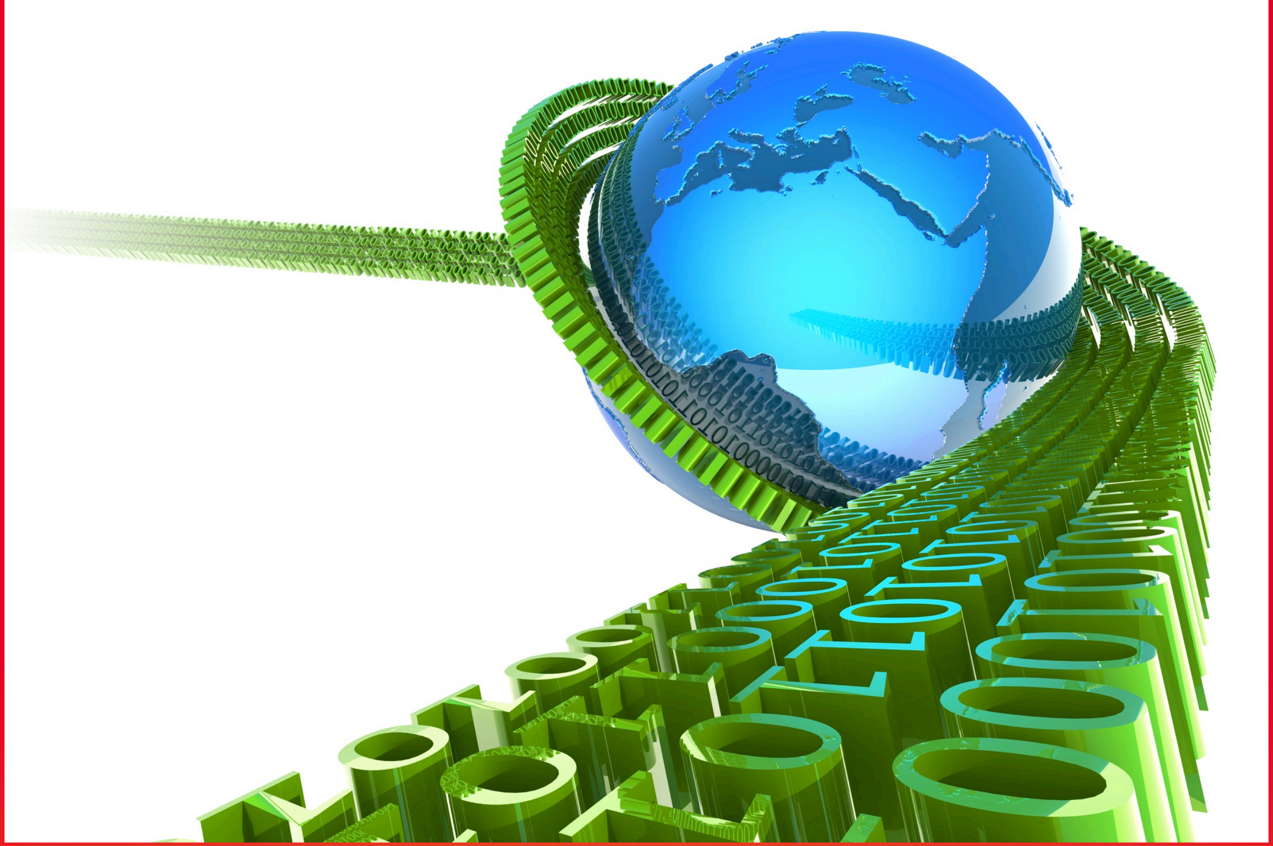
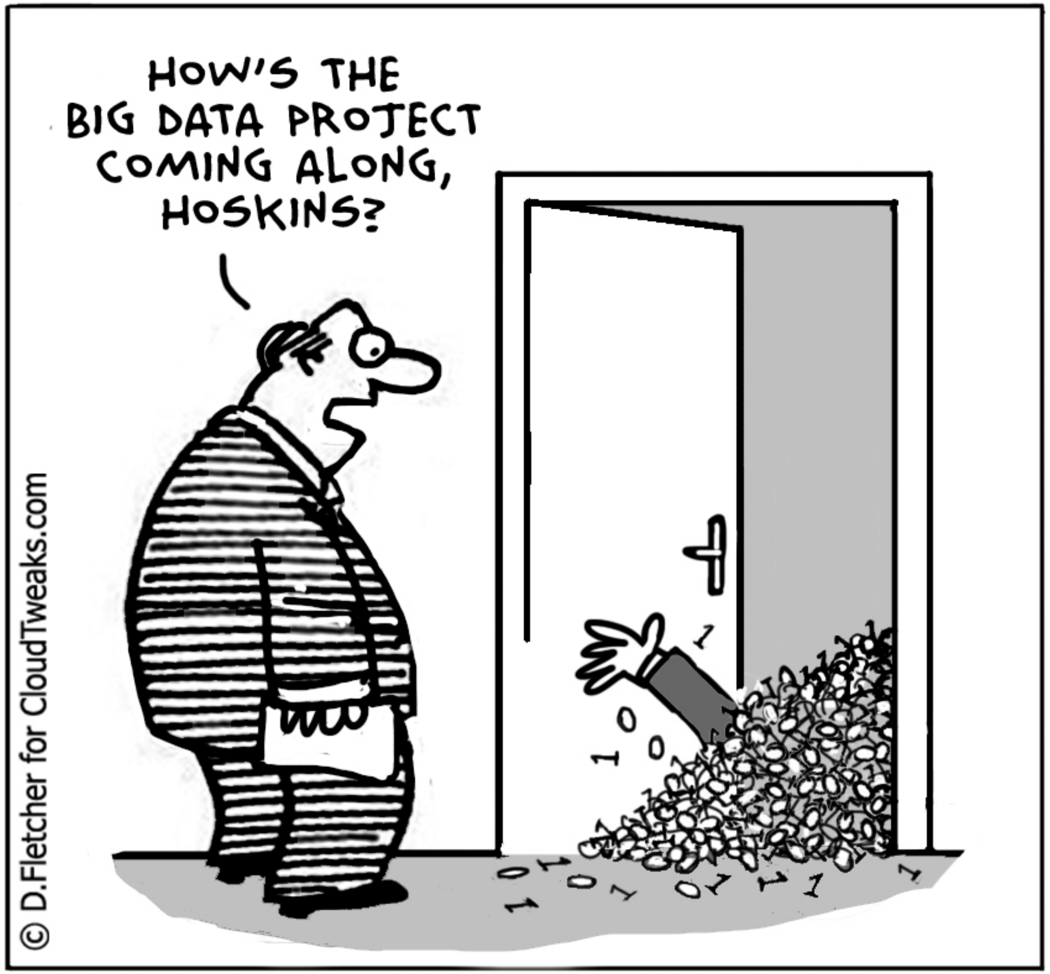
Big Data Technology



Gabriel Antoniu
KerData Project-Team, Inria, Rennes

After this talk

- Realize the potential:
  - Data vs. **Big Data**
- Understand why we need a different paradigm
- Recognize some of the main terminology
- Know the existing tools and systems
  - Hadoop
  - Spark
  - Flink/Stratosphere



Outline

- What is Big Data?
    - And where does it come from?
- Storage: SQL vs. NoSQL
- Processing
    - MapReduce
    - State of the art: Hadoop
- Future of MapReduce?
    - Spark
    - Flink/Stratosphere
- At 5:15pm: Programming models (Rosa Badia)
- *On Friday: hands-on Hadoop session (Shadi Ibrahim)*

Disclaimer

- **This is an introductory lecture!**
- If you already know about Big Data challenges, what is MapReduce and Hadoop, you may prefer to go to the beach NOW! ☺
- Rosa Badia will talk about advanced component models for managing Big Data at 5:15pm.
- If you have never worked with Hadoop, you can have a taste of it on Friday (Shadi Ibrahim's session)

Acknowledgements

- Dennis Gannon (Microsoft)
- Walfredo Cirne (Google)
- Dhruba Borthakur (Yahoo!)
- María S. Pérez (UPM)
- Michael Franklin (UC Berkeley)
- Kostas Tsoumas (TU Berlin)
- Alexandru Costan (INSA Rennes – Inria)
- Shadi Ibrahim (Inria)

1

What is Big Data?

What is Big Data?

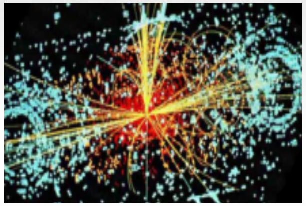
"Big Data refers to data sets whose size is beyond the ability of typical database software tools to capture, store, manage and analyze." *(McKinsey Global Institute)*

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

Context: the Data Deluge

Experiments

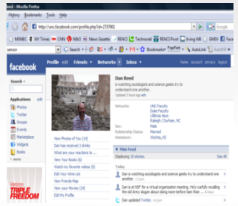
Simulations



Archives

Literature

Consumer



The Challenge:
Enable Discovery

Deliver the capability to mine, search and analyze this data in near real time

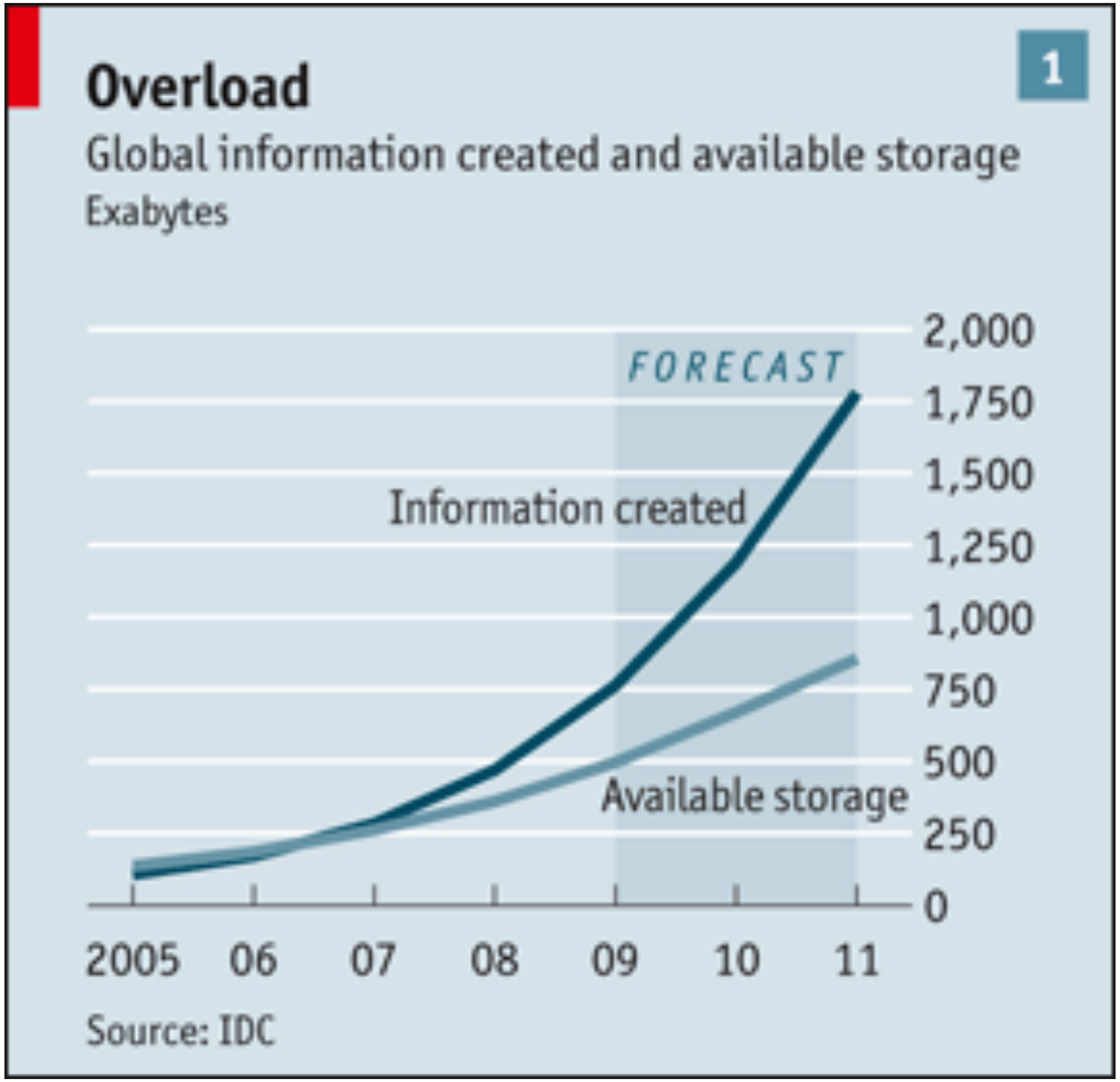
Petabytes Doubling & Doubling

The Response

Science itself is evolving

Credits: Microsoft

The Data Deluge



How *Big* is Big Data ?

Eric Schmidt: “Every 2 days we create as much information as we did up to 2003.” (2011)

We created 5 billion Gigabytes (Exabytes) of data.

In 2013, the same amount of data is created every 10 minutes.

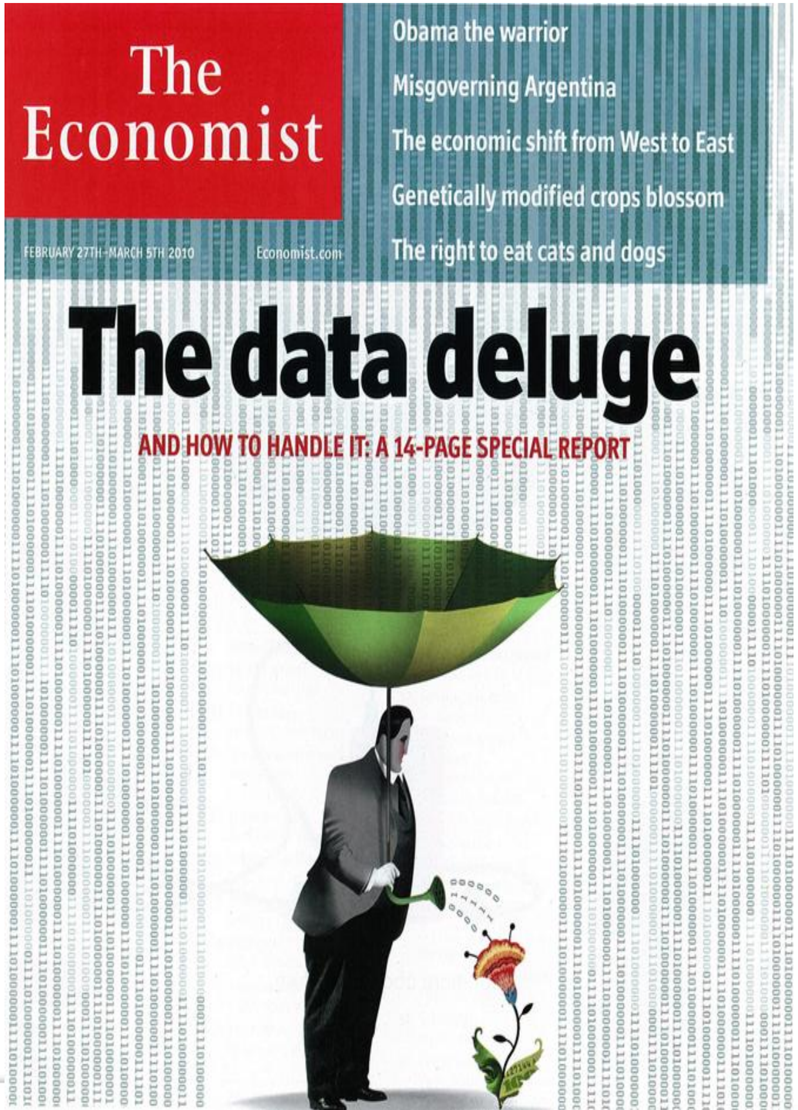
Big Data Units

Unit	Size	What it means
Bit (b)	1 or 0	Short for "binary digit", after the binary code (1 or 0) computers use to store and process data
Byte (B)	8 bits	Enough information to create an English letter or number in computer code. It is the basic unit of computing
Kilobyte (KB)	1,000, or 2^{10}, bytes	From "thousand" in Greek. One page of typed text is 2KB
Megabyte (MB)	1,000KB; 2^{20} bytes	From "large" in Greek. The complete works of Shakespeare total 5MB. A typical pop song is about 4MB
Gigabyte (GB)	1,000MB; 2^{30} bytes	From "giant" in Greek. A two-hour film can be compressed into 1-2GB
Terabyte (TB)	1,000GB; 2^{40} bytes	From "monster" in Greek. All the catalogued books in America's Library of Congress total 15TB
Petabyte (PB)	1,000TB; 2^{50} bytes	All letters delivered by America's postal service this year will amount to around 5PB. Google processes around 1PB every hour
Exabyte (EB)	1,000PB; 2^{60} bytes	Equivalent to 10 billion copies of *The Economist*
Zettabyte (ZB)	1,000EB; 2^{70} bytes	The total amount of information in existence this year is forecast to be around 1.2ZB
Yottabyte (YB)	1,000ZB; 2^{80} bytes	Currently too big to imagine

The prefixes are set by an intergovernmental group, the International Bureau of Weights and Measures. Yotta and Zetta were added in 1991; terms for larger amounts have yet to be established.

Source: *The Economist*

Big Picture of Big Data



In 2010 The digital Universe was

1.2 ZettaBytes

In a decade the Digital Universe will be

35 ZettaByte

Big Picture of Big Data

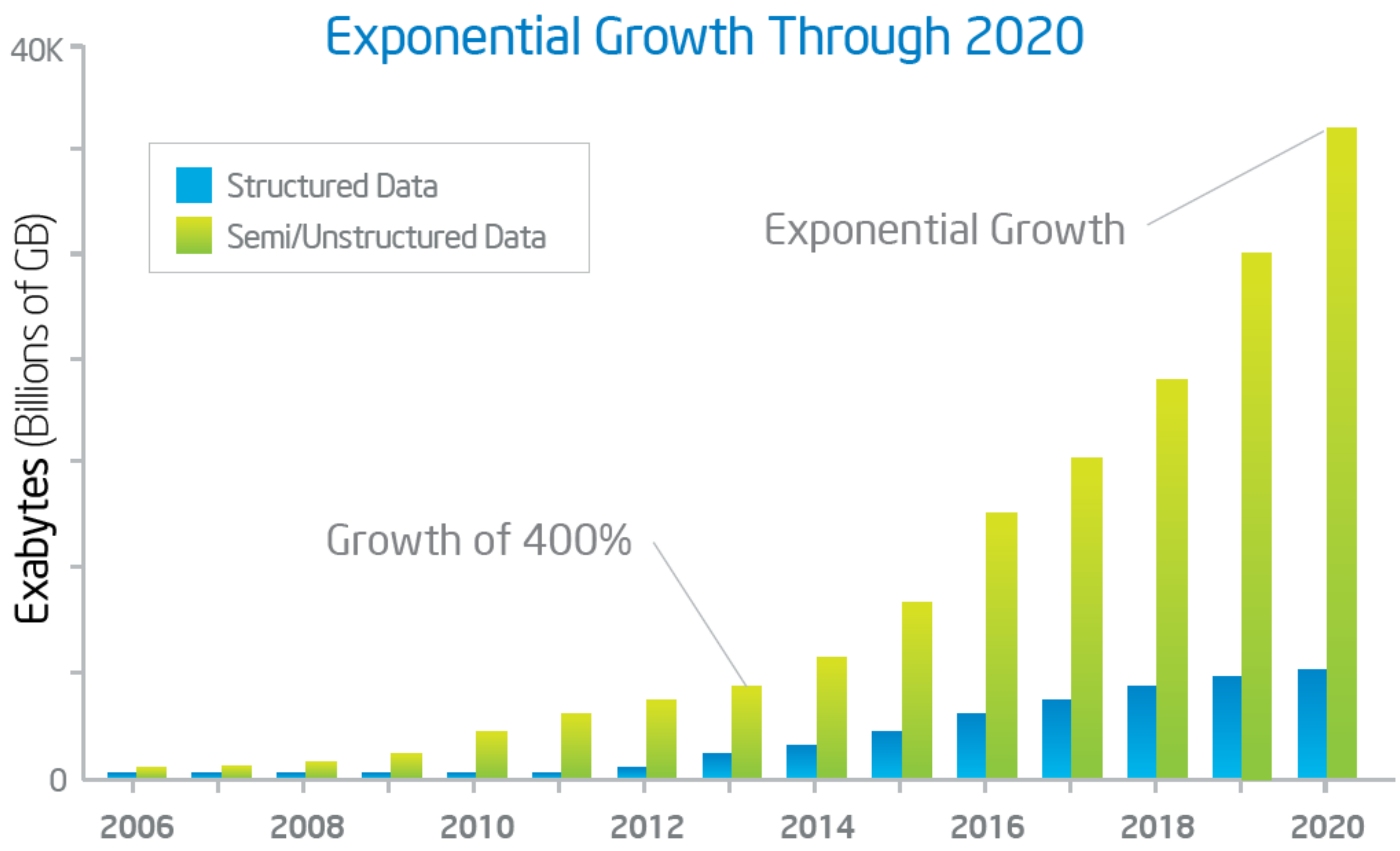
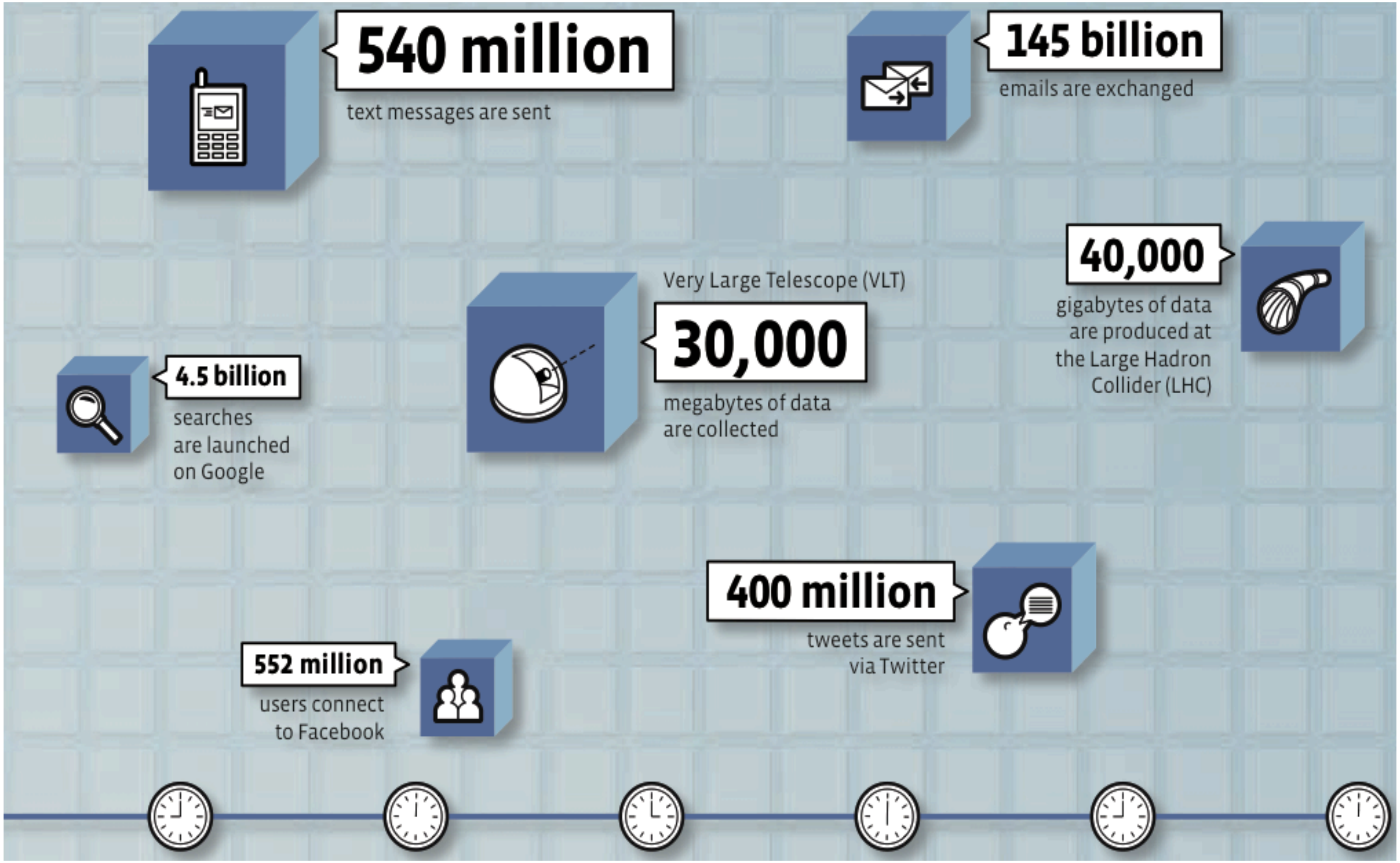


Figure 1. Current and forecasted growth of big data. Source: Philippe Botteri of Accel Partners, Feb. 2013.

What Are the Sources of Big Data?

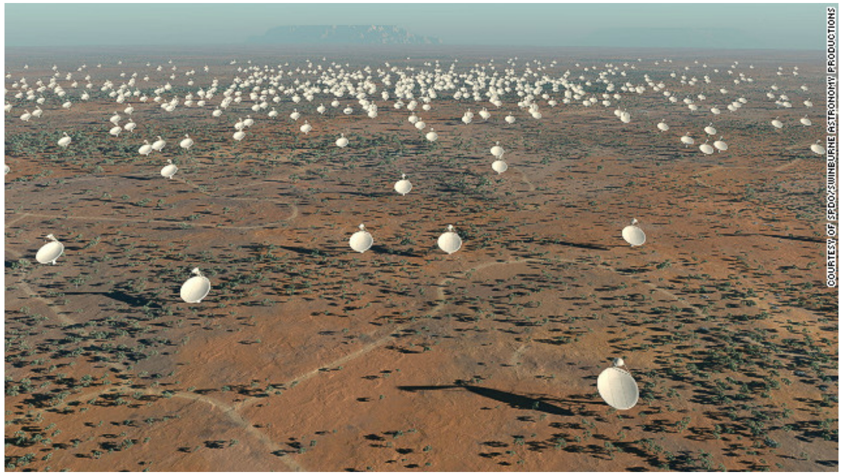
How Much Data Are We Producing in a Day ?

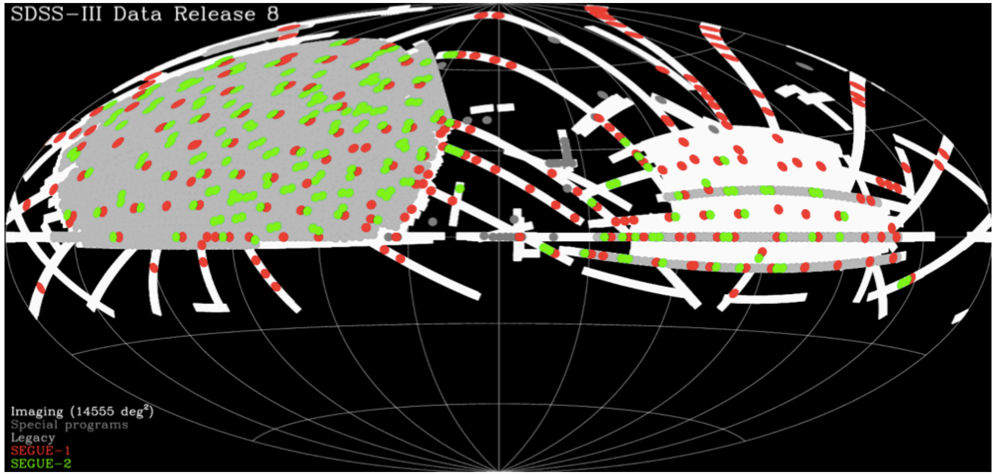


Source: CNRS Magazine 2013

Scientific Applications

Astronomical instruments

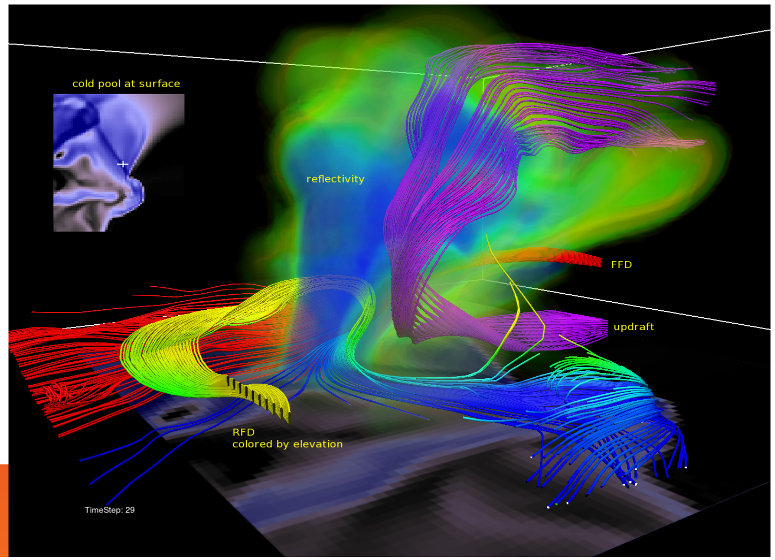




*SQUARE KILOMETRE ARRAY (SKA): World's largest radio telescope will collect **1 PB** a day ~ **400 PB** a year*

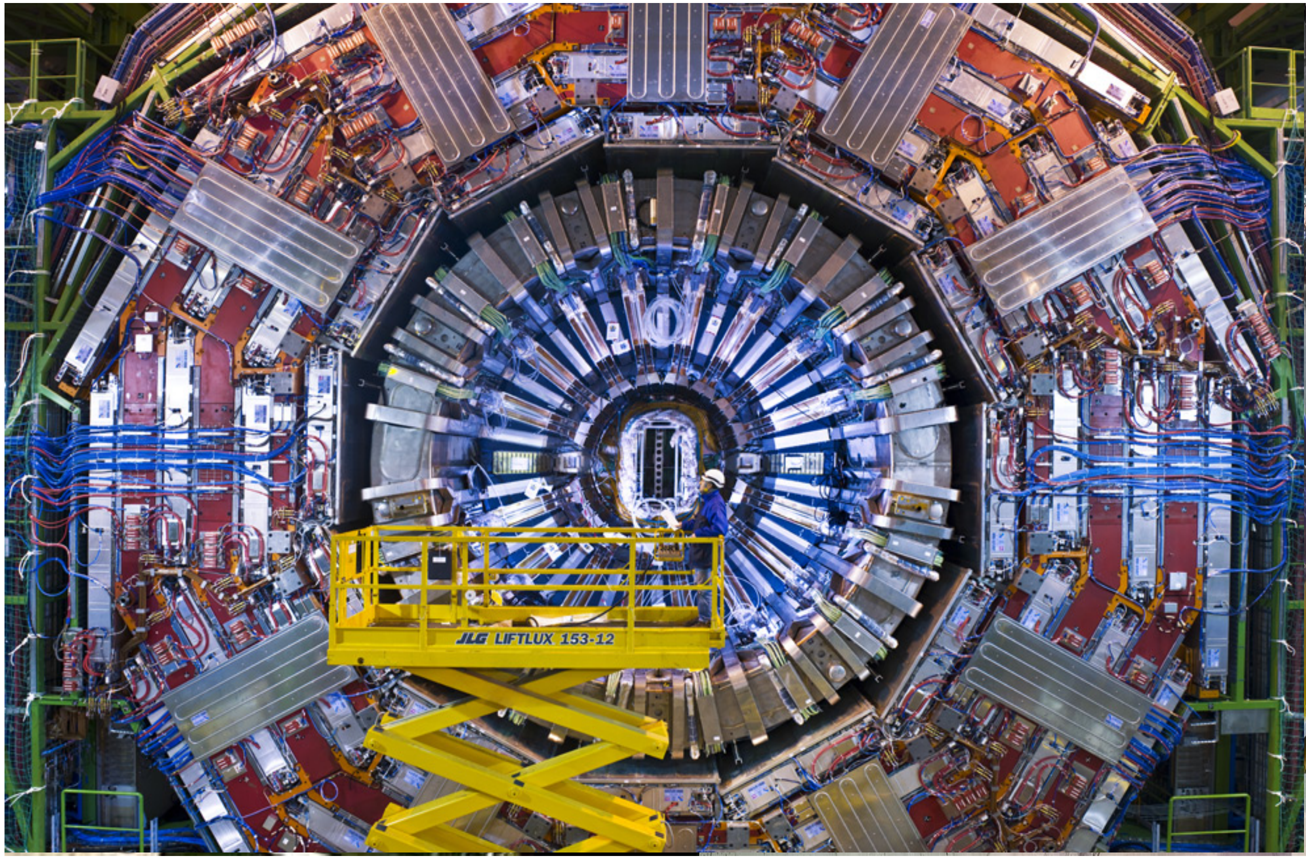
SLOAN DIGITAL SKY SURVEY (SDSS): 35% of the sky mapped, 500 millions objects classified, 50 TB of data

- Climate Simulations
  - The NASA Center for Climate Simulation (NCCS) stores 32 petabytes of climate observations and simulations on the Discover supercomputing cluster.
- Genome sequencers in biology
  - National Center for Biotechnology Information (NCBI) already house petabytes of data, and biologists around the world are churning out 15 petabases (a base is a letter of DNA) of sequence per year.



Scientific Applications

Large Hadron Collider



- 15 PB of data generated annually mostly stored in Oracle databases (SQL)

Internet Proprietary Data

- Social networks
  - Facebook has **2.5 PB** of user data + **100 PB** photos + 15 TB messages / day (2009)
  - Twitter generates approximately **12 TB** of data per day
- Web Data
  - Google processes **20 PB** a day (2008)
  - eBay has **6.5 PB** of user data + 50 TB/day (2009)

ebay

Industry

- Sensor Networks
- A single airplane engine generates more than **10 TB** of data every 30 minutes.

- Business & Commerce
- New York Stock Exchange **1TB** of data everyday
- Walmart's customer transactions feed a database of about **2.5 PB** of customer data.

Common Features

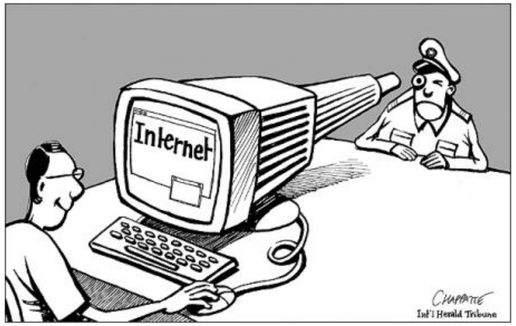
- These are typically unstructured data
- Produced in real-time
- Arrive in streams or batches from geographically distributed sources
- Have metadata (localization, day, hour, etc.)
- Hetereogenous sources (mobile phones, sensorsors, tablets, PCs, clusters)
- Arrive in disorder and unpredictably

Big Data Challenges: The Three Vs

There Are More Vs: Veracity and Value!

Data are ONLY as useful as the decisions they enable

Big Data and Privacy



What is Big Data Used For?

- Harnessing scientific discoveries
- Initiating early warning of natural disasters:
  - floods, volcanic eruptions, and earthquakes
- Reports
  - track business processes, transactions
  - fraud detection

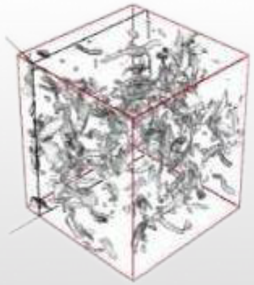
What is Big Data Used For?

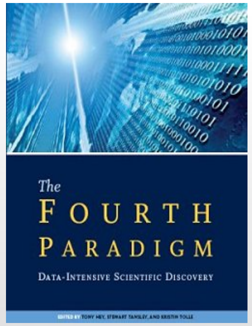
- Diagnosis
  - Why is user engagement dropping?
  - Why is the system slow?
  - Prevent failures
  - Make predictions
  - Detect spam, worms, viruses, DDoS attacks

- Decisions
  - Personalized medical treatment
  - Decide what ads to show

The Data Science:
The 4th Paradigm for Scientific Discovery

$$\left(\frac{\dot{a}}{a}\right)^2 = \frac{4\pi G\rho}{3} - K\frac{c^2}{a^2}$$





Experimental

Thousand years ago

Description of natural phenomena

Theoretical

Last few hundred years

Newton's laws, Maxwell's equations…

Computational

Last few decades

Simulation of complex phenomena

The Fourth Paradigm

Today and the Future

Unify theory, experiment and simulation with large multidisciplinary Data

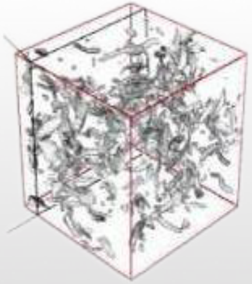
Using data exploration and data mining (from instruments, sensors, humans…)

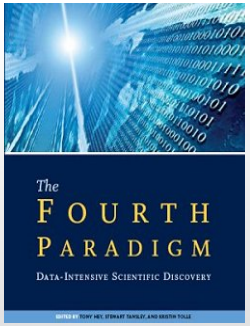
Distributed Communities

Crédits: Dennis Gannon

The Data Science:
The 4th Paradigm for Scientific Discovery

$$\left(\frac{\dot{a}}{a}\right)^2 = \frac{4\pi G\rho}{3} - K\frac{c^2}{a^2}$$





Office of Science and Technology Policy
Executive Office of the President
New Executive Office Building
Washington, DC 20502

FOR IMMEDIATE RELEASE
March 29, 2012

Contact: Rick Weiss 202 456-6037 rweiss@ostp.eop.gov
Lisa-Joy Zgorski 703 292-8311 lisajoy@nsf.gov

OBAMA ADMINISTRATION UNVEILS "BIG DATA" INITIATIVE:
ANNOUNCES $200 MILLION IN NEW R&D INVESTMENTS

The Fourth Paradigm

Today and the Future

Unify theory, experiment and simulation with large multidisciplinary Data

Using data exploration and data mining (from instruments, sensors, humans…)

Distributed Communities

Crédits: Dennis Gannon

Big Data Science: The art of understanding huge volumes of data

- Data Science is not just data analysis.

- Four main topics:
- **Data architecture:** how the data would need to be routed and organized to support the analysis, visualization and presentation of the data
- **Data acquisition:** how the data are collected, and, importantly, how the data are represented prior to analysis and presentation
- **Data analysis:** involves many technical, mathematical, and statistical aspects; still, the results have to be effectively communicated to the data user.
- **Data archiving:** preservation of collected data in a form that makes it highly reusable (data curation)

What Does a Data Scientist Do?

- Amazon’s product recommendation systems
- Google's advertisement valuation systems
- Linkedin's contact recommendation system
- Twitter's trending topics
- Walmart's consumer demand projection systems.

Allowed to make mistakes at some non-negotiable rate.
Not concerned with cause. Successful when the useful correlations are found.

Data Scientist Skills

- Evolution from the data analyst role:
- • Computer science, software engineering methodologies, modeling, statistics, analytics, visualization, databases, machine learning, data mining, big data and maths.
- • Business skills: Influence in making decisions in a business environment
- • The data scientist guides a data science project

Engineer

- collect & scrub disparate data sources manage a large computing cluster

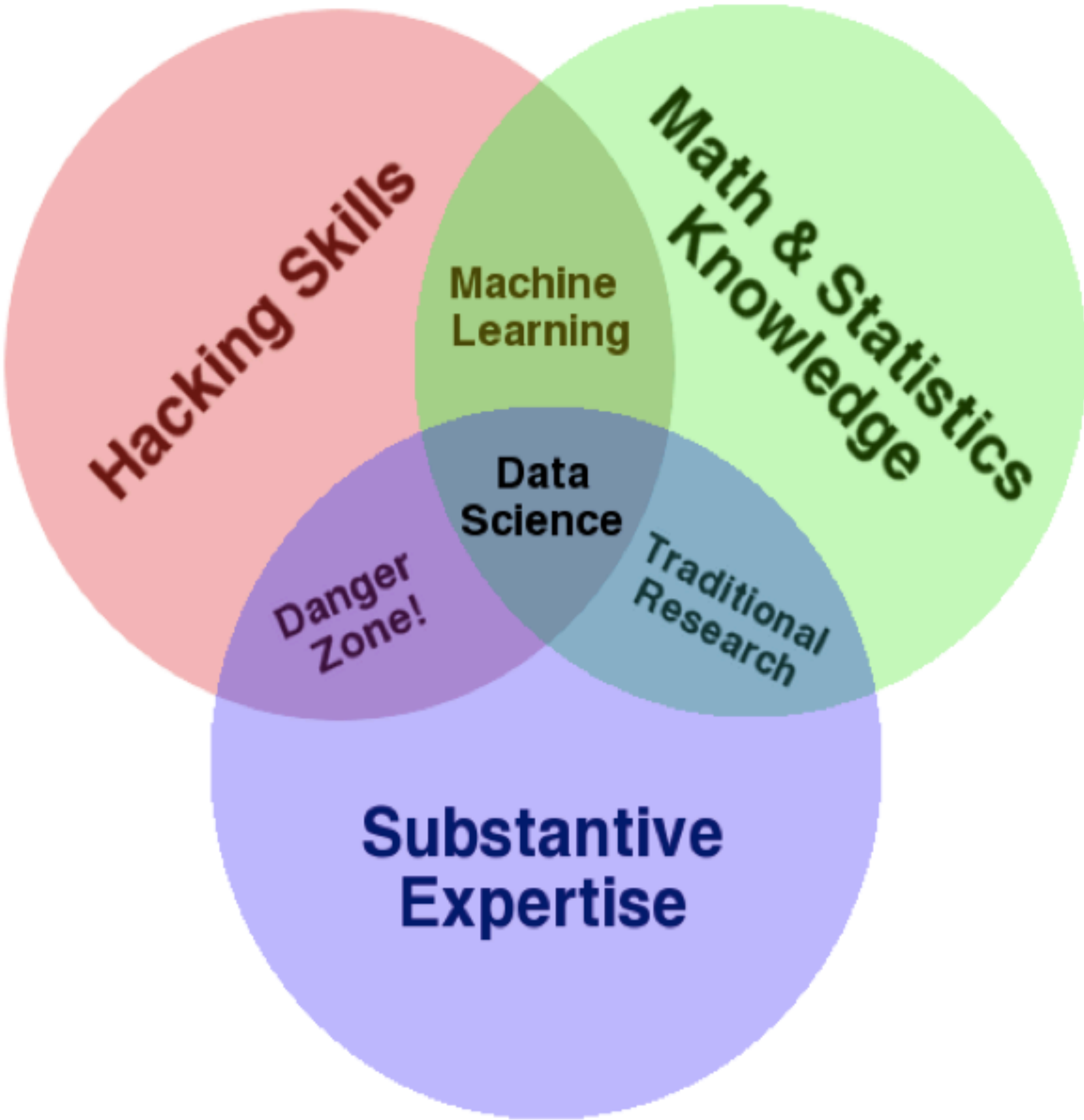
Mathematician

- machine learning statistics

Artist

- visualize data beautifully, tell a convincing story

Data Science Venn Diagram



Data Scientist

- “I keep saying the sexy job in the next ten years will be statisticians. The ability to take data - to be able to understand it, to process it, to extract value from it, to visualize it, to communicate it.”

Hal Varian, Google’s chief economist

What is Needed?

- **Computation/storage power**
  - **Cloud computing:** allows users to lease computing and storage resources in a Pay-As-You-Go manner (details in CM2)
- **Programming model**
  - **MapReduce:** Simple yet scalable model (details in CM3 and CM4)

2

Storage: SQL vs. NoSQL

Relational databases

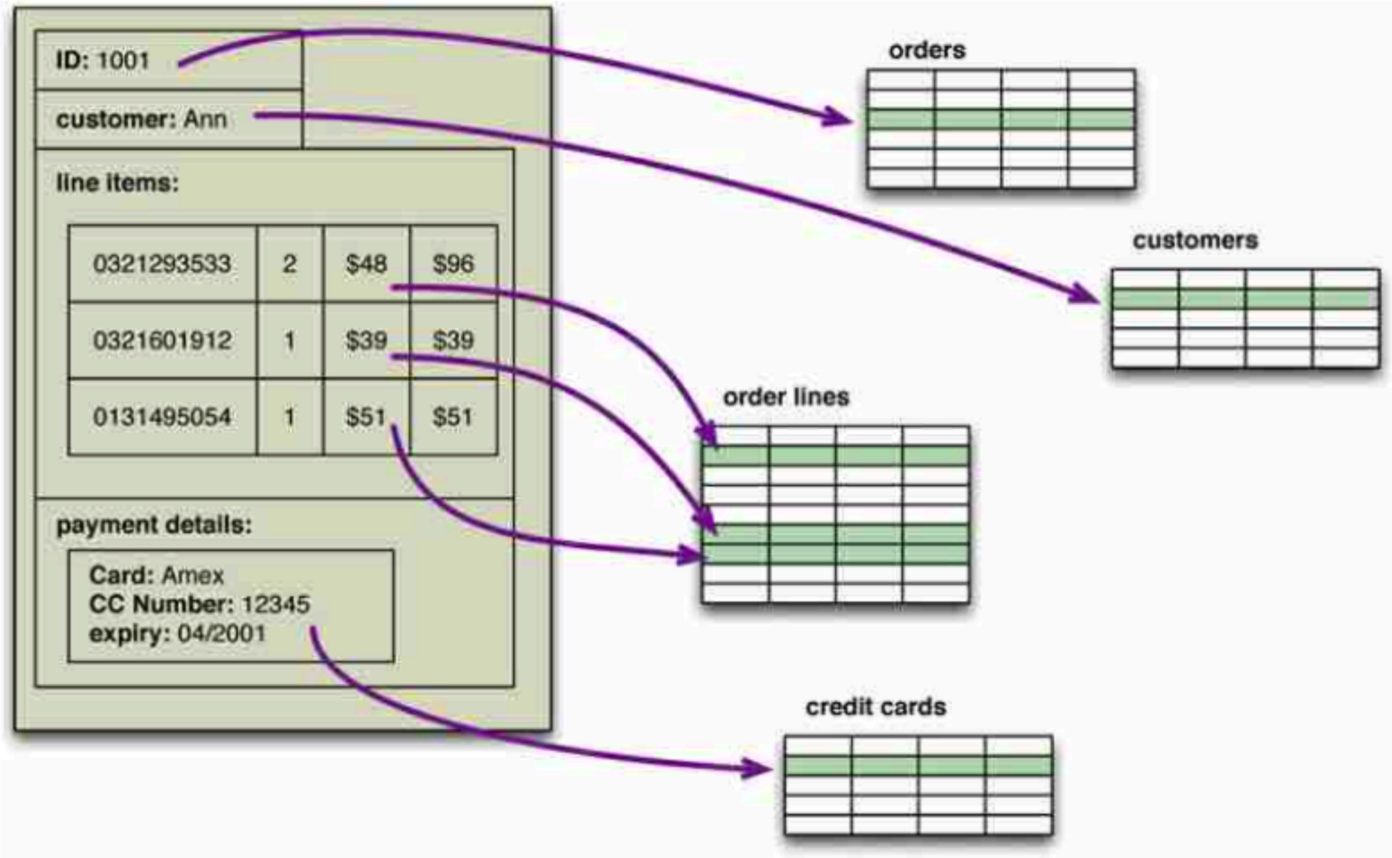
- Dominant model for the last 30 years
- Standard, easy-to-use, powerful query language SQL:
  - Declarative: Users state what they and the database internally assembles an algorithm and extracts the requested results
- Reliability and strong consistency in the presence of failures and concurrent access
- Support for transactions (ACID properties)
- Orthogonal to data representation and storage

Weaknesses

- Relational databases are not designed to run on multiple nodes (clusters)
- Favor vertical scaling
- Cannot cope with large volumes of data and operations (e.g., Big Data applications)

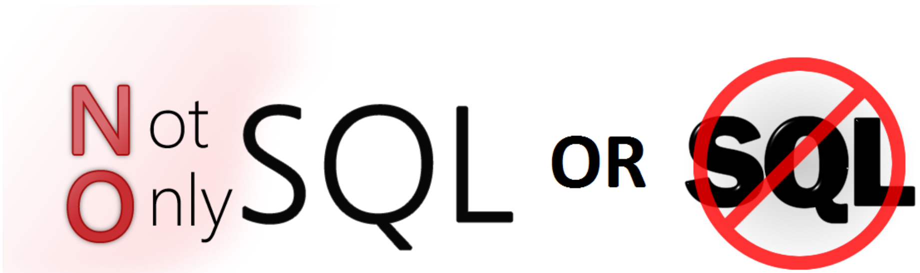
Weaknesses

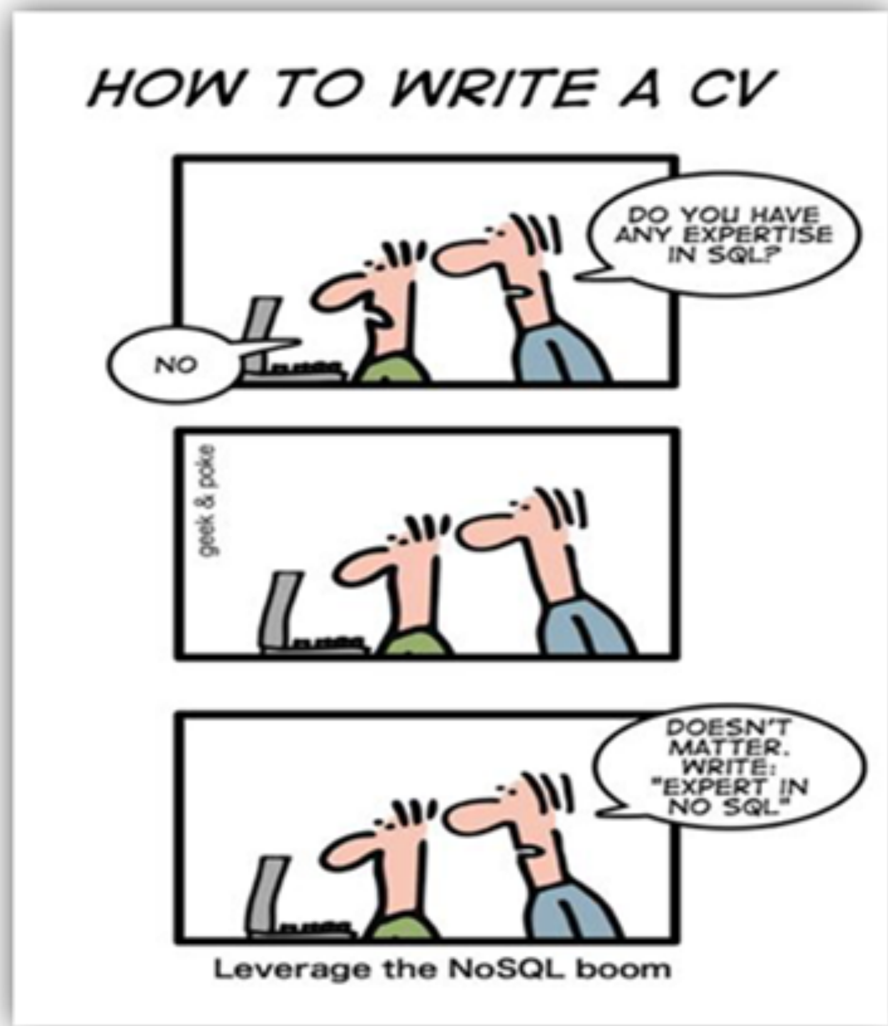
- Mapping objects to tables is notoriously difficult (impedance mismatch)



NoSQL

- Practically, anything that deviates from traditional relational database systems (RDBMSs)
- Running well on clusters
- Not needing a schema (schema-free)
- Typically, relaxing consistency

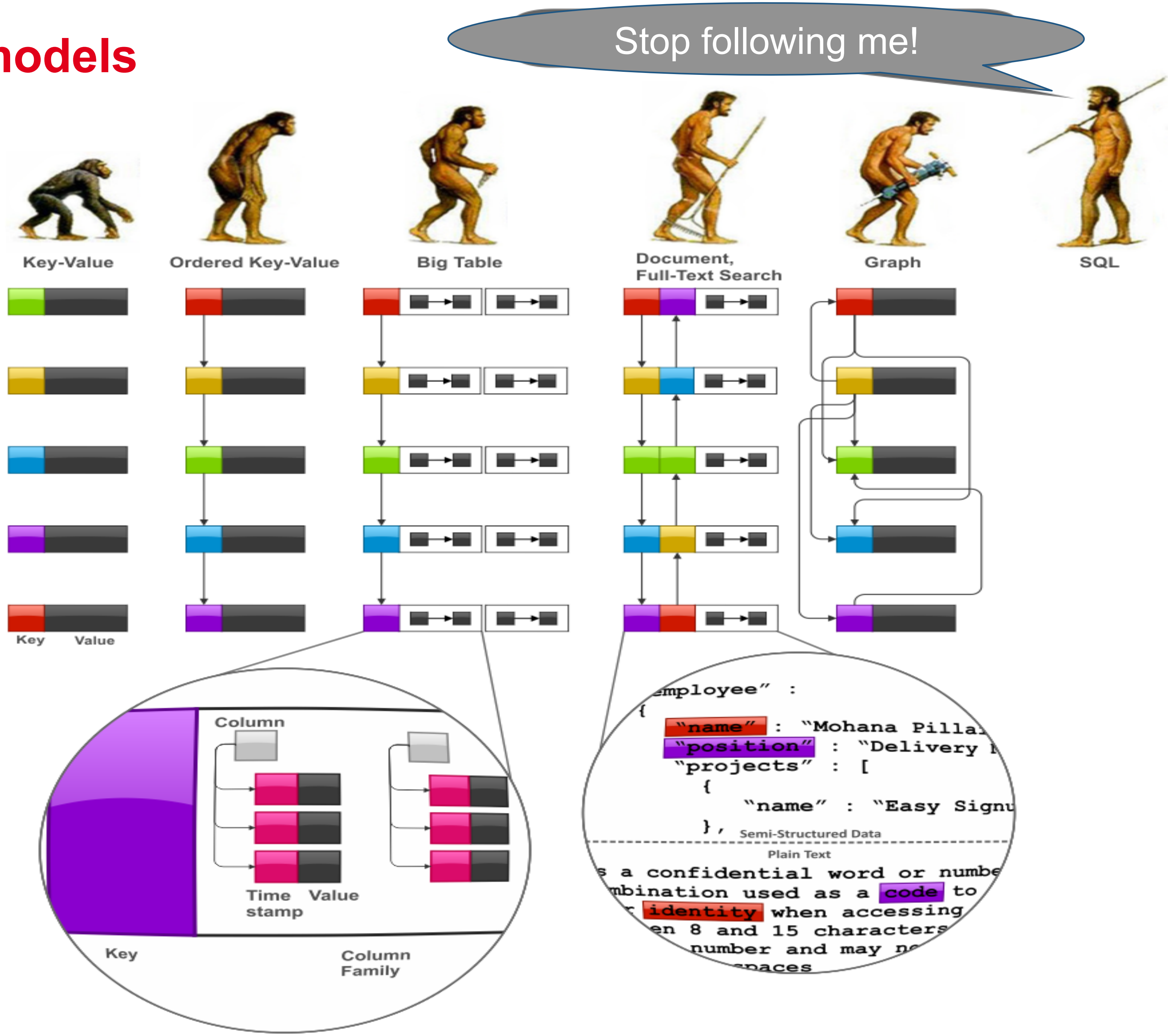




Data models

- Key-value
  - Simple hash table where all access is done via a key
  - *Redis, Riak, Memcached*
- Document
  - Main concept is document
  - Self-describing, hierarchical data structures
  - JSON, BSON, XML, etc.
  - *MongoDB, Couchbase, Terrastore, Lotus Notes*
- Column family
  - Ordered collection of rows, each of which is an ordered collection of columns
  - *Cassandra, HBase, SimpleDB*
- Graph
  - Declarative, domain-specific query languages
  - *Neo4j, Infinite Graph, FlockDB*

Data models



Stop following me!
Key-Value
Ordered Key-Value
Big Table
Document,
Full-Text Search
Graph
SQL
Key
Value
Column
Time
stamp
Value
Key
Column
Family
Semi-Structured Data
Plain Text

3

Processing: MapReduce and Hadoop

MapReduce – Motivation

- Introduced by Google in 2004
- Big Data @ Google:
- 20+ billion web pages x 20KB = 400+ TB
- One computer can read 30-35 MB/sec from disk
  - ~4 months to read the web
  - ~1,000 hard drives just to store the web
- Even more Time/HDD, to do something with the data (e.g., data processing)

Solution

Spread the work over many machines

Good news: “easy” parallelisation

- Reading the web with 1000 machines ⇒ less than 3 hours

Bad news: programming work

- Communication and coordination
- Debugging
- Fault tolerance
- Management and monitoring
- Optimization

Worse news: repeat for every problem you want to solve

And the Problem Size is Ever Growing…

- More users, happier users : more data
- Bigger web, mailbox, blog, etc.: better results
- Find the right information, and find it faster!

Conclusion:
Infrastructure is a Real Challenge

At Google's scale, building and running a computing infrastructure that is efficient, cost-effective, and easy-to-use is one of the most challenging technical points!

Typical Computer

Multicore machine

- ~ 1-2 TB of disk
- ~ 4GB-16GB of RAM

Typical machine runs:

- Google File System (GFS)
- Scheduler daemon for starting user tasks
- One or many user tasks

Tens of thousands of such machines

Problem : *What programming model to use as a basis for scalable parallel processing ?*

What Is Needed?

A simple programming model that applies to many data-intensive computing problems

Approach: hide messy details in a runtime library:

- Automatic parallelization
- Load balancing
- Network and disk transfer optimization
- Handling of machine failures
- Robustness
- Improvements to core library benefit all users of library!

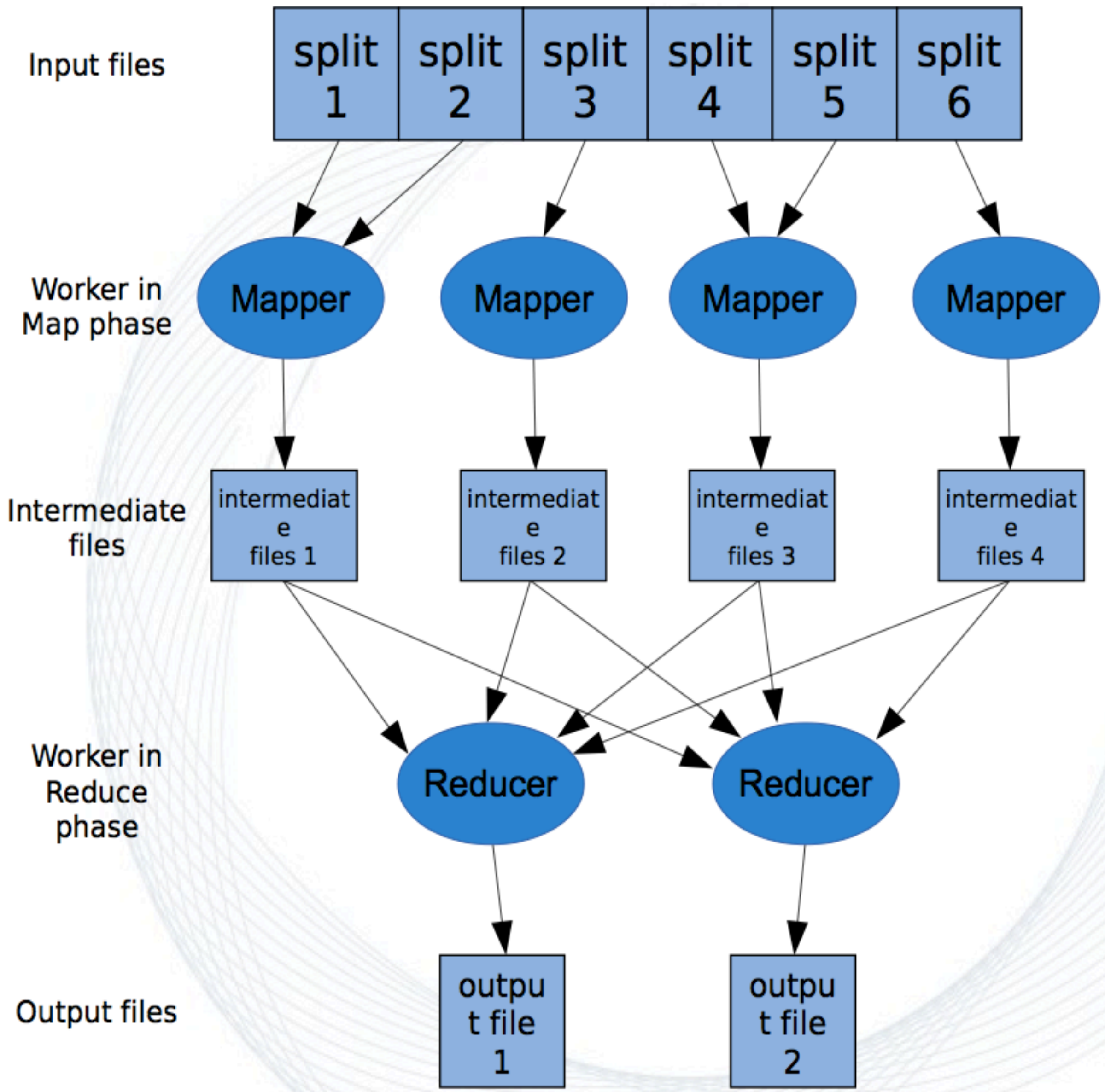
Such a Model is… MapReduce!

Typical problem solved by MapReduce

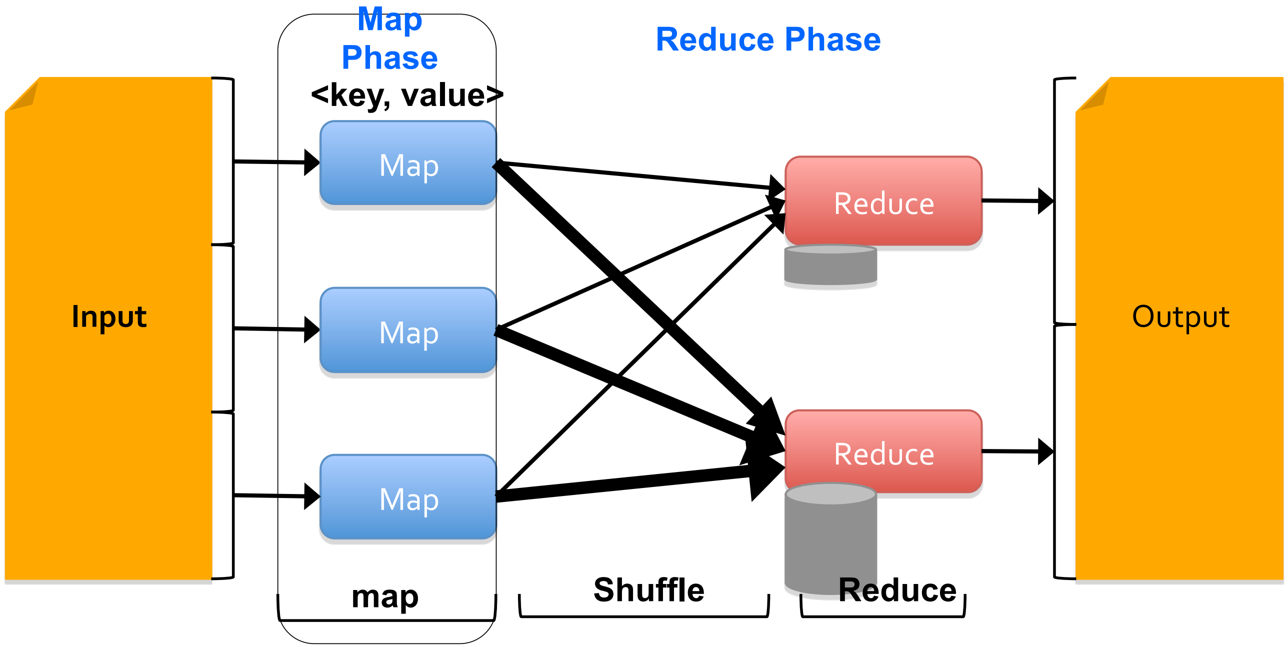
- Read a lot of data
- Map: extract something you care about from each record
- Shuffle and Sort
- Reduce: aggregate, summarize, filter, or transform
- Write the results

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

MapReduce at a Glance



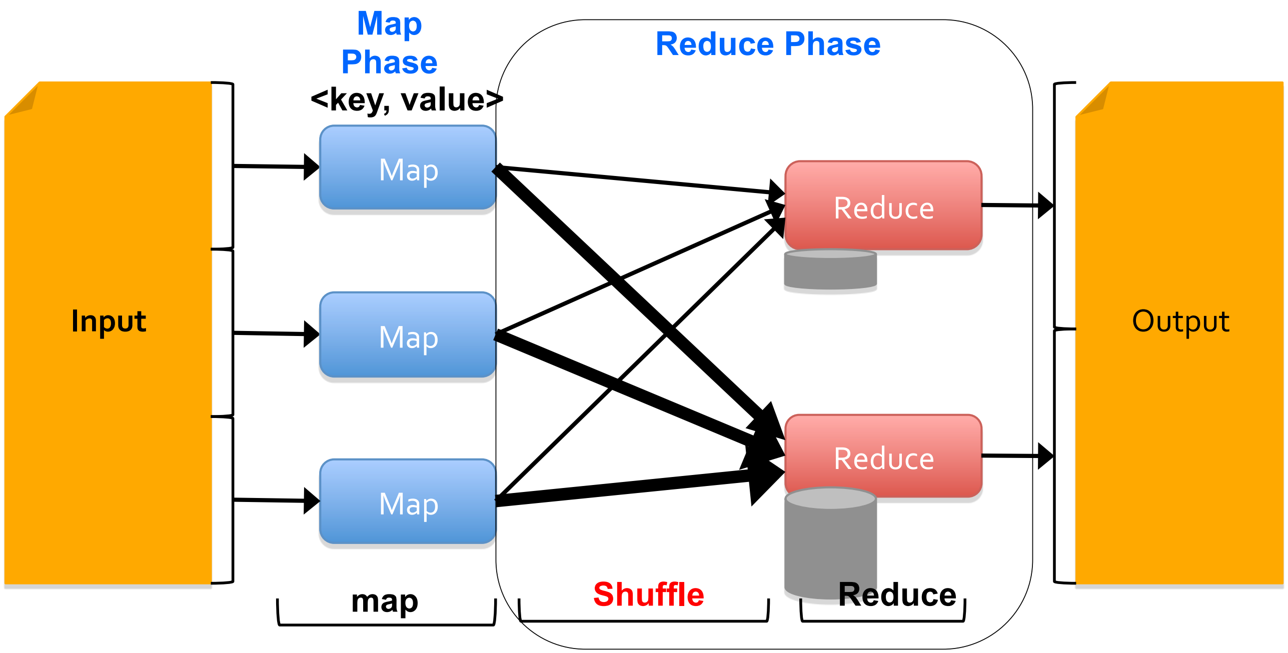
More Specifically…



map(k, v) → <k', v'>*

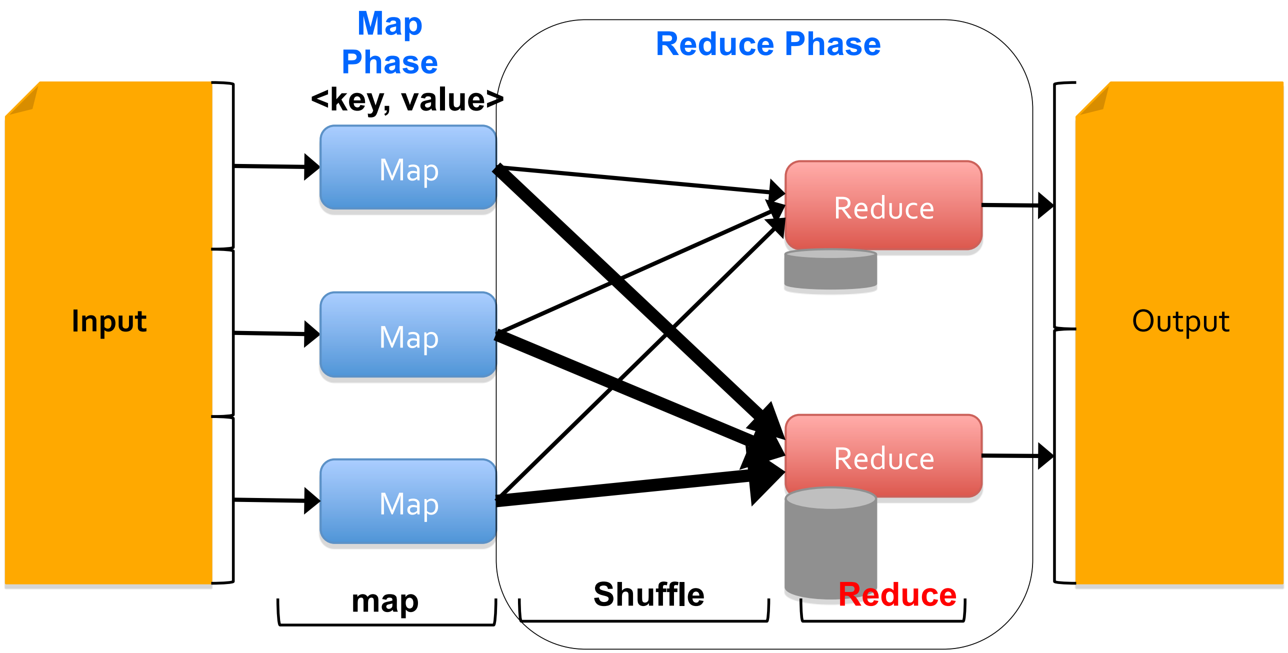
Records from the data source (lines out of files, rows of a database, etc) are fed into the map function as key*value pairs: e.g., (filename, line)

More Specifically…



After the map phase is over, all the intermediate values for a given output key are combined together into a list

More Specifically…



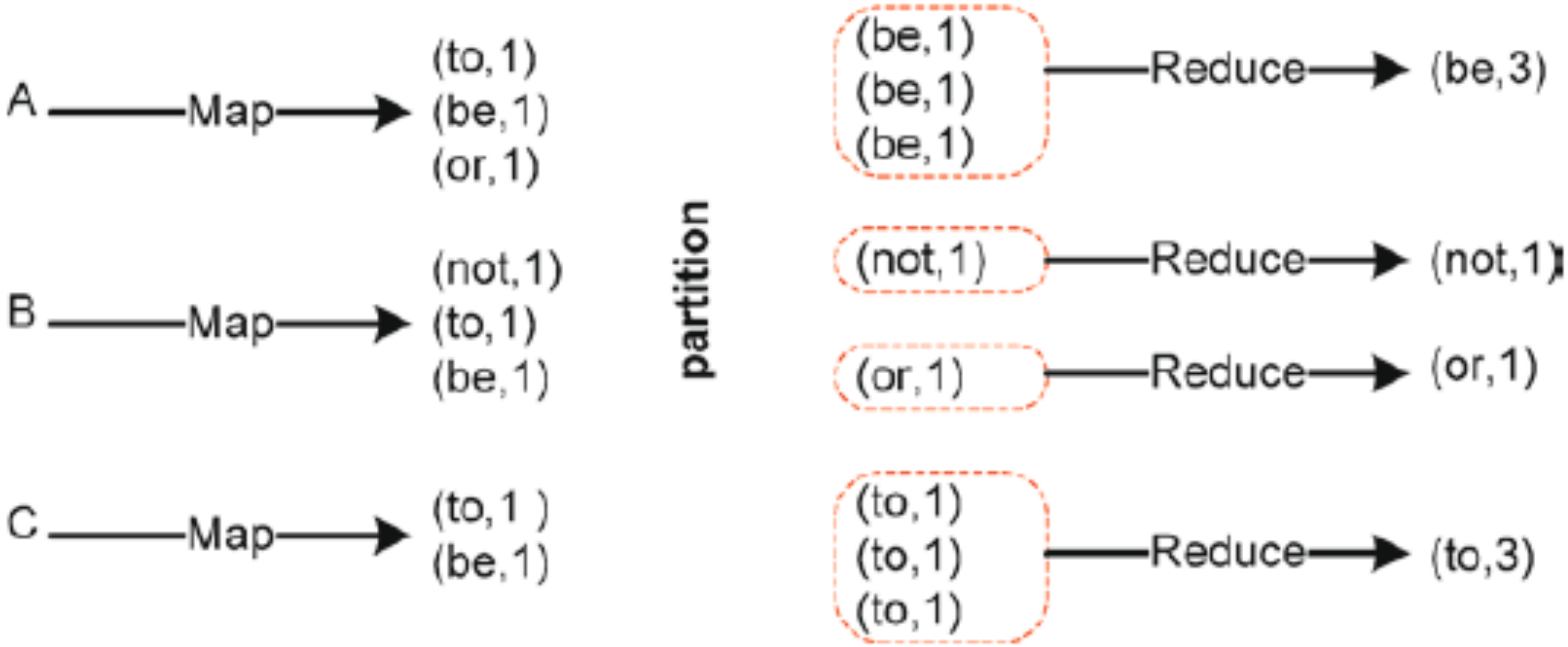
reduce(k', <v'>*) → <k', v''>*

reduce() combines those intermediate values into one or more final values per key (usually only one)

Word Count Example

Count the appearance of each word in a set of documents

(A.txt = to be or) (B.txt = not to be) (C.txt= to be)



Word Count Example

- 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));

Actual Google MapReduce

Example is written in pseudo-code

Actual implementation is in C++, using a MapReduce library

Bindings for Python and Java exist via interfaces

True code is somewhat more involved (defines how the input key/values are divided up and accessed, etc.)

Example 2: Word Length Count

Abridged Declaration of Independence

Map Task 1
(204 words)

A Declaration By the Representatives of the United States of America, in General Congress Assembled.
When in the course of human events it becomes necessary for a people to advance from that subordination in which they have hitherto remained, and to assume among powers of the earth the equal and independent station to which the laws of nature and of nature's god entitle them, a decent respect to the opinions of mankind requires that they should declare the causes which impel them to the change.
We hold these truths to be self-evident; that all men are created equal and independent; that from that equal creation they derive rights inherent and inalienable, among which are the preservation of life, and liberty, and the pursuit of happiness; that to secure these ends, governments are instituted among men, deriving their just power from the consent of the governed; that whenever any form of government shall become destructive of these ends, it is the right of the people to alter or to abolish it, and to institute new government, laying it's foundation on such principles and organizing it's power in such form, as to them shall seem most likely to effect their safety and happiness. Prudence indeed will

(key, value)

(yellow, 17)
(red, 77)
(blue, 107)
(pink, 3)

Yellow: 10+

Red: 5..9

Blue: 2..4

Pink: = 1

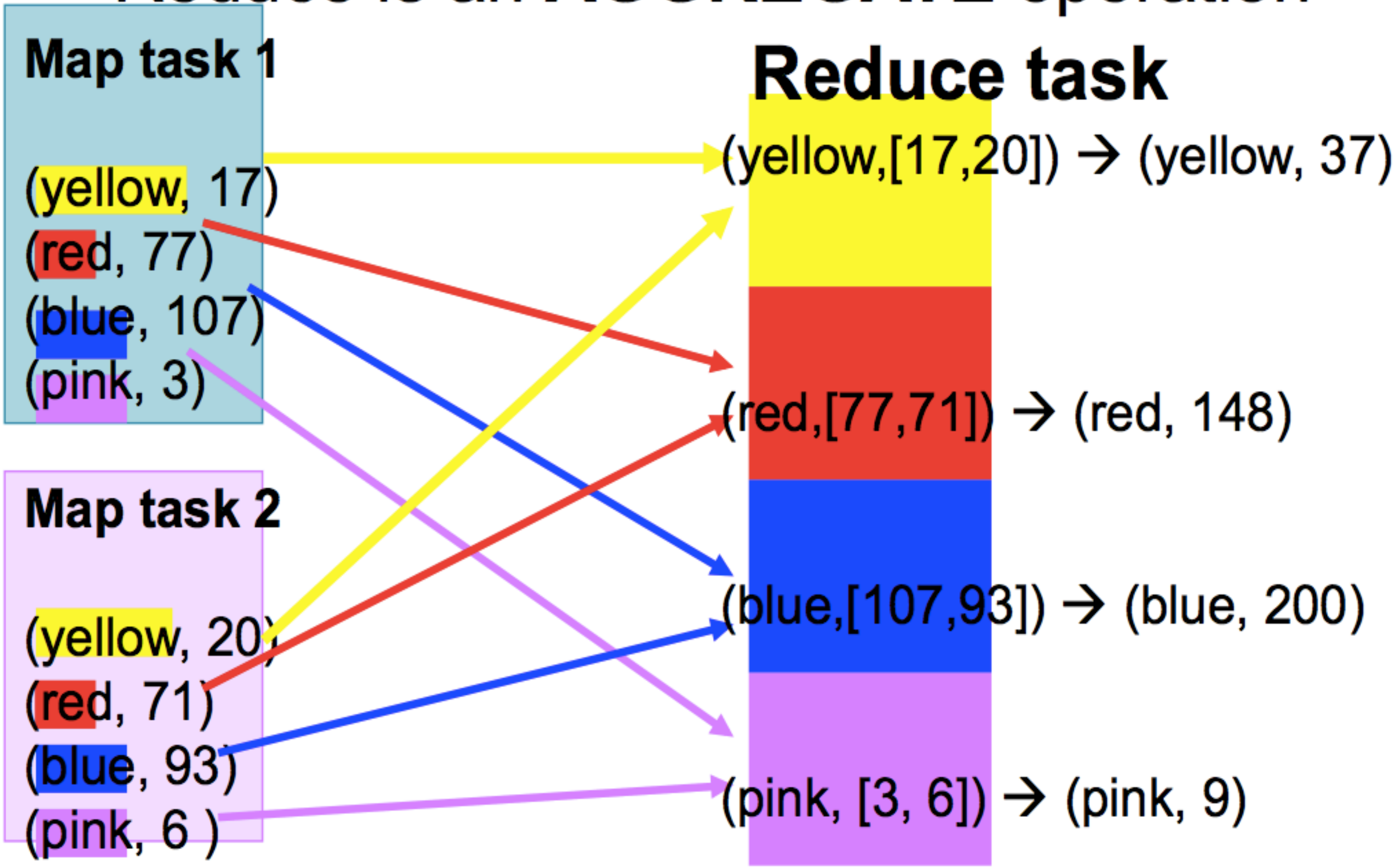
dictate that governments long established should not be changed for light and transient causes: and accordingly all experience hath shewn that mankind are more disposed to suffer while evils are sufferable, than to right themselves by abolishing the forms to which they are accustomed. But when a long train of abuses and usurpations, begun at a distinguished period, and pursuing invariably the same object, evinces a design to reduce them to arbitrary power, it is their right, it is their duty, to throw off such government and to provide new guards for future security. Such has been the patient sufferings of the colonies; and such is now the necessity which constrains them to expunge their former systems of government. the history of his present majesty is a history of unremitting injuries and usurpations, among which no one fact stands single or solitary to contradict the uniform tenor of the rest, all of which have in direct object the establishment of an absolute tyranny over these states. To prove this, let facts be submitted to a candid world, for the truth of which we pledge a faith yet unsullied by falsehood.

(yellow, 20)
(red, 71)
(blue, 93)
(pink, 6)

Map Task 2
(190 words)

Example 2: Word Length Count

Map is a **GROUP BY** operation
Reduce is an **AGGREGATE** operation



MapReduce: Architecture

One master, many workers

- Input data split into M map tasks (typically 64 MB in size)
- Reduce phase partitioned into R reduce tasks
- Tasks are assigned to workers dynamically
- Per worker input size = GFS chunk size!
- Often: M=200,000; R=4,000; workers=2,000

MapReduce: Scheduling

Master assigns each map task to a free worker

- Considers locality of data to worker when assigning task
- Worker reads task input (often from local disk!)
- Worker produces R local files containing intermediate k/v pairs

Master assigns each reduce task to a free worker

- Worker reads intermediate k/v pairs from map workers
- Worker sorts & applies user's Reduce op to produce the output

Parallelism

map() functions run in parallel, creating different intermediate values from different input data sets

reduce() functions also run in parallel, each working on a different output key

All values are processed independently

Bottleneck: reduce phase can't start until map phase is completely finished*

**True only for the original version of MapReduce*

Fault Tolerance

Master detects worker failures

- Re-executes completed & in-progress map() tasks
- Re-executes in-progress reduce() tasks

Master notices particular input key/values that cause crashes in map(), and skips those values on re-execution.

Widely Applicable at Google

distributed grep

distributed sort

term-vector per host

document clustering

machine learning

…

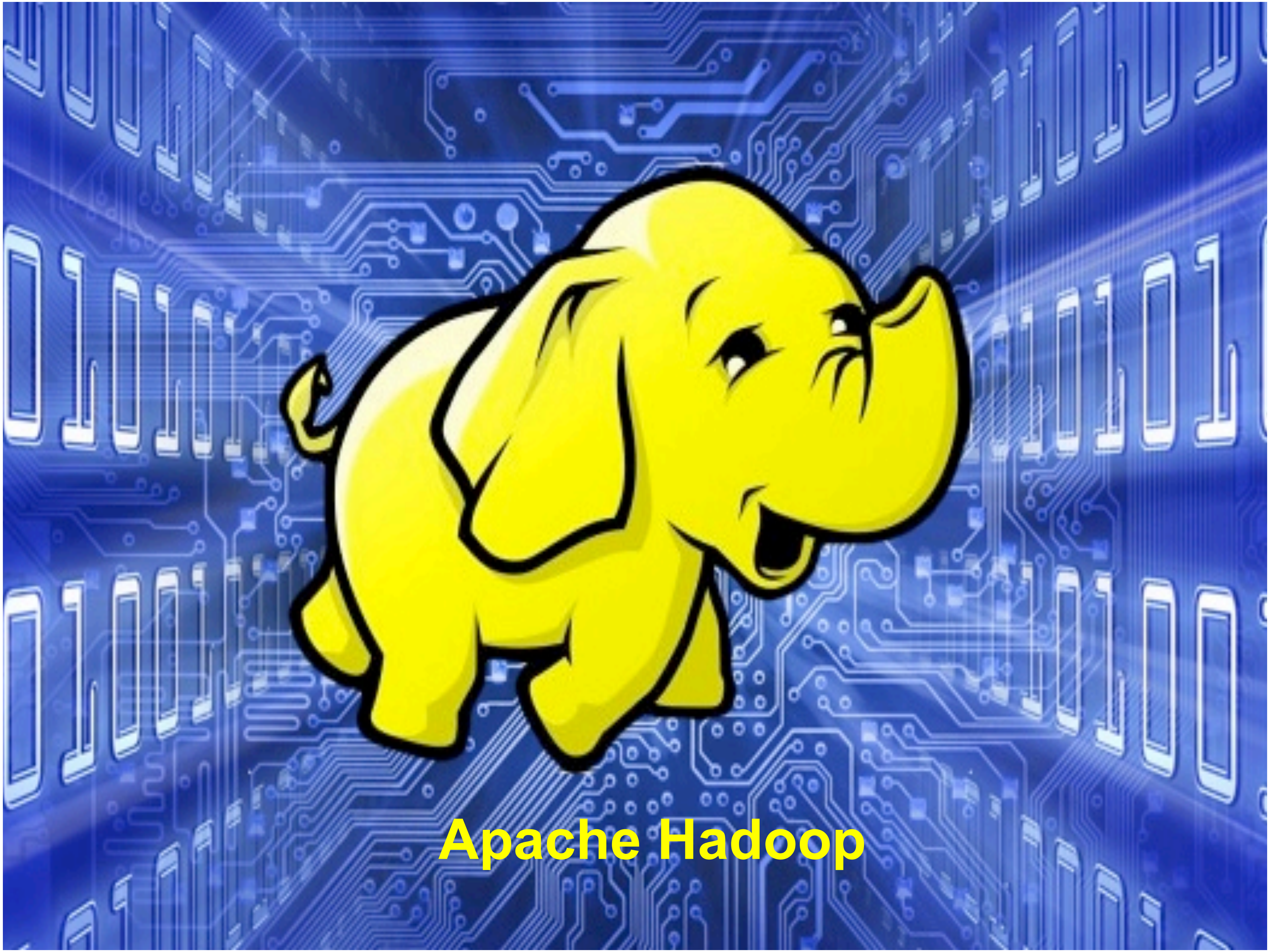
web access log stats

web link-graph reversal

inverted index construction

statistical machine translation

…

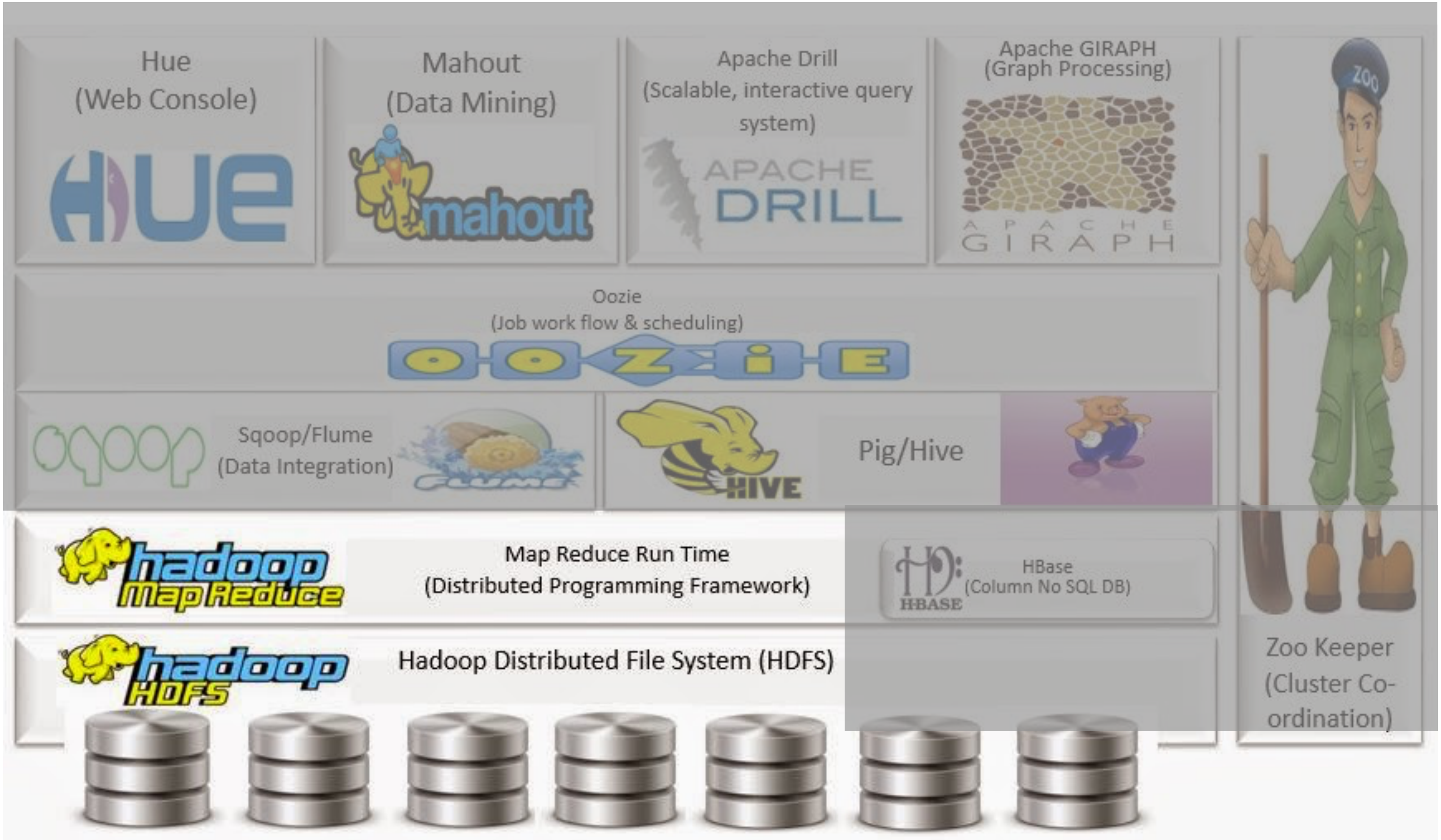

Apache Hadoop

What is Hadoop?

- Hadoop is a top-level Apache project
  - Open source implementation of MapReduce
  - Developed in Java

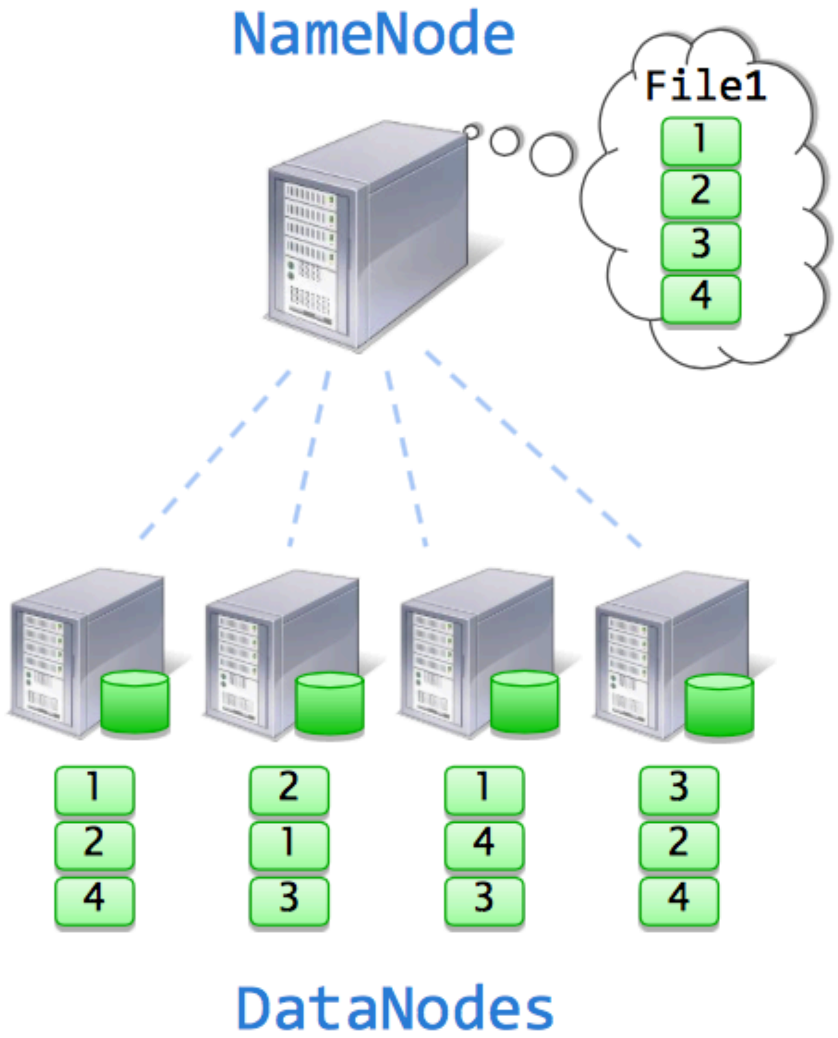
- Platform for data storage and processing
  - Scalable
  - Fault tolerant
  - Distributed
  - Any type of complex data

Hadoop Eco-System



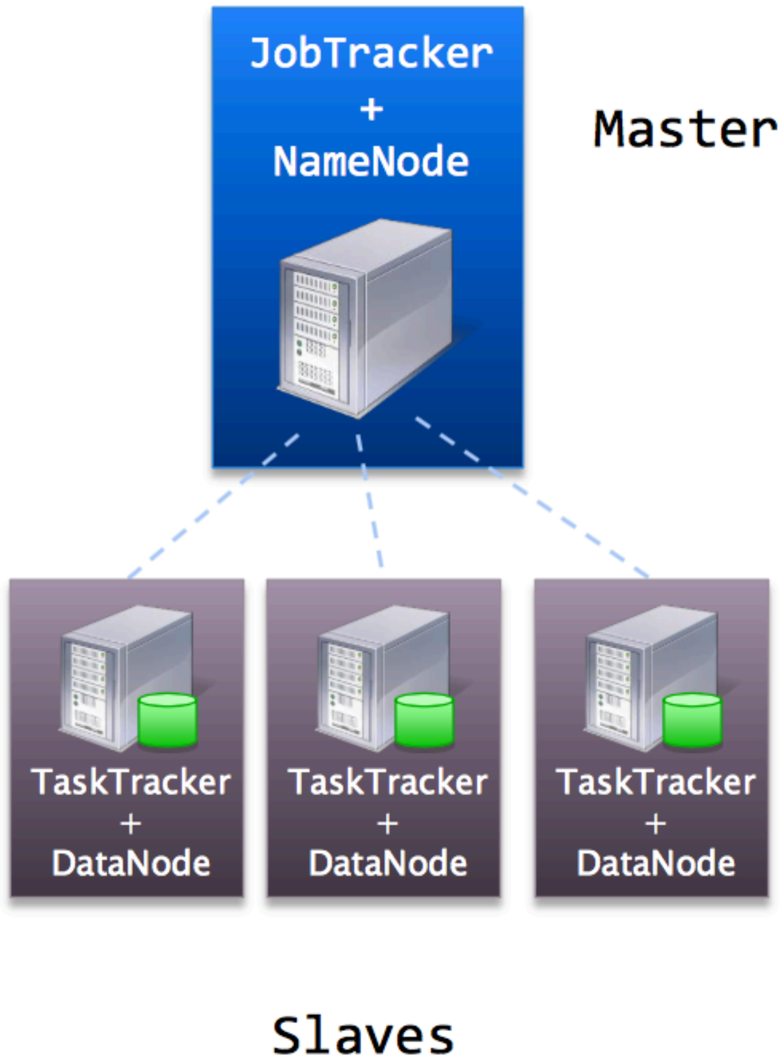
HDFS – Distributed Storage System

- Files split into 128 MB blocks
- Blocks replicated across several DataNodes (usually 3)
- Single NameNode stores metadata (file names, block locations, etc.)
- Optimized for large files, sequential reads
- Files are append-only
- Rack-aware



Hadoop MapReduce

- Parallel processing for large datasets
- Relies on HDFS
- Master-Slave architecture:
  - Job Tracker schedules and manages jobs
  - Task Trackers execute individual map() and reduce() task on each cluster node
- JobTracker and Namenode as well as TaskTrackers and DataNodes are placed on the same machines



Word Count Example In Hadoop

```
public void map(WritableComparable key, Writable value, OutputCollector output, Reporter reporter) throws
      IOException {

   String line = ((UTF8)value).toString();

   StringTokenizer itr = new StringTokenizer(line);

   while (itr.hasMoreTokens()) {
            word.set(itr.nextToken());
             output.collect(word, one);
   }
}


public void reduce(WritableComparable key, Iterator values, OutputCollector output,Reporter reporter) throws
      IOException {
   int sum = 0;

   while (values.hasNext()) {
      sum += ((IntWritable) values.next()).get();
    }

    output.collect(key, new IntWritable(sum));
}
```

MapReduce and Hadoop in a Nutshell

- By providing a data-parallel programming model, MapReduce can control job execution in useful ways:
  - Automatic division of job into tasks
  - Automatic partition and distribution of data
  - Automatic placement of computation near data
  - Recovery from failures

- Hadoop, an open source implementation of MapReduce, enriched by many useful subprojects
- User focuses on application, not on complexity of distributed computing

4

The Future of MapReduce?

MapReduce is important.

Why is MapReduce important?

It solves the Big Data problem.

It works for Google.

Looking back…

- 2004: Google's OSDI paper on MapReduce
- 2004: Open-source MapReduce: Nutch, then Hadoop
- 2008: Cloudera: Apache Hadoop-based software and services

Then…

Today: Who is not Using Hadoop??

2007

Yahoo!
Powerset
last.fm

2008

Google
ablegrape
ImageShack
Cascading
IBM
facebook
ENORMO
A9
University of Edinburgh
krugle
rackspace hosting
Lookery Control freaks welcome
The New York Times
Joost
Zvents Discover Things To Do
Information Sciences Institute
News Corporation
Cornell University Computing and Information Science
Visible Measures
LOTAME
NetSeer
parc Palo Alto Research Center
Security Enhanced Domain Name System
veoh

2009

AOL
cloudera
deepdyve
cooliris
eyealike
TEXTMAP The entity search engine
PSG College of Technology
iterend
tailsweep
hulu
RapLeaf
USCMS
Ning
quantcast
amazon webservices
pressflip
detikSearch
WorldLingo
Systems@ETH Zürich
VK SOLUTIONS Global Solutions Provider
TARAGANA Innovation • Quality • Simplicity
HOSTING HABITAT
HOLA
Terrier
adknowledge
stampede beta

2010

SAMSUNG
rubicon
BERKELEY LAB
VISIBLE TECHNOLOGIES
APOLLO GROUP
ADSDAQ
rackspace hosting
RapLeaf
wordnik
MOBIGEN
comScore
trulia real estate search
Accela Communications
Forward3D
LinkedIn
Microsoft
Infochimps Find the world's data
Pharm2Phork
ADMELD
gumgum
BrainPad
Pronux
The Datagraph Blog
NETFLIX
mobileanalytics.tv
markt24.de
twitter
media6degrees
BEEBLER
SLC Security
ebay

SO, WHAT'S NEXT?

SO, WHAT'S NEXT?
IS MAPREDUCE ENOUGH?

« A giant step backward… »

- *A giant step backward in the programming paradigm for large-scale data intensive applications*
- *A sub-optimal implementation, in that it uses brute force instead of indexing*
- *Not novel at all — it represents a specific implementation of well known techniques developed nearly 25 years ago*
- *Missing most of the features that are routinely included in current DBMS*
- *Incompatible with all of the tools DBMS users have come to depend on*

Stonebraker and DeWitt (2008)

The Brangelina of Big Data: Cassandra mates with Hadoop

Open source celebrity supercouple

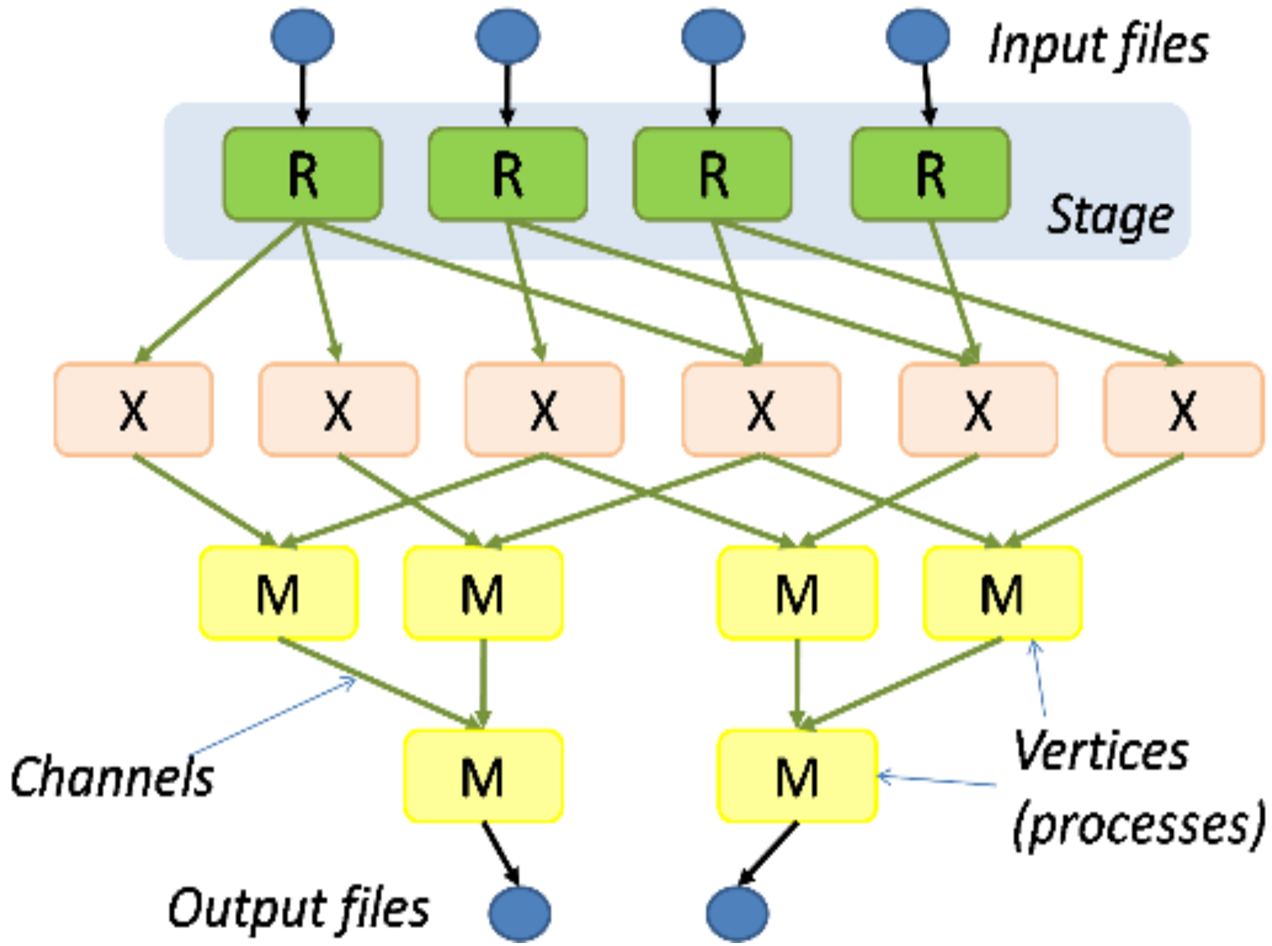
By **Cade Metz in San Francisco**

Posted in Cloud, 23rd March 2011 17:00 GMT

Think of it as the Brangelina [1] of Big Data.

DataStax [2], an open-source startup based in Northern California, has combined Cassandra, the distributed database developed at Facebook [3], with Hadoop, the epic-number-crunching platform [4] based on Google's backend infrastructure [5]. Known as "Brisk", this Big Data mashup was unveiled on Wednesday, with DataStax promising to open source the platform under an Apache licence within 45 days.

Microsoft Dryad: Beyond MapReduce Graphs



Let’s get back to Google

MapReduce worked well enough.

Let’s get back to Google

MapReduce worked well enough.

BUT…

"*MapReduce isn't suited to calculations that need to occur in near real-time* »

What's wrong with MapReduce?

1. Too long latency
2. Too wasteful (full rework of tens-of-PB dataset despite minor input changes)
3. Inappropriate for real-time processing

- "*You can't do anything with it that takes a relatively short amount of time, so we got rid of it*"

The Register®

Hardware Software Music & Media Networks Security Public Sector Business Science Od

PCs & Chips **Servers** HPC Storage Data Networking Virtualization Sysadmin Blog

Google search index splits with MapReduce

Welds BigTable to file system 'Colossus'

By **Cade Metz in San Francisco** • **Get more from this author**

Posted in Servers, 9th September 2010 21:52 GMT

Exclusive Google Caffeine — the remodeled search infrastructure rolled out across Google's worldwide data center network earlier this year — is not based on MapReduce, the distributed number-crunching platform that famously underpinned the company's previous indexing system. As the likes of Yahoo!, Facebook, and Microsoft work to duplicate MapReduce through the open source Hadoop project, Google is moving on.

Google Percolator: MapReduce Death?

« *Large-scale Incremental Processing Using Distributed Transactions and Notifications*, Daniel Peng, Frank Dabek, Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation, 2010. »

- Efficient incremental processing of updates to a large data set
- Big-data ACID-compliant transaction-processing non-relational DBMS
- Replaces the former MapReduce-based batch processing system
- Updates processed hundreds of times faster

Was Stonebraker right?

10 Hadoop-able Problems

Where batch processing is enough…

- Risk Analysis
- Customer Churn
- Recommendation Engines
- Ad Targeting
- Sales Analysis
- Network Analysis
- Fraud Detection
- Trading Surveillance
- Search Quality
- General Data Analytics

Crédits: Cloudscale

More social, more mobile, more real-time

Where batch processing is NOT enough…

- Realtime Location Analytics
- Realtime Game Analytics
- Realtime Algorithmic Trading
- Realtime Government Intelligence
- Realtime Sensor Systems and Grids

Crédits: Cloudscale

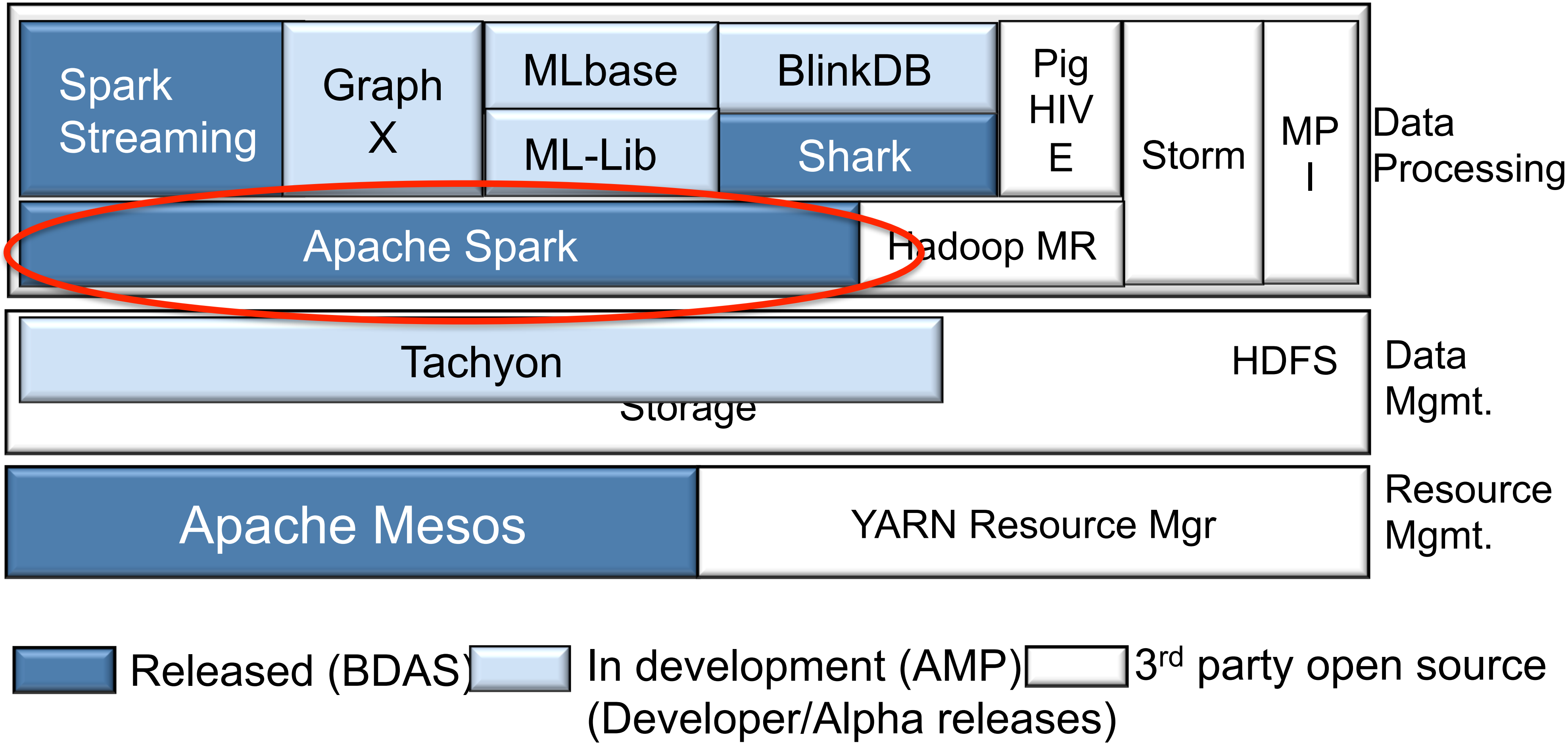
MapReduce is NOT « one size fits all »

- Welcome to the era of Real-time Big Data!

POST-HADOOP APPROACHES:

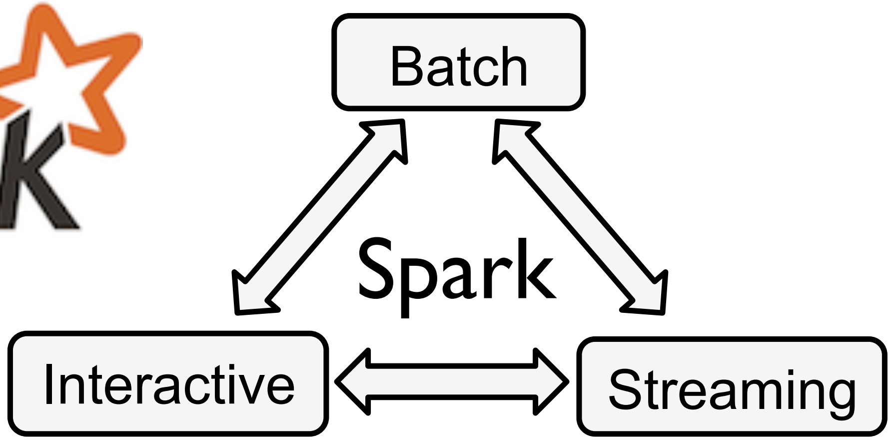
THE BERKELEY DATA ANALYTICS STACK

Berkeley Data Analytics Stack



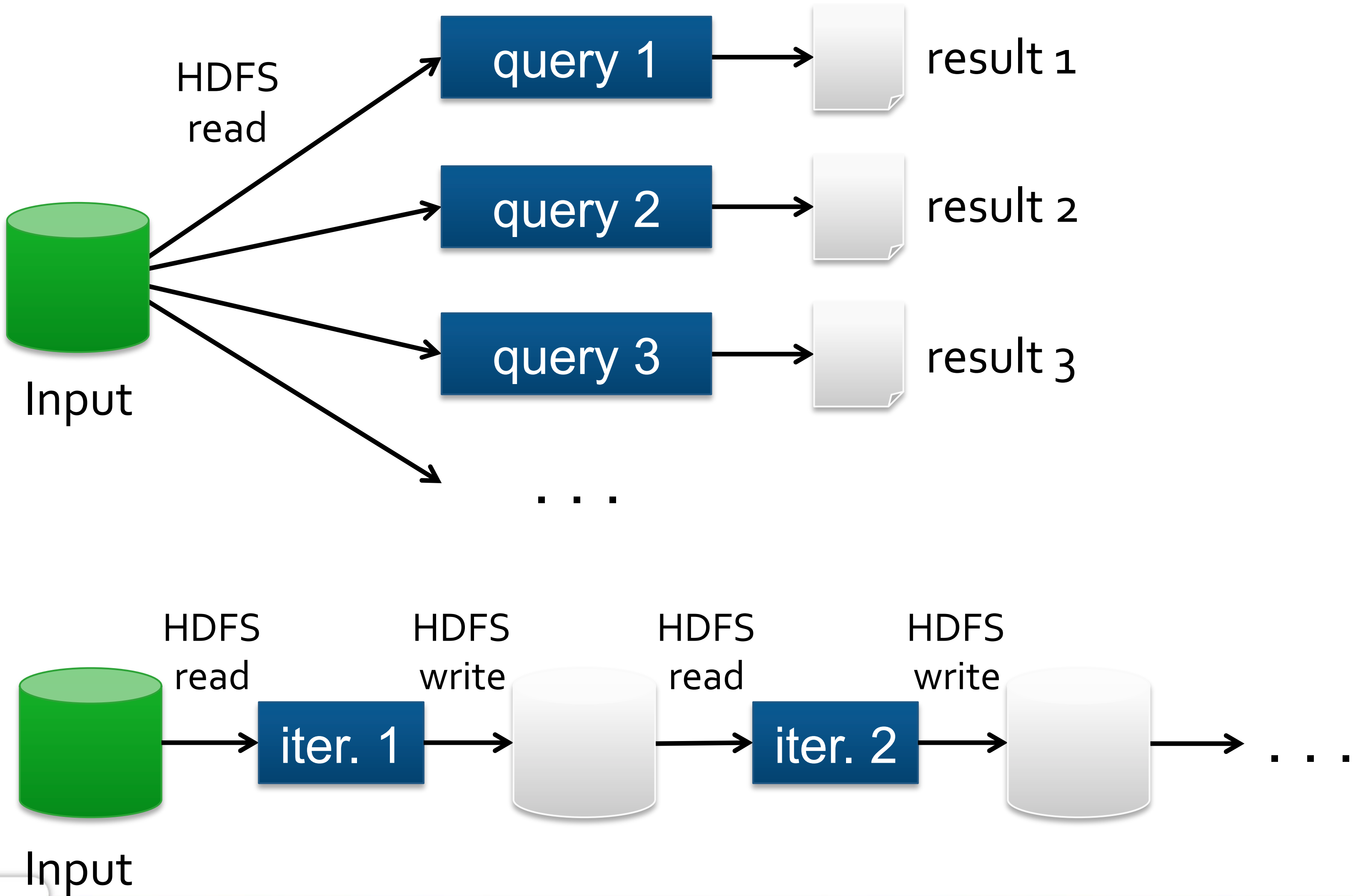
AMP BDAS Components being released under BSD or Apache Open Source License

Introducing

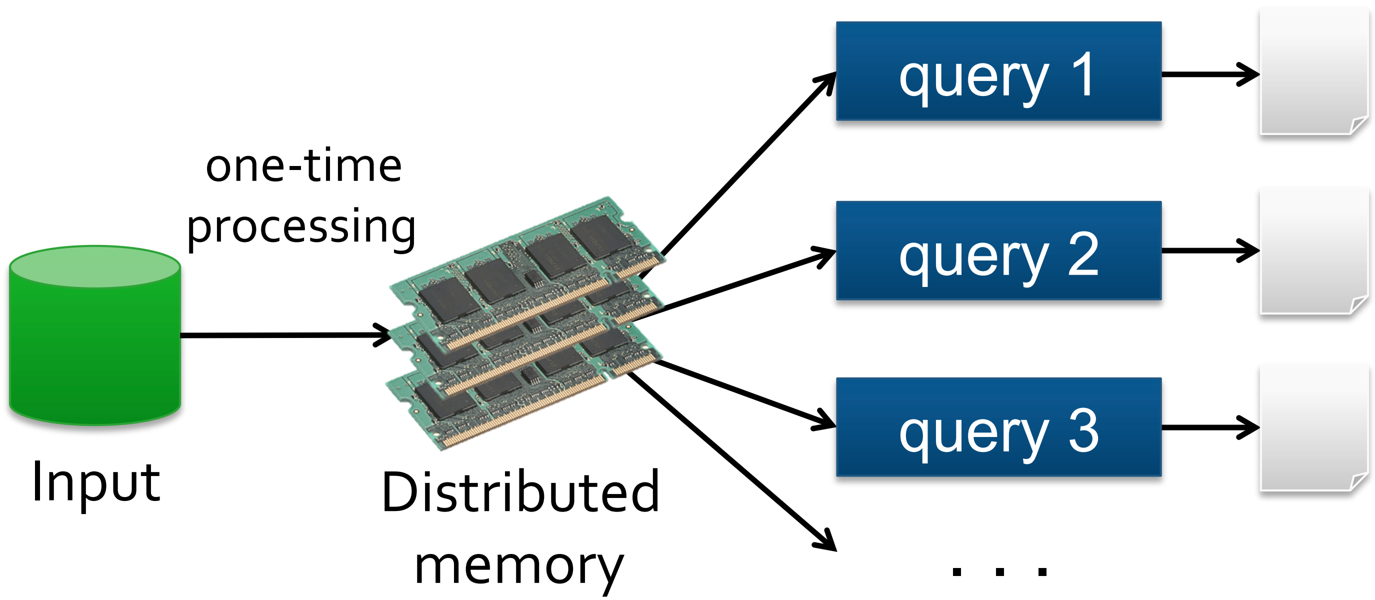


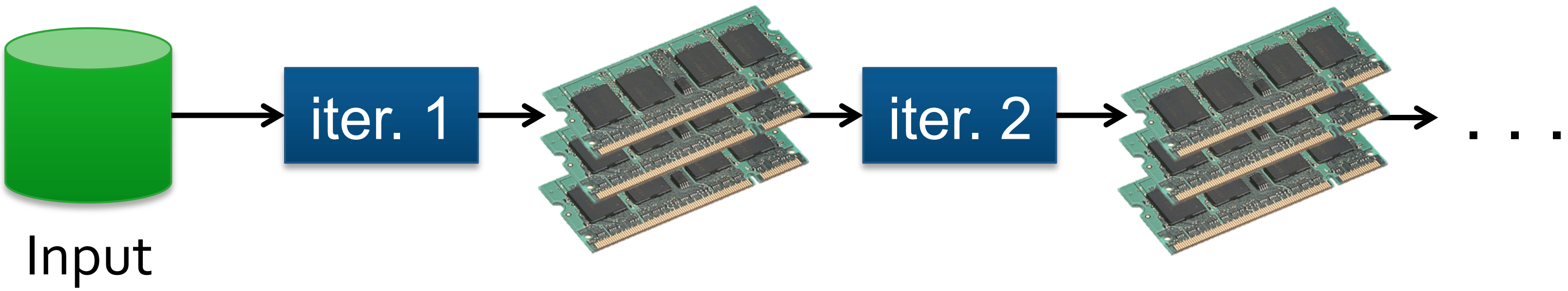
- Fast, MapReduce-like engine
  - In-memory storage abstraction for iterative/interactive queries
  - General execution graphs
  - Up to 100x faster than Hadoop MR (2-10x even for on-disk)
- Compatible with Hadoop's storage APIs
  - Can access HDFS, HBase, S3, SequenceFiles, etc
- Great example of ML/Systems (and eventually DB) collaboration

Queries and ML in Hadoop



In-Memory Data Sharing





Challenge

- How to design a distributed memory abstraction that is both **fault-tolerant** and **efficient**?
- Traditional in-memory storage systems replicate data or update logs across nodes -> slow!
  - Network write is 10-100× slower than memory

Resilient Distributed Datasets

- RDDs provide an interface for coarse-grained *transformations* (map, group-by, join, …) on immutable collections
- Efficient fault recovery using *lineage*
  - Log one operation to apply to many elements
  - Recompute lost partitions of RDD on failure
  - No cost if nothing fails

- Rich enough to capture many models:
  - **Data flow models**: MapReduce, Dryad, SQL, …
  - **Specialized models**: Pregel, Hama, …

M. Zaharia, et al, Resilient Distributed Datasets: A fault-tolerant abstraction for in-memory cluster computing, NSDI 2012. Best Paper Award

Example: Log Mining

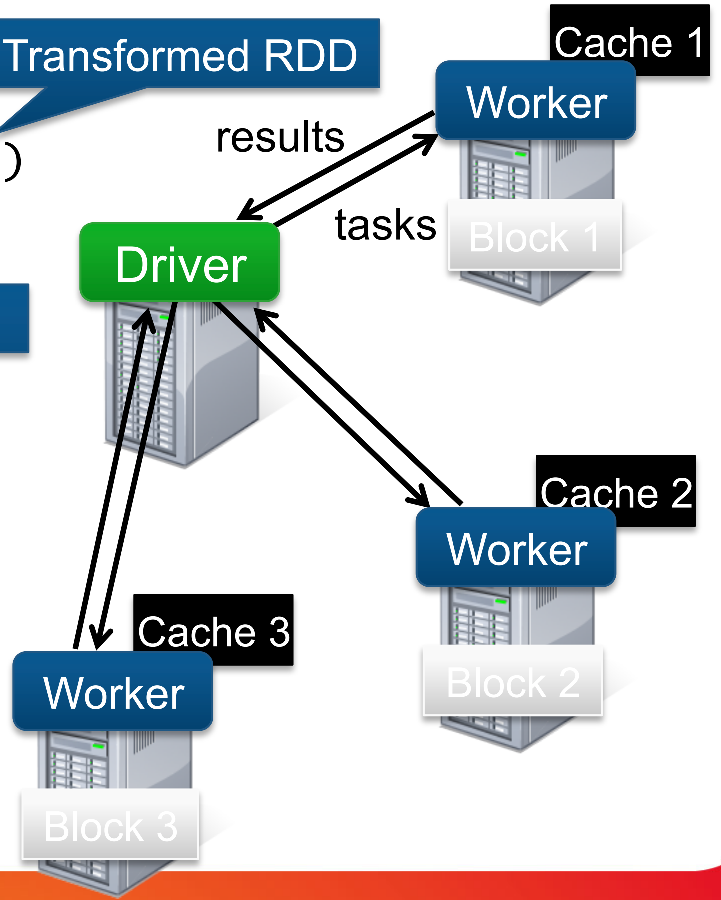
Load error messages from a log into memory, then interactively search for various patterns

Base RDD

Transformed RDD

```
lines = spark.textFile("hdfs://...")
errors = lines.filter(_.startsWith("ERROR"))
messages = errors.map(_.split('\t')(2))
cachedMsgs = messages.cache()

cachedMsgs.filter(_.contains("foo")).count
cachedMsgs.filter(_.contains("bar")).count
```

Action



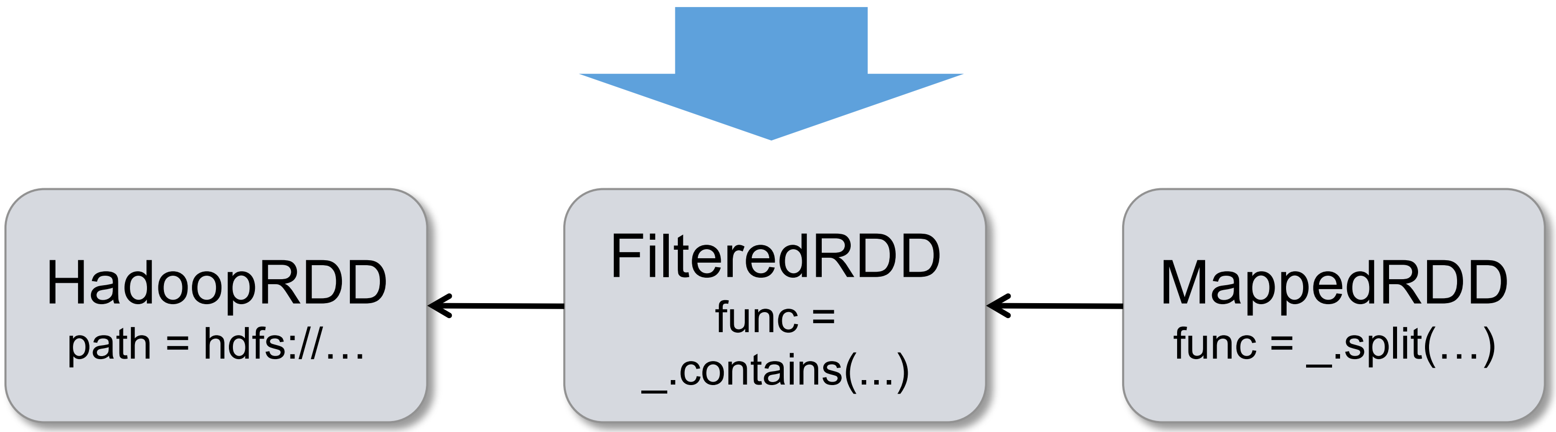
Result: scaled to 1 TB data in 5-7 sec
(vs 170 sec for on-disk data)

Fault Tolerance with RDDs

RDDs track the series of transformations used to build them (their *lineage*)

Enables per-node recomputation of lost data

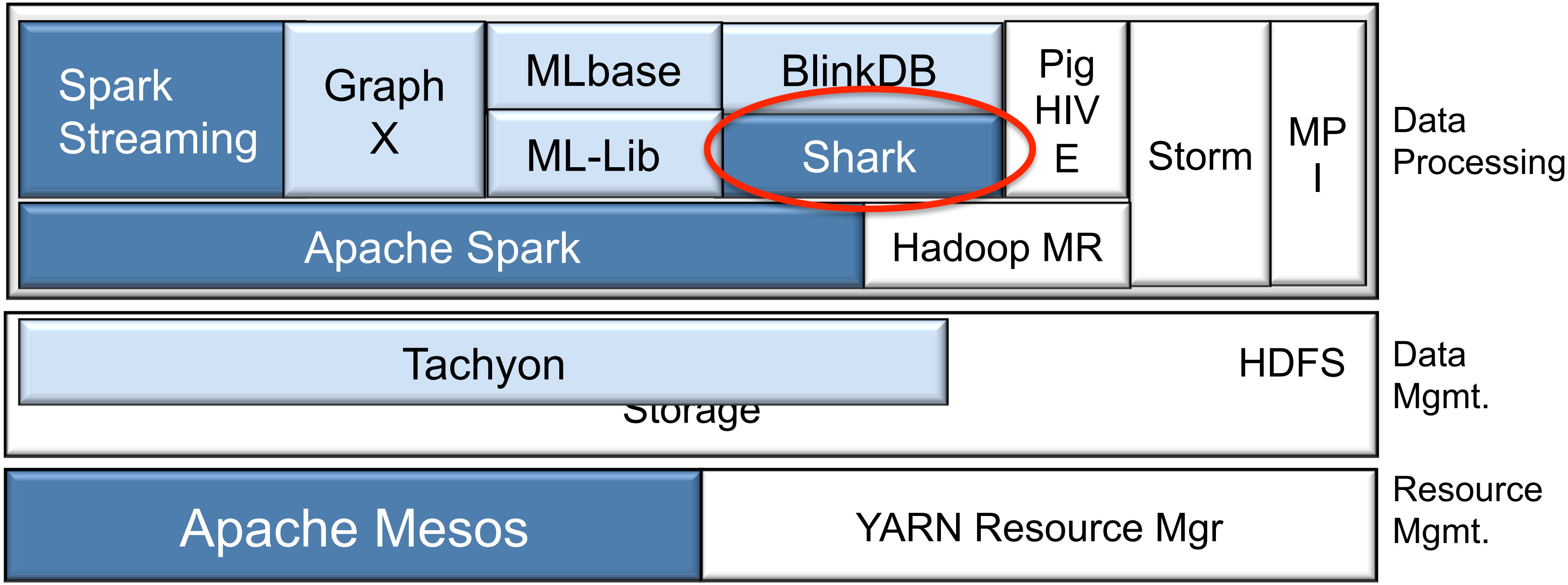
```
messages = textFile(...).filter(_.contains("error"))
                        .map(_.split('\t')(2))
```



Spark Status

- Current release (0.8) includes Java and Python APIs
  - Now in Apache Incubator
  - Includes: mini-batch streaming option, MLlib, YARN integration, EC2
  - 20 Companies contributed code in latest release
  - Spark Bay Area Meetup Group (1250 members) – new groups forming
- Sample Use cases:
  - In-memory analytics on Hive data (Conviva)
  - Interactive queries on data streams (Quantifind)
  - Business intelligence (Yahoo!)
  - Traffic estimation w/ GPS data (Mobile Millennium)
  - DNA sequence analysis (SNAP)

Berkeley Data Analytics Stack



Released (BDAS) In development (AMP) (Developer/Alpha releases) 3rd party open source

AMP BDAS Components being released under BSD or Apache Open Source License

"NoSQL" Irony

GIGAOM Events Research Jobs paidContent

Home Apple Cleantech Cloud Data Europe Mobile Video

analytics / big data / hadoop

SQL is what's next for Hadoop: Here's who's doing it

by Derrick Harris FEB. 21, 2013 - 10:29 AM PDT

4 Comments +1

A A

SUMMARY: *More and more companies and open source projects are trying to let users run SQL queries from inside Hadoop itself. Here's a list of what's available and, on a high level, how they work.*

tweet this

photo: Shutterstock / hauhu

What's Good About MapReduce?

1. Scales out to thousands of nodes in a fault-tolerant manner
2. Great for analyzing semi-structured data and complex analytics
3. Elasticity (cloud computing)
4. Multi-tenancy resource sharing

What Makes MR-based systems slow?

1. Disk-based intermediate outputs.
2. Inferior data format and layout (e.g., no control of data co-partitioning).
3. Execution strategies (lack of optimization based on data statistics).
4. Task scheduling and launch overhead!

None of these should be fundamental!

Shark = Spark + Hive

Uses Spark's in-memory RDD caching and language

- Result reuse and low latency
- Scalable, fault-tolerant, fast

Query Compatible with Hive

- Run HiveQL queries (w/ UDFs, UDAs…) **without modifications**
- Convert logical query plan generated from Hive into Spark execution graph

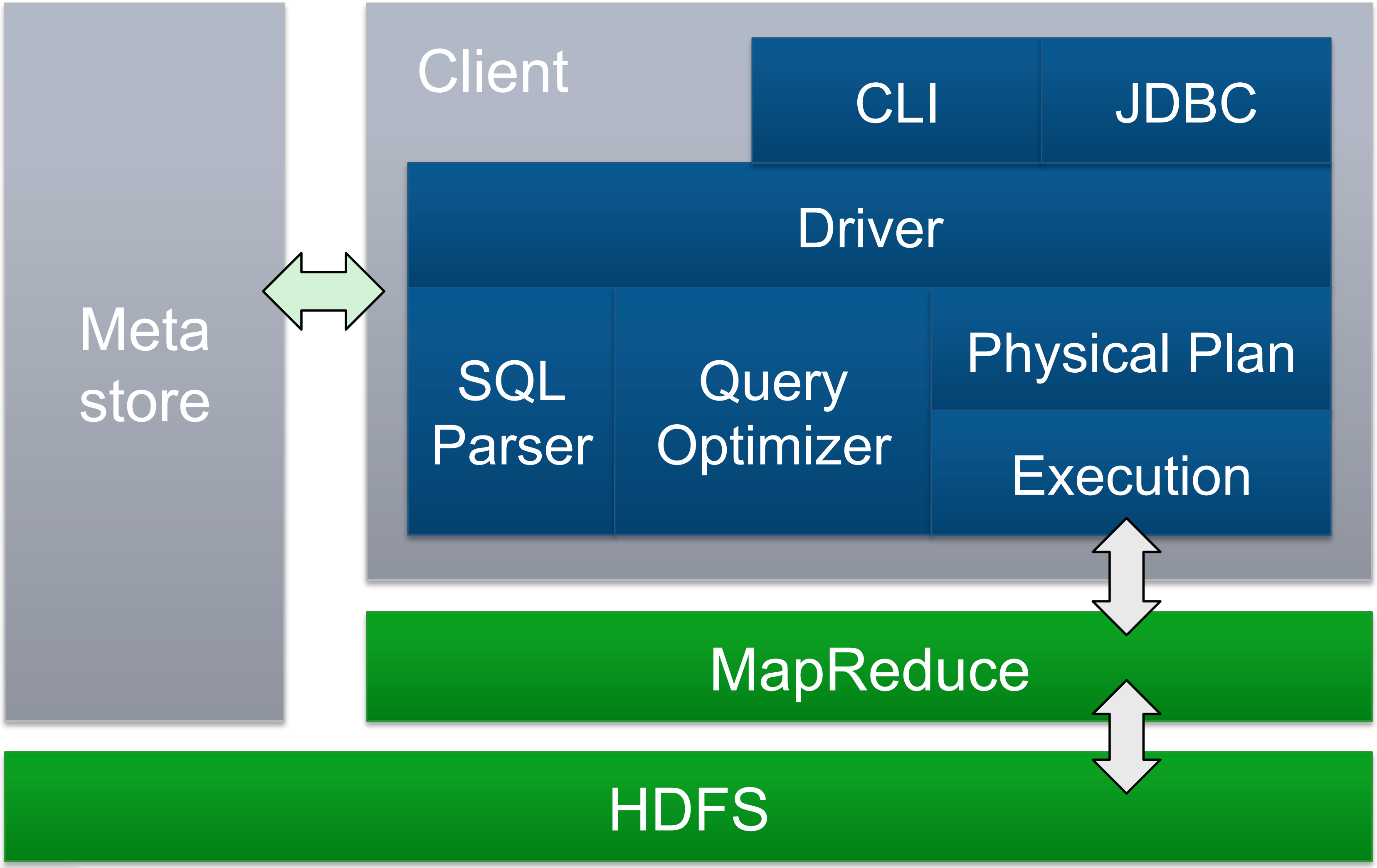
Data Compatible with Hive

- Use existing HDFS data and Hive metadata, **without modifications**

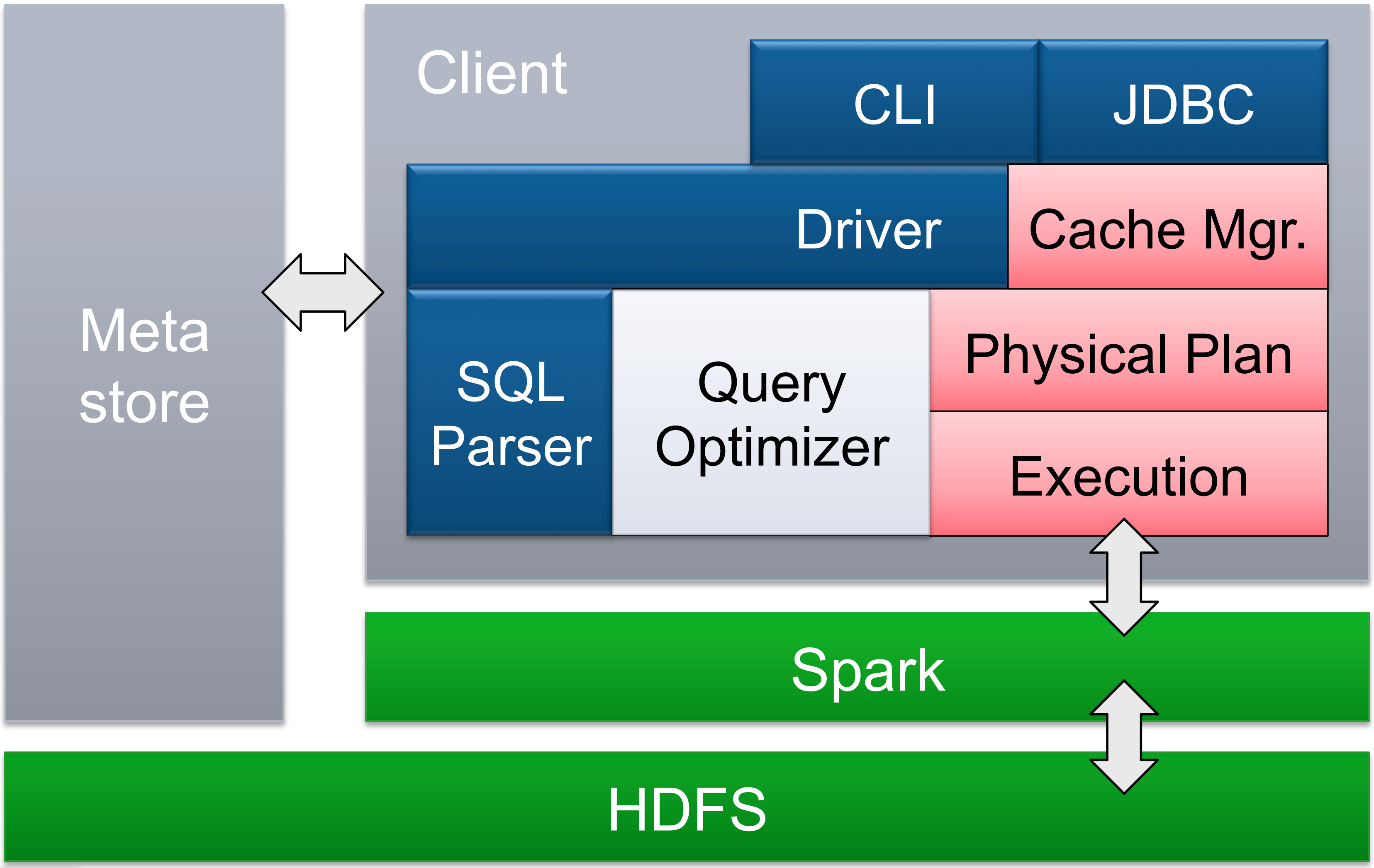
C. Engle, et al, Shark: Fast Data Analysis Using Coarse-grained Distributed Memory, SIGMOD 2012 (system demonstration). Best Demo Award

R. Xin et al., Shark: SQL and Rich Analytics at Scale, SIGMOD 2013.

Hive Architecture

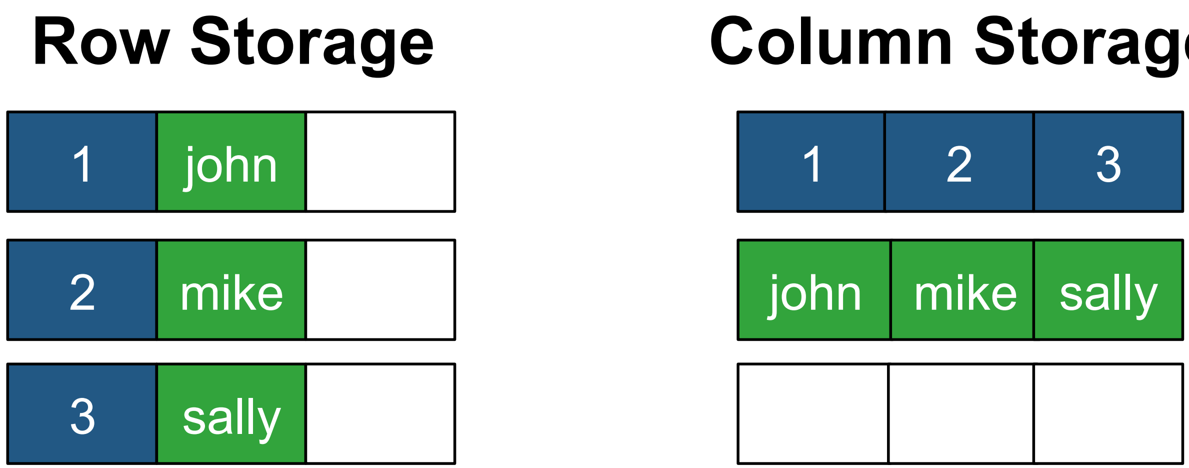


Shark Architecture



Column-Oriented Storage

- Caching Hive records as Java objects is inefficient
- Instead, use *arrays of primitive types* for columns
    - Similar size to serialized form, but 5x faster to process
- *Columnar compression* can further reduce size by 5x

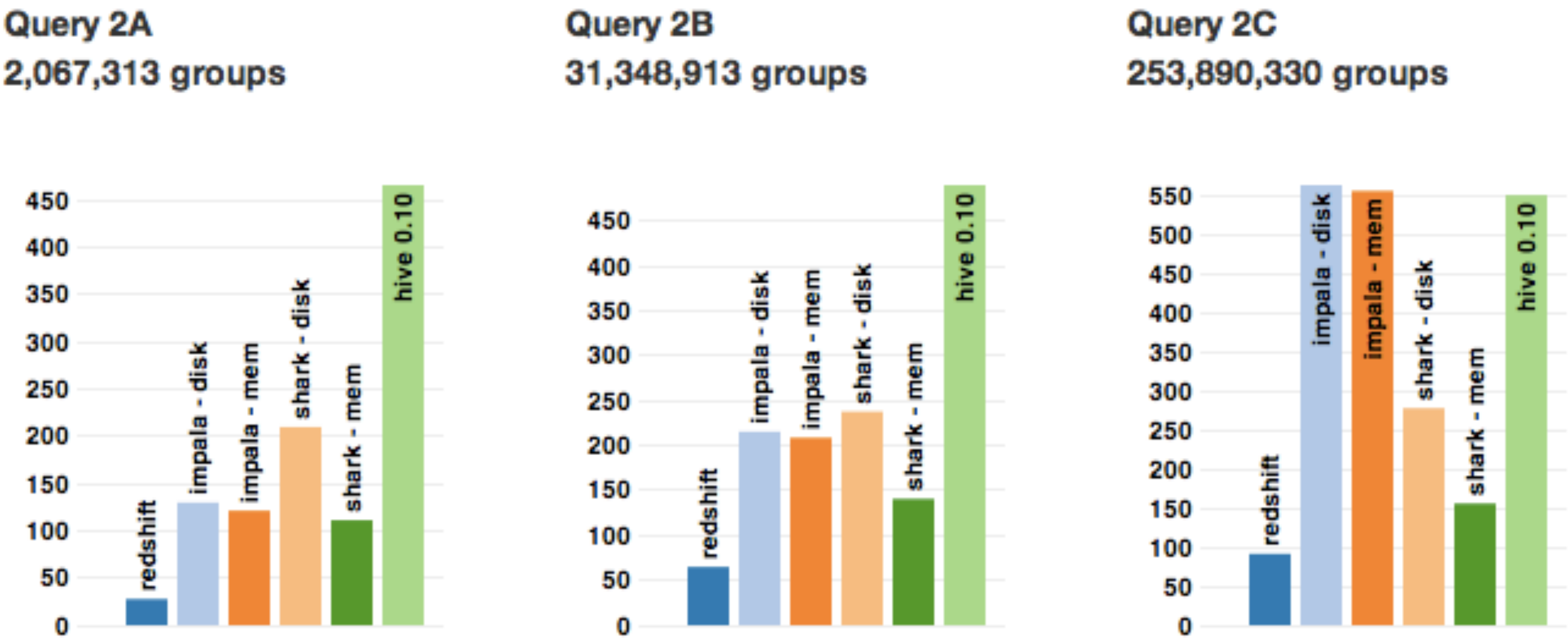


Other Shark Optimizations

- Fast task start-up
- Dynamic (mid-query) join algorithm selection based on statistical properties of data
- Runtime selection of # of reducers
- Partition pruning using range statistics
- Controllable table partitioning across nodes

Benchmark: Grouped Aggregation

```
SELECT SUBSTR(sourceIP, 1, X), SUM(adRevenue)
FROM uservisits
GROUP BY SUBSTR(sourceIP, 1, X)
```



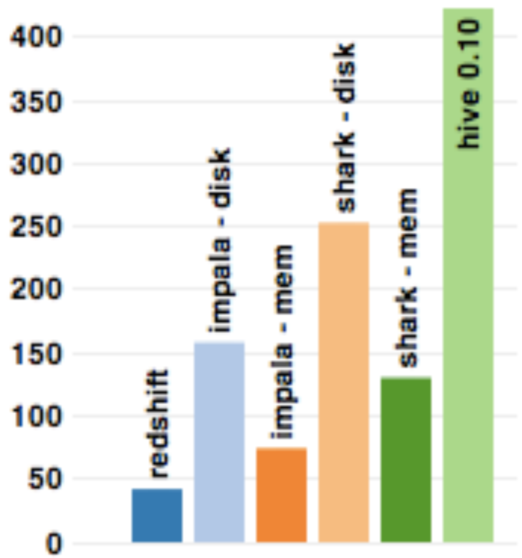
https://amplab.cs.berkeley.edu/benchmark/

Benchmark: Join + Order By

```
SELECT srcIP,  AVG(pageRank), SUM(adRevenue) as totalRev
FROM Rankings AS R, UserVisits AS UV
WHERE R.pageURL = UV.destURL
        AND UV.visitDate BETWEEN Date(`1980-01-01') AND Date(`X')
GROUP BY UV.sourceIP
ORDER BY totalRev DESC LIMIT 1
```

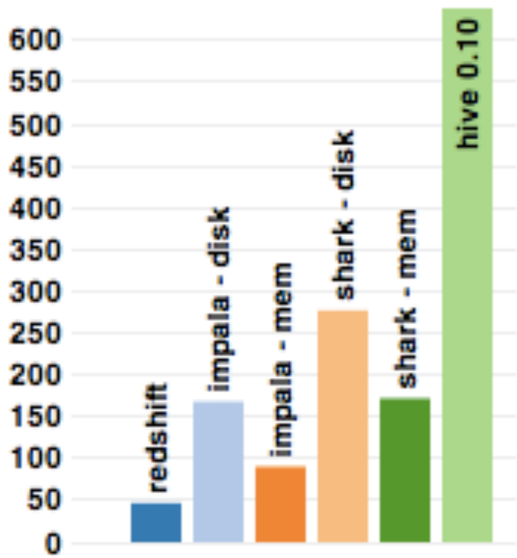
Query 3A

485,312 rows



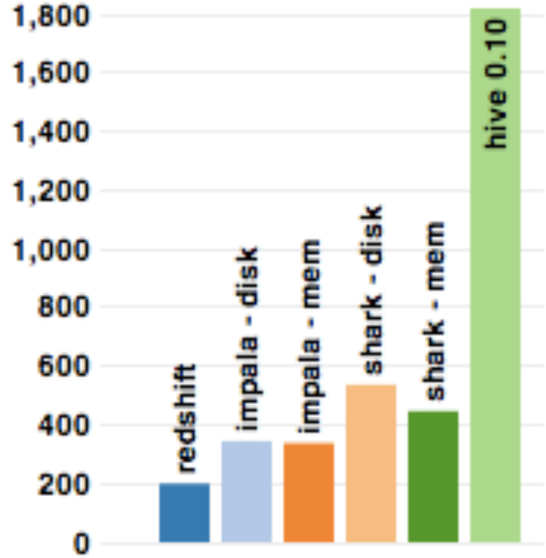
Query 3B

53,332,015 rows



Query 3C

533,287,121 rows

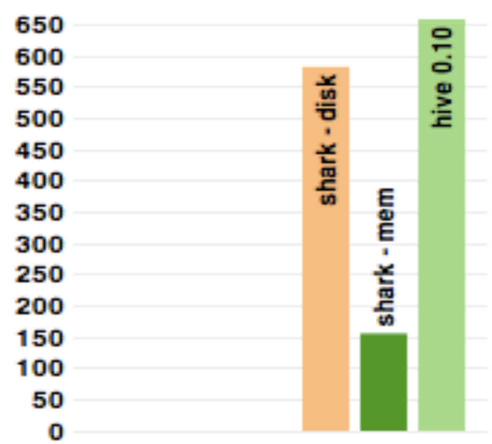


https://amplab.cs.berkeley.edu/benchmark/

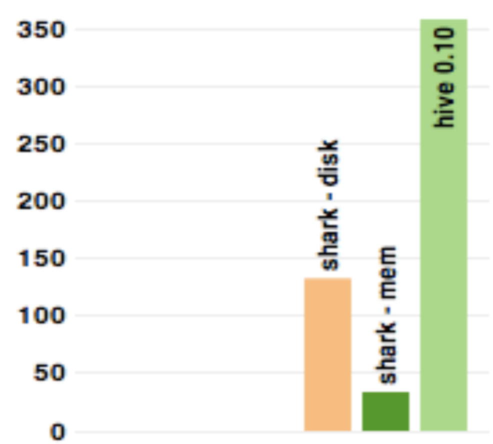
Benchmark: User-Defined Function

```
CREATE TABLE url_counts_partial AS
  SELECT TRANSFORM (line)
    USING "python /root/url_count.py" as (sourcePage, destPage, cnt)
  FROM documents;
CREATE TABLE url_counts_total AS
  SELECT SUM(cnt) AS totalCount, destPage
  FROM url_counts_partial
  GROUP BY destPage;
```

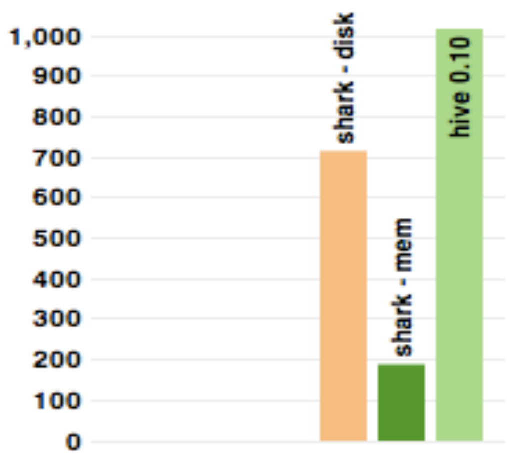
Query 4 (phase 1)



Query 4 (phase 2)



Query 4 (total)



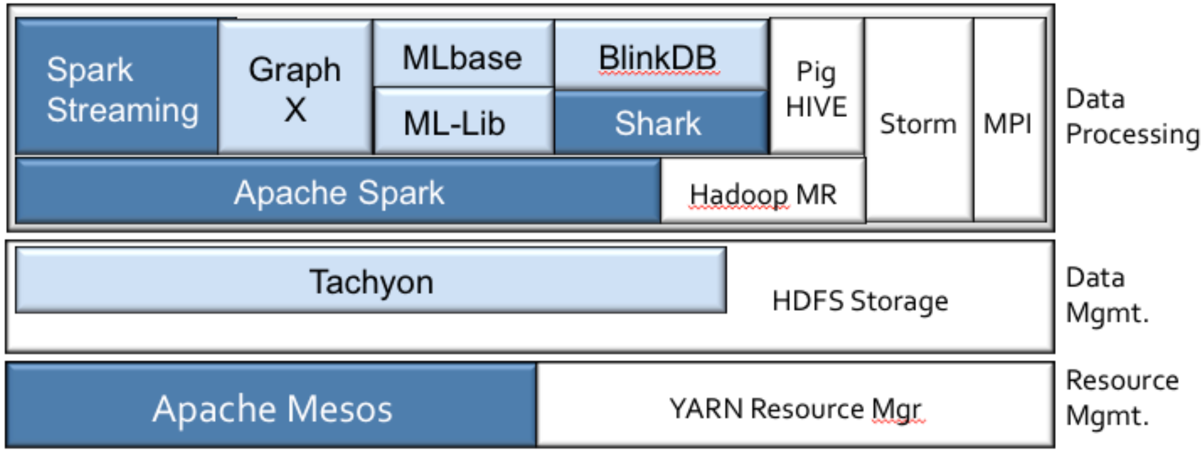
https://amplab.cs.berkeley.edu/benchmark/

A Unified System for SQL, RT & ML

- Can move seamlessly between SQL and Machine Learning worlds

```
def logRegress(points: RDD[Point]): Vector {
  var w = Vector(D, _ => 2 * rand.nextDouble - 1)
  for (i <- 1 to ITERATIONS) {
    val gradient = points.map { p =>
      val denom = 1 + exp(-p.y * (w dot p.x))
      (1 / denom - 1) * p.y * p.x
    }.reduce(_ + _)
    w -= gradient
  }
  w
}

val users = sql2rdd("SELECT * FROM user u
   JOIN comment c ON c.uid=u.uid")

val features = users.mapRows { row =>
  new Vector(extractFeature1(row.getInt("age")),
             extractFeature2(row.getStr("country")),
             ...)}
val trainedVector = logRegress(features.cache())
```

BDAS Present - Summary

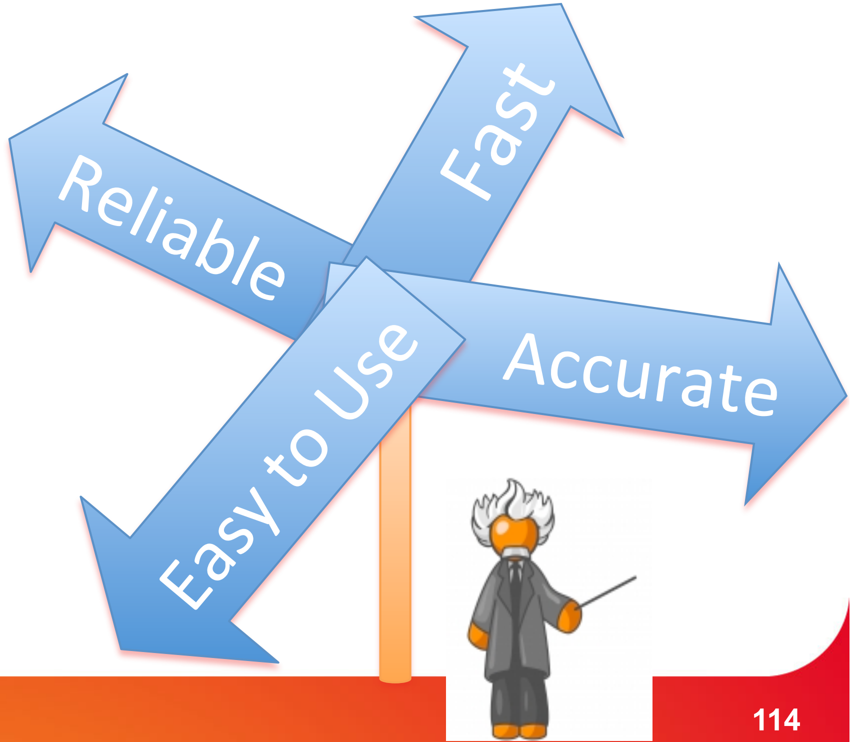


- **Spark Streaming** – Real-time Processing
- **Tachyon** – Memory-based layer on top of HDFS
- **BlinkDB** – Approximate Query Processing
- **MLlib** – Scalable Distributed Machine Learning Library in Spark
- **GraphX** – Graph processing integrated with Spark and Shark

See http://amplab.cs.berkeley.edu/publications for more information

BDAS – What's Next?

Address the "**3 E's of Big Data**"!



The 3 E's of Big Data:

- Extreme Elasticity Everywhere

Extreme Elasticity - Machines

- Option #1 – Build your own Cluster/WSC

Expense (% total)	Category	Monthly cost
Amortized CAPEX (85%)	Servers	$2,000,000
	Networking equipment	$290,000
	Power and cooling infrastructure	$765,000
	Other infrastructure	$170,000
OPEX (15%)	Monthly power use	$475,000
	Monthly people salaries and benefits	$85,000
	Total OPEX	$3,800,000

46K Servers
(2010 estimate from H&P book)

Option #2 – Rent Machines from AWS

High-Memory On-Demand Instances	
Extra Large	$0.460 per Hour
Double Extra Large	$0.920 per Hour
Quadruple Extra Large	$1.840 per Hour

x #Servers needed

Option #3 – Try your luck on the Spot Market

High-Memory Spot Instances	
Extra Large	$0.035 per Hour
Double Extra Large	$0.07 per Hour
Quadruple Extra Large	$0.14 per Hour

x #Servers needed

(US East – Saturday Sept 28 @1:30am)

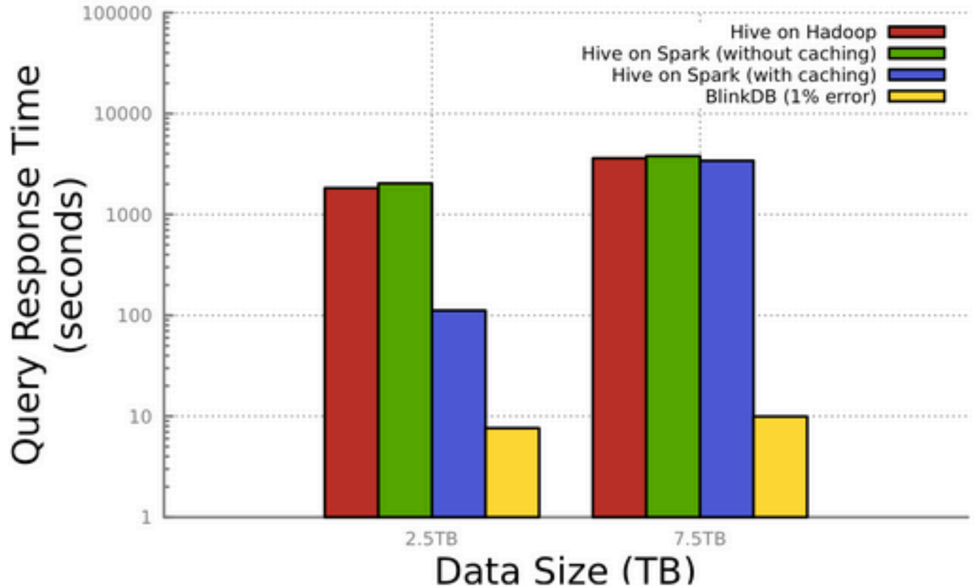
Extreme Elasticity - Algorithms

```
SELECT avg(sessionTime)
FROM Table
WHERE city='San Francisco'
WITHIN 2  SECONDS
```

```
SELECT avg(sessionTime)
FROM Table
WHERE city='San Francisco'
ERROR 0.1 CONFIDENCE 95.0%
```

Queries with Time Bounds

Queries with Error Bounds



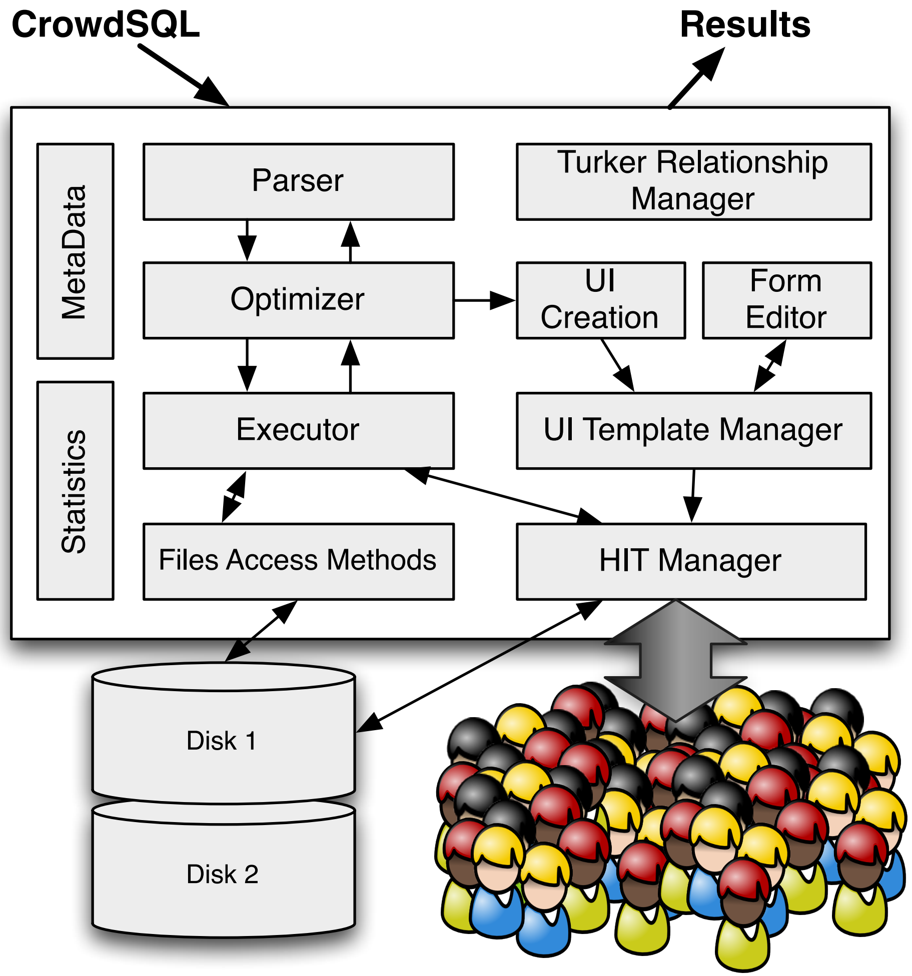
Agarwal et al., BlinkDB: Queries with Bounded Errors and Bounded Response Times on Very Large Data. ACM EuroSys 2013, Best Paper Award

CrowdDB – People-powered Query Processing

Other People-powered Database projects:

Qurk – MIT

Deco – Stanford



Franklin et al., CrowdDB: Answering Queries with Crowdsourcing, *SIGMOD 2011*
Feng et al., Query Processing with the VLDB Crowd, *VLDB 2011 Best Demo Award*
Trushkowsky et al., Crowdsourcing Enumeration Queries, *ICDE 2013 Best Paper Award*

Extreme Elasticity - People

Constantly changing workforce
Adapting/Learning workers
Various Incentives
Fatigue, Fraud, & other Failure Modes
Latency & Prediction
Work Conditions
Interface <-> Answer Quality
Task Structuring & Routing

Extreme Elasticity: Summary

Algorithms

- Approximate Answers
- ML Libraries and Ensemble Methods
- Active Learning

Machines

- Cloud Computing – esp. Spot Instances
- Multi-tenancy
- Relaxed (eventual) consistency/ Multi-version methods

People

- Dynamic Task and Microtask Marketplaces
- Visual analytics
- Manipulative interfaces and mixed mode operation

POST-HADOOP APPROACHES: FLINK/STRATOSPHERE

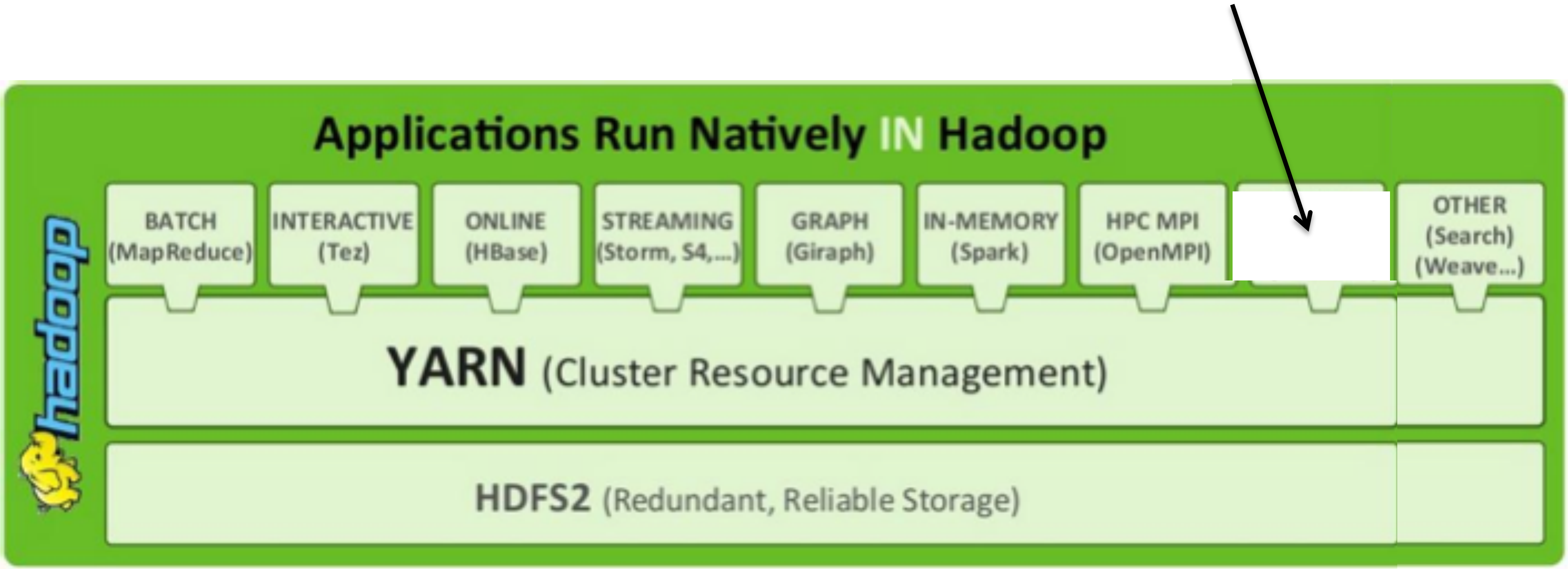
Stratosphere: General-Purpose Programming + Database Execution

Take from database Technology	Add	Take from MapReduce technology
• Declarativity • Query optimization • Robust out-of-core	• Iterations • Advanced Dataflows • General APIs	• Scalability • User-defined functions • Complex data types • Schema on read

Placement in Hadoop Stack

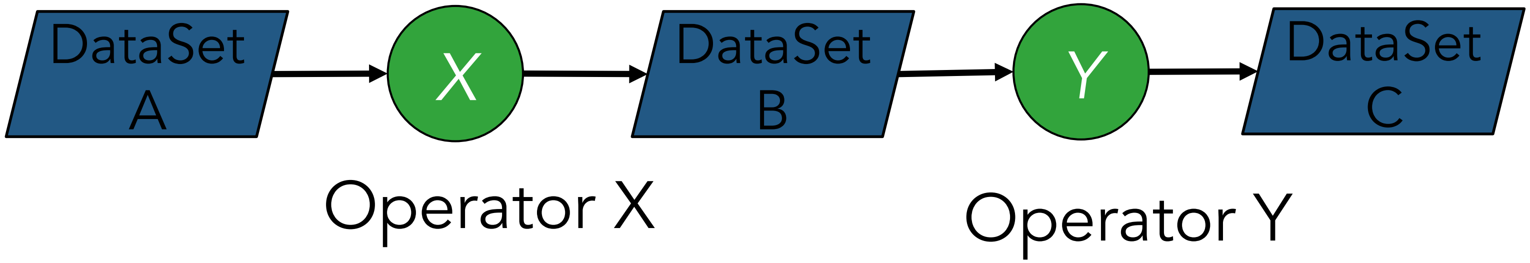
- Analyzes HDFS data directly
- Runs on top of YARN

Big Data looks tiny from Stratosphere

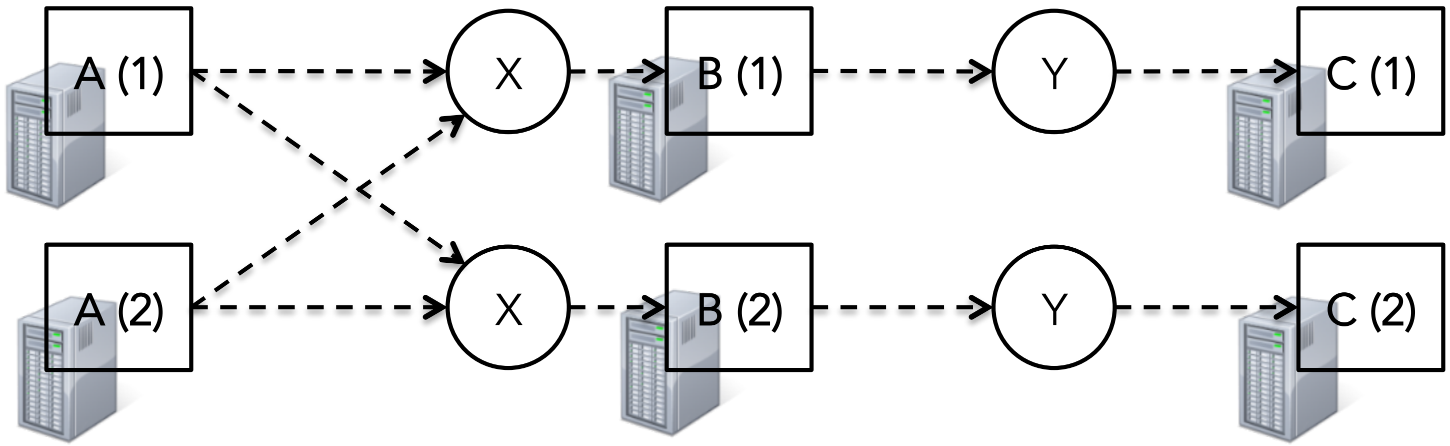


Data Sets and Operators

Program

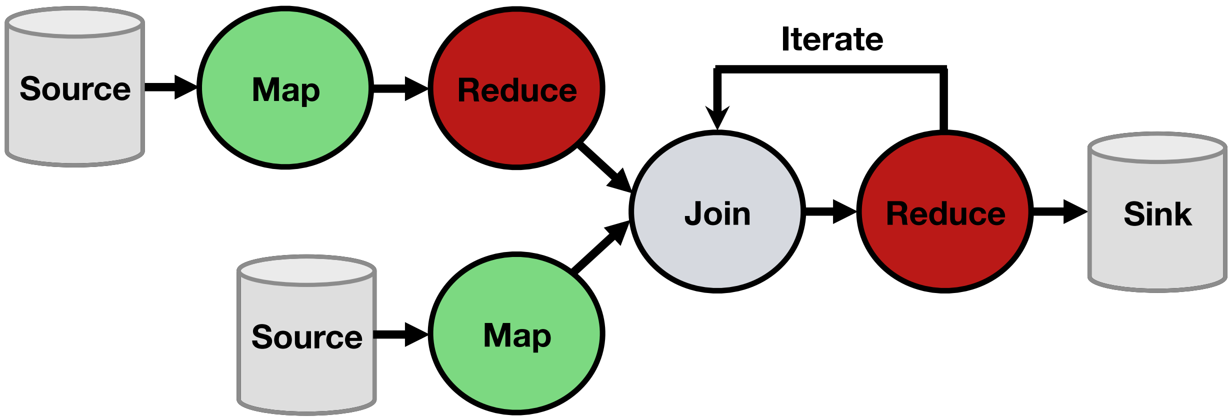


Parallel Execution



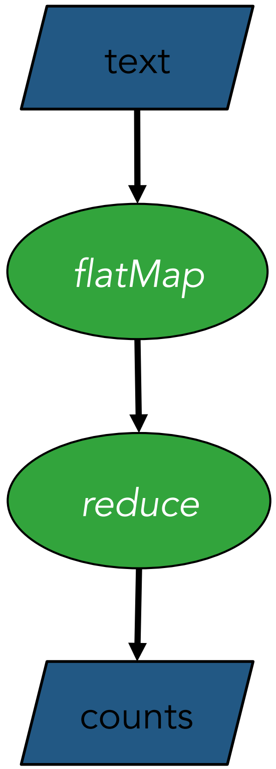
Rich Set of Operators

Map, Reduce, Join, CoGroup, Union, Iterate, Delta Iterate, Filter, FlatMap, GroupReduce, Project, Aggregate, Distinct, Vertex-Update, Accumulators



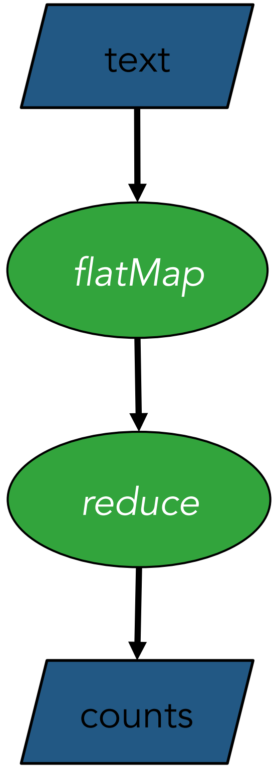
WordCount in Java

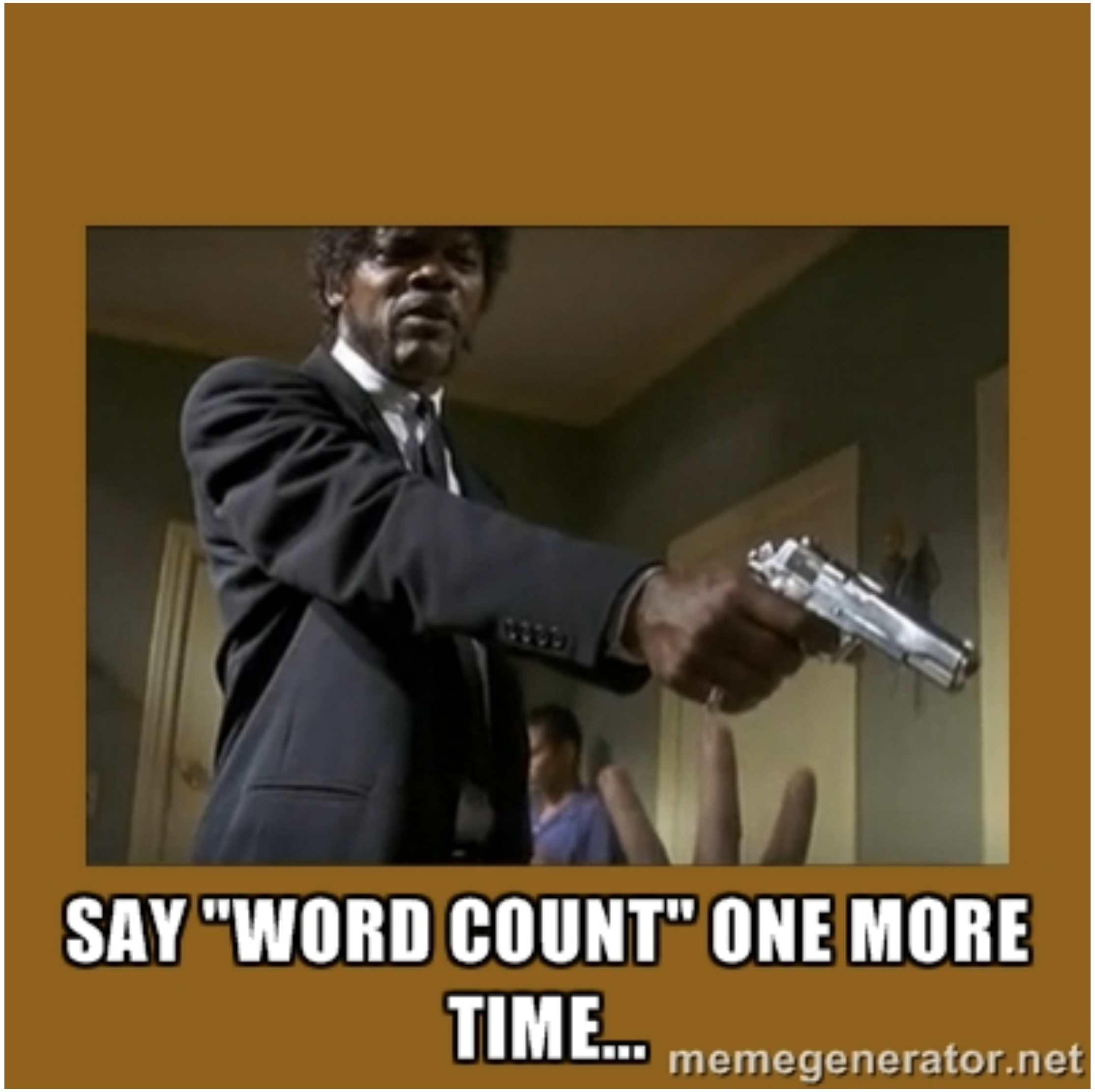
```
final ExecutionEnvironment env =
  ExecutionEnvironment.getExecutionEnvironment();

DataSet<String> text = readTextFile (input);

DataSet<Tuple2<String, Integer>> counts= text
  .flatMap(new LineSplitter())
  .groupBy(0)
  .count();

env.execute("Word Count Example");


public static final class LineSplitter extends
  FlatMapFunction<String, Tuple2<String, Integer>> {

  public void flatMap(String line, Collector<Tuple2<String, Integer>> out) {
    for (String word : line.split(" ")) {
      out.collect(new Tuple2<String, Integer>(word, 1));
    }
  }
}
```



WordCount in Scala

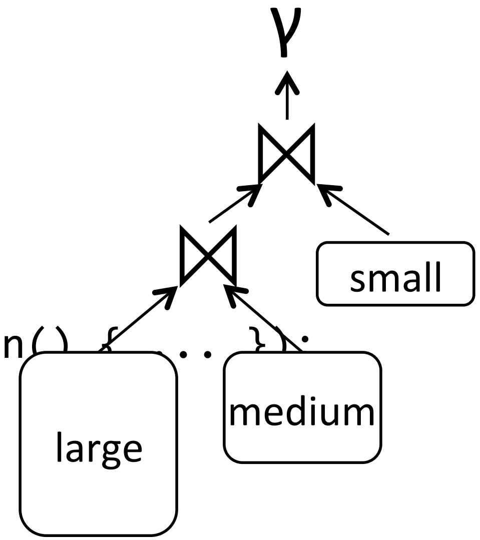
```
val input = TextFile(textInput)

val counts = input
    .flatMap { line =>
line.split("\\W+") }
    .groupBy { word => word }
    .count()
```



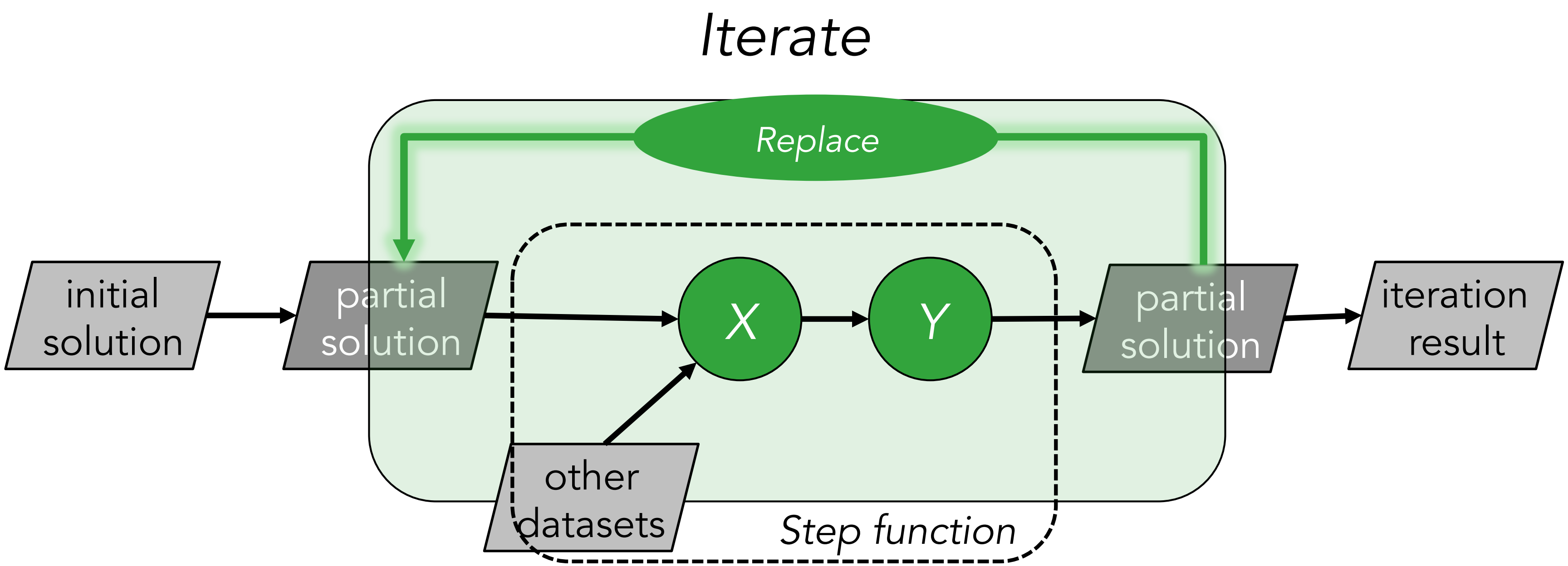

SAY "WORD COUNT" ONE MORE TIME...
memegenerator.net

Longer Operator Pipelines

```
DataSet<Tuple...> large = env.readCsv(...);
DataSet<Tuple...> medium = env.readCsv(...);
DataSet<Tuple...> small = env.readCsv(...);

DataSet<Tuple...> joined1 = large
                                .join(medium)
                                .where(3).equals(1)
                                .with(new JoinFunction() { ... });

DataSet<Tuple...> joined2 = small
                                .join(joined1)
                                .where(0).equals(2)
                                .with(new JoinFunction() { ... });

DataSet<Tuple...> result = joined2
                                .groupBy(3)
                                .max(2);
```



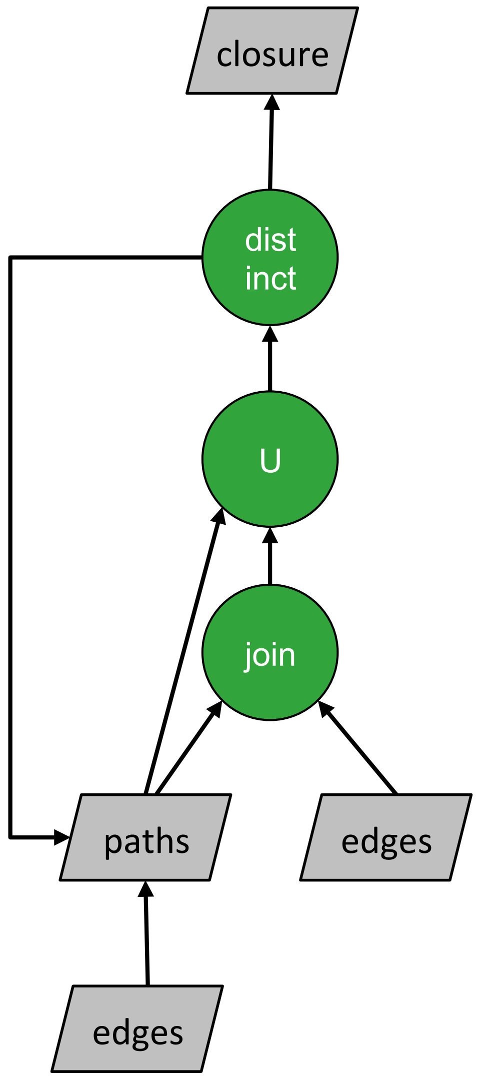
"Iterate" Operator



- Built-in operator to support looping over data
- Applies step function to partial solution until convergence
- Step function can be arbitrary Stratosphere program
- Convergence via fixed number of iterations or custom convergence criterion

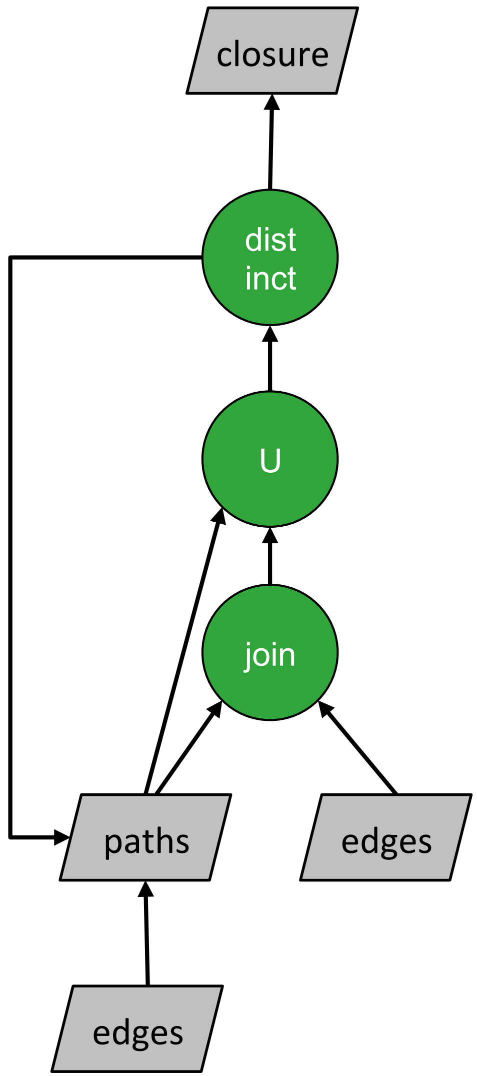
Transitive Closure in Java

```
DataSet<Tuple2<Long, Long>> edges = getEdgeDataSet(env);

IterativeDataSet<Tuple2<Long,Long>> paths = edges.iterate(maxIterations);

DataSet<Tuple2<Long,Long>> nextPaths = paths
  .join(edges)
  .where(1).equalTo(0)
  .with(new JoinFunction<Tuple2<Long, Long>,
                         Tuple2<Long, Long>,
                         Tuple2<Long, Long>>() {
           public Tuple2<Long, Long> join(Tuple2<Long, Long> left,

Tuple2<Long, Long> right)
             throws Exception {
                   return new Tuple2<Long, Long>(
                     new Long(left.f0),
                           new Long(right.f1));
                         }
  })
  .union(paths)
  .distinct();

DataSet<Tuple2<Long, Long>> transitiveClosure = paths.closeWith(nextPaths);
```

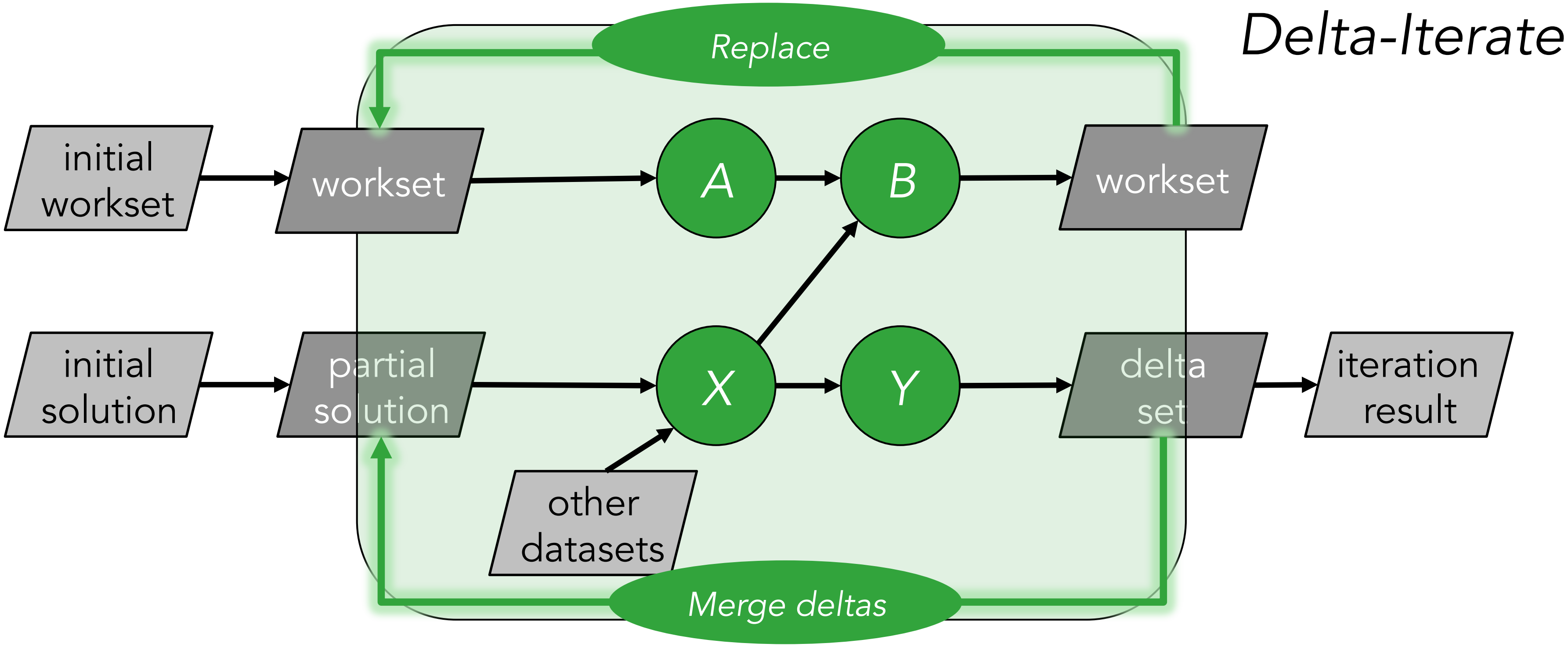


Transitive Closure in Scala

```
val edges = DataSource(…)

def createClosure(paths: DataSet[Path]) = {

  val newPaths = paths
   join edges
   where { p => p.to } isEqualTo { e => e.from }
   map joinPaths
   union paths
   groupBy { p => (p.from, p.to) }
   reduceGroup { p => p }

  newPaths
}

val transitiveClosure = edges
  .iterate(numIterations, createClosure)
```



"Delta Iterate" Operator

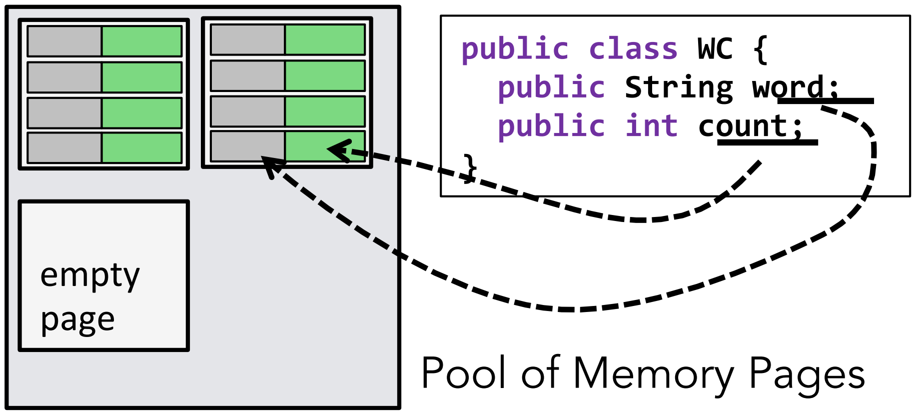


- Compute next workset and changes to the partial solution until workset is empty
- Similar to semi-naïve evaluation in datalog
- Generalizes vertex-centric computing of Pregel and GraphLab

Memory Management

Big Data looks tiny from Stratosphere

Spark



Pool of Memory Pages

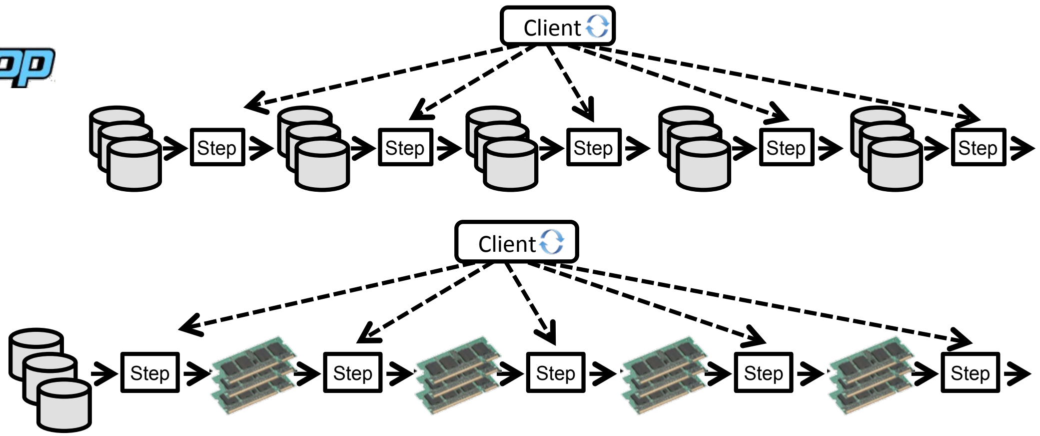
- Works on pages of bytes
- Maps objects transparently to these pages
- Full control over memory, out-of-core enabled
- Algorithms work on binary representation
- Address individual fields (not deserialize whole object)

Distributed Collection

List[WC]

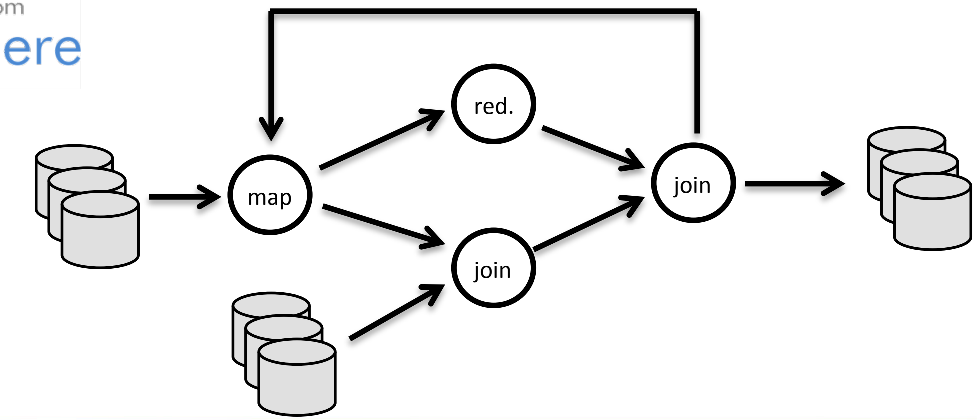
- Collections of objects
- General-purpose serializer (Java / Kryo)
- Limited control over memory & less efficient spilling

Built-in vs. Driver-Based Looping



Loop outside the system, in driver program

Iterative program looks like many independent jobs



Dataflows with feedback edges

System is iteration-aware, can optimize the job

Thank you! Questions?

gabriel.antoniu@inria.fr