http://gnoted.com/wp-content/uploads/2012/02/cloud_43-595x553.jpg

Hi!

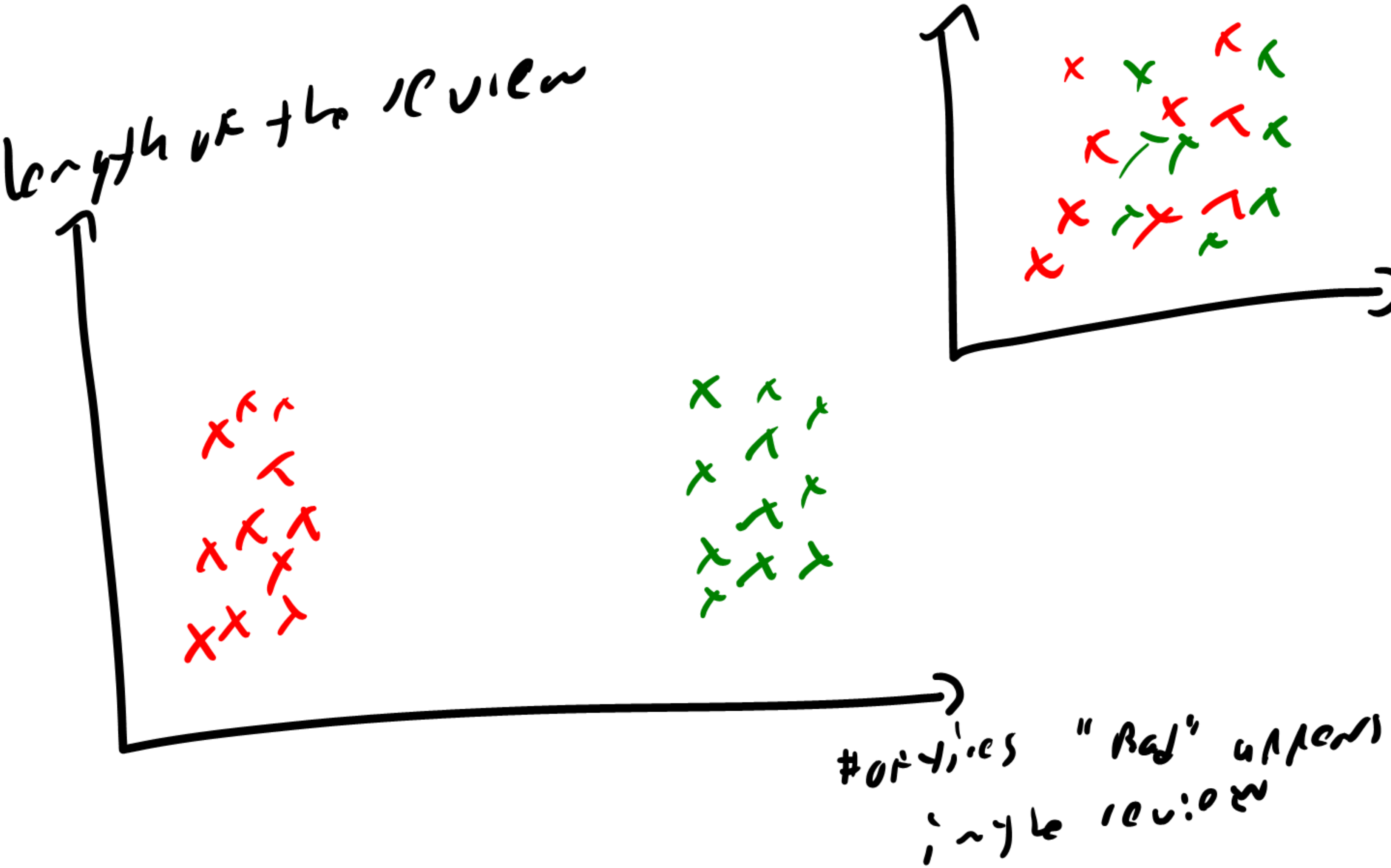
DS501: Large-Scale Data Analysis

Prof. Randy Paffenroth
rcpaffenroth@wpi.edu

Worcester Polytechnic Institute

Announcements

- Case Study 3 due next week, April 6!
  - Can I help with anything?


length of the review
# of times "Bad" appears in the review

Hint on Case Study 3…
and a segue to today's class.

class `sklearn.grid_search.` **`GridSearchCV`** (*estimator*, *param_grid*, *scoring=None*, *fit_params=None*, *n_jobs=1*, *iid=True*, *refit=True*, *cv=None*, *verbose=0*, *pre_dispatch='2*n_jobs'*, *error_score='raise'*) [

n_jobs : int, default=1

Number of jobs to run in parallel.

Changed in version 0.17: Upgraded to joblib 0.9.3.

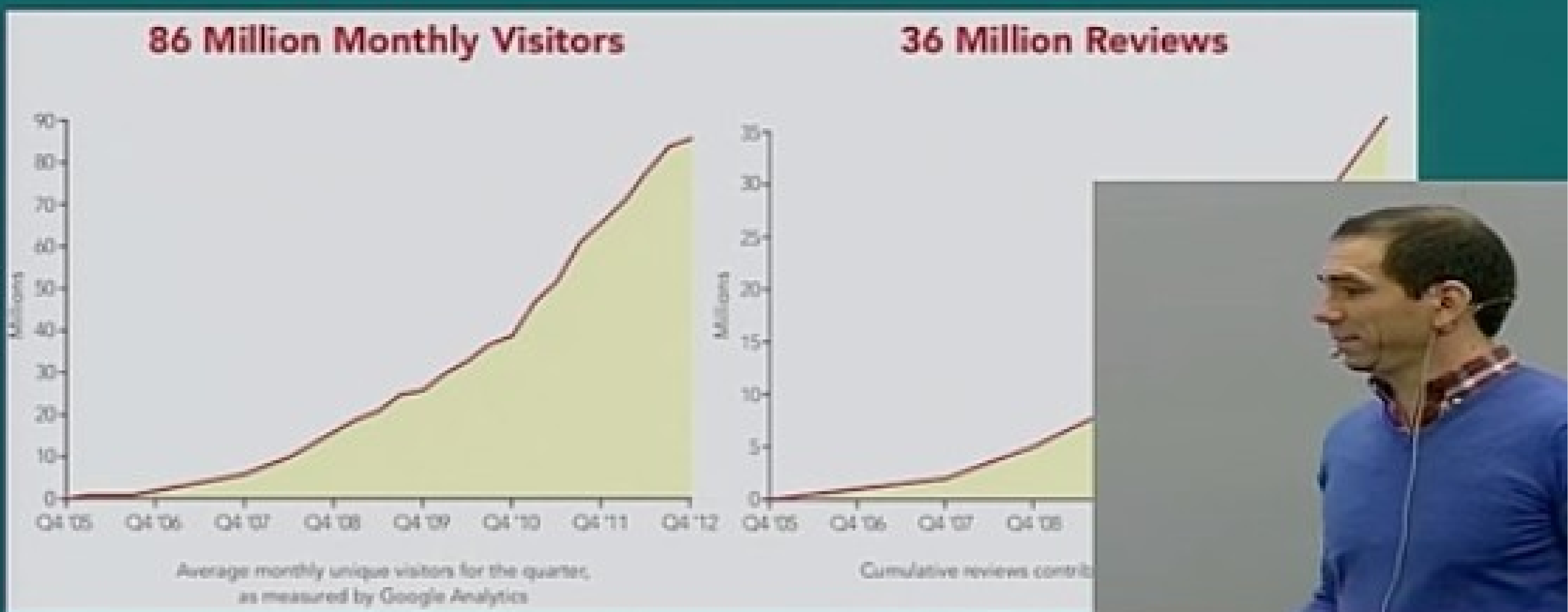
"n_jobs=-1" is a bad idea...

A nice article I saw while preparing the notes for today.

- http://arstechnica.com/information-technology/2016/03/to-sql-or-nosql-thats-the-database-question/

Yelp has a problem

- 250+ GB of logs per day
- Each GB takes 10 minutes to process
- How long to handle a day's logs?



Oops...

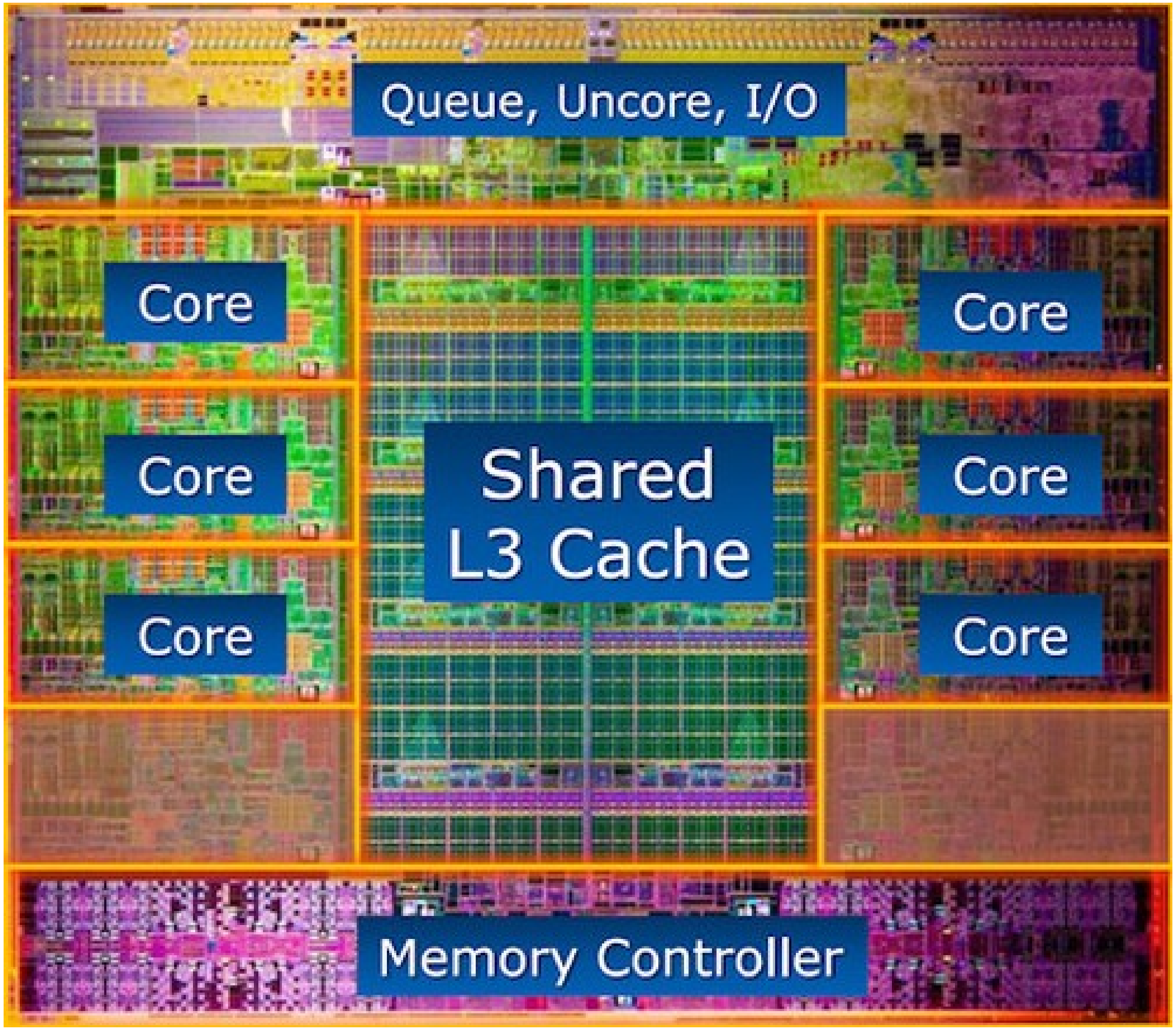
- 250/(6*24) = 1.73 days of work (per day!)
  - You never catch up!
  - What do you do?

What is the answer?

By Wikieditor243 (Own work) [CC BY-SA 3.0 (http://creativecommons.org/licenses/by-sa/3.0)], via Wikimedia Commons

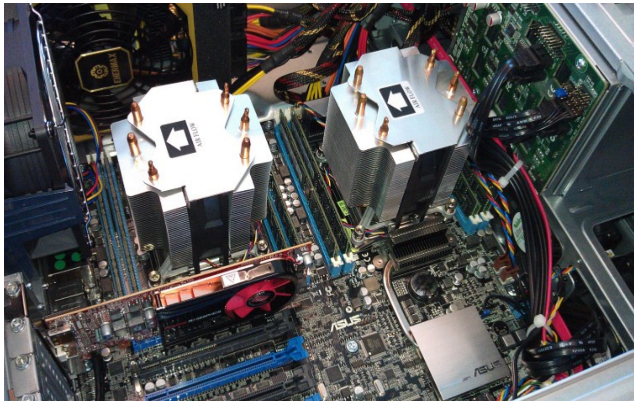
Computer Architecture

CPU architecture



http://cdn.overclock.net/d/df/500x1000px-LL-df146eca_dieshot.jpeg

http://images.anandtech.com/doci/6985/DT_Haswell_i7_FB_678x452.jpg



http://www.2cpu.com/article_images/03062012_romley/romley_system.jpg

Memory hierarchy

- http://computerscience.chemeketa.edu/cs160Reader/_images/Memory-Hierarchy.jpg
- http://en.wikipedia.org/wiki/Memory_hierarchy
- http://web.eecs.utk.edu/~dongarra/WEB-PAGES/SPRING-2005/Lect04.pdf
- **http://web.eecs.utk.edu/~dongarra/WEB-PAGES/SPRING-2015/lect01-overview.pdf**

WPI

More specifically...

- What is the core issue:
  - CPU?
  - Memory?
  - Disk space?
  - Network Access?

Warmup: An example of a problem where the CPU is the bottleneck?

Computing complicated queries

Heavy math

Grid search

graph algorithms

Warmup: An example of a problem where the memory is the bottleneck?

compute matrix inverse

k-NN

small log processing

machine learning on small data

Warmup: An example of a problem where the disk is the bottleneck?

SQL, noSQL...

Large Logs

Games video processing

<u>Warmup:</u> An example of a problem where the network is the bottleneck?

Getting from twitter API

Remote databases

Bittorrent

Youtube ...

Modern trend: GPUs aren't just for gaming....

https://www.flickr.com/photos/juanpol/8232592/in/photostream/

They are for linear algebra!

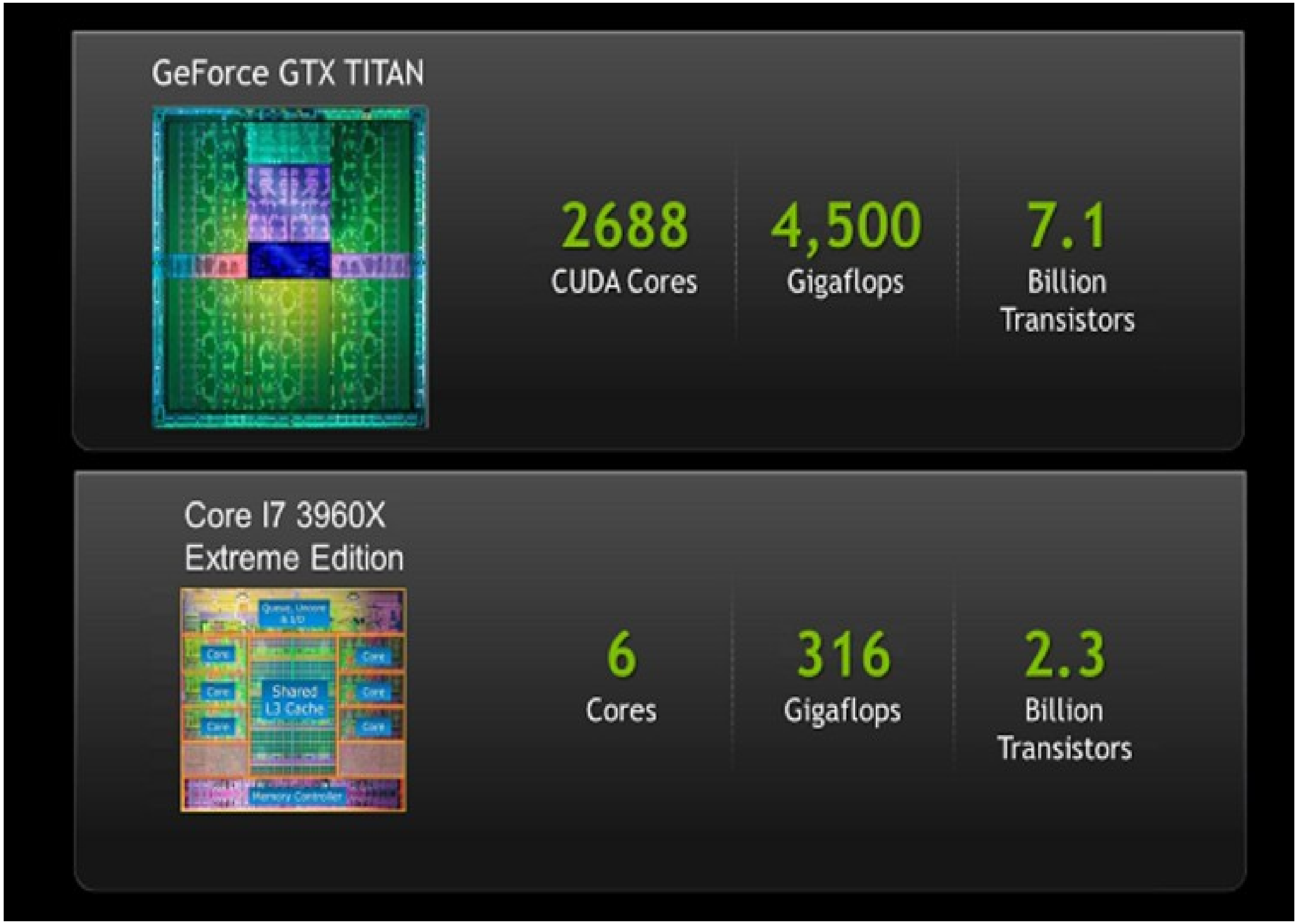
Nvidia Tesla K40

"NvidiaTesla" by Mahogny - CameraTransferred from en.wikipedia. Licensed under Public domain via Wikimedia Commons - http://commons.wikimedia.org/wiki/File:NvidiaTesla.jpg#mediaviewer/File:NvidiaTesla.jpg

Example: CPU versus GPU

GPU

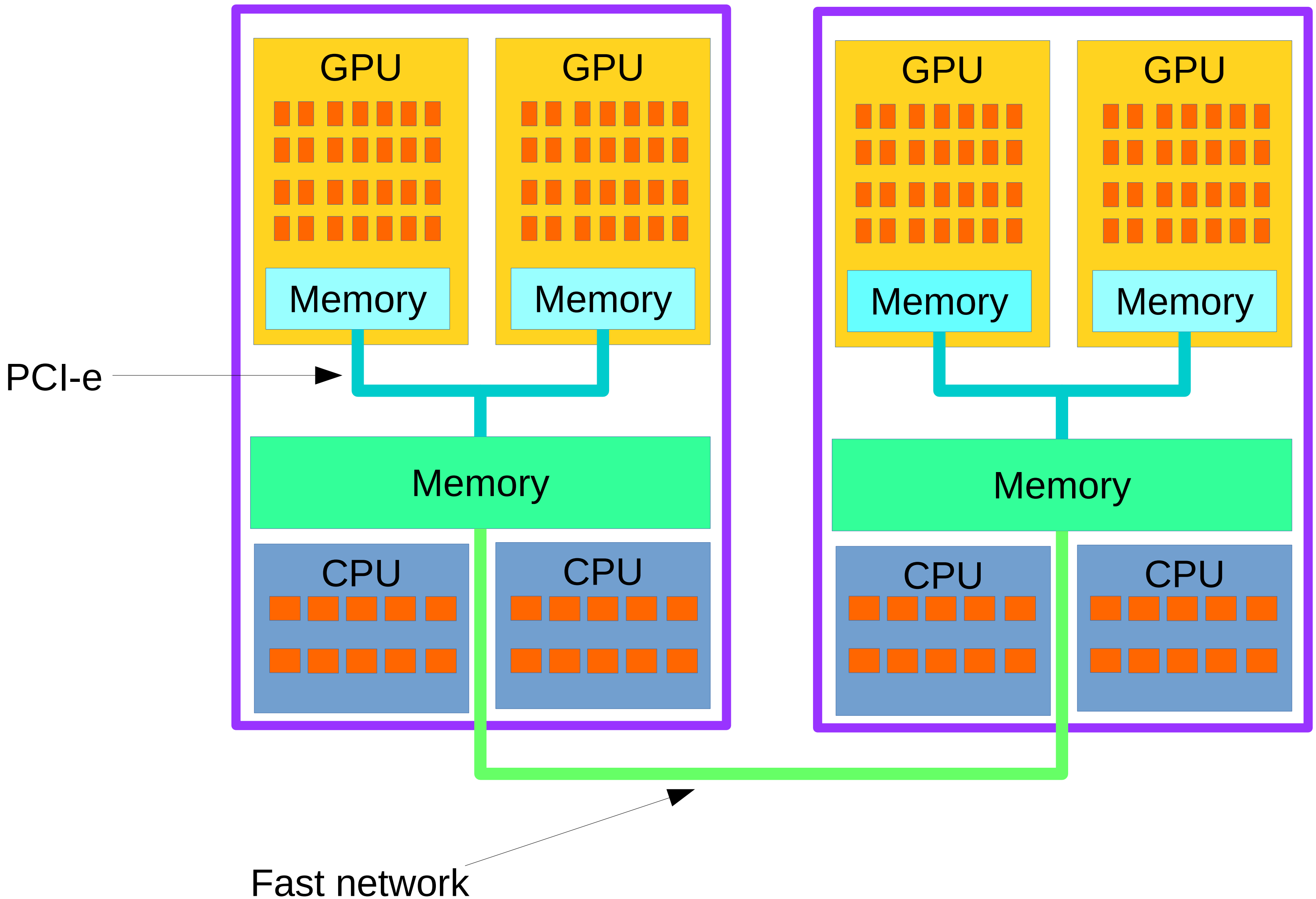
CPU



http://cdn3.wccftech.com/wp-content/uploads/2013/02/er_photo_184846_52.png

WPI

Modern High Performance Computing (HPC) architecture



Amazing computational power...

"Titan render" by Unknown - http://www.olcf.ornl.gov/2012/12/17/titan-trainers-take-road-trip/. Licensed under Public domain via Wikimedia Commons - http://commons.wikimedia.org/wiki/File:Titan_render.png#mediaviewer/File:Titan_render.png

http://oakridgetoday.com/wp-content/uploads/2012/12/jeff-nichols-and-titan-at-ornl.jpg

Titan has 18,688 nodes (4 nodes per blade, 24 blades per cabinet)

- each containing a 16-core AMD Opteron 6274 CPU with 32 GB of DDR3 ECC memory and
- an Nvidia Tesla K20X GPU with 6 GB GDDR5 ECC memory.

There are a total of 299,008 processor cores, and a total of 693.6 TiB of CPU and GPU RAM

AWS let’s you rent these!

GPU

G2

This family includes G2 instances intended for graphics and general purpose GPU compute applications.

Features:

- High Frequency Intel Xeon E5-2670 (Sandy Bridge) Processors
- High-performance NVIDIA GPUs, each with 1,536 CUDA cores and 4GB of video memory
- Each GPU features an on-board hardware video encoder designed to support up to eight real-time HD video streams (720p@30fps) or up to four real-time full HD video streams (1080p@30fps)
- Support for low-latency frame capture and encoding for either the full operating system or select render targets, enabling high-quality interactive streaming experiences

Model	GPUs	vCPU	Mem (GiB)	SSD Storage (GB)
g2.2xlarge	1	8	15	1 x 60
g2.8xlarge	4	32	60	2 x 120

Use Cases

3D application streaming, machine learning, video encoding, and other server-side graphics or GPU compute workloads.

GPU Instances - Current Generation

g2.2xlarge	8	26	15	60 SSD	$0.65 per Hour
g2.8xlarge	32	104	60	2 x 120 SSD	$2.6 per Hour

Are we done? Can we go home early tonight?

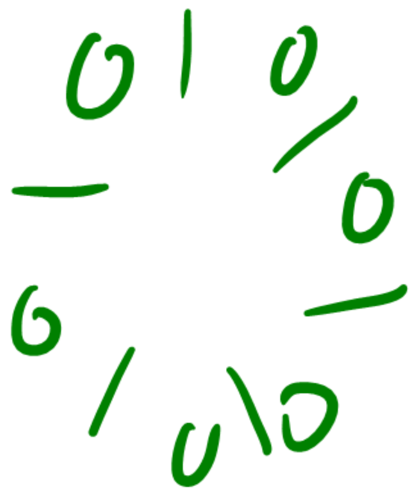
Big issues

automating the use of TITAN

is a very hard problem

Distributed computing is hard...

- I mean really hard.
  - Parallelization
  - Synchronization
  - Resource contention
    - Deadlock
    - Dining Philosophers...
  - Fault Tolerance
  - Distributed I/O
  - Etc.



But can’t you automate it?

- There have been many tries.
  - For example, many extensions based on Fortran 90.
- Doing anything like this in **general** is very hard.
  - I mean, you can’t even solve the **Halting Problem,** much less more general problems.
- However, specific **subsets** of the problem have shown great progress.

Case study:

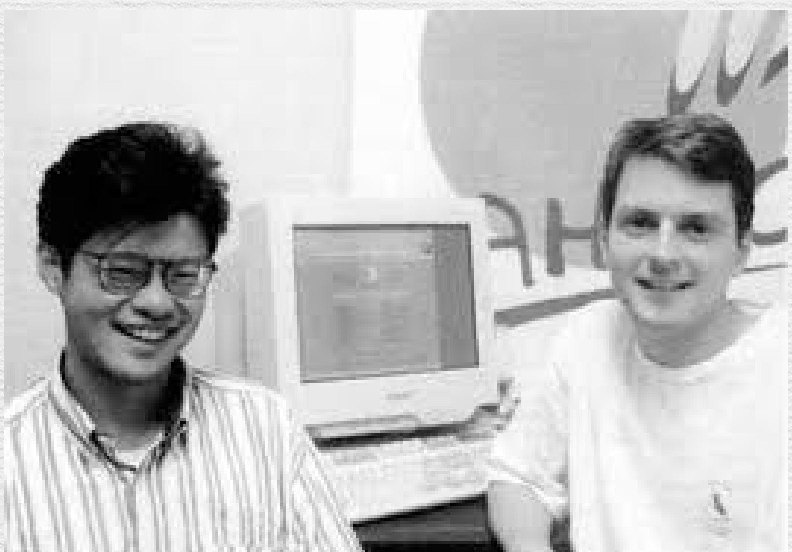
Organizing the Web

1994

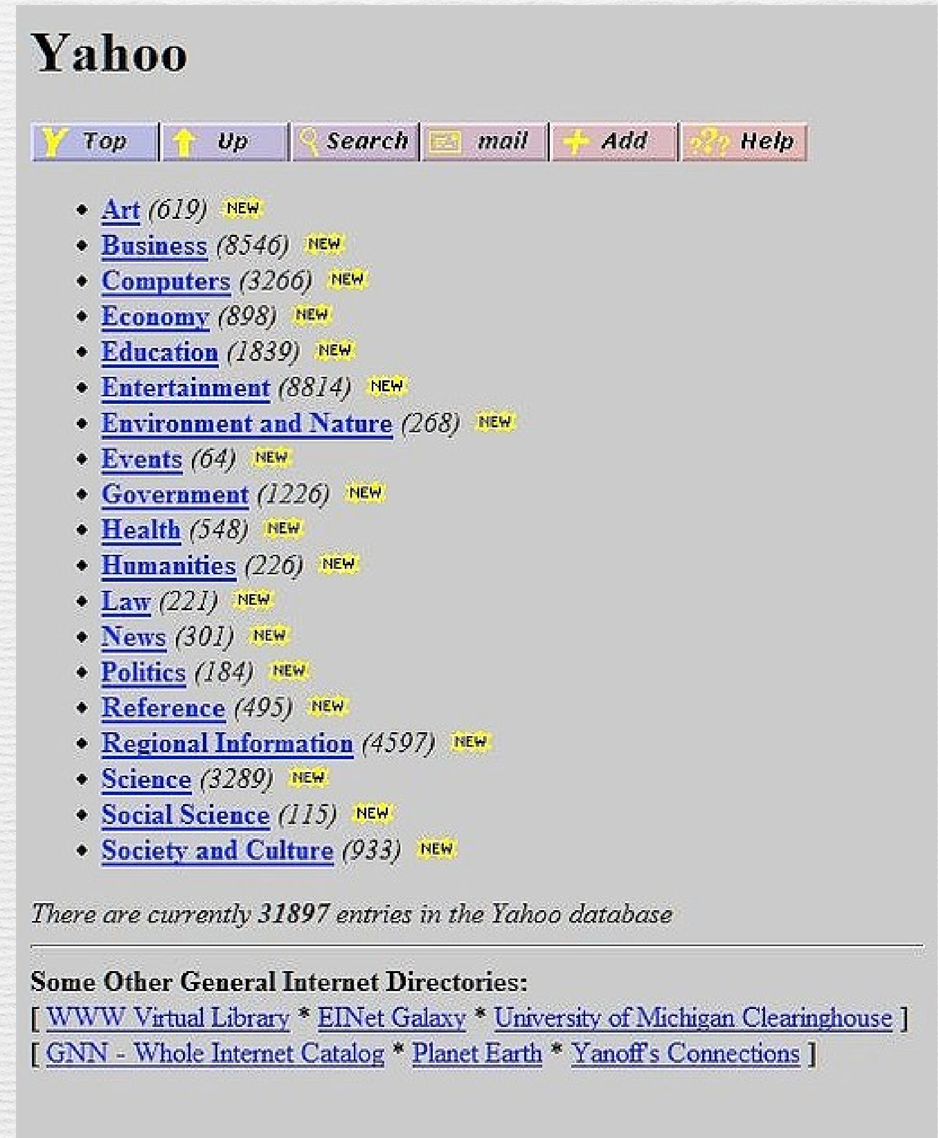
Jerry and David's Guide to the World Wide Web

How was the Web organized?

Find Webpage



Jerry Yang and David Filo



Yahoo

[Top | Up | Search | mail | Add | Help]

- Art (619) NEW
- Business (8546) NEW
- Computers (3266) NEW
- Economy (898) NEW
- Education (1839) NEW
- Entertainment (8814) NEW
- Environment and Nature (268) NEW
- Events (64) NEW
- Government (1226) NEW
- Health (548) NEW
- Humanities (226) NEW
- Law (221) NEW
- News (301) NEW
- Politics (184) NEW
- Reference (495) NEW
- Regional Information (4597) NEW
- Science (3289) NEW
- Social Science (115) NEW
- Society and Culture (933) NEW

*There are currently **31897** entries in the Yahoo database*

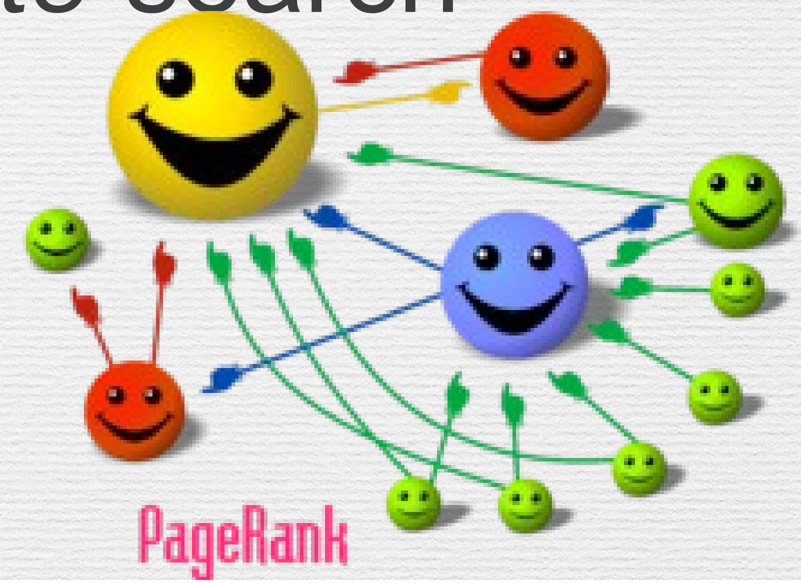
Some Other General Internet Directories:

[WWW Virtual Library * EINet Galaxy * University of Michigan Clearinghouse]

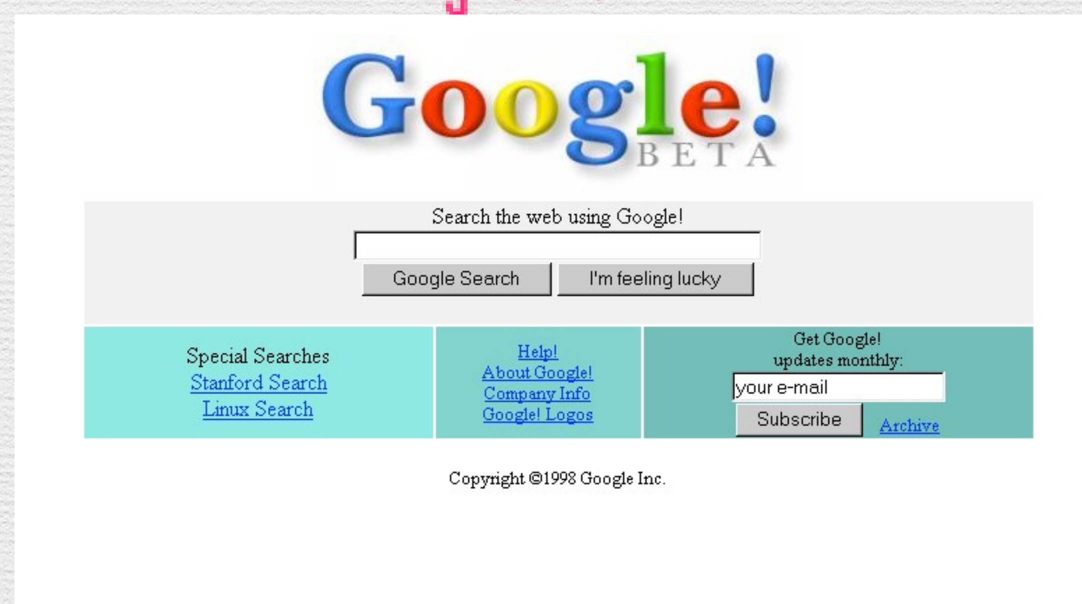
[GNN - Whole Internet Catalog * Planet Earth * Yanoff's Connections]

1997

The desire to *automate* search



Larry Page and Sergey Brin



The PageRank Citation Ranking: Bringing Order to the Web

January 29, 1998

Abstract

The importance of a Web page is an inherently subjective matter, which depends on the readers interests, knowledge and attitudes. But there is still much that can be said objectively about the relative importance of Web pages. This paper describes PageRank, a method for rating Web pages objectively and mechanically, effectively measuring the human interest and attention devoted to them.

We compare PageRank to an idealized random Web surfer. We show how to efficiently compute PageRank for large numbers of pages. And, we show how to apply PageRank to search and to user navigation.

1 Introduction and Motivation

The World Wide Web creates many new challenges for information retrieval. It is very large and heterogeneous. Current estimates are that there are over 150 million web pages with a doubling life of less than one year. More importantly, the web pages are extremely diverse, ranging from "What is Joe having for lunch today?" to journals about information retrieval. In addition to these major challenges, search engines on the Web must also contend with inexperienced users and pages engineered to manipulate search engine ranking functions.

However, unlike "flat" document collections, the World Wide Web is hypertext and provides considerable auxiliary information on top of the text of the web pages, such as link structure and

The PageRank Paper

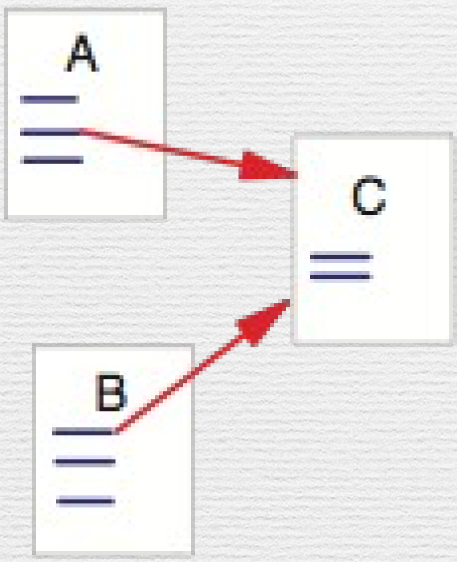


Figure 1: A and B are Backlinks of C

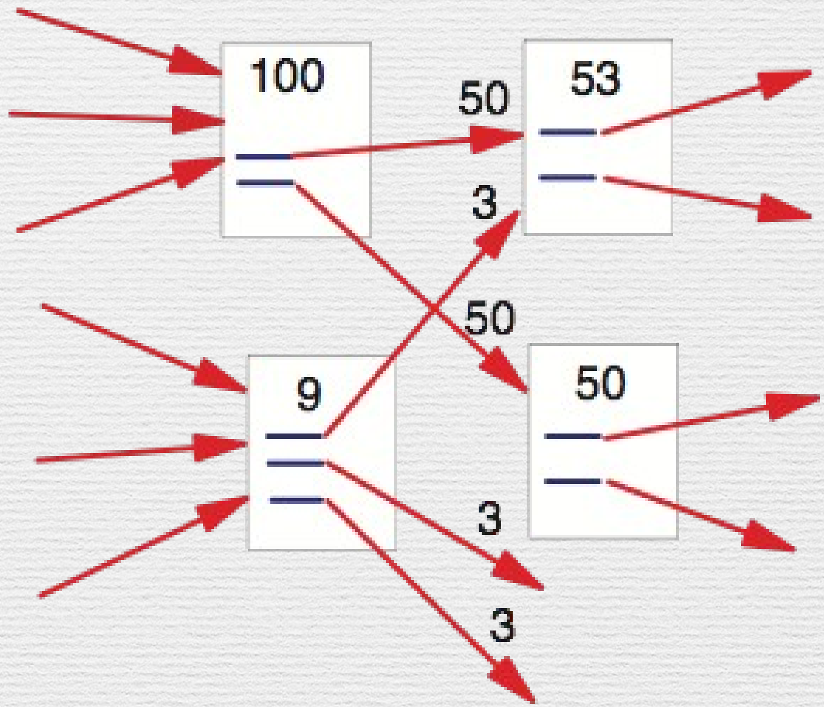


Figure 2: Simplified PageRank Calculation

MapReduce: History

Some slides based on:

cecs.wright.edu/~tkprasad/courses/cs707/L06MapReduce.ppt

www.eecg.toronto.edu/~amza/ece1747h/slides/MapReduce.1.4.pptx

2003

MapReduce

Sanjay Ghemawat and Jeffrey Dean

How do you make large scale, data centric parallelism accessible for the masses?

Any ideas?

Where do you focus?

MAP Reduce

Happy Place

Abstract

everything
else
automatically

Algorithm
correctly

MapReduce: Simplified Data Processing on Large Clusters

Jeffrey Dean and Sanjay Ghemawat

jeff@google.com, sanjay@google.com

Google, Inc.

Abstract

MapReduce is a programming model and an associated implementation for processing and generating large data sets. Users specify a *map* function that processes a key/value pair to generate a set of intermediate key/value pairs, and a *reduce* function that merges all intermediate values associated with the same intermediate key. Many real world tasks are expressible in this model, as shown in the paper.

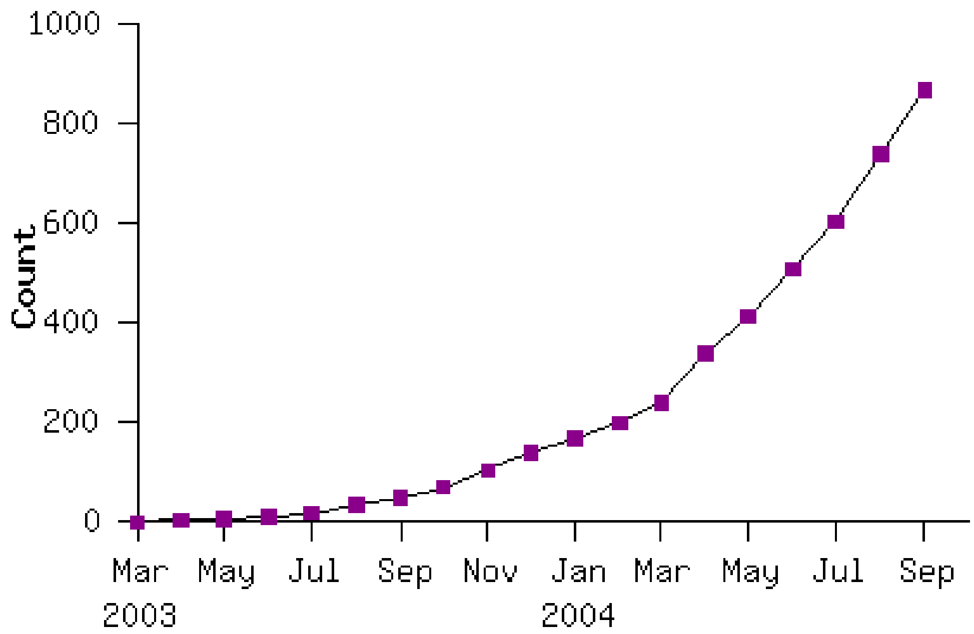
Programs written in this functional style are automatically parallelized and executed on a large cluster of commodity machines. The run-time system takes care of the details of partitioning the input data, scheduling the pro-

given day, etc. Most such computations are conceptually straightforward. However, the input data is usually large and the computations have to be distributed across hundreds or thousands of machines in order to finish in a reasonable amount of time. The issues of how to parallelize the computation, distribute the data, and handle failures conspire to obscure the original simple computation with large amounts of complex code to deal with these issues.

As a reaction to this complexity, we designed a new abstraction that allows us to express the simple computations we were trying to perform but hides the messy details of parallelization, fault-tolerance, data distribution and load balancing in a library. Our abstraction is in-

Model is Widely Applicable

MapReduce Programs In Google Source Tree

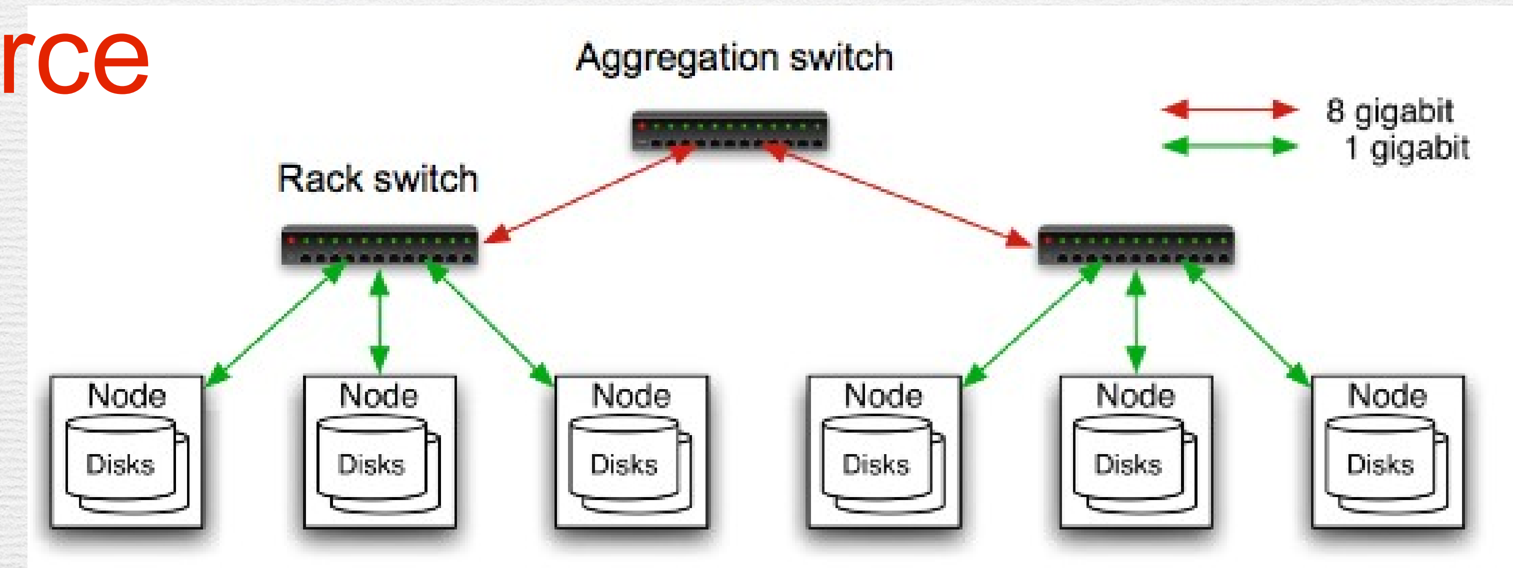


Example uses:

distributed grep	distributed sort	web link-graph reversal
term-vector per host	web access log stats	inverted index construction
document clustering	machine learning	statistical machine translation
...	...	...

2005

MapReduce *Implementation* + Open source



Doug Cutting and Mike Cafarella

now used by Yahoo Facebook Amazon

MapReduce Vs. Hadoop

- MapReduce is an **idea**
  - **A way to organize code so that it is easy to parallelize.**
- Hadoop is (one) implementation of the MapReduce idea.
  - hadoop.apache.org

Who has it?

- Google:
  - Original proprietary implementation
- Apache Hadoop MapReduce
  - Most common (open-source) implementation
  - Built to specs defined by Google
- Amazon Elastic MapReduce
  - Uses Hadoop MapReduce running on Amazon EC2

MapReduce/Hadoop: Diving more deeply

Some slides based on:

cecs.wright.edu/~tkprasad/courses/cs707/L06MapReduce.ppt

www.eecg.toronto.edu/~amza/ece1747h/slides/MapReduce.1.4.pptx

MapReduce

- Programming model for distributed computations
- Software framework for clusters
- Massive data processing
- No hassle with low level programming
  - Partitioning input data
  - Scheduling execution
  - Handling failures
  - Intermachine communication

- Open source implementation
- MRJob: Python class for Hadoop Streaming

Part 1: Distributed Filesystem

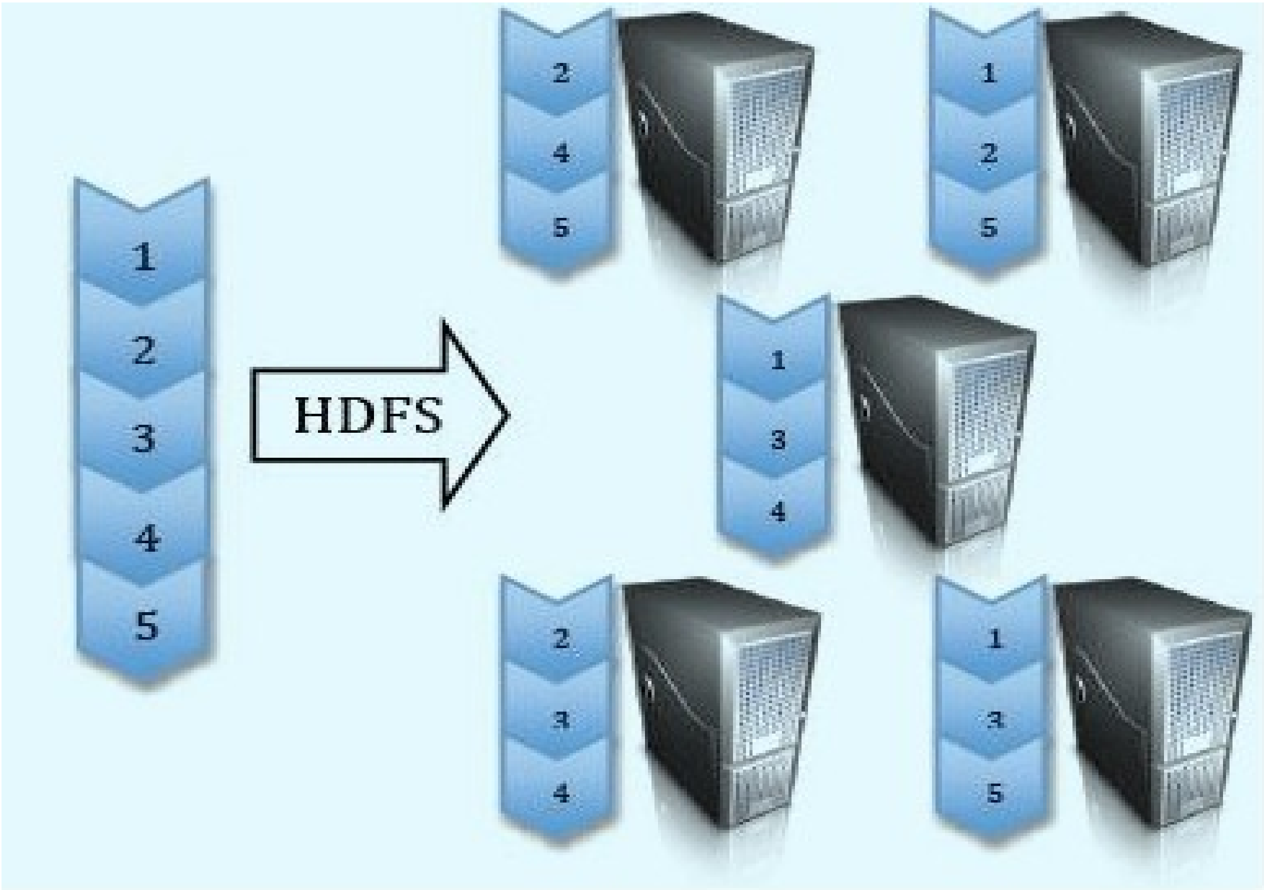
Stable storage

- First order problem: if nodes can fail, how can we store data persistently?
- Answer: Distributed File System
  - Provides global file namespace
  - Google GFS; Hadoop HDFS; Kosmix KFS
- Typical usage pattern
  - Huge files (100s of GB to TB)
  - Data is rarely updated in place
  - Reads and appends are common

cecs.wright.edu/~tkprasad/courses/cs707/L06MapReduce.ppt

Google File System (GFS)
Hadoop Distributed File System (HDFS)

- Split data and store 3 replica on commodity servers



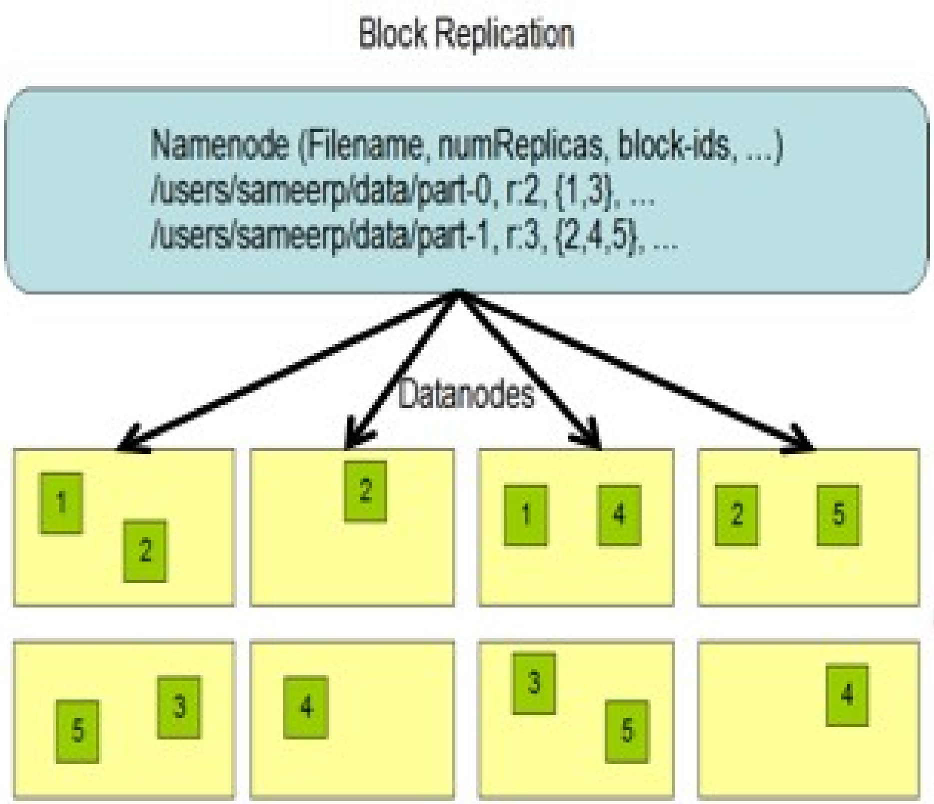
www.eecg.toronto.edu/~amza/ece1747h/slides/MapReduce.1.4.pptx

Distributed File System

- Chunk Servers
  - File is split into contiguous chunks
  - Typically each chunk is 16-64MB
  - Each chunk replicated (usually 2x or 3x)
  - Try to keep replicas in different racks
- Master node
  - a.k.a. Name Nodes in HDFS
  - Stores metadata
  - Might be replicated
- Client library for file access
  - Talks to master to find chunk servers
  - Connects directly to chunkservers to access data

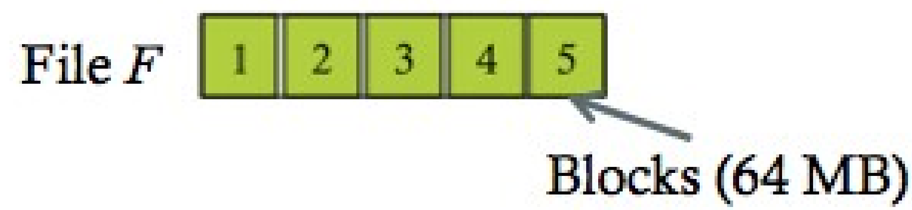
cecs.wright.edu/~tkprasad/courses/cs707/L06MapReduce.ppt

Hadoop Distributed File System (HDFS)



Centralized namenode

- Maintains metadata info about files



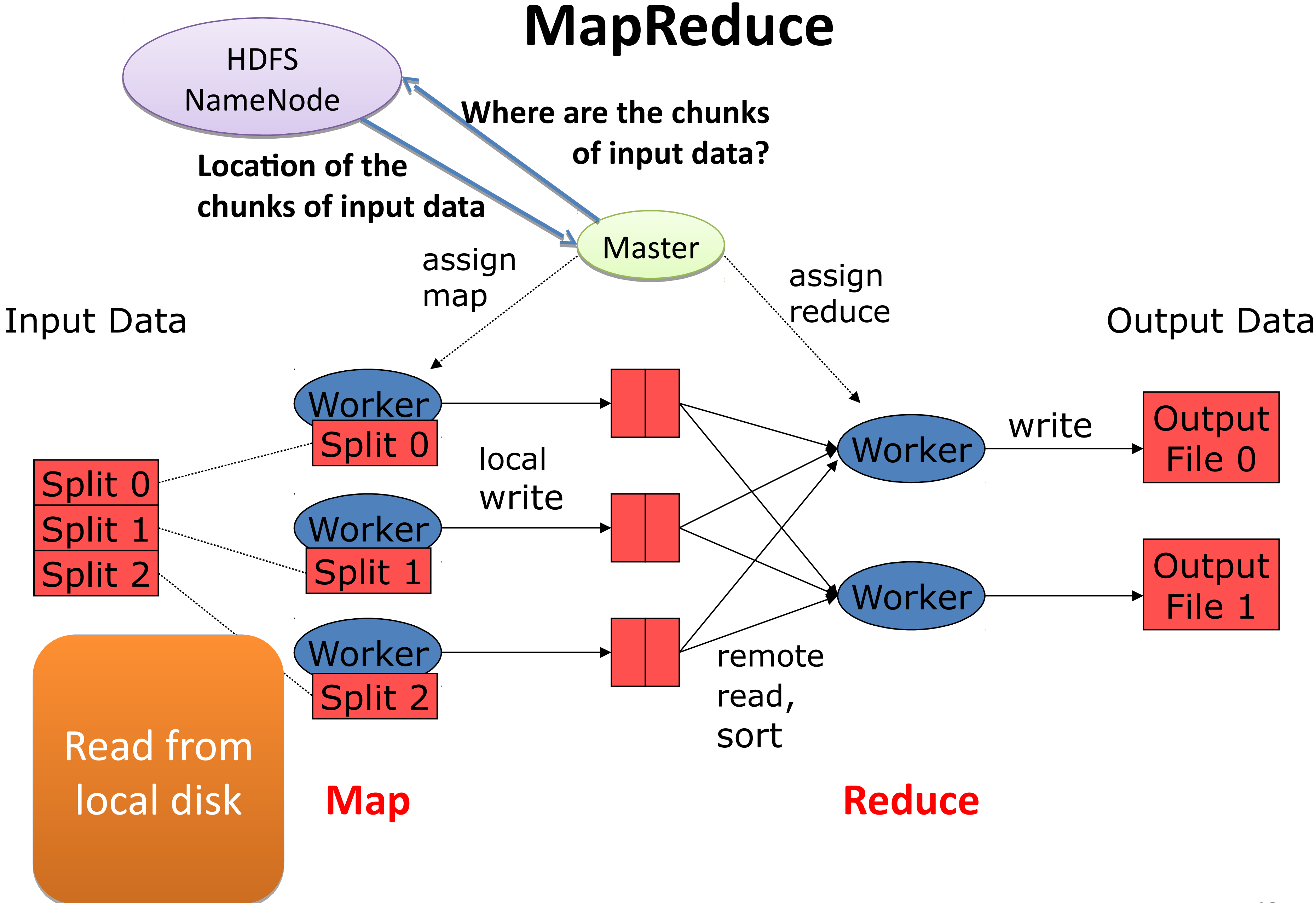
Many datanode (1000s)

- Store the actual data
- Files are divided into blocks
- Each block is replicated *N* times (Default = 3)

Main Properties of HDFS

- ***Large:*** A HDFS instance may consist of thousands of server machines, each storing part of the file system's data
- ***Replication:*** Each data block is replicated many times (default is 3)
- ***Failure:*** Failure is the norm rather than exception
- ***Fault Tolerance:*** Detection of faults and quick, automatic recovery from them is a core architectural goal of HDFS
  - Namenode is consistently checking Datanodes

MapReduce



www.eecg.toronto.edu/~amza/ece1747h/slides/MapReduce.1.4.pptx

Part 1: The “map and reduce” part

MapReduce

- Map

  Grab the relevant data from the source

  User function gets called for each chunk of input

- Reduce

  Aggregate the results

  User function gets called for each unique key

Map example

- (map ***f* *list* *[list*$_2$ *list*$_3$ *…]***)

Unary operator

- (map square ‘(1 2 3 4))
  - (1 4 9 16)

cecs.wright.edu/~tkprasad/courses/cs707/L06MapReduce.ppt

Reduce Example

- (reduce ***f id list***)

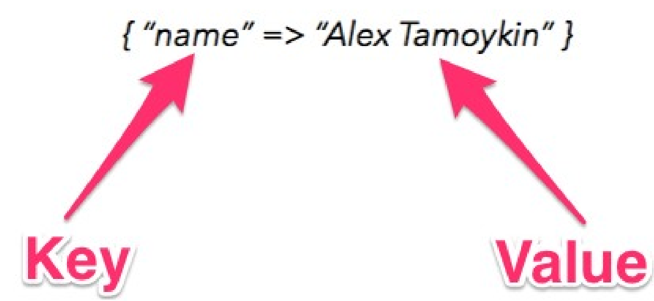
Binary operator

- (reduce + 0 ‘(1 4 9 16))
  - (+ 16 (+ 9 (+ 4 (+ 1 0))))
  - 30

cecs.wright.edu/~tkprasad/courses/cs707/L06MapReduce.ppt

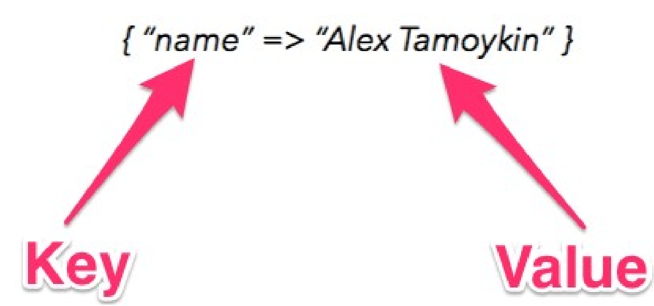
Key- Value Pairs

- Mappers and Reducers are users' code (provided functions)
- Just need to obey the Key-Value pairs interface
- **Mappers:**
  - Consume <key, value> pairs
  - Produce <key, value> pairs
- **Reducers:**
  - Consume <key, <list of values>>
  - Produce <key, value>

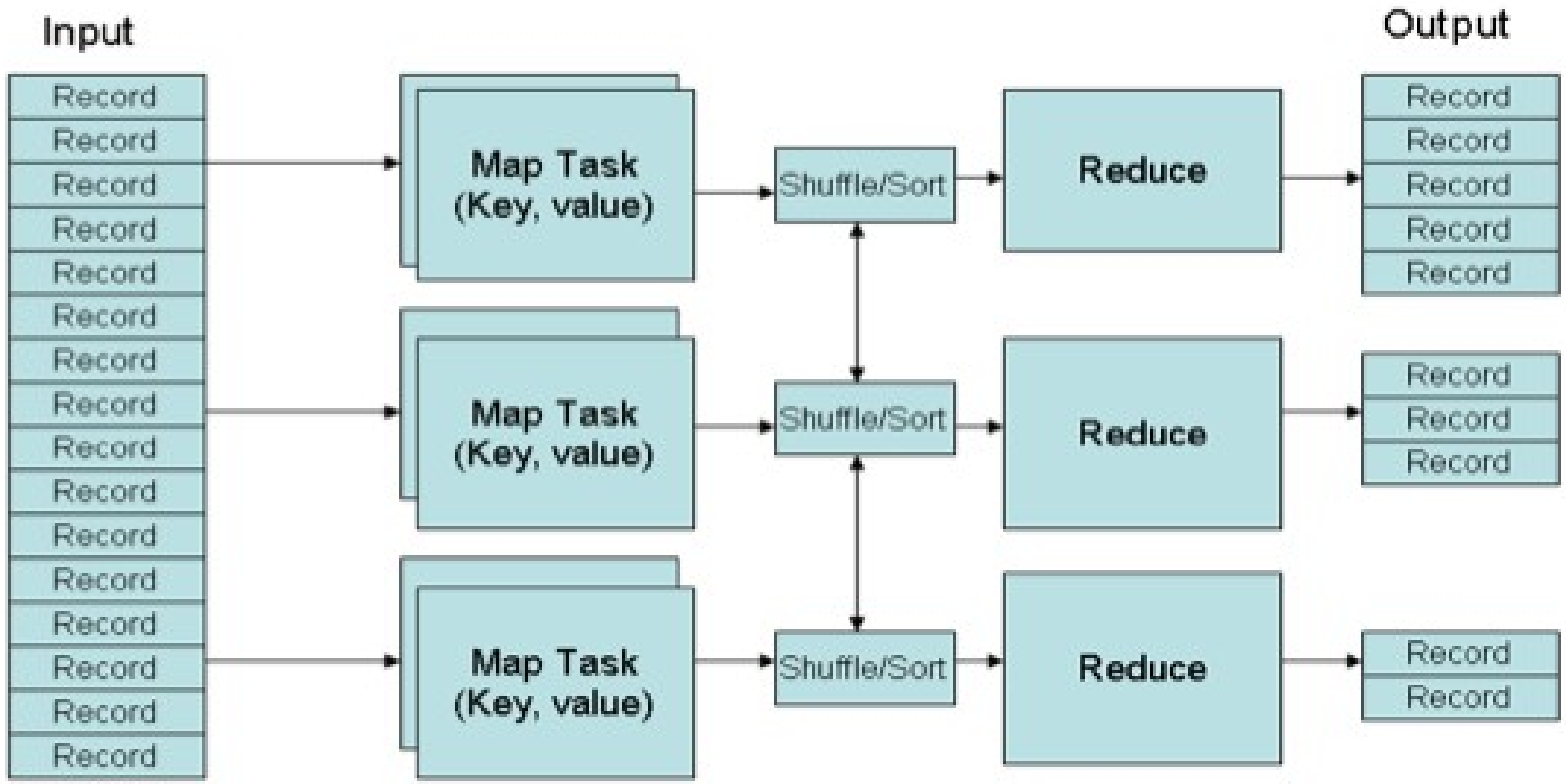


Key- Value Pairs

- Mappers and Reducers are users' code (provided functions)
- Just need to obey the Key-Value pairs interface
- **Mappers:**
  - Consume <key, value> pairs
  - Produce <key, value> pairs
- **Reducers:**
  - Consume <key, <list of values>>
  - Produce <key, value>
- **Shuffling and Sorting:**
  - Hidden phase between mappers and reducers
  - Groups all similar keys from all mappers, sorts and passes them to a certain reducer in the form of <key, <list of values>>



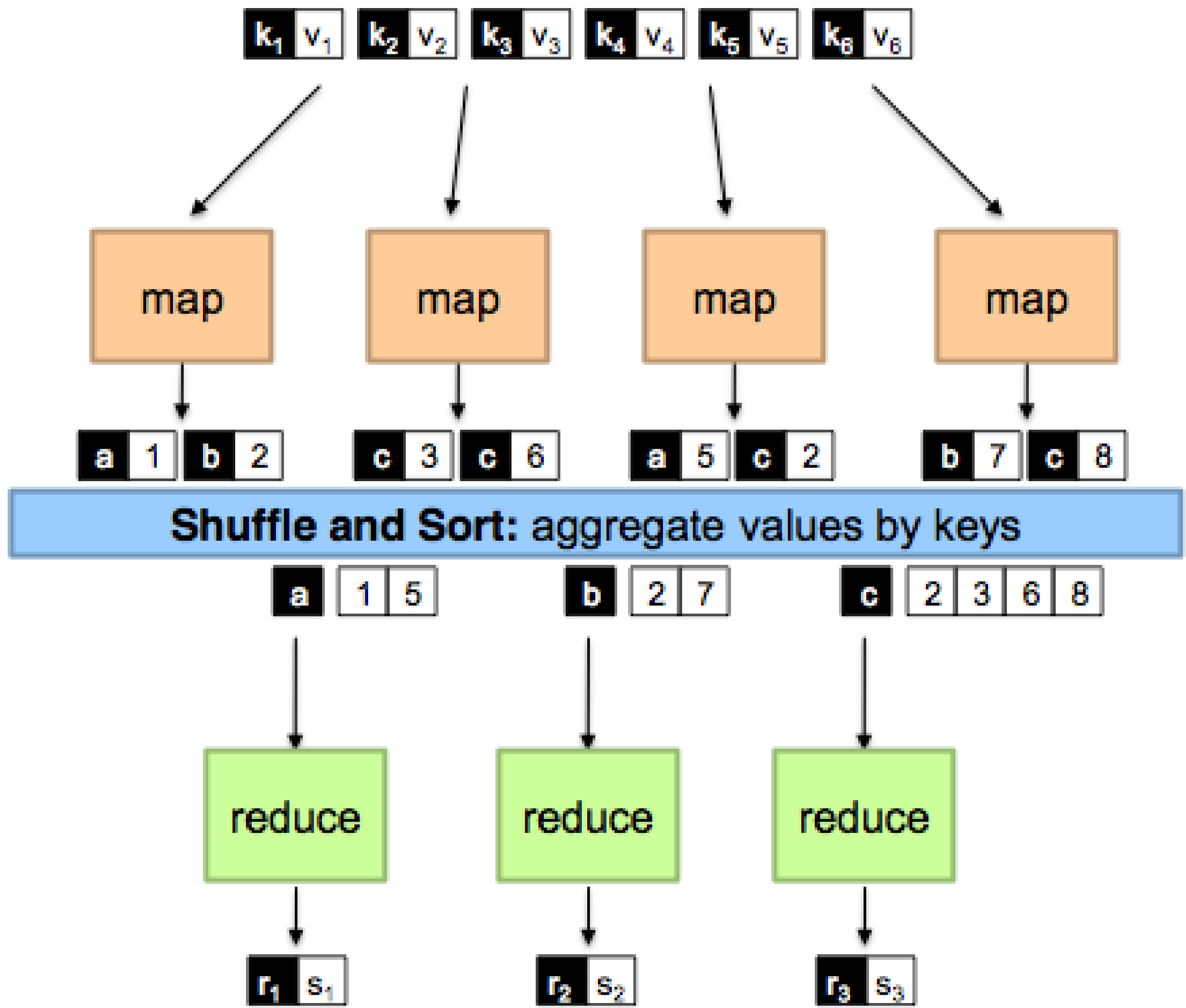
MapReduce Phases



Deciding on what will be the key and what will be the value ➔ developer's responsibility

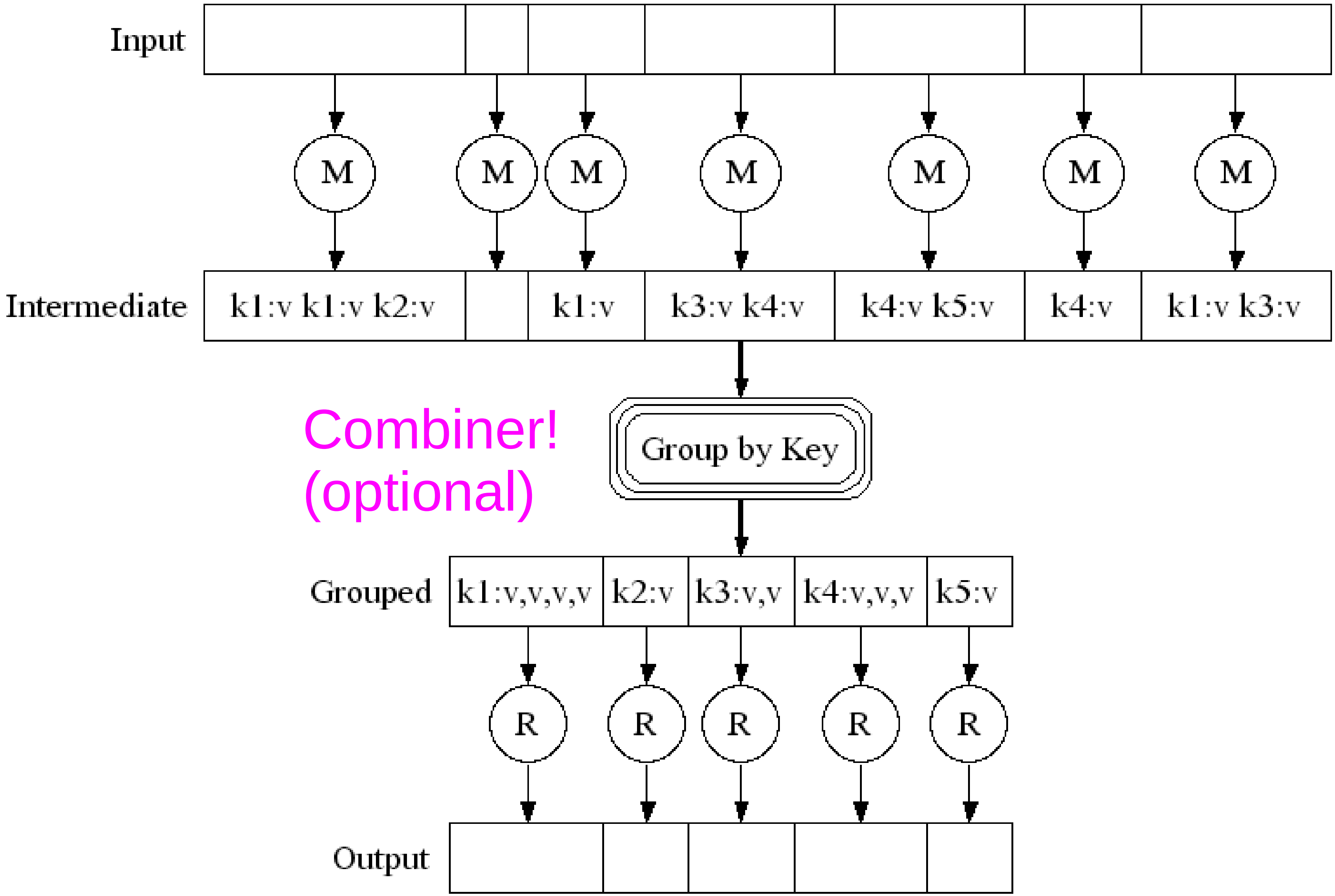
function 1

function 2



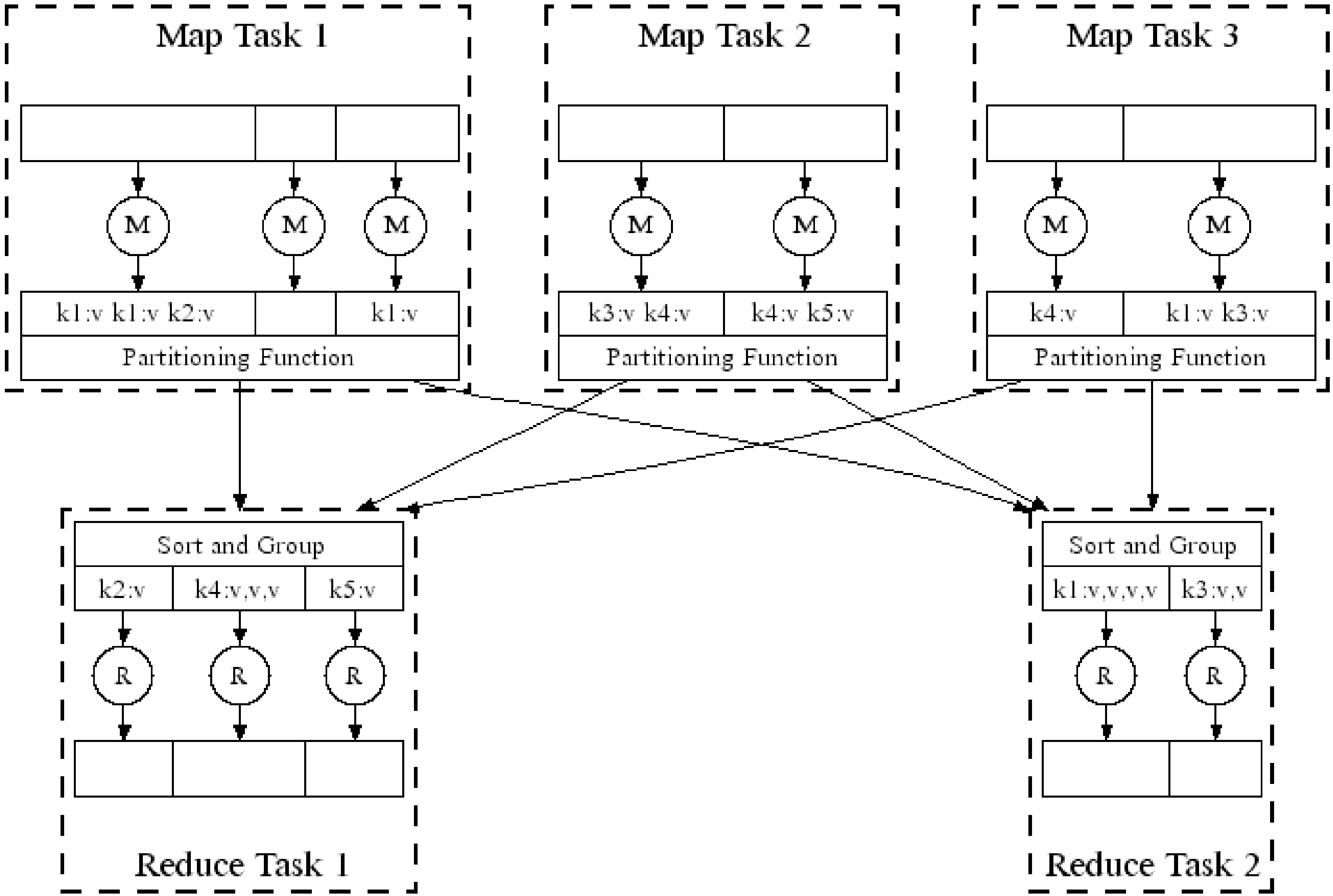
What it really should be called...

- Not “MapReduce”
- But “MapShuffleReduce”


Input
M
M
M
M
M
M
M
Intermediate
k1:v k1:v k2:v
k1:v
k3:v k4:v
k4:v k5:v
k4:v
k1:v k3:v
Combiner!
(optional)
Group by Key
Grouped
k1:v,v,v,v
k2:v
k3:v,v
k4:v,v,v
k5:v
R
R
R
R
R
Output

What it really should be called...

- Not “MapReduce”
- But “MapShuffleCombineReduce”



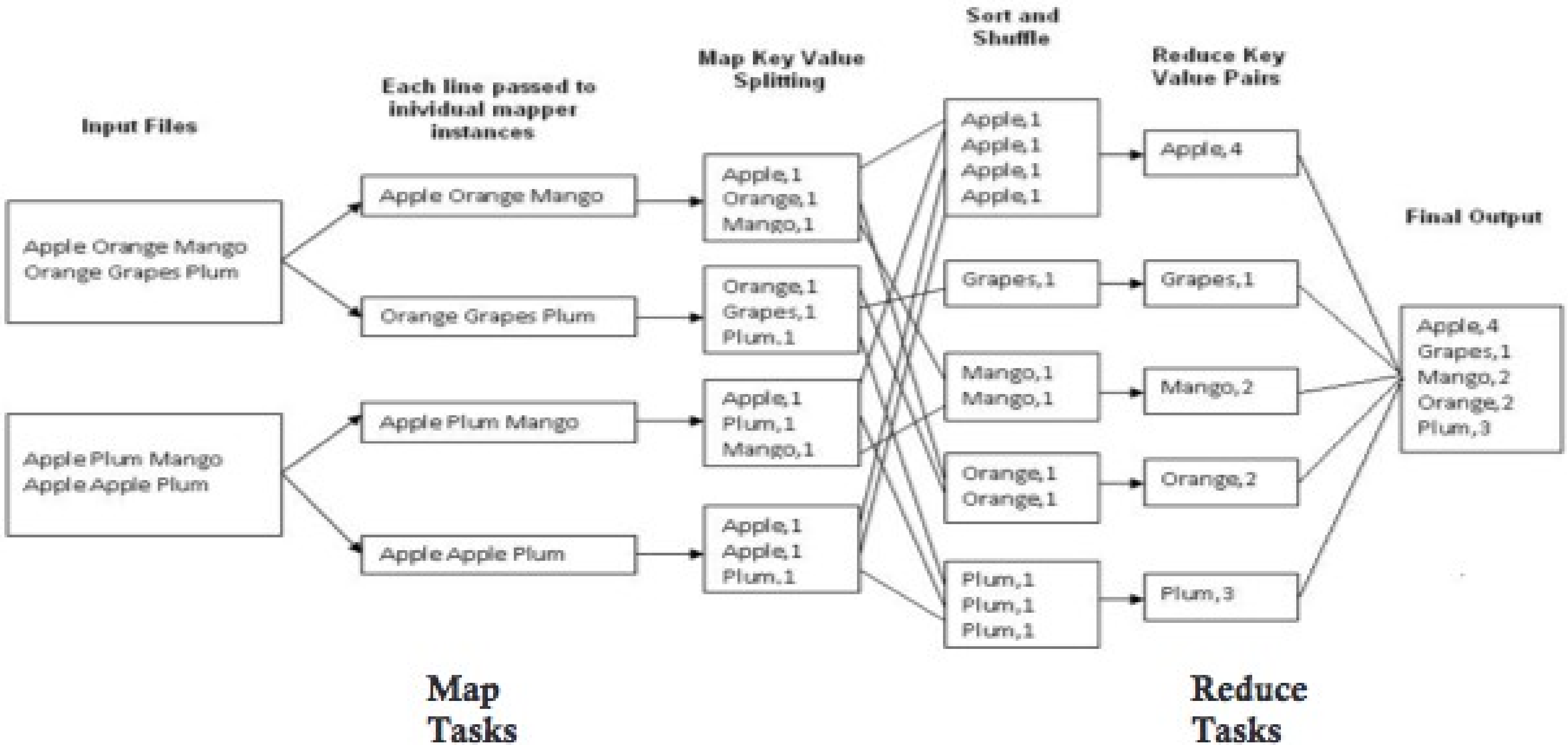
Map Task 1
Map Task 2
Map Task 3
M
k1:v k1:v k2:v
k1:v
k3:v k4:v
k4:v k5:v
k4:v
k1:v k3:v
Partitioning Function
Sort and Group
k2:v
k4:v,v,v
k5:v
k1:v,v,v,v
k3:v,v
R
Reduce Task 1
Reduce Task 2

What it really should be called...

- Not “MapReduce”
- But “MapShuffleCombinePartitionReduce”

Example 1: Word Count

- **Job: Count the occurrences of each word in a data set**



Example: Count word occurrences

```
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");


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));
```

MRjob package

pythonhosted.org/mrjob/guides/quickstart.html

mrjob v0.4.2 documentation

Table Of Contents

Need help?

Join the mailing list by visiting the Google group page or sending an email to mrjob+subscribe @googlegroups.com.

type to search

Fundamentals

Installation

Install with `pip`:

```
pip install mrjob
```

or from a git clone of the source code:

```
python setup.py test && python setup.py install
```

Writing your first job

Open a file called `word_count.py` and type this into it:

```
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)
```

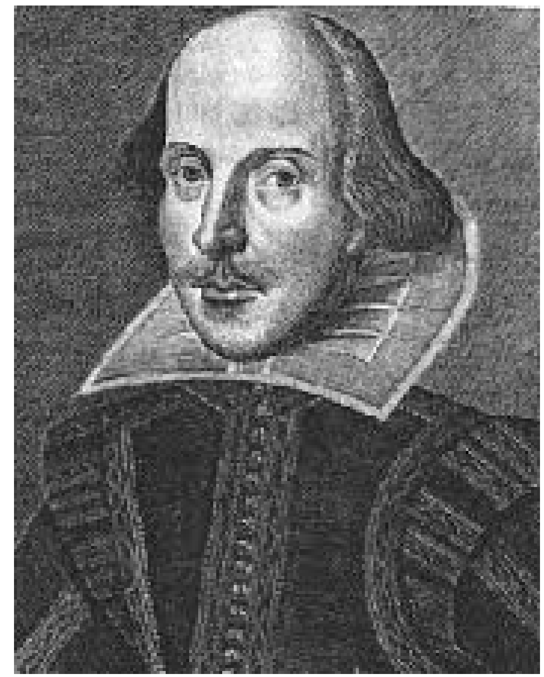
http://pythonhosted.org/mrjob/guides/quickstart.html

The Famous Word Count Example

```
from mrjob.job import MRJob

class mrWordCount(MRJob):

    def mapper(self,key,line):
        for word in line.split(' '):
            yield word.lower(),1

    def reducer(self, word, occurrences):
        yield word, sum(occurrences)

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

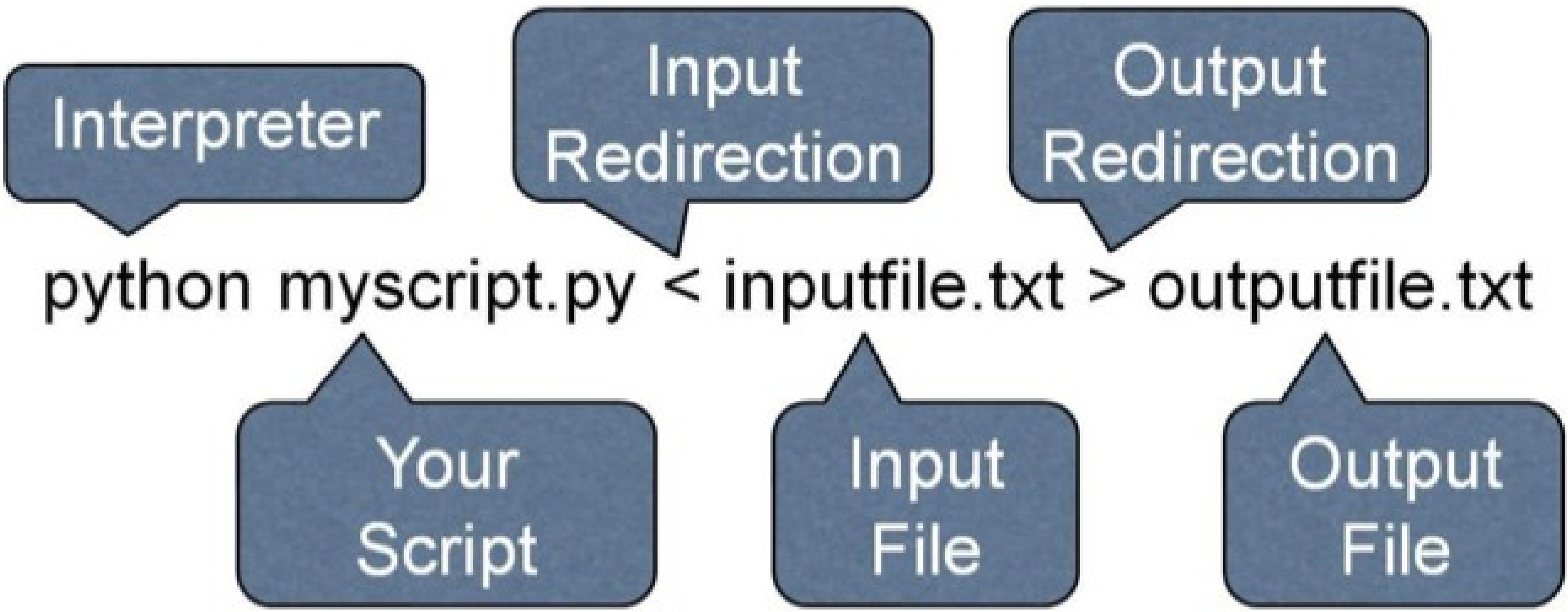
Example Input Data

Hamlet

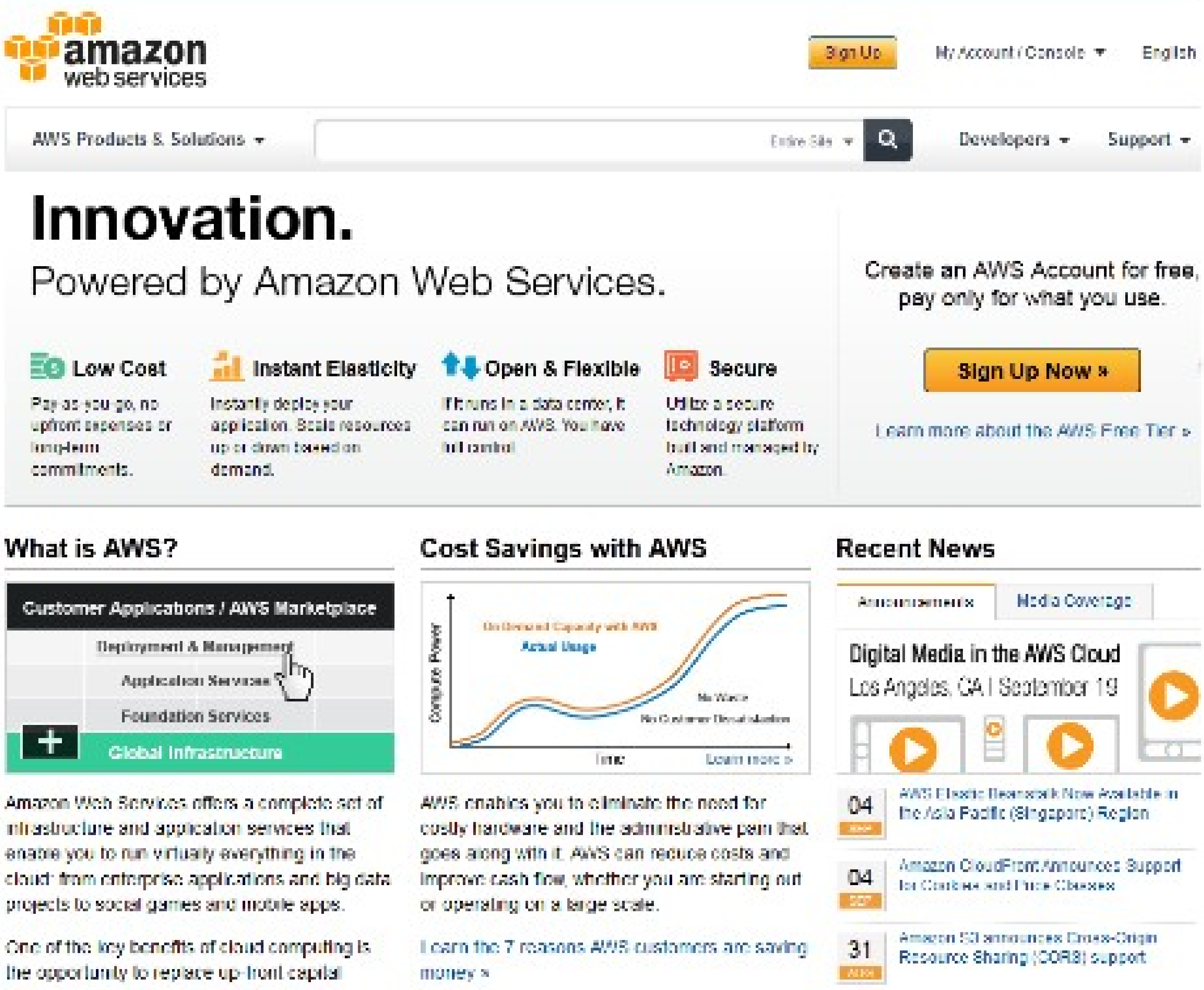


Shakespeare

Launching the Job



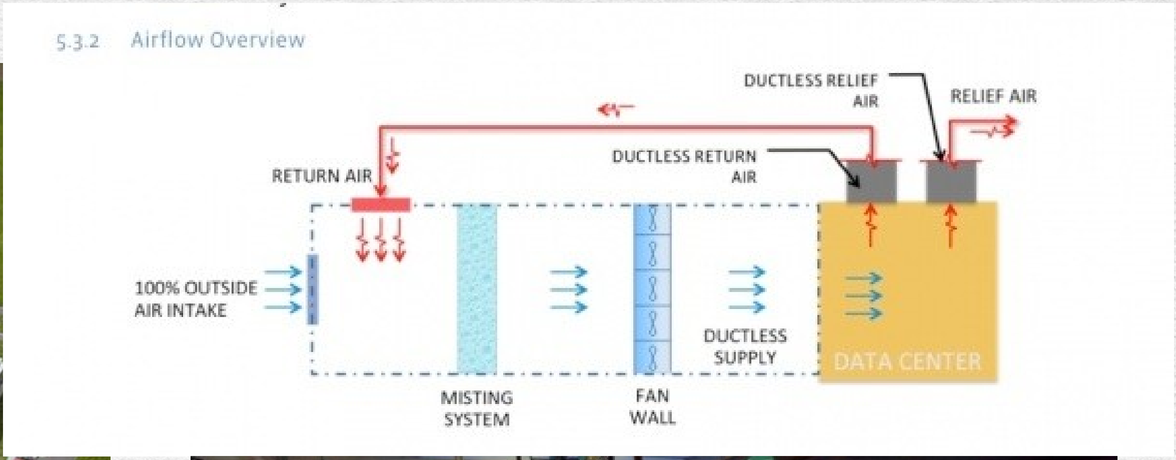
Go to aws.amazon.com

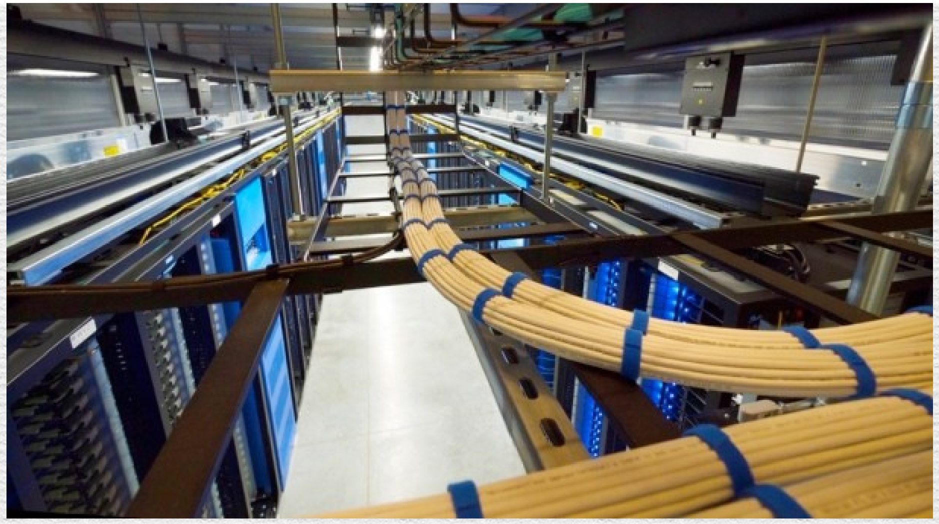


2011

Data Center Design + Open source

OPEN Compute Project





https://www.youtube.com/watch?v=Y8Rgje94iI0

facebook

Example 2: Color Count

Job: Count the number of each color in a data set