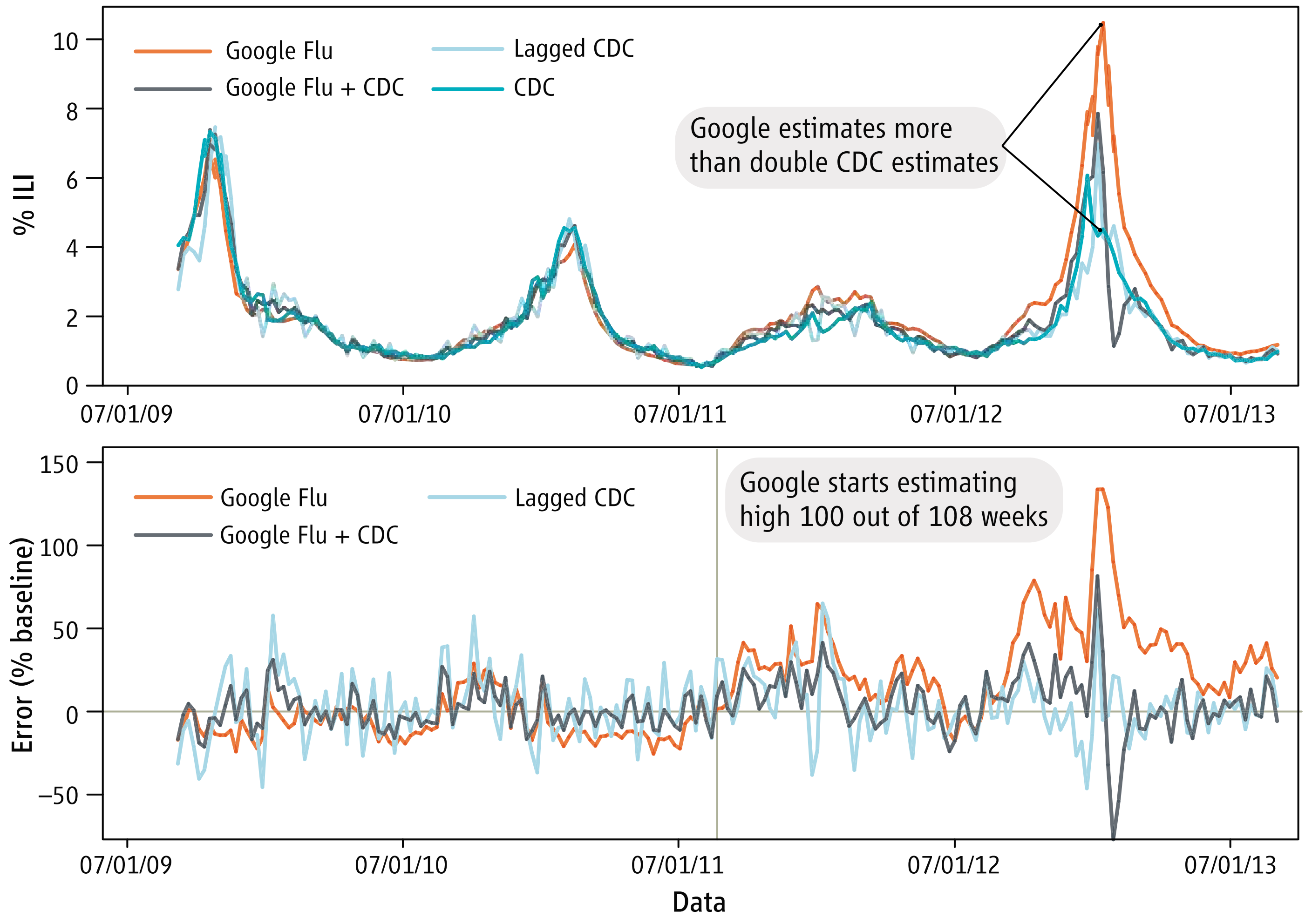
The Parable of Google Flu: Traps in Big Data Analysis

Large errors in flu prediction were largely avoidable, which offers lessons for the use of big data.

David Lazer,[1,2]* Ryan Kennedy,[1,3,4] Gary King,[3] Alessandro Vespignani[3,5,6]

In February 2013, Google Flu Trends (GFT) made headlines but not for a reason that Google executives or the creators of the flu tracking system would have hoped. *Nature* reported that GFT was predicting more than double the proportion of doctor visits for influenza-like illness (ILI) than the Centers for Disease Control and Prevention (CDC), which bases its estimates on surveillance reports from laboratories across the United States (*1*, *2*). This happened despite the fact that GFT was built to predict CDC reports. Given that GFT is often held up as an exemplary use of big data (*3*, *4*), what lessons can we draw from this error?

The problems we identify are not limited to GFT. Research on whether search or social media can



GFT overestimation. GFT overestimated the prevalence of flu in the 2012–2013 season and overshot the actual level in 2011–2012 by more than 50%. From 21 August 2011 to 1 September 2013, GFT reported overly high flu prevalence 100 out of 108 weeks. (**Top**) Estimates of doctor visits for ILI. "Lagged CDC" incorporates 52-week seasonality variables with lagged CDC data. "Google Flu + CDC" combines GFT, lagged CDC estimates, lagged error of GFT estimates, and 52-week seasonality variables. (**Bottom**) Error [as a percentage {[Non-CDC estmate)−(CDC estimate)]/(CDC) estimate)}. Both alternative models have much less error than GFT alone. Mean absolute error (MAE) during the out-of-sample period is 0.486 for GFT, 0.311 for lagged CDC, and 0.232 for combined GFT and CDC. All of these differences are statistically significant at $P < 0.05$. See SM.

Suspect science:

- Attempted to fit 50 million search terms to 1152 data points. Google then threw out search terms, "such as those regarding high school basketball."
- "The odds of finding search terms that match the propensity of the flu but are structurally unrelated, and so do not predict the future, were quite high."
- "GFT completely missed the nonseasonal 2009 influenza A-H1N1 pandemic... The initial version of GFT was part flu detector, part winter detector."
- Revised GFT consistently overestimated flu prevalence
- Google did not document the 45 search terms used, so people couldn't replicate.
- Google changed search engine — engine started "returning potential diagnoses for searches."

Why study massive data?

Better understanding:

- Unlock truths of the past and present
- Predict the future.

Improve society and the planet:

- Public health
- Environmental monitoring & mitigation
- “Data for good” — e.g. Facebook demographics
- Cybersecurity

We have a data-oriented economy

- We are surrounded by data collectors.
- It’s much easier to collect data than to analyze it.
- We should be able to do *something* with all this data.

Another reason to study massive data!

… ?

what: Scientist $150,000-$300,000
job title, keywords or company

where: 22201
city, state, or zip

Find Jobs

Advanced Job Search

~~Required Skills~~ Our promise to you:

- PhD or Master’s Degree in Computer Science or equivalent
- At least 6 years of professional experience
- Background in statistics, data mining, or algorithm development
- AI experience or Machine Learning
- Hadoop, Pig, Hive
- Knowledge of MATLAB, Java, Python, or C++
- NLP expereince
- Candidate must be a US citizen

You will even learn the difference between Pig and Hive!

Keyword Tags MATLAB, Java, C++, Python, HP Vertica, AI, CI, Data, Mining, Science, Scientist, algorithm, analytics, research, development, statistics, SQL, DC
5 days ago

“Learning Outcomes”

At the end of this class, you will be able to:

- Identify technical and social trends in the creation, collection, analysis and storage of massive data.
- Design, cost, and assemble cloud-based computational infrastructure required to perform massive data analysis.
- Perform large-scale data analysis with Python on high-performance workstations using multithreading/multiprocessing and clusters using Hadoop, Map Reduce, Apache Spark, and other advanced technologies.
- Locate, download, “wrangle,” and query structured and unstructured data from Internet sources.
- Research and present information about a new “Big Data” tool on the Internet.
- Understand and discuss academic papers about big data technology and related social issues.

Let us know if you want more learning outcomes!

This course also introduces us to teaching about "massive data!"

Both of us have been working with Big Data for years.

- This is the first time that we've taught this course.
- This is the fist time we've worked together!

Simson L. Garfinkel, Ph.D.

Started working with "Big Data" in 1985
(Made the second CDROM in the US: 600MB of Data. Massive!

Created digital forensics data sets 500GB — 200TB in size

Developed software for forensics processing on 64-core workstations and 2000-core clusters.

Ghaleb M. Abdulla, Ph.D.

Helped develop an early prototype for the ACM digital library (1994).

Developed a parallel ad hoc query engine to mine scientific simulation data on the MCR cluster at LLNL (ranked number 7 on the top 500 list at that time).

Analyzed 7 TB of the CAIDA network traffic data using Map Reduce and PIG to answer the question: Are the 54 traceroute servers setup by CAIDA sufficient to achieve convergence of network coverage?

ANLY 502, by the numbers

First year ANLY 502 taught:	2016
Class sessions:	13
Class length:	2 hours, 40 minutes
Weeks missed because of holidays:	3
Enrolled students:	19 (as of Jan 3, 2015)
Homework assignments:	5
Presentations:	3*
Projects:	1
Online participation:	Weekly

Class Statistics

Class Deliverables

***includes final project**

What you need to take this course

From the catalog:

- Prerequisites: “Good command of R or Python, some knowledge of data structures.”

Additional:

- Ability to read and write Python* code.
- Familiarity with Unix command line and a text editor (e.g. EMACS, vi, nano, etc.)
- A laptop/desktop with at least 8GiB of RAM and +50GB of free drive storage space
  - *I have 16GiB of RAM and a 512GB SSD*
- Commitment to homework and *working beyond the assignments*
- Access to massive data infrastructure (we’ve got this covered!)

Class technologies:

- VMWare and Cloudera QuickStart VM for local processing. (You may use VirtualBox if your desire.)
- Amazon Web Services (AWS) for hands-on “big data” work.

*Most big-data work is done with Python 2.7 due to legacy issues

"Is my Python good enough to take this course?"

You should understand basic functions:

```
def fact(x):
    return x * (x-1) if x>2 else 1
```

dir() and string operations:

```
>>> dir("")
>>> ['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__',
'__getattribute__', '__getitem__', '__getnewargs__', '__gt__', '__hash__', '__init__', '__iter__', '__le__', '__len__', '__lt__',
'__mod__', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__rmod__', '__rmul__', '__setattr__',
'__sizeof__', '__str__', '__subclasshook__', 'capitalize', 'casefold', 'center', 'count', 'encode', 'endswith', 'expandtabs',
'find', 'format', 'format_map', 'index', 'isalnum', 'isalpha', 'isdecimal', 'isdigit', 'isidentifier', 'islower', 'isnumeric',
'isprintable', 'isspace', 'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip', 'maketrans', 'partition', 'replace', 'rfind',
'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split', 'splitlines', 'startswith', 'strip', 'swapcase', 'title',
'translate', 'upper', 'zfill']
```

Control flow:

```
if, else, elif, range, xrange(), class, yield, try, except
```

Need help?

- *http://learnpython.org/*
- *Blackboard Forums*

Class Deliverables — What you need to do!

5 Problem Sets:

- PS01 — Cloudera VM on your Laptop — released L01 (jan 13); due jan 22
- PS02 — Amazon AWS — Calculation on a large dataset — released L02 (jan 25); due feb 5
- PS03 — Apache Spark — released L04 (feb 8); due feb 19
- PS04 — Big Databases — released L05 (feb 22); due mar 4
- PS05 — LLNL — released L09 (apr 4); due apr 22

3 Presentations

- 1. A massive data tool or website
- 2. A massive data paper
- 3. Your final project

+ class participation (in class & online)

1 Midterm — In class, March 21

1 Final Project

- An original project involving massive data analysis. A paper describing what you did.
- A presentation about your project.

Presentations — Each student is responsible for three.

Presentation #1: An open source "big data" tool.

- 5 minutes, 3-10 slides
- What problem are the authors trying to solve?
- What does the program do?
- User report (if you can get it to work)

Peer assessment:
Each student will be able to submit anonymous comments and questions regarding the presentations.

Presentation #2: A "big data" research paper.

- 5 minutes
- What the authors did, and why it's important.

Presentation # 3: Your final project

- 15 min presentation, 10-20 slides
- What you wanted to do, what you actually did, what you found out, and the next steps
- Groups of 2-4; level of work commensurate with group size.

Final project sequence and timeline

Tue. Mar 22 — Initial proposal

- Each student must write two 1-paragraph proposals.
- These will be posted for the class and used for discussion & group formation

March 23 — March 28 — Easter Break

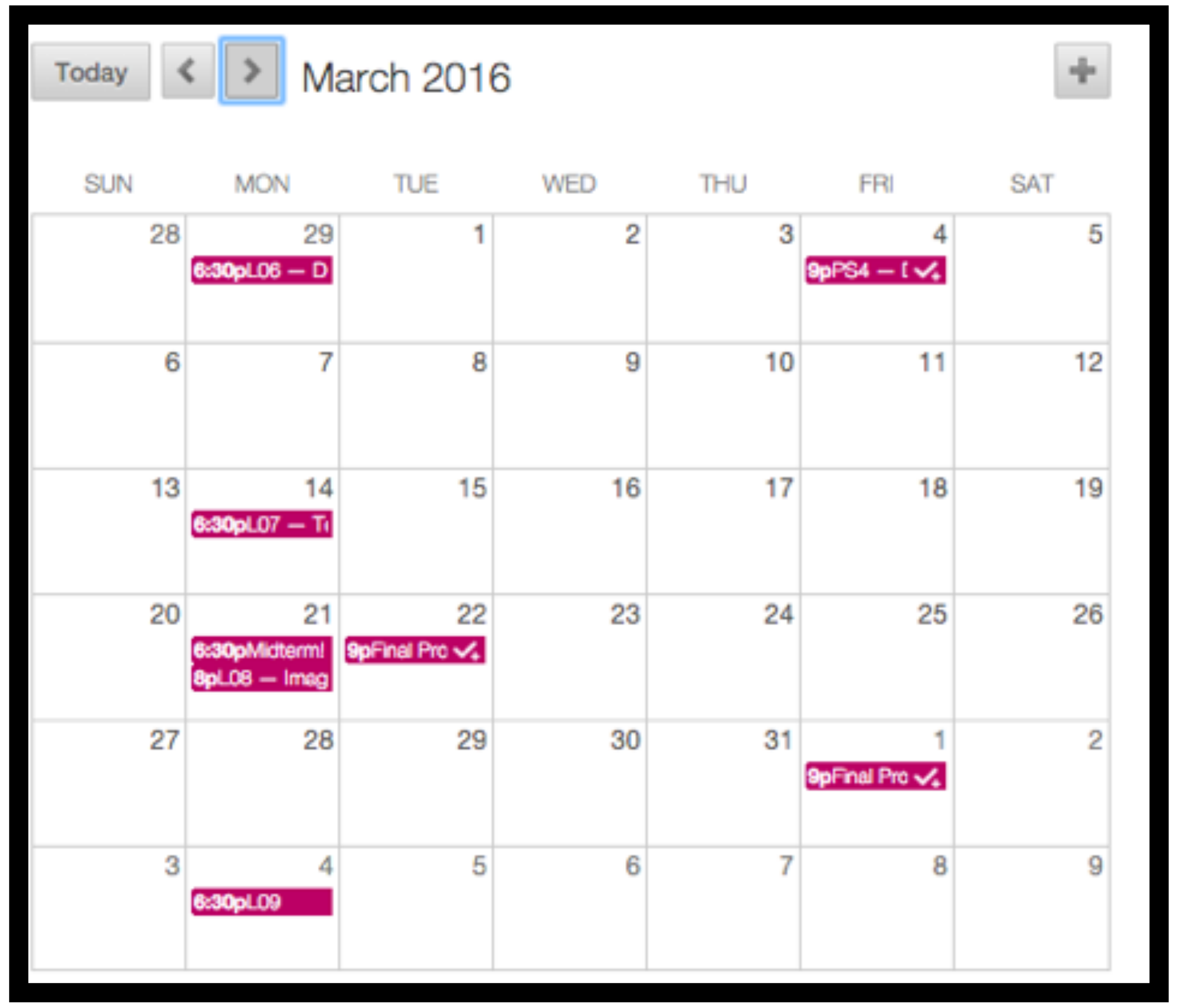
Fri. April 1st — Final project group proposals due

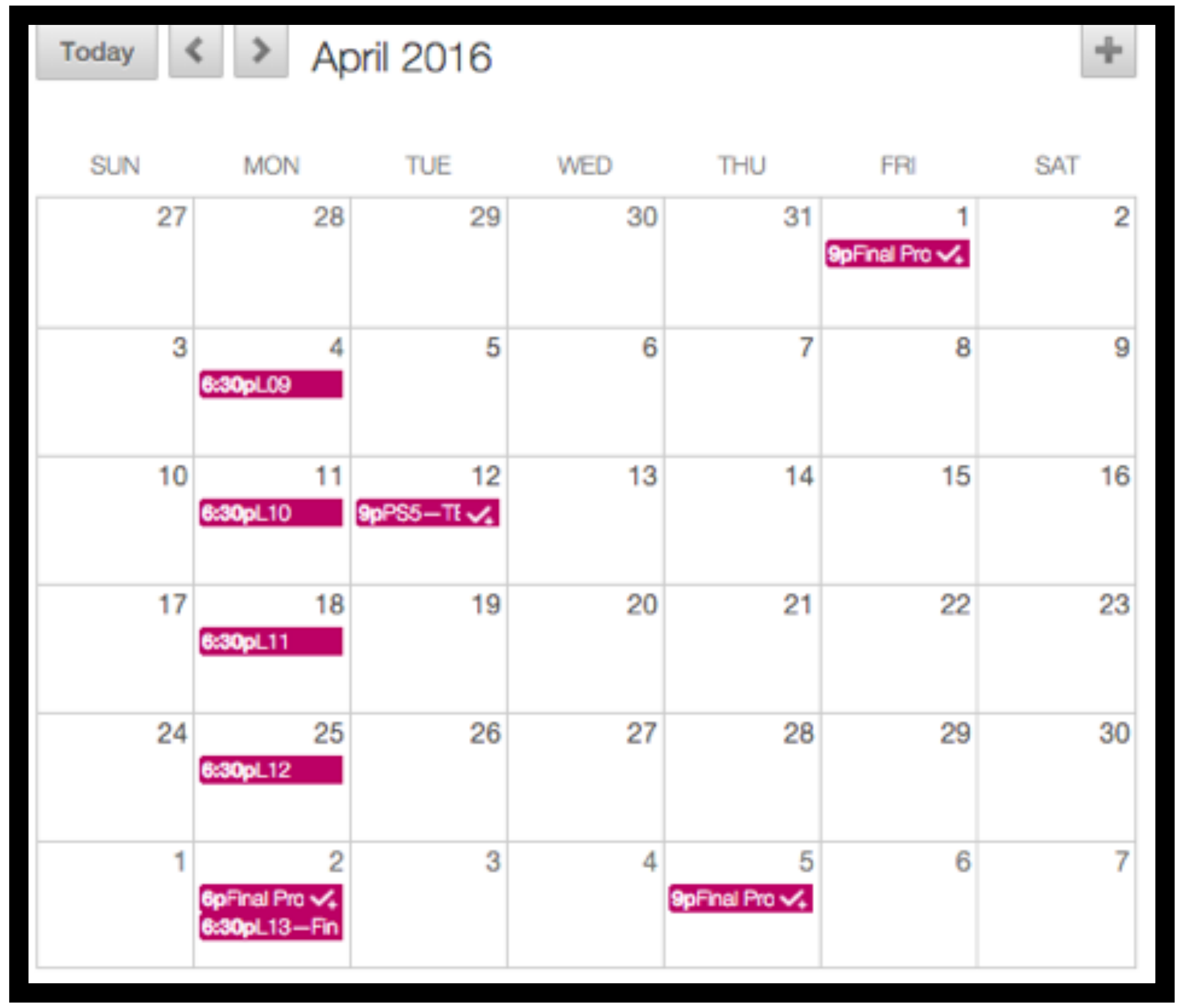
- Each group must submit a 1-2 page proposal clearly documenting what will be done, by whom, with a timeline.

Sun. April 3rd — Proposal response: “accepted” or “revise”

Mon., May 2 — Final projects presented in class

Thu., May 5 — Final projects paper due





More about your final projects.

Your final project must include:

- Literature review
- Clear contribution — data analysis, tool development, etc.
- Validation — how do you prove that you did what you said you did?
- Conclusion
  - —*Start thinking about your final projects now!*

Your final project deliverables include:

- Proposals 1 & 2
- A paper with an abstract, background, literature search, main body, and conclusion
- A slide presentation
- Optional video demo:
  - —*Demos should be 60-120 seconds of video*
  - —*Demos should be uploaded to Black Board or YouTube. Fri., March 25 — Final project individual proposals due*

Required Readings & Optional Readings

Readings are associated with every class

- So we can discuss them in class.

Readings should be completed before class starts!

You are responsible for the content of the required readings.

- You will not be tested on readings that are not discussed in class.
- You may be tested on important aspects of the readings that are not explicitly discussed.

Each lesson may have one or more “optional” readings.

- *These readings are for your personal edification.*
- *Please let the class know if you find them interesting.*
- *They are pre-approved for presentations!*

Class Weights

Student Deliverable	Assigned	Due	Weight
Problem Set #1: Hadoop Hello!	Jan 13	Jan 22	9
Problem Set #2: AWS Cluster	Jan 25	Feb 5	9
Problem Set #3: Apache Spark	Feb 8	Feb 19	9
Problem Set #4: Scoop, Hive, and re-identification	Feb. 22	Mar. 4	9
Problem Set #5: LLNL	Apr. 4	Apr. 12	9
Midterm		Mar. 21	15
Proposed dates for two presentations		Jan. 15	1
Presentation #1 — An open source big data software tool			4
Presentation #2 — A big data paper			4
Final Project Proposals (2)		Mar. 22	1
Final Project Group Proposal		April 1	1
Final Project Presentation		May 2	5
Final Project Paper		May 5	14
Classroom Participation			10
Total:			100

Class Style

Class meets 6:30 — 9:00

- Typically class will involve:
  - *Introduction to the day*
  - *Recap of reading*
  - *Student presentation (!)*
  - *Break*
  - *Lab work / problem sets / projects*

Please bring your laptops to class!

Class Materials

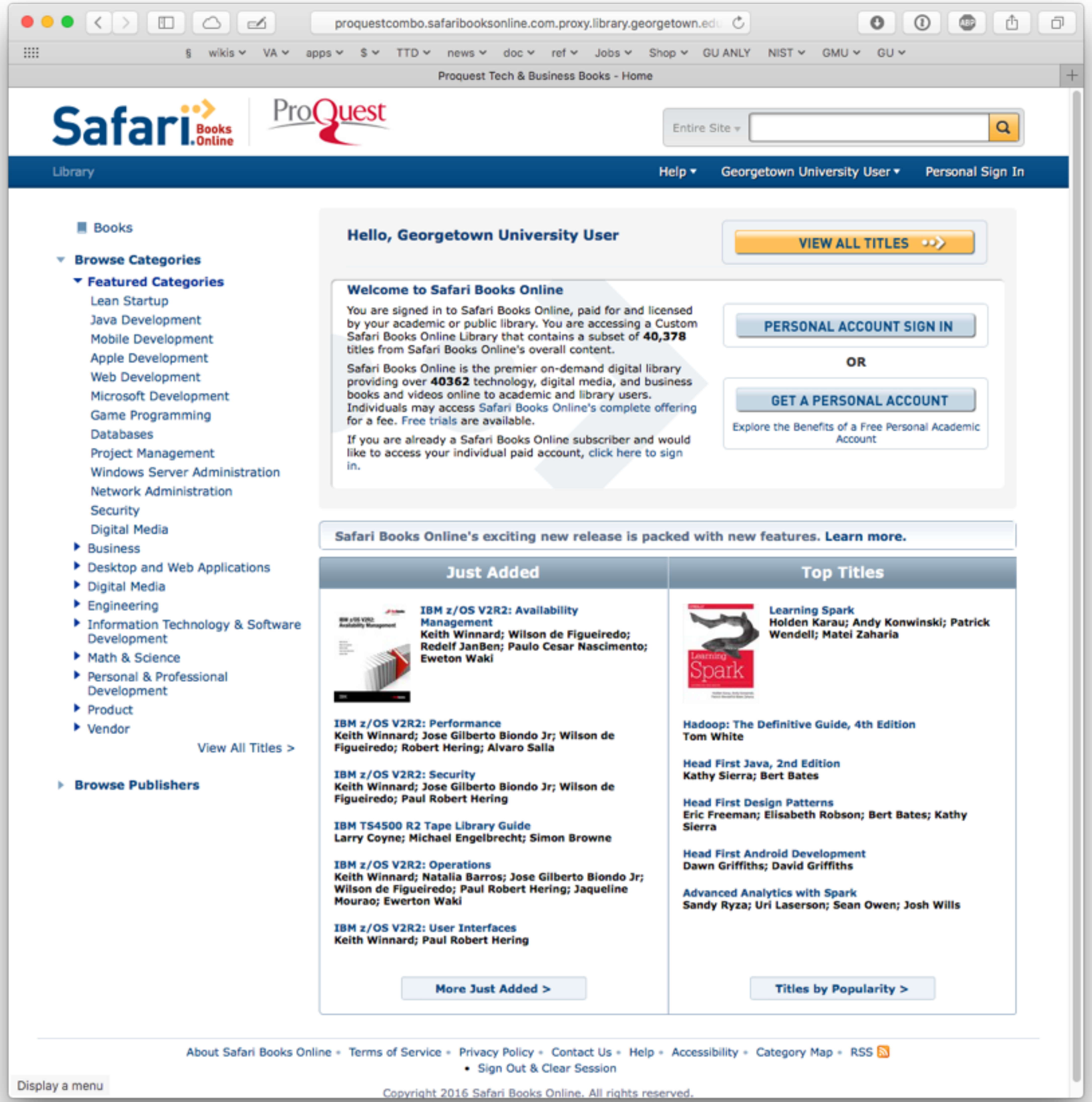
Class materials are on Blackboard

- Slides (PDF)
- Break fast the slides
- Lab work / problem sets / projects
- www.library.georgetown.edu

Class calendar:

- Blackboard
- You can sync to Google Calendar / Phone / etc.

O'Reilly books are available on Safari



Class Calendar — how does this all fit together?

Fewer slides, more class participation!

L01: Introduction, Scaling on a Single Computer, & Cloudera VM
L02: Scaling on Multiple Computers, Grids, Hadoop Architecture,
L03: Storage at Scale, Hadoop Distributed File System (HDFS)
L04: Spark
L05: Massive Databases (special guest: Donald Miner)
L06: Data Wrangling and De-Identification (special guest: Jim Koenig)
L07: Text Processing at Scale (special guest: Peter Wayner)
L08: Midterm (90 min) & Image Processing
L09: LANL #1 — Big Data in High Performance Computing
L10: LANL #2 — Power and Performance for High Performance Computing (HPC)
L11: LANL #3 — Scientific and Simulation Data
L12: LANL #4 — Scientific Data Analysis Approaches, Architectures, and Workflow Systems
L13: Final Projects Presented

SUBJECT TO CHANGE!

Class Policies

Check BlackBoard Frequently! (or sign up for e-mail alerts)

Class participation is expected.

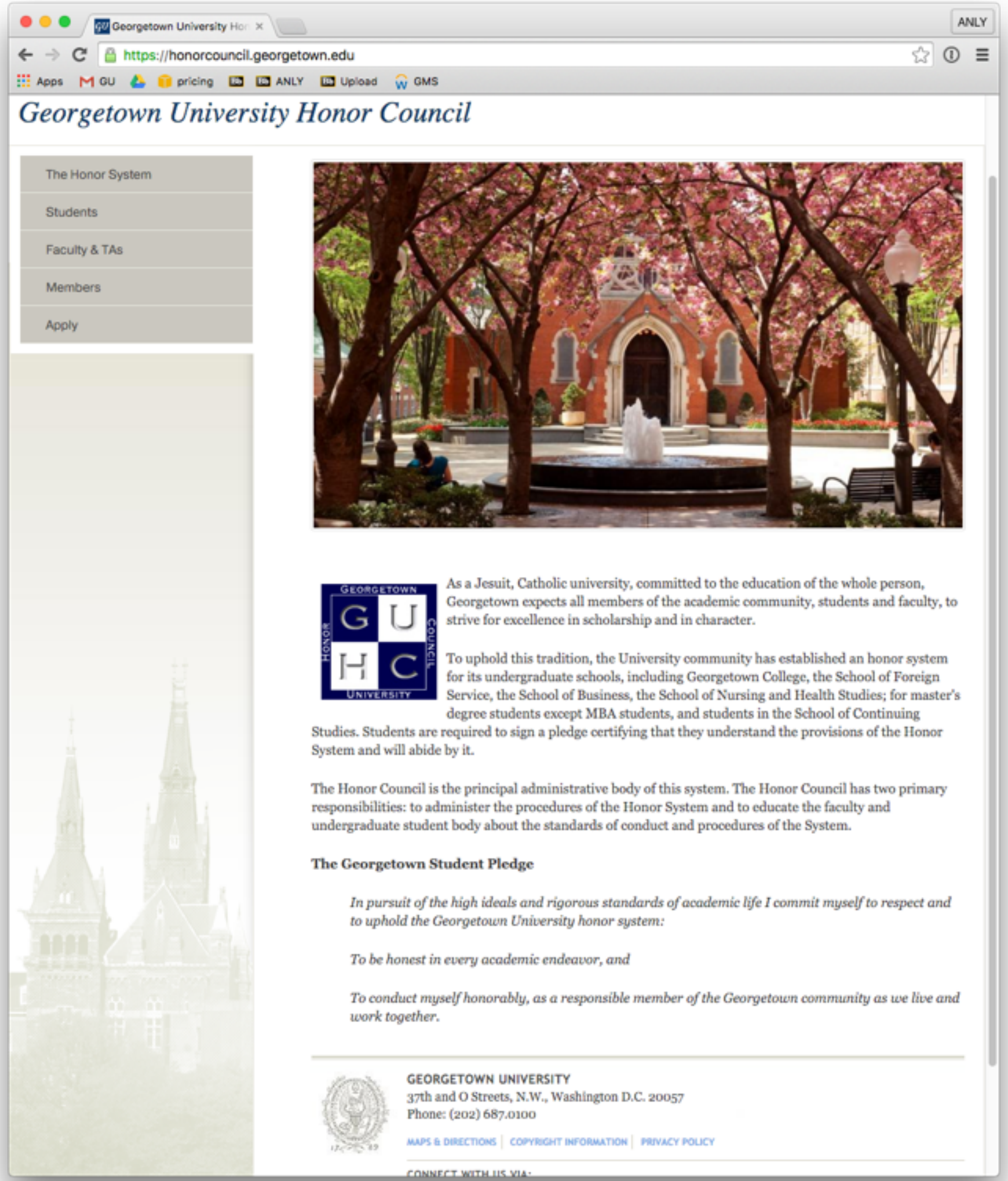
Do the mandatory reading!

Problem sets are due at the start of class.

- Late homework will only be accepted in exceptional circumstances.
- Collaboration is allowed, but must be documented.
- You are expected to submit your own work.

Follow the Georgetown Student Pledge

- Please confirm that you will follow using BlackBoard.



Class management and philosophy

I view education as a collaborative process.

My role — create an environment in which you can excel.

—*Written materials, assignments, online resources.*

Your role — be an engaged learner!

—*Do the readings & assignments*

—*Come to class*

—*Seek out additional information and bring it to class.*

Preferred contacts:

- Email doesn't scale for class management—One person has a hard time keeping up with 20!
- Please post your questions on the materials in the online discussion forum
  —*I will see them and try to answer them within 36 hours*
  —*Other students can answer as well! (please!)*
- Please use email for administrative issues—grades, late assignments, etc.
  —*I will try to answer email within 36 hours.*

Google Survey Results

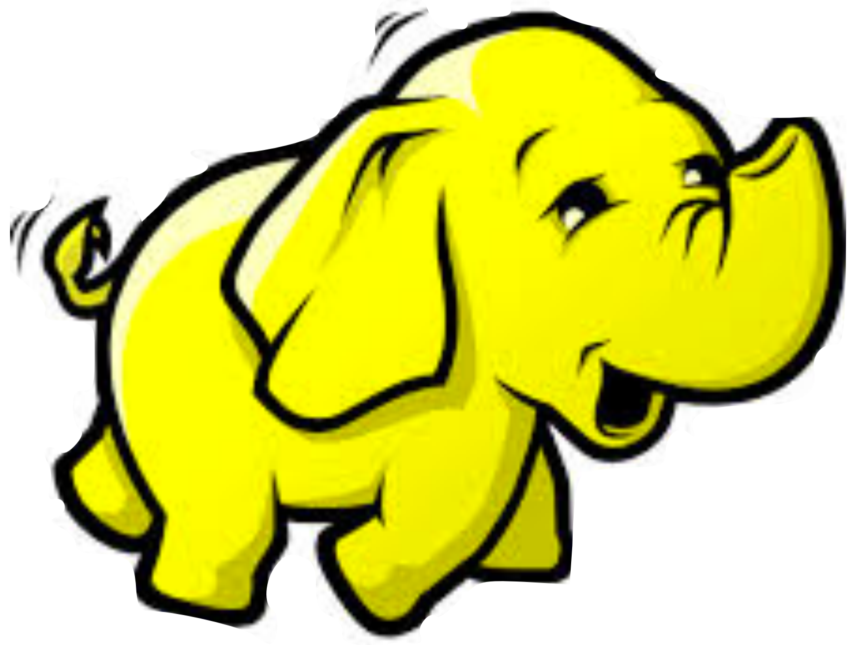
http://bit.ly/ANLY502-2016-Responses

Massive Data Technology (for ANLY 502)

Massive Data Technology:
Specific technology that we use in this course.

Program Layer — code that you write to manipulate the data

- Mostly Python, some Scala.
- Optionally Java



Software Infrastructure Layer — where your code runs

- This course is based on Hadoop, MapReduce, and Spark
- A little with massive databases: HBase / Pig / Impala / Spark

Virtualization Layer — the runtime environment

- Most “massive data” technology runs on Linux
- Some products have been ported to Windows
- A few products were developed on Windows, or have front-ends that run on Windows
- Virtual Machines (VMs) can run on your laptop or on the server.

Hardware Layer — The physical hardware on which the VMs run

Quick Survey: How many people have used VMs?

What host operating system do you use?

- Windows
- MacOS
- Linux

Have you used virtualization? If so, which kind? Check all

- VMWare Workstation
- VMWare Fusion
- VMWare VCenter
- VirtualBox
- Xen
- KVM

What have you used virtualization for?

- ______________________________

Exercises/Labs will be on your laptop and Amazon Web Services.

Amazon Elastic MapReduce

*http://theatln.tc/1GVLpOM

For "small data" and initial testing, you will use your own computer.

VirtualBox — Virtualization platform made by Oracle (free)

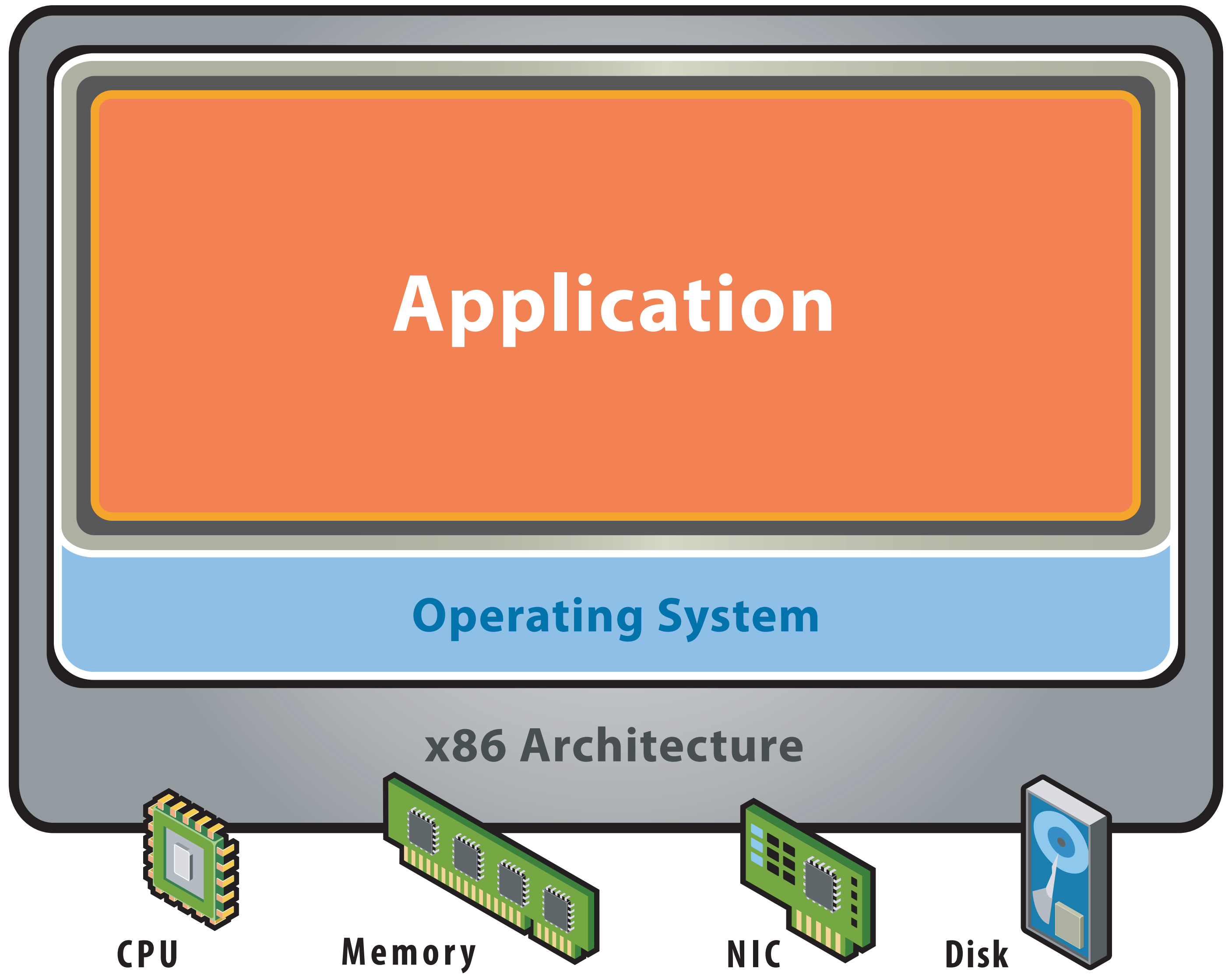
VMWare — Virtualization platform made by VMWare ($$)

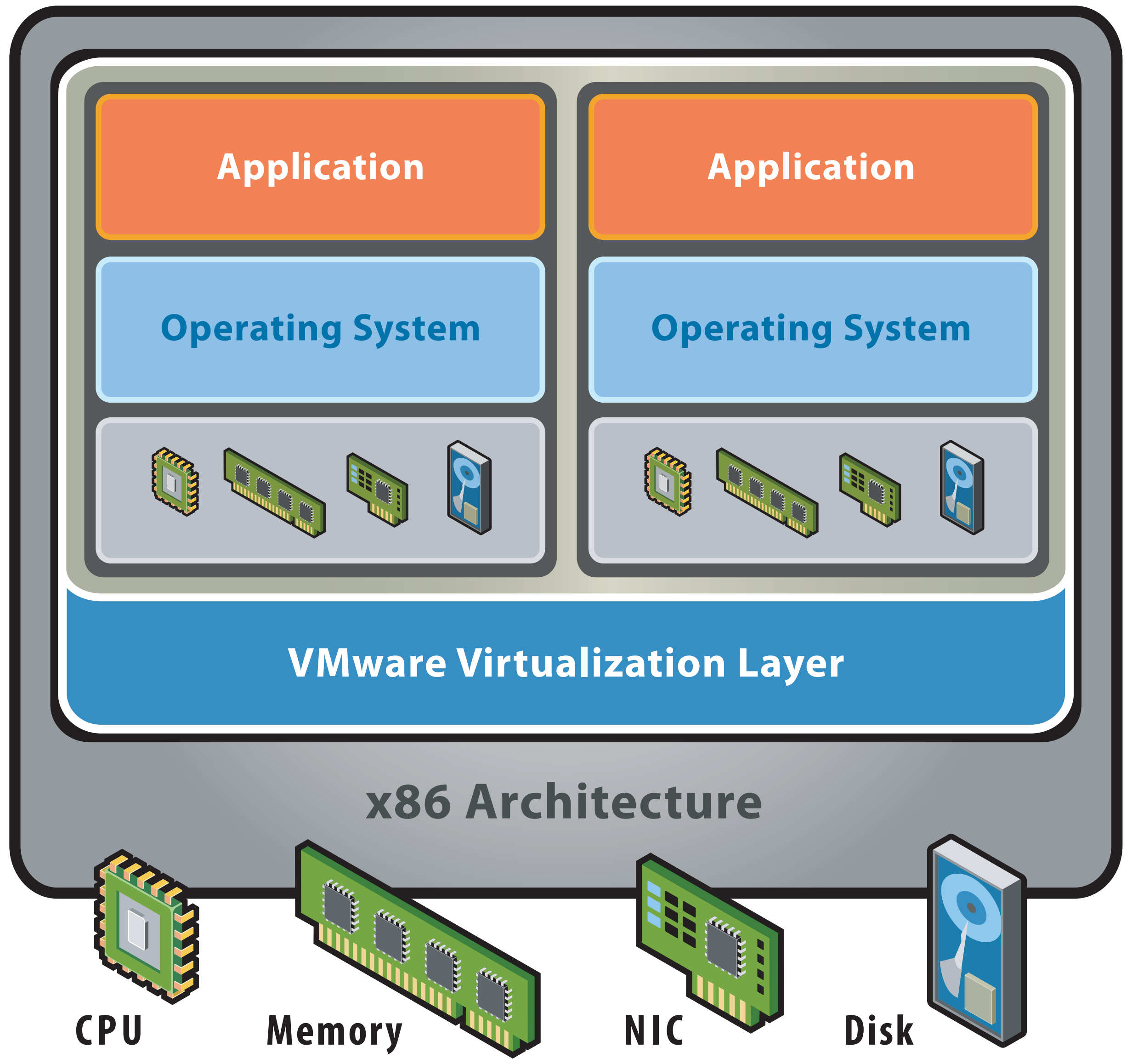
- Simulates PC hardware.
- You specify:
  - *RAM, Video RAM, # processors, network interfaces, disk*

"Guest OS" is running inside

"Host OS" is what you are running on: PC, Mac, or Linux.

Basic Idea: A computer within a computer





Before Virtualization:

- Single OS image per machine
- Software and hardware tightly coupled
- Running multiple applications on same machine often creates conflict
- Underutilized resources
- Inflexible and costly infrastructure

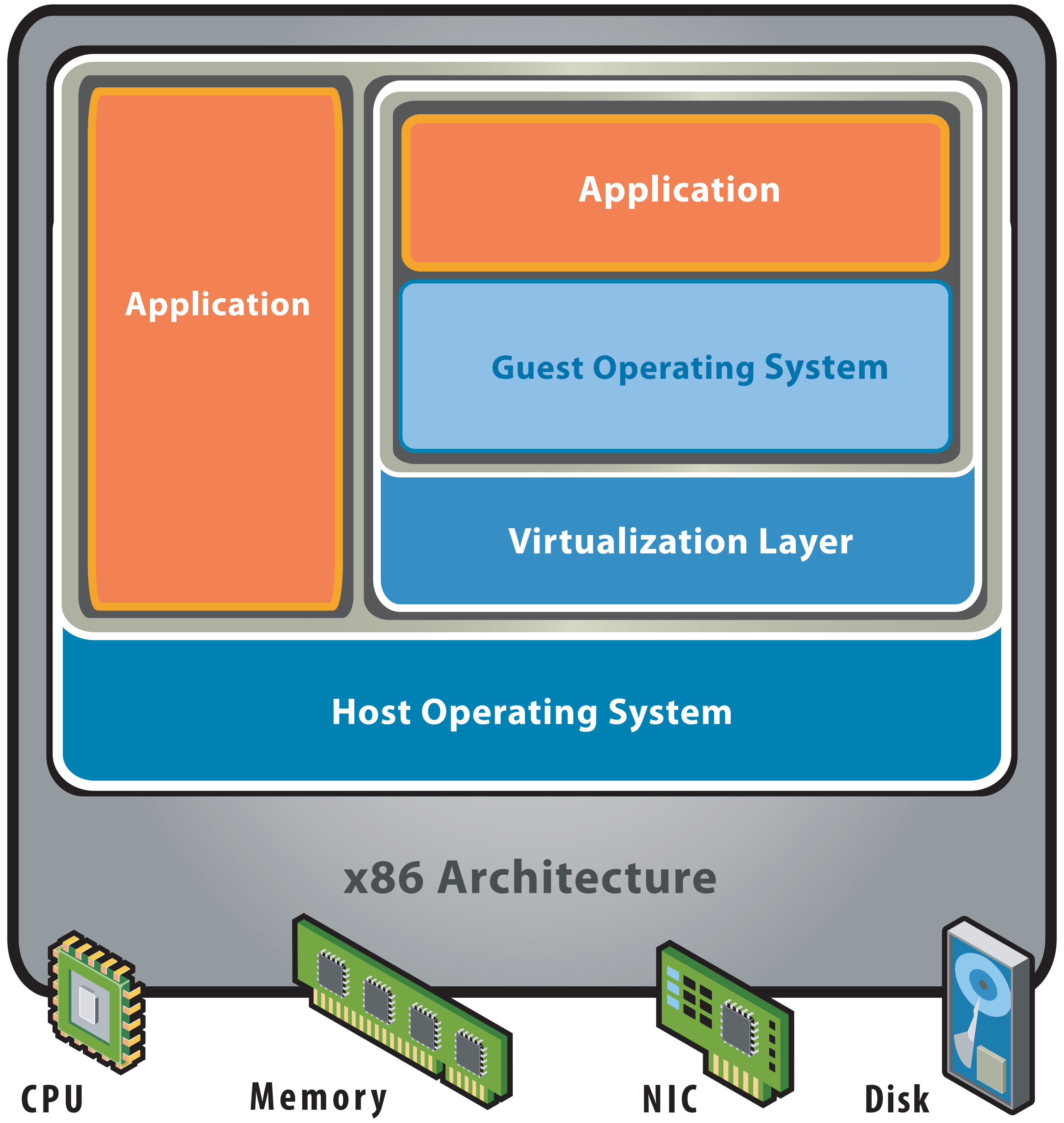
After Virtualization:

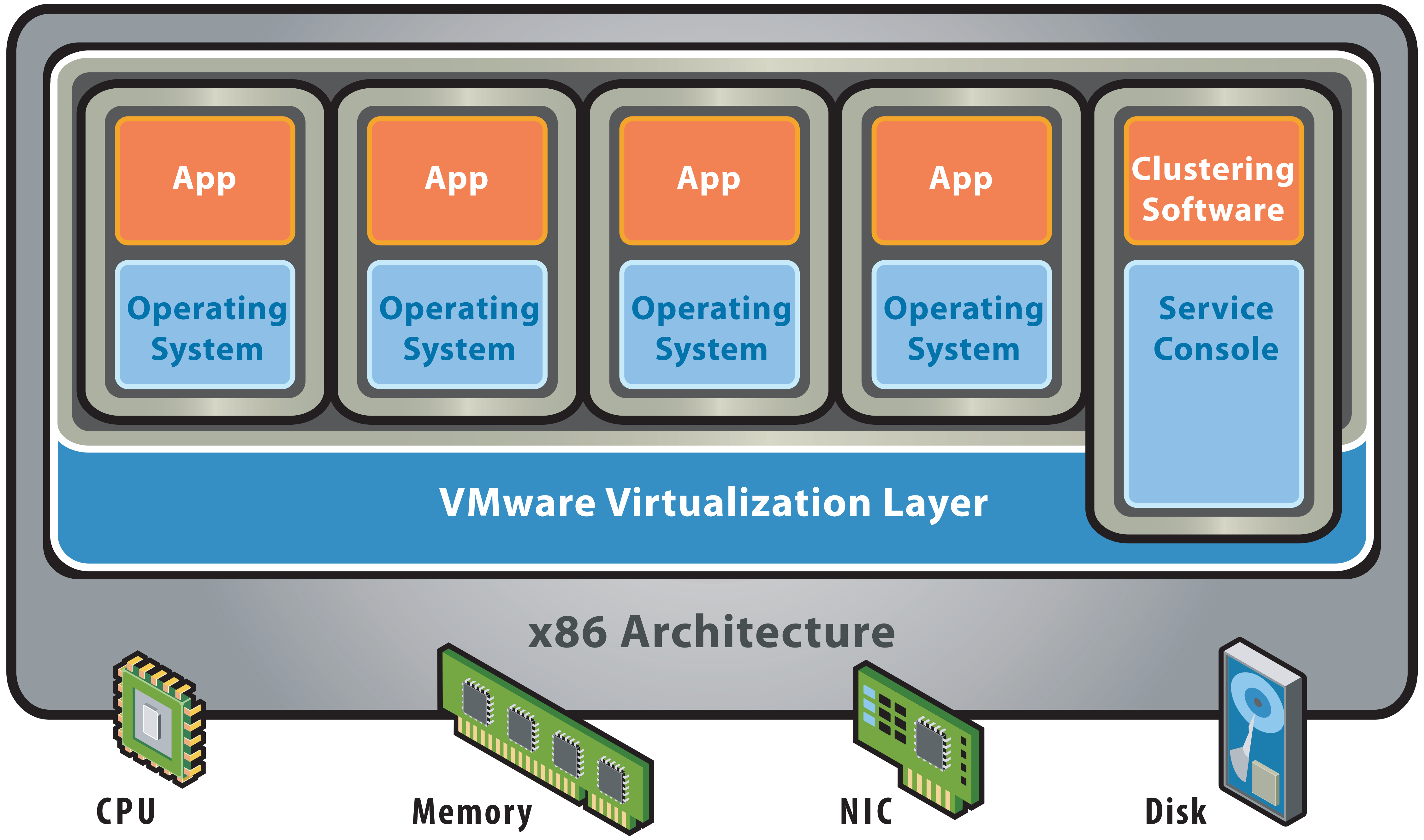
- Hardware-independence of operating system and applications
- Virtual machines can be provisioned to any system
- Can manage OS and application as a single unit by encapsulating them into virtual machines

VMWare Virtualization Ovewview

- https://www.vmware.com/pdf/virtualization.pdf

Two virtualization architectures: "Hosted" and "Bare-Metal"





Hosted Architecture

- Installs and runs as an application
- Relies on host OS for device support and physical resource management

Bare-Metal (Hypervisor) Architecture

- Lean virtualization-centric kernel
- Service Console for agents and helper applications

Big data problems you can work on the Amazon Web Services.

July 5, 1994 — Amazon.com was founded by Jeff Bezos

- (Originally named “Cadabra”)
- Renamed “Amazon” in 1995 with goal of being the “biggest” store in the world.
- First book ordered in 1995, *Fluid Concepts and Creative Analogies.**

By 1998, showing a single web page on Amazon.com required computation from more than 100 computers.

- Amazon made organizing thousands of computers an institutional priority.

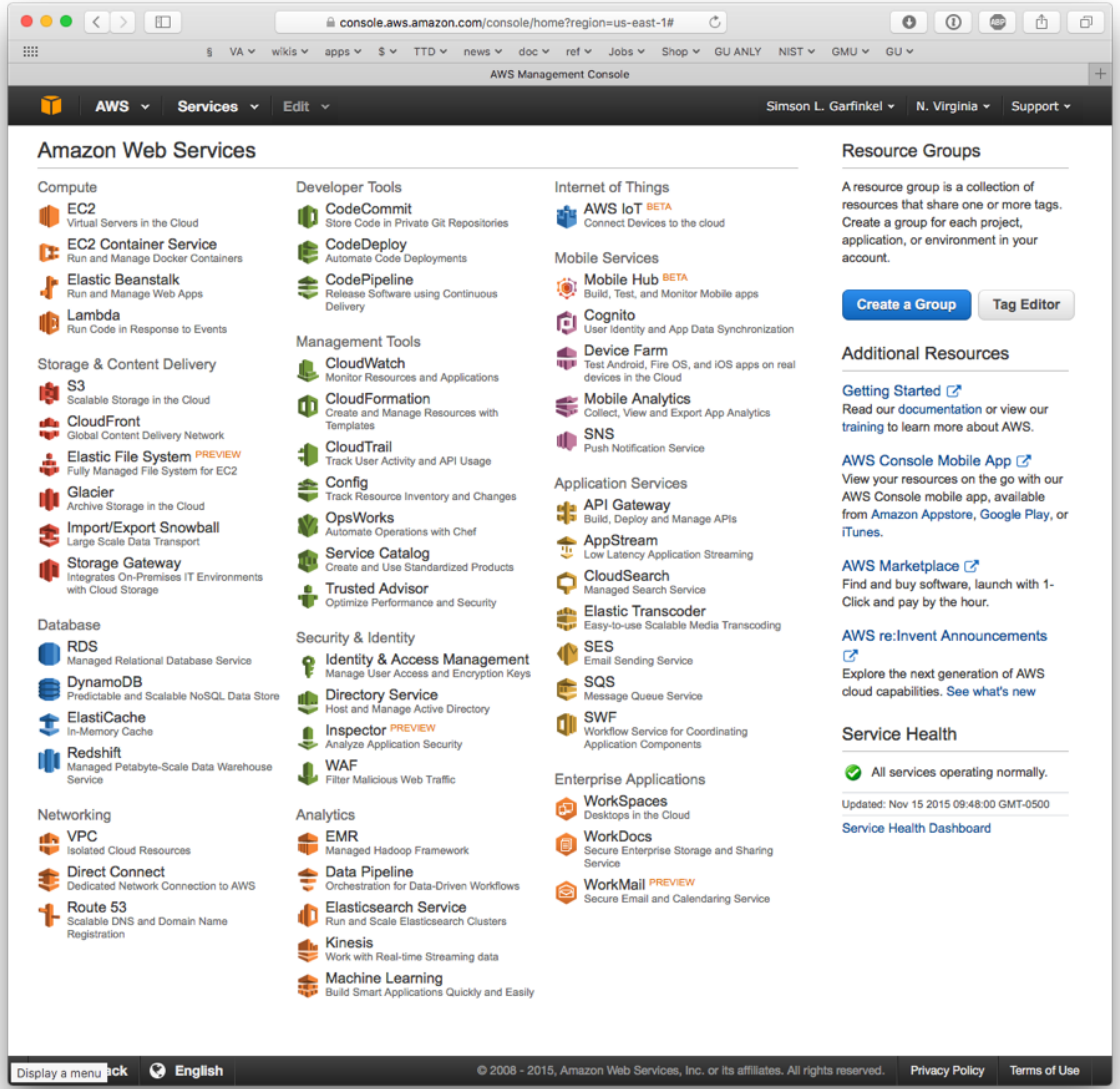
In 2006, Amazon started making its systems available as a commodity

- Simple Queue Service (SQS) — Reliable messages up to 256KB in size.
- Elastic Compute Cloud (EC2) — virtual machines
- Simple Storage Service (S3) — unlimited storage

Amazon Elastic MapReduce

*http://theatln.tc/1GVLpOM

Today Amazon Web Services (AWS) has dozens of offerings



In 2011, the National Institute of Standards and Technology defined a standard terminology for cloud computing.

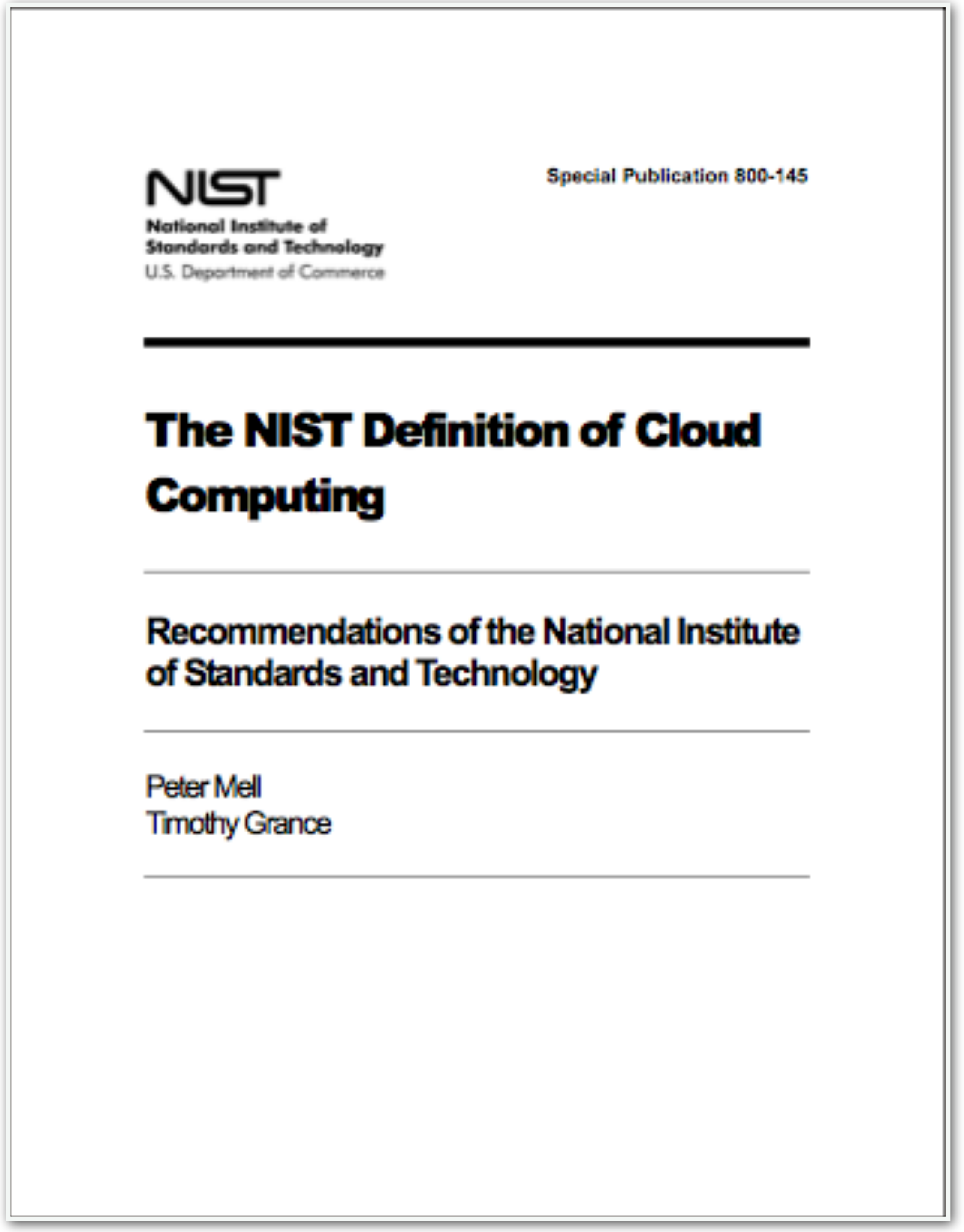
Special Publication 800-145: The NIST Definition of Cloud Computing

Essential Characteristics:

- On-demand self-service ✔
- Broad network access ✔
- Resource pooling ✔
- Rapid elasticity ✔
- Measures service ✔

Deployment Models:

- Private cloud ✔
- Community cloud
- Public cloud ✔
- Hybrid cloud


NIST
National Institute of Standards and Technology
U.S. Department of Commerce
Special Publication 800-145

The NIST Definition of Cloud Computing

Recommendations of the National Institute of Standards and Technology

Peter Mell
Timothy Grance

Service Models:

- Software as a Service (SaaS) ✔
- Platform as a Service (PaaS) ✔
- Infrastructure as a Service (IaaS) ✔

Amazon offers most of these models!

Pictures of our physical data center are not very useful…

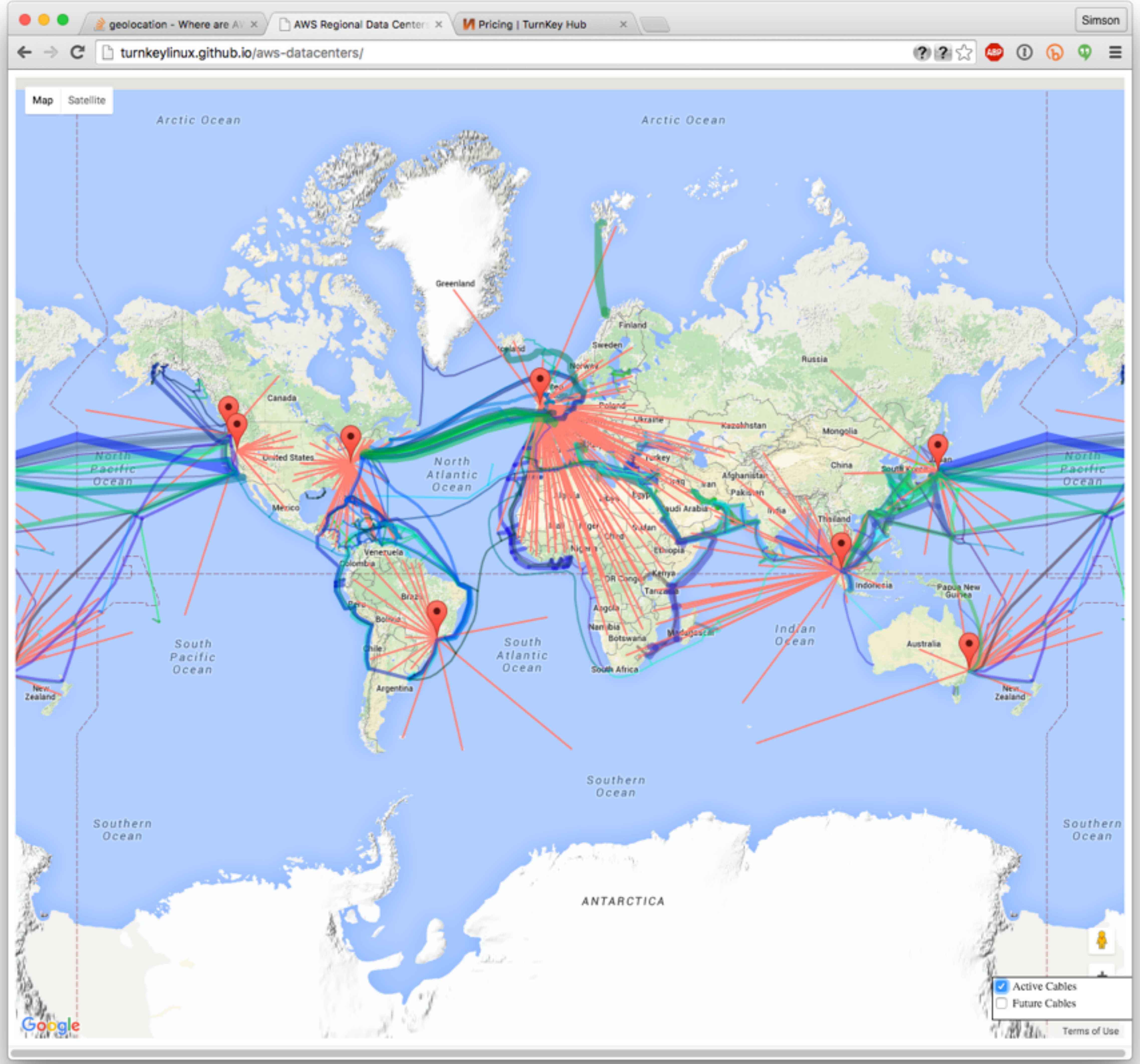
http://bit.ly/amazon-data-center1

More important:
Where the data centers are and the links between them.

Location matters:

- Speed of light: 300,000 Km/sec = 300Mm/s
- Distance to Seattle: ≈5,000 Km = 5Mm
- Minimum time to Seattle: 5 Mm ÷ 300 Mm/s = 1.6 msec

- Distance to Reston: ≈ 10Km
- Minimum time to Reston: 10 Km ÷ 300,000 Km/sec = 33μsec



http://www.turnkeylinux.org/blog/aws-datacenters

For more details, see James Hamilton's presentation from Amazon's 2011 Technology Open House.

Perspective on Scaling

Every day, Amazon Web Services adds enough new capacity to support all of Amazon.com's global infrastructure through the company's first 5 years, when it was a $2.76B annual revenue enterprise

2011/5/5 http://perspectives.mvdirona.com 4

http://bit.ly/1GVKI86

There are many alternatives to Amazon

Top Tier — scalable VMs, Services, etc.:

- Google Cloud Platform — https://cloud.google.com
- Microsoft Azure — https://azure.microsoft.com/en-us/
- Rackspace — http://www.rackspace.com

Bargain basement:

- Dreamhost — http://www.dreamhost.com/
- WebFaction — https://www.webfaction.com

Services may charge for:

- Computation — Virtual Machines
- Storage
- Bandwidth
- Special APIs and Services
- Setup

We are using Amazon because:

- Currently best developed of the services
- Excellent documentation
- Many online tutorials

Amazon has “first mover advantage” and has not slipped behind — at least, not yet!

Each student will have $100 of "free" Amazon time.

You can do a lot with $100:

Price for a General Purpose t2.medium (2 CPU, 4 GB, Variable ECU, EBS) [1]	$.052/hour ($8.74/week)
Price for m3.2xlarge Elastic Map Reduce (4 CPU, 26 ECU, 30GB, 2x80 SSD) [2]	$0.532/hour (EC2) + $0.140/hour (EMR) = $0.672/hour ($112.90/week)
EBS General Purpose Storage (SSD) [3]	$0.10/GB-month ($2.25 to store 100GB for a week)
EBS Magnetic volumes	$0.05/GB-month + $0.05 per 1 million I/O requests
EBS "snapshots"	$0.95/GB-month
Price to access public EBS datasets from EC2	FREE

[1] https://aws.amazon.com/ec2/pricing/
[2] https://aws.amazon.com/elasticmapreduce/pricing/
[3] https://aws.amazon.com/ebs/pricing/

Introduction in Summary

Welcome to ANLY 502

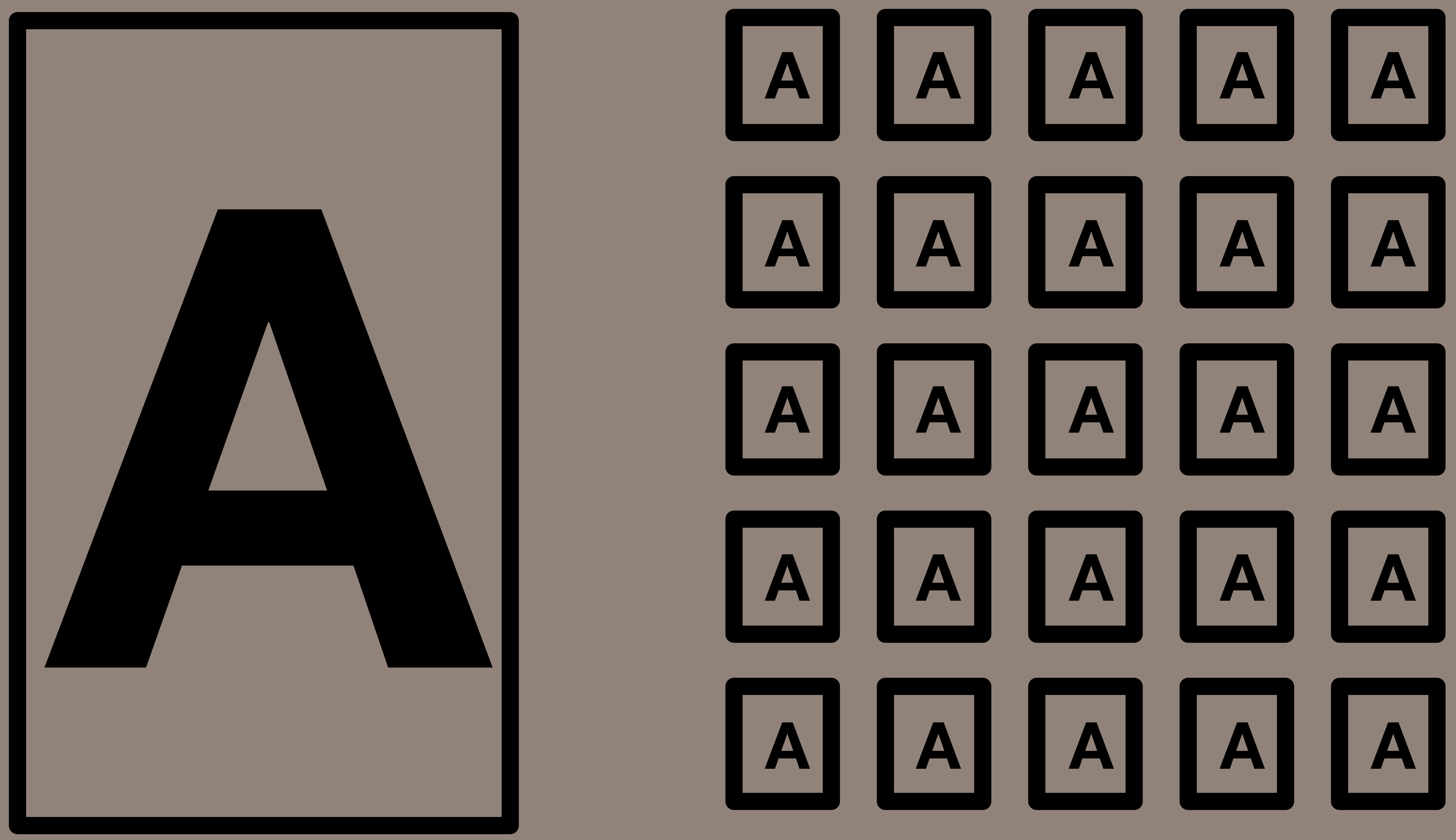
This course teaches you to think about and work with massive data.

We will use the Cloudera VM and Amazon AWS

You are responsible for:

- Coming to class
- Doing the reading
- Working the problem sets
- An in-class presentation about a software package or research paper
- Researching your own project and writing a paper.

Questions?



Making computers run fast:
Moore’s Law and Parallelization

Before we get started, a word about units

Decimal			Binary				
Value	Metric (SI)		Value	IEC		JEDEC	
1000	kB	kilobyte	1024	KiB	**kibibyte**	KB	kilobyte
1000^2	MB	megabyte	1024^2	MiB	mebibyte	MB	megabyte
1000^3	GB	gigabyte	1024^3	GiB	gibibyte	GB	gigabyte
1000^4	TB	terabyte	1024^4	TiB	tebibyte	–	
1000^5	PB	petabyte	1024^5	PiB	pebibyte	–	
1000^6	EB	exabyte	1024^6	EiB	exbibyte	–	
1000^7	ZB	zettabyte	1024^7	ZiB	zebibyte	–	
1000^8	YB	yottabyte	1024^8	YiB	yobibyte	–	

https://en.wikipedia.org/wiki/Kibibyte

IEC — International Electrotechnical Commission

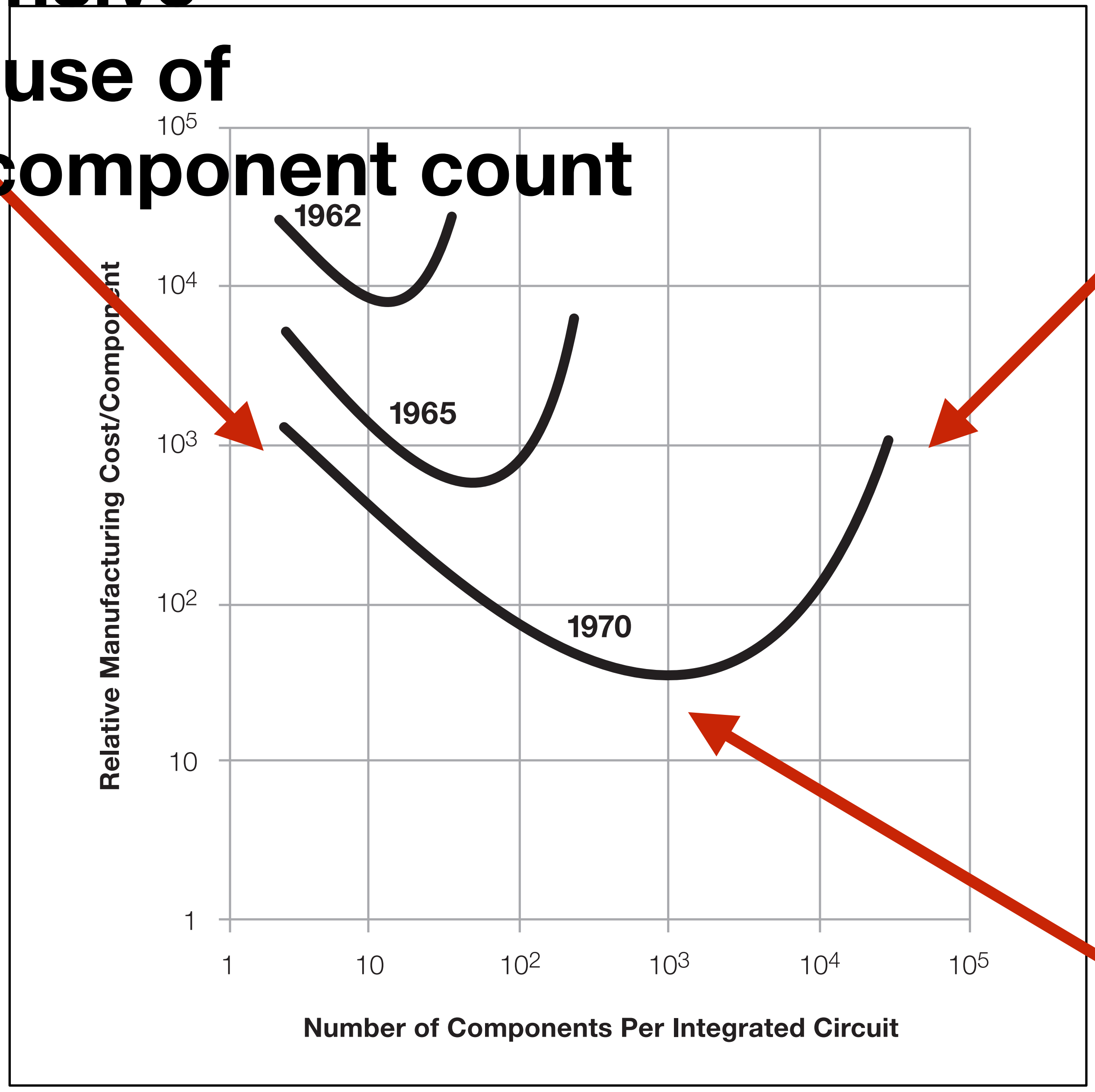
JEDEC — Joint Electron Device Engineering Council — Just for DRAM

You've heard of "Moore's Law"

1965: Gordon Moore suggested:

- Integrated circuits will be cost-effective.
- Increased integration resulted in increased reliability.
- From 1965 to 1970, the cost of of manufacturing should drop by 90%, and integration increase 100 fold:

Expensive because of low component count



Expensive because of high integration

Sweet Spot

The experts look ahead

Cramming more components onto integrated circuits

With unit cost falling as the number of components per circuit rises, by 1975 economics may dictate squeezing as many as 65,000 components on a single silicon chip

By Gordon E. Moore

Director, Research and Development Laboratories, Fairchild Semiconductor division of Fairchild Camera and Instrument Corp.

The future of integrated electronics is the future of electronics itself. The advantages of integration will bring about a proliferation of electronics, pushing this science into many new areas.

Integrated circuits will lead to such wonders as home computers—or at least terminals connected to a central computer—automatic controls for automobiles, and personal portable communications equipment. The electronic wristwatch needs only a display to be feasible today.

But the biggest potential lies in the production of large systems. In telephone communications, integrated circuits in digital filters will separate channels on multiplex equipment. Integrated circuits will also switch telephone circuits and perform data processing.

Computers will be more powerful, and will be organized in completely different ways. For example, memories built of integrated electronics may be distributed throughout the machine instead of being concentrated in a central unit. In addition, the improved reliability made possible by integrated circuits will allow the construction of larger processing units. Machines similar to those in existence today will be built at lower costs and with faster turn-around.

The author

Dr. Gordon E. Moore is one of the new breed of electronic engineers, schooled in the physical sciences rather than in electronics. He earned a B.S. degree in chemistry from the University of California and a Ph.D. degree in physical chemistry from the California Institute of Technology. He was one of the founders of Fairchild Semiconductor and has been director of the research and development laboratories since 1959.

Present and future

By integrated electronics, I mean all the various technologies which are referred to as microelectronics today as well as any additional ones that result in electronics functions supplied to the user as irreducible units. These technologies were first investigated in the late 1950's. The object was to miniaturize electronics equipment to include increasingly complex electronic functions in limited space with minimum weight. Several approaches evolved, including microassembly techniques for individual components, thin-film structures and semiconductor integrated circuits.

Each approach evolved rapidly and converged so that each borrowed techniques from another. Many researchers believe the way of the future to be a combination of the various approaches.

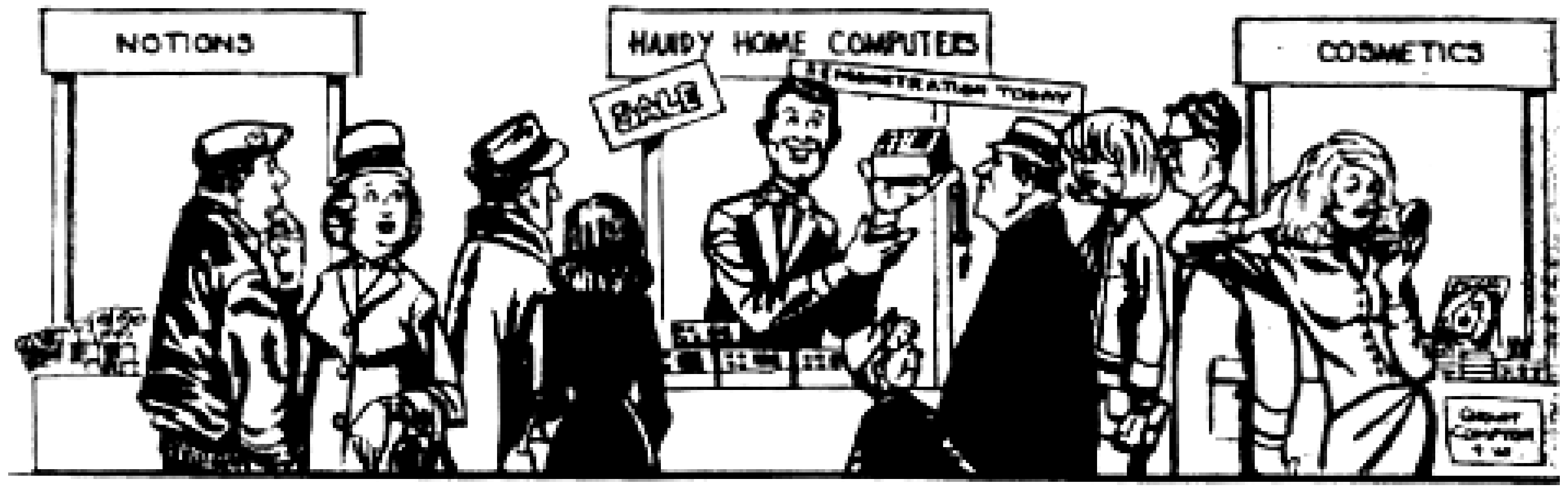
The advocates of semiconductor integrated circuitry are already using the improved characteristics of thin-film resistors by applying such films directly to an active semiconductor substrate. Those advocating a technology based upon films are developing sophisticated techniques for the attachment of active semiconductor devices to the passive film arrays.

Both approaches have worked well and are being used in equipment today.

Electronics, Volume 38, Number 8, April 19, 1965

“The electronic wrist-watch needs only a display to be feasible today.” —Gordon Moore, 1965

Pulsar watch advertisement, 1972



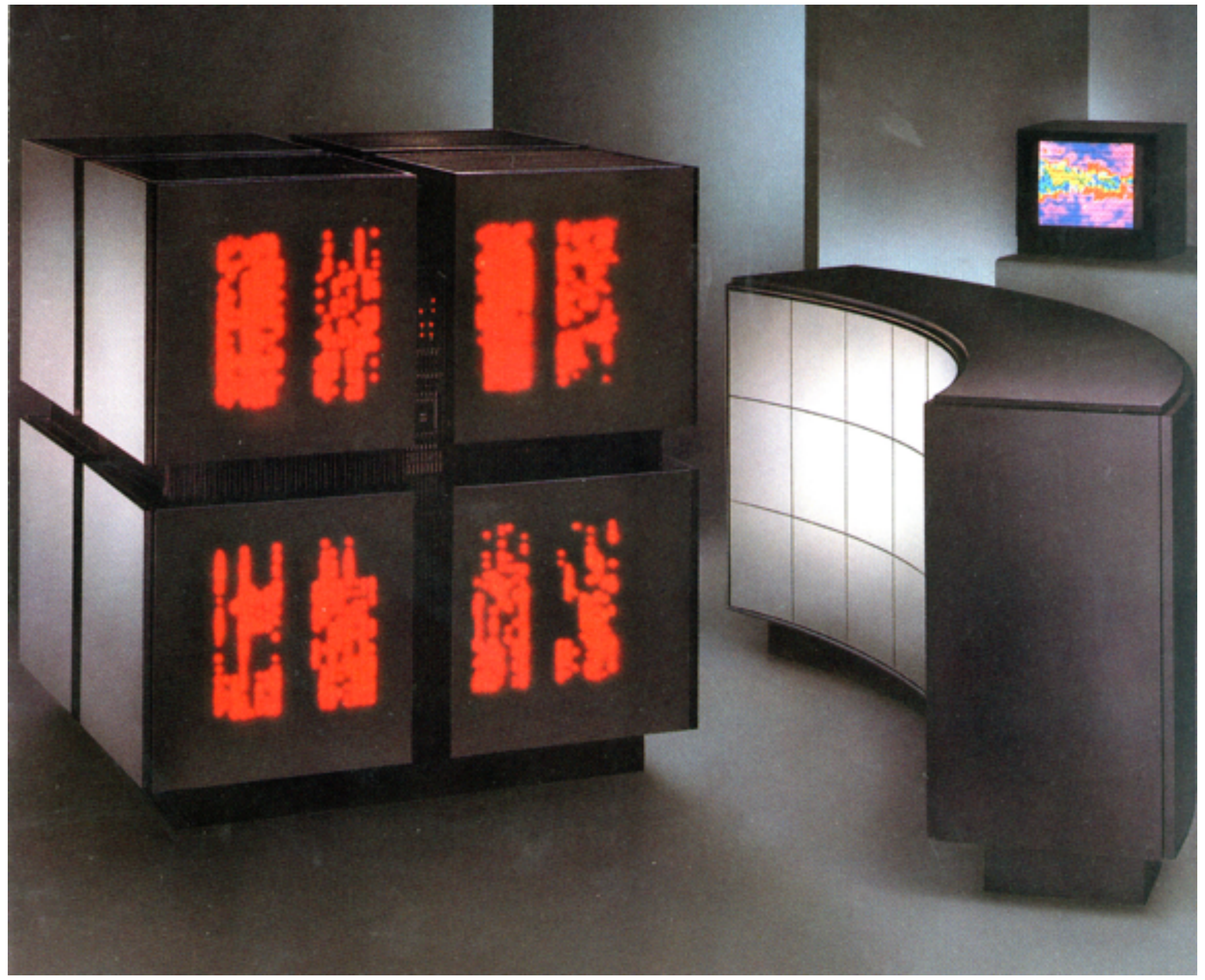
Art for Gordon Moore's 1965 article

iPod vending machine, Macy's

https://www.flickr.com/photos/toasty/289027219

In 1975, Danny Hillis was at a conference in New York City Hilton
"In the future computers will be everywhere. There will be more computers than people."*

Co-founder of Thinking Machines 1983



Connection Machine 1
65,536 1-bit processors

He was heckled:

- "What are you going to do with all of them?
  it's not as if you would put one in every doorknob."

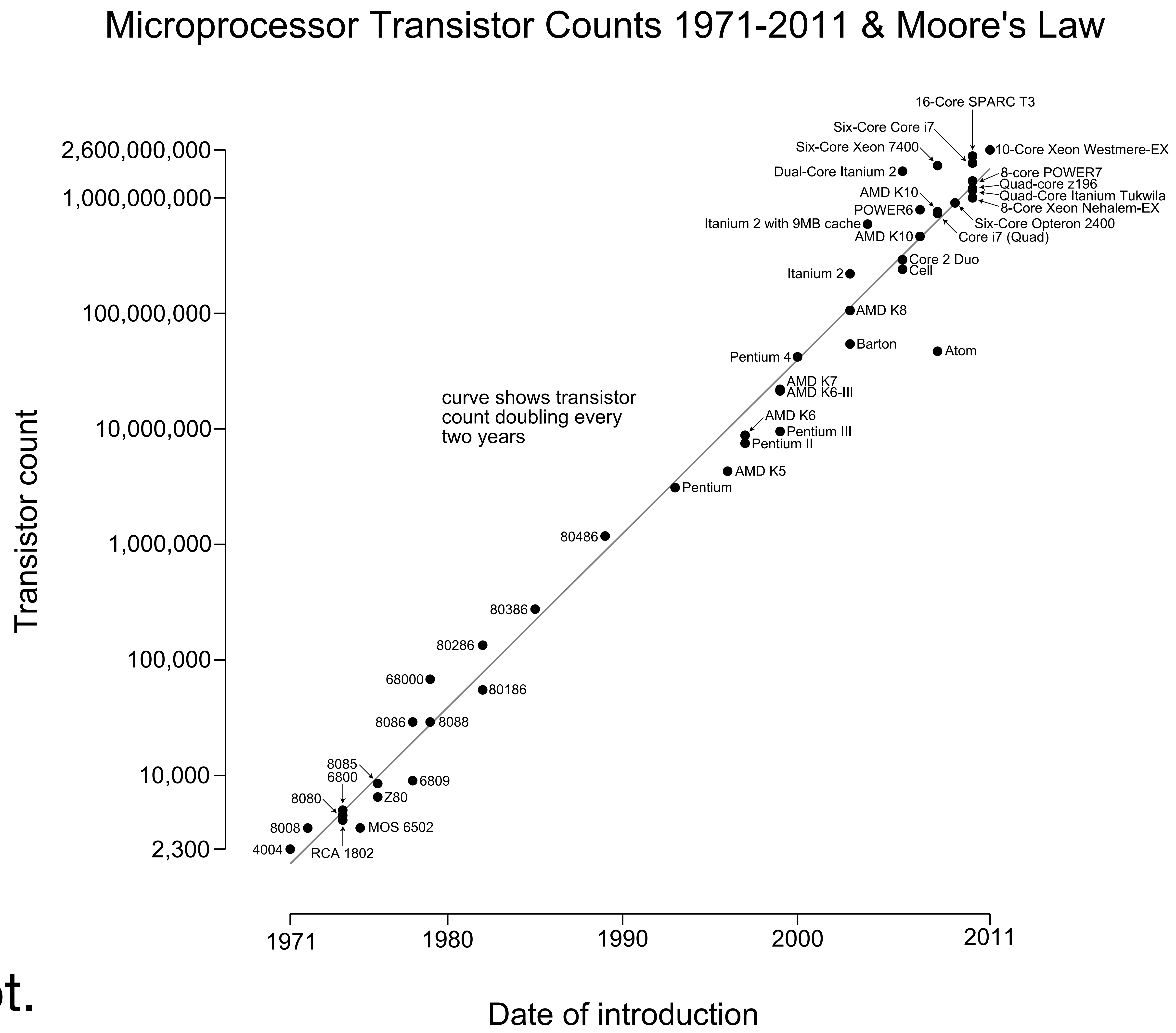
In 1995 Hillis went back Hilton.

- There was a computer in every doorknob

***http://simson.net/clips/1995/95.SJMN.Digital_Decade_MIT_Media_Lab.pdf**

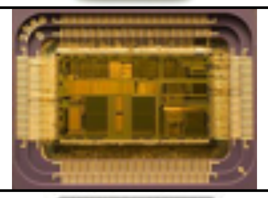
Transistor counts have followed Moore's prediction….

015 https://upload.wikimedia.org/wikipedia/commons/0/00/Transistor_Count_and_Moore%27s_Law_-_2011.svg



… but performance has not.

Between 1971 and 2005, microprocessors doubled in speed ≈every 18 months

Production	Intel	Max Speed	Photo
1971-1981	4004	740 Khz	
1972-1983	8008	800 Khz	
1974-	8080	2Mhz	
1979-1990s	8088	10Mhz	
1982-1990s	80286	25Mhz	
1986-2007	80386	40Mhz	
1989-2007	80486	150Mhz	
1993-	Pentium P5	300Mhz*	

If you wanted your program to run faster… just wait a few months.

- *Note: Pentium chips run different clock speed and bus speed.

Many factors made chips faster.

Faster clock speed

- More instructions per second

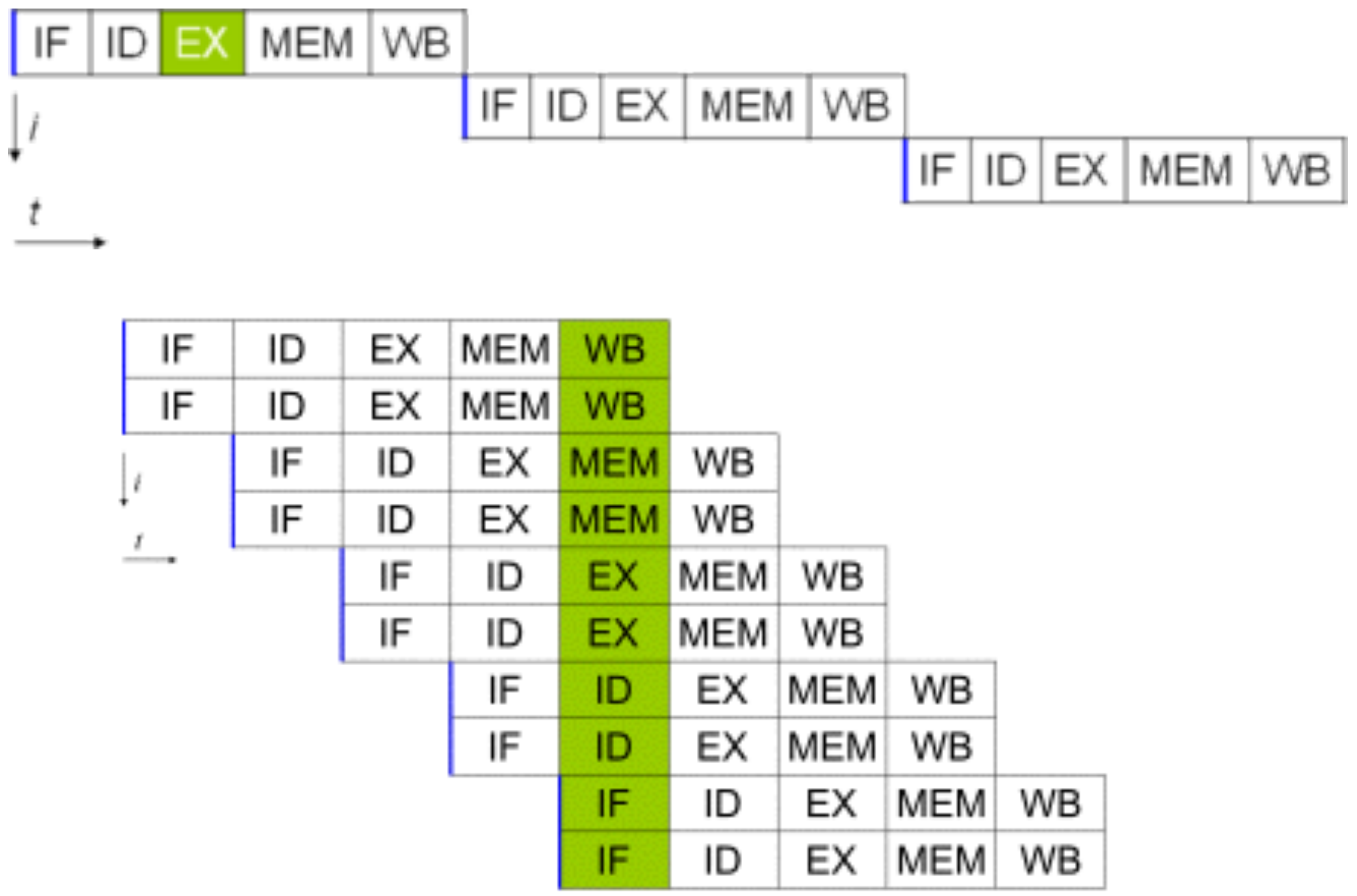
Large caches

- Faster to load and save data to memory

"Execution Optimization" (Instruction level parallelization)

- Pipelining: Overlapping instructions
- Super scalar: Multiple instructions at a time
  - $A \leftarrow B$; $C \leftarrow D$

Intel Core i7-970

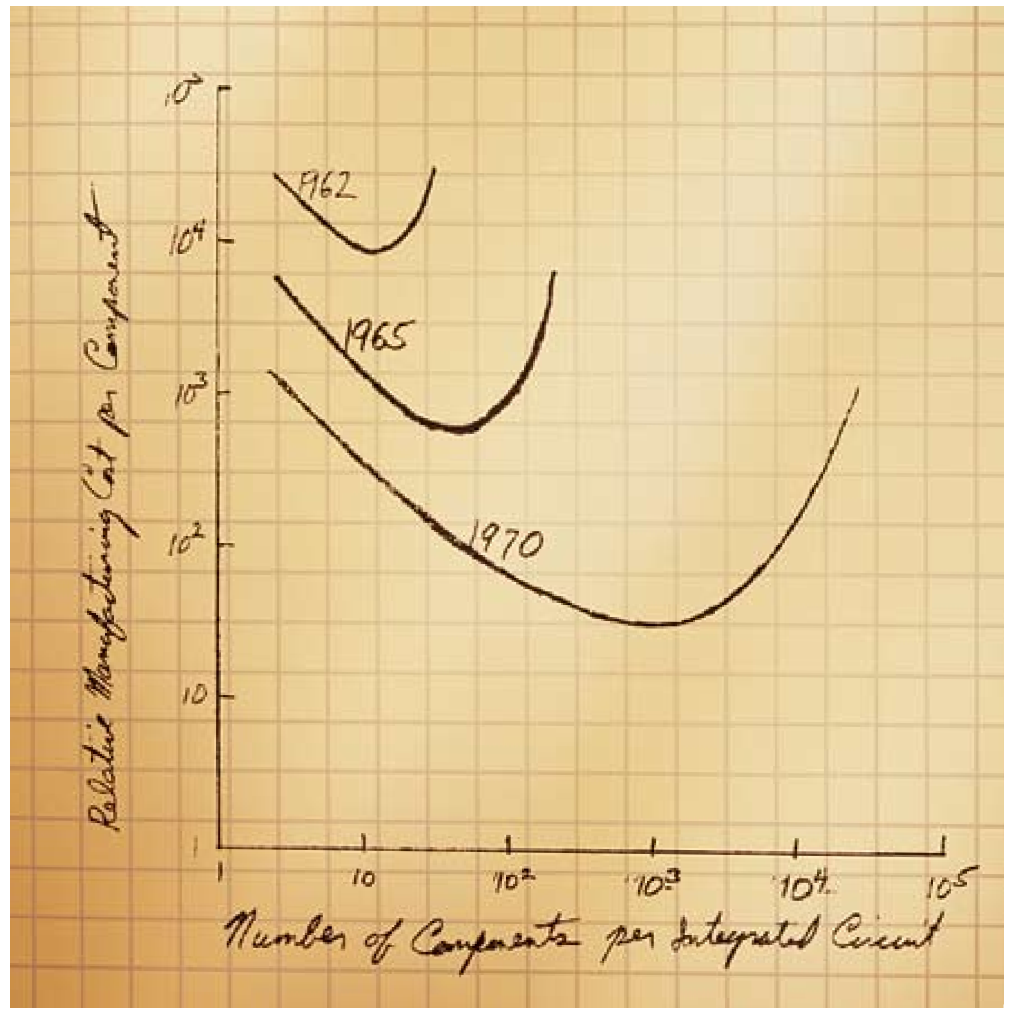


https://en.wikipedia.org/wiki/Central_processing_unit

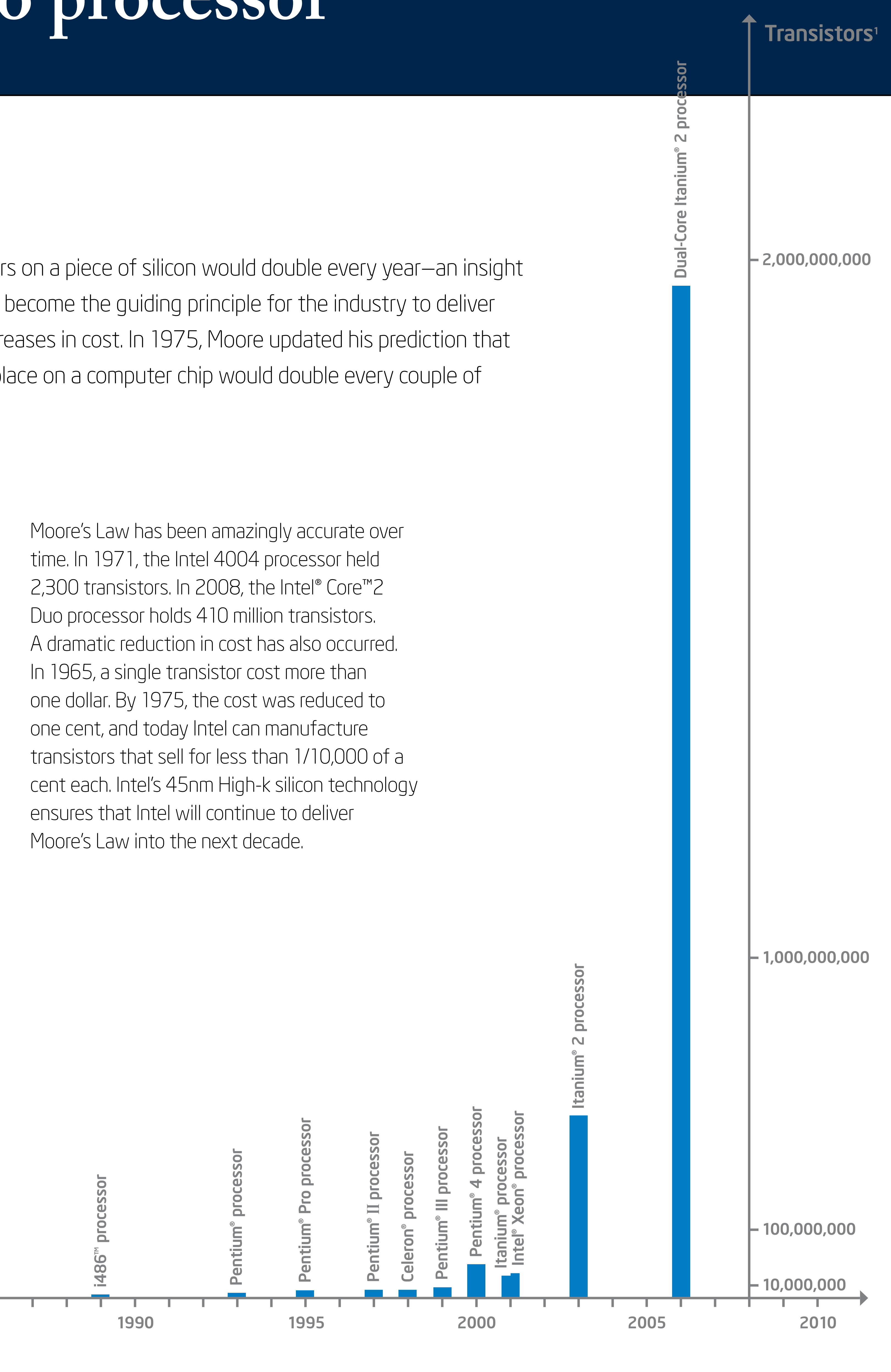
Intel's celebration of the Intel Core 2 Duo processor

Moore's Law

In 1965, Gordon Moore predicted that the number of transistors on a piece of silicon would double every year—an insight later dubbed "Moore's Law." Intended as a rule of thumb, it has become the guiding principle for the industry to deliver ever more powerful semiconductor chips at proportionate decreases in cost. In 1975, Moore updated his prediction that the number of transistors that the industry would be able to place on a computer chip would double every couple of years. The original Moore's Law graph is shown here.



Moore's Law has been amazingly accurate over time. In 1971, the Intel 4004 processor held 2,300 transistors. In 2008, the Intel® Core™2 Duo processor holds 410 million transistors. A dramatic reduction in cost has also occurred. In 1965, a single transistor cost more than one dollar. By 1975, the cost was reduced to one cent, and today Intel can manufacture transistors that sell for less than 1/10,000 of a cent each. Intel's 45nm High-k silicon technology ensures that Intel will continue to deliver Moore's Law into the next decade.



http://www.intel.com/Assets/PDF/General/308301003.pdf

“The Free Lunch Is Over”
Herb Sutter, Dr. Dobb’s Journal, 30(3), March 2005

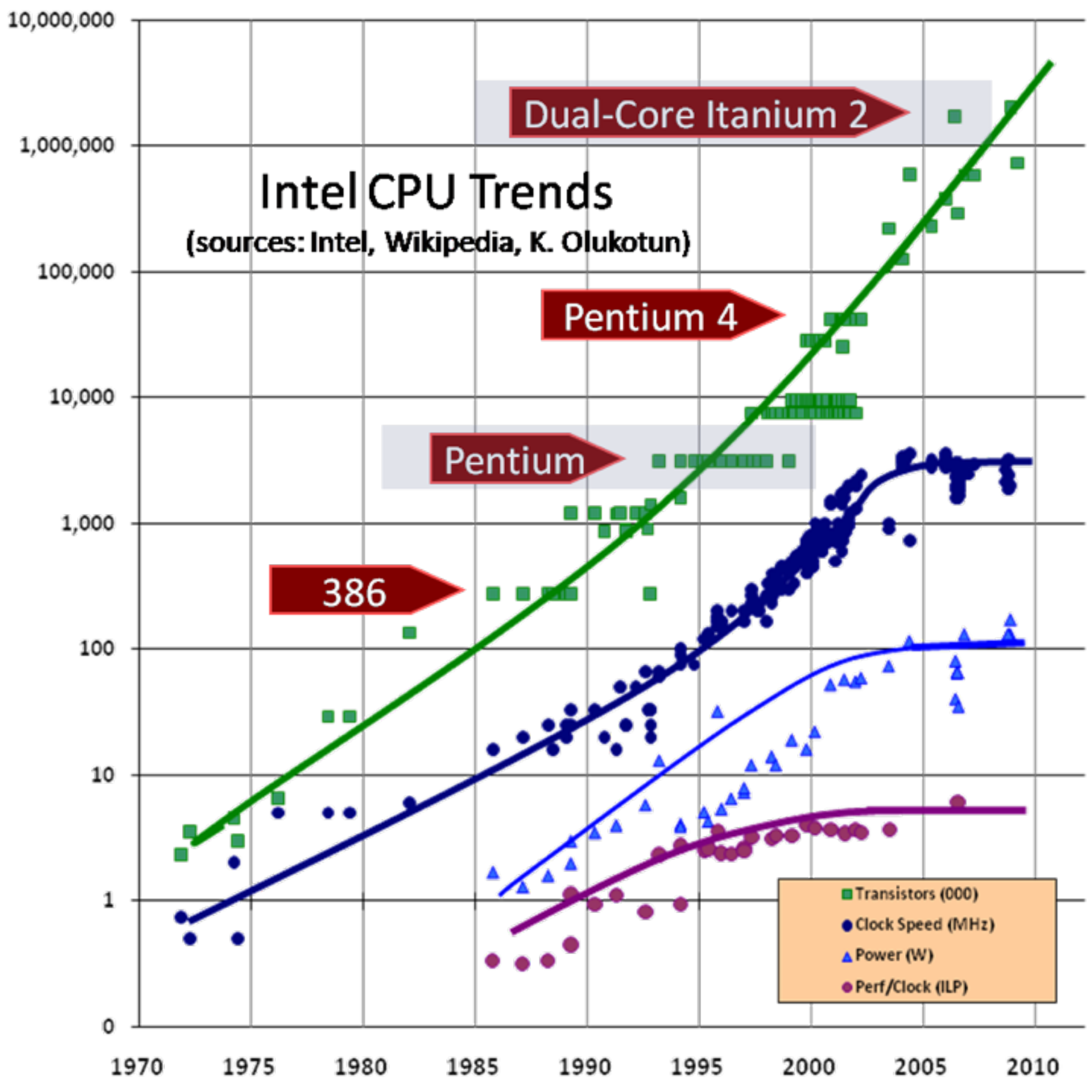
“Your free lunch will soon be over. What can you do about it? What *are* you doing about it?

By 2005, the tricks for making computers run faster had stopped working.

- Too much complexity
- Too much heat

That’s why today, 10 years later, “fast” microprocessors are still running at 1-3Ghz…

—*they just have more cores.*



http://www.gotw.ca/publications/concurrency-ddj.htm

Multicore, Multithreading, Hyperthreading: Instead of faster computers, have more computers!

Instead of running a single computer faster, we use the extra transistors to run multiple computers (“cores”) on the same physical device.

Multithreading: multiple execution threads in a single program.

- Requires compiler and language support.
  - *POSIX threads*
  - *Java “Thread” class and “Runnable” interface.*

Hyperthreading: Intel approach to run 2 threads on a single core:

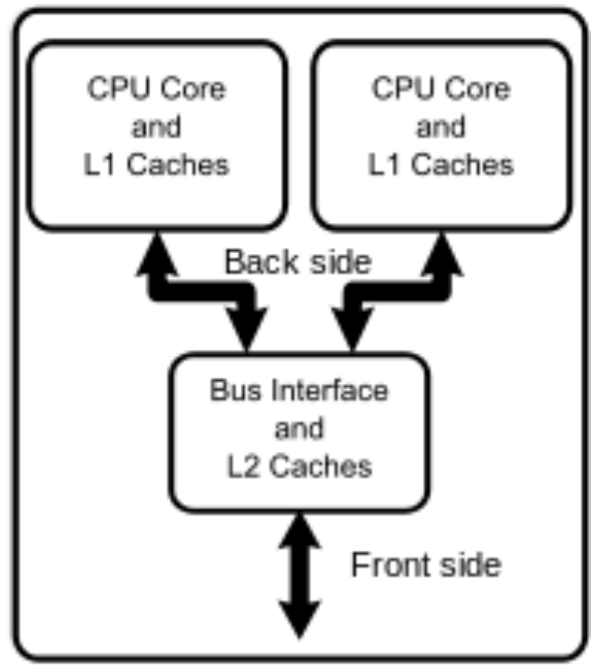
- One ALU (Arithmetic Logic Unit) (expensive part)
- Two sets of registers (A, B, C, D, SP, PC, etc.)
- One Cache

Multicore: two (or more) CPUs on a single chip

- Each core has: ALU, registers, L1 Cache.
- Shared: L2 cache & bus interface

Multiprocessor: two (or more) chips on a motherboard

- AMD Opteron 6000 Series supports 64 core servers (4 x 16)
- MSRP: <$10,000 with 1TB of DDR3 RAM

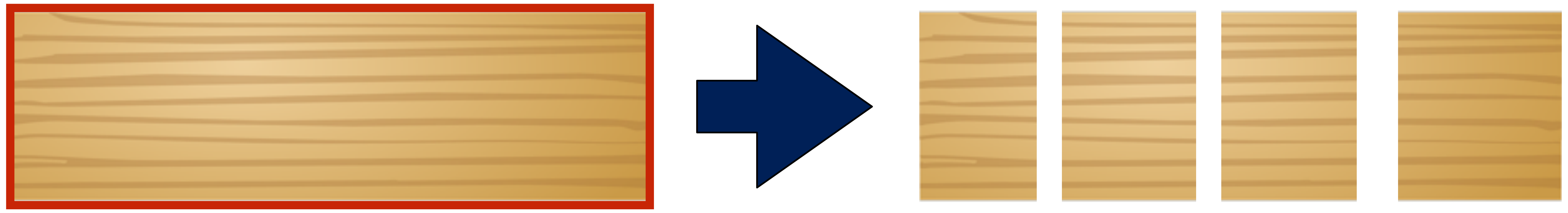


Generic dual-core processor
https://en.wikipedia.org/wiki/Multi-core_processor

i860 from 1989! 3 CPUs
https://en.wikipedia.org/wiki/Intel_i860

The basic idea of parallelism: divide and conquer

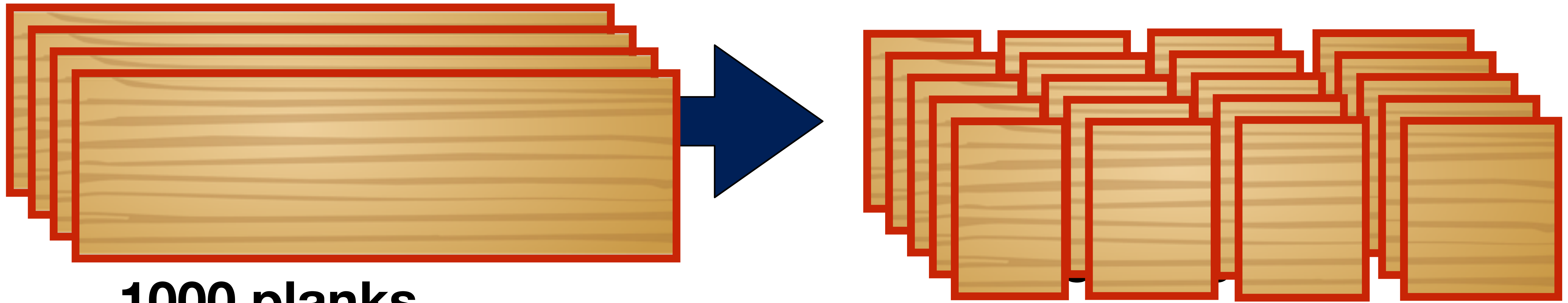
A house with 1000 planks of wood that need to be cut into quarters:→



1 planks

4 segments in 60 seconds

table saw:
1 cut every 20 seconds



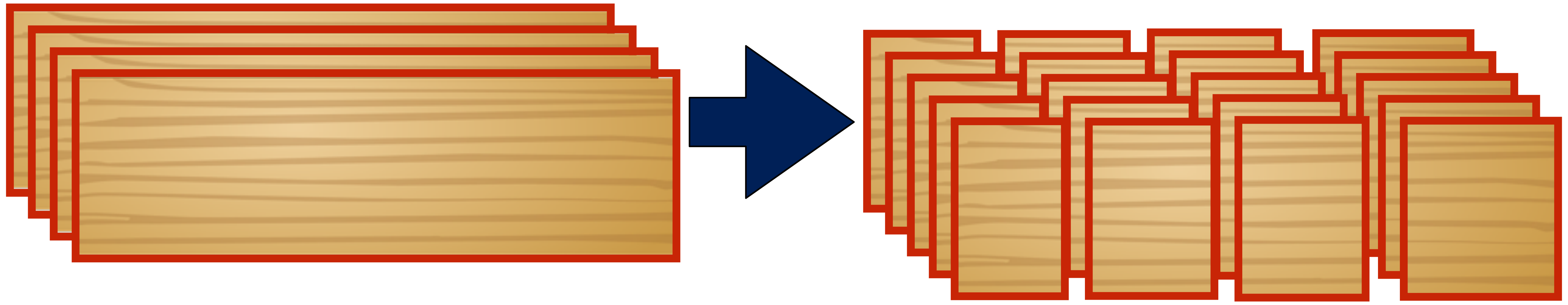
1000 planks

4000 segments in 60,000 seconds

3000 cuts x (20 sec/cut) = 60,000 seconds

You could get a faster saw… but saws only go so fast.

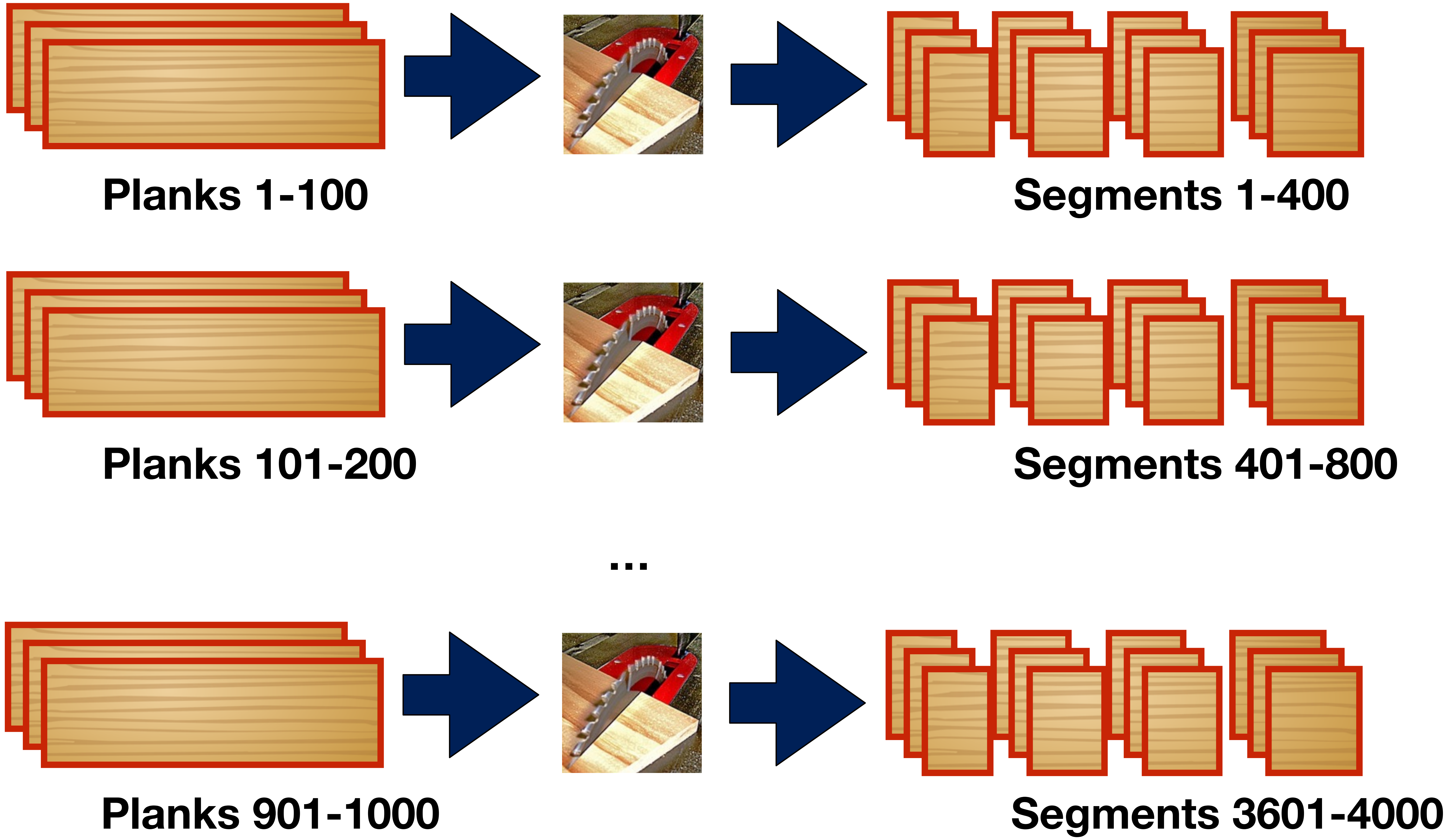
Use an industrial saw mill … 1 cut every 4 seconds (5x faster)



1000 planks **4000 segments in 12,000 seconds**

3000 cuts x (4 sec/cut) = 12,000 seconds

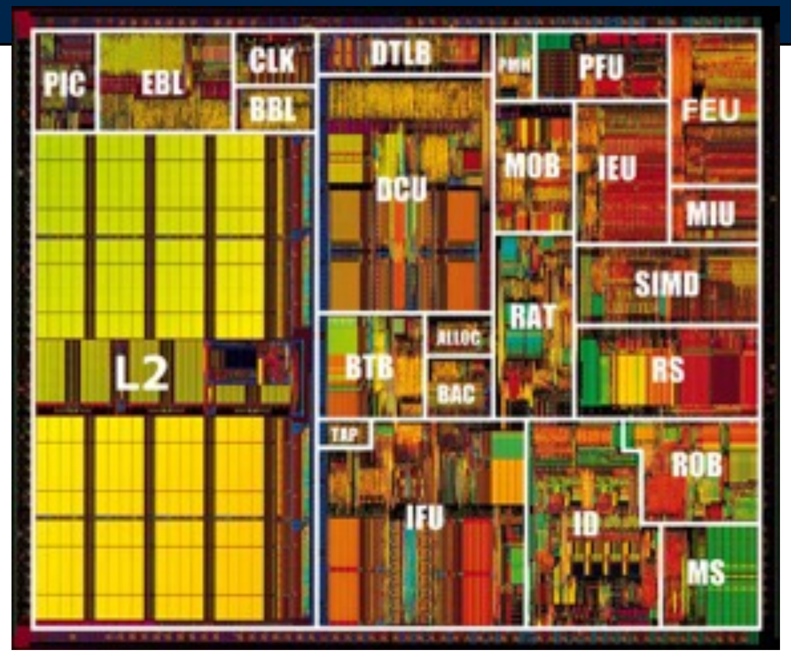
Instead, you could use 100 table saws in parallel



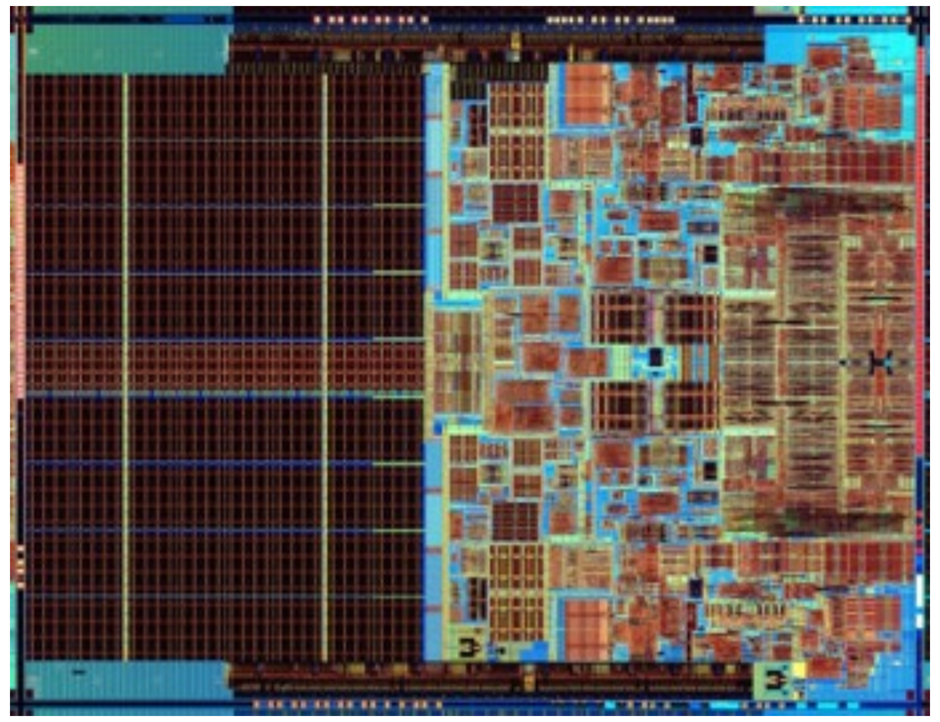
Total throughput: 100 cuts/20 seconds = 5 cuts/sec = 0.2 sec/cut
3000 cuts x (0.2 sec/cut) = 600 seconds
An “embarassingly parallel” problem

Modern computers work the same way.

Intel Pentium 3: 1 core

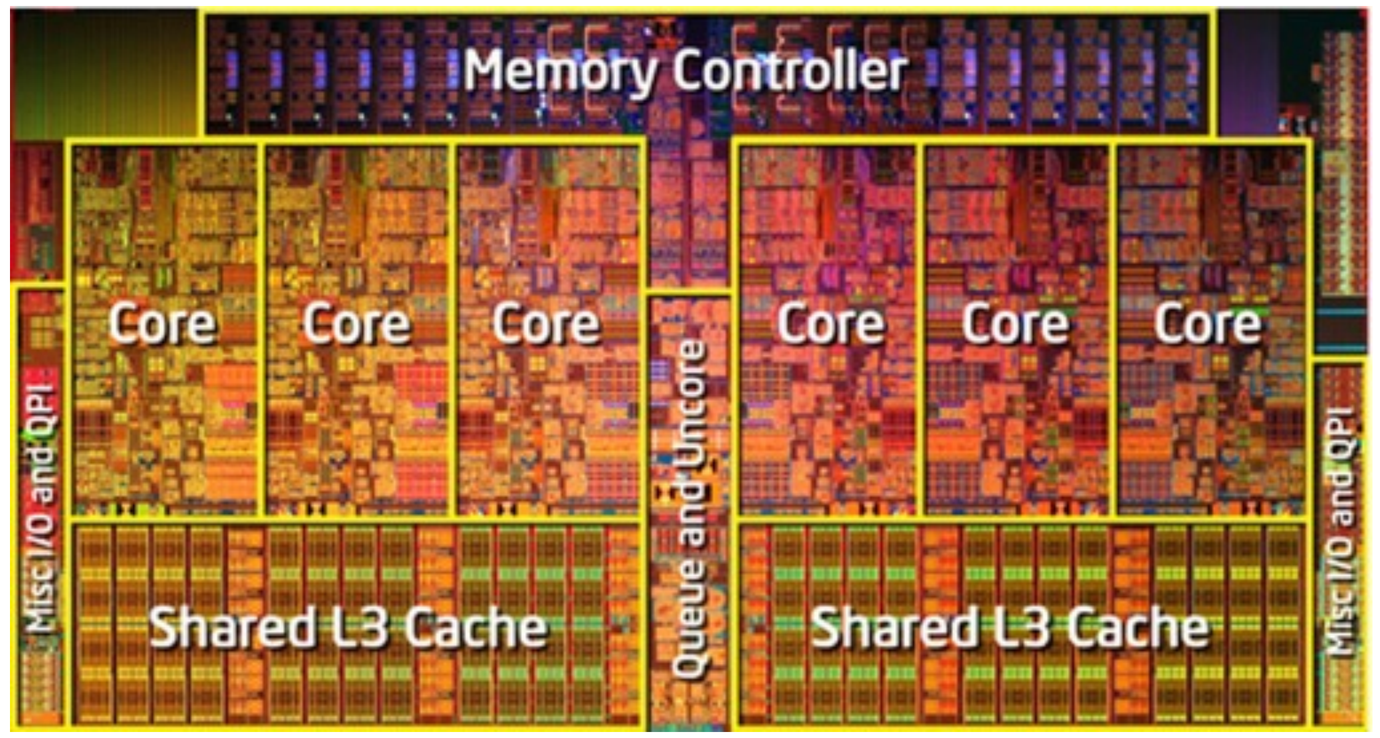


Intel Core Duo: 2 cores



Intel i7-970

6 cores!



More cores
let the computer
do more work
at the same time.

The cores share
RAM and I/O.

Operating systems have two models for using multiple cores: multithreading & multiprocessing

Process abstraction:

Thread of execution:	**~~~~~ TID 1**
Memory Map:	**2GB RAM**
System Resources:	**Open files & libraries**

Process 123

Multithreading:

Thread of execution:	**~~~~~ TID 2**
Thread of execution:	**~~~~~ TID 3**
Thread of execution:	**~~~~~ TID 4**
Memory Map:	**2GB RAM**
System Resources:	**Open files & libraries**

Process 124

Multiprocessing

Thread of execution:	**~~~~~ TID 10**
Memory Map:	**2GB RAM**
System Resources:	**Open files & libraries**

Process 200

Thread of execution:	**~~~~~ TID 11**
Memory Map:	**2GB RAM**
System Resources:	**Open files & libraries**

Process 201

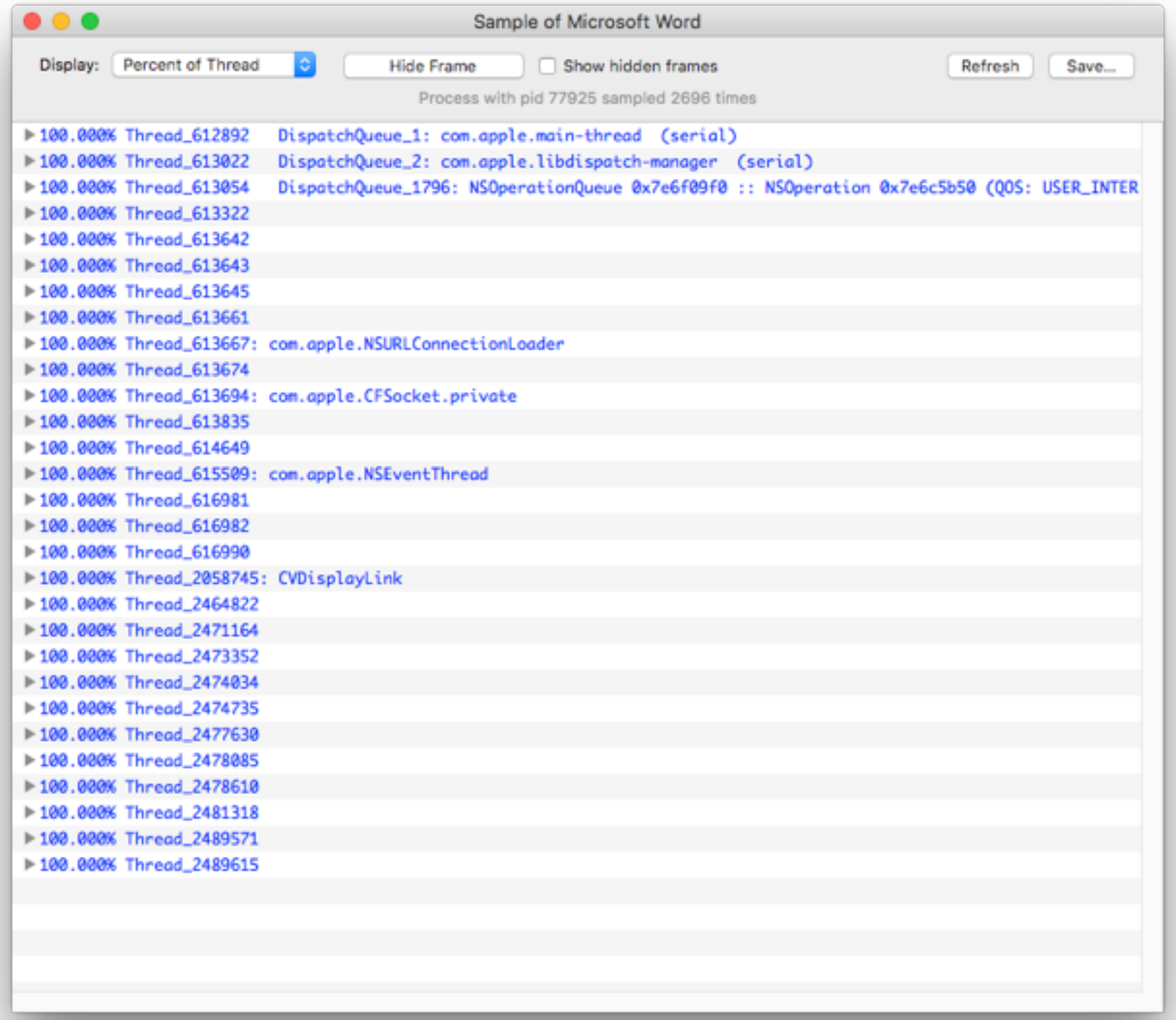
Thread of execution:	**~~~~~ TID 12**
Memory Map:	**2GB RAM**
System Resources:	**Open files & libraries**

Process 202

Programmers have two basic approaches to utilizing multiple cores:
1. Do different things in each thread.

Different things in each thread: Microsoft Word

- Microsoft word on my Mac runs 28 different threads.
  - *Keyboard & mouse events*
  - *Spell checking*
  - *Document formatting*
  - *Grammar Checking*



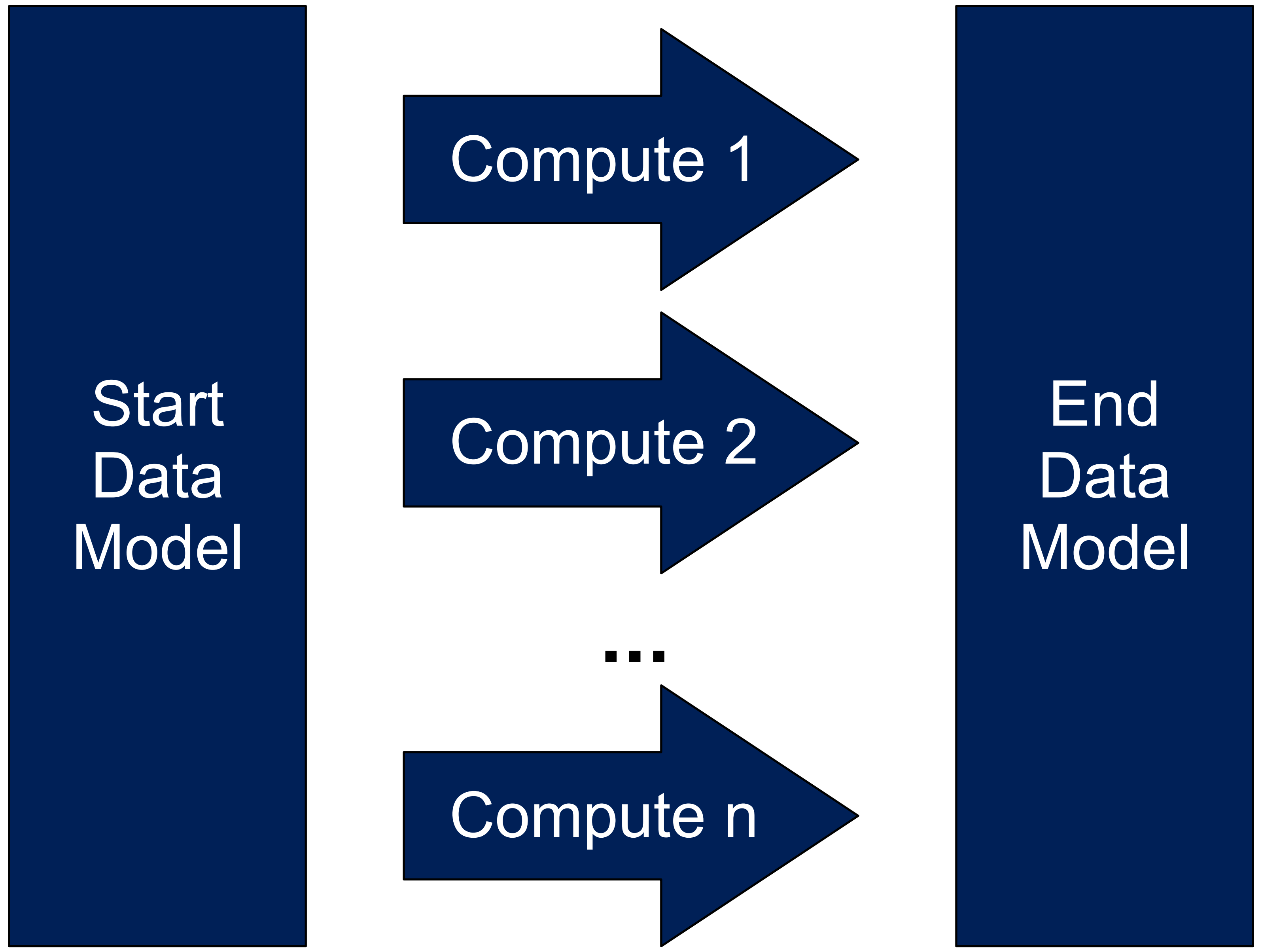
Pros:

- Threads can do a lot of different things.
- Takes advantage of multi-core processors

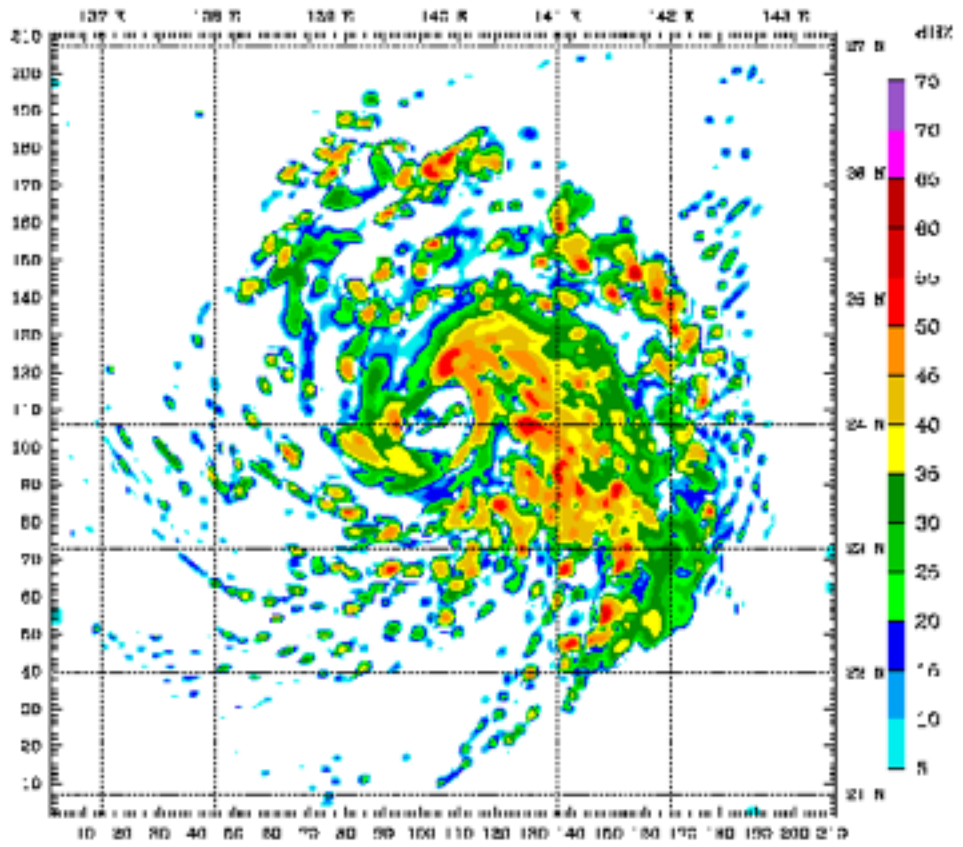
Cons:

- Hard to write & debug.
- Each task is limited in how fast it can go.

This works much better for "massive data*" problems.



https://en.wikipedia.org/wiki/Ray_tracing_(graphics)

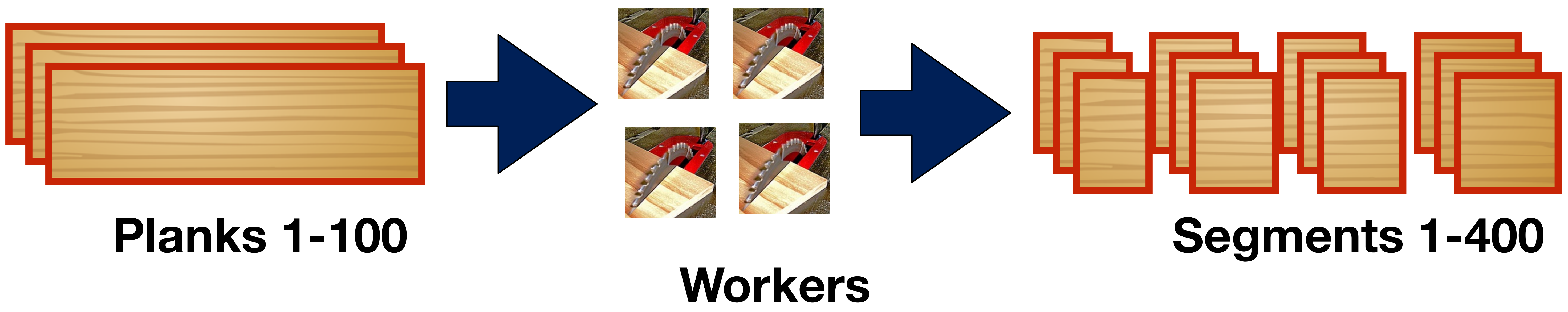


https://en.wikipedia.org/wiki/Weather_Research_and_Forecasting_Model

* "Embarrassingly Parallelizable problems."

Parallelizing code is relatively easy to do on a single computer.

Most languages provide mechanisms for processing a block of data with a “worker pool” and combining the results:



Examples:

- Intel’s Thread Building Blocks
- Fortran, C and C++: MPI & OpenMPI
- Python Multiprocessing module

Multithreading in Python:
import threading

Threading runs multiple Python functions in the same process:

Main thread: __main__
Thread a: foo(“a”)
Thread b: foo(“b”)
Thread c: bar(1,2,3)

Process A

Called functions can be the same or different

Advantages:

- Each thread can run on its own core!
- Each thread has access to all of the memory & state as the main thread.
- Low cost to start up.
- Easier communication between threads
    - —*Lock-free access to read-only data structures.*

Disadvantages:

- Python’s “Giant Lock” limits concurrency.
- Locking may be required when updating complex data structures

Python multi-threading cont.

```python
import threading,time

class CountingThread(threading.Thread):
    def __init__(self,name,high):
        super(CountingThread,self).__init__()
        self.name = name
        self.high = high
        self.count = 0

    def run(self):
        for i in range(0,self.high):
            print("{}: {}".format(self.name,i))
            time.sleep(.5)
            self.count += 1

if __name__=="__main__":
    foo = CountingThread("foo",5)
    foo.start()
    bar = CountingThread("bar",5)
    bar.start()
    foo.join()                  # Waits for completion
    bar.join()
    print("foo.count:{}  bar.count:{}"
        .format(foo.count,bar.count))
```

Output Run 1:

```
$ python demo_threading.py
foo: 0bar: 0

foo: 1
bar: 1
foo: 2
 bar: 2
foo: 3
bar: 3
foo: 4
bar: 4
foo: 5
foo.count:5  bar.count:5
```

Output Run 2:

```
$ python demo_threading.py
foo: 0
 bar: 0
bar: 1foo: 1

bar: 2
 foo: 2
foo: 3
 bar: 3
bar: 4
foo: 4
foo.count:5  bar.count:5
```

Multithreading in Python:
import multiprocessing

Multiprocessing runs multiple Python functions in different processes:

Main thread: __main__

Process A

Main thread: foo("a")

Process B

Advantages:

- Separate Python interpreter for each thread.
- Better concurrency — no giant lock.

Disadvantages:

- Limited communication between each process.

Python's Multiprocessing module

```
import multiprocessing,os

def worker(val):
    return "worker {} PID {}".format(val,os.getpid())

if __name__=="__main__":
    pool = multiprocessing.Pool(processes=4)
    result = pool.map(worker,range(0,16))
    print(result)
```

Output*

```
$ python demo_multiprocessing.py
['worker 0 PID 69172', 'worker 1 PID 69174', 'worker 2 PID 69173', 'worker 3 PID 69175’,
 'worker 4 PID 69174', 'worker 5 PID 69172', 'worker 6 PID 69173', 'worker 7 PID 69175’,
 'worker 8 PID 69174', 'worker 9 PID 69172', 'worker 10 PID 69173','worker 11 PID 69175’,
 'worker 12 PID 69174','worker 13 PID 69172','worker 14 PID 69173','worker 15 PID 69175']
```

***slightly reformatted**

OpenMP has a similar mechanism

C with OpenMP:

```c
#include <omp.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main(int argc,char **argv)
{
#pragma omp parallel
    {
        // Code inside this region runs in parallel.
        printf("Greetings from thread %d process %d\n",
               omp_get_thread_num(),getpid());
    }
    exit(0);
}
```

Run:

```
$ g++-mp-4.9 -o demo_openmp -fopenmp demo_openmp.cpp
$ ./demo_openmp
Greetings from thread 1 process 75836
Greetings from thread 0 process 75836
Greetings from thread 2 process 75836
Greetings from thread 3 process 75836
```

Multithreading pays off — but so do better implementations.

Microsoft's Revolution Analytics examined speedup of R:

Calculation	Size	Command	R 2.9.2	Revolution R (1 core)	Revolution R (4 cores)
Matrix Multiply A'*A	10000x5000	B <- crossprod(A)	243 sec	22 sec	5.9 sec
Cholesky Factorization	5000x5000	C <- chol(B)	23 sec	3.8 sec	1.1 sec
Singular Value Decomposition	5000x5000	S <- svd (A,nu=0,nv=0)	62 sec	13 sec	4.9 sec
Principal Components Analysis	10000x5000	P <- prcomp(A)	237 sec	41 sec	15.6 sec

Notes:

- R2.9.2 ➜ Revolution R speedup: moving from R's built-in BLAS (Basic Linear Algebra Subprogarms) to Intel's Math Kernel Libraries. (6 to 11 fold speedup)
- Revolution R (1core) ➜ Revolution R (4 cores) speedup is ≈ 4x.
  - http://blog.revolutionanalytics.com/2010/06/performance-benefits-of-multithreaded-r.html
  - https://software.intel.com/en-us/intel-mkl/

Modern data centers have many computers, each with many processors

Hardware failures are a BIG deal!

Say a typical computer fails in 5 years.

What happens if you have 1000 computers in a data center?

Key challenges:

- Solve a single problem on multiple systems?
- Keeping data consistent.
- Responding to hardware failures.

The big data “Map Reduce” paradigm extends parallelization do multiple machines and more complex problems.

This is the basic idea of modern “big data” analysis.

A “Map” phase that produces intermediate results

A “Reduce” phase that combines the intermediate results into a final value

Start Data Model

Compute 1

Compute 2

...

Compute n

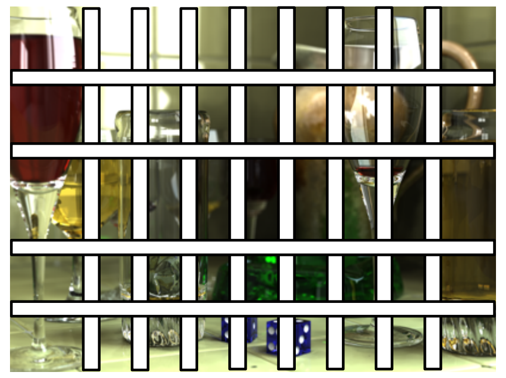
K1, Result 1

K3, Result 3

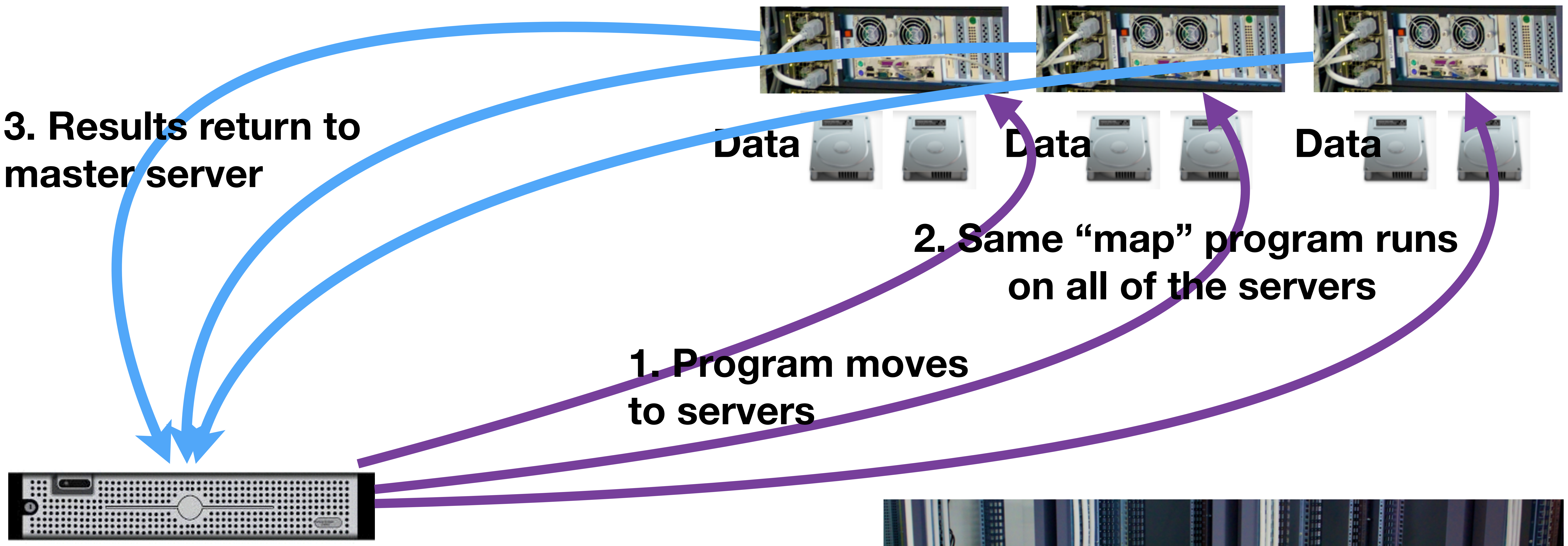
If a computer fails, the job is re-run on another computer

Reduce

End Data Model



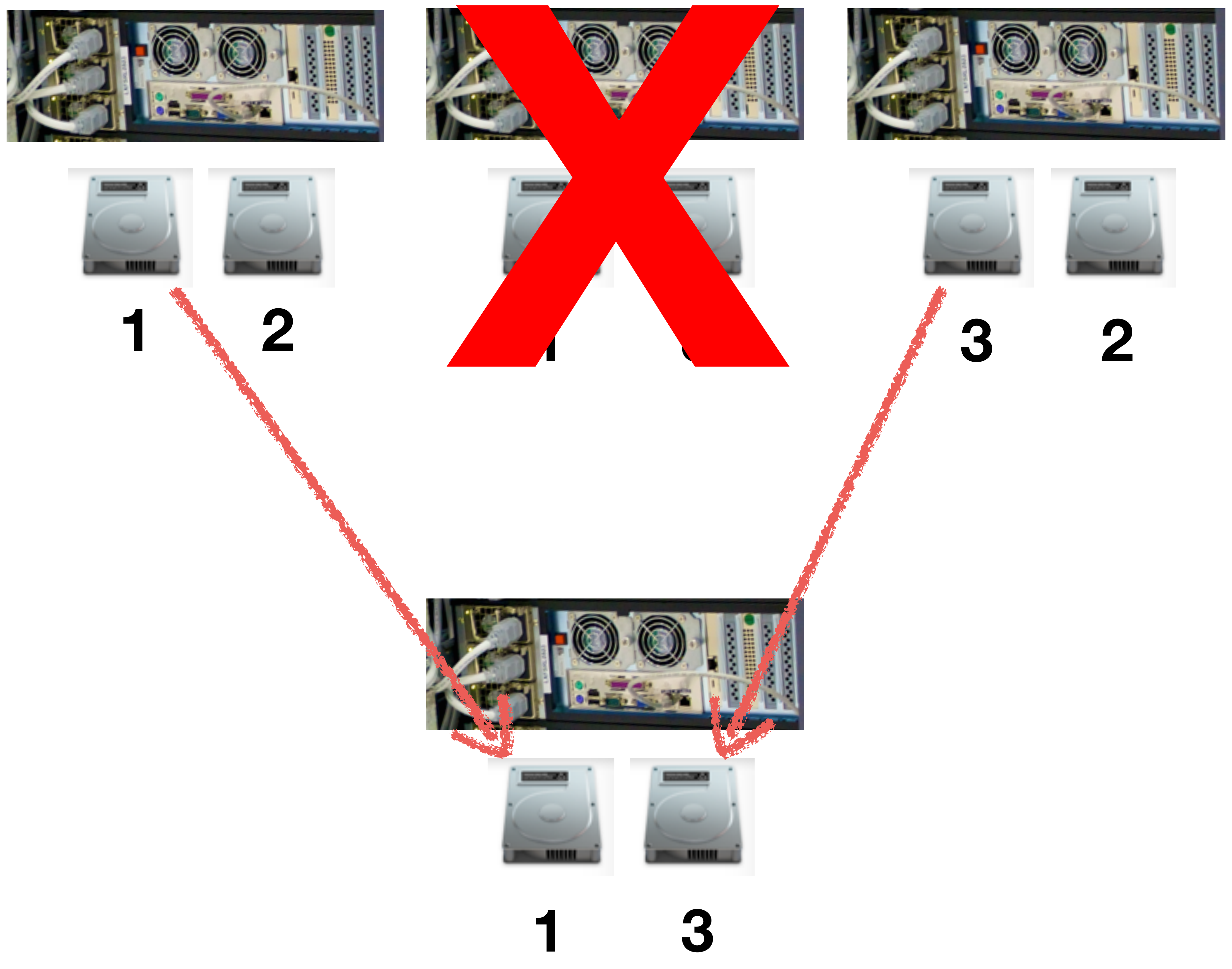
In big data systems, data are stored on each node. The “map” jobs are sent to the nodes.



The data returns in the “reduce” step.

Apache Hadoop makes it each to write, run, and manage these jobs.

Data are stored on multiple computers, so if one system fails, there is a copy



Replication is provided by:

- HDFS
- Amazon S3
- Other redundant file systems

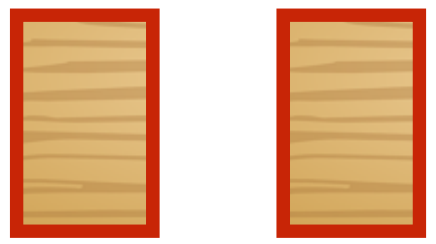
The promise of Hadoop is *linear scaling:*
You process more data by adding more computers.

If it takes 100 computers 10 hours to process 1TB of data…

- Process 1TB of data in 1 hour with 1000 computers
- Process 100TB of data in 10 hours with 10,000 computers
  - *—This is typical of "embarrassingly parallel" problems.*

The catch:

- Some jobs can't be parallelized.
- If it takes 10 people a year to build a house, you can't build the same house in 3.7 days with 1000 people.



If you need more data, just add data nodes.

https://commons.wikimedia.org/wiki/File:Wood-framed_house.jpg

The catch:

- Overhead is 2x, 3x or more.
- RAID systems have overhead of 16% - 20%

Introducing Hadoop and MapReduce

Back in the early 2000s, companies were building bigger and bigger data centers. They needed some way to scale computation.

Companies kept facing the same problems.

Price vs. Reliability:

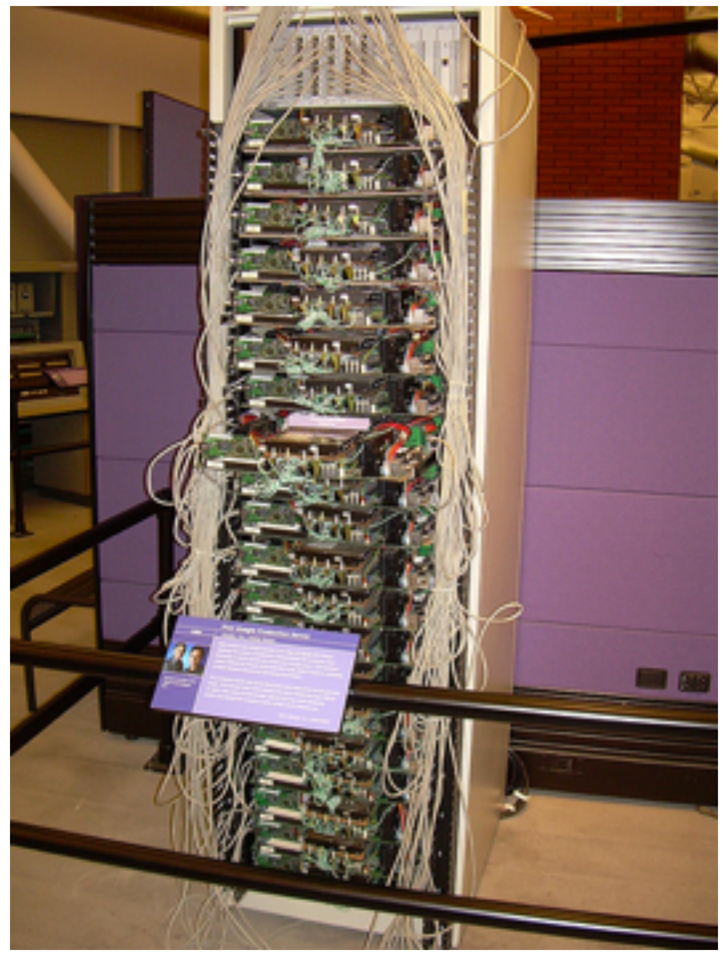
- Cheap machines failed.
- Reliable machines were expensive.

Hardware vs. Software Diversity:

- "Data centers" were designed with computers having similar hardware, but different software configurations.
  - *Hard to keep the system going.*
  - *Hard to install, configure, administer and manager.*

WWW Server	Biz Logic	DB Server

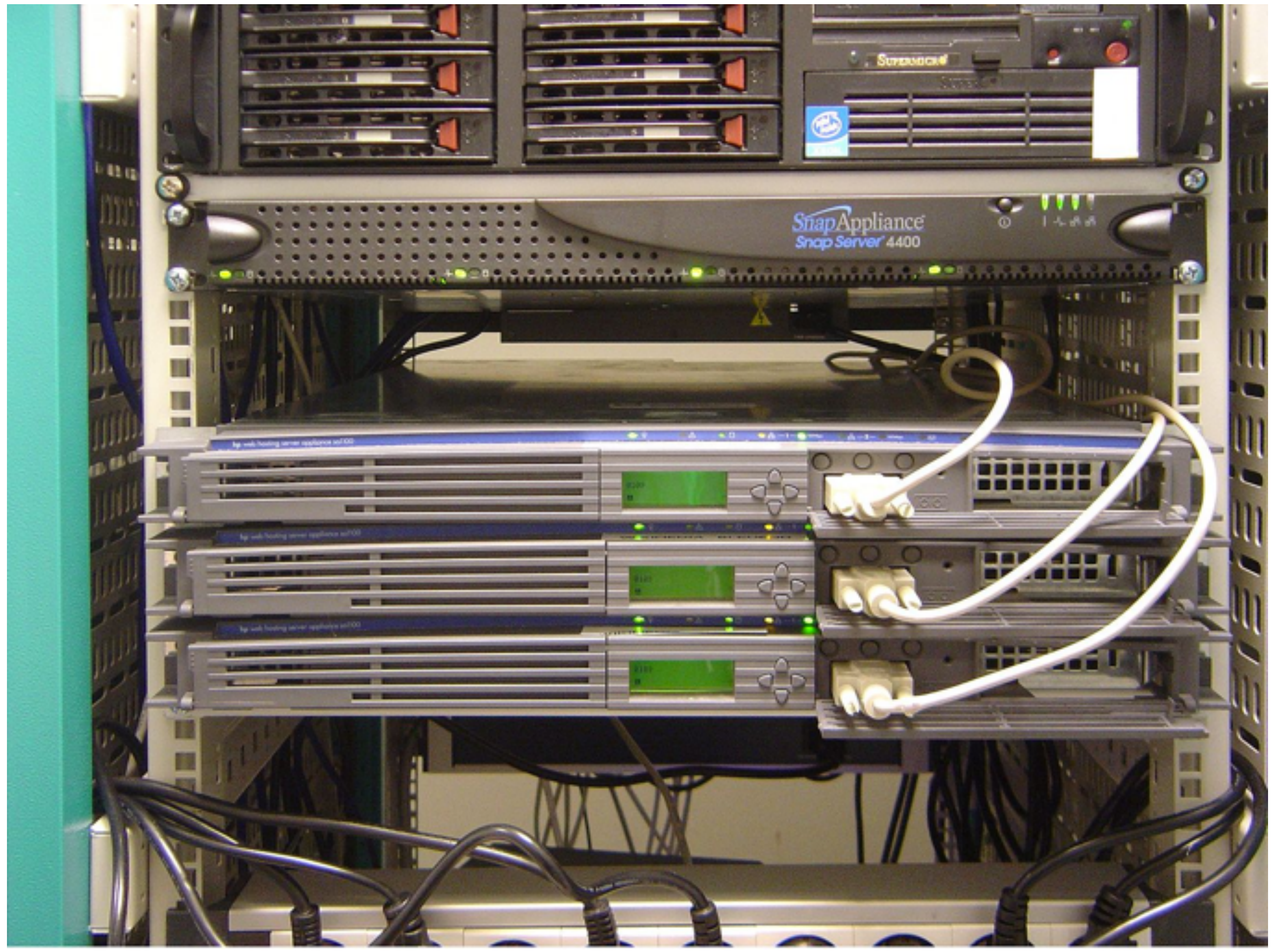
First Google Computer Lego enclosure



First Google Production Server

This is one approach to scaling…
A fast database server and lots of clients

Provides isolation between the front end and the database.



https://en.wikipedia.org/wiki/Data_center

SuperMicro server
6 high-speed SCSI drives
8 core processor?

HP Web Hosting
Server Appliance SA1100 (3)
2 x 100 Mbps ethernet
10 GB hard drive
533 MHz processor
128MB RAM

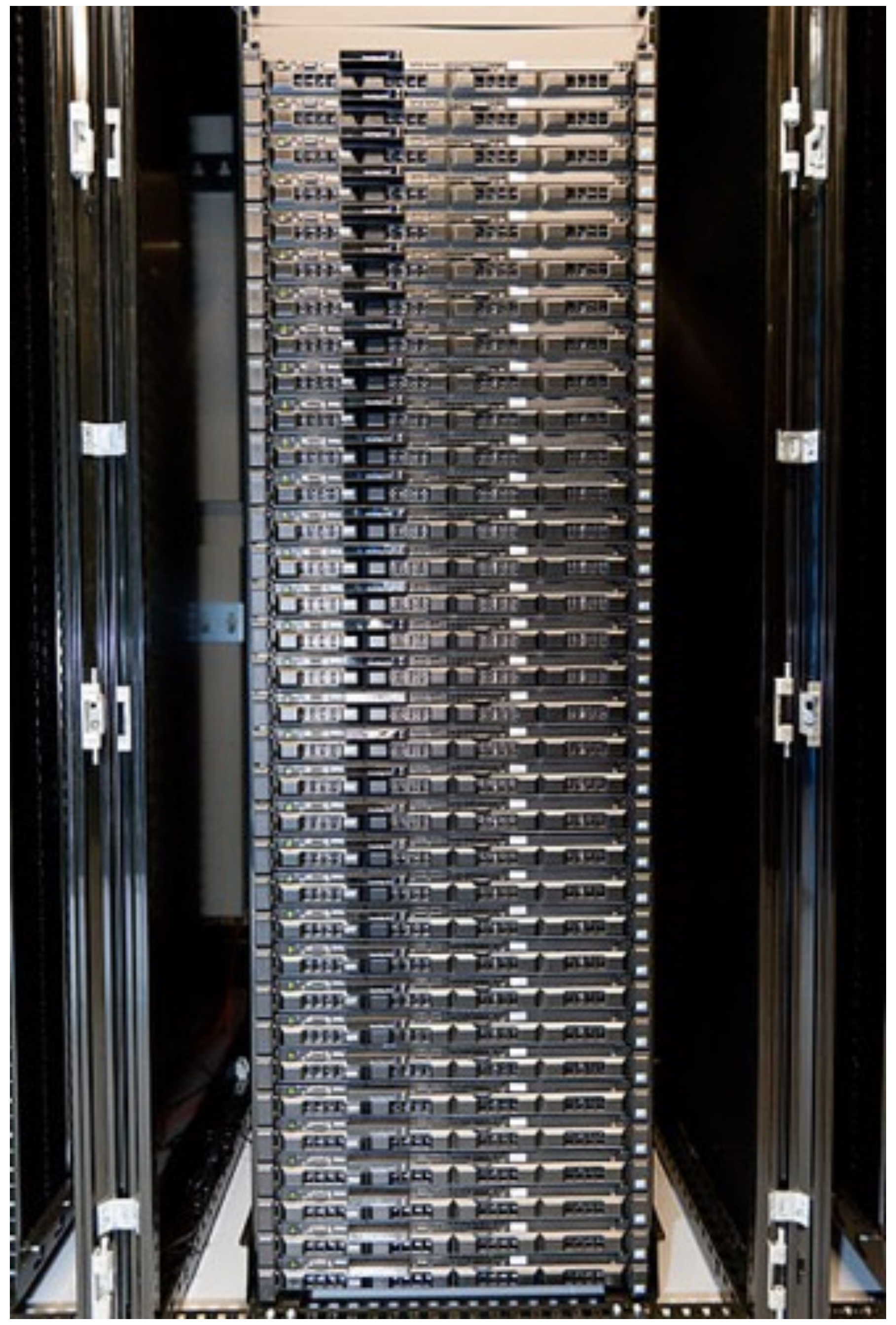
The database is a bottleneck.

An easier approach: identical servers.

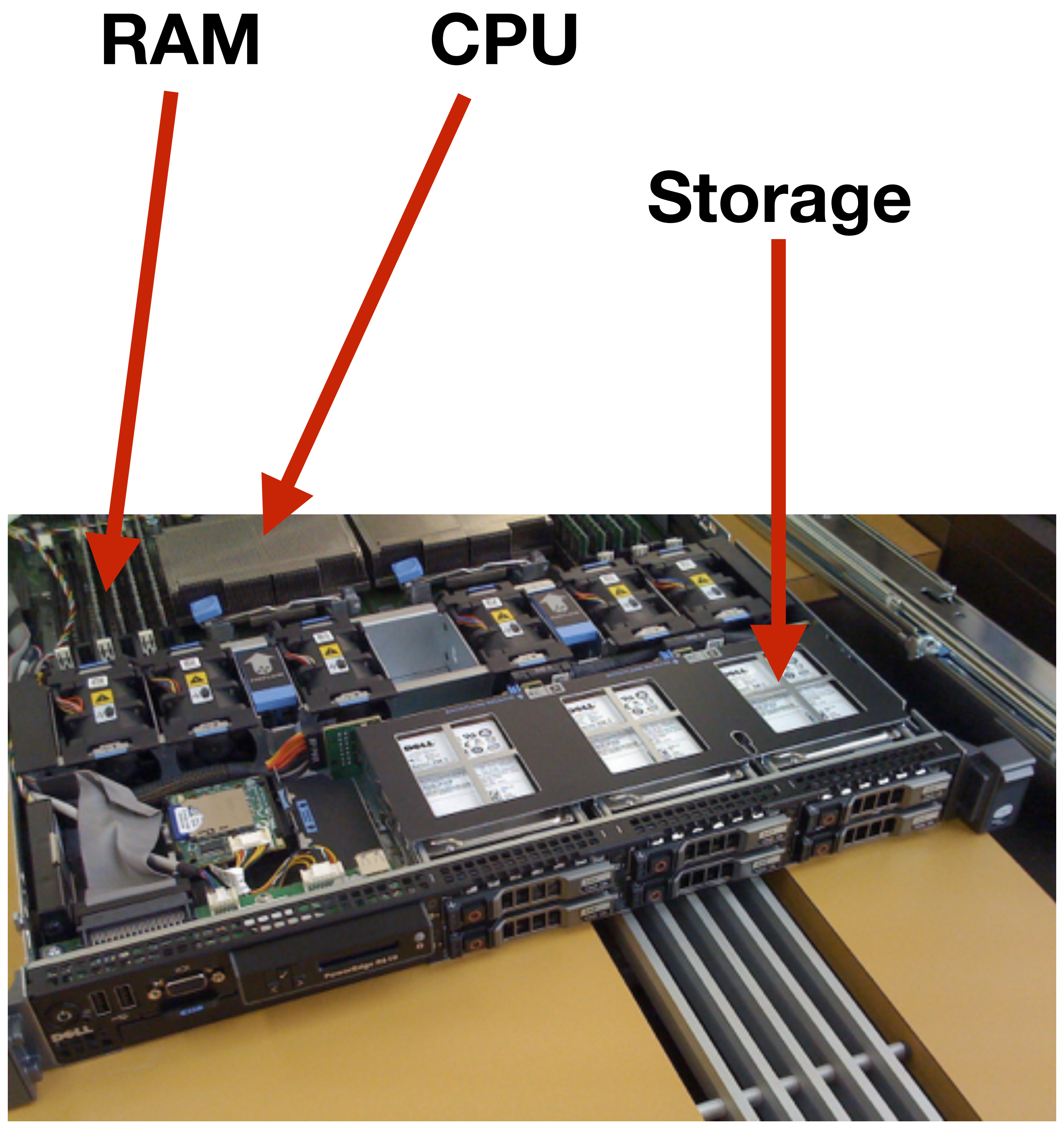
Every computer has same hardware configuration.

Distributed storage.

Distributed computation.



https://commons.wikimedia.org/wiki/File:Wikimedia_Servers-0001_43.jpg



https://en.wikipedia.org/wiki/Dell_PowerEdge

In 2003 & 2004 Google Research published two seminal papers.

Google File System Paper

- How Google stored its information — at scale
- The Google File System
  Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung
  19th ACM Symposium on Operating Systems Principles,
  Lake George, NY, October, 2003.
  http://research.google.com/archive/gfs.html

Google MapReduce

- How Google processed its information — at scale
- MapReduce: Simplified Data Processing on Large Clusters
  Jeffrey Dean and Sanjay Ghemawat
  OSDI'04: Sixth Symposium on Operating System Design and Implementation,
  San Francisco, CA, December, 2004.
  http://research.google.com/archive/mapreduce.html

These papers showed how Google had overcome the scaling problem.

Google File System (GFS) Requirements

Design assumptions:

- System built from many inexpensive components that often fail. These store DATA.
- A high-performance, high-reliability, system. The MASTER stores METADATA.
- Workload consists of two kinds of reads:
  - *large, streaming reads. (typically 1MB of more)*
  - *small random reads. (typically batched by performance-critical applications)*
- Workload consists of two kinds of writes:
  - *large, streaming writes. (sequential, >>1MB)*
  - *small writes at arbitrary locations (infrequent; need not be efficient)*
- Well-defined semantic for multiple clients writing to the same file
- High sustained bandwidth is more important than low latency.
  - *Designed for bulk, batch processing. (building the index, not searching the index.)*

GFS Implementation

Files are divided into fixed-size (**64MB**) *chunks*.

- Each chunk has a unique 64-bit *chunk handle*.
- Each chunk replicated on multiple GFS chunkservers.

Single master:

- Maps filenames to chunks
- Global directory of where each chunk is stored
- Metadata stored in RAM
- Checkpoints to hot backups
- Shadows for read-only access
- Replicates data on node fail

Clients:

- Send filename to master.
- Get chunk handles from master.
- Get chunks from chunkservers.

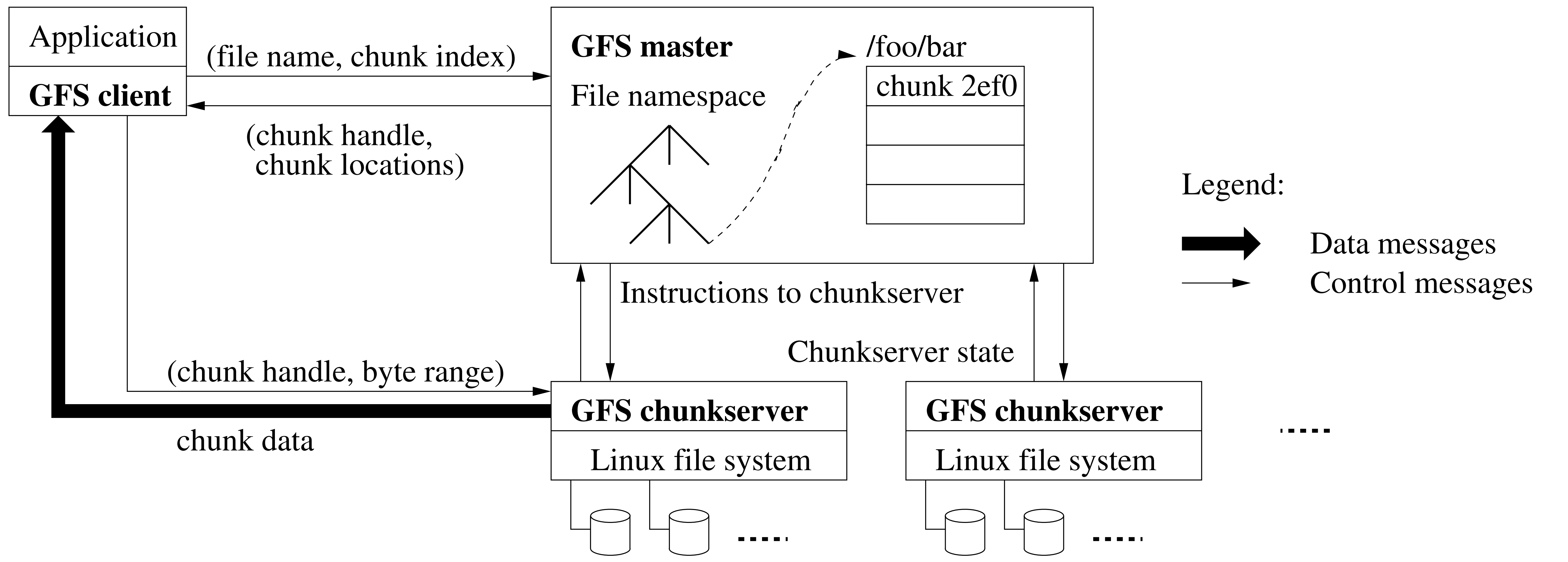
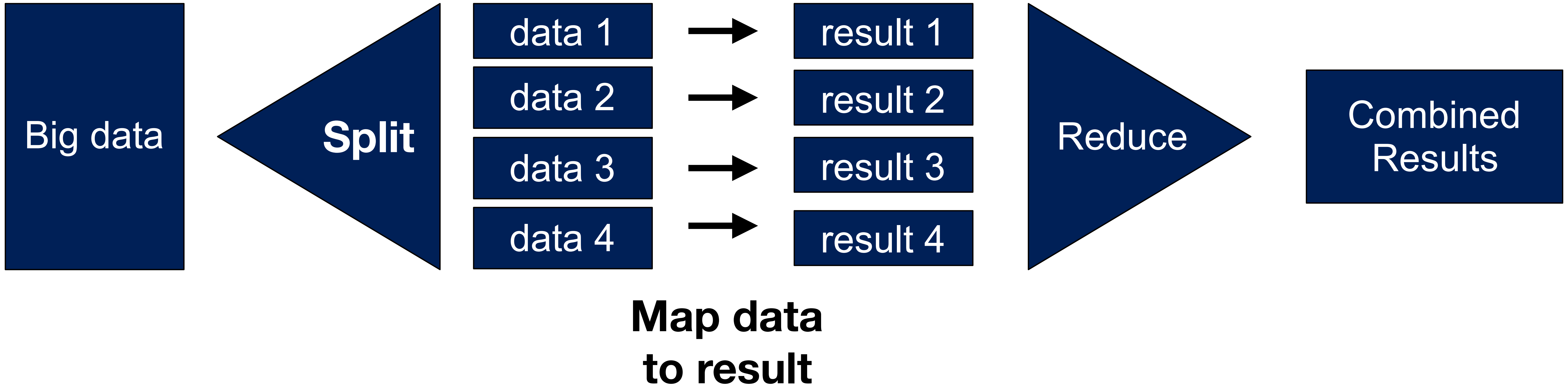


Figure 1: GFS Architecture

Google's MapReduce:
A programming paradigm based on functional programming.

Previous work on "grid" computing was based on the idea of splitting up jobs, performing work in parallel, and combining the work:



MapReduce is an *approach* and *infrastructure* for doing this at scale.

Provides:

- Automatic parallelization and distribution
- Fault-tolerance
- I/O scheduling
- Integrated status and monitoring
- Speculative execution for slow jobs.

MapReduce Limitations

Programmer:

- Everything is a Map or Reduce, but we don't think of problems that way
- No control over the *order* in which map() or reduce() runs.
- Mapper & Reducer must be stateless — can't depend on other map() or reduce() operations.
- No random access to the data
- Hard to implement algorithms that can't be easily partitioned

Performance:

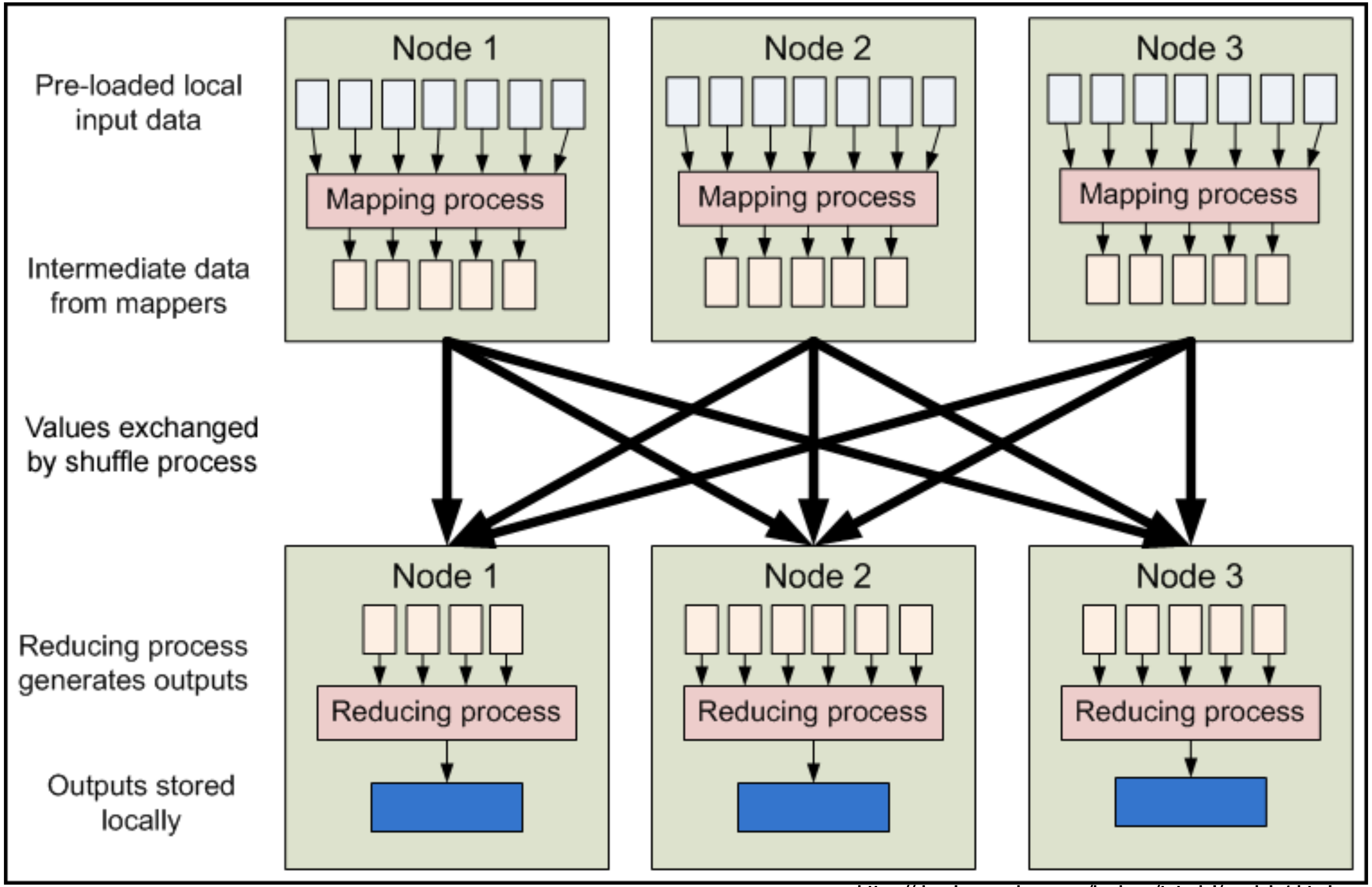
- Data is not indexed
- Finding MIN() or MAX() requires scanning *all the data*.
- No memory-to-memory transfers
  —*Everything must be written back to the disk*
- Entire map() must finish before reduce() starts
- Memory limitations on HDFS "Name node"

Operational:

- High overhead
- Batch processing
- Does not produce immediate answers

https://www.quora.com/What-are-some-limitations-of-MapReduce
http://stackoverflow.com/questions/18585839/what-are-the-disadvantages-of-mapreduce

Diagram from Yahoo! developer tutorial



https://developer.yahoo.com/hadoop/tutorial/module1.html

The MapReduce programming model is based on functional programming.

Input & Output is a set of key/value pairs.

Programmer specifies a mapper:

- map (in_key, in_value) → list(out_key, intermediate_value)
- reduce (out_key, list(intermediate_values)) → list(out_value)

Compare with Python:

```
>>> def square(x): return x*x
...
>>> a = range(0,10)
>>> a
[1, 2, 3, 4, 5, 6, 7, 8, 9]

>>> map(square, a)
[1, 4, 9, 16, 25, 36, 49, 64, 81]

>>> def add(x,y): return x+y
...
>>> reduce(add, a)
45
```

Google's map & reduce operate on (name, value) pairs.

https://docs.python.org/2/tutorial/datastructures.html

Map/Reduce

The programmer writes a **map()** function:

```
map(input) -> key:value
```

The framework calls map() for every piece of input

```
map(input1) -> key1:v1
map(input2) -> key2:v2
map(input3) -> key3:v3
```

Map/Reduce

The programmer writes a map() function:

```
map(input) -> key:value
```

The framework sends the map() function to every computer with data:

```
map(input1) -> key1:v1
map(input2) -> key2:v2
map(input3) -> key3:v3
```

```
map(input11) -> key1:v11
map(input12) -> key2:v12
map(input13) -> key3:v13
```

```
map(input21) -> key1:v21
map(input22) -> key1:v22
map(input23) -> key1:v23
```

“embarrassingly parallel”

Map/Reduce

The programmer writes a **map()** function:

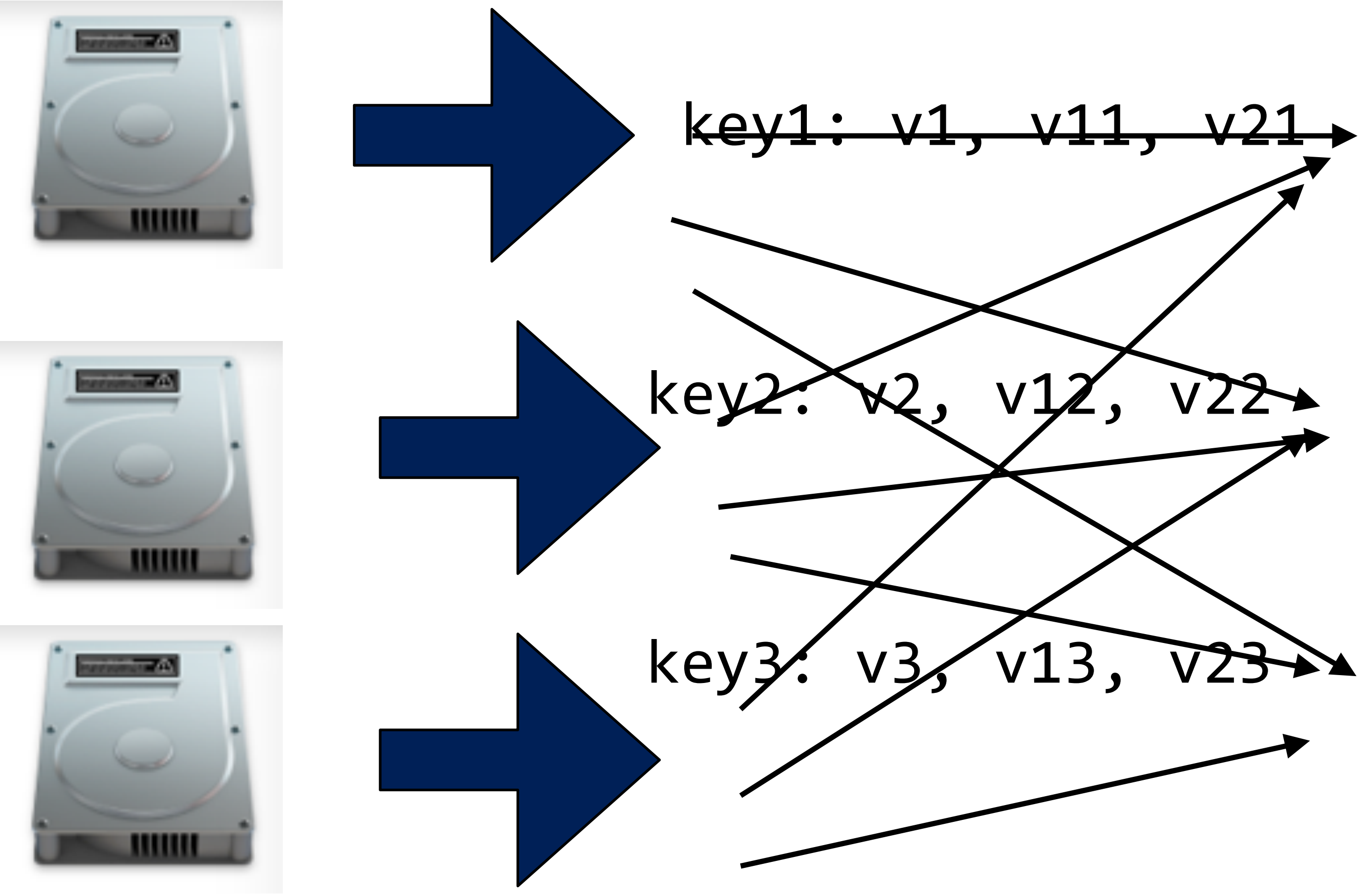
```
map(input) -> key:value
```

The framework sorts the output by key:

```
map(input1) -> key1:v1
map(input2) -> key2:v2
map(input3) -> key3:v3

map(input11) -> key1:v11
map(input12) -> key2:v12
map(input13) -> key3:v13

map(input21) -> key1:v21
map(input22) -> key1:v22
map(input23) -> key1:v23
```



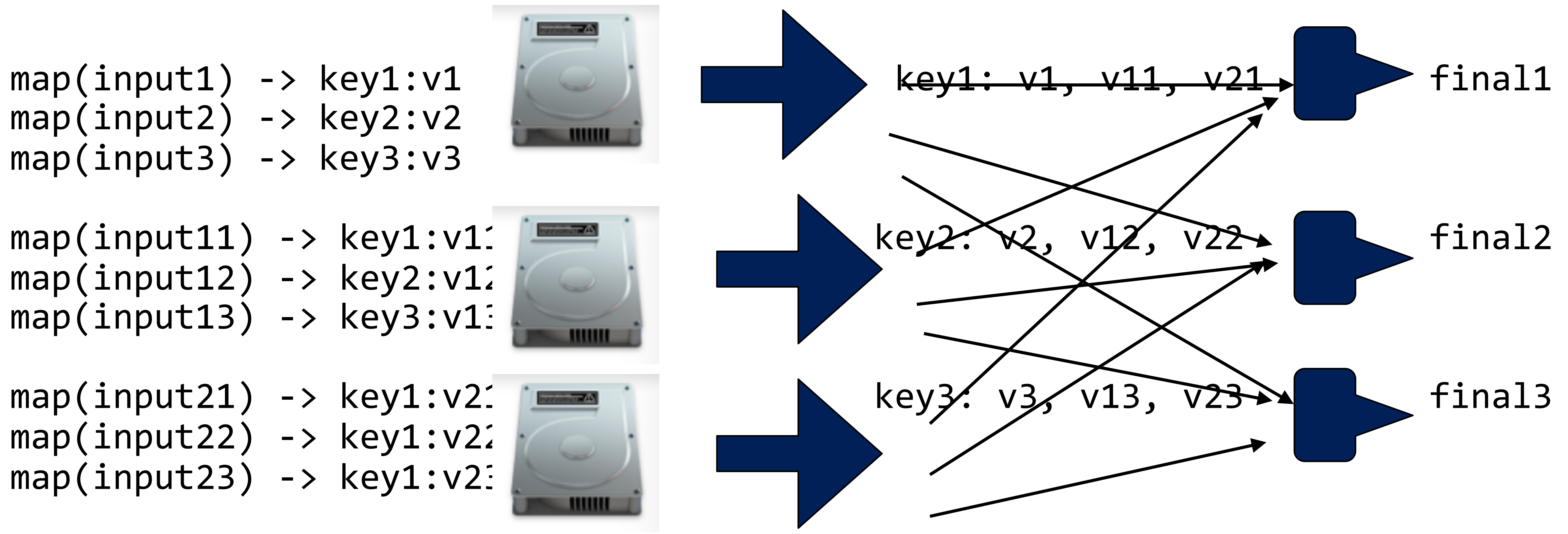
“embarrassingly parallel”

group by key (expensive)

The programmer writes a **reduce()** function:

```
reduce(key,values) -> key:value
```

The framework sorts the output by key:



"embarrassingly parallel"

group by key (expensive)

reduce

“Word Count” is a common MapReduce demonstration program. This Word Count generates a word histogram.

The mapper:

```
map(String input_key, String input_value):
    // input_key: document name
    // input_value: document contents
    for each word w in input_value:
      EmitIntermediate(w, "1");
```

The reducer:

```
reduce(String output_key, Iterator intermediate_values):
  // output_key: a word
  // output_values: a list of counts
  int result = 0;
  for each v in intermediate_values:
    result += ParseInt(v);
  Emit(AsString(result));
```

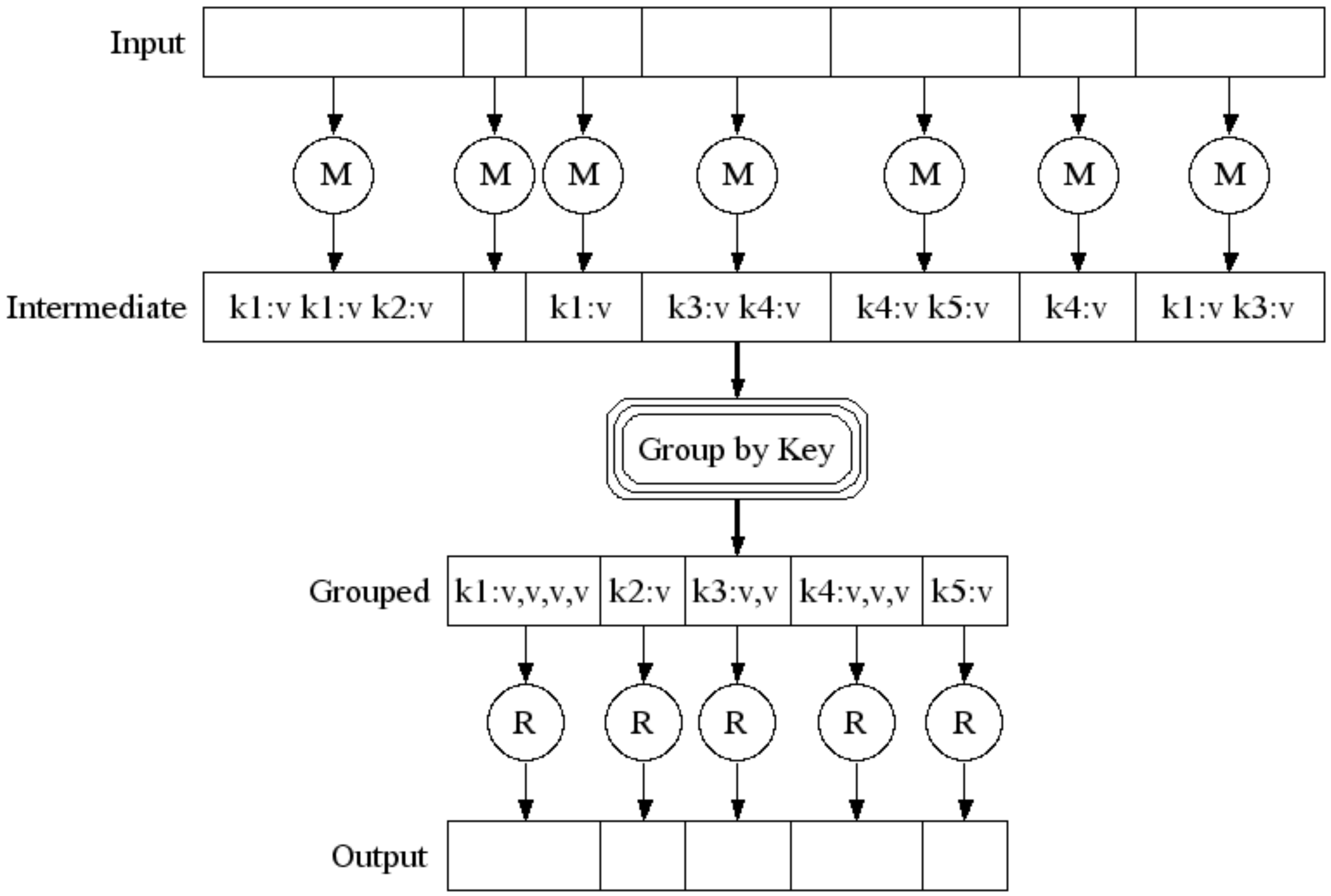
The output:

“to be or not to be”
becomes:
to:1
be:1
or:1
not:1
to:1
be:1

The framework guarantees that “reduce” is called with all pairs of the same key.

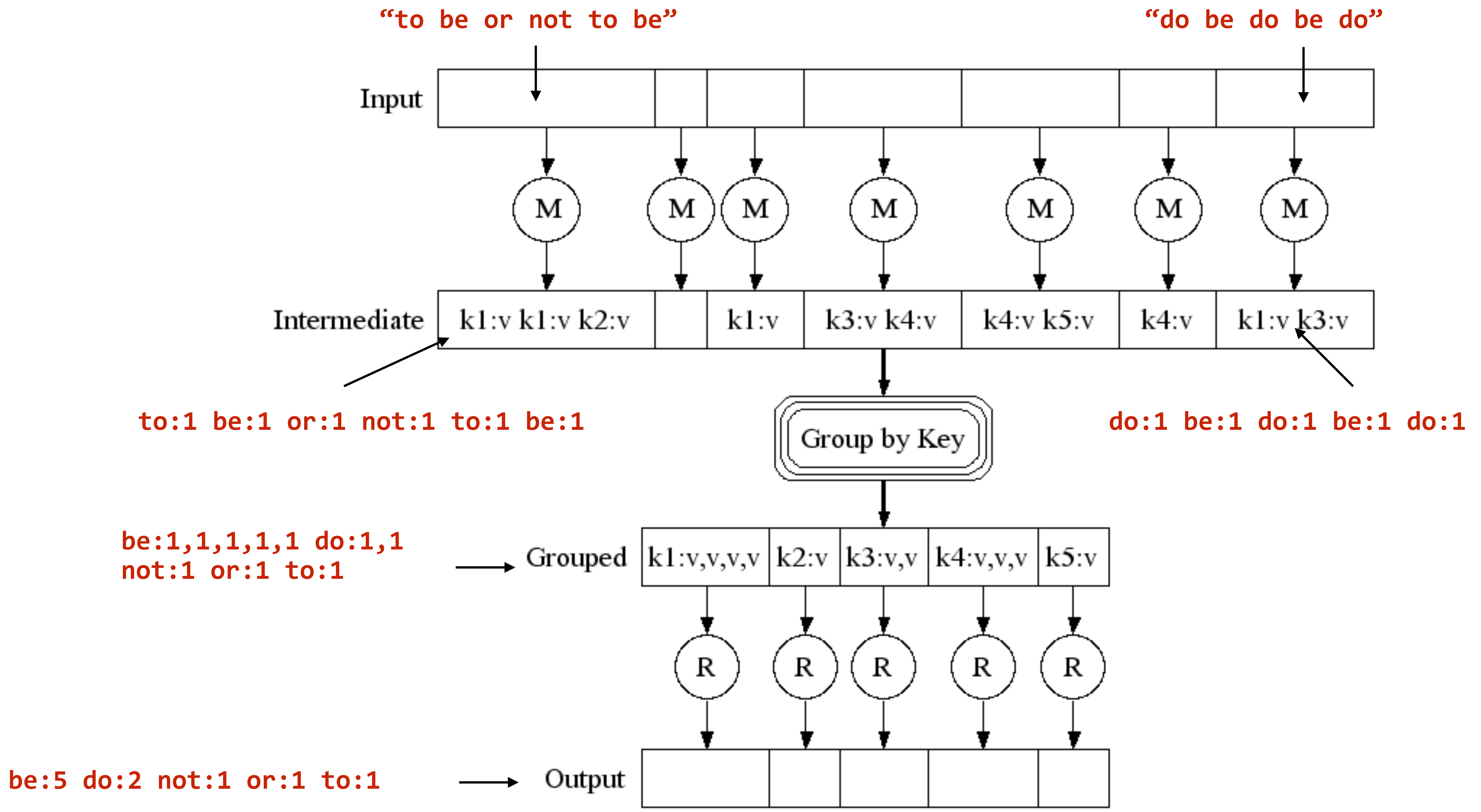
to:2
be:2
or:1
not:1

How this gets put together:



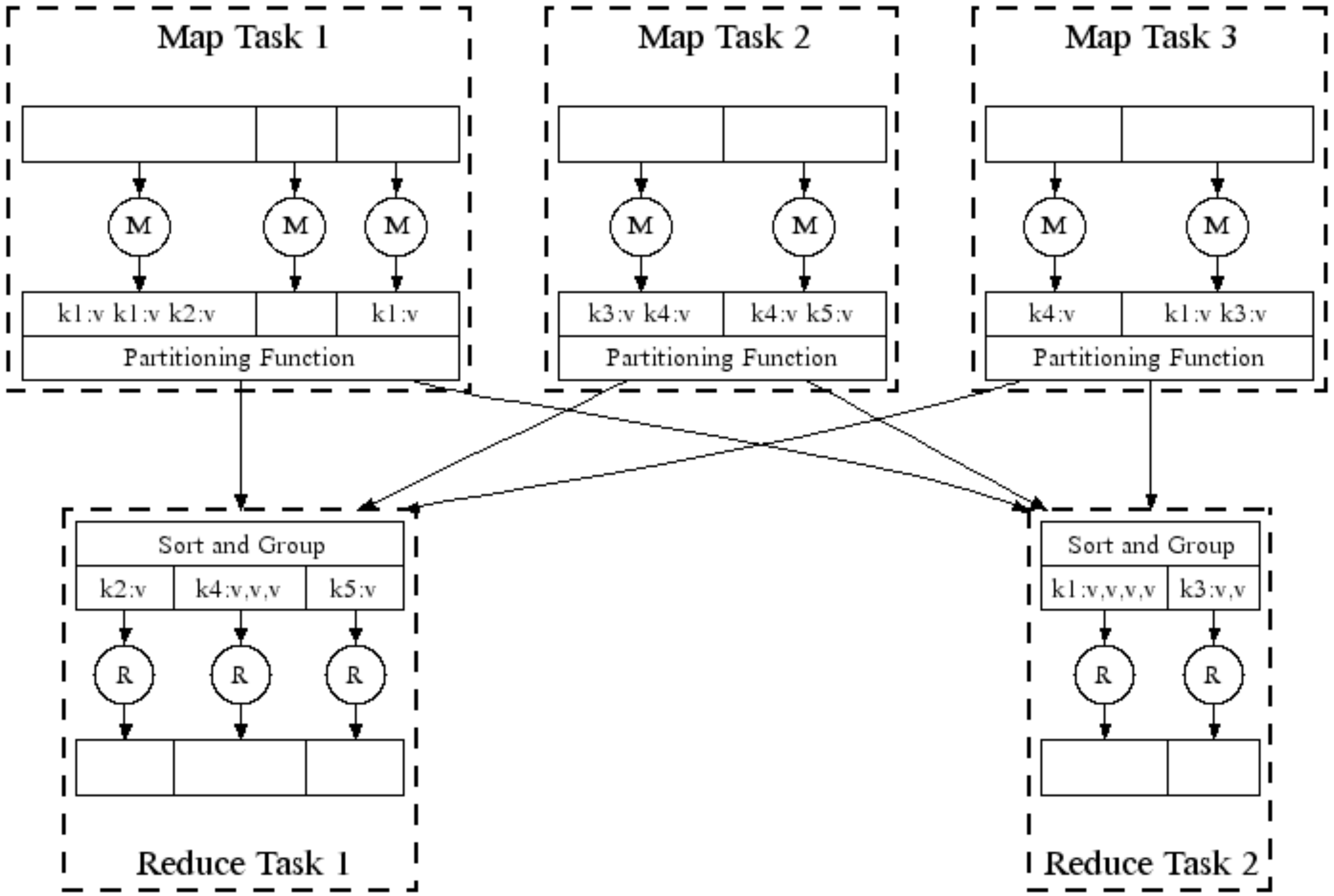
http://research.google.com/archive/mapreduce-osdi04-slides/index-auto-0007.html

How this gets put together.
This time we'll use multiple inputs.



http://research.google.com/archive/mapreduce-osdi04-slides/index-auto-0007.html

Behind the scenes, MapReduce sorts and combines the data.



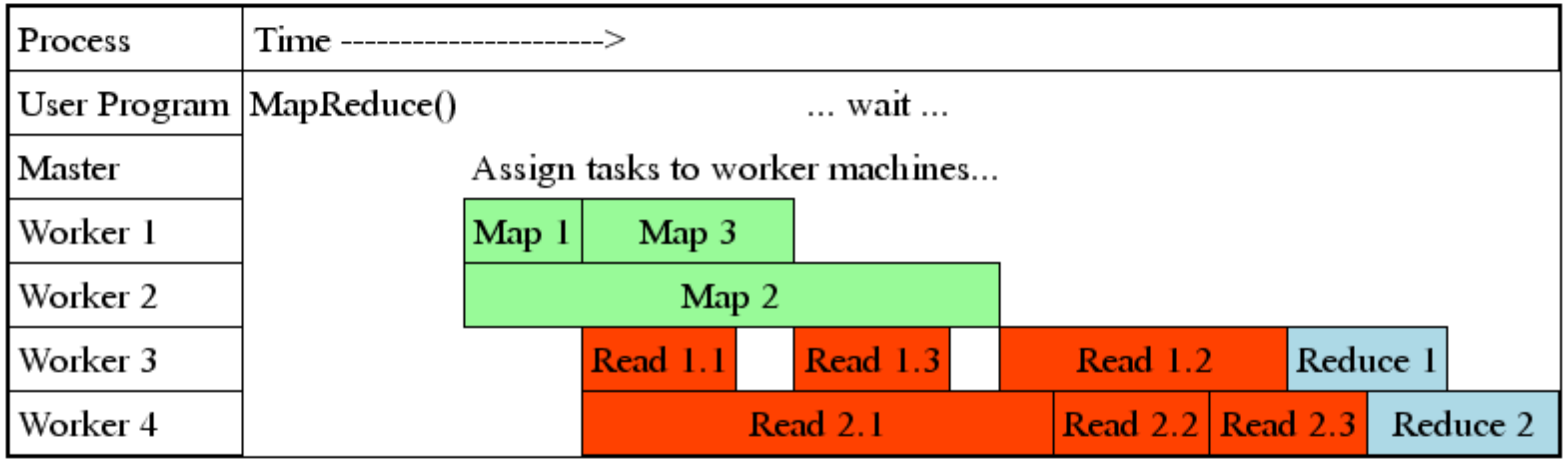
http://research.google.com/archive/mapreduce-osdi04-slides/index-auto-0008.html

MapReduce pipelines execution and provides fault recovery

Workers run both map & reduce tasks.

- Each task is scheduled when data are available.
- Failed tasks (or slow machines) are automatically rescheduled.
- If the same data causes two mappers to fail, the data is ignored.

"Often use 200,000 map/5000 reduce tasks with 2000 machines"



http://research.google.com/archive/mapreduce-osdi04-slides/index-auto-0009.html

MapReduce is a powerful programming paradigm.

The previous example used strings, but map & reduce can apply to any kind of data value.

Examples (from paper)

- Parallelized String search
  - *map emits a pair of string is present (line #, string)*
  - *Reduce is the "identity" function — copies input to output. (line #, string)*
- Count URL Access Frequency:
  - *map reads logfiles and outputs (URL, 1)*
  - *reduce adds up all of the URLs   (URL, total count)*
- Reverse Web-Link Graph (what points to page P?)
  - *map outputs (target, source) for each link found on each web page.*
  - *reduce concatenates the sources: (target, list(source))*
    `e.g. (target, (source1, source2, source3))`

MapReduce benefits

Fault tolerant: all of the inputs are pre-determined from the data.

- If a worker fails, that job can be run on another machine.
- The master writes periodic checkpoints. If it dies, it is restarted.
- “However, given that there is only a single master, it’s failure is unlikely; therefore our current implementation about the MapReduce computation if the master fails.”

Minimizes network bandwidth:

- Attempts schedule workers on the same network node as the data resides.
- Failing that, it tries to schedule the worker on the same network switch

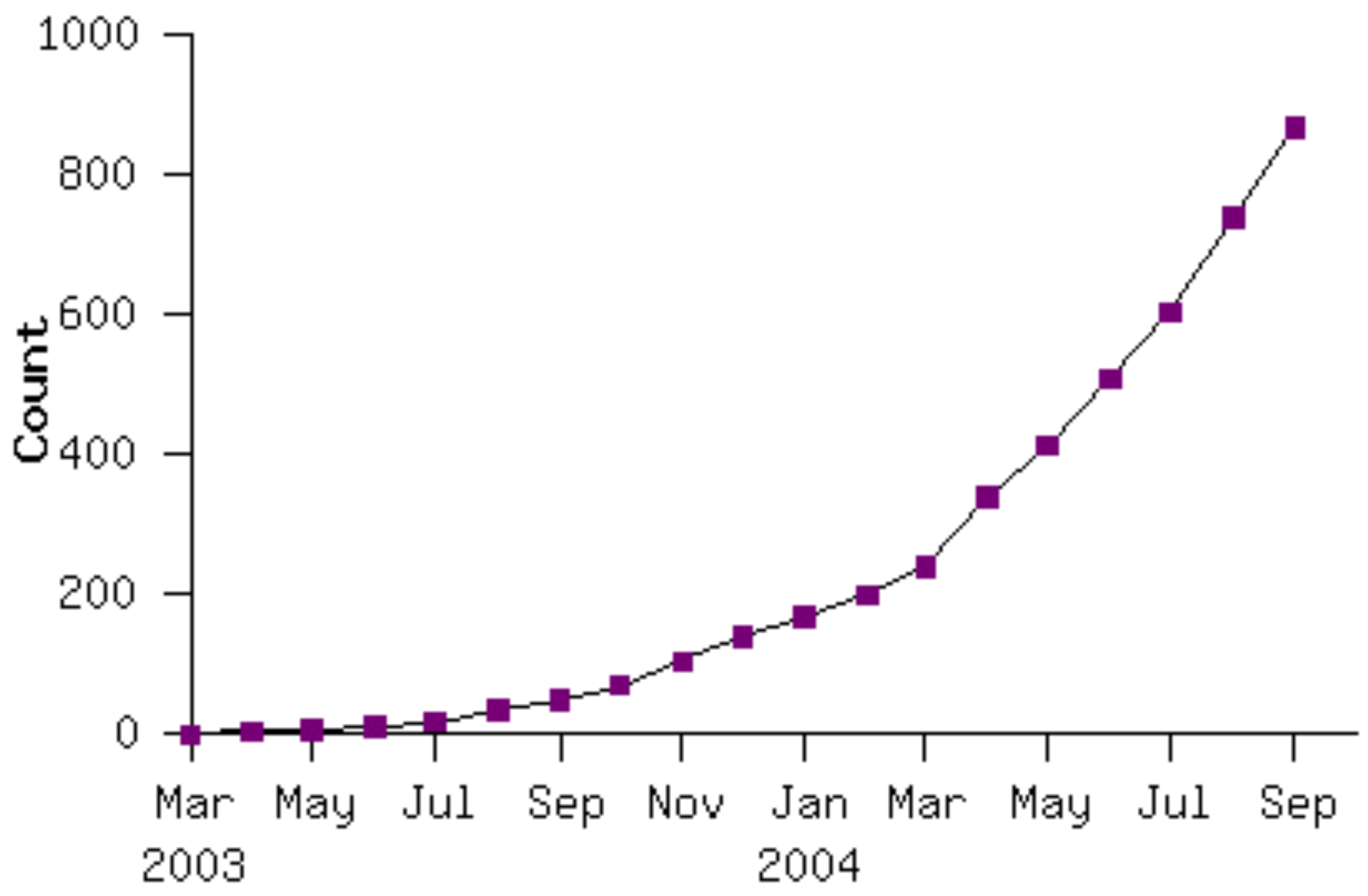
Easier to program!

- Map & Reduce functions are simple and easy to understand.
- Complexity is taken care of by infrastructure.
- Most tasks go faster when you add more machines.

MapReduce was hugely successful at Google.

Easy to modify existing tasks to run on MapReduce famework.

- MapReduce Programs in Google Source Tree:



Hadoop Origins — An open source version of Google's stack

Doug Cutting had been trying to build a search engine at the Internet archive.

- It could only run on certain kinds of machines.
- It required reliable computers.
- When it crashed, it needed to be manually restarted.

Cutting & Cafarella decided to build an open source version of the Google stack to handle the Internet Archive's search.

- In Java, so it would be portable.
  - —and because it's what they knew

In 2006, Cutting moved to Yahoo.

- It was difficult scaling to larger # of nodes.
- Hadoop wasn't good enough to replace Yahoo's search, but it could be used for data analytics.

In 2011, Yahoo had 42,000 nodes and 100s of PBs of storage.

Yahoo spun out Hartonworks as a Hadoop-focused software company.

- https://gigaom.com/2013/03/04/the-history-of-hadoop-from-4-nodes-to-the-future-of-data/

Doug Cutting

Mike Cafarella

Doug Cutting named Hadoop after his son's toy elephant

The New York Times

BUSINESS COMPUTING

Hadoop, a Free Software Program, Finds Uses Beyond Search

By ASHLEE VANCE   MARCH 16, 2009

- http://www.nytimes.com/2009/03/17/technology/business-computing/17cloud.html

Hadoop runs on individual computers in a data center. These computers are called "nodes."

A typical small Hadoop system might have:

- 1 master node
- 1-10 Data Nodes
- 0-10 Compute Nodes

Worker Nodes:

Compute Node.
- Runs Map/Reduce jobs

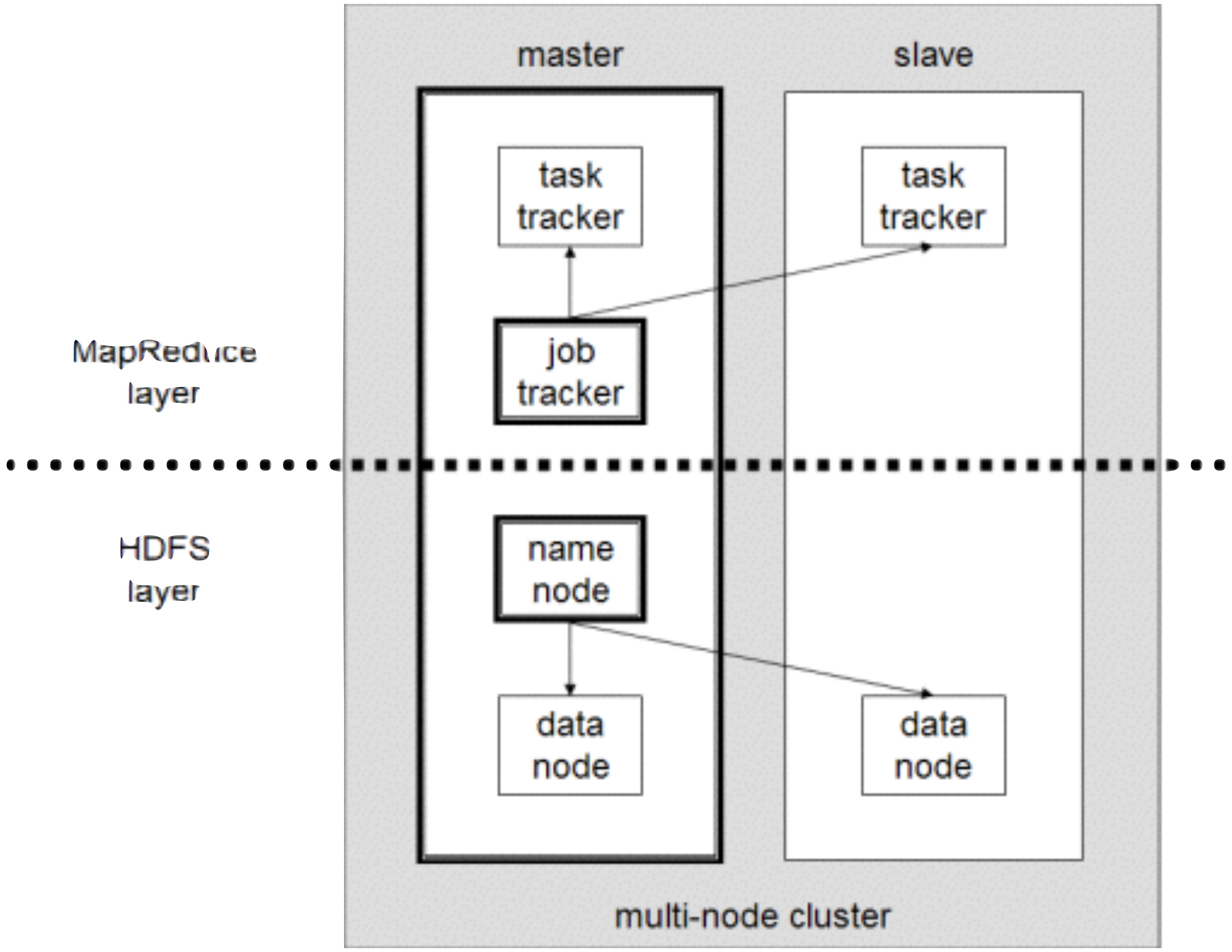
Master Node.
- **Batch jobs submitted.**
- **Tracks progress of jobs.**

Data Node.
- **Holds data**
- **(Can also run jobs)**

Hadoop has two main systems: MapReduce and HDFS

MapReduce — Performs computation

- Job Tracker — Master planner
- Task Tracker — Runs each task



HDFS — Stores the data

- Data Node — Stores the blocks for each file.
- Name Node — Keeps track of every file and where it is stored.
  - Controls the Data Nodes.
- (We will discuss HDFS more in L03)

Real-world Map Reduce.

MapReduce is run as a “batch” operation with a *job configuration*:

- Map function
- Reduce function
- Job parameters

The Hadoop *job client* submits the job (e.g. jar file) to the ResourceManager.

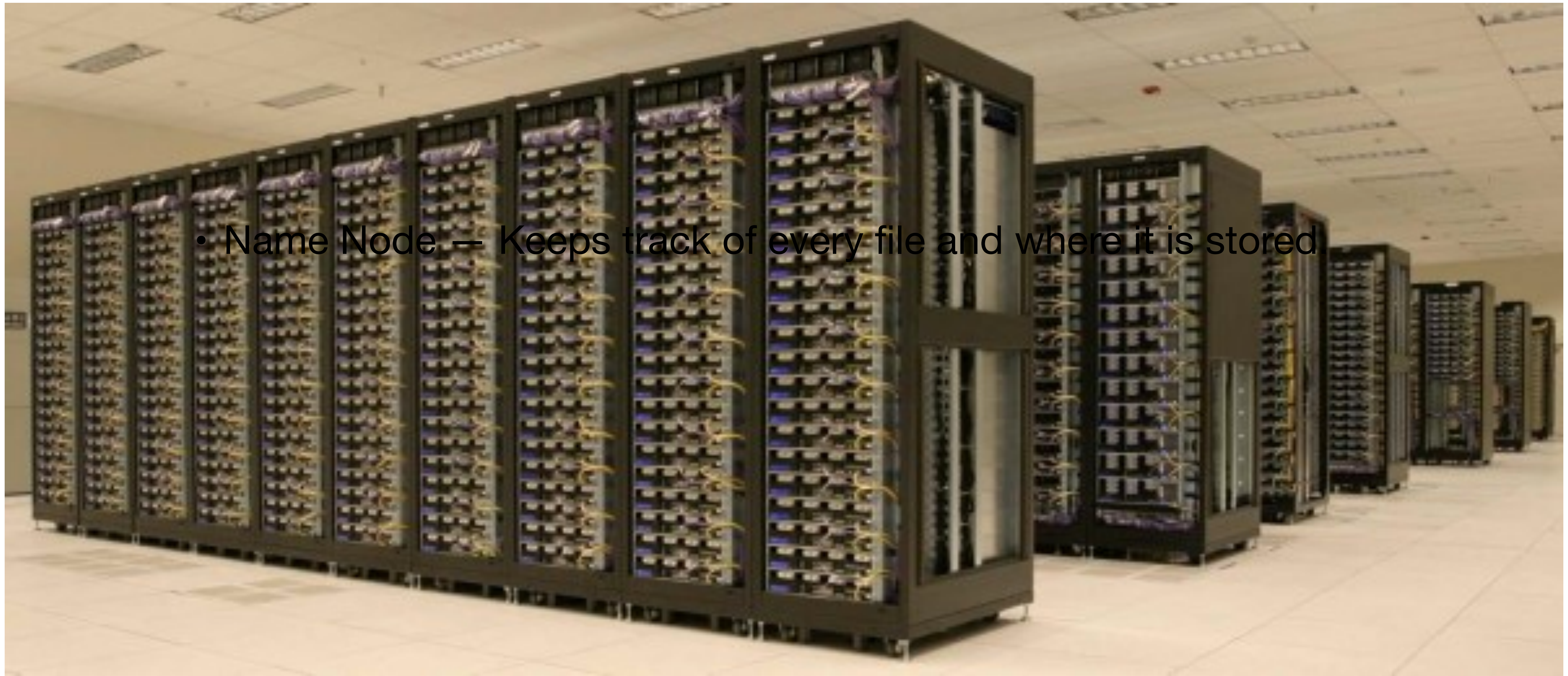
Hadoop Streaming lets jobs be run with any executable.

Hadoop Pipes is a SWIG C++ API for running from C++, python, etc.

https://hadoop.apache.org/docs/stable/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html

https://github.com/chenmiao/Big_Data_Analytics_Web_Text/wiki/Hadoop-with-Cloudera-VM-(the-Word-Count-Example)

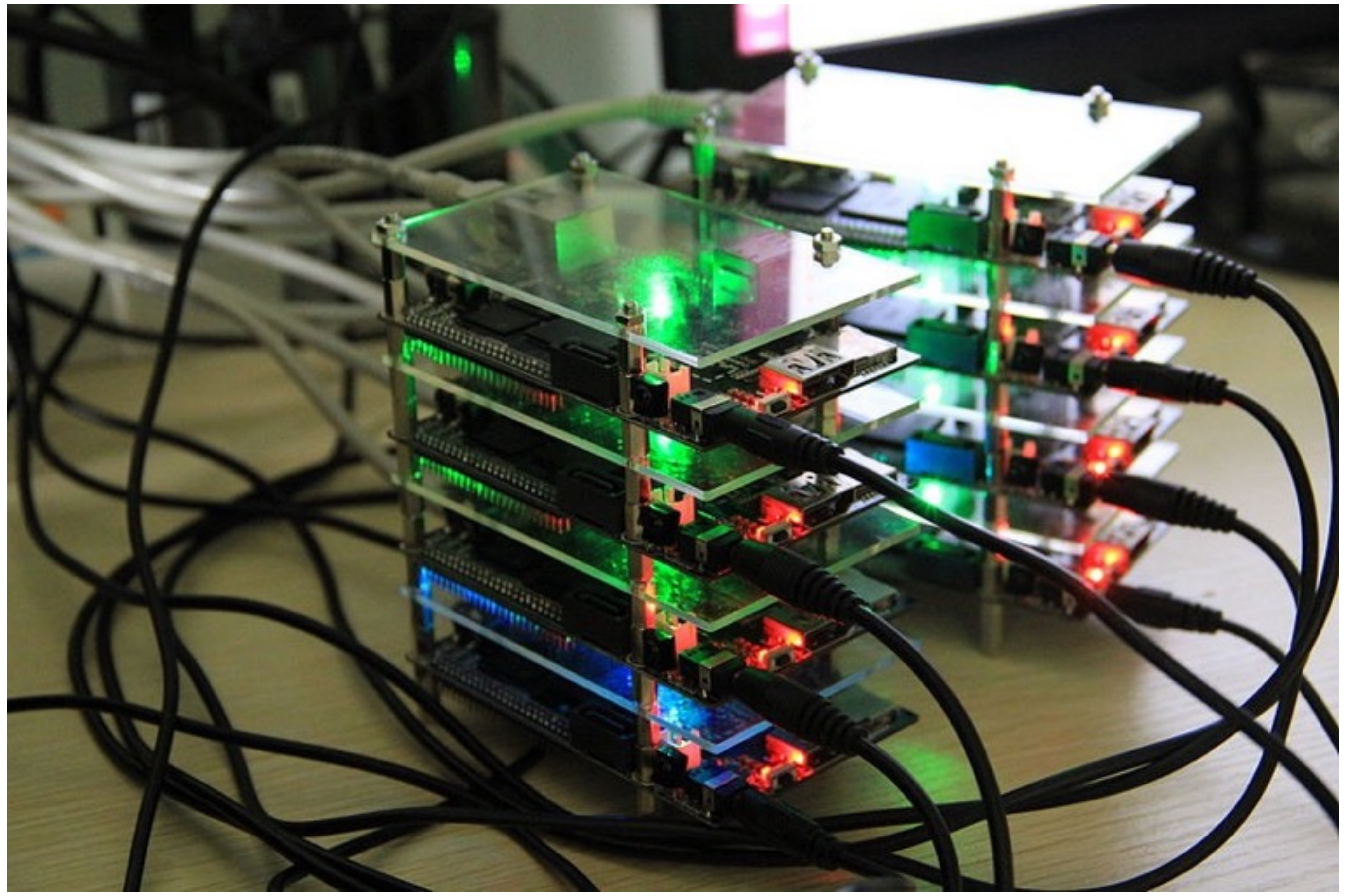
Real Hadoop clusters can be huge.



Hadoop cluster at Yahoo!

Hadoop doesn't *need* huge clusters

Hadoop running on 8 cubieboards:



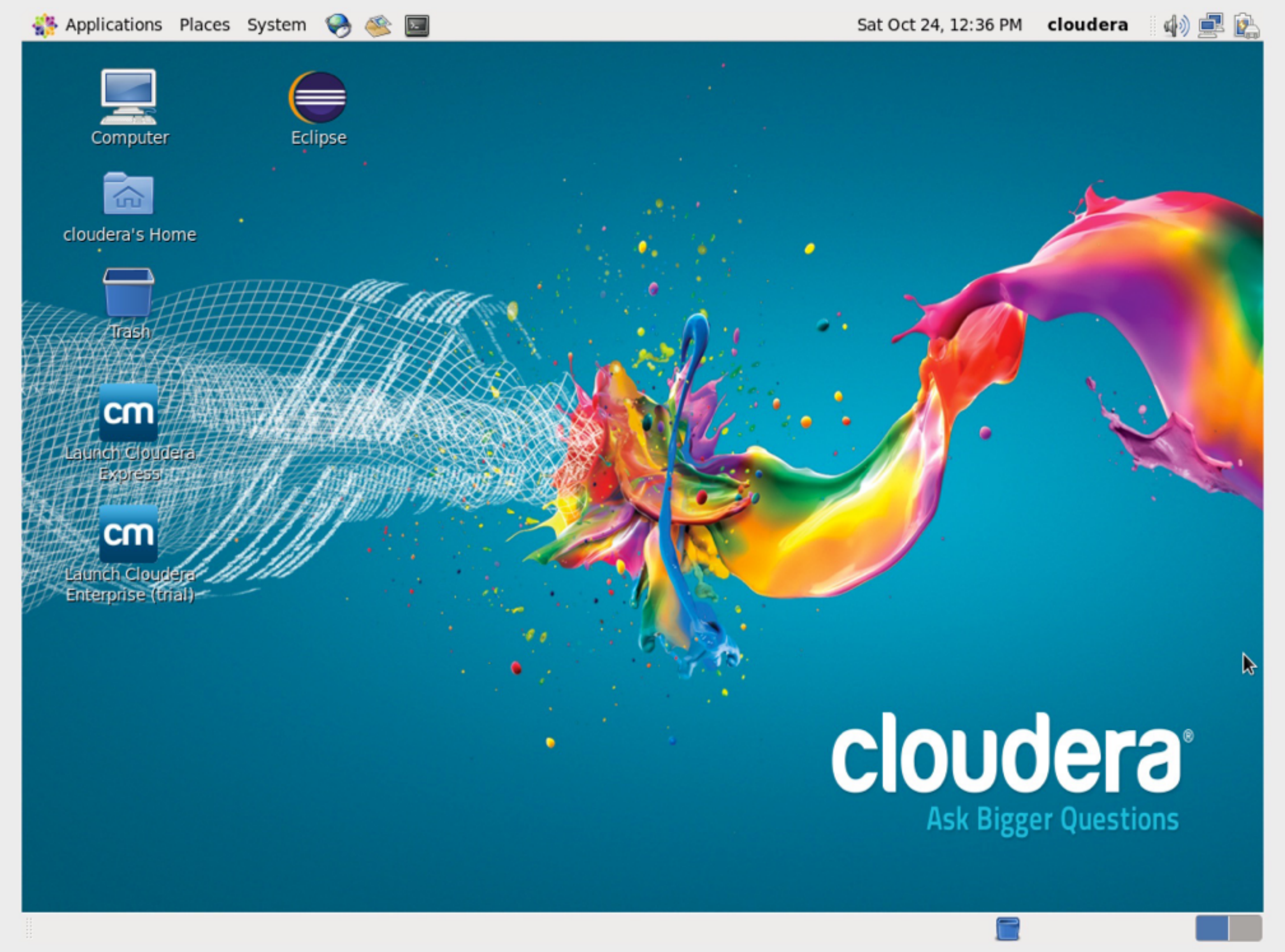
But you would never do this in practice.

Why not?

http://cubieboard.org/2013/08/01/hadoophigh-availability-distributed-object-oriented-platform-on-cubieboard/

The power of Hadoop (and MapReduce) is that it:

- Provides a framework for having a distributing a workflow to multiple physical computers.
- Integrates management of computation and storage.



Getting to know the Cloudera QuickStart VM

Virtualization: Running an OS inside an OS

(Who here has used virtualization before?)

Parts of a Virtual Machine:

- Virtual disk
- RAM
- Machine configuration

Some advantage of Virtualization:

- Checkpointing
- Templates / Copying
- Resource management
- Security

Cloudera makes a "QuickStart VM" that has many "big data" programs pre-installed.

There are many ways to run Word Count on Cloudera

Traditional Hadoop:

- MapReduce job in Java
- MapReduce job with a shell command and Hadoop Streaming
- MapReduce job in Python with mrjob.

Spark:

- Spark with Python
- Spark with Scala

The VM is a Virtual Machine Image File

The VM includes:

- CentOS 6.4 (similar to RedHat Enterprise & Fedora) (uses “yum” not “apt-get”)
- 64-bit OS (requires 64-bit host OS)
- Available for VMWare (Player & Fusion), KVM and *VirtualBox*.
- Requires: 4GiB of RAM on your computer to run.

username/password info:

- Main account: cloudera/cloudera
- Root: root/cloudera
- MySQL root: cloudera
- Hue and Cloudera Manager: cloudera/cloudera

More data:

- http://www.cloudera.com/content/www/en-us/documentation/enterprise/latest/topics/cloudera_quickstart_vm.html
- http://www.cloudera.com/content/support/en/downloads/quickstart_vms.html

Creating the VM with VirtualBox and the Cloudera Distribution

OVF — Open Virtualization Format

VirtualBox Changes to Cloudera Quickstart VM:

Setting	Distribution	Change to
Video RAM	1MB	>64MB
RAM	4GB	≥8
CPUs	1	# in host
Paravirtualization	Legacy	Default
Shared Folders	n/a	Homedir
Motherboard	PIIX3	ICH9
I/O APC	?	enable

(make similar changes for VMWare.)

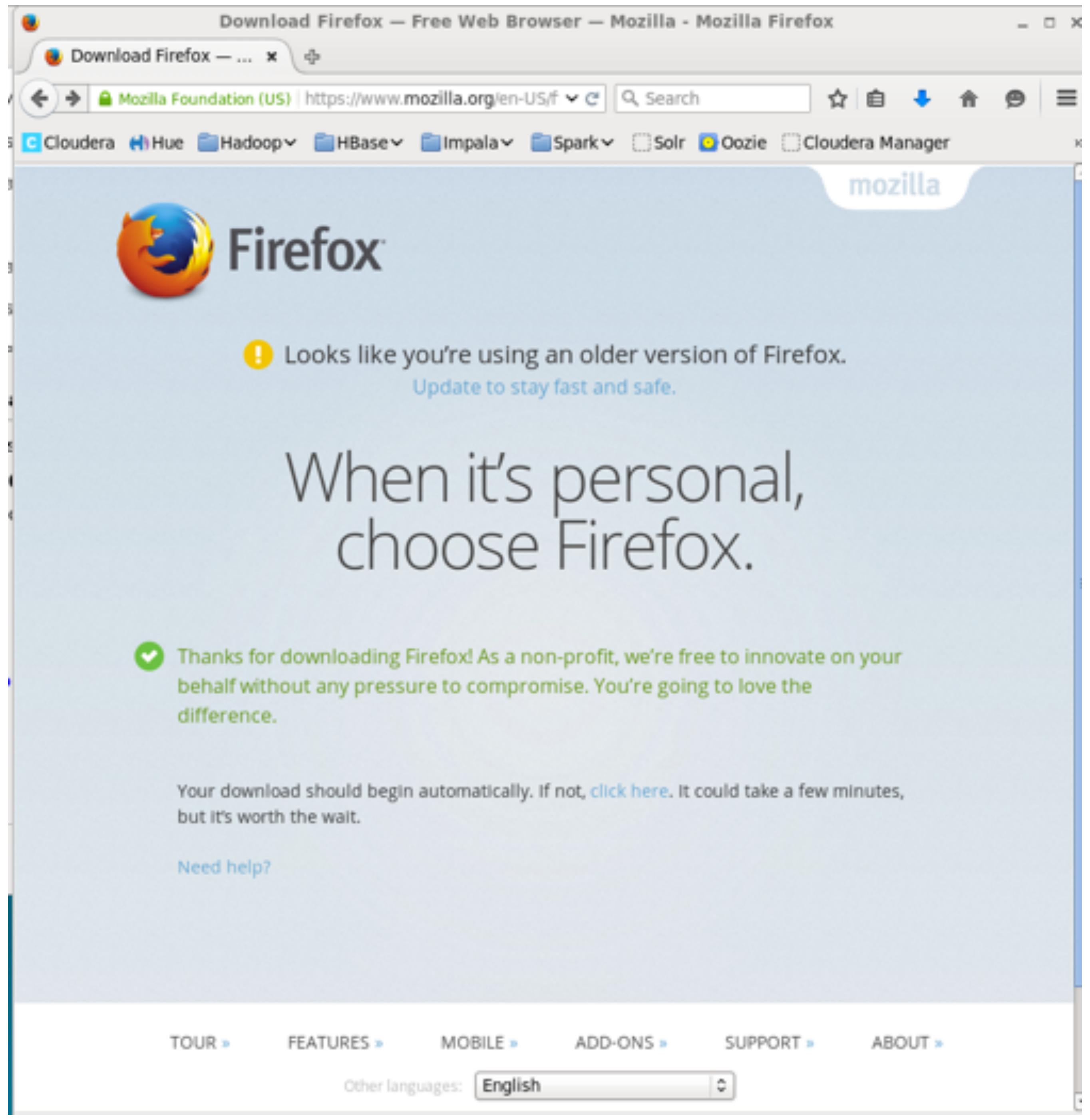
VirtualBox Console for the VM

Keeping your VM up to date: Firefox

Manually download the new Firefox for Linux

— *For some reason, the automatic upgrade doesn't work.*

```
$ cd Downloads/
$ tar xfv firefox-f1.0.2.tar.gz2
$ sudo mv /usr/local/firefox /usr/local/firefox.old.$$
$ sudo mv firefox /usr/local/firefox
```



It's traditional to do "word count" as the map reduce equivalent of "Hello World."

Mapper:

```
public class WordCount {

  public static class TokenizerMapper
       extends Mapper<Object, Text, Text, IntWritable>{

    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(Object key, Text value, Context context
                    ) throws IOException, InterruptedException {
      StringTokenizer itr = new StringTokenizer(value.toString());
      while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());
        context.write(word, one);
      }
    }
  }
```

It's traditional to do "word count" as the map reduce equivalent of "Hello World."

Reducer:

```
public static class IntSumReducer
      extends Reducer<Text,IntWritable,Text,IntWritable> {
   private IntWritable result = new IntWritable();

   public void reduce(Text key, Iterable<IntWritable> values,
                      Context context
                      ) throws IOException, InterruptedException {
     int sum = 0;
     for (IntWritable val : values) {
       sum += val.get();
     }
     result.set(sum);
     context.write(key, result);
   }
 }
```

It’s traditional to do “word count” as the map reduce equivalent of “Hello World.”

Header:

```
import java.io.IOException;
import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
```

Main:

```
  public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "word count");
    job.setJarByClass(WordCount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

The whole program

```java
import java.io.IOException;
import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordCount {

    public static class TokenizerMapper
        extends Mapper<Object, Text, Text, IntWritable>{

        private final static IntWritable one = new IntWritable(1);
        private Text word = new Text();

        public void map(Object key, Text value, Context context
                        ) throws IOException, InterruptedException {
            StringTokenizer itr = new StringTokenizer(value.toString());
            while (itr.hasMoreTokens()) {
                word.set(itr.nextToken());
                context.write(word, one);
            }
        }
    }

    public static class IntSumReducer
        extends Reducer<Text,IntWritable,Text,IntWritable> {
        private IntWritable result = new IntWritable();

        public void reduce(Text key, Iterable<IntWritable> values,
                           Context context
                           ) throws IOException, InterruptedException {
            int sum = 0;
            for (IntWritable val : values) {
                sum += val.get();
            }
            result.set(sum);
            context.write(key, result);
        }
    }

   public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "word count");
        job.setJarByClass(WordCount.class);
        job.setMapperClass(TokenizerMapper.class);
        job.setCombinerClass(IntSumReducer.class);
        job.setReducerClass(IntSumReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

To run this program…

Set up environment (not done for you by CVM):

```
$ export JAVA_CLASSPATH='/usr/lib/hadoop/client-0.20/*:/usr/lib/hadoop/*'
```

Compile WordCount.java and create a jar file:

```
$ javac -d wordcount_classes/ WordCount.java
$ jar -cvf wordcount.jar -C wordcount_classes
```

Put some data in HDFS:

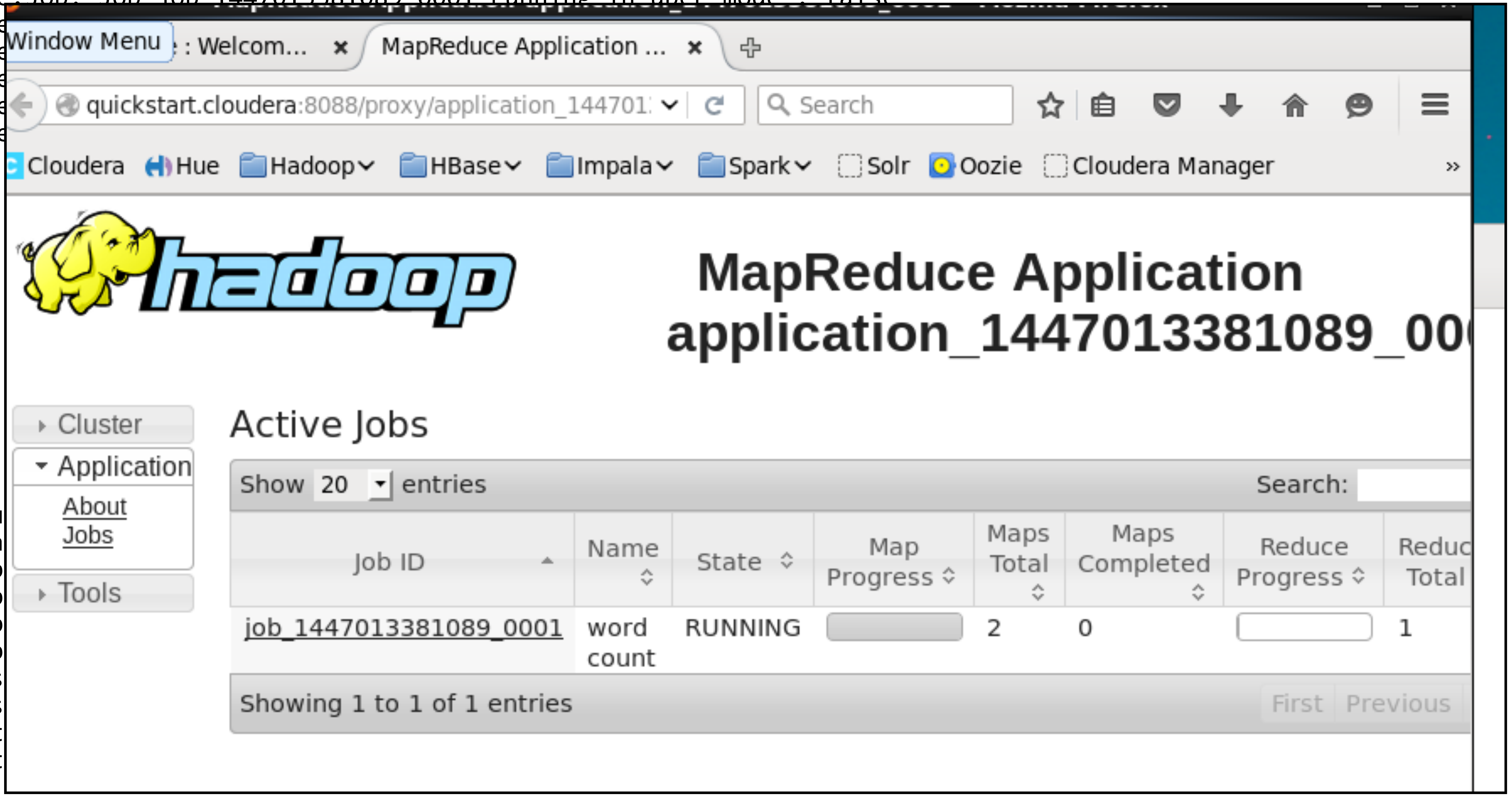
```
$ echo "to be or not to be" > file0
$ echo "do be do be do" > file1
$ hdfs fs -mkdir /user/cloudera/wordcount
$ hdfs fs -mkdir /user/cloudera/wordcount/input
$ hdfs fs -put file0 /user/cloudera/wordcount/input/
$ hadoop fs -put file1 /user/cloudera/wordcount/input/
```

Run it!

```
$ hadoop jar wordcount.jar WordCount /user/cloudera/wordcount/input/ \
                                     /user/cloudera/wordcount/output/
```

Runtime output…

```
$ hadoop jar wordcount.jar WordCount /user/cloudera/wordcount/input/ /user/cloudera/wordcount/output/
15/11/08 13:57:01 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
15/11/08 13:57:02 WARN mapreduce.JobSubmitter: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application
with ToolRunner to remedy this.
15/11/08 13:57:02 INFO input.FileInputFormat: Total input paths to process : 2
15/11/08 13:57:03 INFO mapreduce.JobSubmitter: number of splits:2
15/11/08 13:57:03 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1447013381089_0001
15/11/08 13:57:03 INFO impl.YarnClientImpl: Submitted application application_1447013381089_0001
15/11/08 13:57:03 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application_1447013381089_0001/
15/11/08 13:57:03 INFO mapreduce.Job: Running job: job_1447013381089_0001
15/11/08 13:57:13 INFO mapreduce.Job: Job job_1447013381089_0001 running in uber mode : false
15/11/08 13:57:13 INFO mapreduce
15/11/08 13:57:24 INFO mapreduce
15/11/08 13:57:30 INFO mapreduce
15/11/08 13:57:31 INFO mapreduce
15/11/08 13:57:31 INFO mapreduce
        File System Counters
                FILE: Number
                FILE: Number
                FILE: Number
                FILE: Number
                FILE: Number
                HDFS: Number
                HDFS: Number
                HDFS: Number
                HDFS: Number
                HDFS: Number
        Job Counters
                Launched map
                Launched redu
                Data-local ma
                Total time sp
                Total time sp
                Total time sp
                Total time sp
                Total vcore-s
                Total vcore-s
                Total megabyt
                Total megabyt
```



To see the output:

What it looks like:

```
$ hdfs dfs -ls /user/cloudera/wordcount/output/
Found 2 items
-rw-r--r--   1 cloudera cloudera          0 2015-11-08 13:57 /user/cloudera/wordcount/output/_SUCCESS
-rw-r--r--   1 cloudera cloudera         26 2015-11-08 13:57 /user/cloudera/wordcount/output/part-r-00000
```

— *(remove “cloudera cloudera”)*

```
$ hdfs dfs -ls /user/cloudera/wordcount/output/
Found 2 items
-rw-r--r--   1          0 2015-11-08 13:57 /user/cloudera/wordcount/output/_SUCCESS
-rw-r--r--   1         26 2015-11-08 13:57 /user/cloudera/wordcount/output/part-r-00000
```

And the output:

```
$ hdfs dfs -tail /user/cloudera/wordcount/output/part-r-00000
be      4
do      3
not     1
or        1
to        2
```

Possible points of confusion

/user/cloudera — home directory in HDFS

/home/cloudera — home directory in Linux host file system (ext4)

Real-world Hadoop: combiner & partitioner

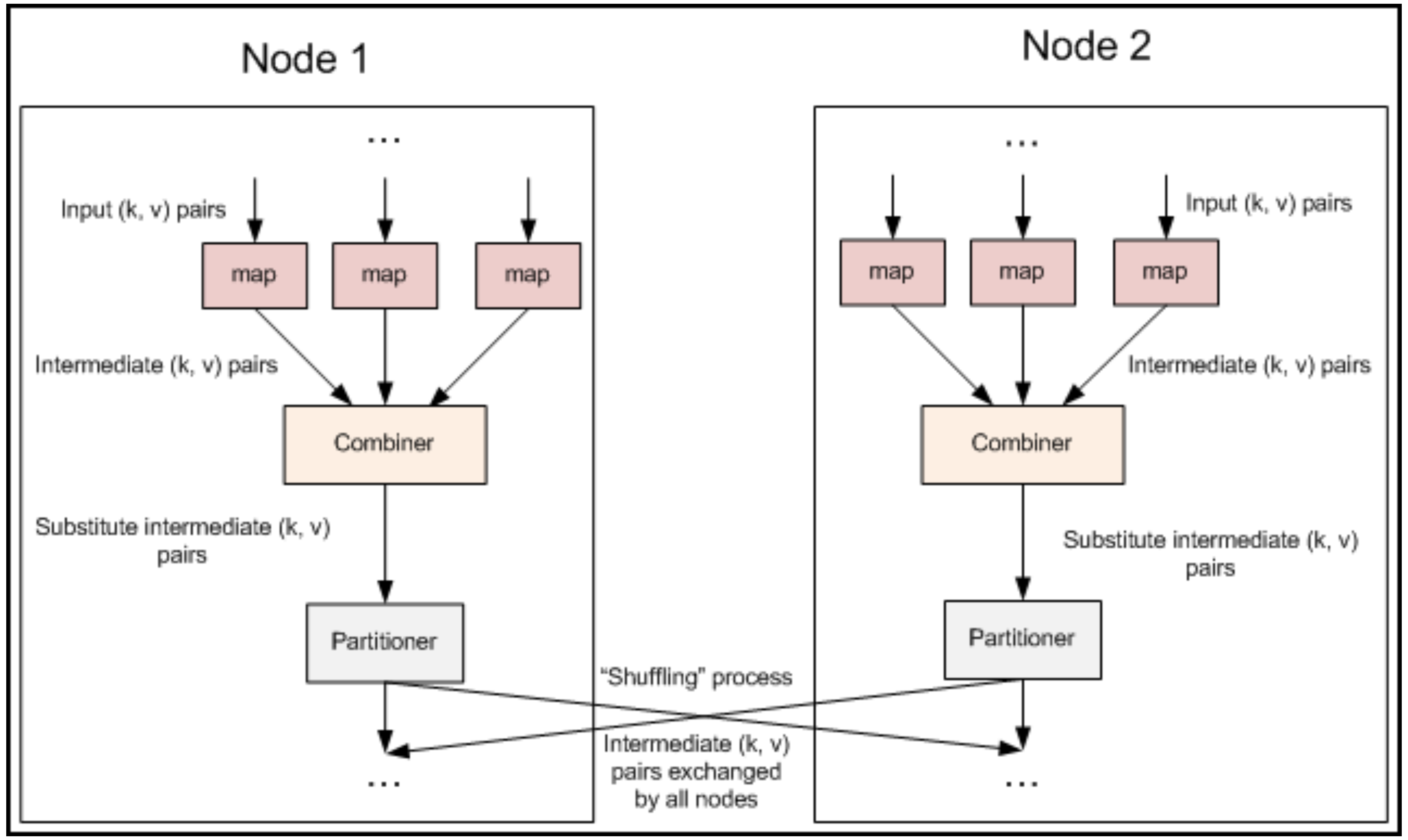
For better performance, you would specify a combiner and a partitioner

Combiner:

- Like a reducer, but just for the node.
- Not necessary — an optimization.

Partitioner:

- When working with more than one reducer.
- Decides which reducer gets which data.



Excellent online tutorials:

- http://www.tutorialspoint.com/map_reduce/map_reduce_partitioner.htm
- http://www.tutorialspoint.com/map_reduce/map_reduce_combiners.htm

Approaches for running Hadoop MapReduce jobs

Java (native) ✔

- Advantages:
  - *Fast — data stays within Java VM*
  - *Few dependencies — Everything in a .jar file*
- Disadvantages:
  - *Not everybody knows Java*
  - *Text processing in Java is hard*

Hadoop “Streaming” API

- Mapper & Reducer read from stdin to stdout. Fields separated by \t
- Advantage — Easy to integrate with existing code.
- Disadvantage — High overhead

mrjob

- Python implementation sits on top of Hadoop Streaming.
- Advantage — Powerful. Local testing.
- Disadvantage — High overhead

Hadoop Streaming

Hadoop streaming — reads from stdin & writes to stdout.

- Allows using Hadoop MapReduce with any language.
- Performance penalty — all I/O has to go over pipes.

MRJOB is based on top of Hadoop Streaming.

Debugging

if you see this:

```
Caused by: java.lang.reflect.InvocationTargetException
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:57)
    at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
    at java.lang.reflect.Method.invoke(Method.java:606)
    at org.apache.hadoop.util.ReflectionUtils.setJobConf(ReflectionUtils.java:106)
    ... 17 more
Caused by: java.lang.RuntimeException: configuration exception
    at org.apache.hadoop.streaming.PipeMapRed.configure(PipeMapRed.java:221)
    at org.apache.hadoop.streaming.PipeMapper.configure(PipeMapper.java:66)
    ... 22 more
Caused by: java.io.IOException: Cannot run program "wordcount_mapper.py": error=2, No such file or directory
    at java.lang.ProcessBuilder.start(ProcessBuilder.java:1047)
    at org.apache.hadoop.streaming.PipeMapRed.configure(PipeMapRed.java:208)
    ... 23 more
Caused by: java.io.IOException: error=2, No such file or directory
    at java.lang.UNIXProcess.forkAndExec(Native Method)
    at java.lang.UNIXProcess.<init>(UNIXProcess.java:186)
    at java.lang.ProcessImpl.start(ProcessImpl.java:130)
    at java.lang.ProcessBuilder.start(ProcessBuilder.java:1028)
    ... 24 more
```

Look for errors that you can understand

Hadoop “mrjob”

With mrjob:

- You write a class that implements mapper, reducer, etc.
- You run the program, which runs the MRJob routines…
  - *Sample program:*

```
from mrjob.job import MRJob

class MRWordFrequencyCount(MRJob):

    def mapper(self, _, line):
        yield "chars", len(line)
        yield "words", len(line.split())
        yield "lines", 1
    def reducer(self, key, values):
        yield key, sum(values)

if __name__ == '__main__':
    MRWordFrequencyCount.run()
```

mrjob help

```
$ python word_count.py --help
Usage: word_count.py [options] [input files]

Options:
  --help-emr            show EMR-related options
  --help-hadoop         show Hadoop-related options
  --help                show this message and exit
  --help-runner         show runner-related options

  Running specific parts of the job:
    --combiner          run a combiner
    --mapper            run a mapper
    --reducer           run a reducer
    --steps             print the mappers, combiners, and reducers that this
                        job defines
    --step-num=STEP_NUM
                        which step to execute (default is 0)

  Protocols:
    --strict-protocols  If something violates an input/output protocol then
                        raise an exception
    --no-strict-protocols
                        If something violates an input/output protocol then
                        increment a counter and continue
$
```

mrjob config file: YAML or JSON
(YAML only if YAML libraries are installed)

"YAML Ain't Markup Language" — http://www.yaml.org/

- A "human friendly data serialization standard for all programming languages."
- Structure conveyed through indentation — whitespace is significant (like python, unlike XML or JSON)

All files begin with "---" and end with "..."

Lists:

```
---
fruits:
    - Apple
    - Orange
    - Strawberry
    - Mango
...
```

Dictionary:

```
---
martin:
   name: Marin D'vloper
   job: Developer
   skin: Elite
...
```

Abbreviations:

```
---
fruits: ['Apple', 'Orange', 'Strawberry', 'Mango']
...
```

Dictionary:

```
---
martin: {name: Marin D'vloper, job: Developer, skin: Elite}
...
```

Boolean Values:

```
true_values: [yes, True, TRUE]
false_values: [no, false, FALSE]
```

Examples of mrjob config files:

https://pythonhosted.org/mrjob/guides/configs-basics.html :

```
---
runners:
  emr:
    cmdenv:
      TZ: America/Los_Angeles
...

---
runners:
  emr:
    aws_access_key_id: HADOOPHADOOPBOBADOOP
    aws_region: us-west-1
    aws_secret_access_key: MEMIMOMADOOPBANANAFANAFOFADOOPHADOOP
...
```

Precedence:

- Command Line
- Config File

Three ways to run mrjob:

Runner:

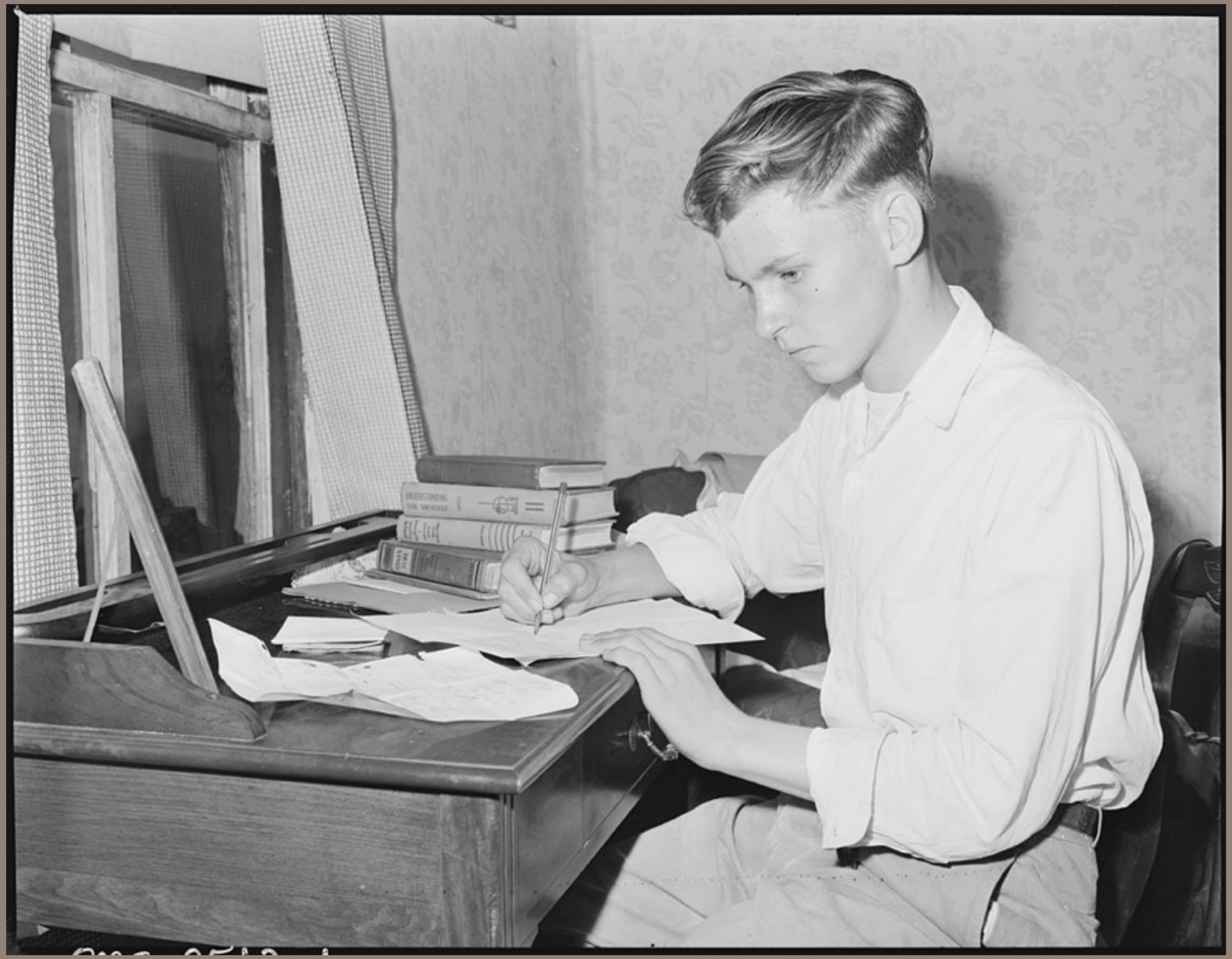
- run jobs locally without Hadoop
  - *Within a single Python Process*
  - *With sub-processes and PIPE I/O*
- run jobs on local Hadoop Cluster — You need to install mrjob first and log into the master node.
- run jobs on ElasticMapReduce — mrjob starts up EMR and runs it.

General approach:

1. Run locally within a single python process and a reduced data set
2. Run locally with PIPE IO
3. Spin up a cluster, install mrjob, and try it out.
4. Have mrjob create and kill clusters for production.

Remember: anything stored in HDFS is lost when an EMR cluster shuts down!

- *But things stored in S3 are preserved*


http://bit.ly/louis_sergent_homework_1946

For ~~next week~~ January 25, 2016

Technologies you should know

git

emacs

eclipse

CentOS vs. Ubuntu

- Centos: yum-cron; yum makecache

VMWare

Things to watch out for when searching the Internet…

When you read articles on the Internet, be sure to check:

- When was the article written?
- What version of the software is being referenced?
- How do you know that it's right?

Things to beware of:

- Hadoop 1 is similar to Hadoop 2, but different.
- Hadoop 1 had a JobTracker and TaskTracker; Hadoop 2 has YARN
- Hadoop 1 required more configuration (e.g. setting # of reducers)
- Hadoop 2 entered beta in 2013.
- Different distributions of Hadoop behave differently.
- Many things that people present as "facts" are actually opinions.

Homework — Reading

Required:

- Cloudera tutorials
  - *WordCount v1.0*
    - http://www.cloudera.com/content/www/en-us/documentation/other/tutorial/CDH5/Hadoop-Tutorial/ht_wordcount1.html
  - *WordCount v2.0*
    - http://www.cloudera.com/content/www/en-us/documentation/other/tutorial/CDH5/Hadoop-Tutorial/ht_wordcount2.html
  - *WordCount v3.0*
    - http://www.cloudera.com/content/www/en-us/documentation/other/tutorial/CDH5/Hadoop-Tutorial/ht_wordcount3.html
- mrjob documentation
- hadoop-stream documentation
  - https://hadoop.apache.org/docs/stable/hadoop-streaming/HadoopStreaming.html

Optional:

- GFS paper
- MapReduce paper
- A Guide to Python Frameworks for Hadoop —
    - http://blog.cloudera.com/blog/2013/01/a-guide-to-python-frameworks-for-hadoop/

Videos

Cloudera: Introducing MapReduce and HDFS

- http://www.cloudera.com/content/www/en-us/resources/training/introduction-to-apache-mapreduce-and-hdfs.html

The Free Lunch Is Over

- http://www.gotw.ca/publications/concurrency-ddj.htm

Homework — Problem Set

Problem #1: Pricing Cloud Computing

- Determine the price of storage and processing a massive data problem on Amazon AWS, Google Compute Engine, Microsoft Azure, and (optionally) a budget provider.
  - *Simplifying assumptions for transfer in, storage, processing.*

Problem #2: Install Cloudera

Problem #3: Cloudera Word Count example in Java

Problem #4: Cloudera Word Count example in Python (Streaming API)

- *Note: with Hadoop streaming, the reducer gets all keys and must determine when the key changes.*

Problem #5: mrjob word count example with local and hadoop

Problem #6: Find 20 most common words in Shakespeare with MRJob

Resources

Yahoo! Hadoop Tutorial

- https://developer.yahoo.com/hadoop/tutorial/

Apache Hadoop FAQ:

- https://wiki.apache.org/hadoop/FAQ

Hadoop-user mailing list archives:

- http://mail-archives.apache.org/mod_mbox/hadoop-user/

Frontiers in Massive Data Analysis (prepublication)

- http://www.nap.edu/catalog/18374/frontiers-in-massive-data-analysis (read online for free)

Revolution R Open

- http://www.revolutionanalytics.com/revolution-r-open

mrjob

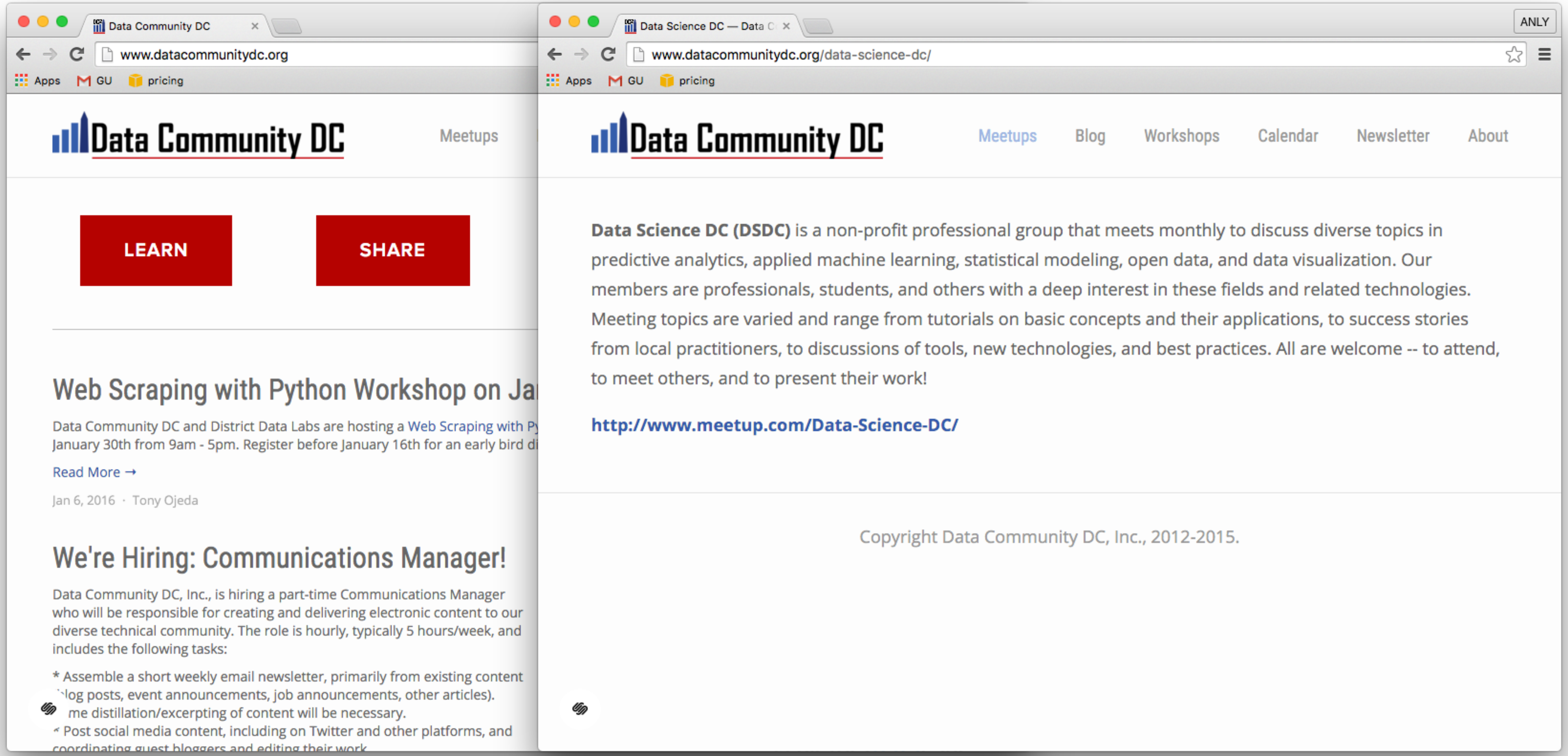
- https://pythonhosted.org/mrjob/index.html
- http://stackoverflow.com/questions/tagged/mrjob
- https://github.com/Yelp/mrjob

More resources

Excellent blog post comparing different python frameworks for MR:

- http://blog.cloudera.com/blog/2013/01/a-guide-to-python-frameworks-for-hadoop/
- Slides: http://www.slideshare.net/slideshow/embed_code/key/9jAfDIRMoJiKPP
- Uses Google Books Ngram data as a demo, not wordcount!
    - See https://books.google.com/ngrams for more

Data Community DC
http://www.datacommunitydc.org/data-science-dc/



Contact Information

Simson L. Garfinkel
sg1224@georgetown.edu
Google Voice: 202-649-0029

Text or Call

Ghaleb Abdulla
abdulla1@llnl.gov
Lab: (925) 423-5947

Voice Only!