Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Data Mining
Learning from Large Data Sets

Lecture 1 – Introduction

263-5200-00L

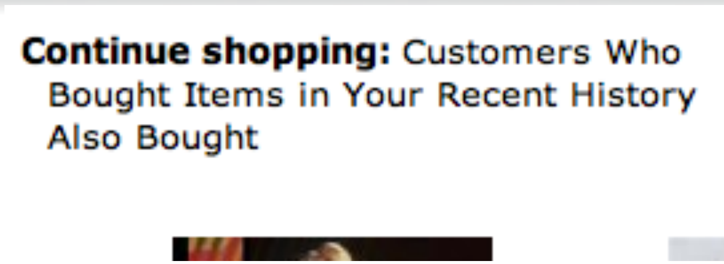
Andreas Krause

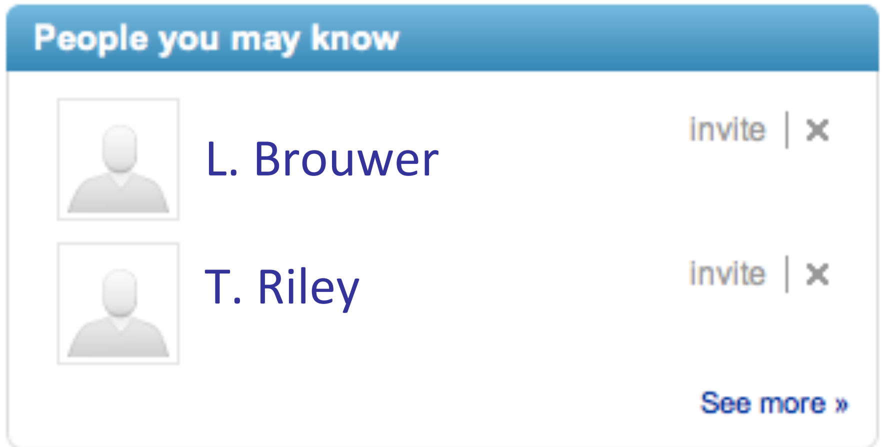
How can we **extract useful information** from **massive, noisy data sets?**

Web-scale machine learning / DM

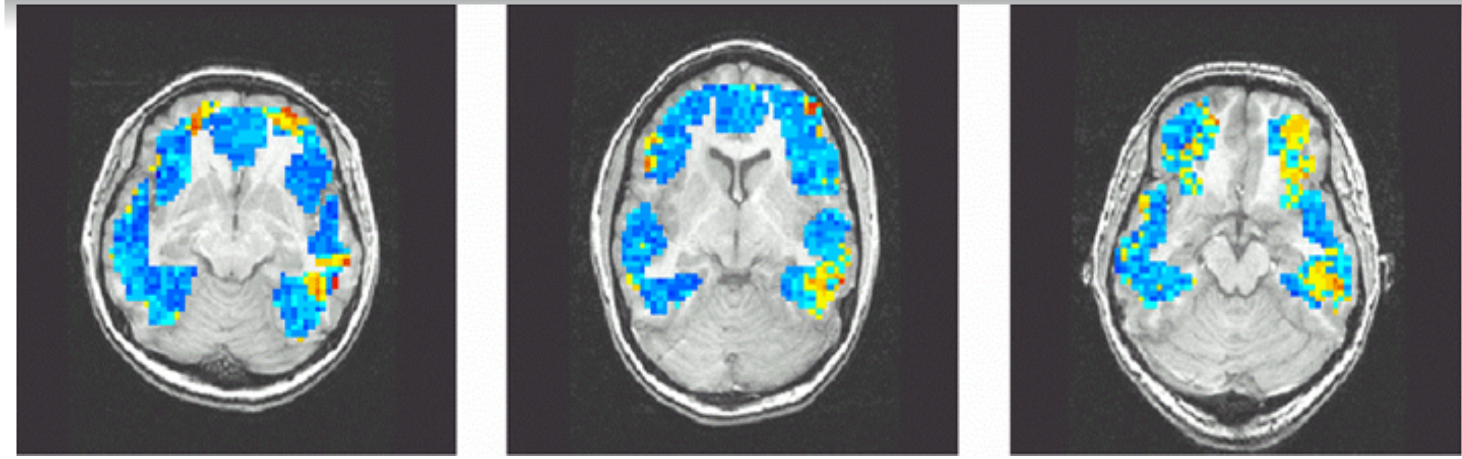
- Recommender systems
- Online advertising
- Predict relevance of search results from click data
- Learning to index
- Machine translation
- Spam filtering
- Fraud detection
- …

>21 billion indexed web pages





Analyzing fMRI data

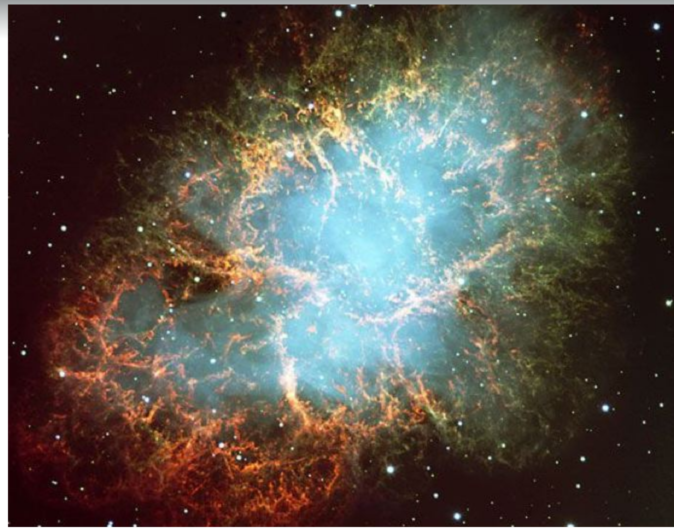


Mitchell et al., *Science,* 2008

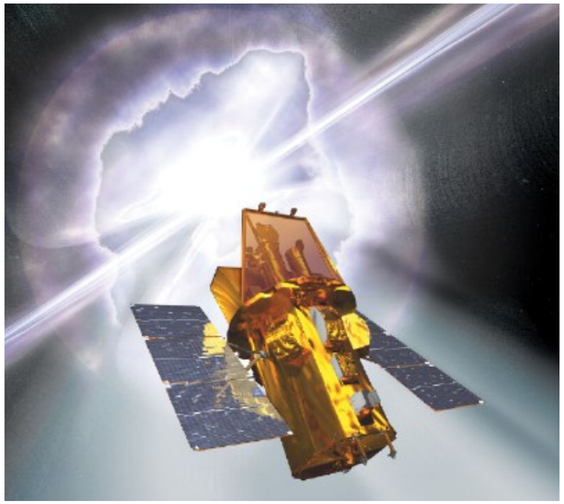
- Predict activation patterns for nouns
- Google's trillion word corpus used to measure co-occurrence

Monitoring transients in astronomy [Djorgovski]

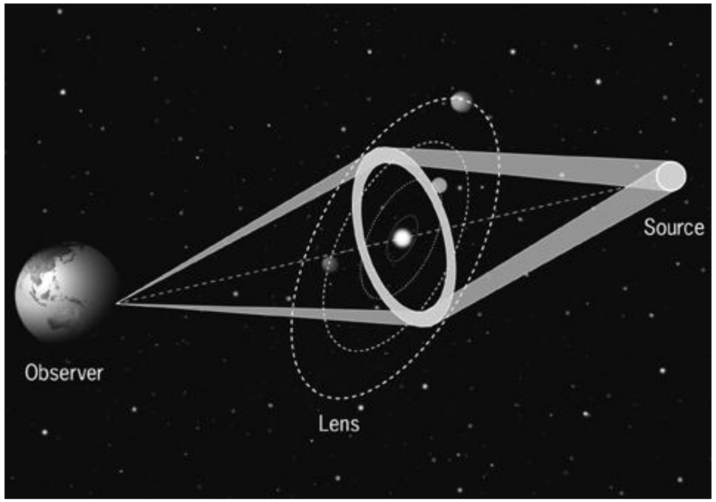
Novae, Cataclysmic Variables



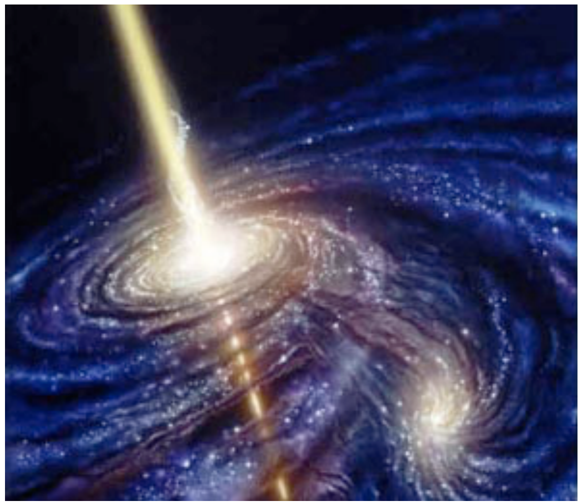
Supernovae



Gamma-Ray Bursts



Gravitational Microlensing



Accretion to SMBHs

Data-rich astronomy [Djorgovski]

- Typical digital sky survey now generates ~ 10 - 100 TB, plus a comparable amount of derived data products
  - PB-scale data sets are on the horizon
- Astronomy today has ~ 1 - 2 PB of archived data, and generates a few TB/day
  - Both data volumes and data rates grow exponentially, with a doubling time ~ 1.5 years
  - Even more important is the growth of *data complexity*
- For comparison:

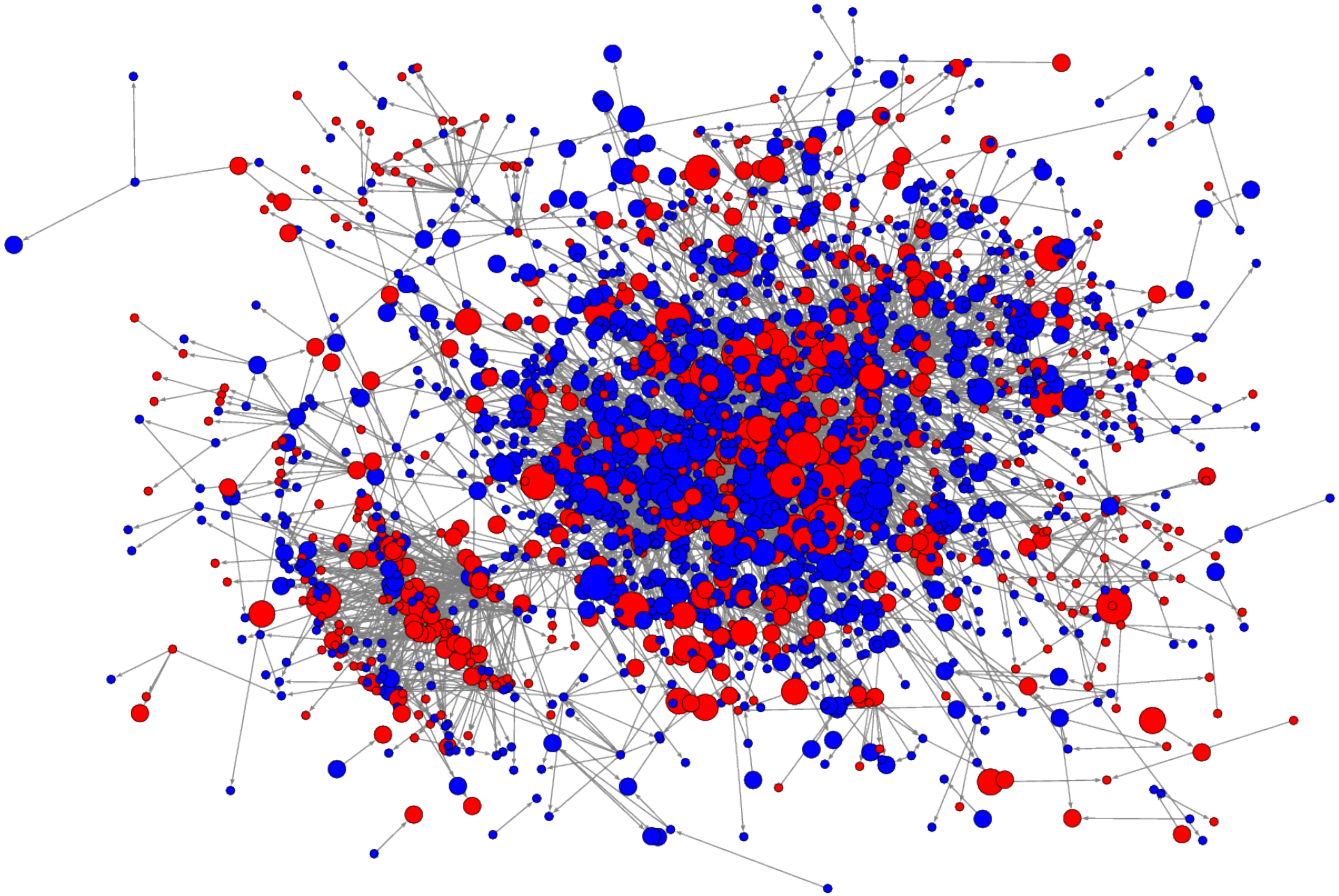
  Human memory ~ a few hundred MB

  Human Genome < 1 GB

  1 TB ~ 2 million books

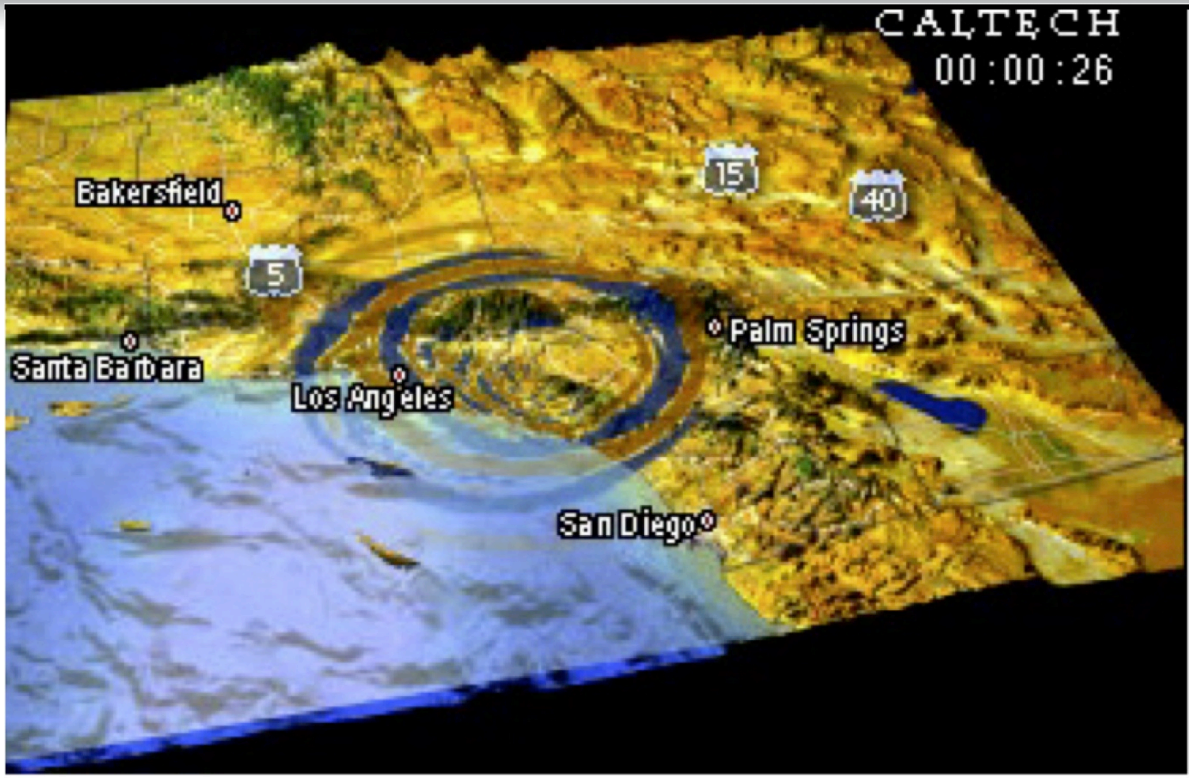
  Library of Congress (print only) ~ 30 TB

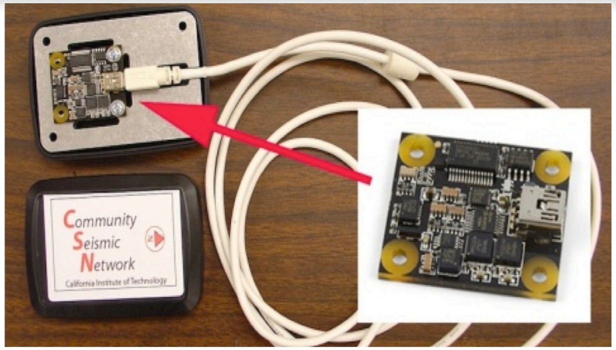
Computational Social Science



Community Seismic Network

[with Chandy, Clayton, Heaton, Kohler, Faulkner, Olson et al.]





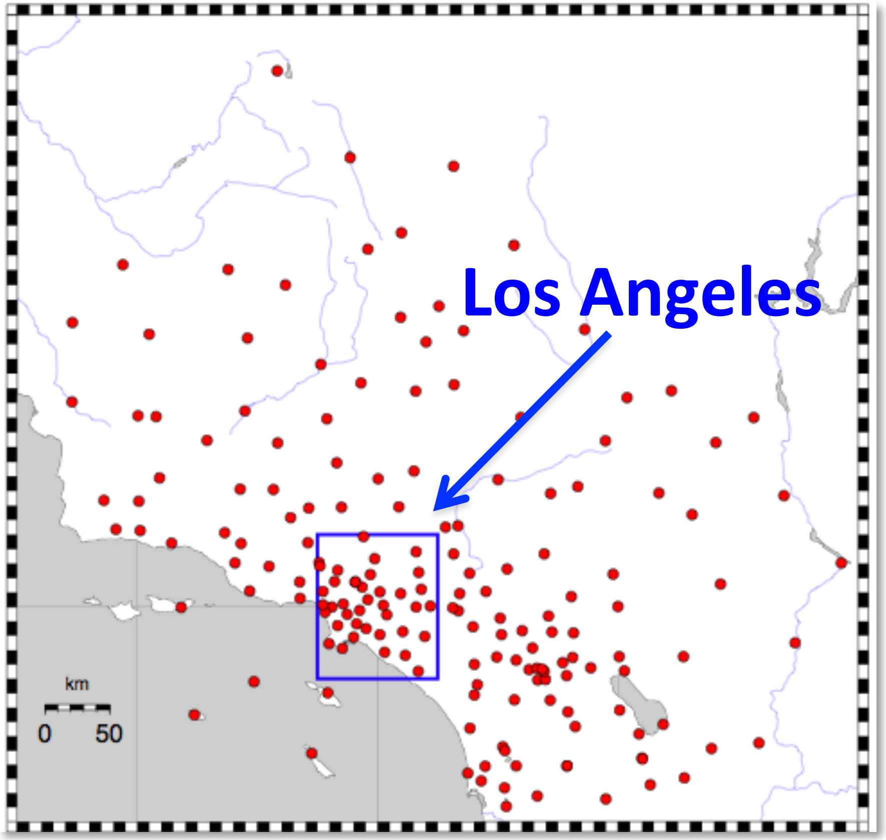
Detect and monitor earthquakes using cheap accelerometers in cell phones and other consumer devices

[See also Quake-Catcher (Cochran et al.), NetQuakes (USGS)]

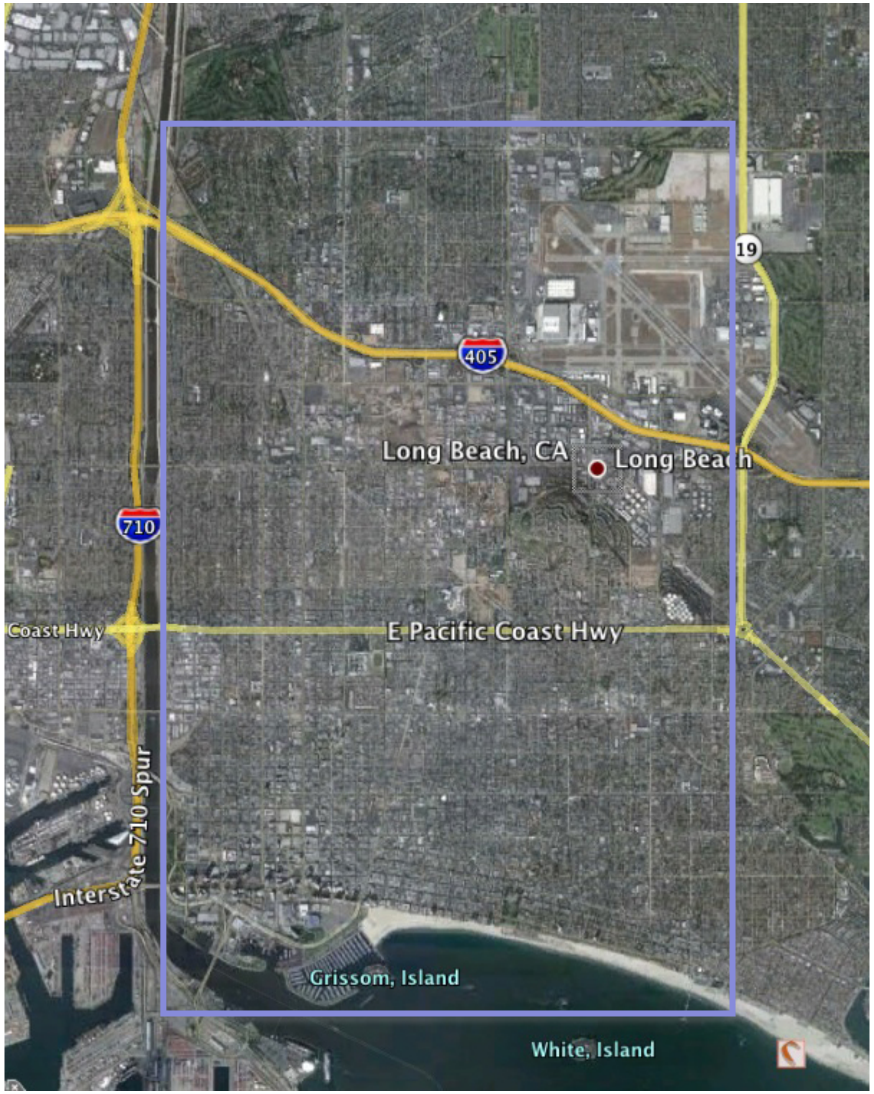
Traditional Seismic Networks

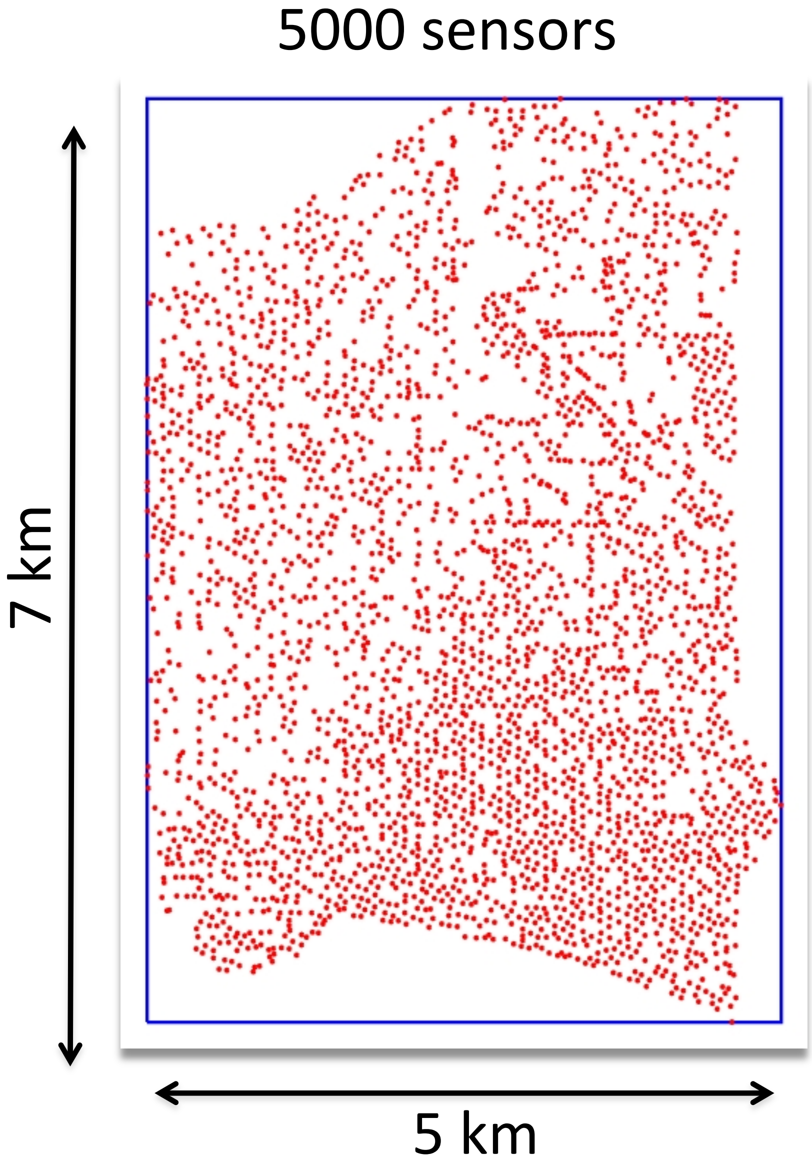
Few sensors. Highly accurate.

Installations are expensive ($10,000) but low noise



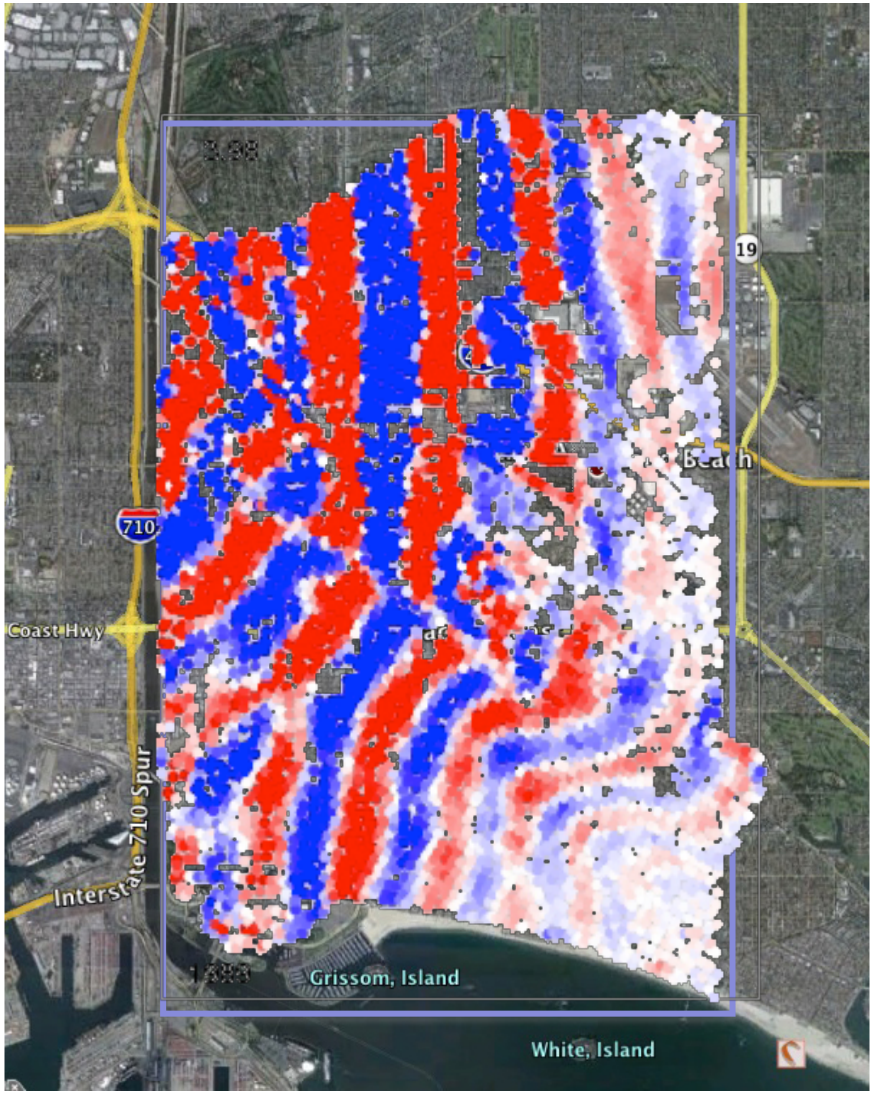
Benefit from higher density

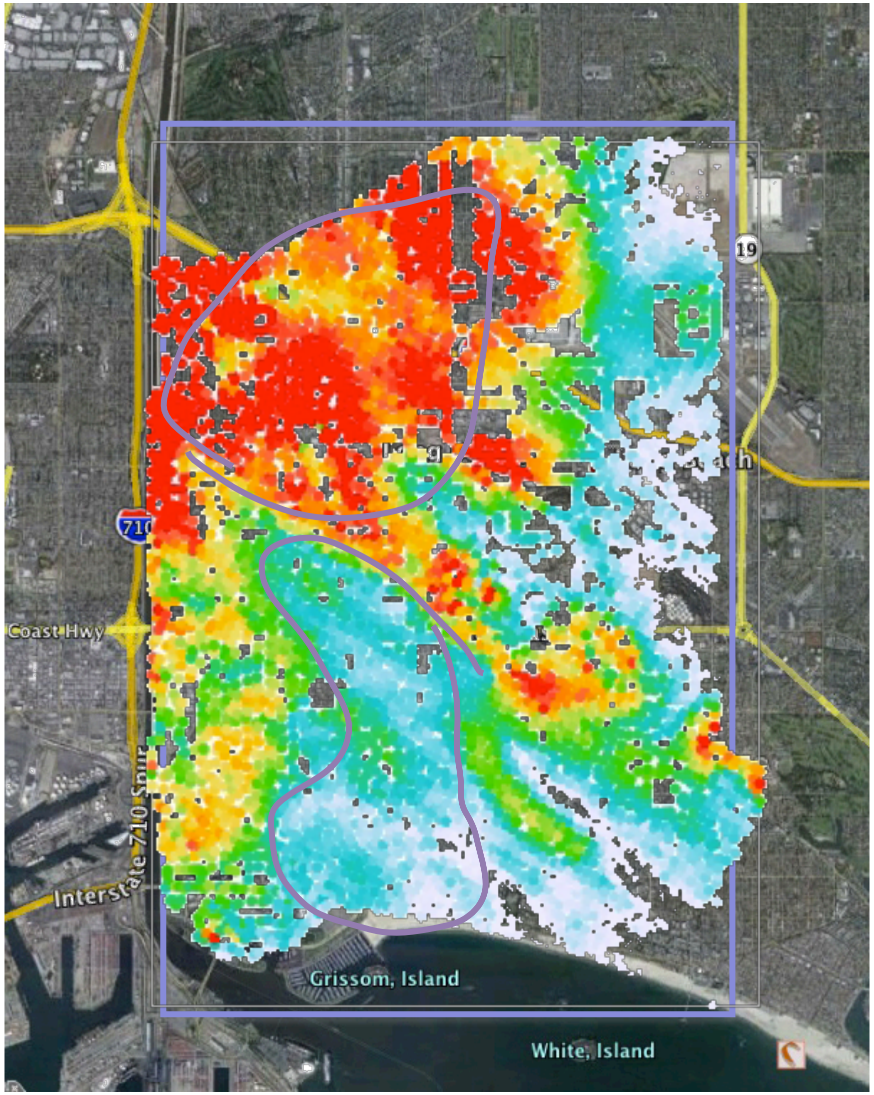




[Nodal Seismic Inc.]

Benefit from higher density



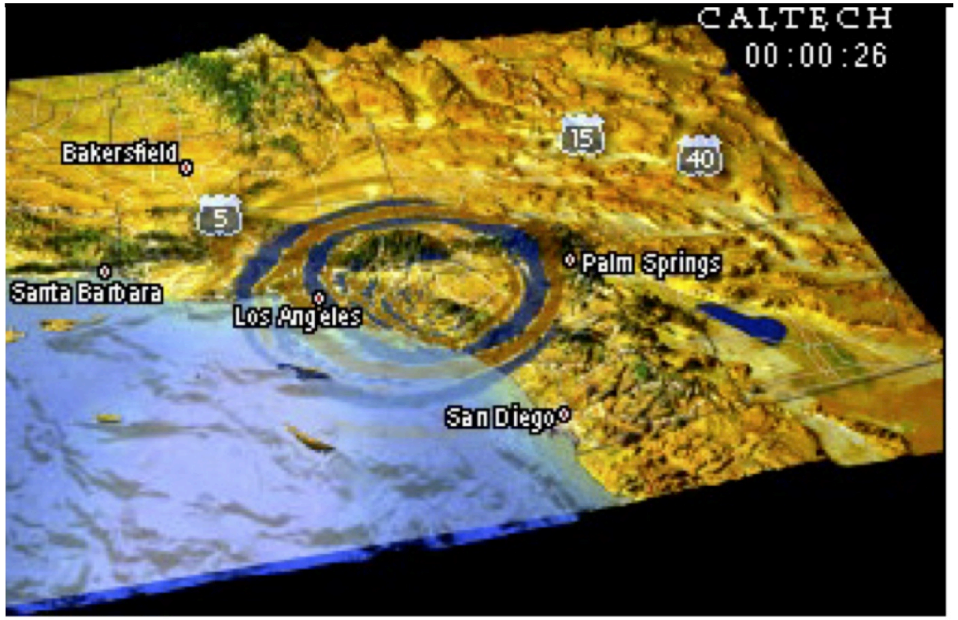


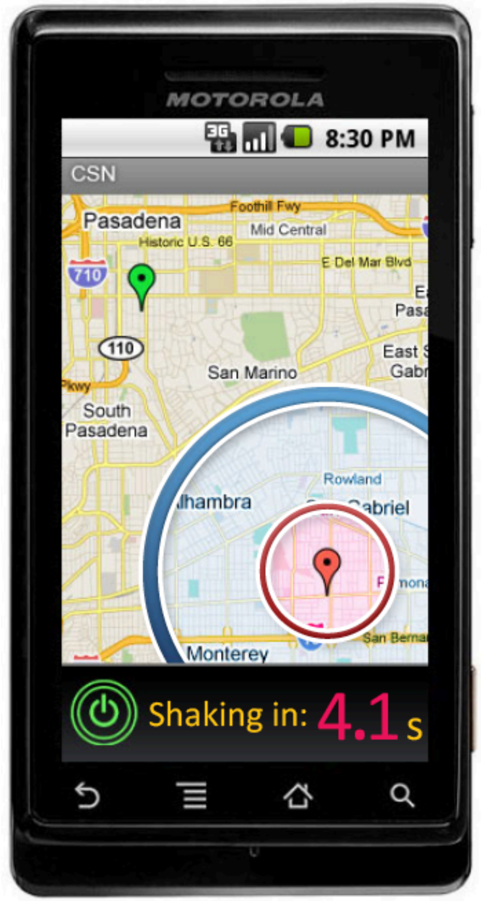
Wavefront

Carson Earthquake 2011/05/14 M=2.5

Peak Amplitude

Early Warning: Decision making under massive uncertainty

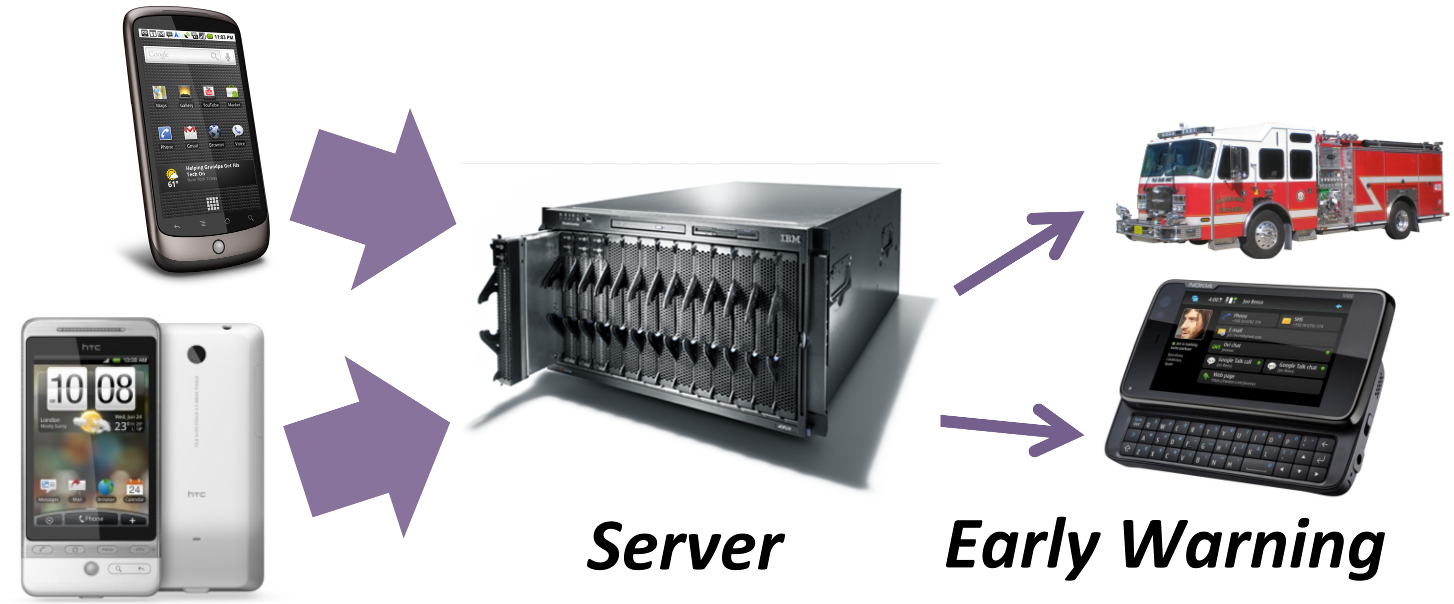




- Opportunities for early warning:
  - Stop trains, elevators, …
  - Shut valves, stabilize grid, …
- False alarms can have high cost
- Missed detections can cost lives…

Naïve approach

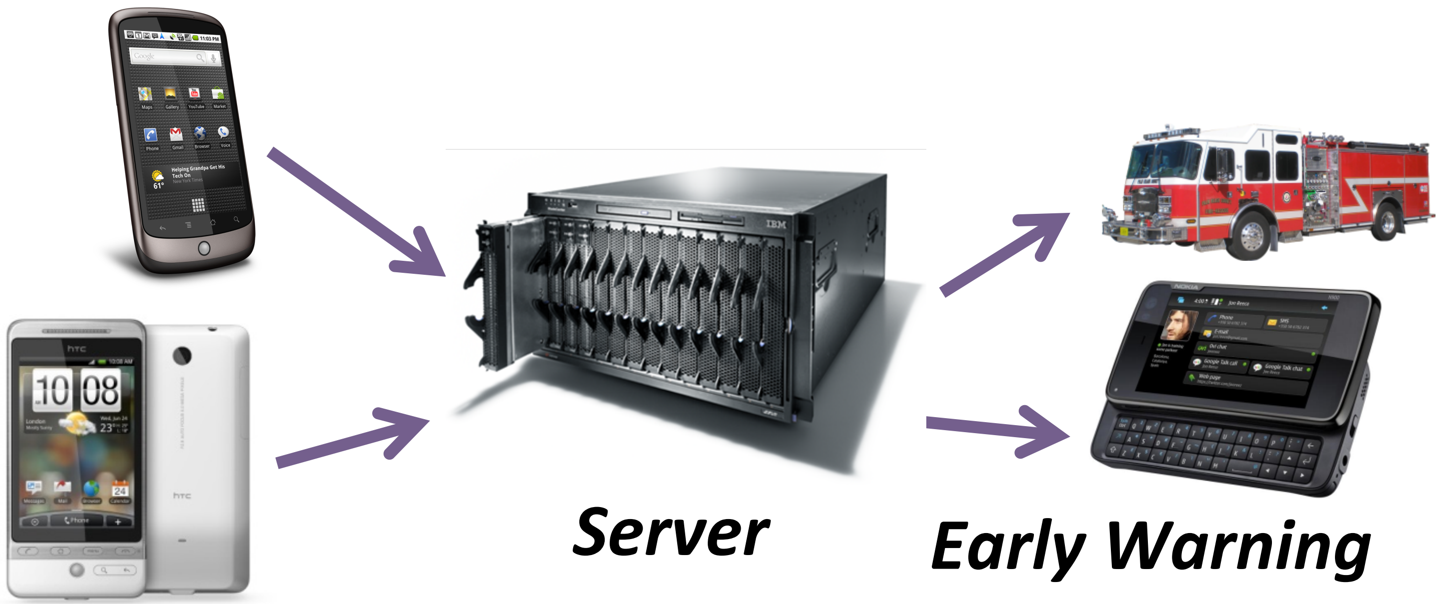
- Sensors send all data to a server
- Server analyzes data, decides whether to raise an alarm



- 1 million phones → 30 TB data/day!!
- “Drinking from the fire hose”

How do we do it?

- Sensors analyze the data *locally* on the phones
- Communicate only if they experience *unusual* motion



- Local decisions affect global decision!
- Need to *learn* to send *most useful* information

Community sensing

Contribute sensor data

Sensing:

traffic jams,

cascading failures,

…

Decision making:

Regulate traffic,

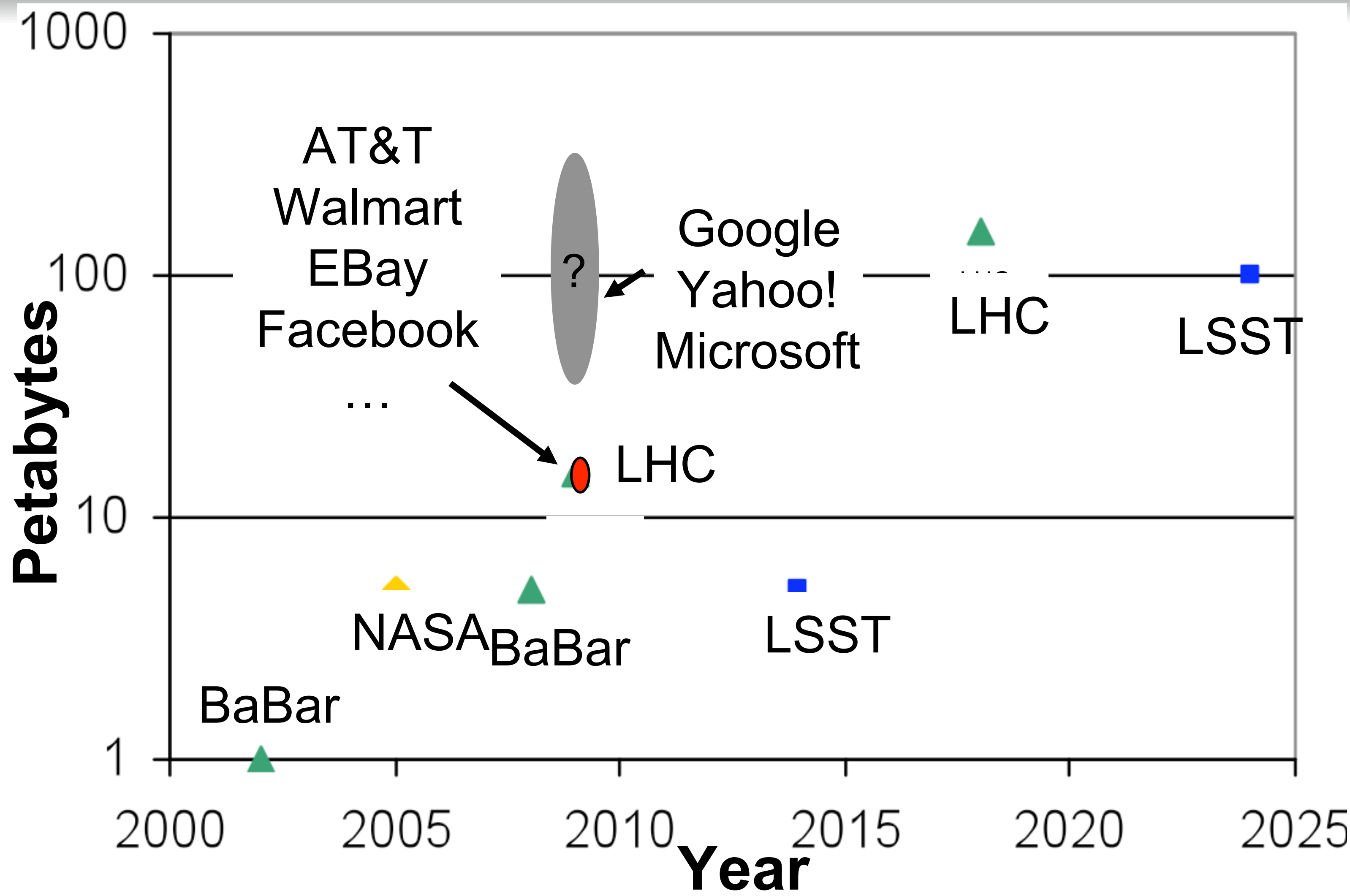
power grid,

…

Learning from massive data

- Many applications require gaining insights from massive, noisy data sets
- Science
  - Physics (LHC, …), Astronomy (sky surveys, …), Neuroscience (fMRI, micro-electrode arrays, …), Biology (proteomics, …), Geology (sensor arrays, …), …
  - Social science, economics, …
- Commercial / civil / engineering applications
  - Consumer data (online advertising, viral marketing, …)
  - Health records (evidence based medicine, …)
  - Traffic monitoring / earthquake detection …
- Security / defense related applications
  - Spam filtering / intrusion detection / surveillance, …

Data volume in scientific and industrial applications



[Meiron et al]

How can we **extract useful information** from **massive, noisy data sets?**

What is data mining?

Semi-automatic procedures to find patterns that are

Useful: help making better decisions (make money...)

General: hold on unseen data with some probability

The Search for ESP

- In the 1950s, a parapsychologist hypothesized that some people had Extra-Sensory Perception (ESP)
- In an experiment, subjects where asked to guess 10 hidden cards – red or blue
- He discovered that almost 1 in 1000 got all ten right, thus he concluded they had ESP

The Search for ESP cont'd

- He called the people with ESP for another test
- This time, almost all had lost their ESP
- His conclusion:

Don't tell people they have ESP or they'll lose it! ☺

Data Mining Goals

- **Approximate retrieval**
  - Given a query, find "most similar" item in a large data set
  - *Applications*: GoogleGoggles, Shazam, …
- **Supervised learning** (Classification, Regression)
  - Learn a concept (function mapping queries to labels)
  - *Applications*: Spam filtering, predicting price changes, …
- **Unsupervised learning** (Clustering, dimension reduction)
  - Identify clusters, "common patterns"; anomaly detection
  - *Applications*: Recommender systems, fraud detection, …
- **Interactive data mining**
  - Learning through experimentation / from limited feedback
  - *Applications*: Online advertising, opt. UI, learning rankings, …

Challenges for Data Mining

Main memory vs. disk access

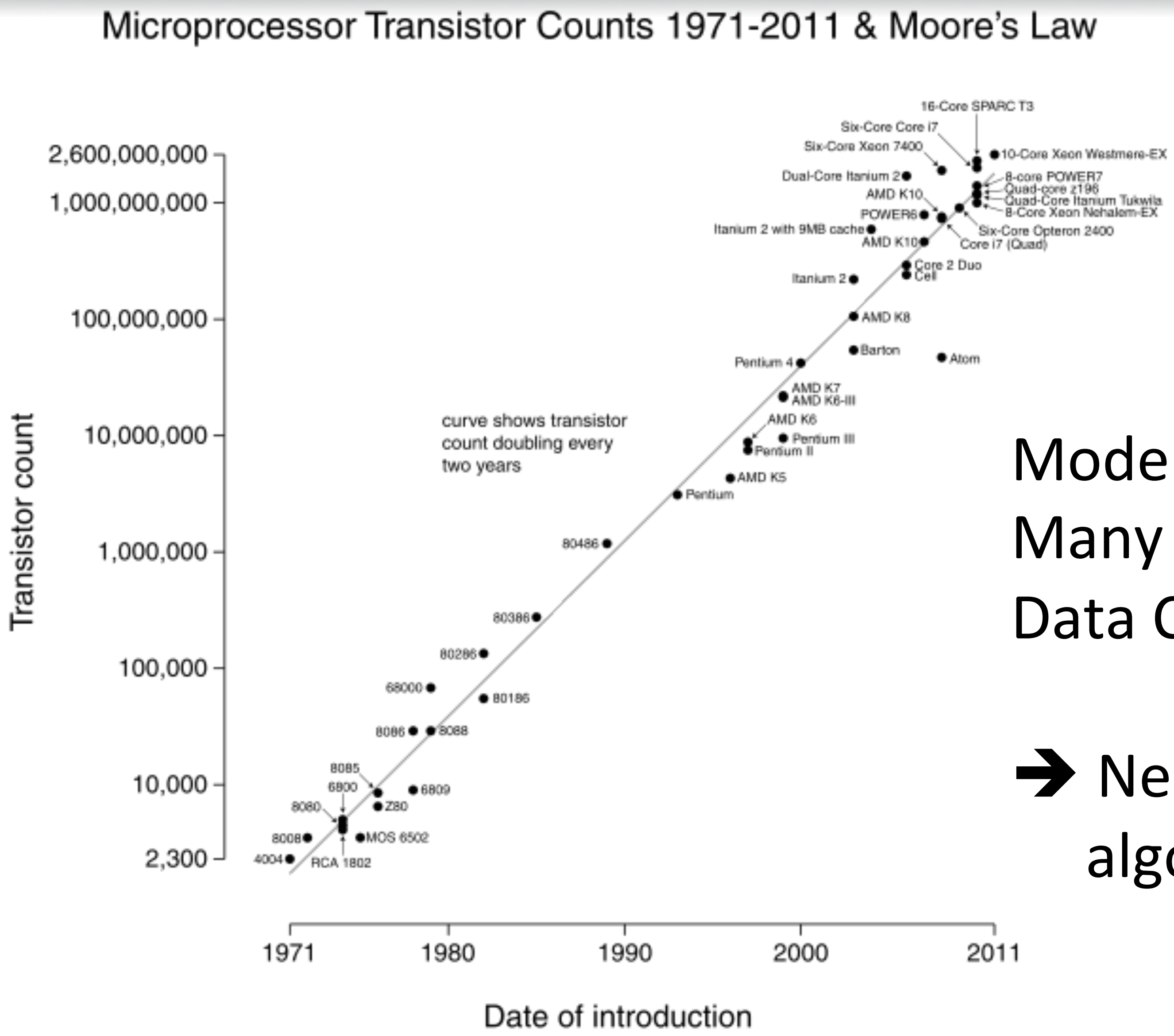
Main memory:
Fast, random access, expensive

Secondary memory (hard disk)
~10^4 slower, sequential access, inexpensive

Massive data → Sequential access

How can we learn from streaming data?

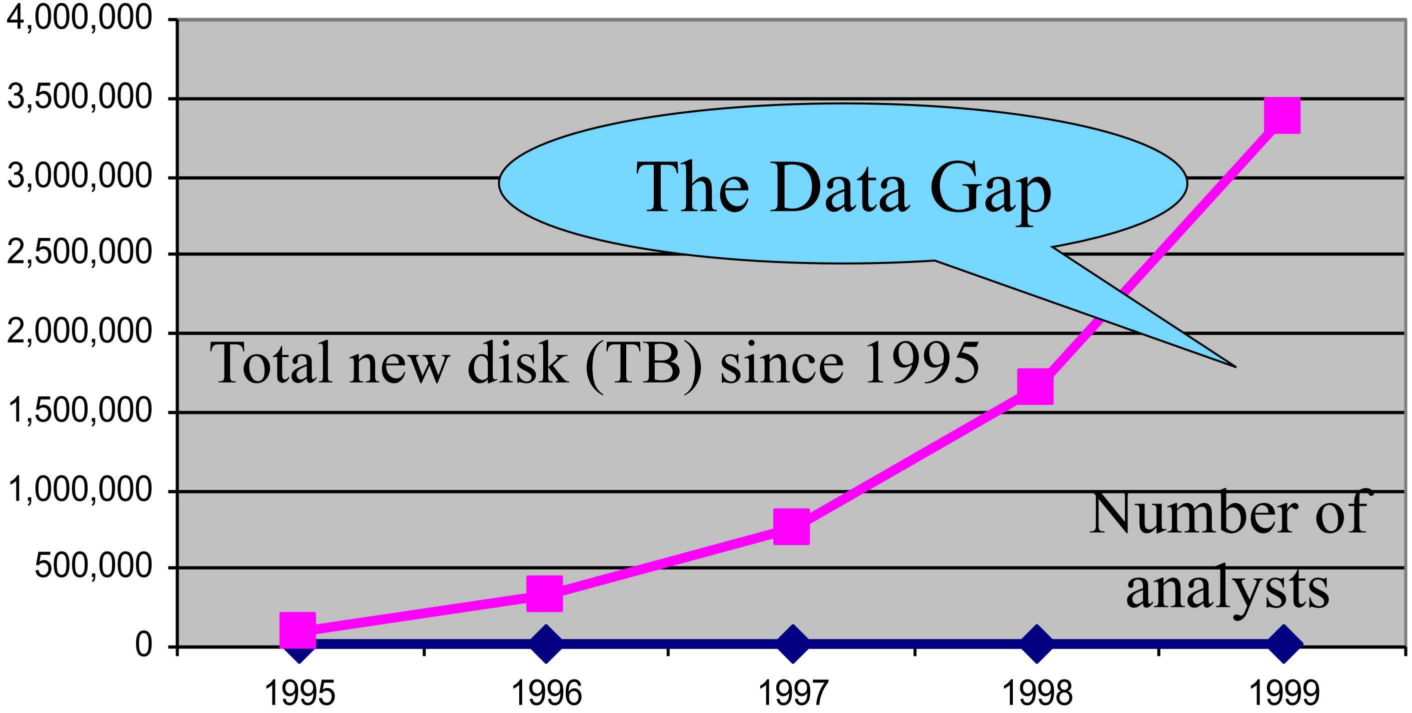
Moore's Law



Modern architectures:
Many Cores
Data Centers

➔ Need distributed algorithms

The Data Gap



[R. Grossman et al. “Data Mining for Scientific and Engineering Applications”]

Data Mining Challenges

- Can't fit data set in main memory
  - Access from disk much slower
  - Can't afford "random access" to the data
  - Often can't store data as quickly as it is arriving
- Need for parallelism
  - Data centers as the new means of cost effective computing
  - "Cloud computing"
- Humans don't scale
  - Need to deal with human attention as a scarce resource

→ Need specialized models and algorithms to cope with these challenges

→ **This is the focus of this class**

Other challenges

- Data quality
- Data ownership and distribution
- Privacy
- Security
- …

Overview

- Advanced graduate course
- Four main topics
  - Approximate retrieval
  - Supervised learning
  - Unsupervised learning
  - Interactive data mining

  all in the context of very large data sets
- Both theory and applications
- Handouts etc. on course webpage
  - http://las.ethz.ch/courses/datamining-s12/
- Textbook:
  - http://infolab.stanford.edu/~ullman/mmds/book.pdf

Overview

- *Instructors*:
  Andreas Krause (krausea@ethz.ch)

- *Teaching assistants*:
  Yuxin Chen (yuxin.chen@inf.ethz.ch)
  Hasta Vanchinathan (hastagiri@inf.ethz.ch)
  Adish Singla (singlaa@inf.ethz.ch)

- *Administrative assistant*:
  Rita Klute (rita.klute@inf.ethz.ch)

Background & Prequisites

- **Required**: Solid basic knowledge in statistics, algorithms and programming.
- Background in machine learning is helpful but **not** required.

- We review necessary background, but will move quickly…

Coursework

- Grade based on written **session exam**
- **Approx. six homeworks** (not graded)
  - Mix of theory and programming assignments (Python recommended)
- Two parallel **recitations**
  - Discussion of homework solutions
  - Opportunities to ask questions
  - Watch course webpage for updates (rooms, group assignment)
- **Next week no class, but recitations**

What we will cover

- Fundamental tools from optimization, algorithms and statistics for dealing with large data
- “What makes Google, Facebook, Amazon et al. tick”
- Topics include (syllabus on webpage)
  - Fast nearest neighbor methods (shingling, LSH)
  - Online learning / no regret optimization
  - Fast training of SVM classifiers
  - Bandit algorithms with applications online advertising
  - Active Learning
  - Sketching / Coresets
  - Recommender Systems

What we will *not* cover

- Systems issues (e.g., databases; architecture and management of data centers; …)
  - See specialized courses
  - We focus on models and algorithms
- Data structures (KD-trees / R-trees, etc.)
  - See specialized courses
- Domain specific algorithms, heuristics
  - We focus on fundamental principles

Today:

Modern computing infrastructure
for data mining

Algorithmic primitives for using
this infrastructure

Infrastructure for modern data mining

- Data Centers
  - Commodity hardware
  - Many machines connected in a network
- Challenges
  - How to distribute computation?
  - Machines fail regularly
- MapReduce is designed to handle these challenges

lbl.gov

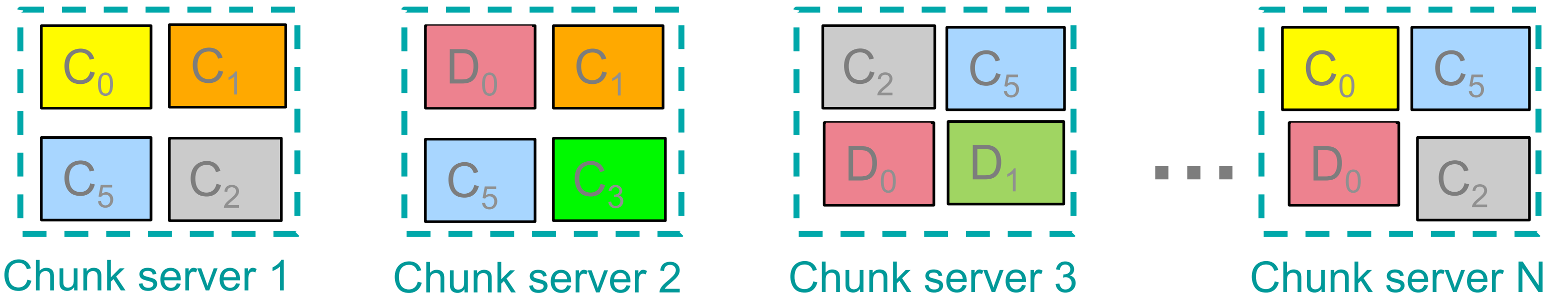
MapReduce

- Idea:
  - Store data redundantly for reliability
  - Bring computation close to the data
  - Provide unified programming model to simplify parallelism

- Builds on Distributed File Systems

Distributed File Systems

- Provides global namespace
- *Examples*: Google GFS, Hadoop HDFs, Kosmix KFS
- Optimized for the common use case:
  - Huge files (hundreds of GB to TB)
  - Infrequent updates
  - Frequent reads and appends



Example: Counting words

- **Given**: Large file with one word per line
- **Goal**: Count the number of times each word appears

- Applications:
  - Analyze logs to find popular queries, bots, …

How would you do it?

- **Case 1**:
  - Entire file fits in memory
- **Case 2**:
  - File too large for memory, but all <word, count> pairs fit in memory
- **Data Mining Case**:
  - File on multiple disks, too many distinct words to fit in memory

????

Map-Reduce: Overview

- Read a lot of data
- **Map:**
  - Extract something you care about
- Shuffle and Sort
- **Reduce:**
  - Aggregate, summarize, filter or transform
- Write the result

Keep general outline;
adapt **map** and **reduce** to fit the problem

More specifically

- Program specifies two primary methods:
  - Map(k,v) → <k', v'>*
  - Reduce(k', <v'>*) → <k', v''>*
- All v' with same k' are reduced together and processed in v' order

Map-Reduce: Word counting

Provided by the programmer

MAP:
reads input and produces a set of key value pairs

Group by key:
Collect all pairs with same key

Provided by the programmer

Reduce:
Collect all values belonging to the key and output

The crew of the space shuttle Endeavor recently returned to Earth as ambassadors, harbingers of a new era of space exploration. Scientists at NASA are saying that the recent assembly of the Dextre bot is the first step in a long-term space-based man/machine partnership. '"The work we're doing now -- the robotics we're doing -- is what we're going to need to do to build any work station or habitat structure on the moon or Mars," said Allard Beutel.

Big document

(the, 1)
(crew, 1)
(of, 1)
(the, 1)
(space, 1)
(shuttle, 1)
(endeavor, 1)
(recently, 1)
....

(key, value)

(crew, 1)
(crew, 1)
(space, 1)
(the, 1)
(the, 1)
(the, 1)
(shuttle, 1)
(recently, 1)
…

(key, value)

(crew, 2)
(space, 1)
(the, 3)
(shuttle, 1)
(recently, 1)
…

(key, value)

Only sequential reads

Word Count using MapReduce

```
map(key, value):
// key: document name; value: text of document
  for each word w in value:
      emit(w, 1)


reduce(key, values):
// key: a word; value: an iterator over counts
      result = 0
      for each count v in values:
            result += v
      emit(key, result)
```

Example: Language modeling

- Statistical machine translation:
  - Need to count number of times every 5-word sequence occurs in a large corpus of documents

- How to implement in MapReduce:
  - Map: extract (5-word sequence, count) from document
  - Reduce: combine counts

Example: Distributed Grep

- Find all occurrences of the given pattern in a very large set of files

- **Map**:
  - Apply grep on assigned documents
  - Emit list of documents that contain term
- **Reduce**:
  - Merge lists

Example: Calculating statistics

- **Input**: Data set D with one number x_i per line i
- **Output**:

$$\mu(D) = \frac{1}{n} \sum_i x_i$$

$Var(X) = \mathbb{E}(X^2) - \mathbb{E}(X)^2$

- **Map**:
  - Compute n_i and $\mu(D_i)$ for each chunk D_i
- **Reduce**:

$$\mu(D) = \frac{\sum_i n_i \mu(D_i)}{\sum_i n_i}$$

Example: Shakemaps

- Want to figure out how strongly different regions are shaken through earthquakes
- **Input**
  - Each line: epicenter location; magnitude
- **Map**
  - Reads a line of input and simulate the earthquake
  - Output: (region ID, earthquake id, amount of shaking)
- **Reduce**
  - Collect the region IDs and compute average (or maximum etc.) amount of shaking

Map-Reduce: Environment

- Map-Reduce environment takes care of
  - Partitioning the input data
  - Scheduling the program's execution across a set of machines
  - Handling machine failures
  - Managing required inter-machine communication

➔ The programmer doesn't need to deal with this!

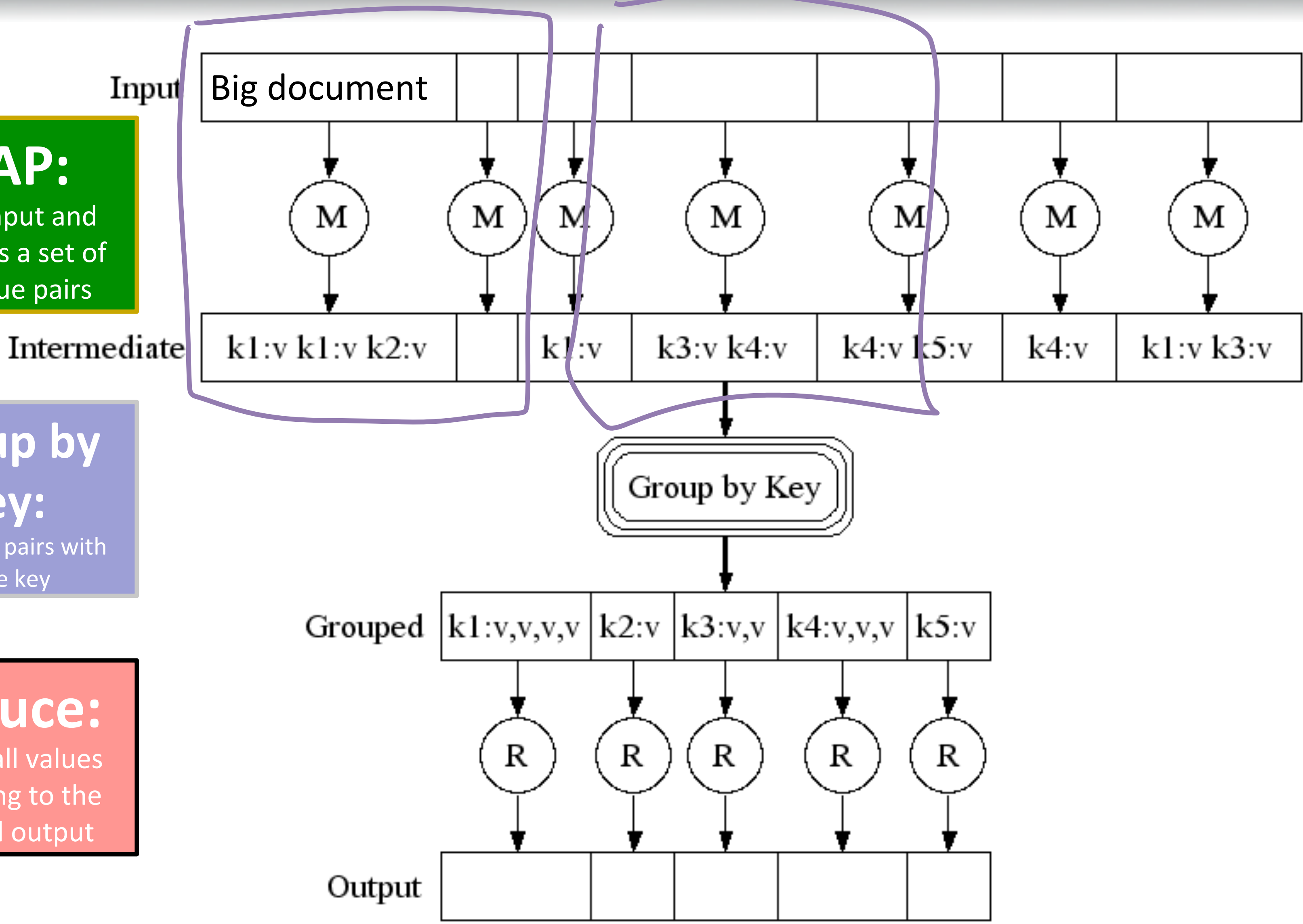
➔ Drastically simplifies writing massively parallel code!

Map-Reduce: A diagram

MAP:
reads input and produces a set of key value pairs

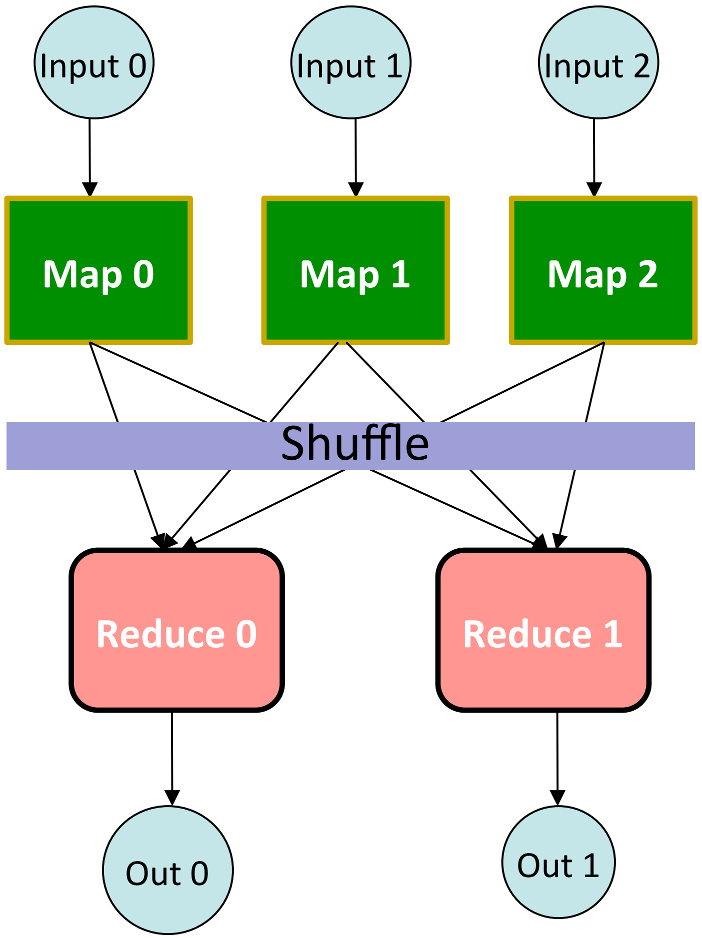
Group by key:
Collect all pairs with same key

Reduce:
Collect all values belonging to the key and output

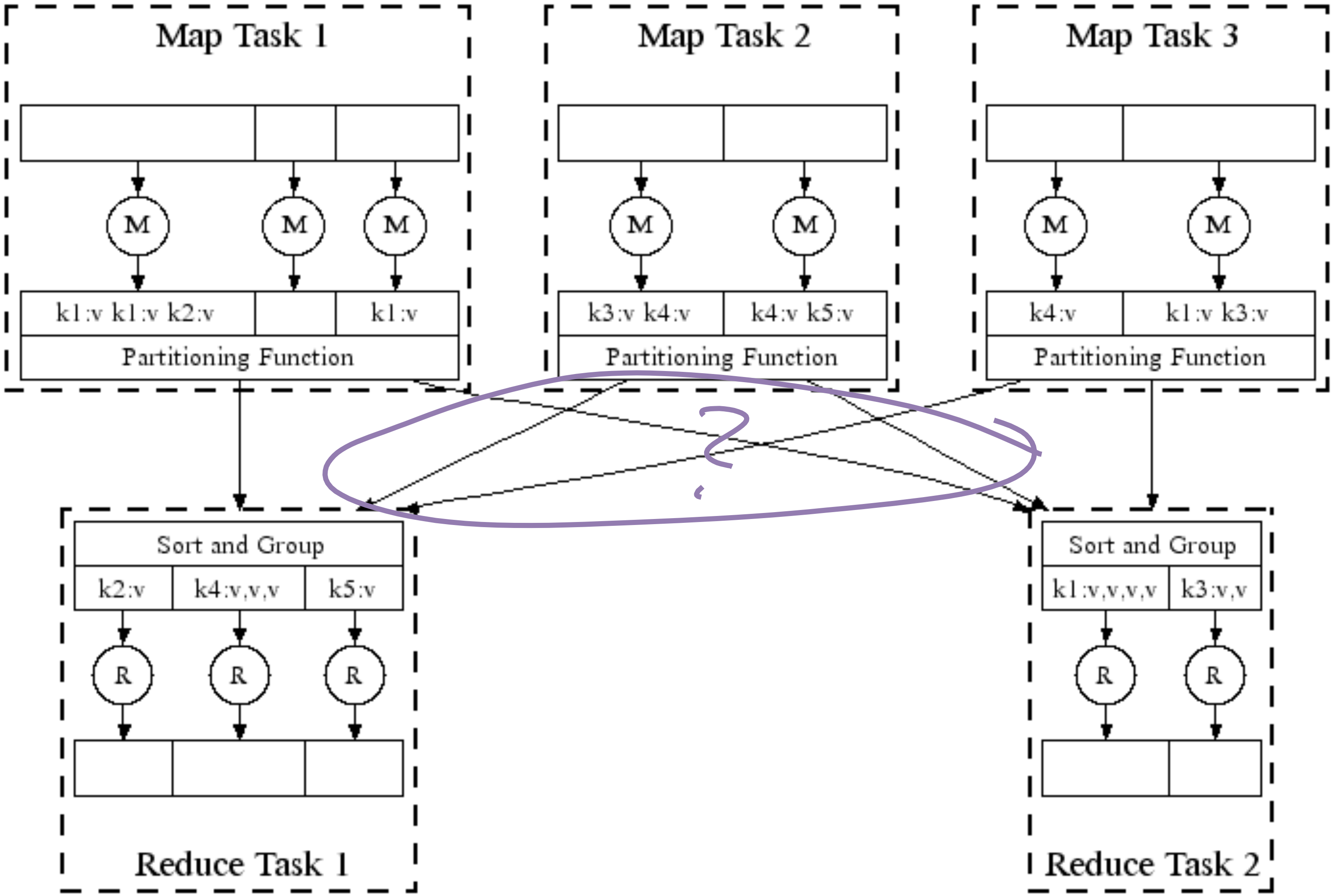


Map-Reduce

- Programmer specifies:
  - **Map** and **Reduce** and input files
- MapReduce environment does
  - Read inputs as a set of key-value-pairs
  - **Map** transforms input <k,v>-pairs into a new set of <k',v'>-pairs
  - Sort & Shuffle the <k',v'>-pairs to output nodes
  - All <k',v'>-pairs with a given k' are sent to the same **reduce**
  - **Reduce** processes all <k',v'>-pairs grouped by key into new <k'',v''>-pairs
  - Write the resulting pairs to files
- All phases are distributed with many tasks doing the work



Parallel Map-Reduce



Data flow

- Input and final output are stored on a distributed file system:
  - Scheduler tries to schedule map tasks "close" to physical storage location of input data

- Intermediate results are stored on local FS of map and reduce workers

- Output is often input to another map reduce task
  - → Application composed from multiple MR stages
  - → Will see examples later in the course

Coordination

- Master data structures:
  - *Task status*: (idle, in-progress, completed)
  - Idle tasks get scheduled as workers become available
  - When a map task completes, it sends the master the location and sizes of its R intermediate files, one for each reducer
  - Master notifies reducers

- Master pings workers periodically to detect failures

Failures

- Map worker failure
  - Map tasks completed or in-progress at worker are reset to idle
  - Reduce workers are notified when task is rescheduled on another worker
- Reduce worker failure
  - Only in-progress tasks are reset to idle
- Master failure
  - MapReduce task is aborted and client is notified

How many Map and Reduce jobs?

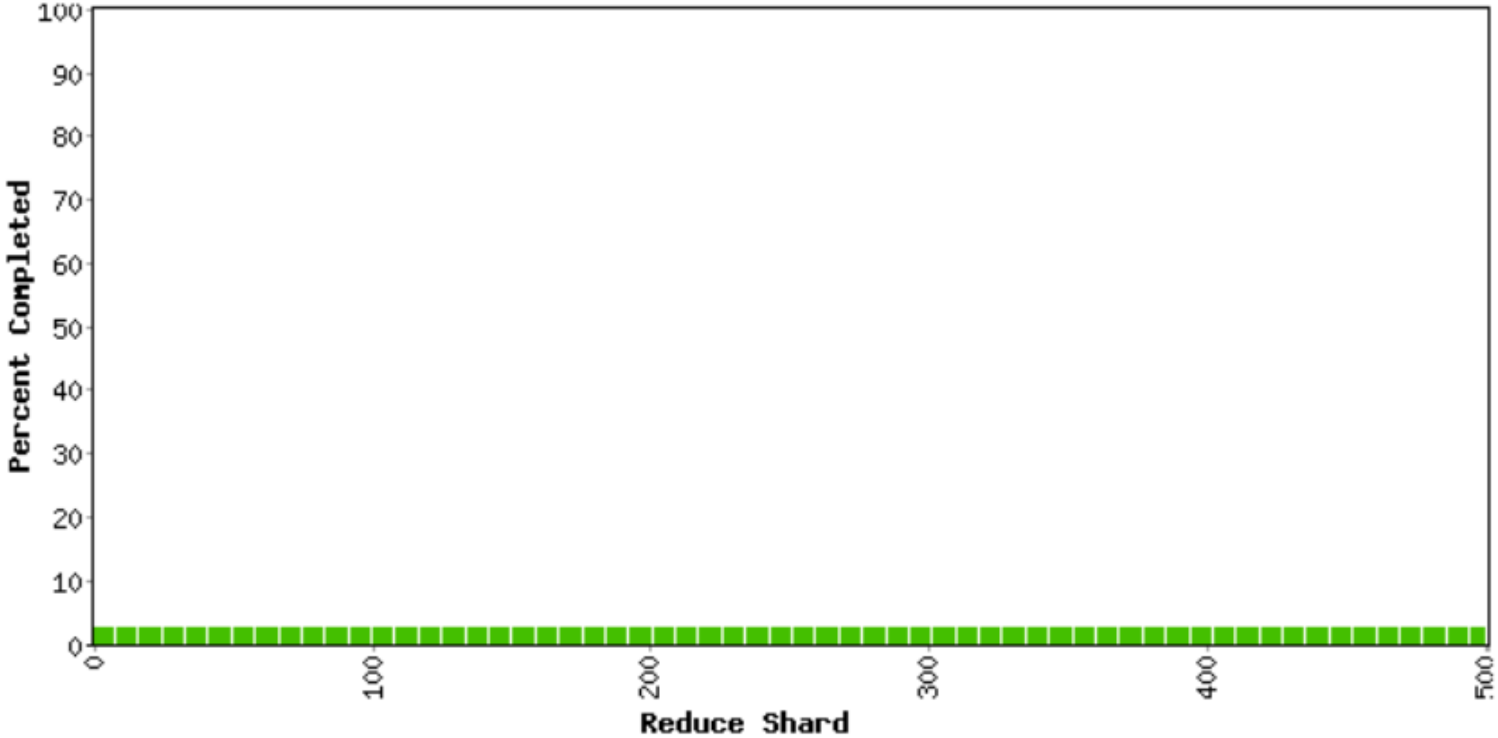
- M map tasks, R reduce tasks
- Rule of thumb:
  - M and R >> number of nodes in cluster
  - One DFS chunk per map is common
  - Improves dynamic load balancing and speeds recovery from worker failure
- Usually R is smaller than M
  - output is spread across R files; want to deal with small number of outputs

MapReduce status: MR_Indexer-beta6-large-2003_10_28_00_03

Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 00 min 18 sec

323 workers; 0 deaths

Type	Shards	Done	Active	Input(MB)	Done(MB)	Output(MB)
Map	13853	0	323	878934.6	1314.4	717.0
Shuffle	500	0	323	717.0	0.0	0.0
Reduce	500	0	0	0.0	0.0	0.0



Counters

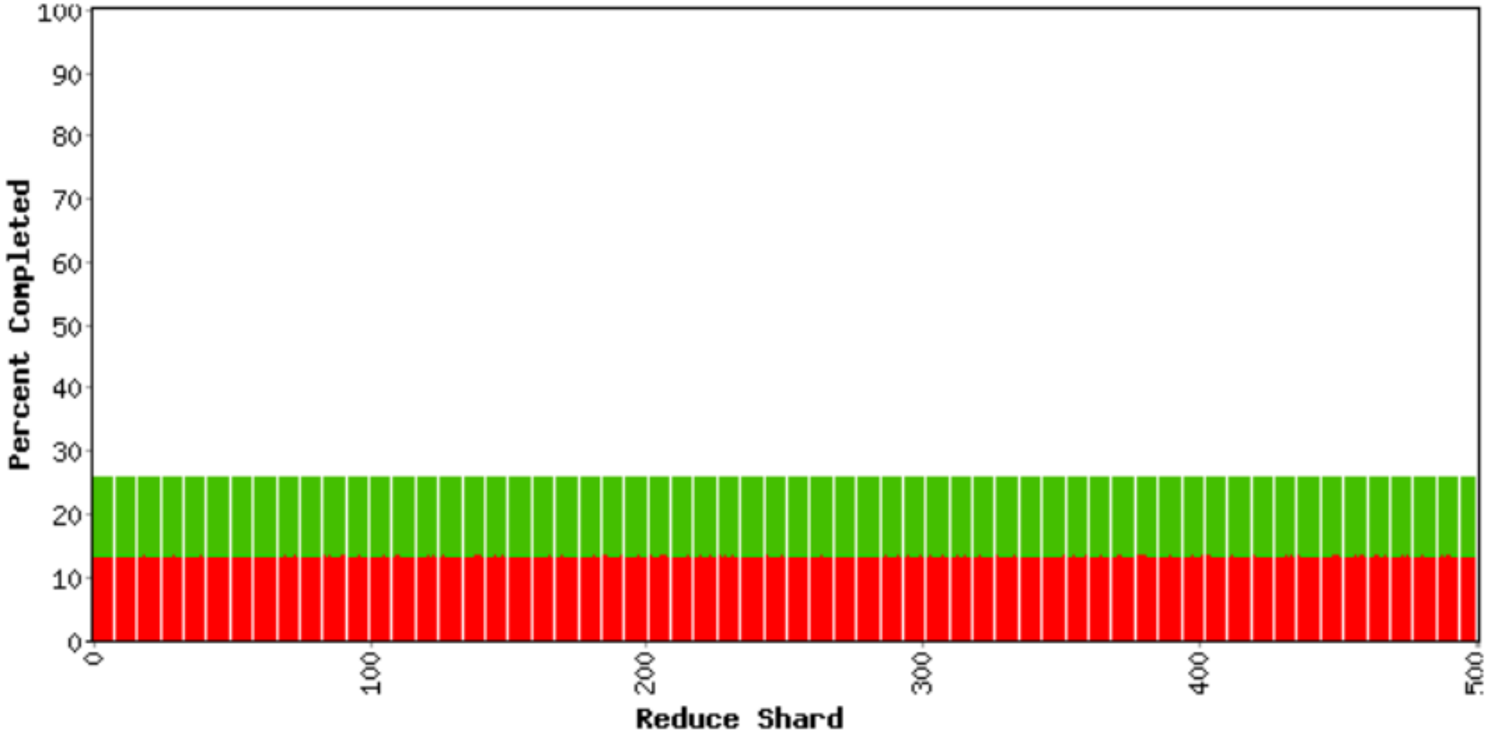
Variable	Minute
Mapped (MB/s)	72.5
Shuffle (MB/s)	0.0
Output (MB/s)	0.0
doc-index-hits	145825686
docs-indexed	506631
dups-in-index-merge	0
mr-operator-calls	508192
mr-operator-outputs	506631

MapReduce status: MR_Indexer-beta6-large-2003_10_28_00_03

Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 05 min 07 sec

1707 workers; 1 deaths

Type	Shards	Done	Active	Input(MB)	Done(MB)	Output(MB)
Map	13853	1857	1707	878934.6	191995.8	113936.6
Shuffle	500	0	500	113936.6	57113.7	57113.7
Reduce	500	0	0	57113.7	0.0	0.0



Counters

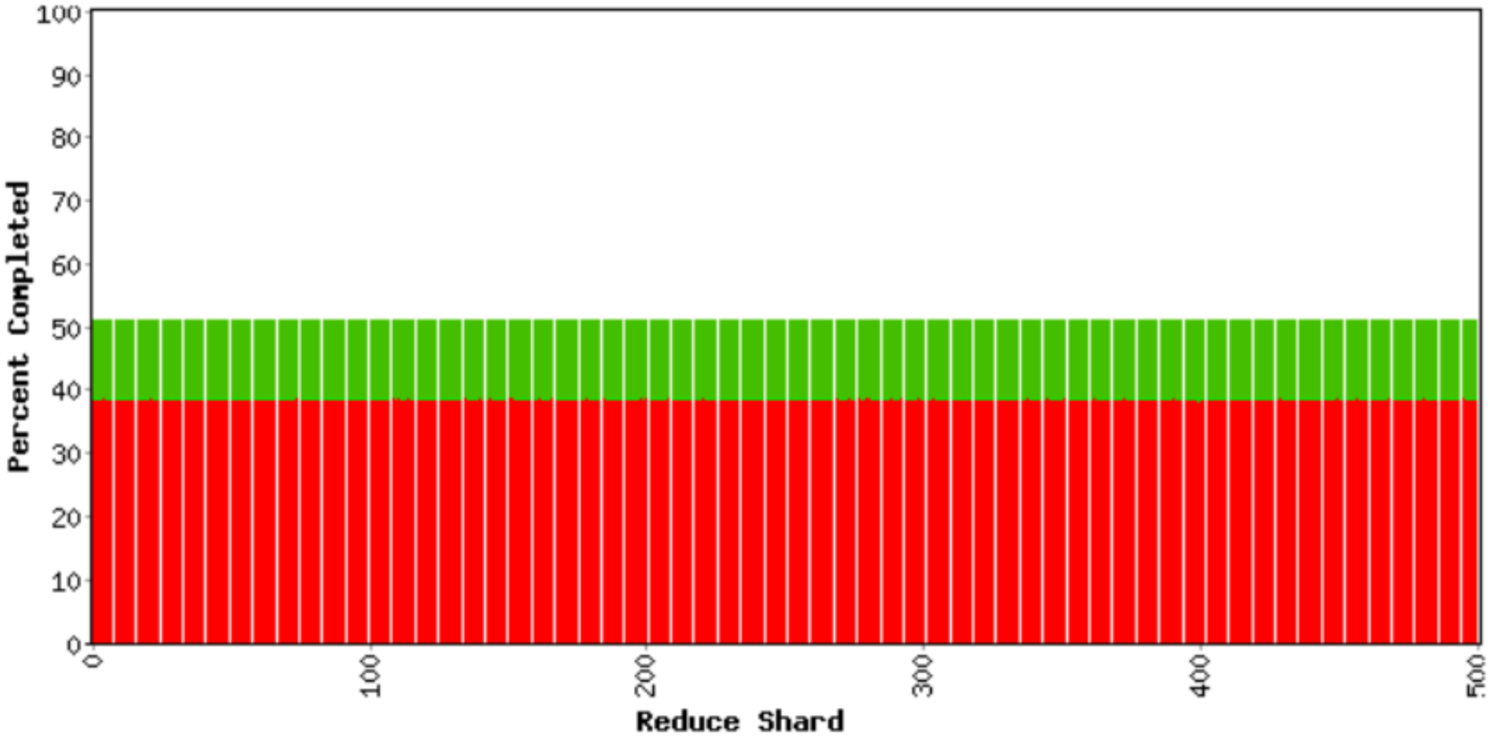
Variable	Minute
Mapped (MB/s)	699.1
Shuffle (MB/s)	349.5
Output (MB/s)	0.0
doc-index-hits	5004411944
docs-indexed	17290135
dups-in-index-merge	0
mr-operator-calls	17331371
mr-operator-outputs	17290135

MapReduce status: MR_Indexer-beta6-large-2003_10_28_00_03

Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 10 min 18 sec

1707 workers; 1 deaths

Type	Shards	Done	Active	Input(MB)	Done(MB)	Output(MB)
Map	13853	5354	1707	878934.6	406020.1	241058.2
Shuffle	500	0	500	241058.2	196362.5	196362.5
Reduce	500	0	0	196362.5	0.0	0.0



Counters

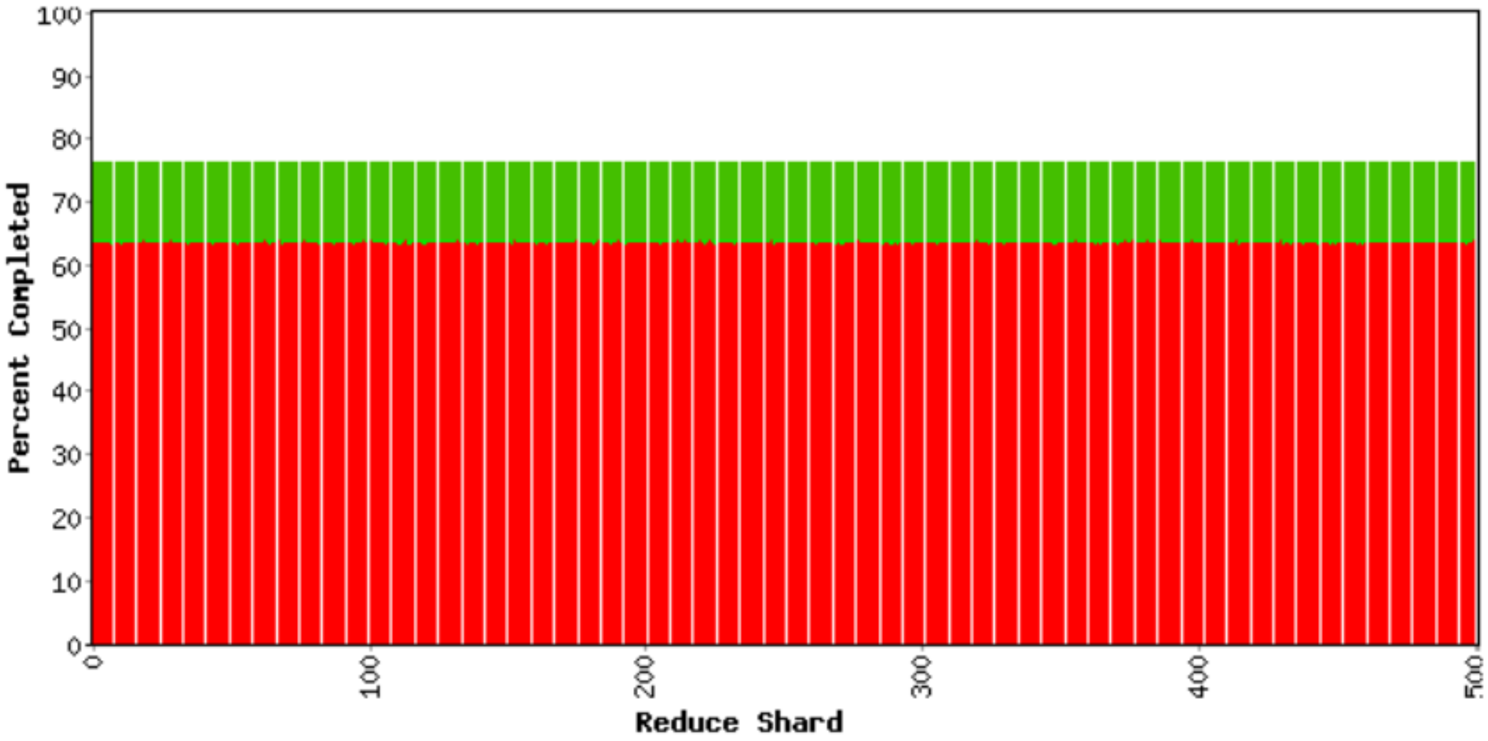
Variable	Minute
Mapped (MB/s)	704.4
Shuffle (MB/s)	371.9
Output (MB/s)	0.0
doc-index-hits	5000364228
docs-indexed	17300709
dups-in-index-merge	0
mr-operator-calls	17342493
mr-operator-outputs	17300709

MapReduce status: MR_Indexer-beta6-large-2003_10_28_00_03

Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 15 min 31 sec

1707 workers; 1 deaths

Type	Shards	Done	Active	Input(MB)	Done(MB)	Output(MB)
Map	13853	8841	1707	878934.6	621608.5	369459.8
Shuffle	500	0	500	369459.8	326986.8	326986.8
Reduce	500	0	0	326986.8	0.0	0.0



Counters

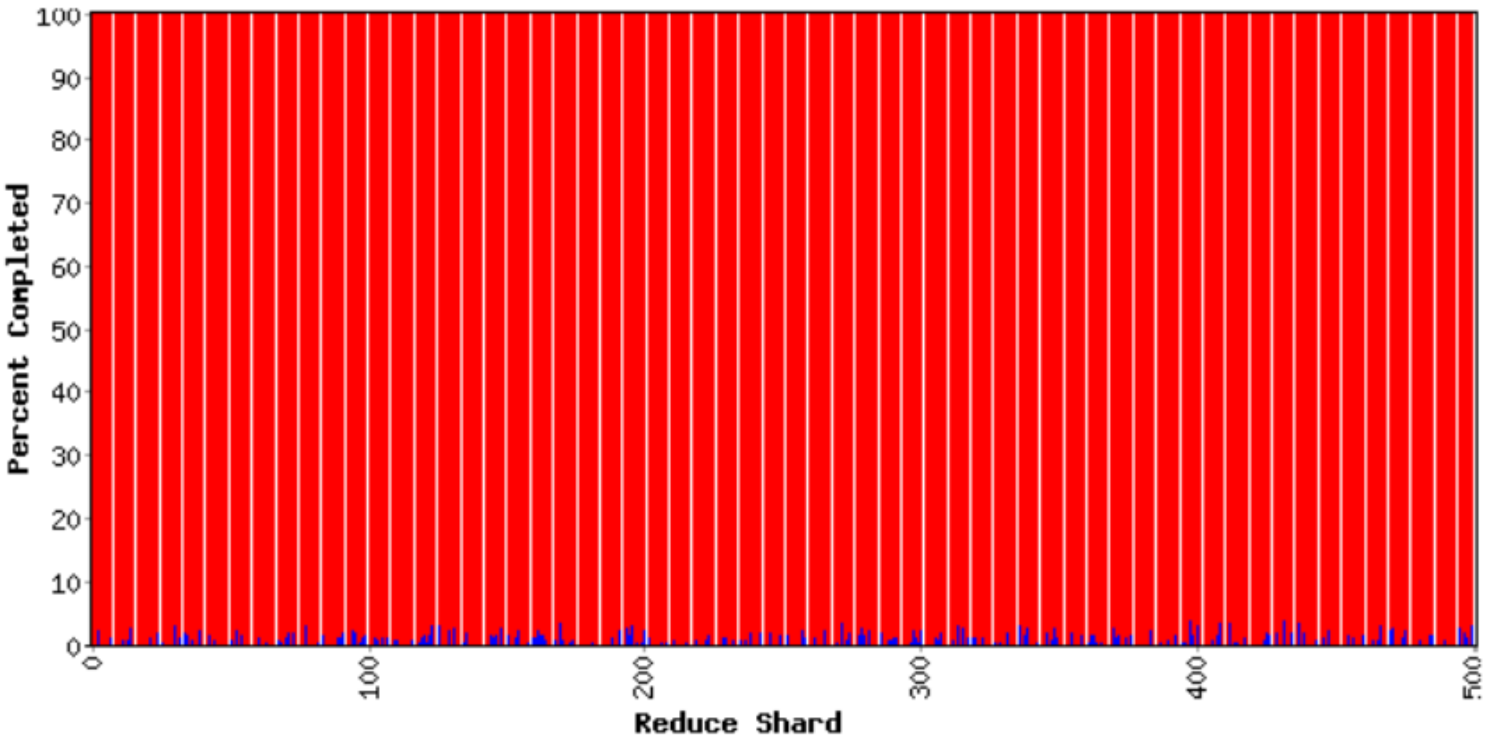
Variable	Minute
Mapped (MB/s)	706.5
Shuffle (MB/s)	419.2
Output (MB/s)	0.0
doc-index-hits	4982870667
docs-indexed	17229926
dups-in-index-merge	0
mr-operator-calls	17272056
mr-operator-outputs	17229926

MapReduce status: MR_Indexer-beta6-large-2003_10_28_00_03

Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 29 min 45 sec

1707 workers; 1 deaths

Type	Shards	Done	Active	Input(MB)	Done(MB)	Output(MB)
Map	13853	13853	0	878934.6	878934.6	523499.2
Shuffle	500	195	305	523499.2	523389.6	523389.6
Reduce	500	0	195	523389.6	2685.2	2742.6



Counters

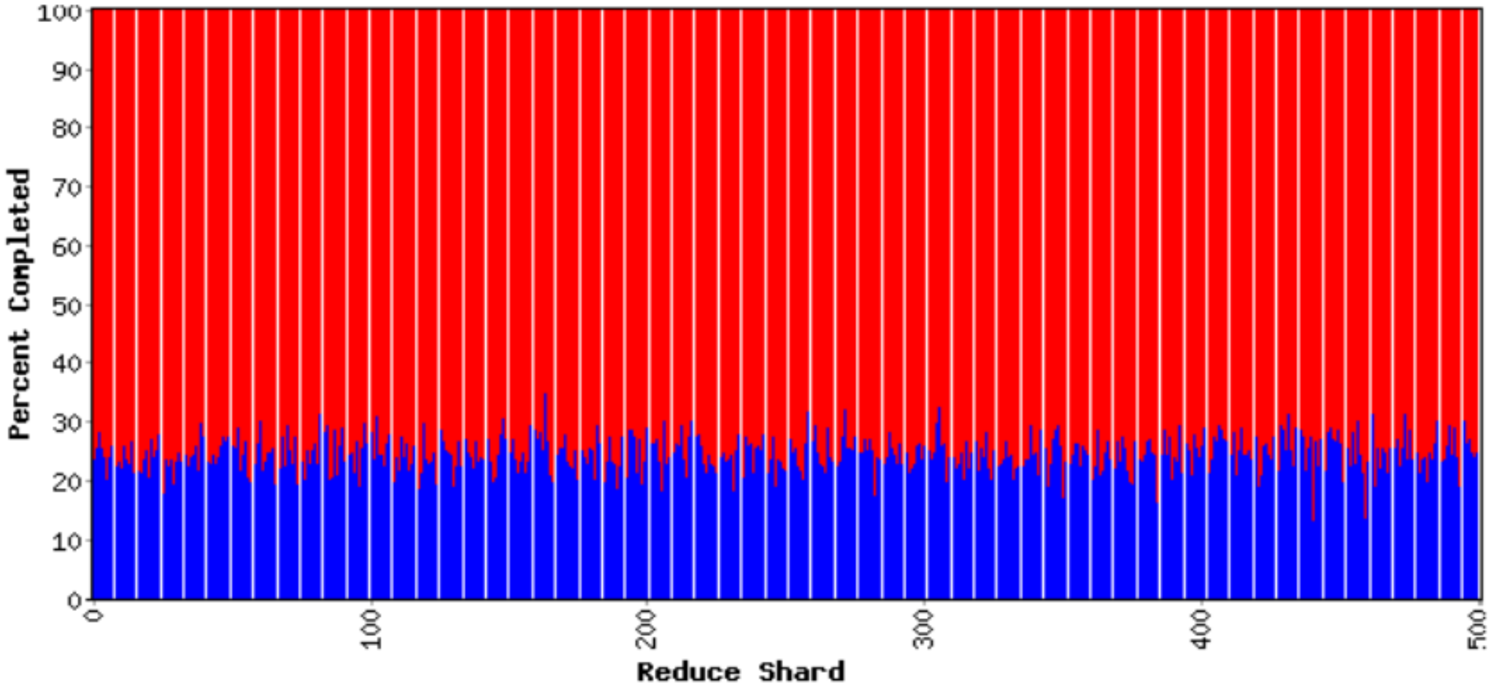
Variable	Minute	
Mapped (MB/s)	0.3	
Shuffle (MB/s)	0.5	
Output (MB/s)	45.7	
doc-index-hits	2313178	10
docs-indexed	7936	
dups-in-index-merge	0	
mr-merge-calls	1954105	
mr-merge-outputs	1954105	

MapReduce status: MR_Indexer-beta6-large-2003_10_28_00_03

Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 31 min 34 sec

1707 workers; 1 deaths

Type	Shards	Done	Active	Input(MB)	Done(MB)	Output(MB)
Map	13853	13853	0	878934.6	878934.6	523499.2
Shuffle	500	500	0	523499.2	523499.5	523499.5
Reduce	500	0	500	523499.5	133837.8	136929.6



Counters

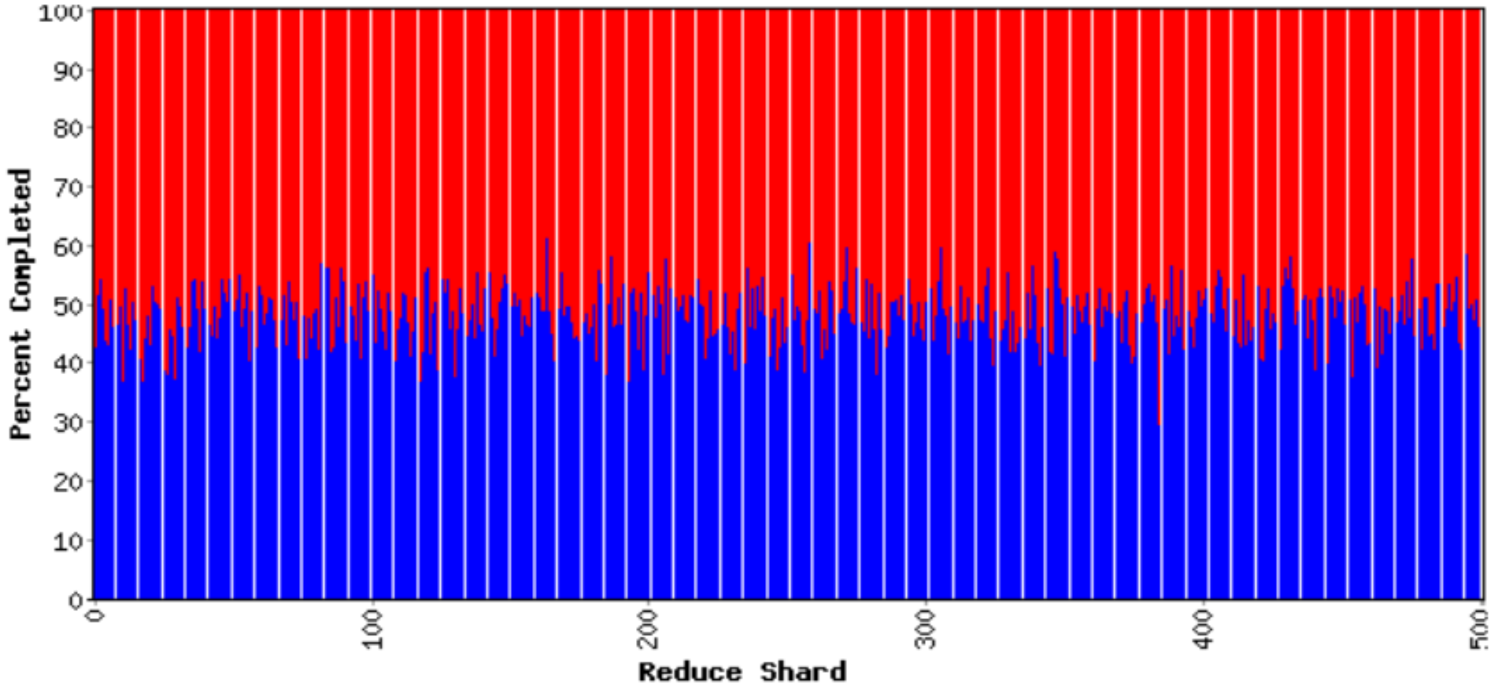
Variable	Minute
Mapped (MB/s)	0.0
Shuffle (MB/s)	0.1
Output (MB/s)	1238.8
doc-index-hits	0
docs-indexed	0
dups-in-index-merge	0
mr-merge-calls	51738599
mr-merge-outputs	51738599

MapReduce status: MR_Indexer-beta6-large-2003_10_28_00_03

Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 33 min 22 sec

1707 workers; 1 deaths

Type	Shards	Done	Active	Input(MB)	Done(MB)	Output(MB)
Map	13853	13853	0	878934.6	878934.6	523499.2
Shuffle	500	500	0	523499.2	523499.5	523499.5
Reduce	500	0	500	523499.5	263283.3	269351.2



Counters

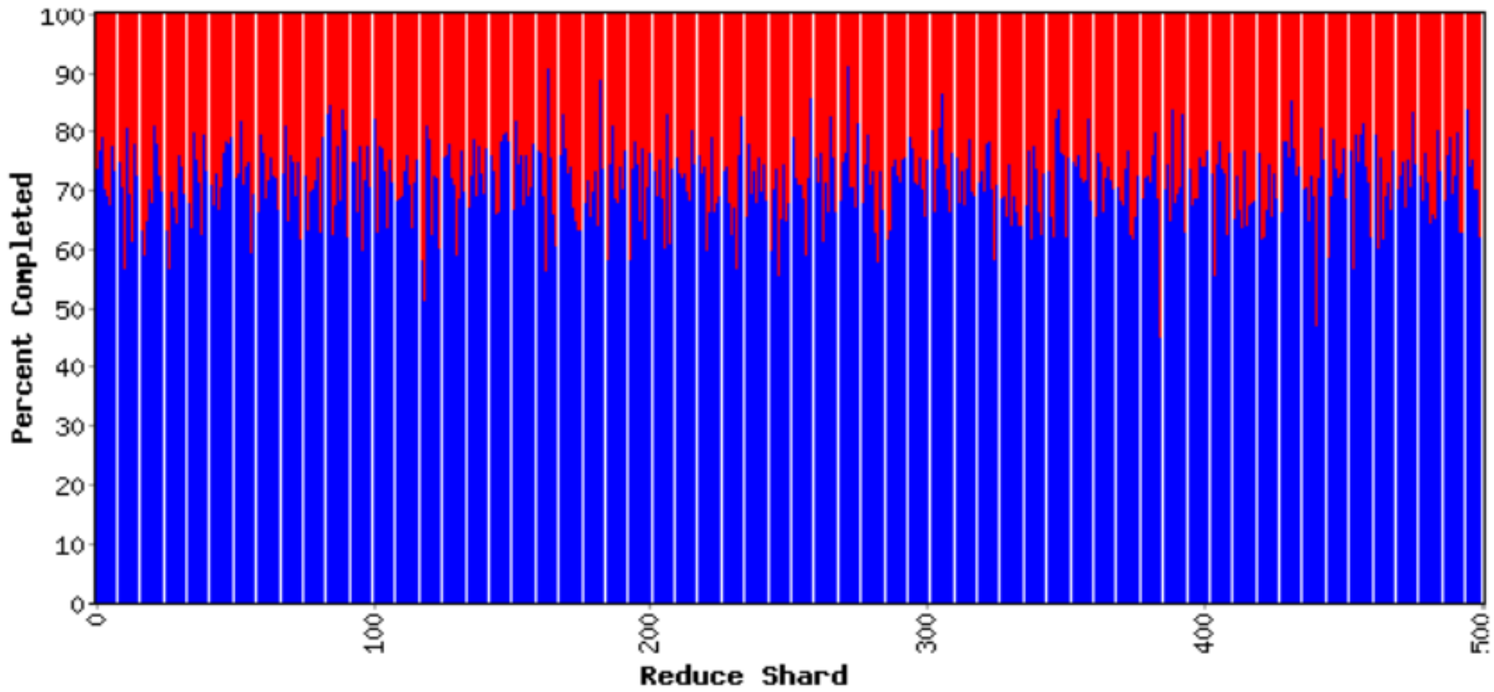
Variable	Minute
Mapped (MB/s)	0.0
Shuffle (MB/s)	0.0
Output (MB/s)	1225.1
doc-index-hits	0
docs-indexed	0
dups-in-index-merge	0
mr-merge-calls	51842100
mr-merge-outputs	51842100

MapReduce status: MR_Indexer-beta6-large-2003_10_28_00_03

Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 35 min 08 sec

1707 workers; 1 deaths

Type	Shards	Done	Active	Input(MB)	Done(MB)	Output(MB)
Map	13853	13853	0	878934.6	878934.6	523499.2
Shuffle	500	500	0	523499.2	523499.5	523499.5
Reduce	500	0	500	523499.5	390447.6	399457.2



Counters

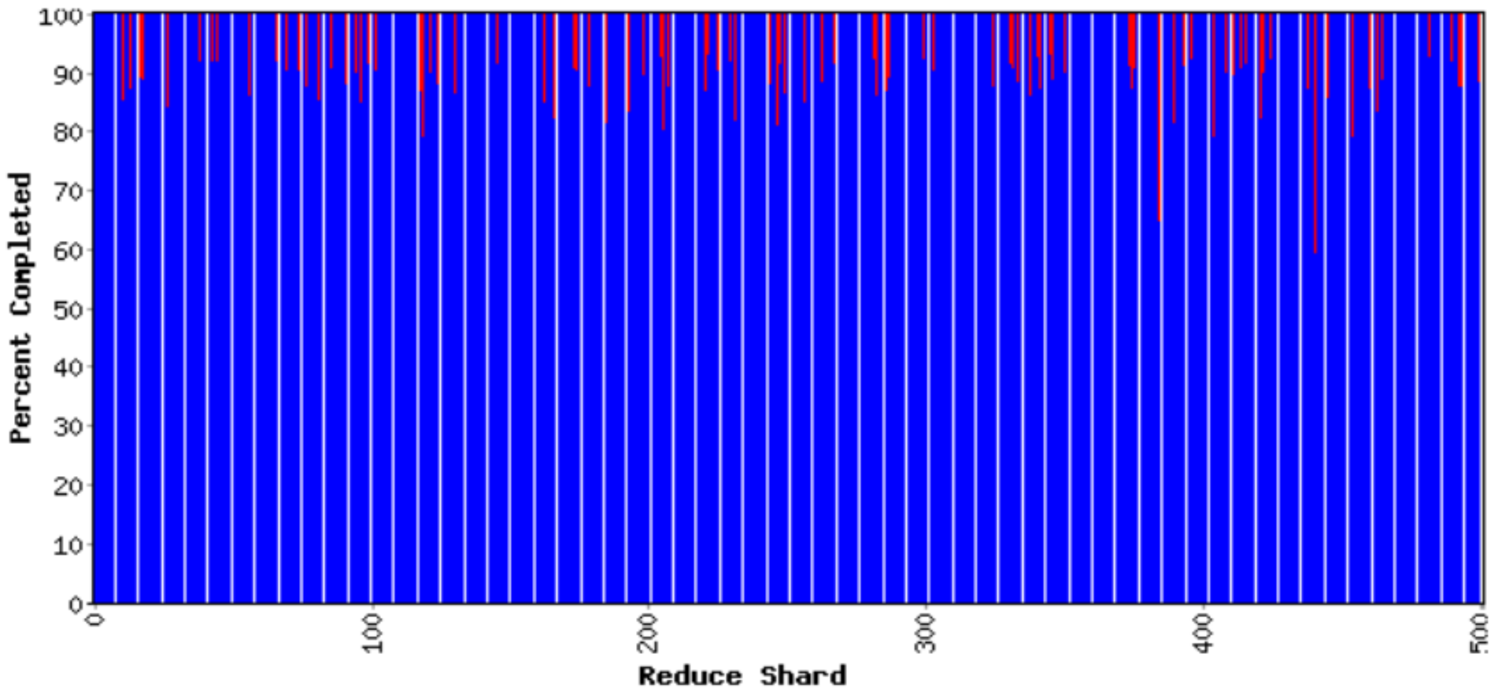
Variable	Minute
Mapped (MB/s)	0.0
Shuffle (MB/s)	0.0
Output (MB/s)	1222.0
doc-index-hits	0
docs-indexed	0
dups-in-index-merge	0
mr-merge-calls	51640600
mr-merge-outputs	51640600

MapReduce status: MR_Indexer-beta6-large-2003_10_28_00_03

Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 37 min 01 sec

1707 workers; 1 deaths

Type	Shards	Done	Active	Input(MB)	Done(MB)	Output(MB)
Map	13853	13853	0	878934.6	878934.6	523499.2
Shuffle	500	500	0	523499.2	520468.6	520468.6
Reduce	500	406	94	520468.6	512265.2	514373.3



Counters

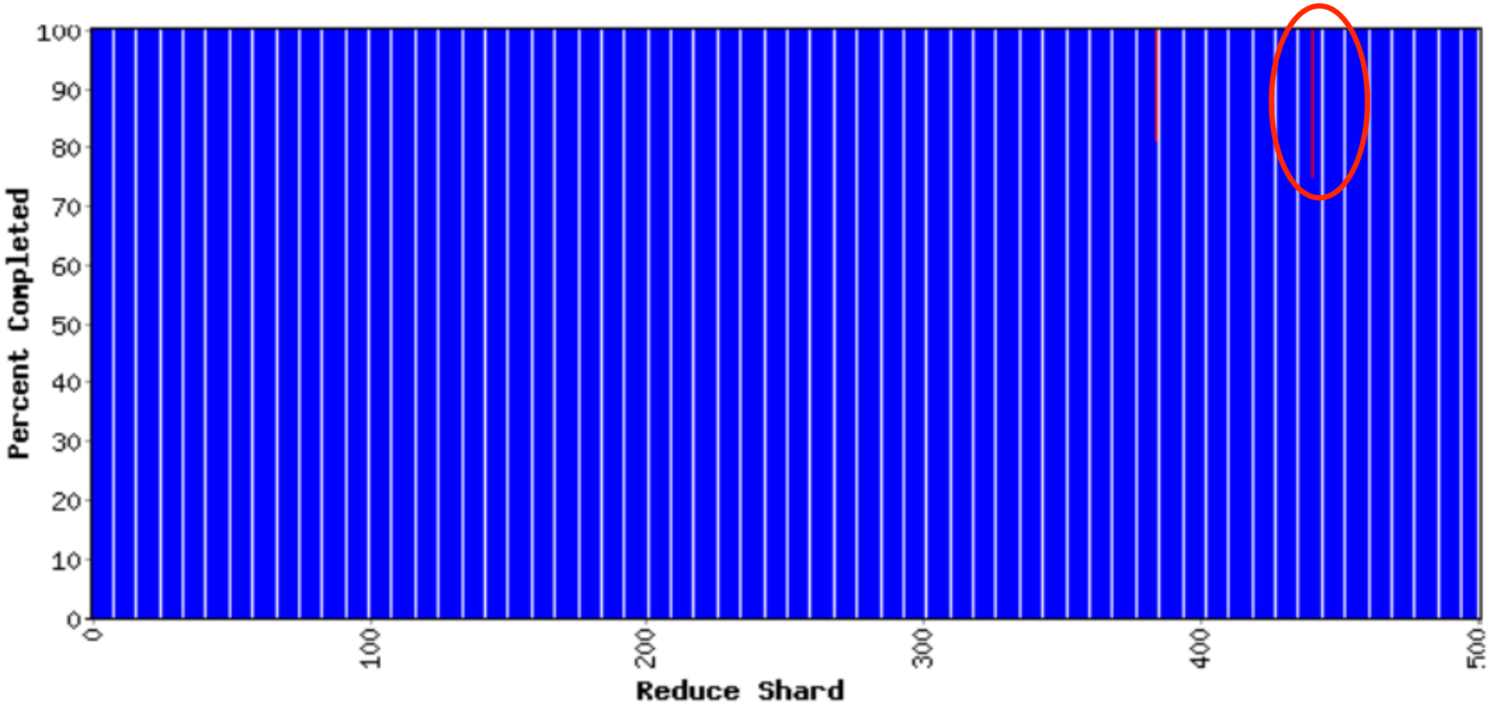
Variable	Minute	
Mapped (MB/s)	0.0	
Shuffle (MB/s)	0.0	
Output (MB/s)	849.5	
doc-index-hits	0	1
docs-indexed	0	
dups-in-index-merge	0	
mr-merge-calls	35083350	
mr-merge-outputs	35083350	

MapReduce status: MR_Indexer-beta6-large-2003_10_28_00_03

Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 38 min 56 sec

1707 workers; 1 deaths

Type	Shards	Done	Active	Input(MB)	Done(MB)	Output(MB)
Map	13853	13853	0	878934.6	878934.6	523499.2
Shuffle	500	500	0	523499.2	519781.8	519781.8
Reduce	500	498	2	519781.8	519394.7	519440.7



Counters

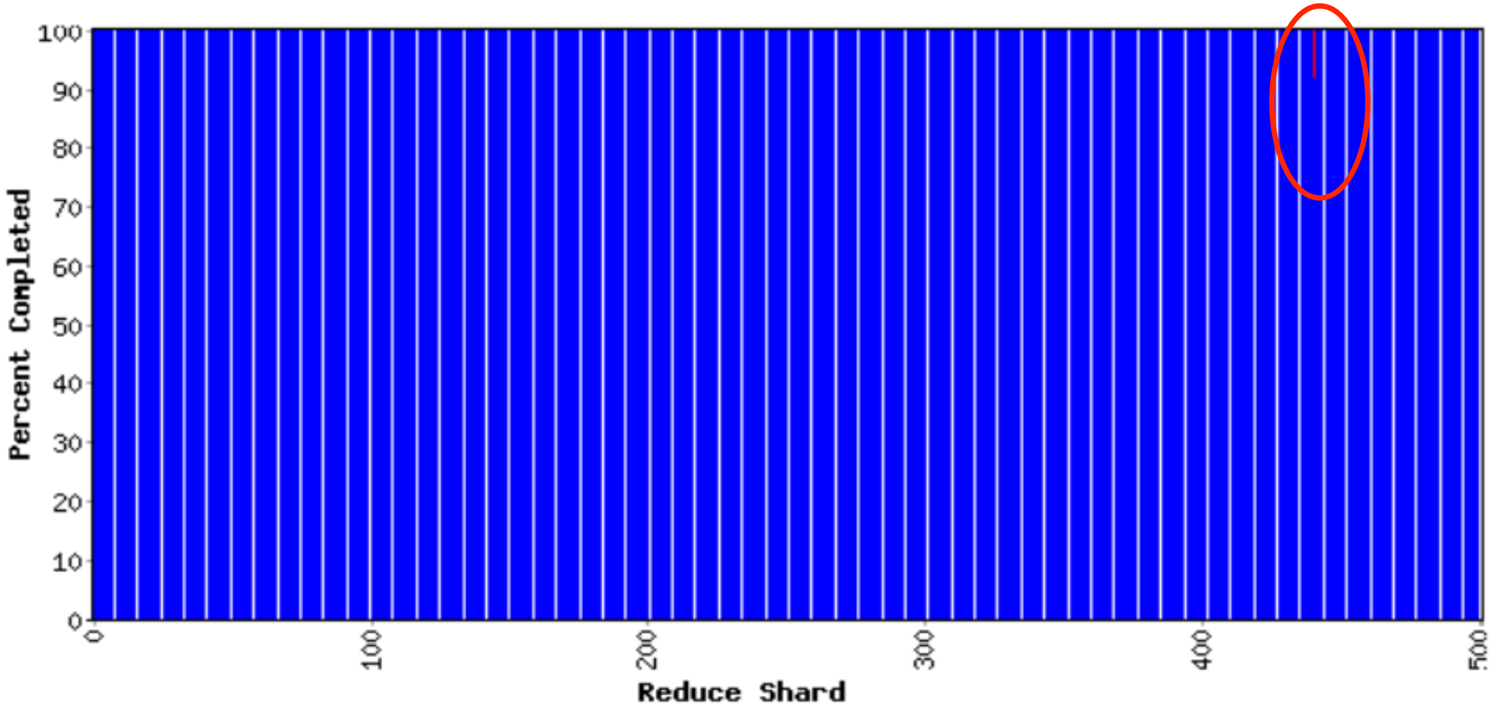
Variable	Minute	
Mapped (MB/s)	0.0	
Shuffle (MB/s)	0.0	
Output (MB/s)	9.4	
doc-index-hits	0	105
docs-indexed	0	
dups-in-index-merge	0	
mr-merge-calls	394792	
mr-merge-outputs	394792	

MapReduce status: MR_Indexer-beta6-large-2003_10_28_00_03

Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 40 min 43 sec

1707 workers; 1 deaths

Type	Shards	Done	Active	Input(MB)	Done(MB)	Output(MB)
Map	13853	13853	0	878934.6	878934.6	523499.2
Shuffle	500	500	0	523499.2	519774.3	519774.3
Reduce	500	499	1	519774.3	519735.2	519764.0



Counters

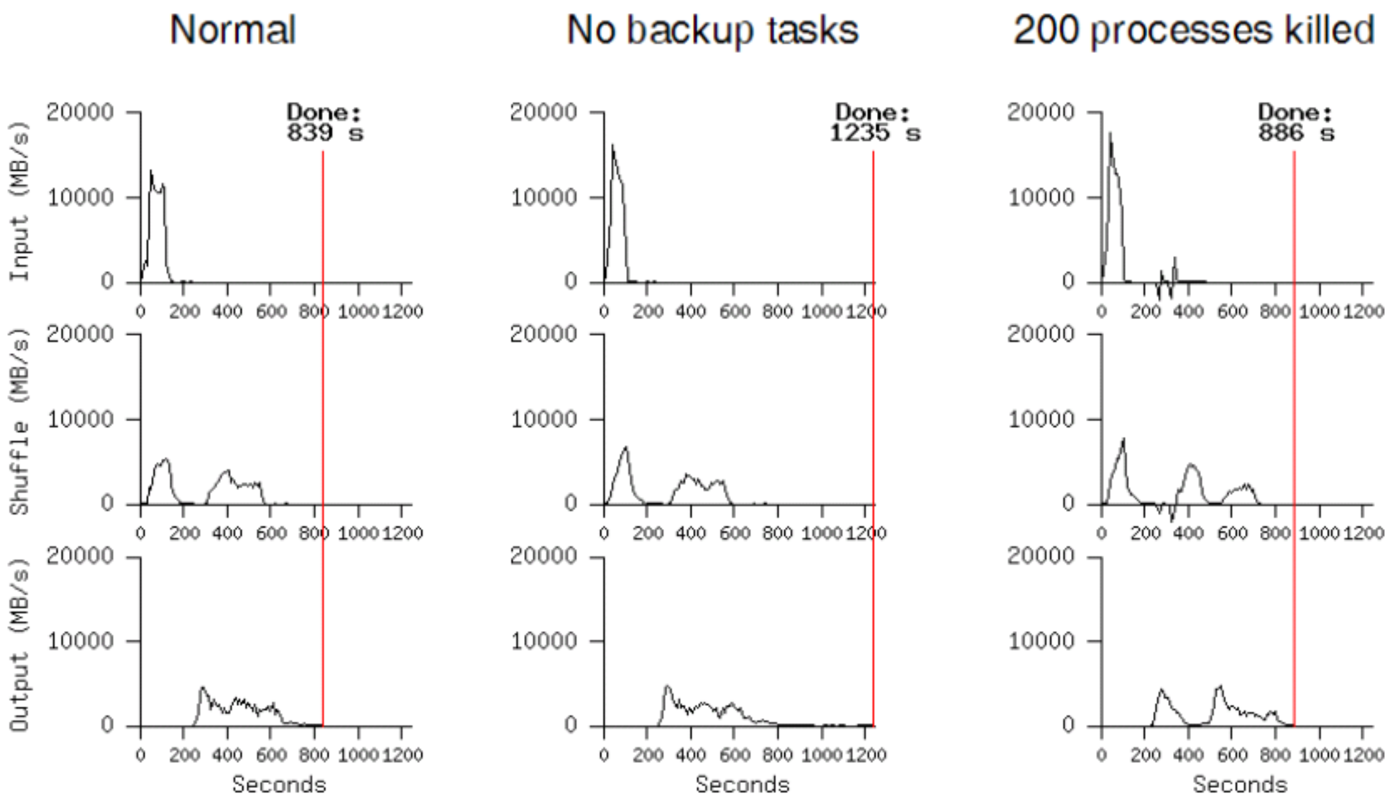
Variable	Minute	
Mapped (MB/s)	0.0	
Shuffle (MB/s)	0.0	
Output (MB/s)	1.9	
doc-index-hits	0	105
docs-indexed	0	
dups-in-index-merge	0	
mr-merge-calls	73442	
mr-merge-outputs	73442	

Refinement: Backup tasks

- **Problem**:
  - Slow workers significantly lengthen the job completion time:
    - Other jobs on the machine
    - Bad disks
    - Weird things
- **Solution**:
  - Near end of phase, spawn backup copies of tasks
    - Whichever one finishes first “wins”
- **Effect**:
  - Dramatically shortens job completion time

Refinements: Backup tasks

- Backup tasks reduce job time
- System deals with failures



Refinements: Combiners

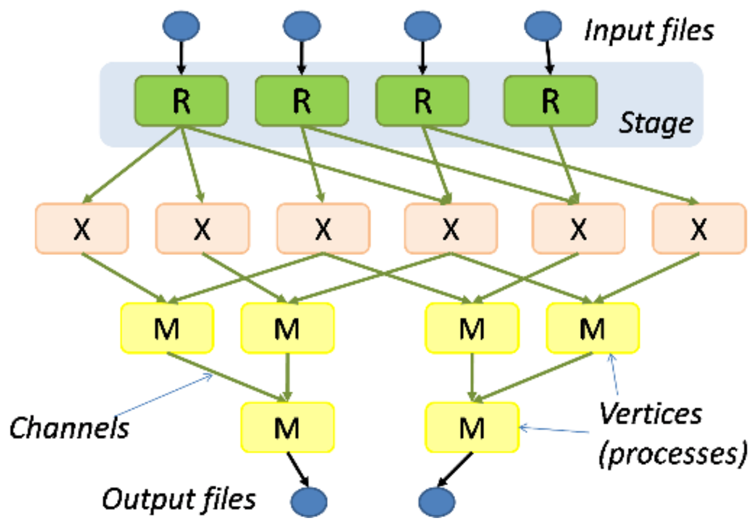
- Often a map task will produce many pairs of the form (k,v1), (k,v2), … for the same key k
  - E.g., popular words in Word Count
- Can save network time by pre-aggregating at mapper:
  - combine(k1, list(v1)) → v2
  - Usually same as reduce function
- Works whenever reduce function is commutative and associative

Refinements: Partition Function

- Inputs to map tasks are created by contiguous splits of input file
- Reduce needs to ensure that records with the same key end up at the same worker
- System uses a default partition function:
  - hash(key) mod R
- Sometimes useful to override:
  - E.g., hash(hostname(URL)) mod R ensures URLs from a host end up in the same output file

Implementations

- Google
  - Patented MapReduce in 2004
  - Not available outside Google
- Hadoop
  - An open-source implementation in Java
  - Uses HDFS for stable storage
  - Download: http://lucene.apache.org/hadoop/
- Disco
  - MapReduce for Python
- Microsoft DryadLINQ
  - Generalize MapReduce data flow



Cloud Computing

- Ability to rent computing by the hour
  - Additional services e.g., persistent storage
- Examples
  - Amazon Elastic Cloud (EC2)
  - Microsoft Azure
  - Google AppEngine
- All of those have some MapReduce implementations

What you need to know

- MapReduce
  - Simple paradigm for writing bug-free massively parallel code
  - User specifies map() and reduce() functions, MR framework does the rest
- Which type of problems fit the framework
- In future lectures, we'll see examples of more complex algorithms implemented in MR
- In HW1, you get to try it ☺

Acknowledgments

- Several slides adapted from Jeff Dean (Google) and Jure Leskovec (Stanford)