ETH zürich

Big Data and Machine Learning

Iza Moise, Olivia Woolley Meza, Lloyd Sanders, Nino Antulov-Fantulin

Roadmap

- Motivation
- Big Data
  - Definition
  - MapReduce
  - Hadoop
  - Spark
- ML on Text
  - Text Representation (tf-idf)
  - Topic Models
  - Examples on Twitter Data

Why Big Data for ML?

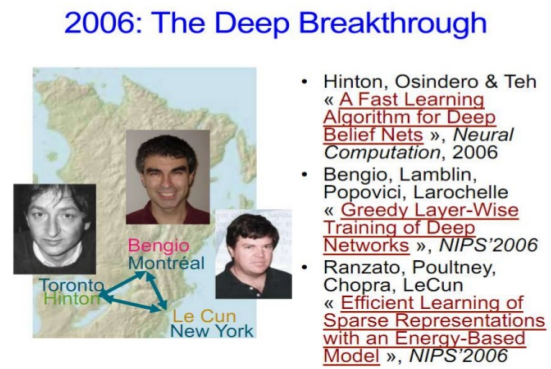
- the AI Winter: before 2006 people thought deep neural networks cannot be trained.

- Theoretical breakthroughs in 2006
  - Learned how to train deep neural networks
- Cheap computational power
  - GPUs can run neural networks in parallel
- Big data
  - collecting massive data about the world
  - used for “schooling” ML

Why Big Data for ML?

- the AI Winter: before 2006 people thought deep neural networks cannot be trained.

- Theoretical breakthroughs in 2006
  - Learned how to train deep neural networks



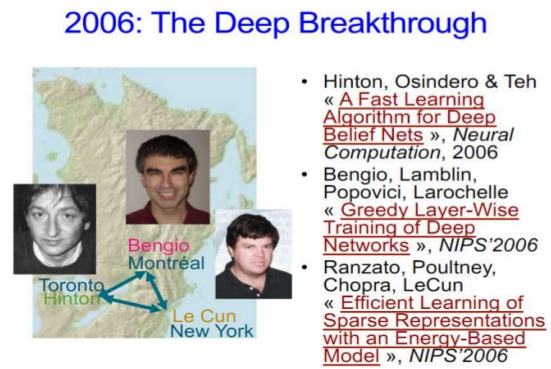
- Cheap computational power
  - GPUs can run neural networks in parallel

- Big data
  - collecting massive data about the world
  - used for "schooling" ML

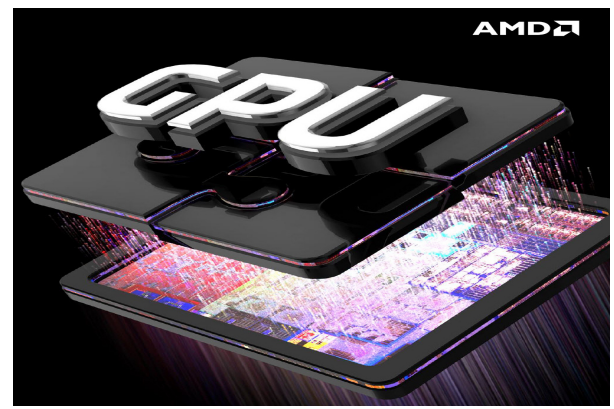
Why Big Data for ML?

- the AI Winter: before 2006 people thought deep neural networks cannot be trained.

- Theoretical breakthroughs in 2006
  - Learned how to train deep neural networks



- Cheap computational power
  - GPUs can run neural networks in parallel

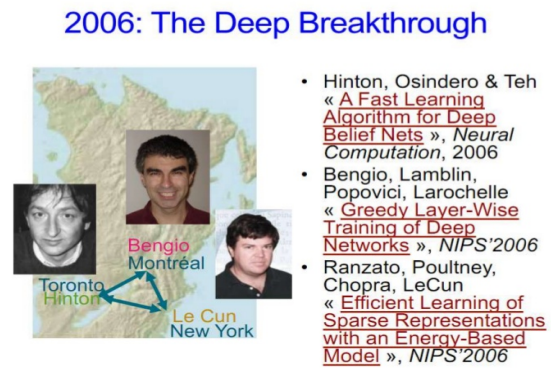


- Big data
  - collecting massive data about the world
  - used for "schooling" ML

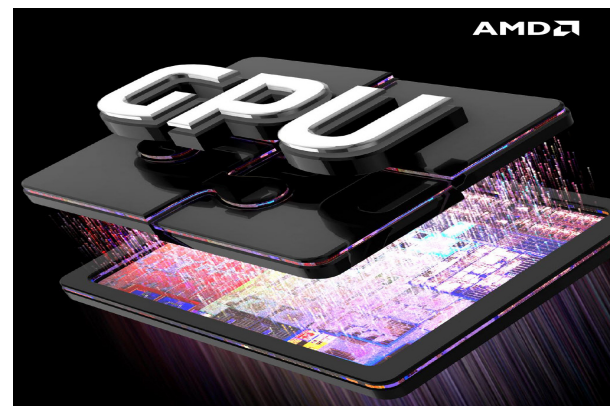
Why Big Data for ML?

- the AI Winter: before 2006 people thought deep neural networks cannot be trained.

- Theoretical breakthroughs in 2006
  - Learned how to train deep neural networks



- Cheap computational power
  - GPUs can run neural networks in parallel



- Big data
  - collecting massive data about the world
  - used for "schooling" ML

What is Big Data?

"Big Data refers to datasets whose size is beyond the ability of typical database software tools to capture, store, manage and analyze." (*McKinsey Global Institute*)

"Big Data is the term for a collection of datasets so large and complex that it becomes difficult to process using on-hand database management tools or traditional data processing applications." (*Wikipedia*)

Social Media Landscape 2015

FredCavazza.net

- Social media → 90%of today's data
- Fastest means of population feedback and provides cheap two-way information sharing
- Personal nature of the data

Social Media Landscape 2015

FredCavazza.net

- Social media → 90%of today's data
- Fastest means of population feedback and provides cheap two-way information sharing
- Personal nature of the data

Social Media Landscape 2015

FredCavazza.net

- Social media → 90%of today's data
- Fastest means of population feedback and provides cheap two-way information sharing
- Personal nature of the data

Social Media Landscape 2015

FredCavazza.net

- Social media $\rightarrow$ 90%of today's data
- Fastest means of population feedback and provides cheap two-way information sharing
- Personal nature of the data

Every minute of every day

- More than 204 million email messages
- Over 2 million Google search queries
- 48 hours of new YouTube videos
- 684,000 bits of content shared on Facebook
- More than 100,000 tweets
- $272,000 spent on e-commerce

Big picture

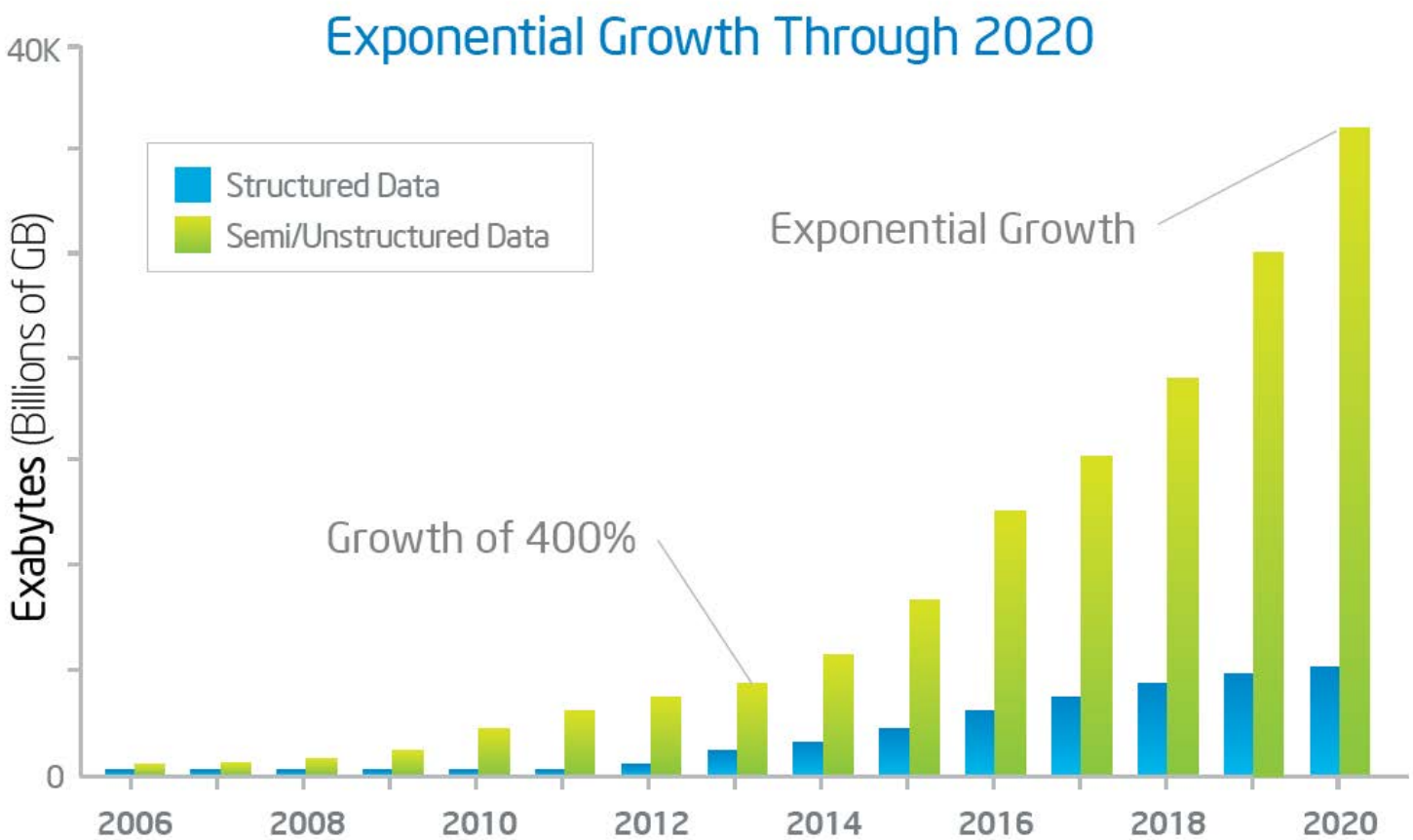
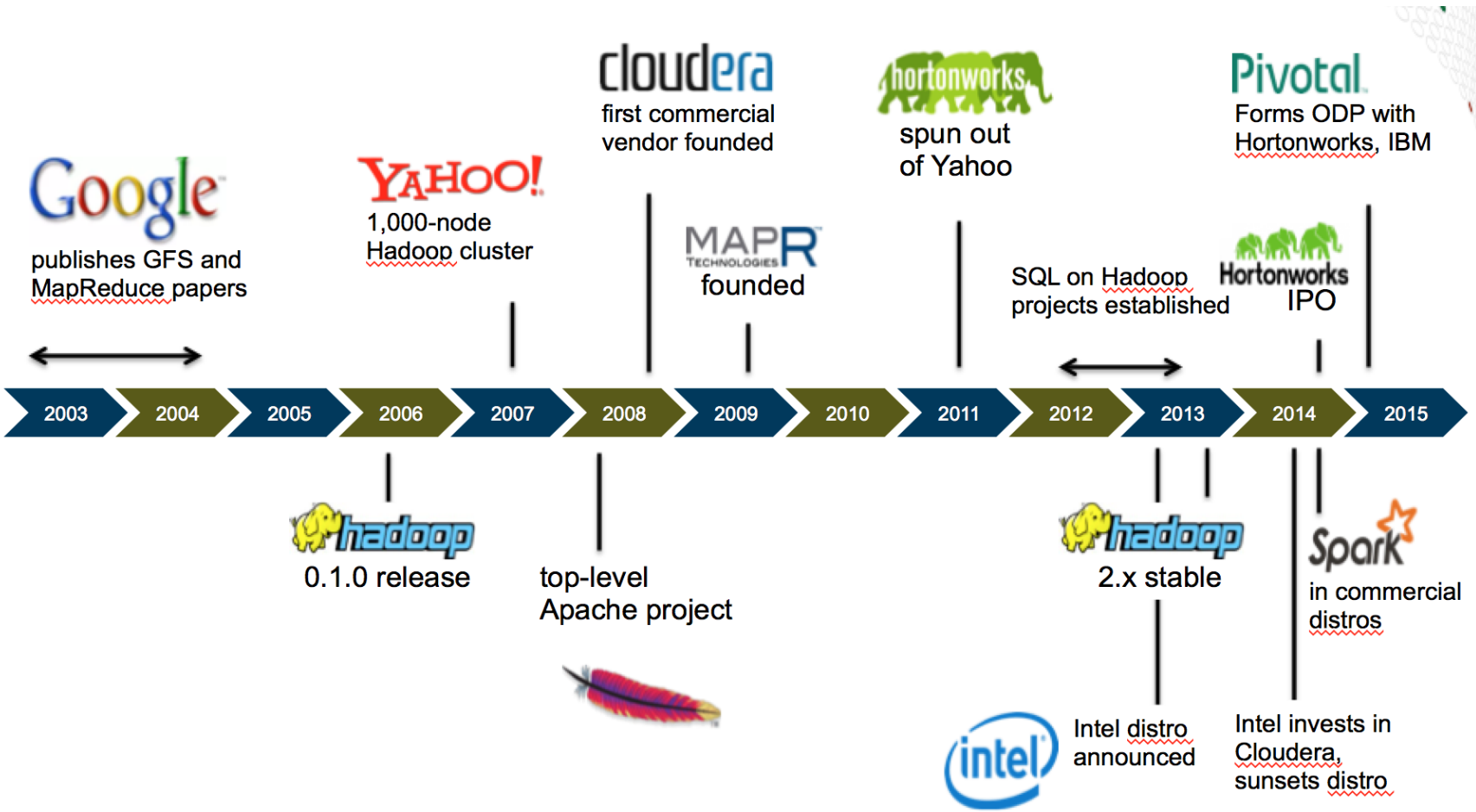


Figure 1. Current and forecasted growth of big data. Source: Philippe Botteri of Accel Partners, Feb. 2013.

The 5 V's

- Volume
  - → large amounts of data generated every second (emails, twitter messages, videos, sensor data...)
- Velocity
  - → the speed of data moving in and out data management systems (videos going viral...)
  - → "on-the-fly"
- Variety
  - → different data formats in terms of structured or unstructured (80%) data
- Value
  - → insights we can reveal within the data
- Veracity
  - → trustworthiness of the data

Big Data tools over the years



The First Big Data Problem

The Internet Search Engine:

> In 2002, Google wanted to be able to crawl the web and index the content so that they could produce an internet search engine. The standard method to organize and store data in 2002 was by means of relational database management systems (RDBMS) which were accessed in a language called SQL. But almost all SQL and relational stores were not appropriate for internet search engine storage and retrieval because they were costly, not terribly scalable, not as tolerant to failure as required and not as performant as desired.

MapReduce

MapReduce

- An abstraction for performing computations on data
  - ✓ automatic parallelization of computations
  - ✓ large-scale data distribution
  - ✓ simple, yet powerful interface
  - ✓ user-transparent fault tolerance
  - ✓ commodity hardware
- Introduced by Google in 2004: paradigm and implementation

Motivation: Common operations on data

- Iterate over a large number of records
- Extract something of interest from each
- Shuffle and sort intermediate results
- Aggregate intermediate results
- Generate final output

Motivation: Common operations on data

- Iterate over a large number of records
- Extract something of interest from each
- Shuffle and sort intermediate results
- Aggregate intermediate results
- Generate final output

Provide a functional abstraction for these two operations

Motivation: Common operations on data

- Iterate over a large number of records
- Extract something of interest from each
- Shuffle and sort intermediate results
- Aggregate intermediate results
- Generate final output

Map

Reduce

Provide a functional abstraction for these two operations

MapReduce model outline

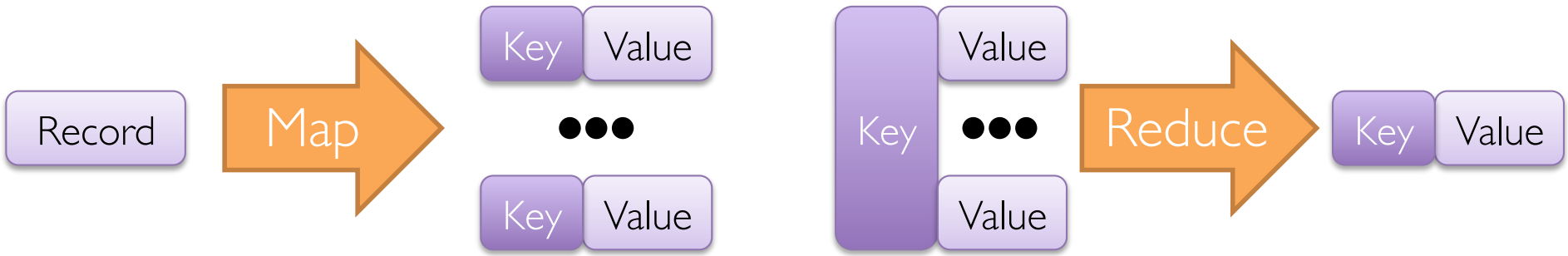
Typical problem solved by MapReduce:

- Read a lot of data
- Map: extract something interesting from each record
- Shuffle and Sort
- Reduce: aggregate, summarize, filter or transform
- Write the results

Outline stays the same, map and reduce change to fit the problem

Map-Reduce Abstraction

[Dean & Ghemawat, OSDI'04]



Example: *Word-Count*

```
Map(docRecord) {
  for (word in docRecord) {
    emit (word, 1)
  }
}
```

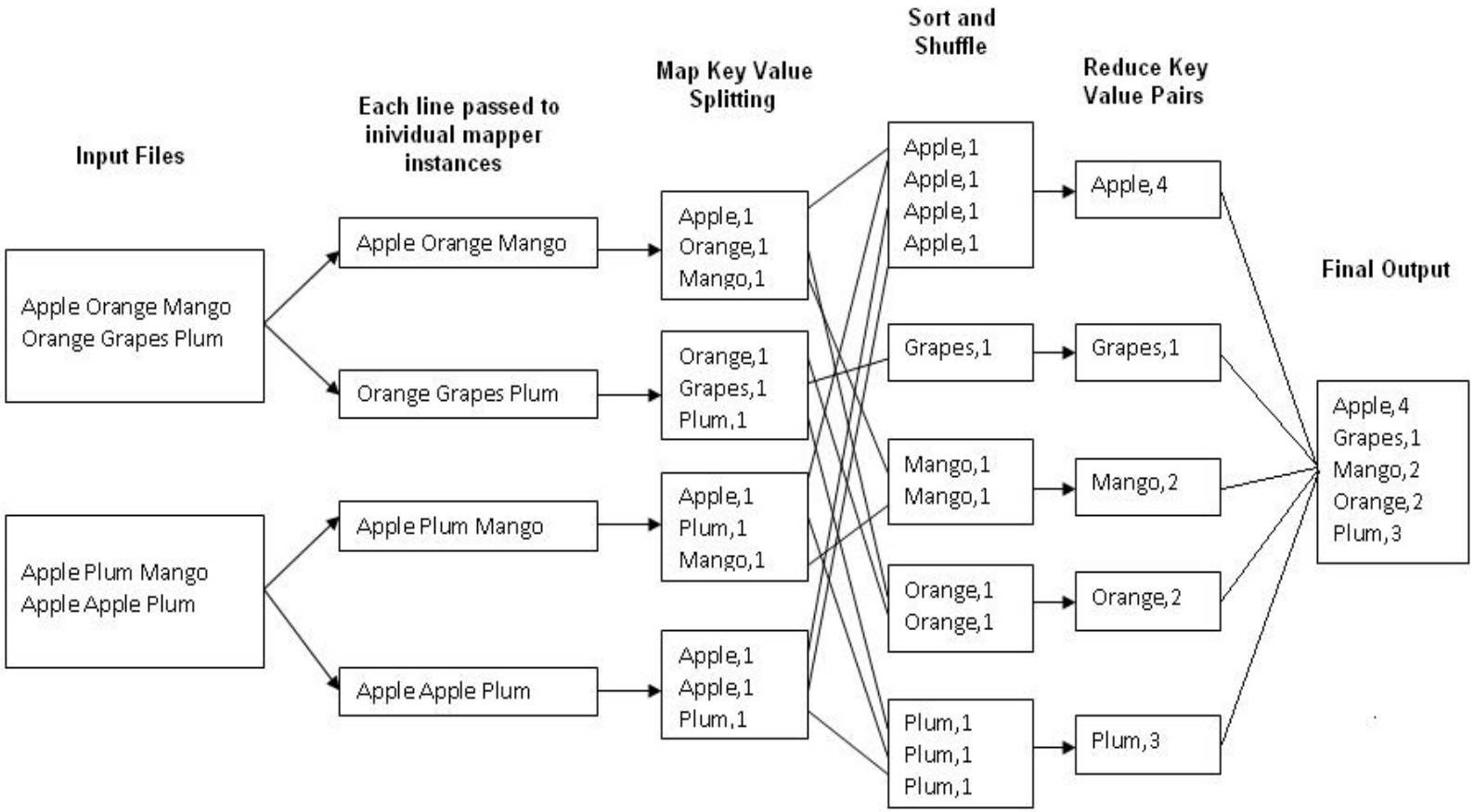
Key Value

```
Reduce(word, counts) {
  emit (word, SUM(counts))
}
```

Map: Idempotent

Reduce: Commutative and Associative

Example: Word Count



Reduce phase is optional: Jobs can be Map-only

What is MapReduce used for?

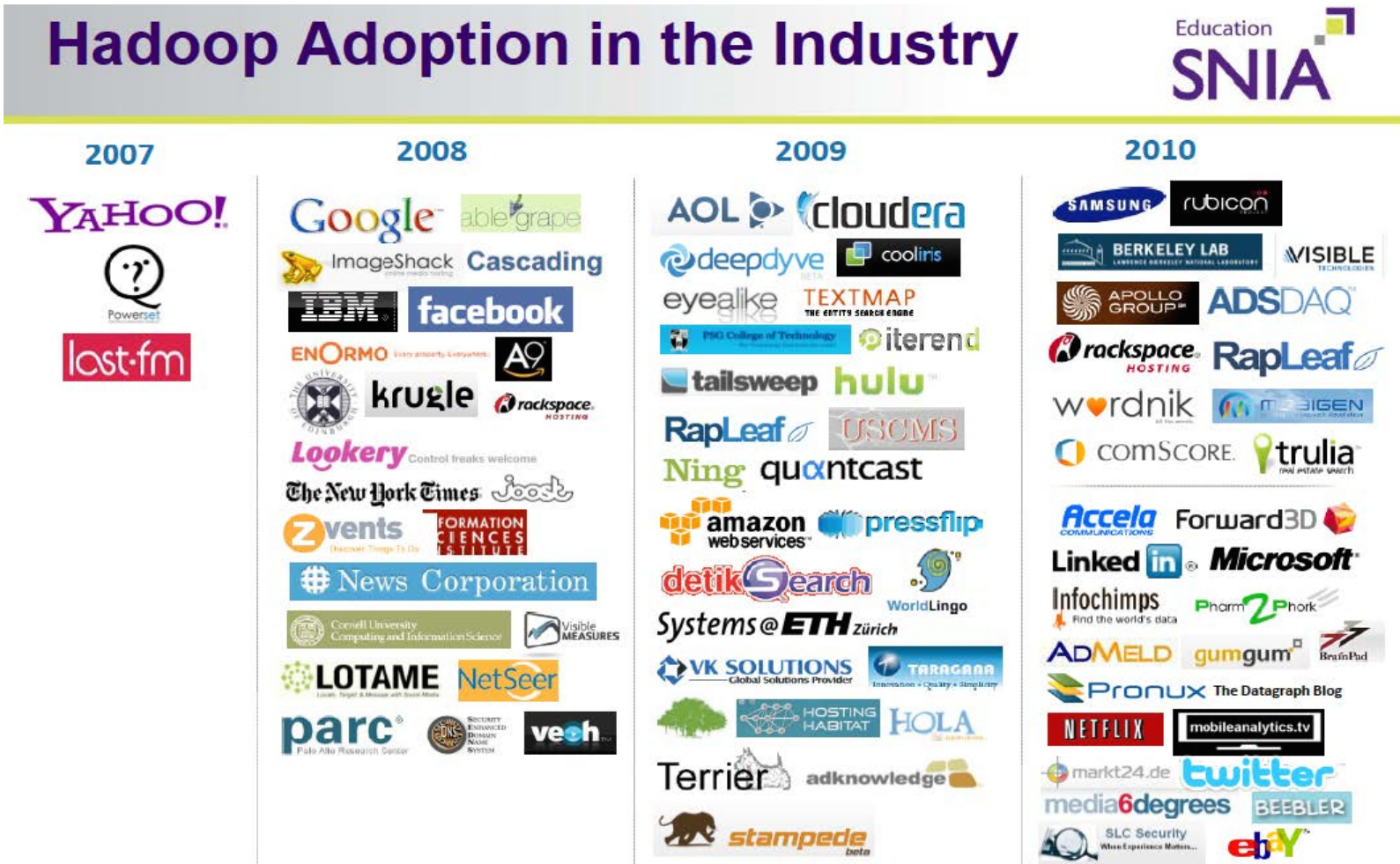
- At Google
  - index construction for Google Search
  - article clustering for Google News
  - statistical machine translation
  - distributed grep, distributed sort
  - web access log stats
  - web link-graph reversal
  - inverted index construction
- At Yahoo!
  - “Web map” powering Yahoo! Search
  - Spam detection for Yahoo! Mail
- At Facebook
  - Data mining
  - Ad optimization
  - Spam detection

Elephant in the room

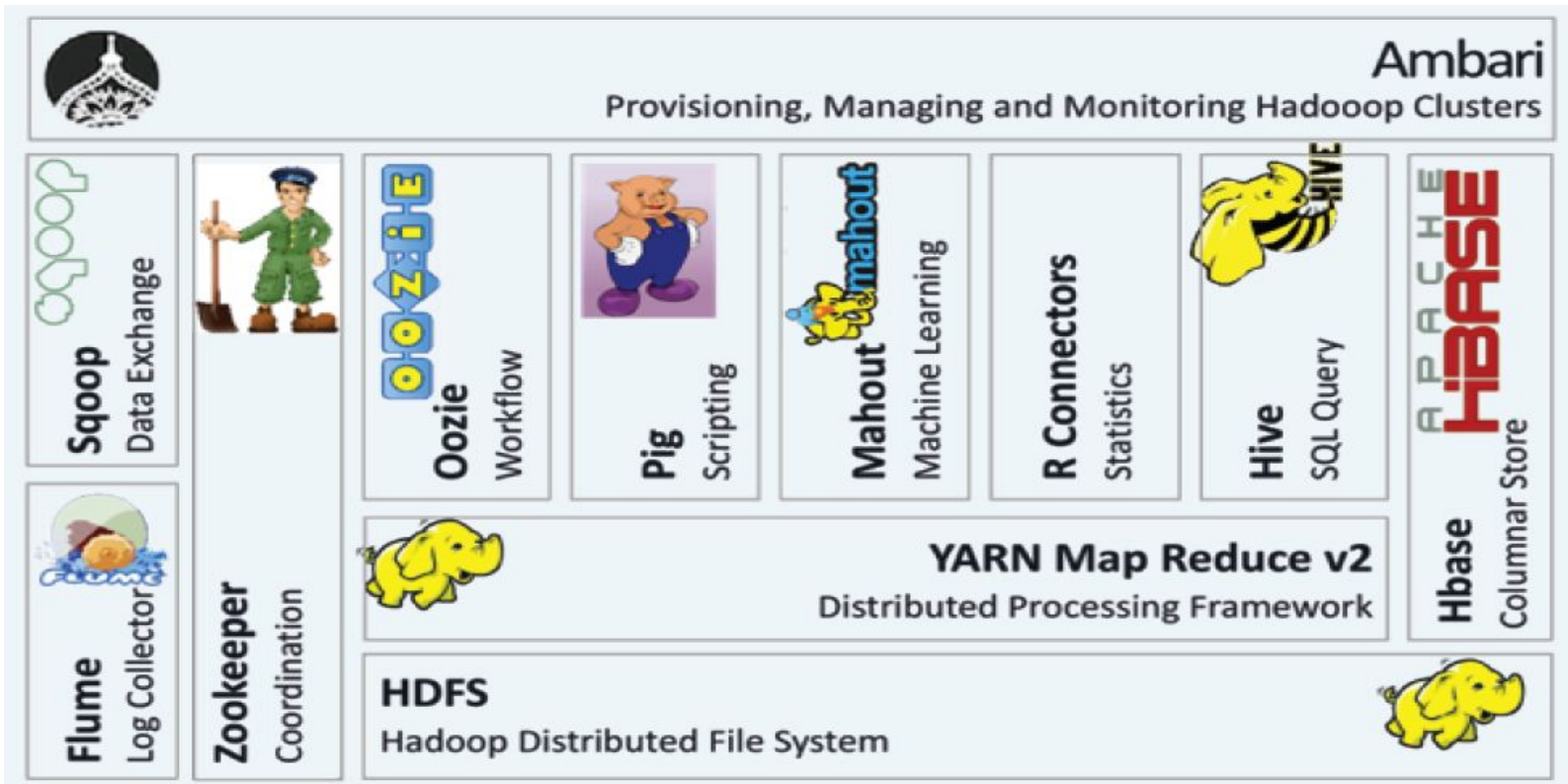
The Hadoop Project

- Open-source project started in 2006
- Developed in Java
- Two core components: HDFS and MapReduce implementation
- Founded by Apache
- Platform for data storage and processing
  - ✓ Scalable
  - ✓ Fault tolerant
  - ✓ Distributed
  - ✓ Any type of complex data

Who uses Hadoop?



Apache Hadoop Ecosystem



HDFS

Distributed File System

- Data split into large **chunks** of fixed (configurable) size – 64MB default
- Optimised for managing large files:
  - Write-once-read-many access pattern
  - Data I/O involving blocks of data, rather than random access
- Fault-tolerant
  - chunk-level replication
  - 3 replicas: local, same rack, different rack
- File System interface
  - Most FS commands
  - ls, mkdir, cat, rm, etc
  - put, get, distcp, etc

HDFS

Distributed File System

- Data split into large **chunks** of fixed (configurable) size – 64MB default
- Optimised for managing large files:
  - Write-once-read-many access pattern
  - Data I/O involving blocks of data, rather than random access
- Fault-tolerant
  - chunk-level replication
  - 3 replicas: local, same rack, different rack
- File System interface
  - Most FS commands
  - ls, mkdir, cat, rm, etc
  - put, get, distcp, etc

HDFS

Distributed File System

- Data split into large **chunks** of fixed (configurable) size – 64MB default
- Optimised for managing large files:
  - Write-once-read-many access pattern
  - Data I/O involving blocks of data, rather than random access
- Fault-tolerant
  - chunk-level replication
  - 3 replicas: local, same rack, different rack
- File System interface
  - Most FS commands
  - ls, mkdir, cat, rm, etc
  - put, get, distcp, etc

HDFS

Distributed File System

- Data split into large **chunks** of fixed (configurable) size – 64MB default
- Optimised for managing large files:
  - Write-once-read-many access pattern
  - Data I/O involving blocks of data, rather than random access
- Fault-tolerant
  - chunk-level replication
  - 3 replicas: local, same rack, different rack
- File System interface
  - Most FS commands
  - ls, mkdir, cat, rm, etc
  - put, get, distcp, etc

HDFS

Distributed File System

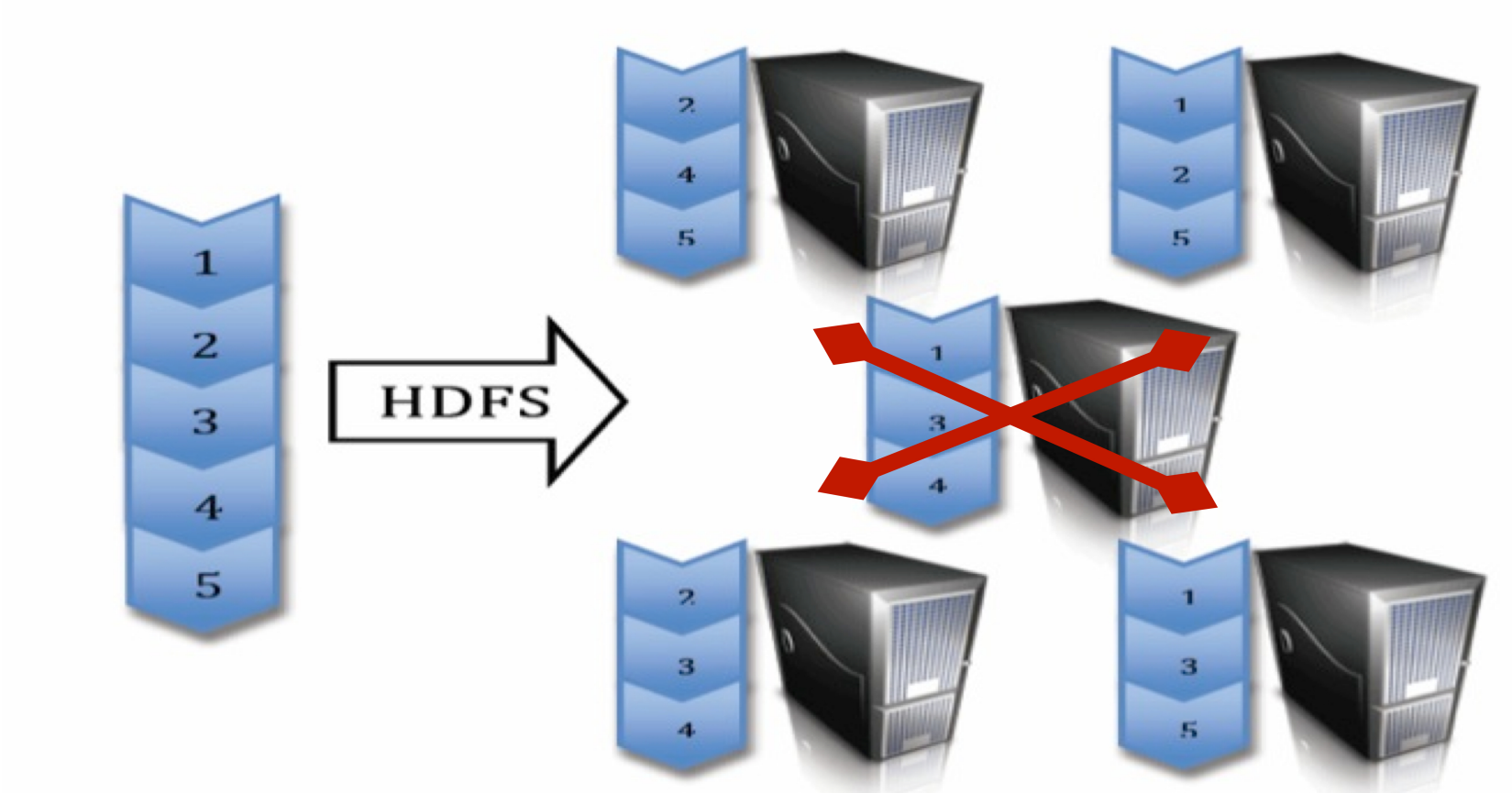
- Data split into large **chunks** of fixed (configurable) size – 64MB default
- Optimised for managing large files:
  - Write-once-read-many access pattern
  - Data I/O involving blocks of data, rather than random access
- Fault-tolerant
  - chunk-level replication
  - 3 replicas: local, same rack, different rack
- File System interface
  - Most FS commands
  - ls, mkdir, cat, rm, etc
  - put, get, distcp, etc

HDFS

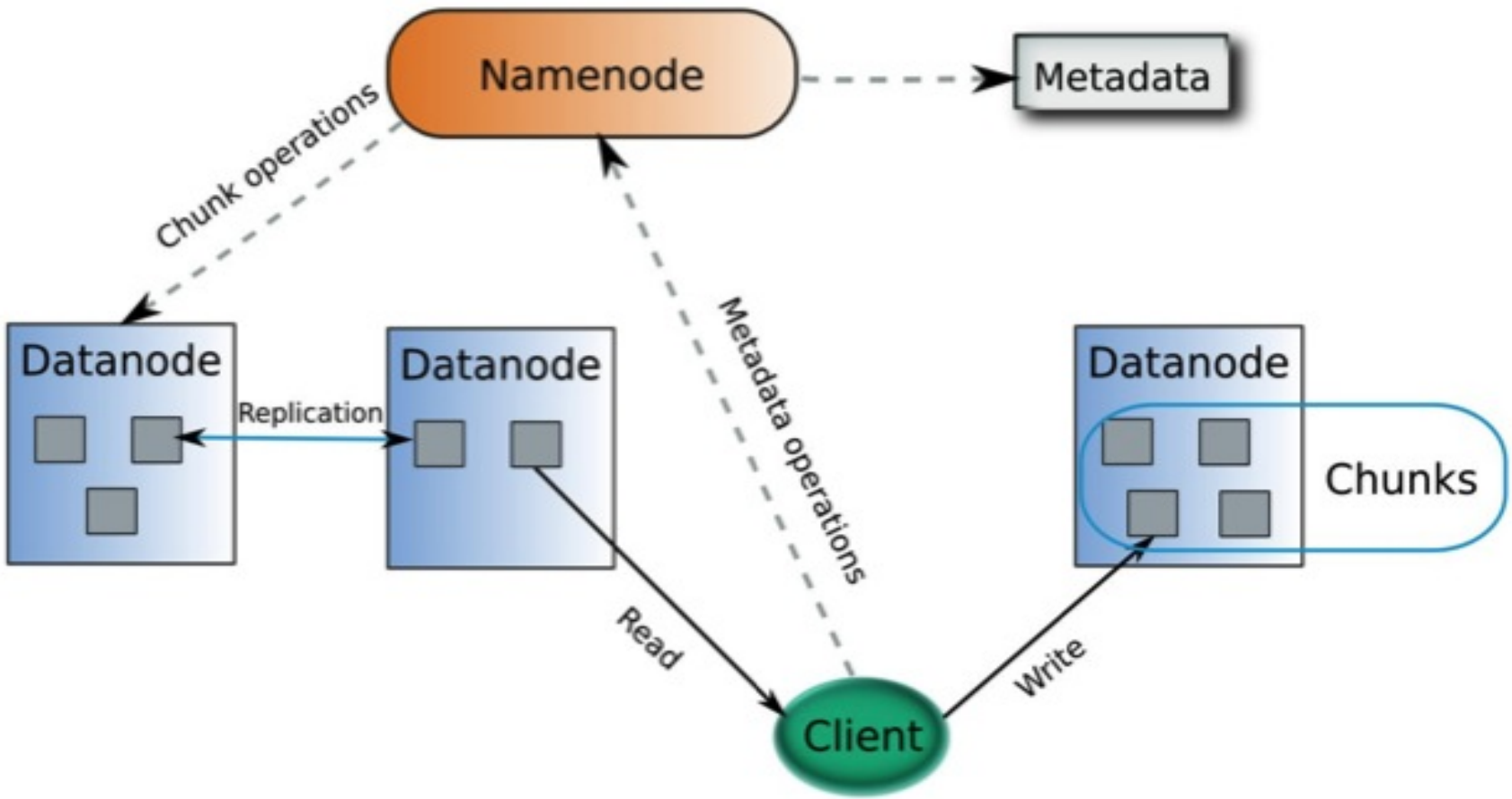
Distributed File System

- Data split into large **chunks** of fixed (configurable) size – 64MB default
- Optimised for managing large files:
  - Write-once-read-many access pattern
  - Data I/O involving blocks of data, rather than random access
- Fault-tolerant
  - chunk-level replication
  - 3 replicas: local, same rack, different rack
- File System interface
  - Most FS commands
  - ls, mkdir, cat, rm, etc
  - put, get, distcp, etc

HDFS-Hadoop Distributed File System



HDFS Design: Master - Slave



HDFS Architecture

HDFS Master: **Namenode**

- manages the filesystem namespace
  - list of files
  - for each file name: a set of blocks
  - for each block: a set of DataNodes
  - file attributes (creation time, replication factor)
- manages block replication
- single point of failure
  - if the Namenode fails, the filesystem is not usable anymore
  - the metadata can be stored on a remote disk so that the namespace can be reconstructed if the Namenode fails

HDFS Architecture

HDFS Slaves: **DataNodes**

- A DataNode is a block server
  - Stores data in the local file system (e.g. ext3)
  - Stores meta-data of a block (e.g. CRC)
  - Serves data and meta-data to Clients
- Block report
  - Periodically sends a report of all existing chunks to the NameNode
- Perform replication tasks upon instruction by NameNode

Hadoop MapReduce

Concepts:

- MapReduce job: the implementation of a MapReduce application (java file, jar file, python, shell script, etc)
- Map Tasks / Reduce Tasks: computations that are executed in parallel
  - Each map task processes a single chunk of the input data stored in HDFS
    - number of map tasks equals the number of input splits
  - Reduce tasks process Map tasks output, by splitting the key range
    - the number of Reduce tasks is usually a lot smaller
    - each Reduce task produces its separate output file in HDFS
- Records are data elements: (*key*, *value*) pairs

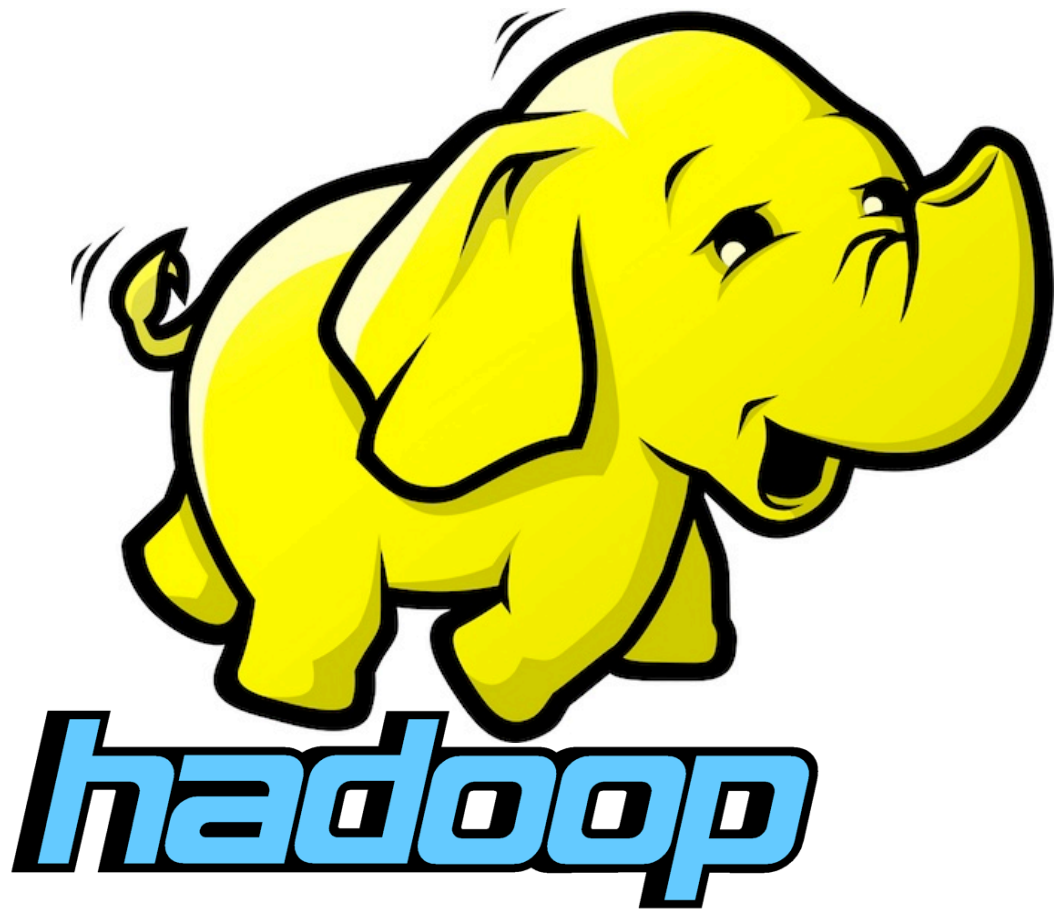
Hadoop MapReduce

The MapReduce Framework:

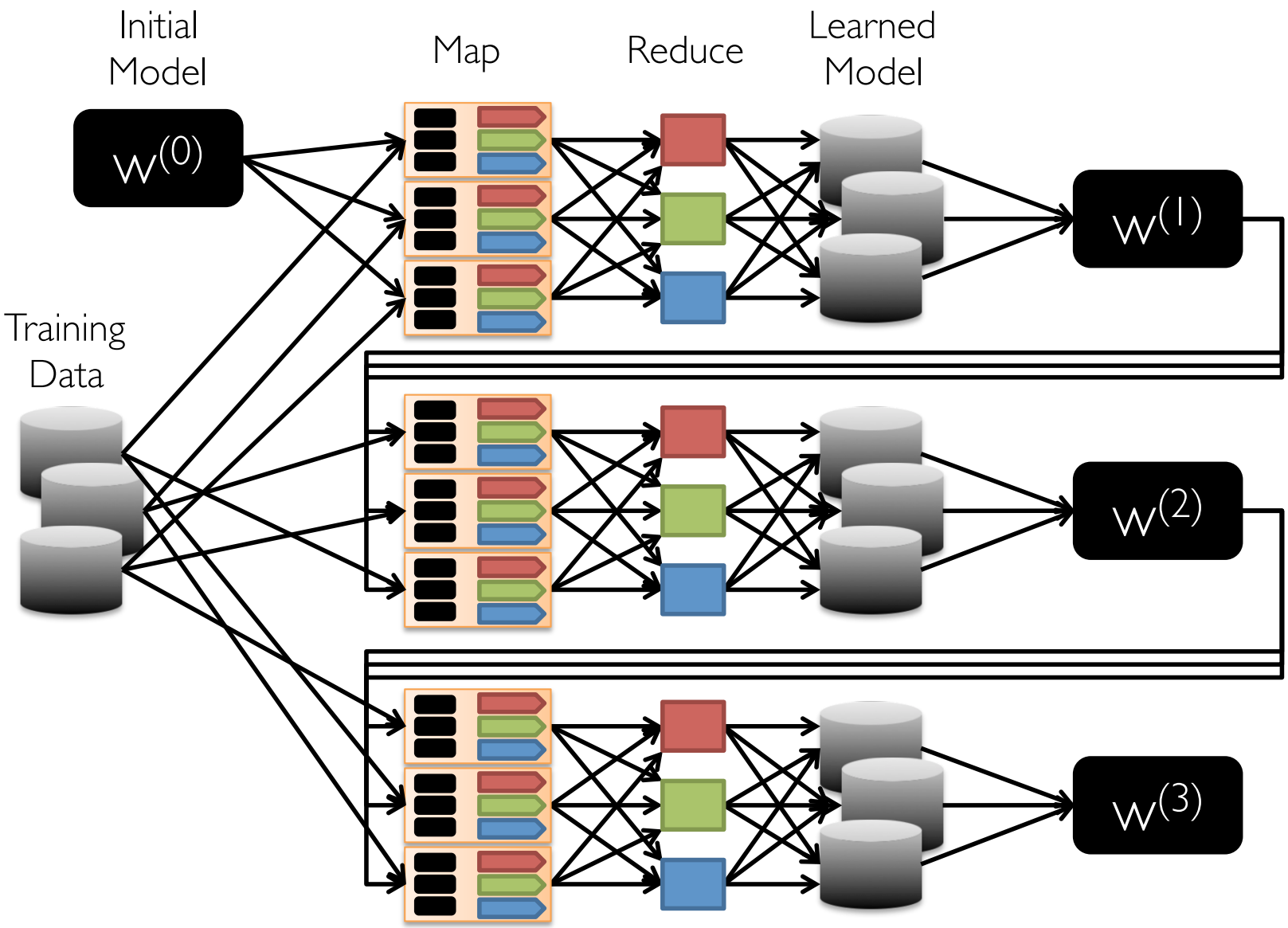
- Sorts the map output and also the reduce output (the final output data)
- Handles transparently tasks scheduling, monitoring and re-execution in case of failure
- Starts map tasks on the same machines that store the assigned data chunks (data locality)

Limitations?

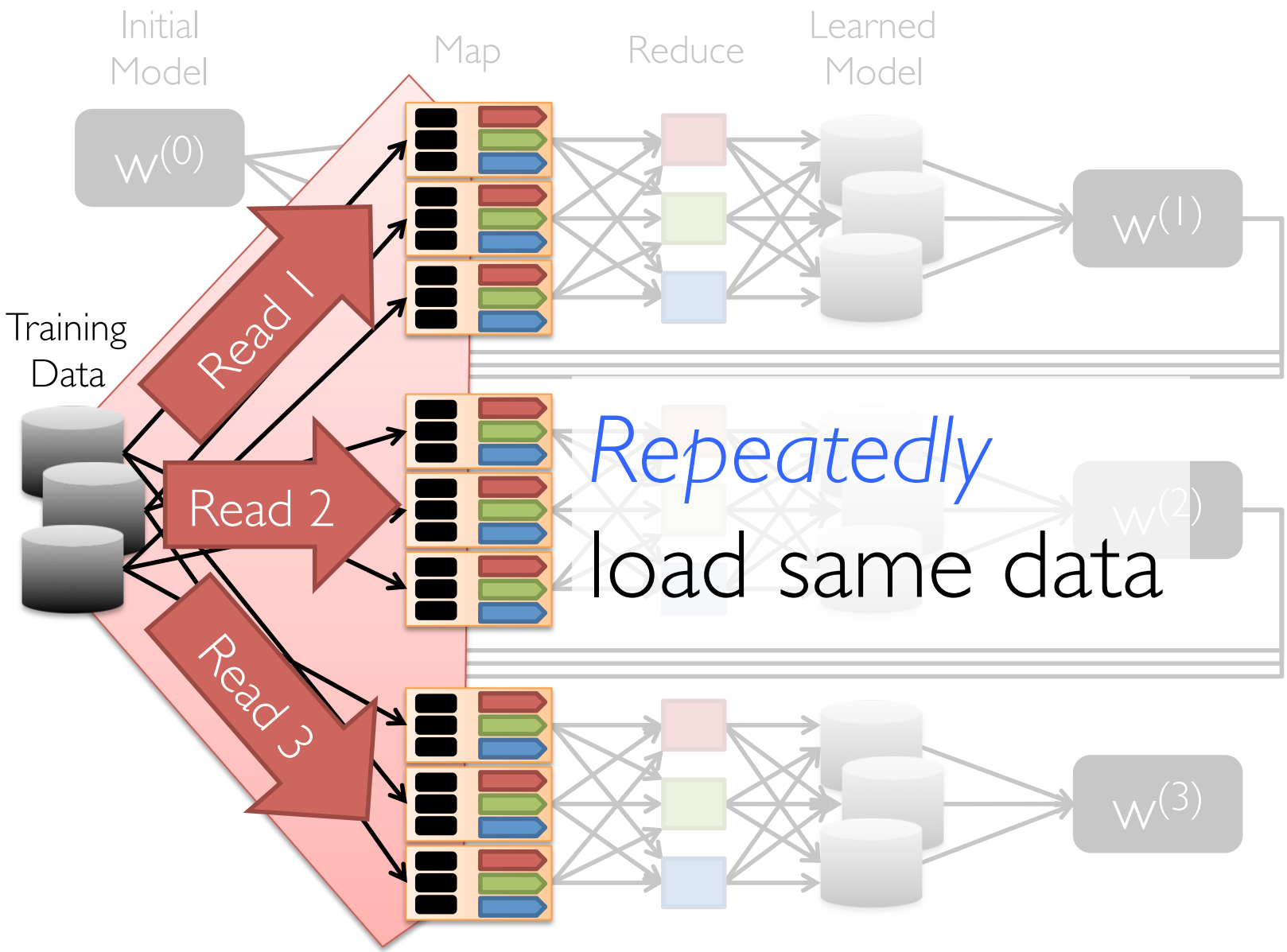
Map-Reduce



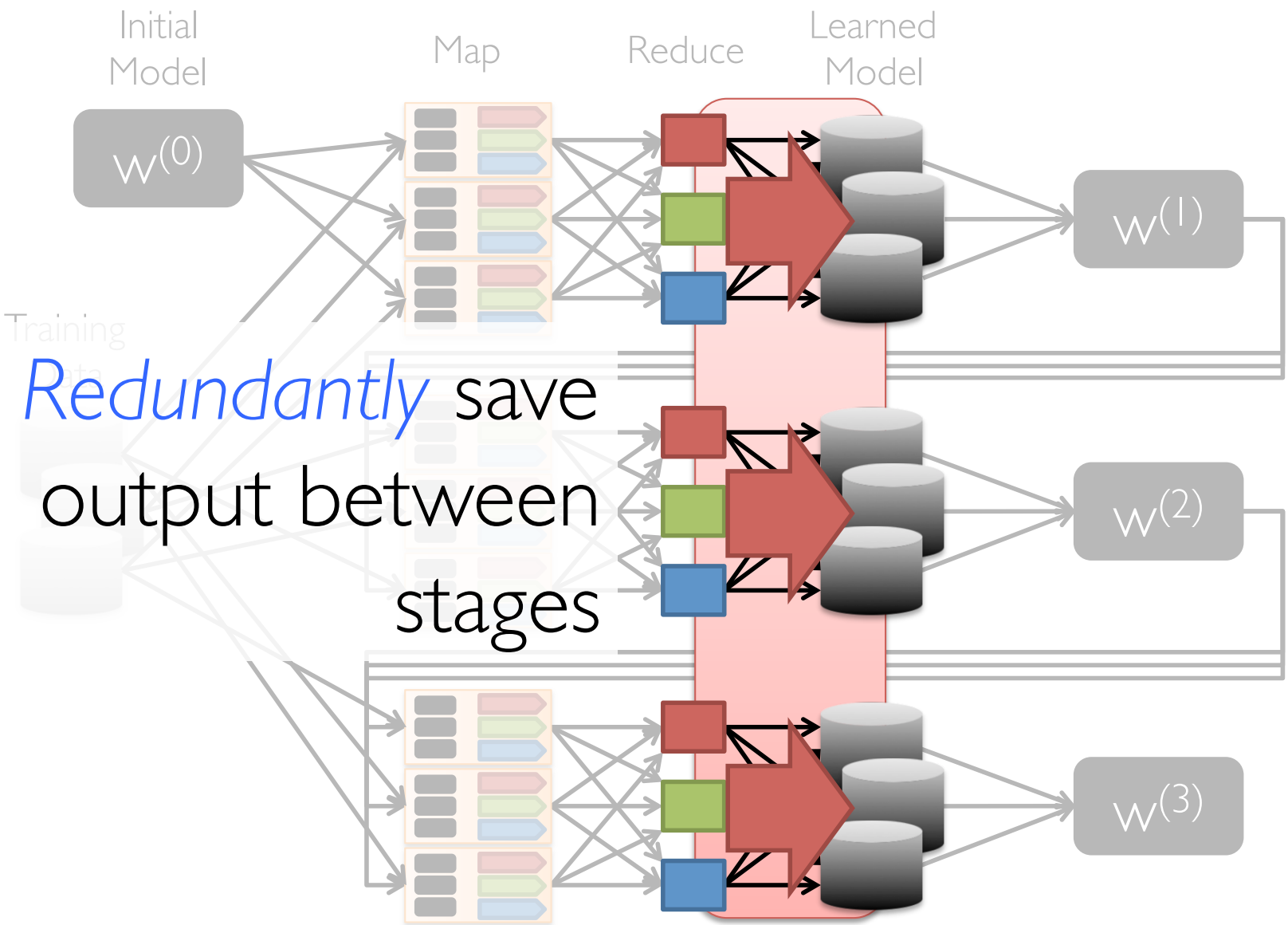
Iteration in Map-Reduce



Cost of Iteration in Map-Reduce



Cost of Iteration in Map-Reduce



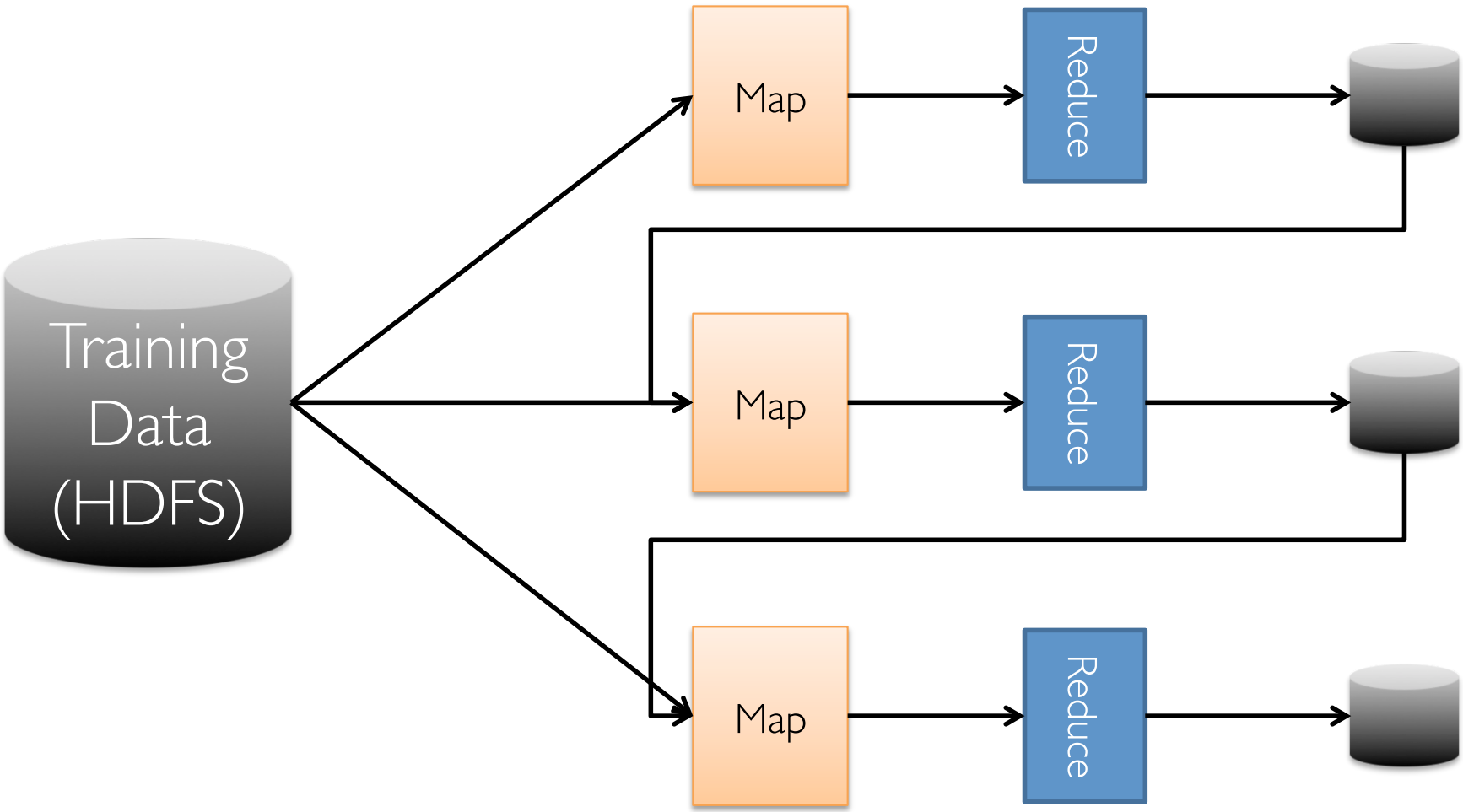
Iteration and Multi-stage computation

In-Memory Dataflow System

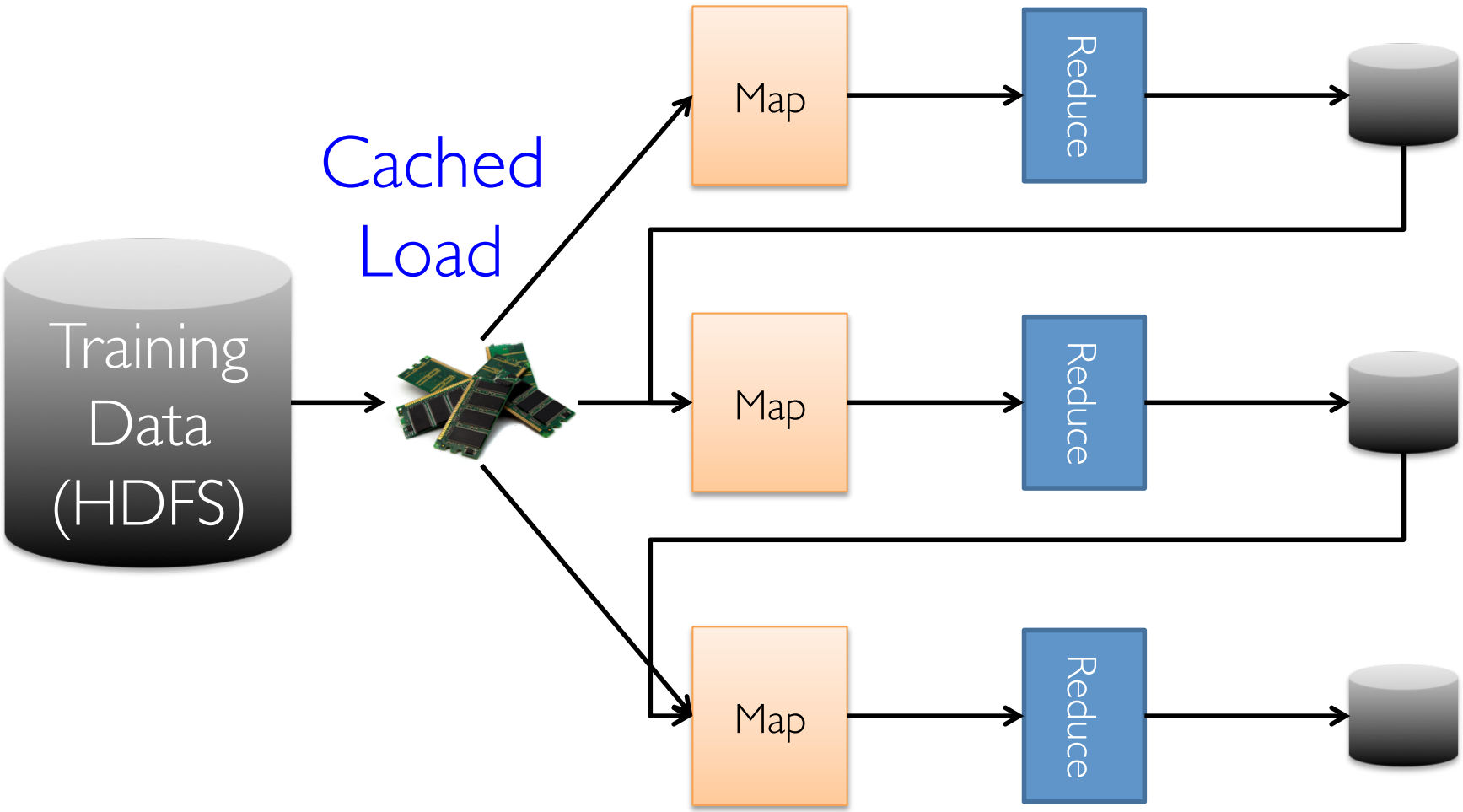
M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica. *Spark: cluster computing with working sets.* HotCloud'10

M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M.J. Franklin, S. Shenker, I. Stoica. *Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing*, NSDI 2012

Dataflow View

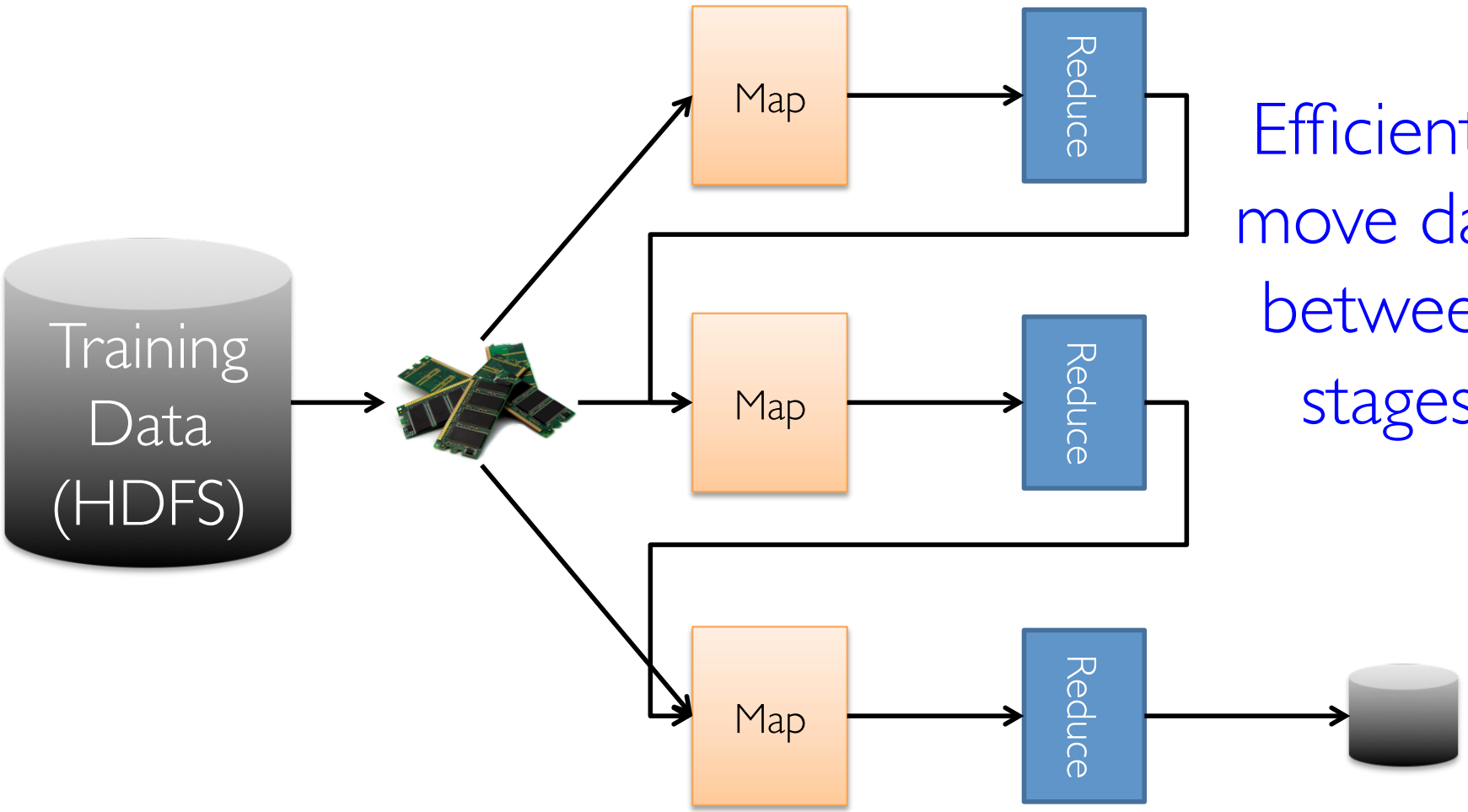


Memory Opt. Dataflow



10-100× faster than network and disk

Memory Opt. Dataflow View



Efficiently move data between stages

SPARK

- Apache Spark is a general-purpose analytics framework
  - Originally from Berkeley AMPLab/BDAS stack, now an Apache project
- Improves efficiency through:
  - In-memory computing primitives
  - Pipelined computation
- Improves usability through:
  - Rich APIs in Scala, but Java, Python, and R APIs also available
  - Interactive Shell

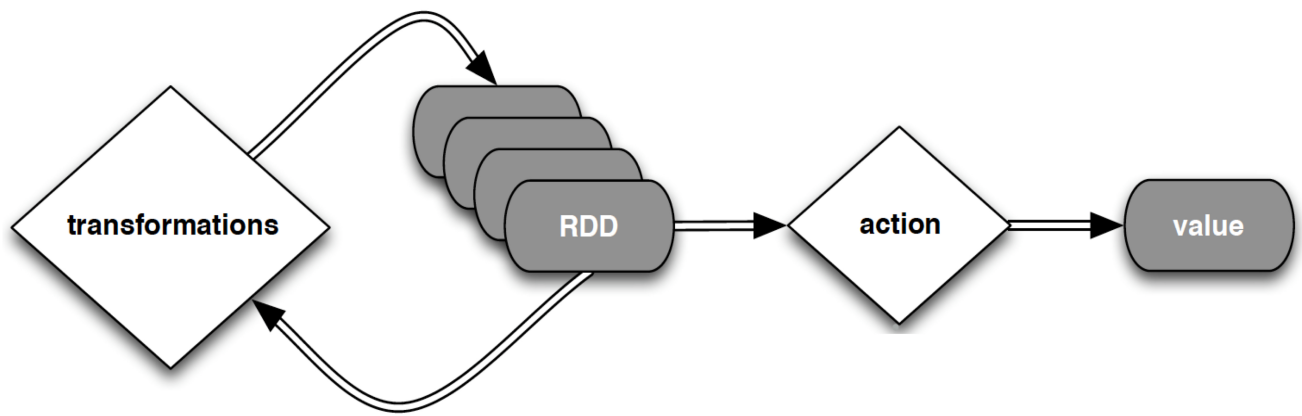
Spark programming abstraction

Write programs in terms of transformations on distributed datasets.

- Resilient Distributed Datasets (RDDs)
  - Distributed collections of objects that can be stored in memory or on disk
  - Built via parallel transformations (map, filter etc.)
  - Automatically rebuilt on failure

RDD Operations

1. Transformations
   - are lazy (not computed immediately)
   - lineage $\rightarrow$ fault-tolerance
2. Actions
   - an RDD is computed when an action is run on it



Simple Spark Apps: *WordCount*

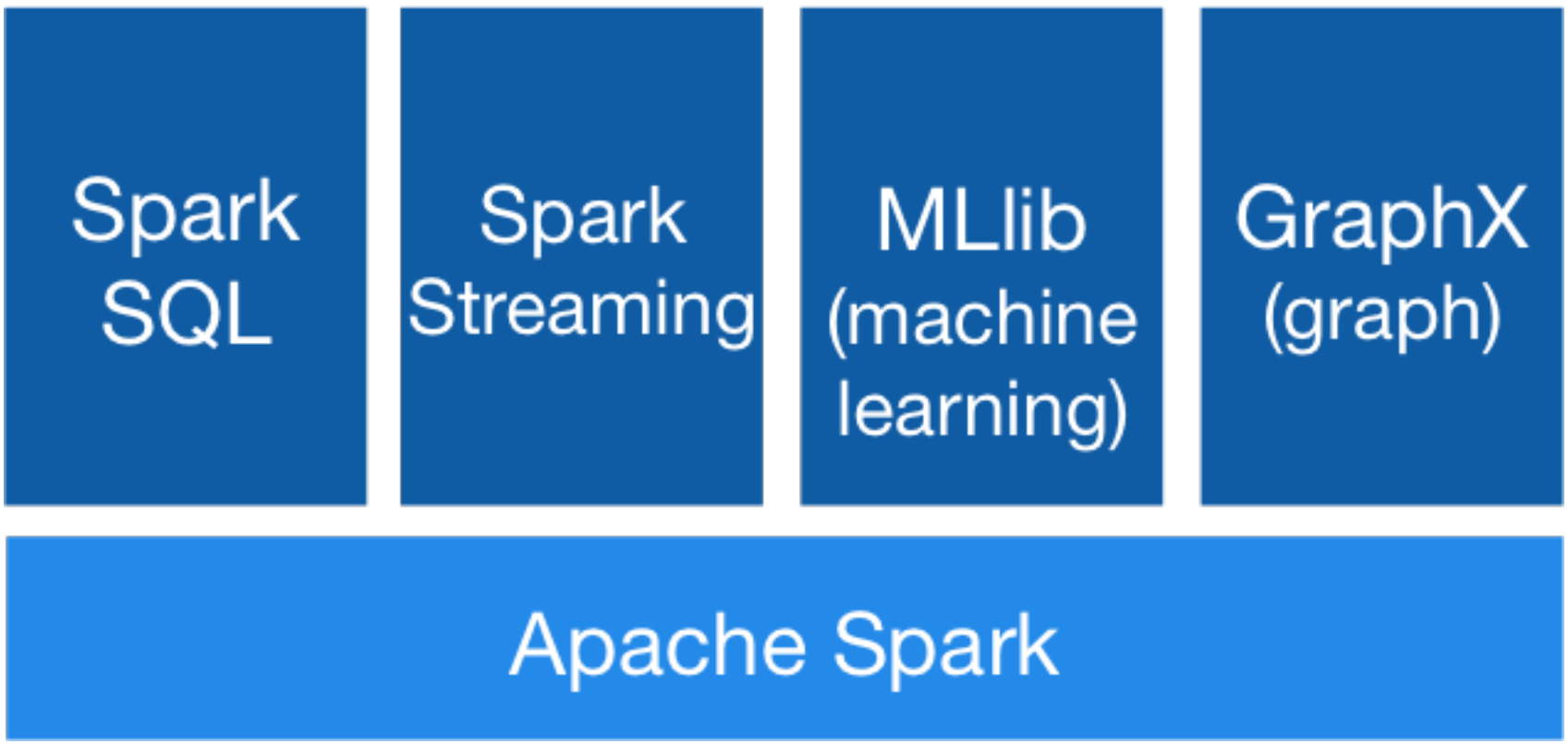
Scala:

```scala
val f = sc.textFile("README.md")
val wc = f.flatMap(l => l.split(" ")).map(word => (word, 1)).reduceByKey(_ + _)
wc.saveAsTextFile("wc_out.txt")
```

Python:

```python
from operator import add
f = sc.textFile("README.md")
wc = f.flatMap(lambda x: x.split(' ')).map(lambda x: (x, 1)).reduceByKey(add)
wc.saveAsTextFile("wc_out.txt")
```

Spark Ecosystem



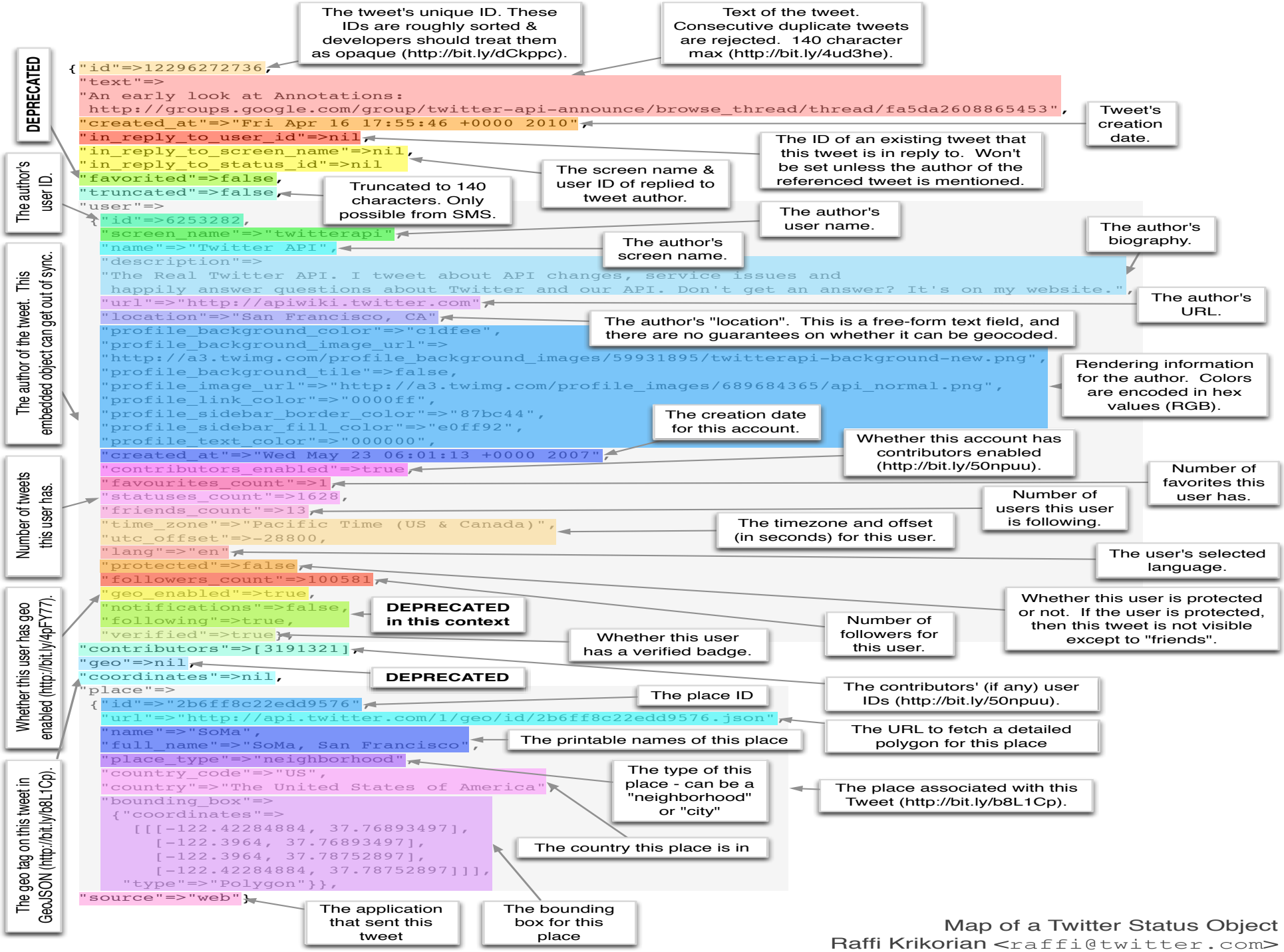
Twitter - the 21st century newspaper

Why Twitter?

- Social media/microblogging service that provides a "Timeline" of user tweets
- Real-time, up to the minute
- 313 million(ish) users
- 460,000 new accounts per day
- 1 billion tweets per week (6000 tweets per second)
- in terms of social media and information proliferation, **Twitter matters**

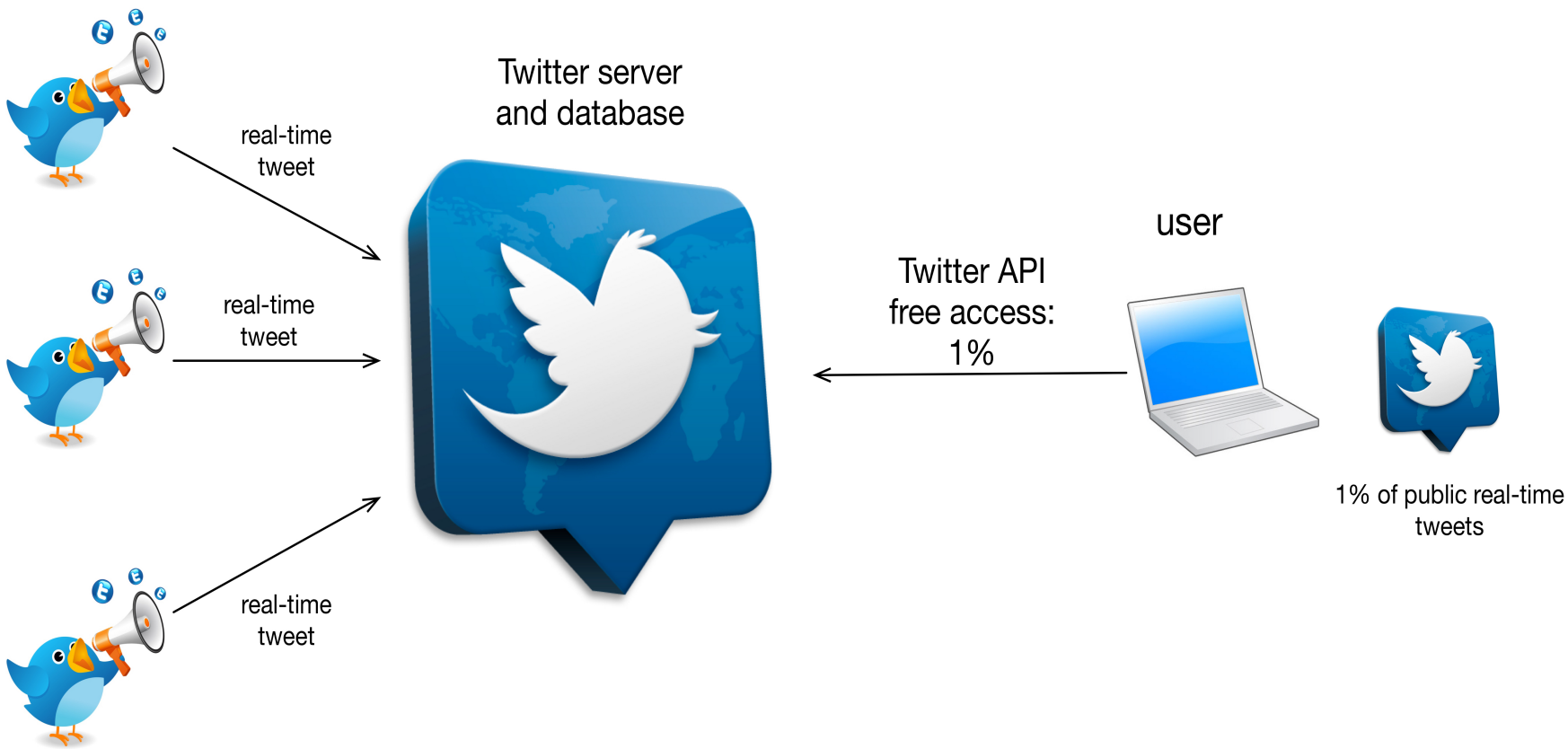
Why Twitter?

- Social media/microblogging service that provides a "Timeline" of user tweets
- Real-time, up to the minute
- 313 million(ish) users
- 460,000 new accounts per day
- 1 billion tweets per week (6000 tweets per second)
- in terms of social media and information proliferation, **Twitter matters**



Map of a Twitter Status Object
Raffi Krikorian <raffi@twitter.com>
18 April 2010

How to get Twitter data?



Twitter data acquisition

- 6000 tweets per second
- Twitter Streaming API
  - ✓ **free** access to **1%** of **public** tweets in **real-time**
  - ✓ authenticated request needs to be renewed every 15 minutes
  - ✓ users specify a set of criteria
  - ✓ queries: keywords, hashtags, user ids, and geographic bounding boxes simultaneously
  - ✓ returns tweets matching keywords
  - ✓ samples data when volume reaches 1%
- Streaming API feed $\rightarrow$ 1% $\rightarrow$ yourself
- Firehose feed $\rightarrow$ 100% $\rightarrow$ data providers
- Get the followers and friends (limited number)

Methods

ML techniques:

1. topic modelling
2. sentiment analysis

- powerful machine learning techniques
- widely wised in social media data analysis
- indicators of:
  - → most discussed topics
  - → individual/public sentiment

Term frequency TF

- $tf_{t,d}$ = number of times that t occurs in d
- *raw TF*:
  - a document with 10 occurrences of the term is more relevant than a document with 1 occurrence of the term.
  - but not 10 times more relevant.
  - relevance does not increase proportionally with term frequency.
- *log TF*

Term frequency TF

- $tf_{t,d}$ = number of times that t occurs in d
- *raw TF*:
  - a document with 10 occurrences of the term is more relevant than a document with 1 occurrence of the term.
  - but not 10 times more relevant.
  - relevance does not increase proportionally with term frequency.
- *log TF*

Term frequency TF

- $tf_{t,d}$ = number of times that t occurs in d
- *raw TF*:
  - a document with 10 occurrences of the term is more relevant than a document with 1 occurrence of the term.
  - but not 10 times more relevant.
  - relevance does not increase proportionally with term frequency.
- *log TF*

Term frequency TF

- $tf_{t,d}$ = number of times that t occurs in d
- *raw TF*:
  - a document with 10 occurrences of the term is more relevant than a document with 1 occurrence of the term.
  - but not 10 times more relevant.
  - relevance does not increase proportionally with term frequency.
- *log TF*

Term frequency TF

- $tf_{t,d}$ = number of times that t occurs in d
- *raw TF*:
  - a document with 10 occurrences of the term is more relevant than a document with 1 occurrence of the term.
  - but not 10 times more relevant.
  - relevance does not increase proportionally with term frequency.
- *log TF*

Term frequency TF

- $tf_{t,d}$ = number of times that t occurs in d
- *raw TF*:
  - a document with 10 occurrences of the term is more relevant than a document with 1 occurrence of the term.
  - but not 10 times more relevant.
  - relevance does not increase proportionally with term frequency.
- *log TF*

Term frequency TF

- $tf_{t,d}$ = number of times that t occurs in d
- *raw TF*:
  - a document with 10 occurrences of the term is more relevant than a document with 1 occurrence of the term.
  - but not 10 times more relevant.
  - relevance does not increase proportionally with term frequency.
- *log TF*

$$w_{t,d} = \begin{cases} 1 + \log_{10} \mathrm{tf}_{t,d}, & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0, & \text{otherwise} \end{cases}$$

Inverse document frequency IDF

- Rare terms are more informative than frequent terms (stop words)
- df_t is the document frequency of t: the number of documents that contain t
  - df_t is an inverse measure of the informativeness of t
  - $df_t \leq N$
- We define the *idf (inverse document frequency)* of t by

Inverse document frequency IDF

- Rare terms are more informative than frequent terms (stop words)
- df_t is the document frequency of t: the number of documents that contain t
  - df_t is an inverse measure of the informativeness of t
  - $df_t \leq N$
- We define the *idf (inverse document frequency)* of t by

Inverse document frequency IDF

- Rare terms are more informative than frequent terms (stop words)
- df_t is the document frequency of t: the number of documents that contain t
  - df_t is an inverse measure of the informativeness of t
  - $df_t \leq N$
- We define the *idf (inverse document frequency)* of t by

Inverse document frequency IDF

- Rare terms are more informative than frequent terms (stop words)
- df_t is the document frequency of t: the number of documents that contain t
  - df_t is an inverse measure of the informativeness of t
  - $df_t \leq N$
- We define the *idf (inverse document frequency)* of t by

Inverse document frequency IDF

- Rare terms are more informative than frequent terms (stop words)
- df_t is the document frequency of t: the number of documents that contain t
  - df_t is an inverse measure of the informativeness of t
  - $df_t \leq N$
- We define the *idf (inverse document frequency)* of t by

Inverse document frequency IDF

- Rare terms are more informative than frequent terms (stop words)
- df_t is the document frequency of t: the number of documents that contain t
  - df_t is an inverse measure of the informativeness of t
  - $df_t \leq N$
- We define the *idf (inverse document frequency)* of t by

$$\text{idf}_t = \log_{10}(N/\text{df}_t)$$

TF-IDF

- The *tf-idf weight* of a term is the product of its tf weight and its idf weight.

$$w_{t,d} = \log(1 + \mathrm{tf}_{t,d}) \times \log_{10}(N / \mathrm{df}_t)$$

- Best known weighting scheme in information retrieval
- Increases with the number of occurrences within a document
- Increases with the rarity of the term in the collection

Topic model

- Probabilistic model for uncovering the underlying semantic structure of a document collection based on a hierarchical Bayesian analysis of the original texts (Blei, 2003)
- **Aim**: discover patterns of word-use and connect documents that exhibit similar patterns
- **Idea**: documents are mixtures of topics and a topic is a probability distribution over words
- popular implementation: *Latent Dirichlet Allocation*

→ Mahout's implementation of LDA

- ✓ collapsed variational Bayes (CVB)
- ✓ pipeline of Hadoop jobs

Topic model

- Observed variables:
  - word distribution per document (tf-idf)
- Latent variables:
  - topic distribution per document: $P(z) = \theta^{(d)}$
  - word distribution per topic: $P(w, z) = \phi^{(z)}$
  - word-topic assignment: $P(z|w)$
- **Training**: learn latent variables on training data
- **Test**: predict topic distribution of an unseen document

Topic model

- Observed variables:
  - word distribution per document (tf-idf)
- Latent variables:
  - topic distribution per document: $P(z) = \theta^{(d)}$
  - word distribution per topic: $P(w, z) = \phi^{(z)}$
  - word-topic assignment: $P(z|w)$
- **Training**: learn latent variables on training data
- **Test**: predict topic distribution of an unseen document

Topic model

- Observed variables:
  - word distribution per document (tf-idf)
- Latent variables:
  - topic distribution per document: $P(z) = \theta^{(d)}$
  - word distribution per topic: $P(w, z) = \phi^{(z)}$
  - word-topic assignment: $P(z|w)$
- **Training**: learn latent variables on training data
- **Test**: predict topic distribution of an unseen document

Topic model

- Observed variables:
  - word distribution per document (tf-idf)
- Latent variables:
  - topic distribution per document: $P(z) = \theta^{(d)}$
  - word distribution per topic: $P(w, z) = \phi^{(z)}$
  - word-topic assignment: $P(z|w)$
- **Training**: learn latent variables on training data
- **Test**: predict topic distribution of an unseen document

Topic model

- Observed variables:
  - word distribution per document (tf-idf)
- Latent variables:
  - topic distribution per document: $P(z) = \theta^{(d)}$
  - word distribution per topic: $P(w, z) = \phi^{(z)}$
  - word-topic assignment: $P(z|w)$
- **Training**: learn latent variables on training data
- **Test**: predict topic distribution of an unseen document

Topic model

- Observed variables:
  - word distribution per document (tf-idf)
- Latent variables:
  - topic distribution per document: $P(z) = \theta^{(d)}$
  - word distribution per topic: $P(w, z) = \phi^{(z)}$
  - word-topic assignment: $P(z|w)$
- **Training**: learn latent variables on training data
- **Test**: predict topic distribution of an unseen document

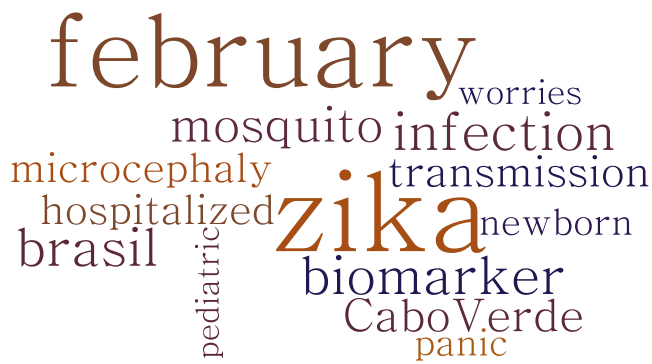
Topic modelling - example

"Arts"	"Budgets"	"Children"	"Education"
NEW	MILLION	CHILDREN	SCHOOL
FILM	TAX	WOMEN	STUDENTS
SHOW	PROGRAM	PEOPLE	SCHOOLS
MUSIC	BUDGET	CHILD	EDUCATION
MOVIE	BILLION	YEARS	TEACHERS
PLAY	FEDERAL	FAMILIES	HIGH
MUSICAL	YEAR	WORK	PUBLIC
BEST	SPENDING	PARENTS	TEACHER
ACTOR	NEW	SAYS	BENNETT
FIRST	STATE	FAMILY	MANIGAT
YORK	PLAN	WELFARE	NAMPHY
OPERA	MONEY	MEN	STATE
THEATER	PROGRAMS	PERCENT	PRESIDENT
ACTRESS	GOVERNMENT	CARE	ELEMENTARY
LOVE	CONGRESS	LIFE	HAITI

The William Randolph Hearst Foundation will give $1.25 million to Lincoln Center, Metropolitan Opera Co., New York Philharmonic and Juilliard School. "Our board felt that we had a real opportunity to make a mark on the future of the performing arts with these grants an act every bit as important as our traditional areas of support in health, medical research, education and the social services," Hearst Foundation President Randolph A. Hearst said Monday in announcing the grants. Lincoln Center's share will be $200,000 for its new building, which will house young artists and provide new public facilities. The Metropolitan Opera Co. and New York Philharmonic will receive $400,000 each. The Juilliard School, where music and the performing arts are taught, will get $250,000. The Hearst Foundation, a leading supporte

LDA on the Twitter dataset - results

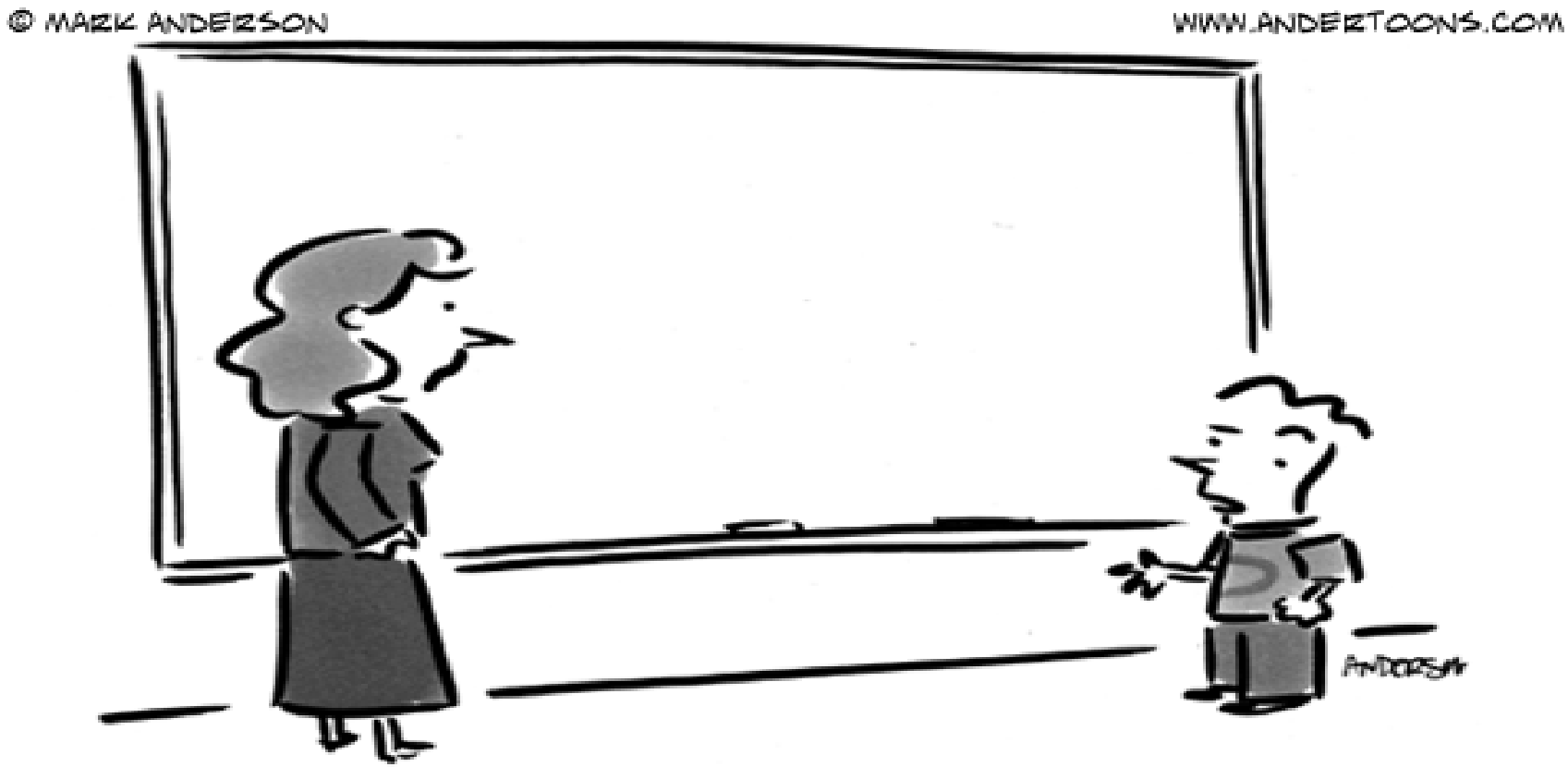
(a) Example Topic 1



(b) Example Topic 2

Figure: An illustration of topics discovered by running LDA on the Twitter dataset. Topic 1 refers to the successful vaccine trial conducted against Ebola during the outbreak in Guinea, in July 2015. The technique used in the vaccine trial is called “ring vaccination”, in which an outbreak is contained by vaccinating all suspected individuals in an area around the outbreak. Topic 2 highlights an outbreak numbering thousands of cases reported in Cabo Verde, western Africa, by the start of February 2016. Local transmission of Zika infection had been reported in more than 20 countries and territories in the Americas, and this was one of the first reportings on the African continent, outside of the range occupied by the Aedes mosquitos.

Q&A



"Before I write my name on the board, I'll need to know how you're planning to use that data."