

Now with  
Hadoop 2.0



**uweseiler**

codecentric 

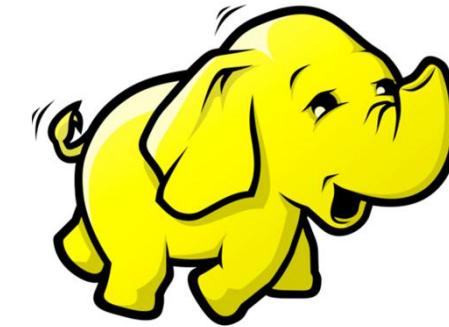


# *Introduction to the Hadoop Ecosystem*

# About me



**Big Data Nerd**



**Hadoop Trainer**



**MongoDB Author**



**Photography Enthusiast**



**Travelpirate**

# About us

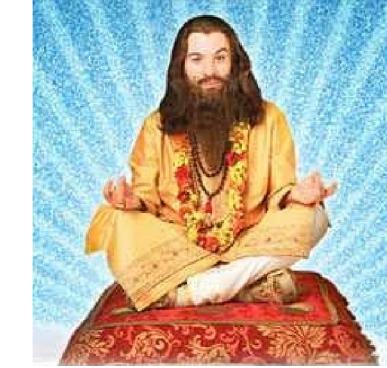
codecentric  is a bunch of...



Big Data Nerds



Agile Ninjas



Continuous Delivery Gurus



Enterprise Java Specialists   Performance Geeks



Join us!

# Agenda

- **What is Big Data & Hadoop?**
- **Core Hadoop**
- **The Hadoop Ecosystem**
- **Use Cases**
- **What's next? Hadoop 2.0!**

# Agenda

- **What is Big Data & Hadoop?**
- **Core Hadoop**
- **The Hadoop Ecosystem**
- **Use Cases**
- **What's next? Hadoop 2.0!**

Big Data

**Big Data is like teenage sex:**  
everybody talks about it,  
nobody really knows how to  
do it, everyone thinks  
everyone else is doing it, so  
everyone claims they are  
doing it...

**\$ 7 to 10 billion**

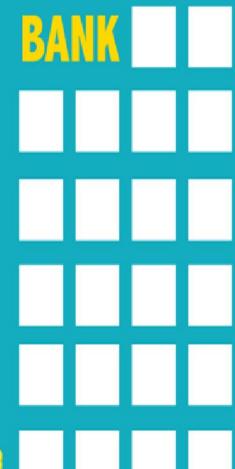
spent annually on technology by each of the **4 largest universal banks**

Around  
**3.8 petabytes**  
of data is managed per firm

## Financial Industries

**25%** of Big Data use comes from the financial industry

J.P.Morgan Chase & Co.  
**BANK**



They combine a database of

**1.5 billion**

pieces of information, with publicly available economic statistics from the U.S. government

Consumer trend reports are generated in seconds, instead of weeks or months

2 of the best techniques to improve client retention:

- ✚ Sentiment analysis
- ✚ Predictive analytics

Commercial banks collect data such as:

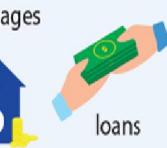
credit cards



customer information



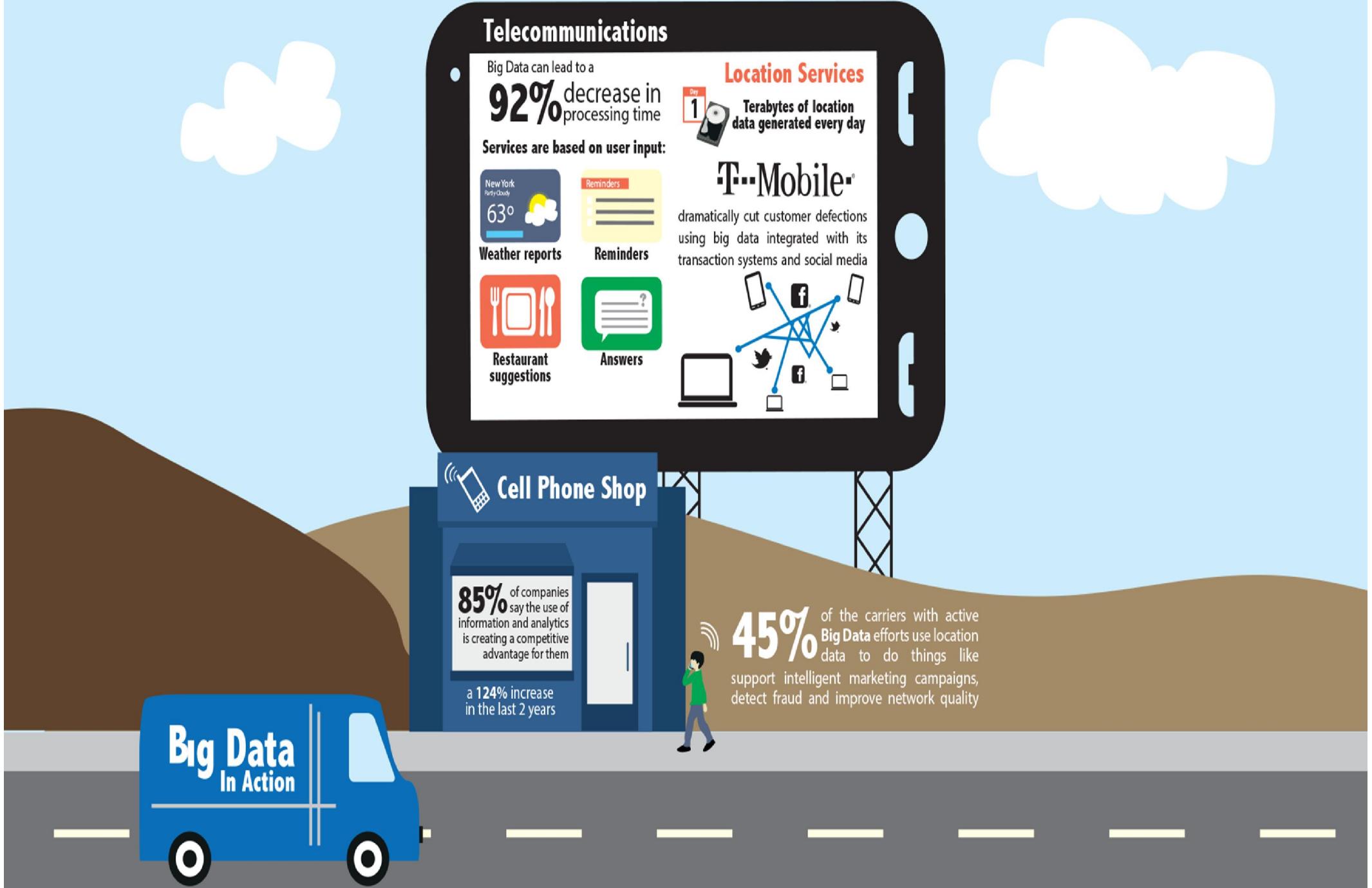
mortgages



deposits salaries



loans



# Retail Industry



collects more than  
**2.5 petabytes** of data every hour  
from its customer transactions.

A point-of-sale system holds  
**3 million to 15 million**  
transactions

**10 gigabytes**  
of data

McKinsey & Company

Estimates that a retailer using  
big data analytics to its fullest  
potential could increase its  
operating margin by  
**more than 60%**

Combining data from:



- Web browsing patterns
- Social media

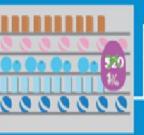


- Industry forecasts
- Existing customer records

Can help to:

- Predict trends
- Prepare for demand
- Pinpoint customers
- Optimize pricing and promotions
- Monitor real-time analytics and results

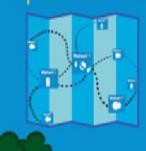
Retailers analyze:



Merchants can use POS data to:

- Reduce out-of-stock situations
- Predict the next top-selling item

Product selections  
at particular stores

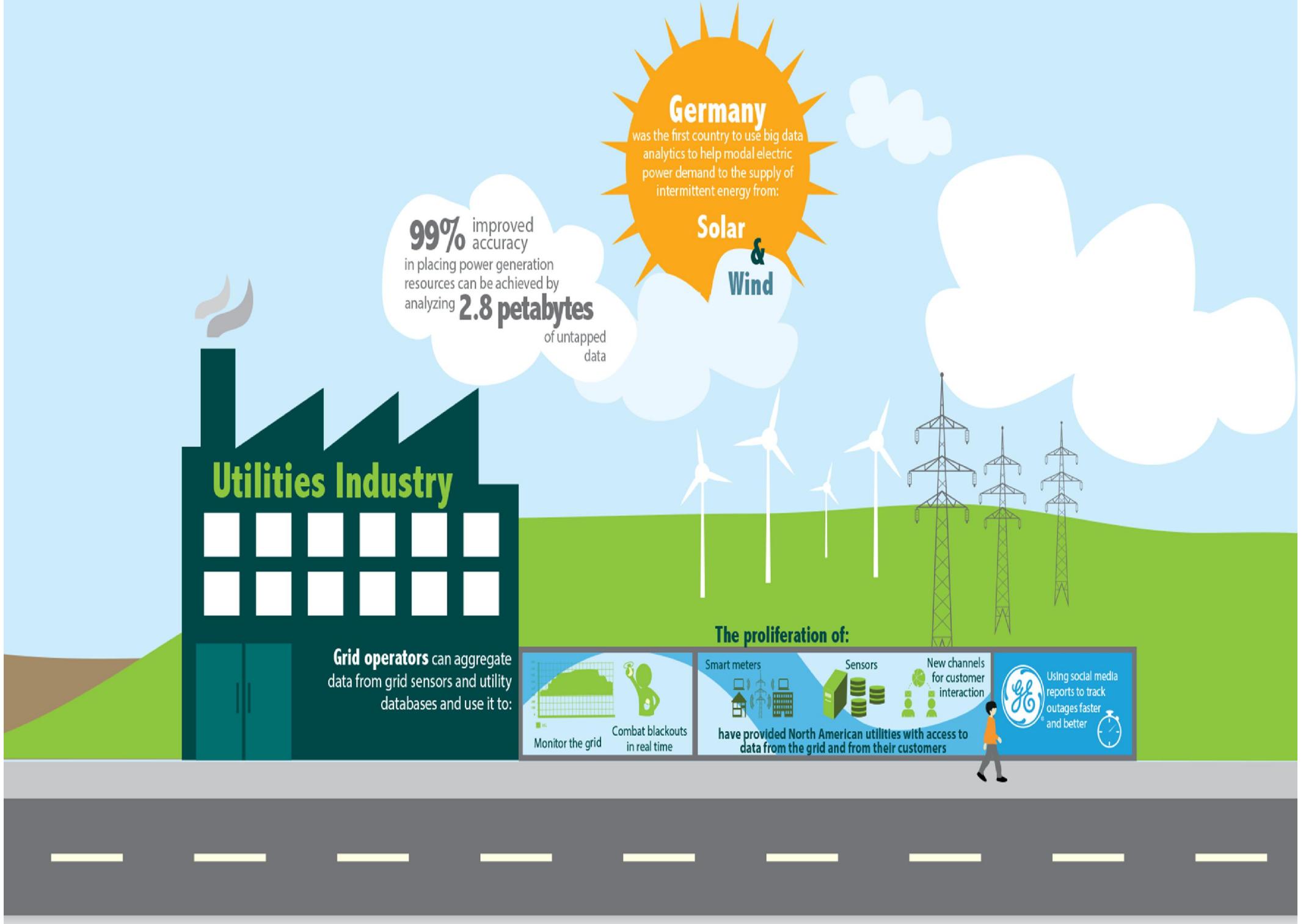


This allows them to tailor:

Product selections  
at particular stores

Timing of price markdowns





### Using text-mining algorithms, researchers:

- Culled more than 10,000 mentions
- Summarized the most relevant comments
- To make an informed decision
- While minimizing costs



## Automobile Industry

Ford Motor Co.™ scoured auto-enthusiast websites and owner forums to see what drivers were saying about turn indicators instead of launching a full-scale market-research effort



There are upwards of 10,000+ sensors per vehicle generating data including:

MPH  
MPG  
RPM  
  
Tire wear  
Oil viscosity  
Fuel efficiency  
  
Oil pressure  
Water temperature  
Engine temperature

### Big Data analysis of social media comments

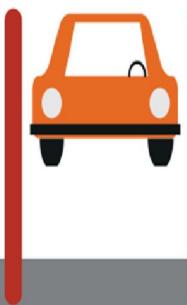
can predict trends in automotive equipment failures.



### Auto-Service

### New sources of Big Data can be leveraged to:

- Accelerate product design
- Improve vehicle performance
- Enhance the driver experience





# Sports

The Oakland A's™ used analytics to identify and obtain relatively undervalued players



VS



In 2001 they won:



2/3 of  
their  
games



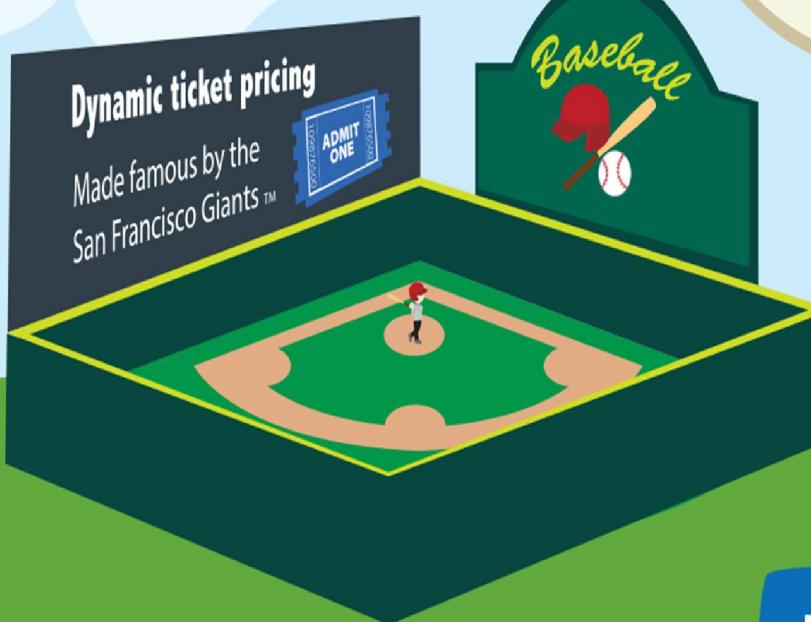
1/4 the payroll  
of the mighty  
NY Yankees™

...and nearly beat them in the MLB™ playoffs

Analytics are used to drive decisions in:



The competition is now between professional sports teams and third parties like StubHub!™





## Healthcare Industry

Researchers have found that:

A couple of weeks before there is an increase in **flu patients** coming to hospital emergency rooms in a region, there is a spike in **Google™ search requests** for terms like "flu symptoms" and "flu treatments"

More and more start-ups are searching for ways to comb through large volumes of clinical data on:

By adopting the electronic patient medical records:



Medication



Allergies



Procedures

Medical researchers can aggregate information about health care outcomes to reveal patterns that lead to:

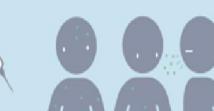
A 20% decrease in patient mortality can be achieved by analyzing patient data

Over \$300 billion in additional annual value can be added to the US Healthcare industry if big data is managed properly

More effective treatments

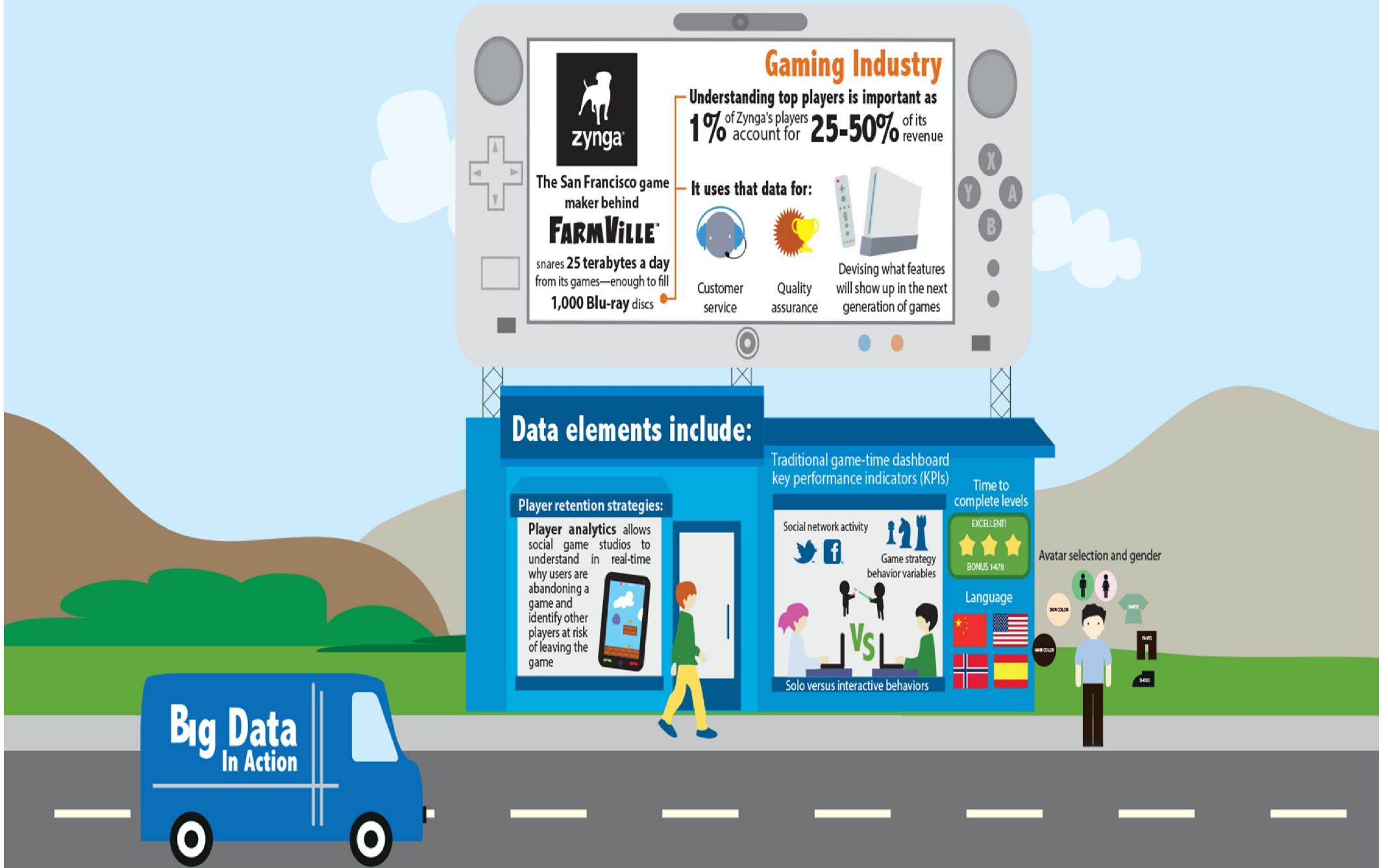


Detection of outbreaks



Kaiser Permanente has fully implemented its HealthConnect System to ensure information exchange across all medical facilities

It has reduced total office visits by 26.2%



## Online Dating Industry

Today, approximately **20% of relationships** begin through some sort of online or mobile dating service.

Online dating services, like  
**match.com**

constantly improve the algorithms for  
matching men and women on dates  
sifting through their Web listings of:

- ♥ Personal characteristics
- ♥ Communications
- ♥ Reactions

The average age of an  
online dater is 48



More than 40% of the  
country's single population  
have an online dating profile

When new users sign up, they are  
required to fill out a **400-question profile**

The system **compares** the user's answers to  
the site's **20 million users**

The process requires **a billion calculations**  
for each bachelor or bachelorette before  
providing matches



**eHarmony®**



Their dataset alone exceeds  
**4 terabytes of total data**,  
not including photos and other information

Out of eHarmony's  
44 million registered users,  
**1.2 million** have been married



# City Impact

**Oslo, Norway**  
reduced street lighting energy consumption by 62% with a smart solution provided by big data analysis



US will implement a predictive analytics and smart metering project to:

Monitor consumption remotely: expected 20% reduction in water consumption and savings of \$1 million per year



Identify water leaks

NYC has leveraged big data to double the city's hit rate in finding stores selling bootleg cigarettes; sped the removal of trees destroyed by Hurricane Sandy and helped steer overburdened housing inspectors directly to lawbreaking buildings



**Portland, Oregon** used technology to optimize the timing of its traffic signals and was able to eliminate more than 157,000 metric tons of CO<sub>2</sub> emissions in just six years – the equivalent of taking 30,000 passenger vehicles off the roads for an entire year

In Los Angeles, Xerox implemented dynamic parking systems that adjust parking rates to influence parking behavior, and the preliminary results show:



A 15% decrease in parking occupancy in the congested areas



This influenced a 10% increase in less desirable parking areas



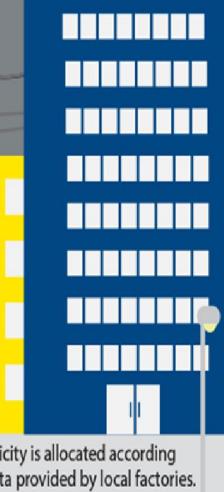
Each city is built on information technology that operates scientifically, based on data.



The smart city project of Rivas Vaciamadrid in **Spain** through combination of smart grid and energy management, access control, air quality monitoring, and traffic management, has realized:

A 50% reduction in Information and Communication Technology spending

Energy savings of 35%



# My favorite definition

**Big Data is when the  
data itself becomes  
part of the problem.**

# The classic definition

**Volume**



**Velocity**



**Variety**

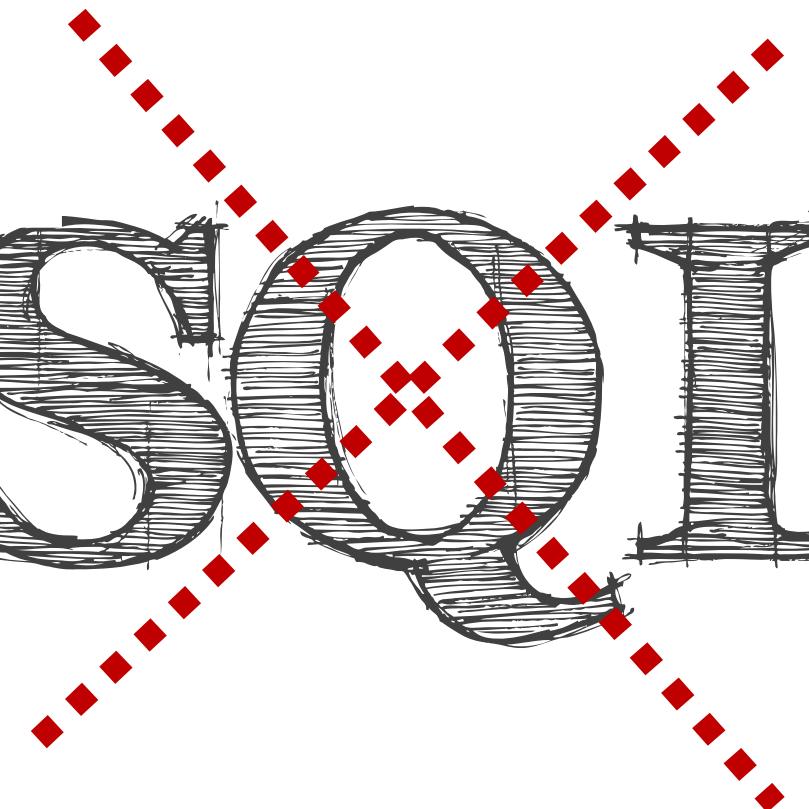


# «Big Data» != Hadoop

Not

Only

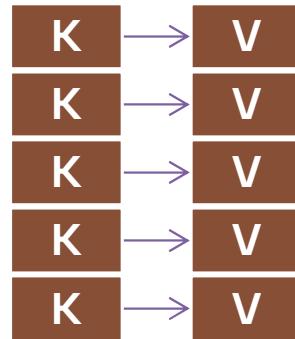
NOSOIL



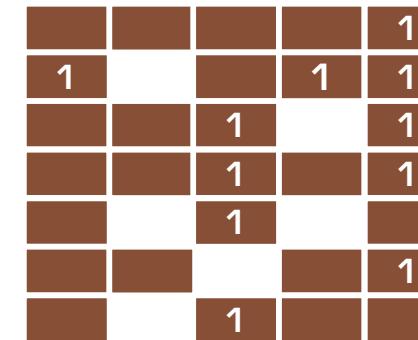
*Relational*

# Classification of NoSQL

## Key-Value Stores



## Column Stores

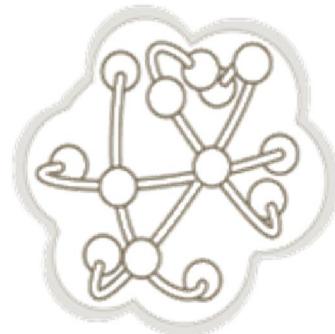


 Google Bigtable

 APACHE HBASE



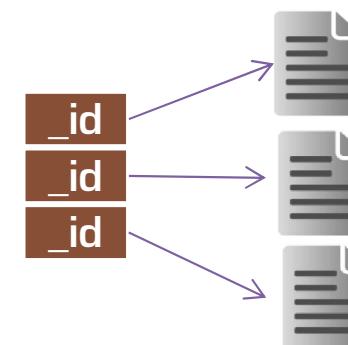
## Graph Databases



 Neo4j  
The World's Leading Graph Database

 HYPERGRAPHDB  
 \*dex

## Document Stores



 mongoDB

 CouchDB  
relax

Horizontal

Scaling

# Vertical Scaling

RAM  
CPU  
Storage

# Vertical Scaling

RAM  
CPU  
Storage

# Vertical Scaling

RAM  
CPU  
Storage

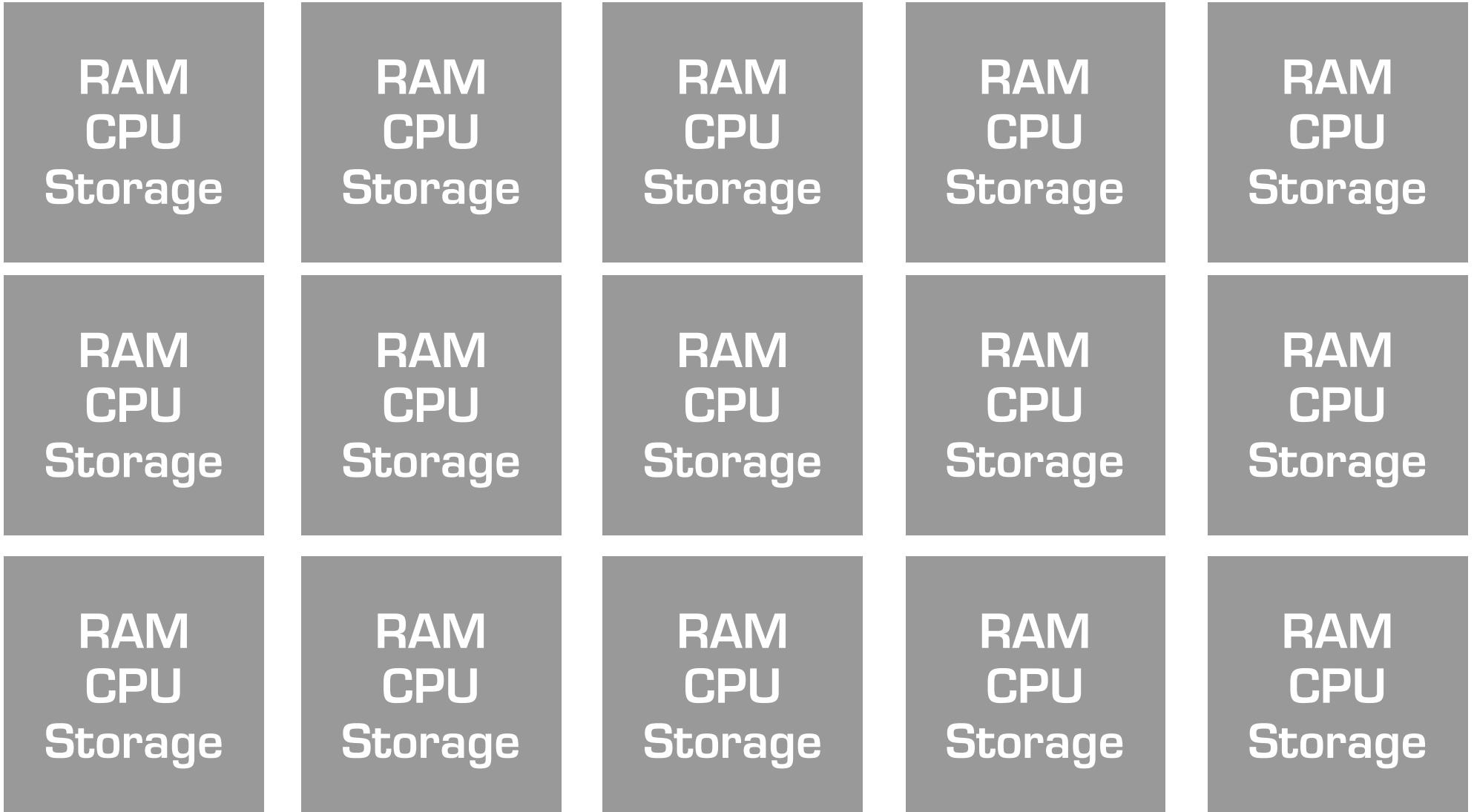
# Horizontal Scaling

RAM  
CPU  
Storage

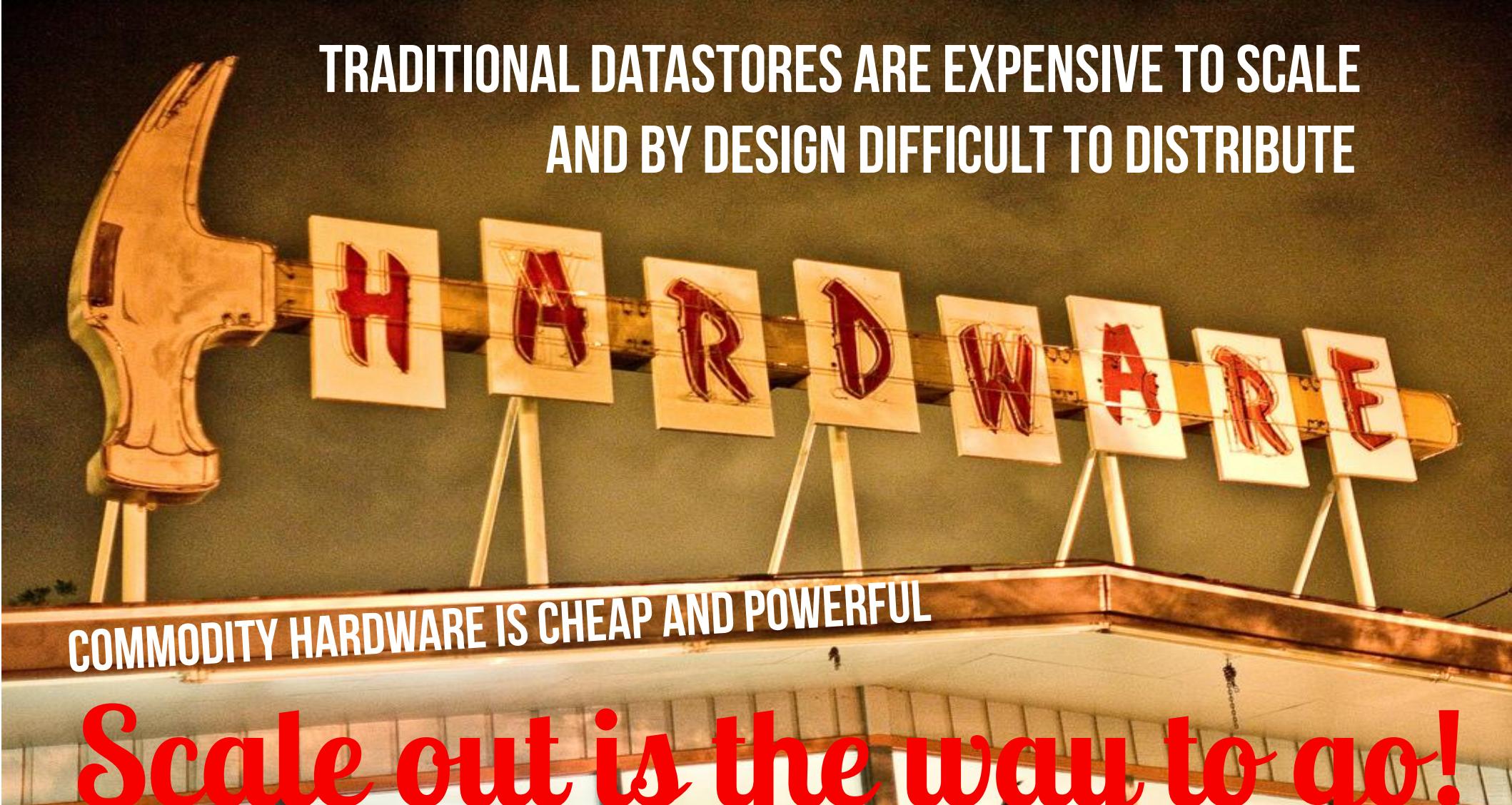
# Horizontal Scaling

RAM  
CPU  
Storage

# Horizontal Scaling



# Why Hadoop?

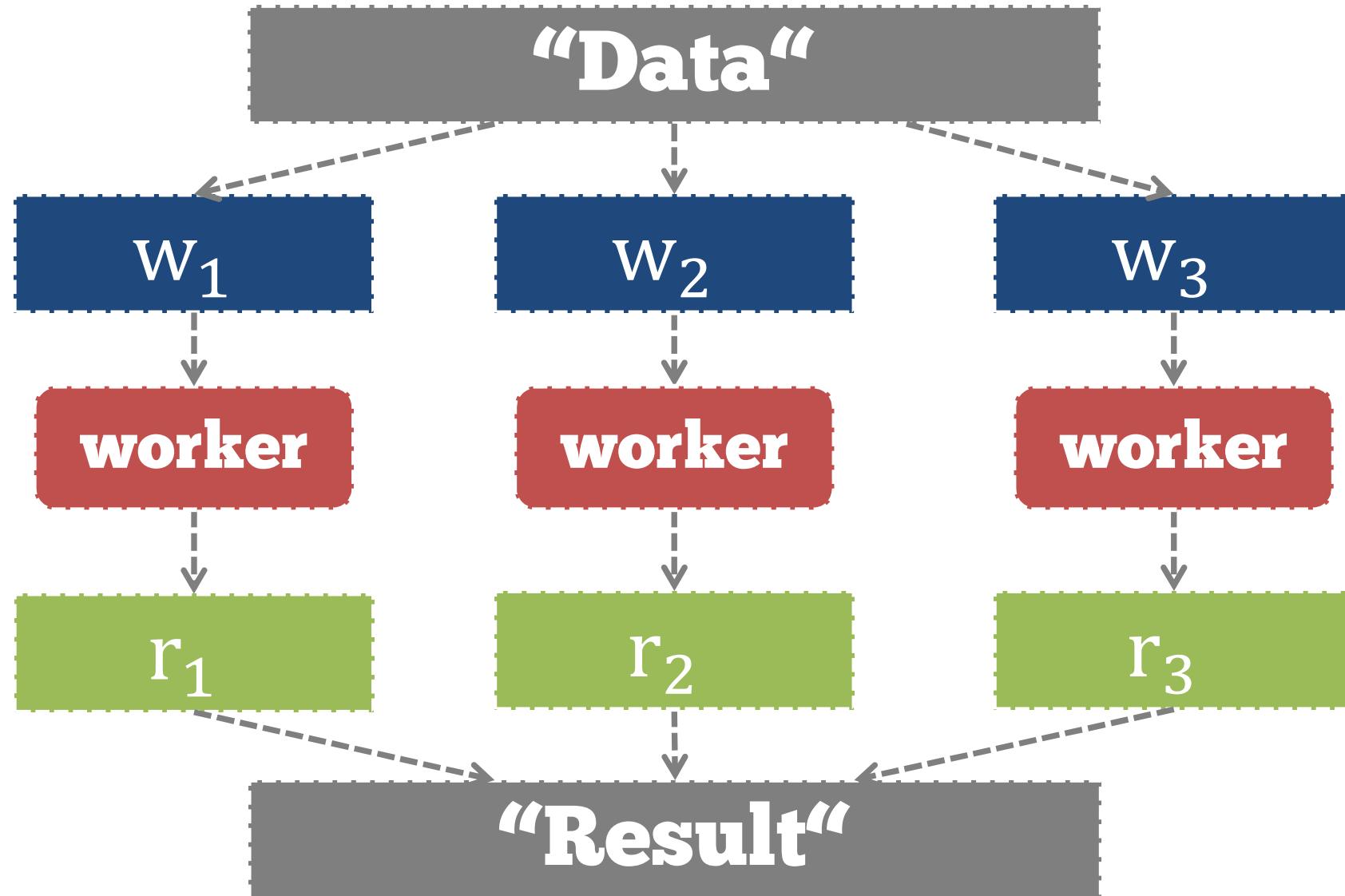


TRADITIONAL DATASTORES ARE EXPENSIVE TO SCALE  
AND BY DESIGN DIFFICULT TO DISTRIBUTE

COMMODITY HARDWARE IS CHEAP AND POWERFUL

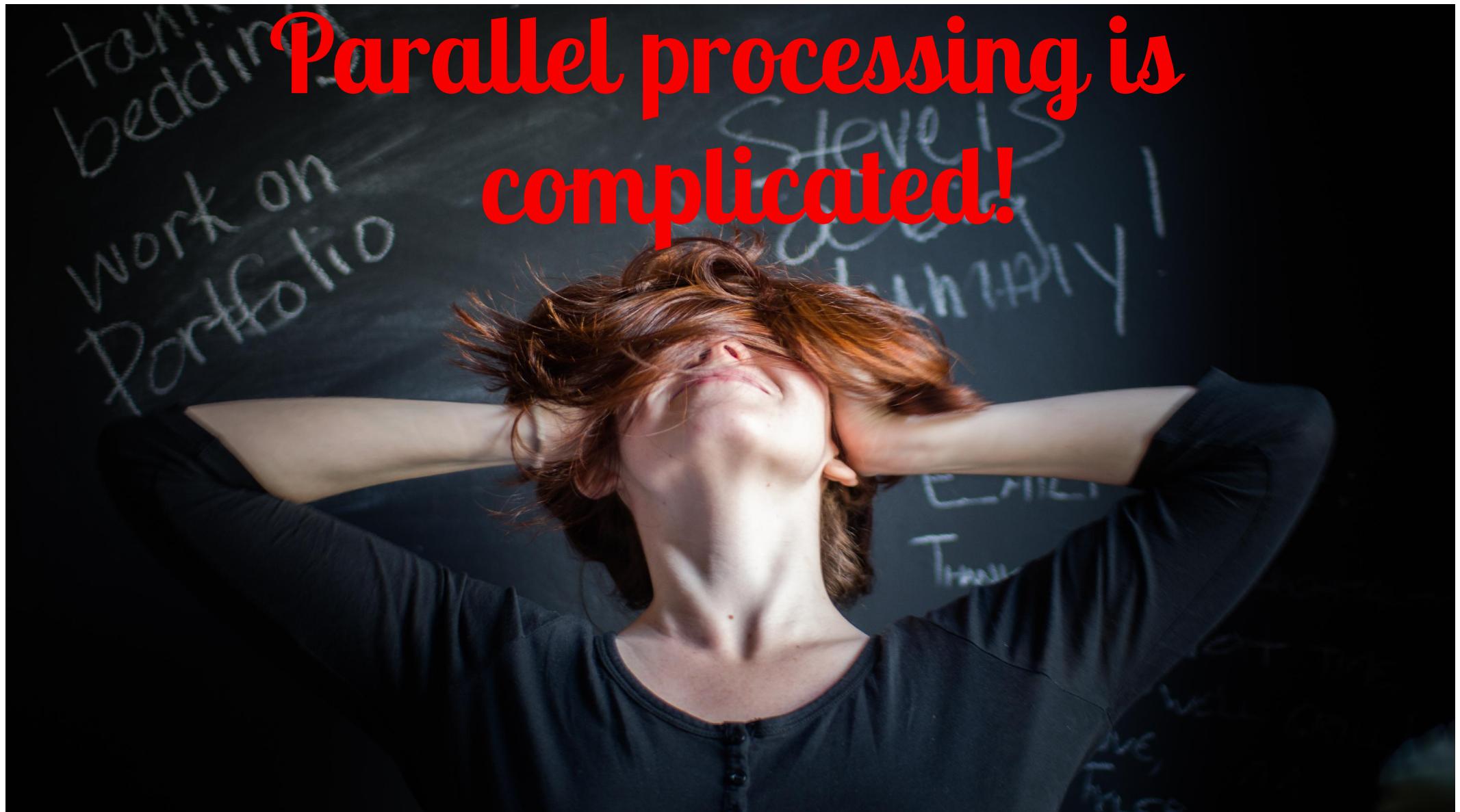
*Scale out is the way to go!*

# How to scale data?



But...

**Parallel processing is  
complicated!**



But...



*Data storage is not trivial!*

# What is Hadoop?

**Distributed Storage and  
Computation Framework**

# What is Hadoop?

Hadoop != Database

# What is Hadoop?

**“Swiss army knife  
of the 21st century”**



<http://www.guardian.co.uk/technology/2011/mar/25/media-guardian-innovation-awards-apache-hadoop>

# The Hadoop App Store



HDFS



MapRed



HCat



Pig



Hive



HBase



Ambari



Avro



Cassandra



Chukwa



Flume



Hama



HyperT



Impala



Mahout



Nutch



Oozie



Scoop



Scribe



Tez



Vertica



Whirr



ZooKee



Horton



Cloudera



MapR



EMC



Intel



IBM



Talend



Teradata



Pivotal



Informaticonica



Microsoft



Pentaho



Jasper



Sync



Kognitio



Tableau



Splunk



Platfora



Rack



Karma



Actuate



MicroStrategy

# The Hadoop App Store

## Apache Hadoop



- HDFS
- MapReduce
- Hadoop Ecosystem
- Hadoop YARN

## Hadoop Distributions



- Test & Packaging
- Installation
- Monitoring
- Business Support

less

Functionality

more



- Integrated Environment
- Visualization
- (Near-)Realtime analysis
- Modeling
- ETL & Connectors

## Big Data Suites

# Agenda

- **What is Big Data & Hadoop?**
- **Core Hadoop**
- **The Hadoop Ecosystem**
- **Use Cases**
- **What's next? Hadoop 2.0!**

# Data Storage

**OK, first things  
first!**

**I want to **store** all of  
my <<Big Data>>**

# Data Storage

There is an app for that!



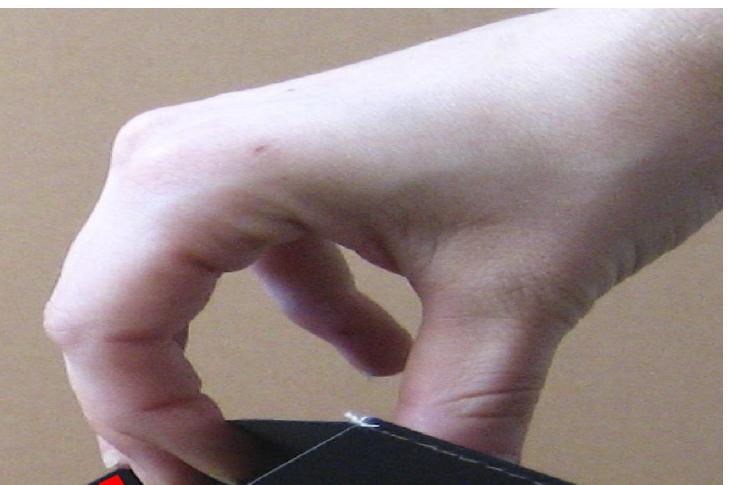
# Hadoop Distributed File System

- **Distributed file system for redundant storage**
- **Designed to reliably store data on commodity hardware**
- **Built to expect hardware failures**

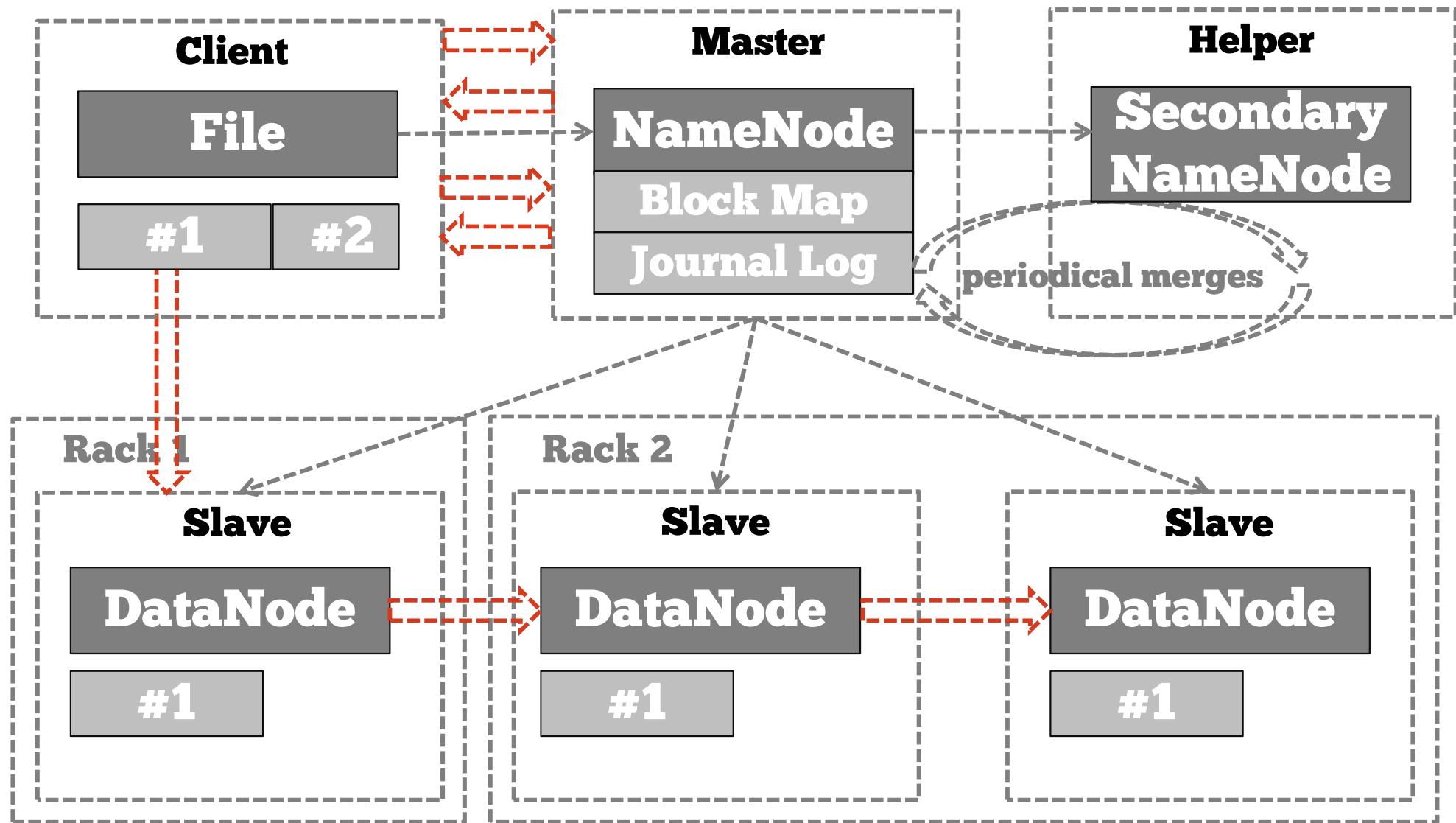
## Intended for

- **large files**
- **batch inserts**

*Write once, read many times*



# HDFS Architecture



# Let's have a look ...

# Data stored, check!

Now I want to  
create insights  
from my data!

# Data Processing

There is an app for that!



# MapReduce

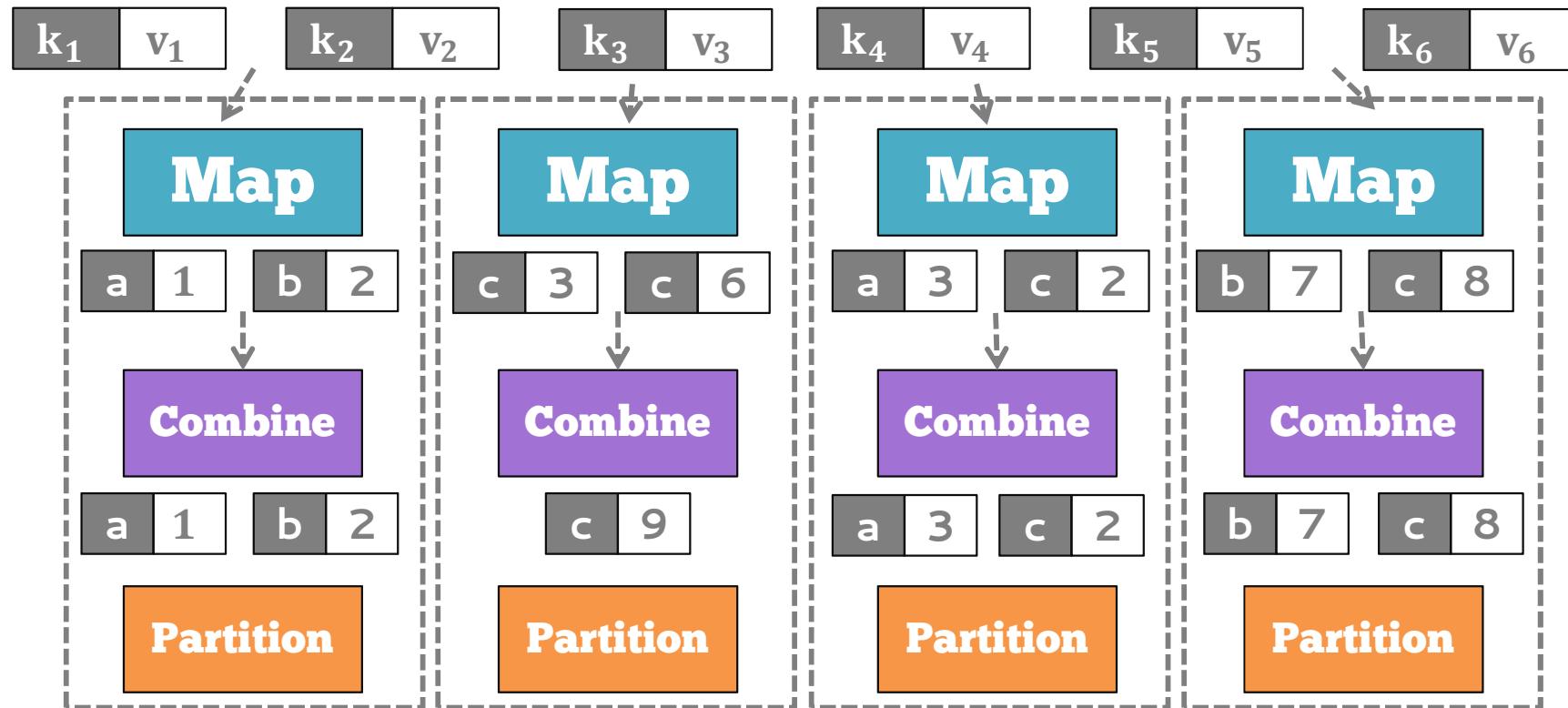
- **Programming model for distributed computations at a massive scale**
- **Execution framework for organizing and performing such computations**
- **Data locality is king**

# Typical large-data problem

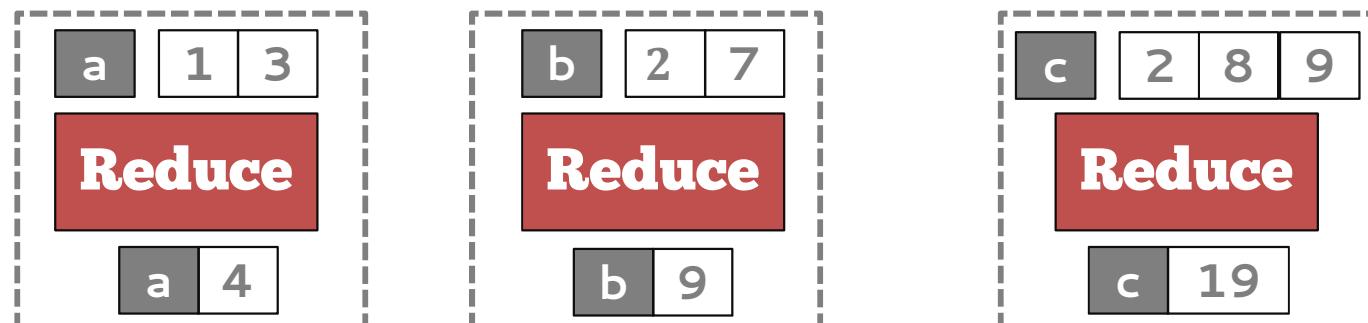
- **Iterate over a large number of records**
- **Extract something of interest from each**
- **Shuffle and sort intermediate results**
- **Aggregate intermediate results**
- **Generate final output**



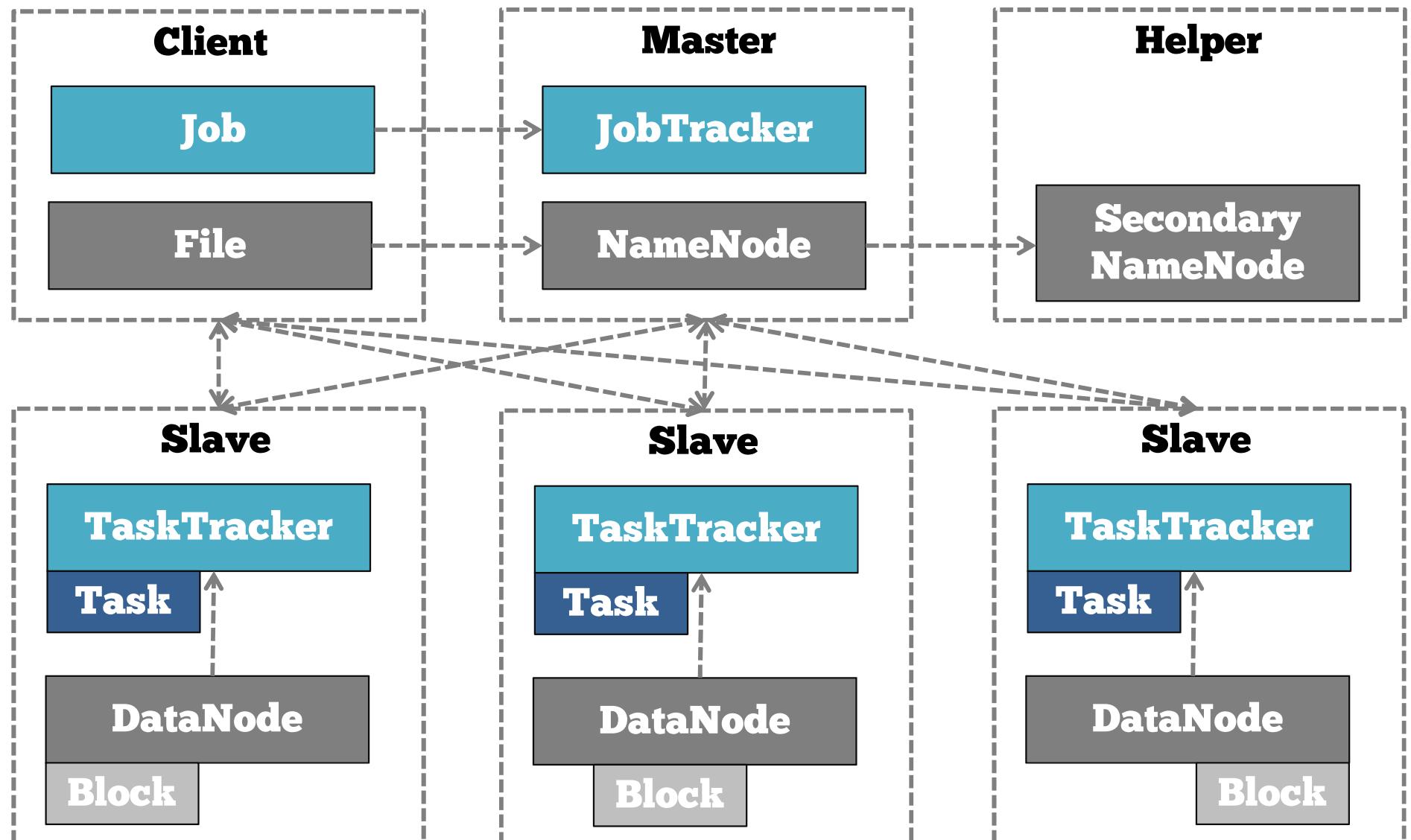
# MapReduce Flow



## Shuffle and Sort



# Combined Hadoop Architecture



# Word Count Mapper in Java

```
public class WordCountMapper extends MapReduceBase implements
    Mapper<LongWritable, Text, Text, IntWritable>
{
    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(LongWritable key, Text value, OutputCollector<Text,
        IntWritable> output, Reporter reporter) throws IOException
    {
        String line = value.toString();
        StringTokenizer tokenizer = new StringTokenizer(line);
        while (tokenizer.hasMoreTokens())
        {
            word.set(tokenizer.nextToken());
            output.collect(word, one);
        }
    }
}
```

# Word Count Reducer in Java

```
public class WordCountReducer extends MapReduceBase
    implements Reducer<Text, IntWritable, Text, IntWritable>
{
    public void reduce(Text key, Iterator values, OutputCollector
                      output, Reporter reporter) throws IOException
    {
        int sum = 0;
        while (values.hasNext())
        {
            IntWritable value = (IntWritable) values.next();
            sum += value.get();
        }
        output.collect(key, new IntWritable(sum));
    }
}
```

# Map/Reduce

**Let's have a look ...**

# Agenda

- **What is Big Data & Hadoop?**
- **Core Hadoop**
- **The Hadoop Ecosystem**
- **Use Cases**
- **What's next? Hadoop 2.0!**

## Java for MapReduce?

I dunno, dude...

I'm more of a  
**scripting guy...**

# Scripting for Hadoop

There is an app for that!

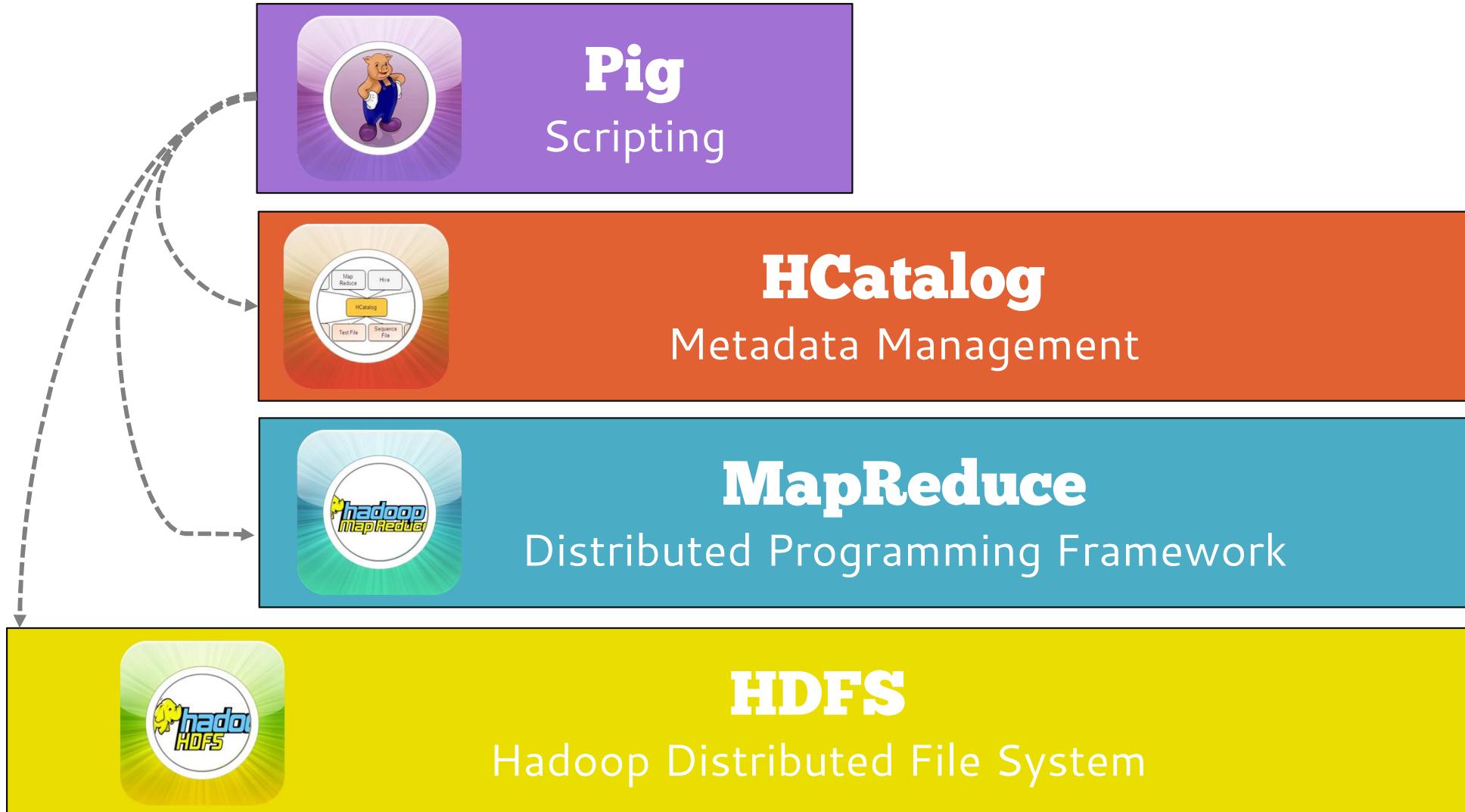


# Apache Pig

- **High-level data flow language**
- **Made of two components:**
  - **Data processing language Pig Latin**
  - **Compiler to translate Pig Latin to MapReduce**

*Abstracts you from specific details and allows you to focus on data processing*

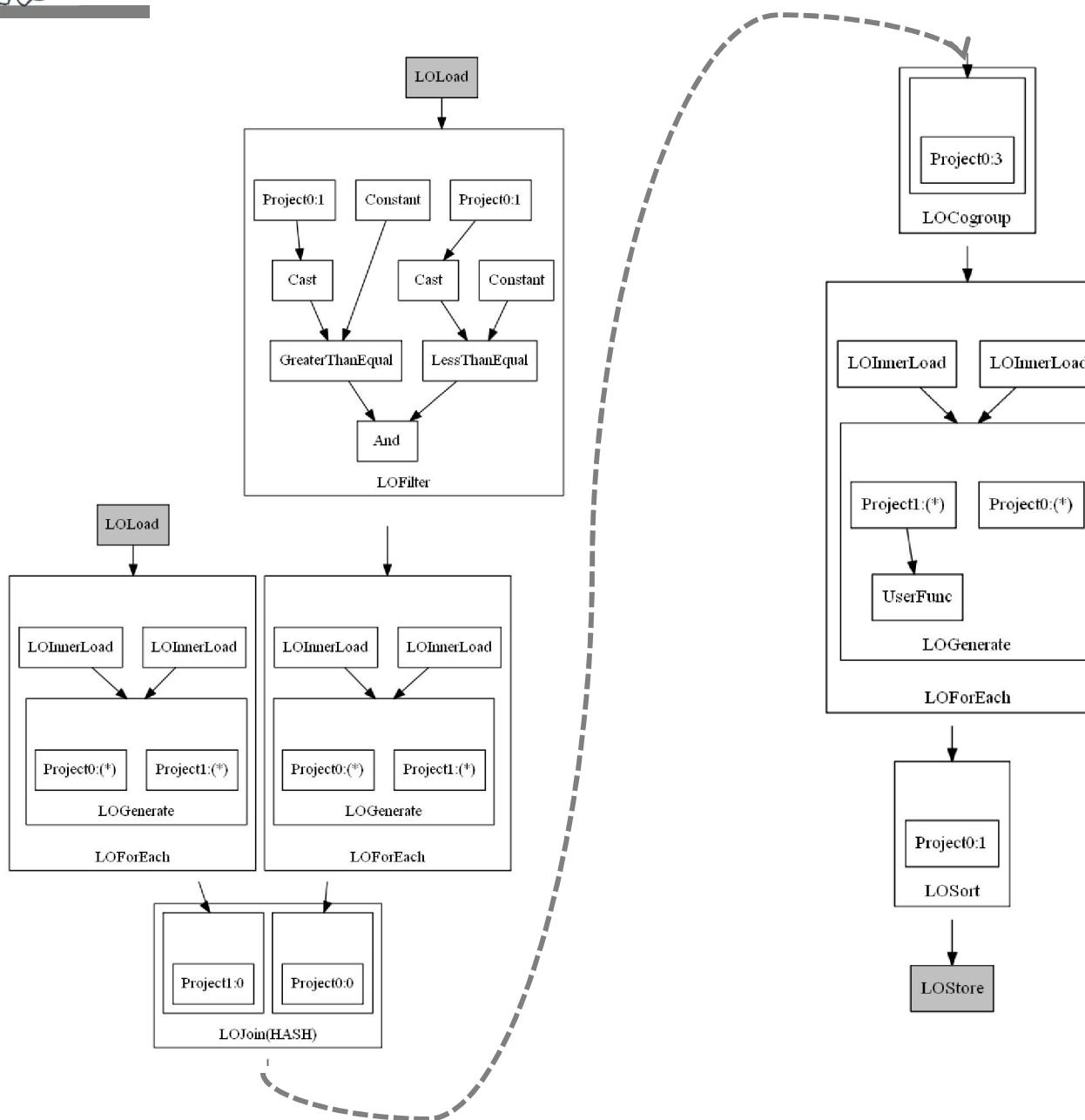
# Pig in the Hadoop ecosystem



# Pig Latin

```
users = LOAD 'users.txt' USING PigStorage(',') AS (name,  
          age);  
  
pages = LOAD 'pages.txt' USING PigStorage(',') AS (user,  
          url);  
  
filteredUsers = FILTER users BY age >= 18 and age <=50;  
joinResult = JOIN filteredUsers BY name, pages by user;  
grouped = GROUP joinResult BY url;  
summed = FOREACH grouped GENERATE group,  
          COUNT(joinResult) as clicks;  
sorted = ORDER summed BY clicks desc;  
top10 = LIMIT sorted 10;  
  
STORE top10 INTO 'top10sites';
```

# Pig Execution Plan



# Try that with Java...

```

import java.io.IOException;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

import org.apache.hadoop.io.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.Writable;
import org.apache.hadoop.io.WritableComparable;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.mapred.KeyValueTextInputFormat;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.RecordReader;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;
import org.apache.hadoop.mapred.SequenceFileInputFormat;
import org.apache.hadoop.mapred.TextInputFormat;
import org.apache.hadoop.mapred.jobcontrol.Job;
import org.apache.hadoop.mapred.jobcontrol.JobControl;
import org.apache.hadoop.mapred.lib.IdentityMapper;
import org.apache.hadoop.mapred.lib.IdentityReducer;

public class NRExample {
    public static class LoadPages extends MapReduceBase
        implements Mapper<LongWritable, Text, Text, Text> {
        public void map(LongWritable k, Text val,
                        OutputCollector<Text, Text> o,
                        Reporter reporter) throws IOException {
            // Pull the key from the value
            String line = val.toString();
            int firstComma = line.indexOf(',');
            String key = line.substring(0, firstComma);
            String value = line.substring(firstComma + 1);
            Text outKey = new Text(key);
            // Prepend an index to the value so we know which file
            // it came from.
            Text outVal = new Text("1" + value);
            o.collect(outKey, outVal);
        }
    }

    public static class LoadAndFilterUsers extends MapReduceBase
        implements Mapper<LongWritable, Text, Text, Text> {
        public void map(LongWritable k, Text val,
                        OutputCollector<Text, Text> o,
                        Reporter reporter) throws IOException {
            // Pull the key from the value
            String line = val.toString();
            int firstComma = line.indexOf(',');
            String key = line.substring(0, firstComma);
            int age = Integer.parseInt(key);
            if (age < 18 || age > 25) return;
            String key2 = line.substring(0, firstComma);
            Text outKey = new Text(key2);
            // Prepend an index to the value so we know which file
            // it came from.
            Text outVal = new Text("2" + value);
            o.collect(outKey, outVal);
        }
    }

    public static class Join extends MapReduceBase
        implements Reducer<Text, Text, Text, Text> {
        public void reduce(Text key,
                           Iterator<Text> iter,
                           OutputCollector<Text, Text> o,
                           Reporter reporter) throws IOException {
            // For each value, figure out which file it's from and
            // store it
            // accordingly.
            List<String> first = new ArrayList<String>();
            List<String> second = new ArrayList<String>();
            while (iter.hasNext()) {
                Text t = iter.next();
                String value = t.toString();
                if (value.charAt(0) == '1')
                    first.add(value.substring(1));
                else
                    second.add(value.substring(1));
            }
            first.add(second.get(0));
            o.collect(key, first);
        }
    }

    public static class LoadJoined extends MapReduceBase
        implements Mapper<Text, Text, Text, LongWritable> {
        public void map(Text k,
                       Text val,
                       OutputCollector<Text, LongWritable> o,
                       Reporter reporter) throws IOException {
            // Find the url
            String line = val.toString();
            int firstComma = line.indexOf(',');
            int secondComma = line.indexOf(',', firstComma + 1);
            String urlString = line.substring(firstComma + 1, secondComma);
            // drop the rest of the record, I don't need it anymore
            // just pass a 1 for the combiner/reducer to sum instead
            Text outKey = new Text(urlString);
            o.collect(outKey, new LongWritable(1));
        }
    }

    public static class ReduceOne extends MapReduceBase
        implements Reducer<Text, LongWritable, WritableComparable,
        Writable> {
        public void reduce(Text key,
                           Iterator<LongWritable> iter,
                           OutputCollector<WritableComparable, Writable> o,
                           Reporter reporter) throws IOException {
            // Add up all the values we see
            long sum = 0;
            while (iter.hasNext()) {
                sum += iter.next().get();
            }
            reporter.setStatus("OK");
        }
        o.collect(key, new LongWritable(sum));
    }

    public static class LoadClicks extends MapReduceBase
        implements Mapper<LongWritable, Text, WritableComparable,
        Text> {
        public void map(
            LongWritable key,
            WritableComparable val,
            OutputCollector<LongWritable, Text> o,
            Reporter reporter) throws IOException {
            o.collect(key, val);
        }
    }

    public static class LimitClicks extends MapReduceBase
        implements Reducer<LongWritable, Text, LongWritable, Text> {
        int count = 0;
        public void reduce(
            LongWritable key,
            Iterator<Text> iter,
            OutputCollector<LongWritable, Text> o,
            Reporter reporter) throws IOException {
            // Only output the first 100 records
            while (count < 100 && iter.hasNext()) {
                o.collect(key, iter.next());
                count++;
            }
        }
    }

    public static void main(JobControl args) throws IOException {
        JobConf j = new JobControl(NRExample.class);
        j.setJobName("Load Pages");
        j.setInputFormat(TextInputFormat.class);
    }
}

```

```

ip.setGetPathKey(ip);
ip.setGetUserClass(Text.class);
ip.setMapperClass(Text.class);
ip.setInputFormat(new InputFormat(ip));
FileInputFormat.addInputPath(ip, new
Paths("/user/system/pages"));
FileInputFormat.setOutputPath(ip,
  new Path("/user/gates/tmp/filtered_pages"));
ip.setMapperTasks(10);
Job loadPages = new Job(ip);

JobConf job = new JobConf(MREexample.class);
itf.setJobName("Load and Filter Users");
itf.setInputFormat(TextInputFormat.class);
itf.setOutputKeyClass(Text.class);
itf.setOutputValueClass(Text.class);
FileInputFormat.addInputPath(itf, new
Paths("/user/system/users"));
FileInputFormat.setOutputPath(itf, new
Paths("/user/gates/tmp/filtered_users"));
itf.setMapperTasks(10);
Job loadUsers = new Job(itf);

JobConf join = new JobConf(MREexample.class);
join.setJobName("Join User and Page");
join.setInputFormat(TextInputFormat.class);
join.setMapperKeyClass(Text.class);
join.setMapperValueClass(Text.class);
join.setMapperClass(JoinMapper.class);
join.setReducesClass(JoinReducer.class);
FileInputFormat.addInputPath(join, new
Paths("/user/gates/tmp/filtered_pages"));
FileInputFormat.addInputPath(join, new
Paths("/user/gates/tmp/filtered_users"));
FileInputFormat.setOutputPath(join, new
Paths("/user/gates/tmp/join"));
join.setMapperTasks(10);
Job joinJob = new Job(join);
joinJob.addDependingJob(loadPages);
joinJob.addDependingJob(loadUsers);

JobConf group = new JobConf(MRE example.class);
group.setJobName("Group Top 10");
group.setMapperFormat(TextMapperFormat.class);
group.setMapperPutTextClass(Text.class);
group.setMapperPutValueClass(Text.class);
group.setOutputFormat(SequenceFileOutputFormat.class);
group.setMapperClass(LoadMapper.class);
group.setCombinerClass(LimitCombiner.class);
group.setReducerClass(LimitReducer.class);
FileInputFormat.addInputPath(group, new
Paths("/user/gates/tmp/join"));
FileInputFormat.setOutputPath(group, new
Paths("/user/gates/tmp/grouped"));
group.setMapperTasks(10);
Job groupJob = new Job(group);
groupJob.addDependingJob(joinJob);

JobConf top10 = new JobConf(MREexample.class);
top10.setJobName("Top 10 sites");
top10.setMapperFormat(TextMapperFormat.class);
top10.setMapperPutTextClass(Text.class);
top10.setMapperPutValueClass(Text.class);
top10.setOutputFormat(SequenceFileOutputFormat.class);
top10.setMapperClass(LimitMapper.class);
top10.setCombinerClass(LimitCombiner.class);
top10.setReducerClass(LimitReducer.class);
FileInputFormat.addInputPath(top10, new
Paths("/user/gates/tmp/grouped"));
FileOutputFormat.setOutputPath(top10, new
Paths("/user/gates/top10sites"));
top10.setMapperTasks(10);
Job limit = new Job(top10);
limit.addDependingJob(groupJob);

JobControl jc = new JobControl("Find top 10 sites for users
10 to 25");
jc.addJob(loadPages);
jc.addJob(loadUsers);
jc.addJob(joinJob);
jc.addJob(groupJob);
jc.addJob(limit);
jc.run();

```

# Let's have a look ...

**OK, Pig seems quite  
useful...**

**But I'm more of a  
SQL person...**

# SQL for Hadoop

There is an app for that!



# Apache Hive

- **Data Warehousing Layer on top of Hadoop**
- **Allows analysis and queries using a SQL-like language**

Hive is best for data analysts familiar with SQL who need to do dynamic queries, summarization and data analysis

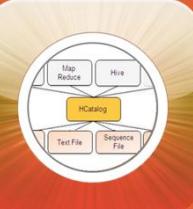
# Hive in the Hadoop ecosystem



**Pig**  
Scripting



**Hive**  
Query



**HCatalog**  
Metadata Management

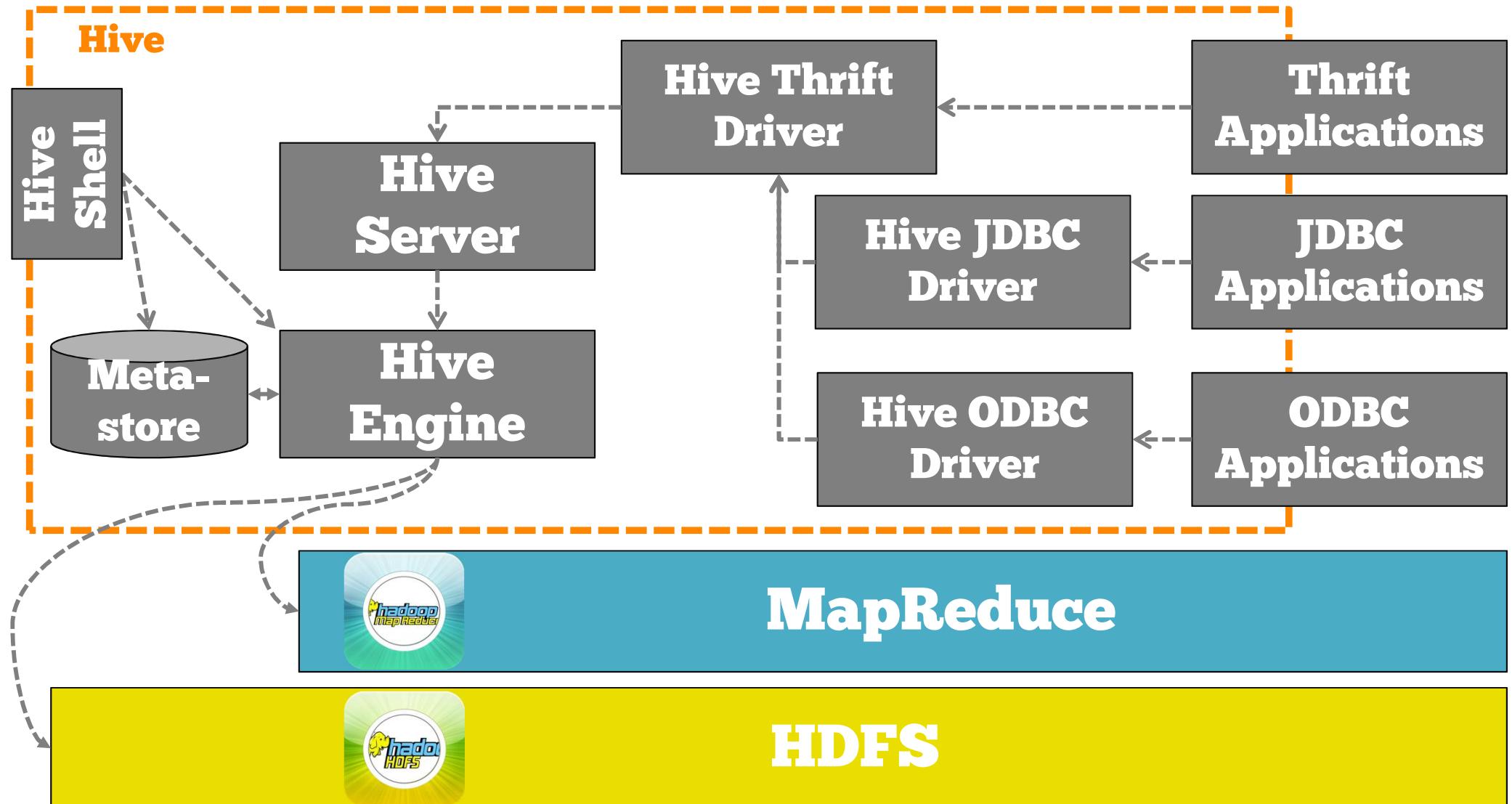


**MapReduce**  
Distributed Programming Framework



**HDFS**  
Hadoop Distributed File System

# Hive Architecture



# Hive Example

```
CREATE TABLE users(name STRING, age INT);  
CREATE TABLE pages(user STRING, url STRING);  
  
LOAD DATA INPATH '/user/sandbox/users.txt' INTO  
TABLE 'users';  
  
LOAD DATA INPATH '/user/sandbox/pages.txt' INTO  
TABLE 'pages';  
  
SELECT pages.url, count(*) AS clicks FROM users JOIN  
pages ON (users.name = pages.user)  
WHERE users.age >= 18 AND users.age <= 50  
GROUP BY pages.url  
SORT BY clicks DESC  
LIMIT 10;
```

**Let's have a look ...**

But wait, there's still more!

# More components of the **Hadoop Ecosystem**



# Mahout

Machine Learning



NoSQL Database

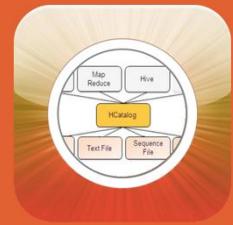


**Pig**  
Scripting



**Hive**

SQL-like queries



## HCatalog

Metadata Management



## MapReduce

Data processing



## HDFS

Data storage



## Scoop

Import & Export of  
relational data



## Flume

Import & Export of  
data flows



Cluster Coordination  
**ZooKeeper**



Cluster installation & management



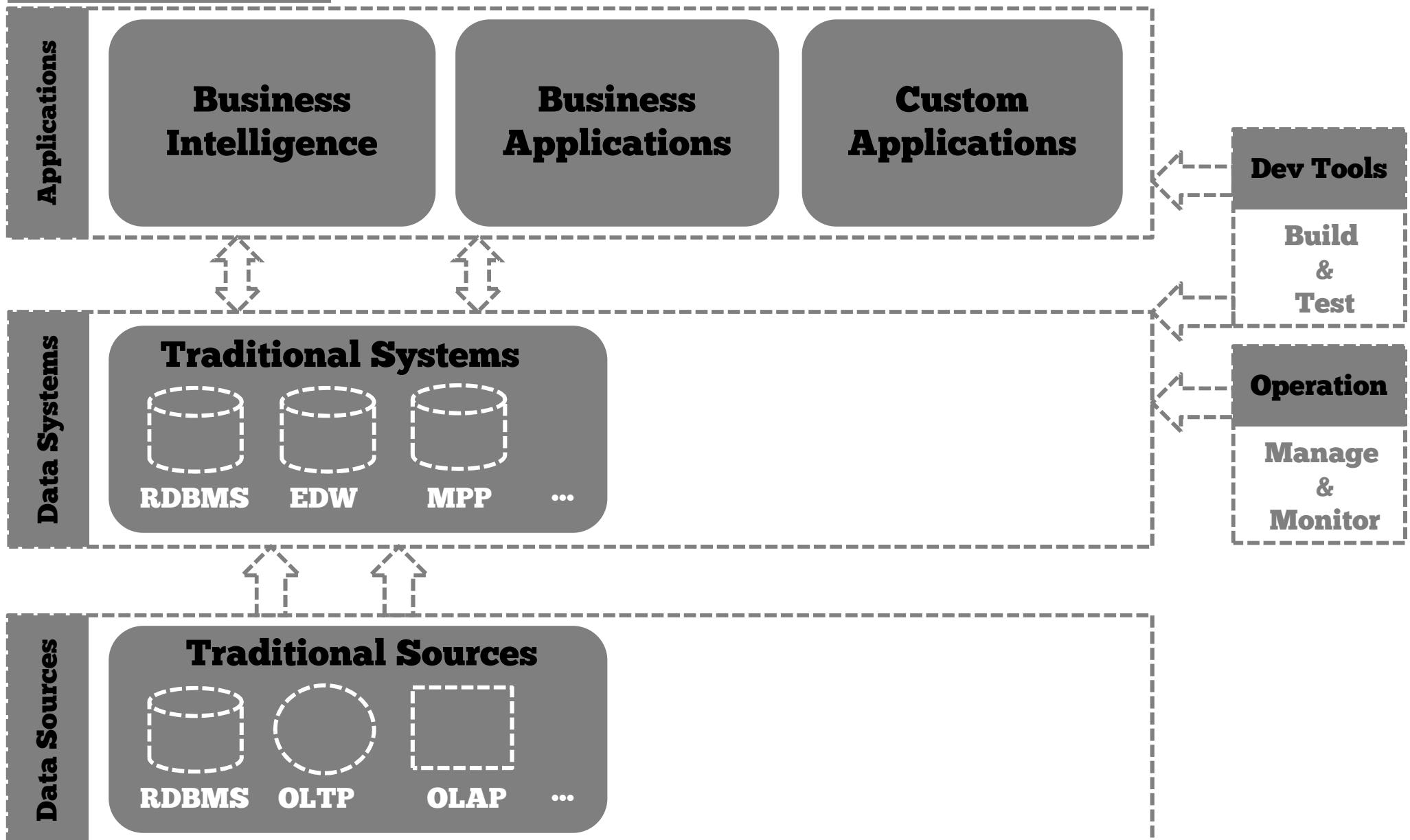
Workflow automation

**Oozie**

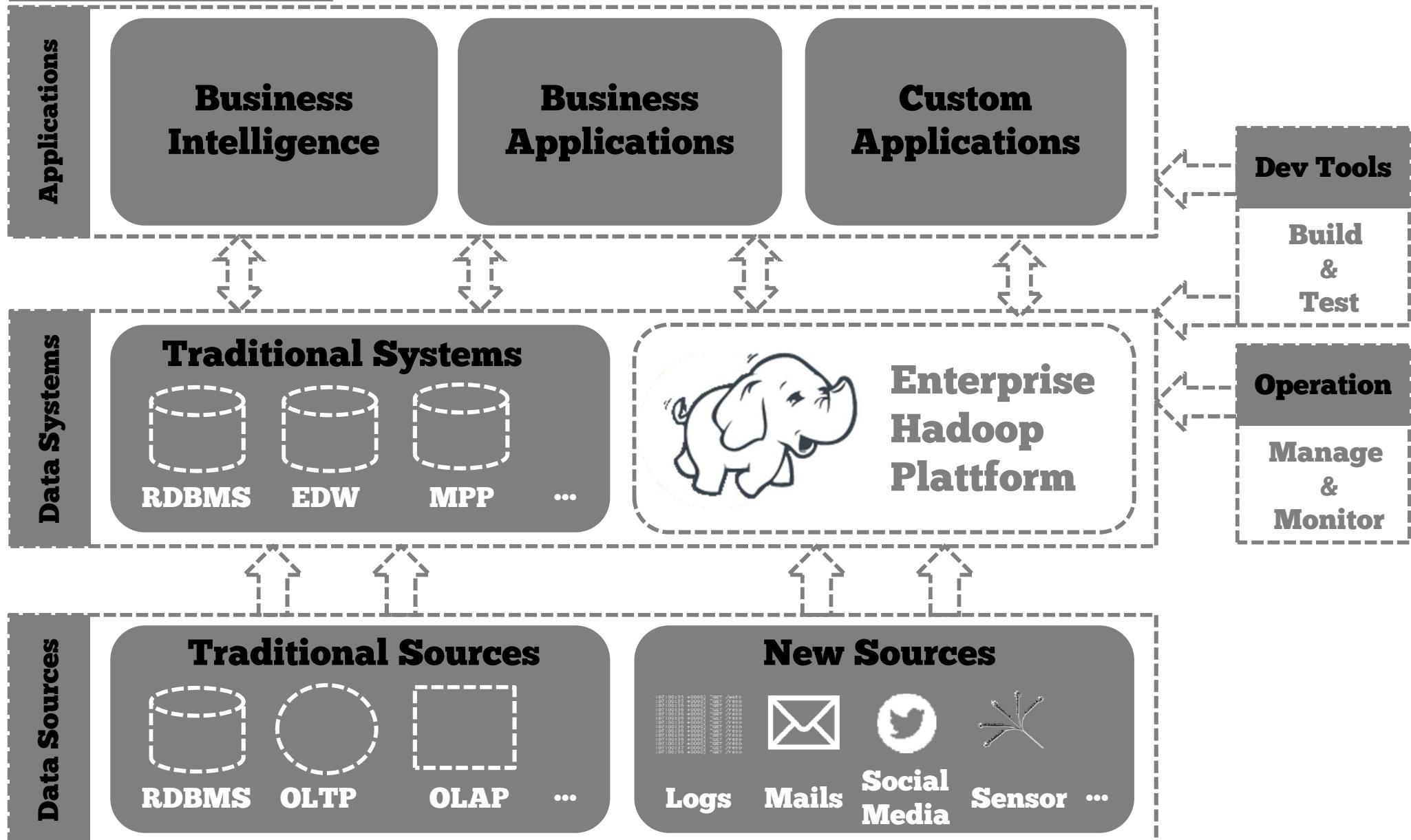
# Agenda

- **What is Big Data & Hadoop?**
- **Core Hadoop**
- **The Hadoop Ecosystem**
- **Use Cases**
- **What's next? Hadoop 2.0!**

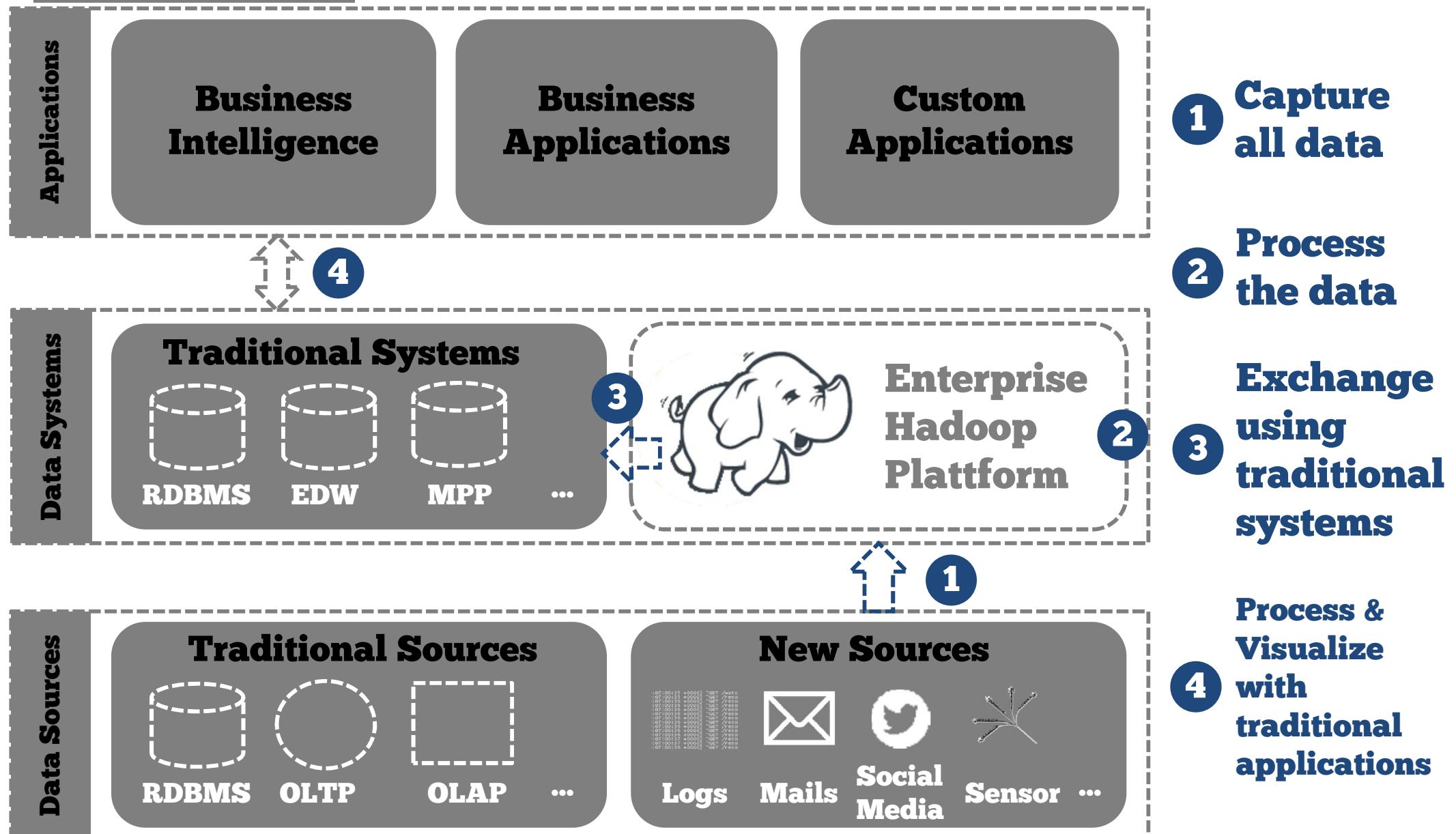
# Classical enterprise platform



# Big Data Platform



# Pattern #1: Refine data



# Pattern #2: Explore data

Applications

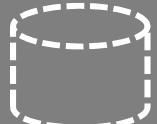
**Business Intelligence**

**Business Applications**

**Custom Applications**

Data Systems

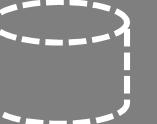
**Traditional Systems**



RDBMS



EDW



MPP

...

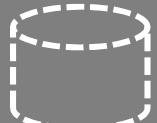
2



**Enterprise Hadoop  
Plattform**

Data Sources

**Traditional Sources**



RDBMS



OLTP



OLAP

...

Logs

**New Sources**



Mails



Social  
Media



Sensor

...

**1 Capture all data**

**2 Process the data**

**3 Explore the data using applications with support for Hadoop**

# Pattern #3: Enrich data

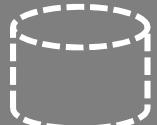
Applications

**Business Applications**

**Custom Applications**

Data Systems

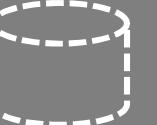
**Traditional Systems**



RDBMS



EDW

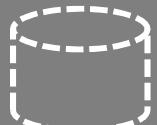


MPP

...

Data Sources

**Traditional Sources**



RDBMS



OLTP



OLAP

...

**New Sources**



Logs



Mails



Social Media



Sensor ...

**1 Capture all data**

**2 Process the data**

**3 Directly ingest the data**

2

3



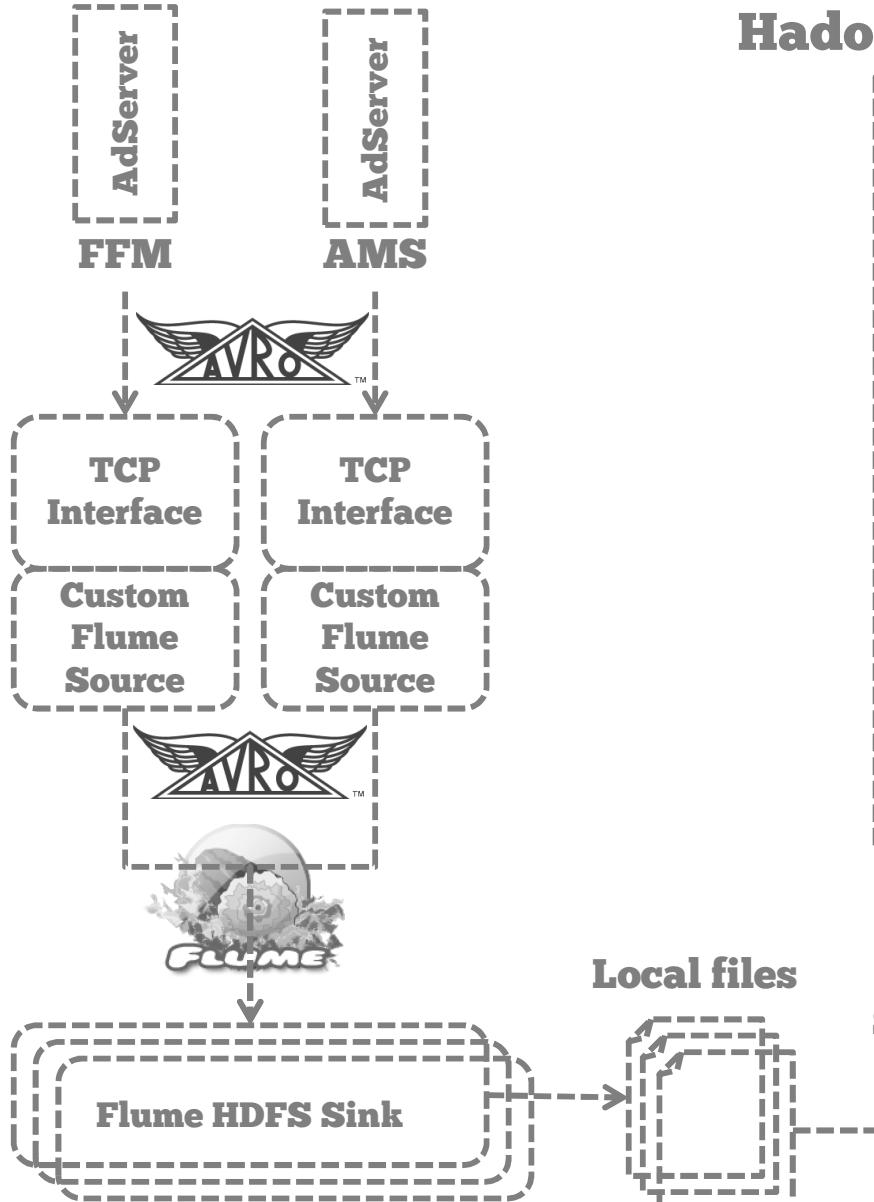
1

# Bringing it all together...

# One example...

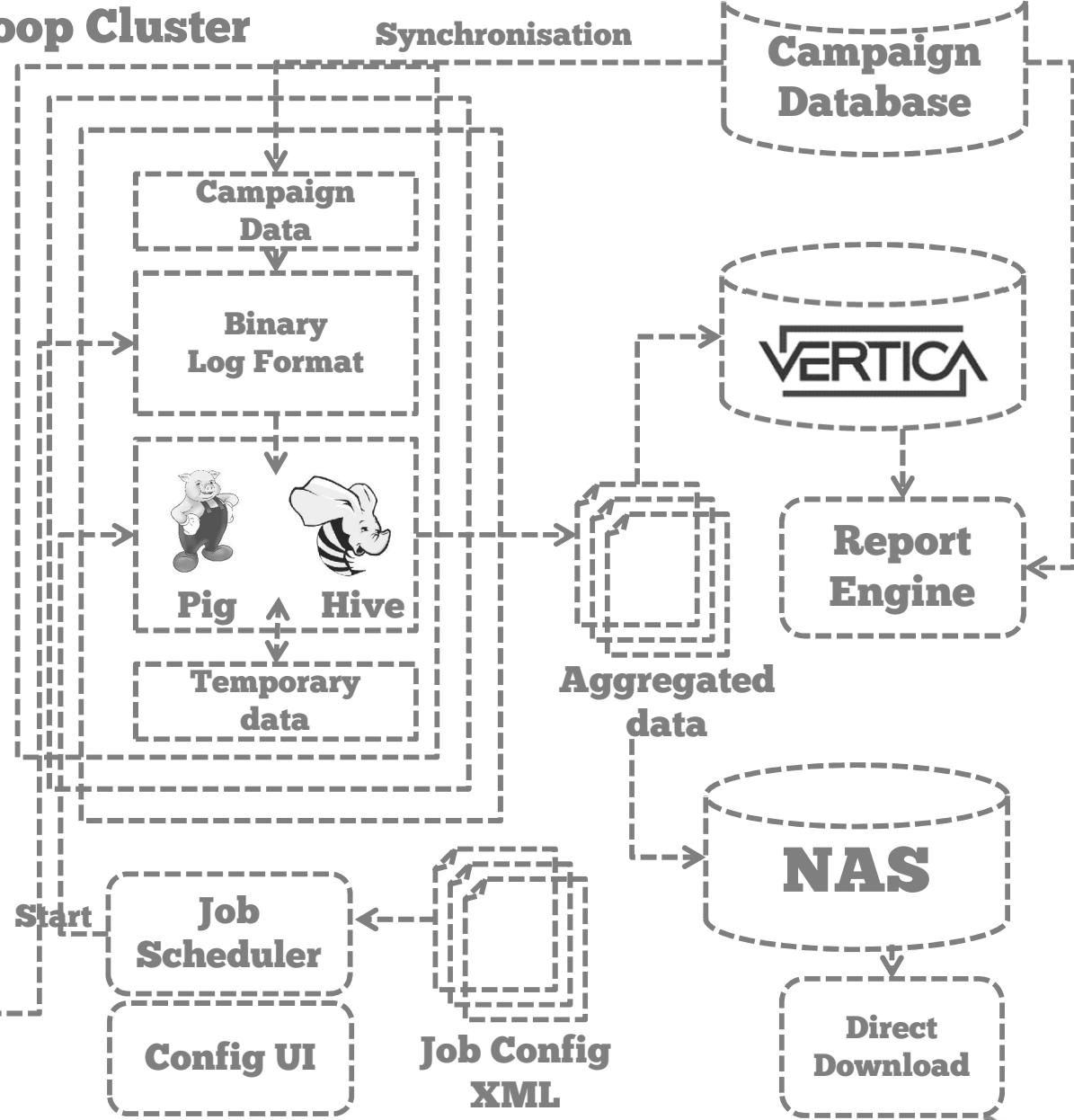
# Digital Advertising

- **6 billion ad deliveries per day**
- **Reports (and bills) for the advertising companies needed**
- **Own C++ solution did not scale**
- **Adding functions was a nightmare**



# AdServing Architecture

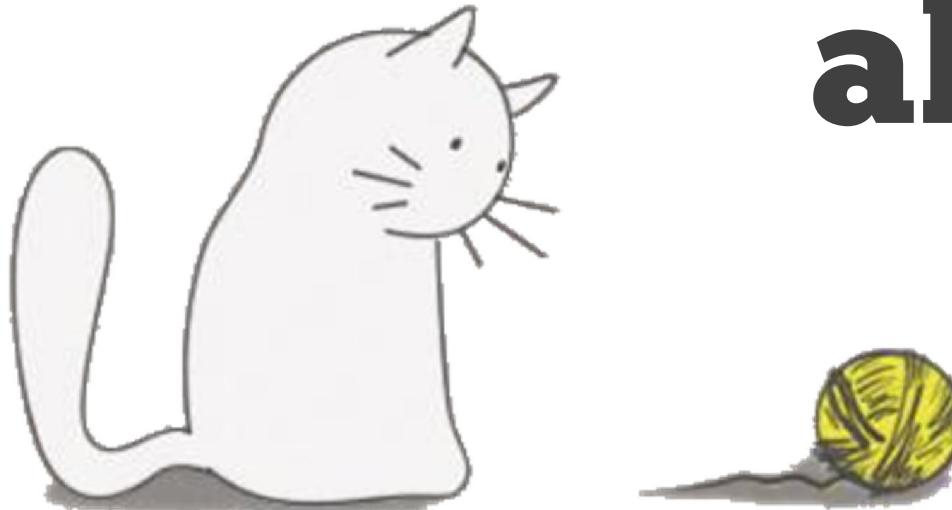
## Hadoop Cluster



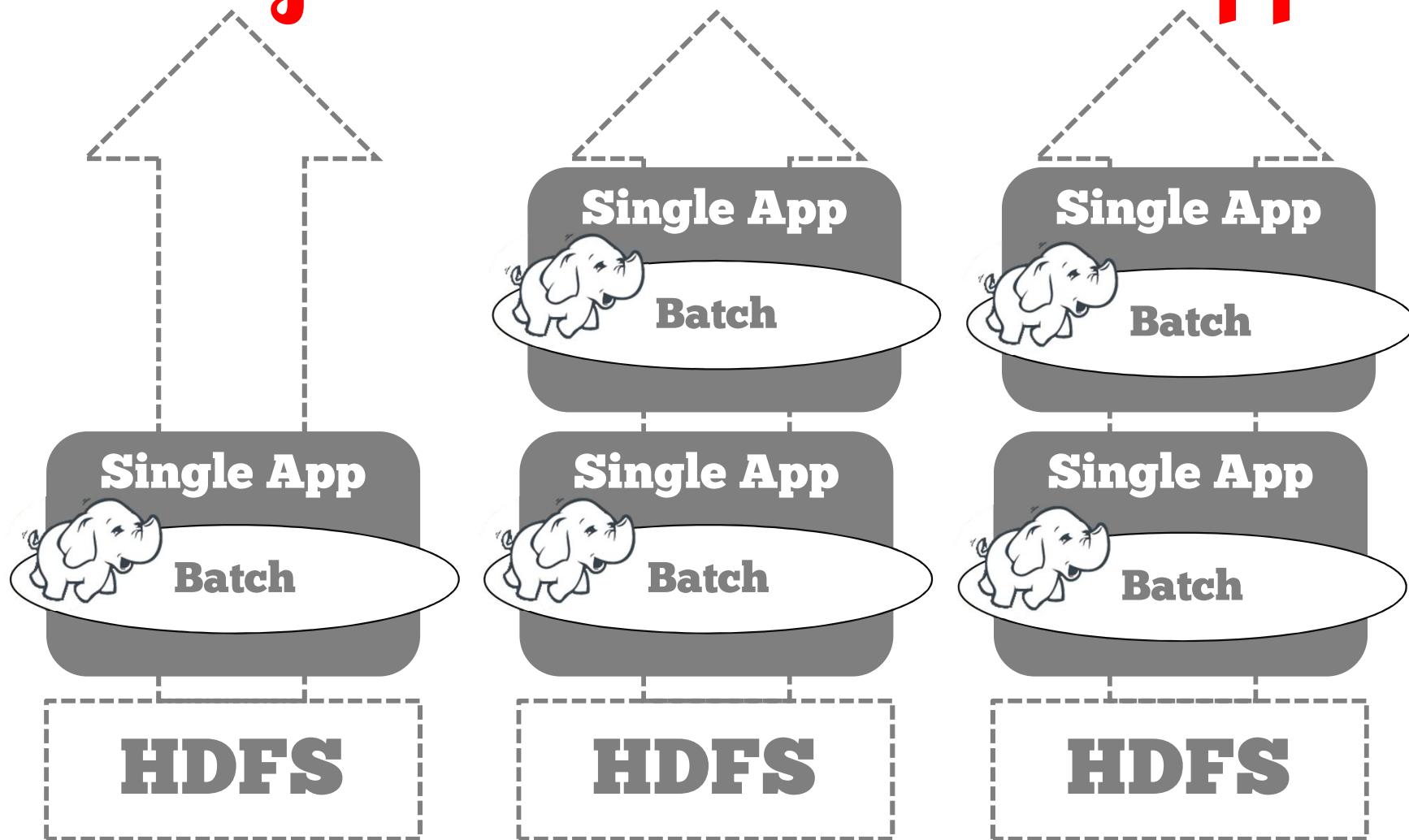
What's next?

# Hadoop 2.0

## aka YARN



## Built for web-scale batch apps



# MapReduce is good for...

- **Embarrassingly parallel algorithms**
- **Summing, grouping, filtering, joining**
- **Off-line batch jobs on massive data sets**
- **Analyzing an entire large dataset**

# MapReduce is OK for...

- **Iterative jobs (i.e., graph algorithms)**
  - **Each iteration must read/write data to disk**
  - **I/O and latency cost of an iteration is high**

# MapReduce is not good for...

- **Jobs that need shared state/coordination**
  - **Tasks are shared-nothing**
  - **Shared-state requires scalable state store**
- **Low-latency jobs**
- **Jobs on small datasets**
- **Finding individual records**

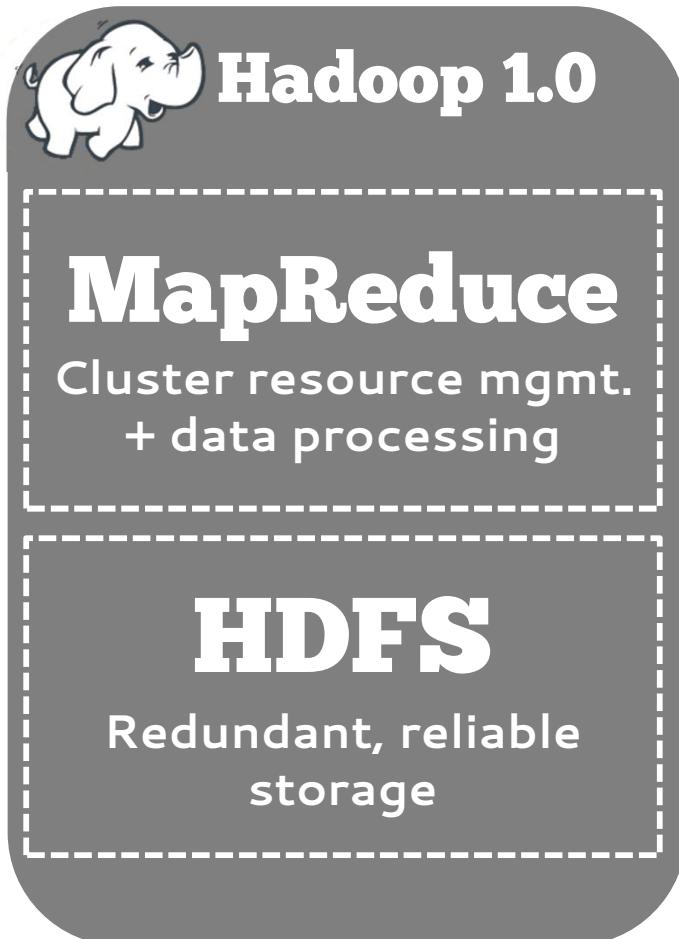
# MapReduce limitations

- **Scalability**
  - Maximum cluster size ~ 4,500 nodes
  - Maximum concurrent tasks - 40,000
  - Coarse synchronization in JobTracker
- **Availability**
  - Failure kills all queued and running jobs
- **Hard partition of resources into map & reduce slots**
  - Low resource utilization
- **Lacks support for alternate paradigms and services**
  - Iterative applications implemented using MapReduce are 10x slower

# Hadoop 2.0: Next-gen platform

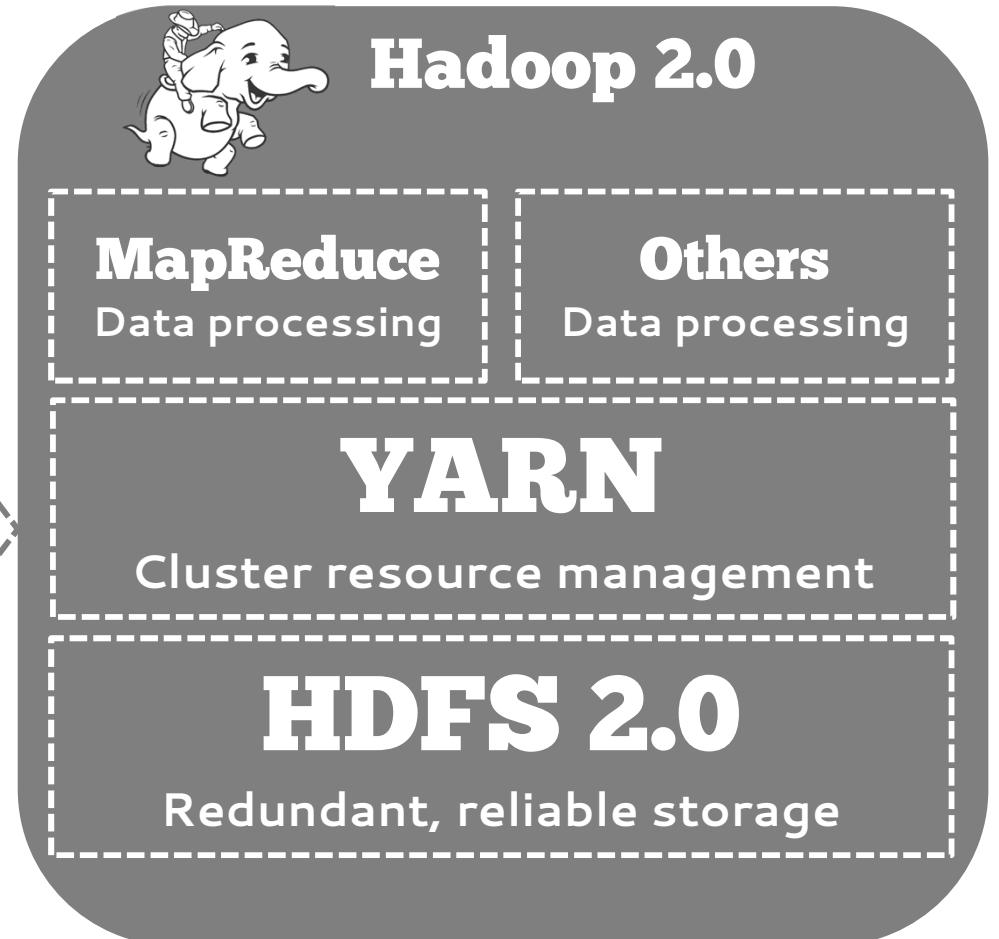
**Single use system**

Batch apps

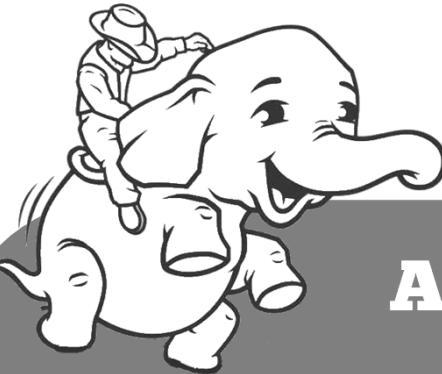


**Multi-purpose platform**

Batch, Interactive, Streaming, ...



# Taking Hadoop beyond batch



**Store all data in one place**  
Interact with data in multiple ways

**Applications run natively in Hadoop**

**Batch**  
MapReduce

**Interactive**  
Tez

**Online**  
HOYA

**Streaming**  
Storm, ...

**Graph**  
Giraph

**In-Memory**  
Spark

**Other**  
Search, ...

## **YARN**

Cluster resource management

## **HDFS 2.0**

Redundant, reliable storage

# A brief history of Hadoop 2.0

- **Originally conceived & architected by the team at Yahoo!**
  - Arun Murthy created the original JIRA in 2008 and now is the YARN release manager
- **The team at Hortonworks has been working on YARN for 4 years:**
  - 90% of code from Hortonworks & Yahoo!
- **Hadoop 2.0 based architecture running at scale at Yahoo!**
  - Deployed on 35,000 nodes for 6+ months

GA in October as Hadoop 2.2

# Hadoop 2.0 Projects

- **YARN**
- **HDFS Federation aka HDFS 2.0**
- **Stinger & Tez aka Hive 2.0**

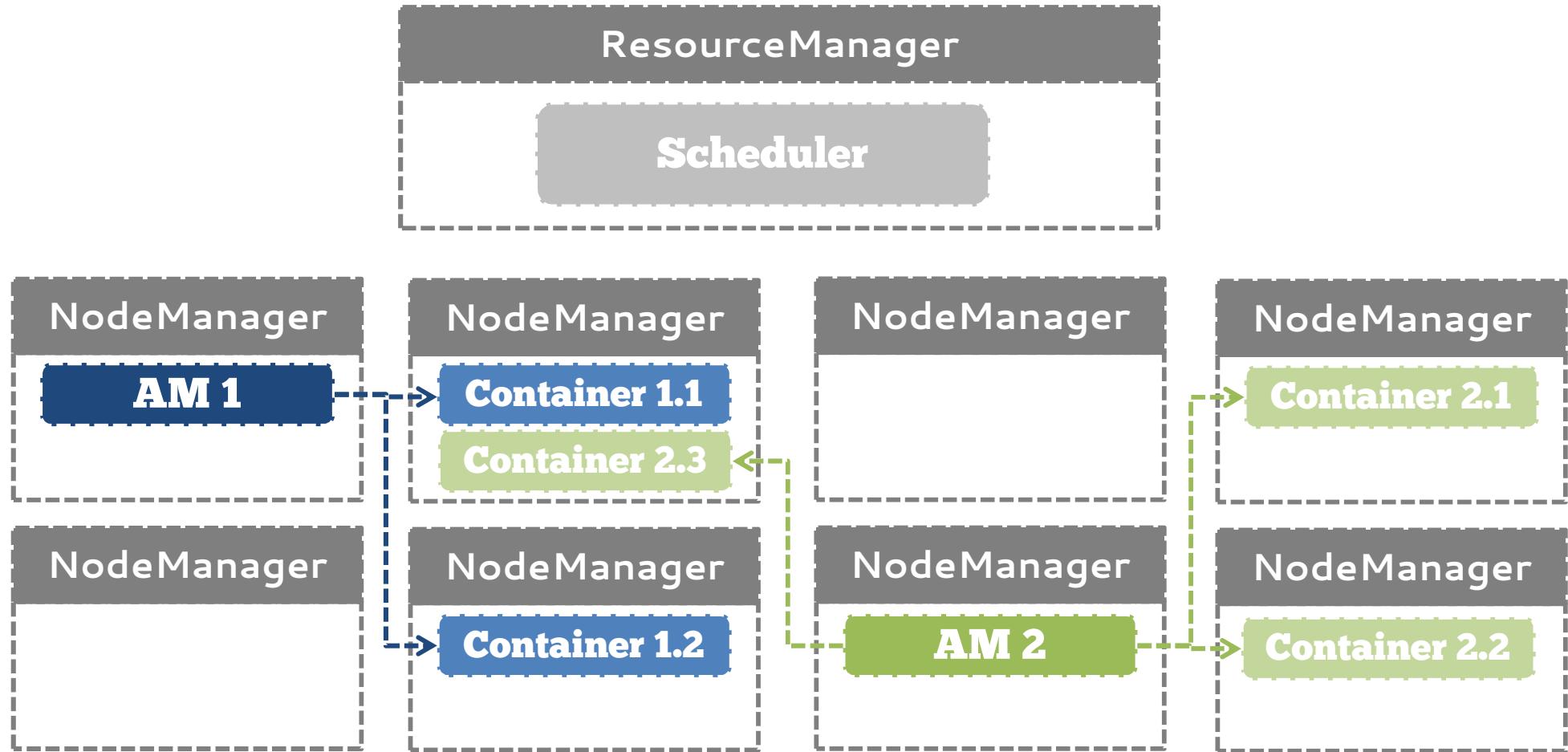
# Hadoop 2.0 Projects

- **YARN**
- **HDFS Federation aka HDFS 2.0**
- **Stinger & Tez aka Hive 2.0**

# YARN: Architecture

**Split up the two major functions of the JobTracker**

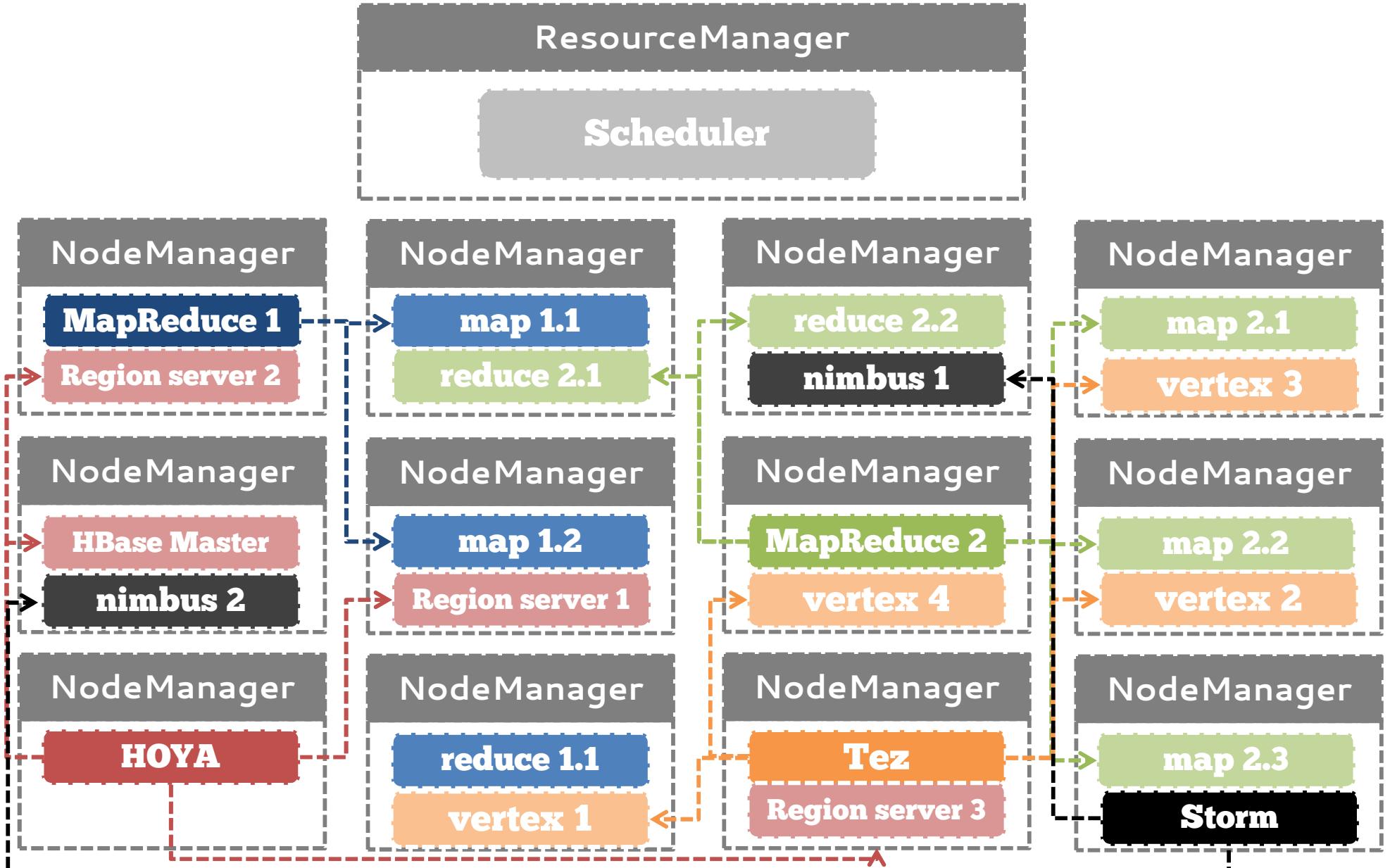
Cluster resource management & Application life-cycle management



# YARN: Architecture

- **Resource Manager**
  - Global resource scheduler
  - Hierarchical queues
- **Node Manager**
  - Per-machine agent
  - Manages the life-cycle of container
  - Container resource monitoring
- **Application Master**
  - Per-application
  - Manages application scheduling and task execution
  - e.g. MapReduce Application Master

# YARN: Architecture



# Hadoop 2.0 Projects

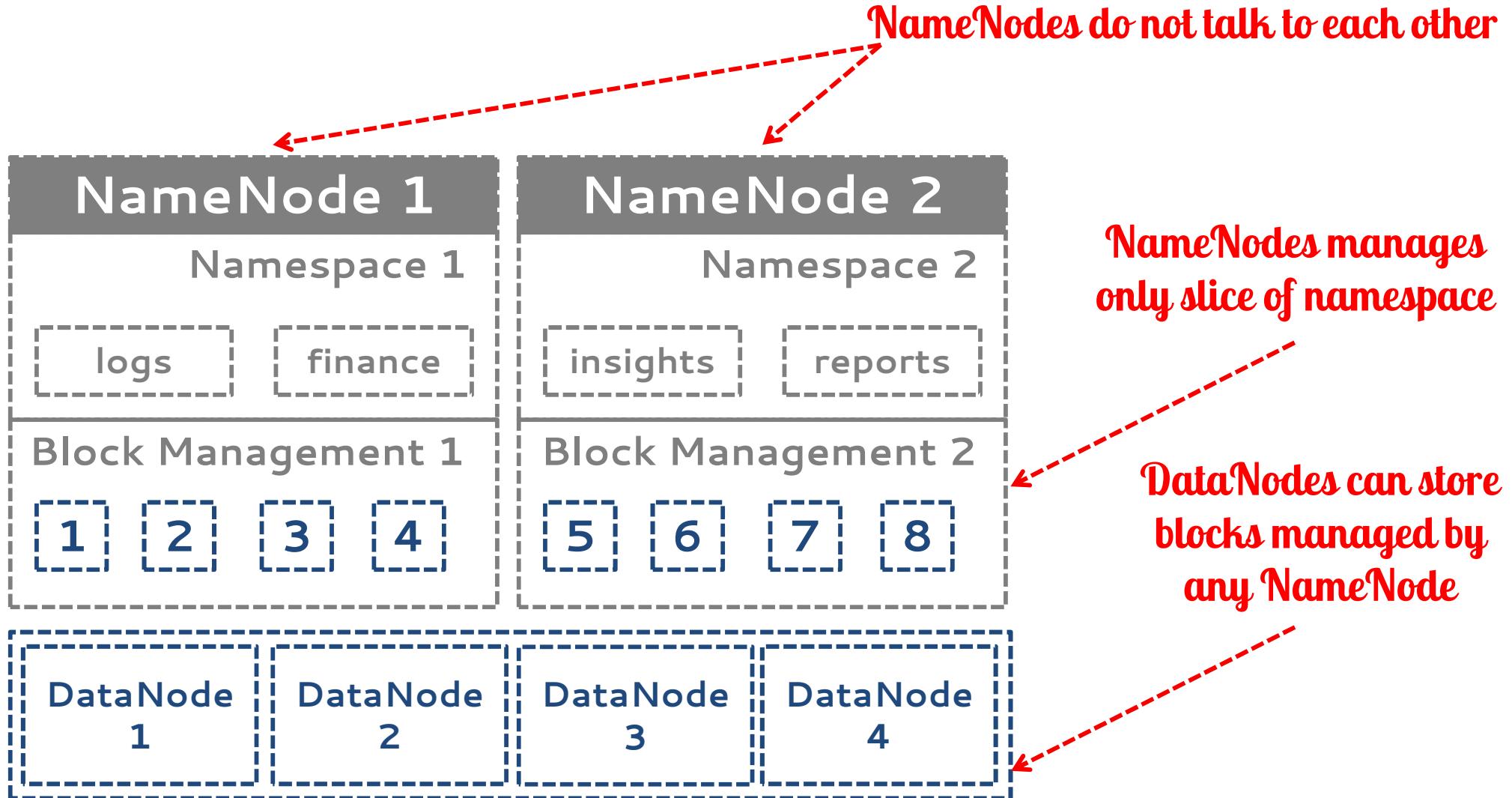
- **YARN**
- **HDFS Federation aka HDFS 2.0**
- **Stinger & Tez aka Hive 2.0**

# HDFS Federation

- **Removes tight coupling of Block Storage and Namespace**
- **Scalability & Isolation**
- **High Availability**
- **Increased performance**

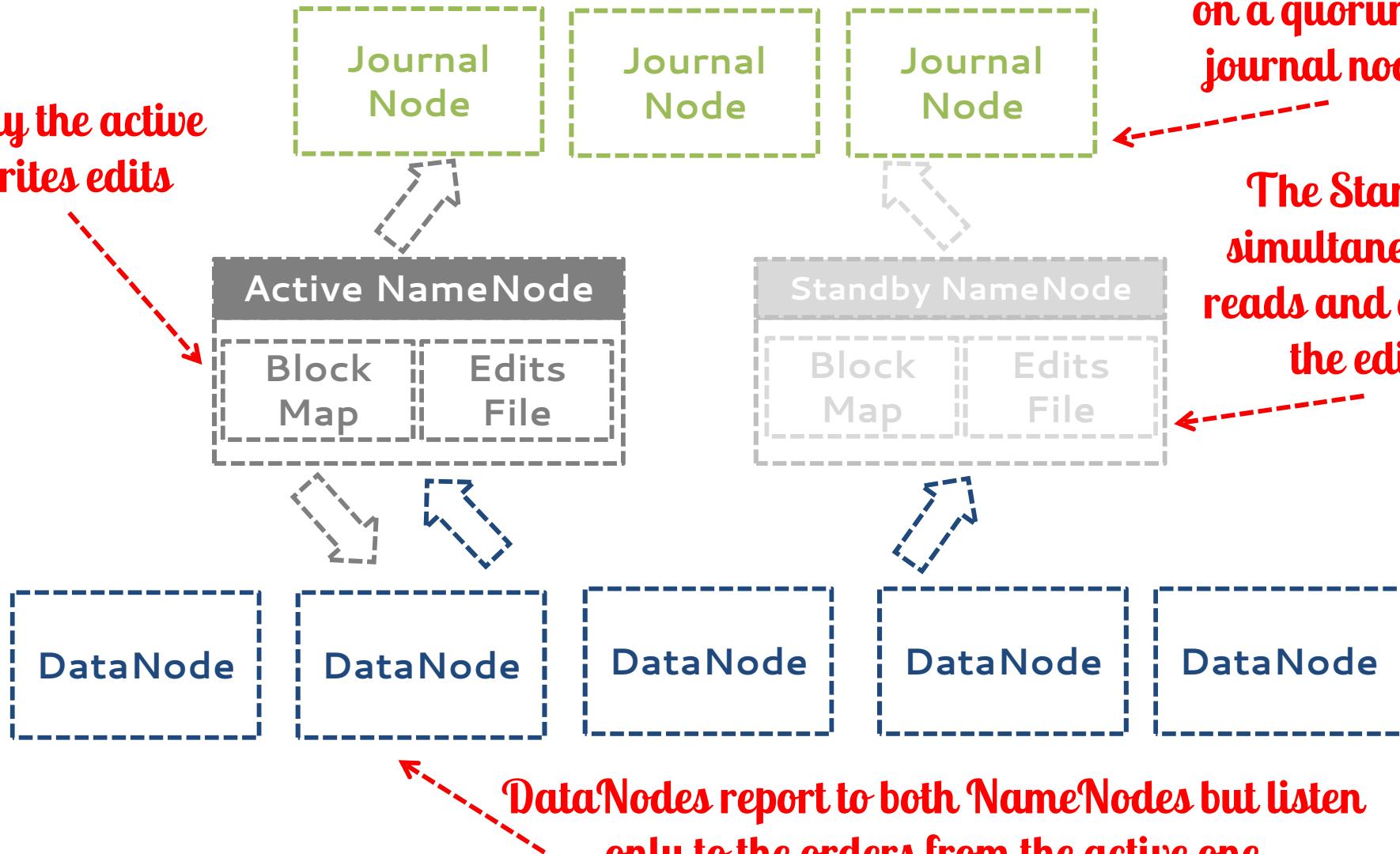
**Details: <https://issues.apache.org/jira/browse/HDFS-1052>**

# HDFS Federation: Architecture



# HDFS: Quorum based storage

Only the active writes edits



The state is shared on a quorum of journal nodes

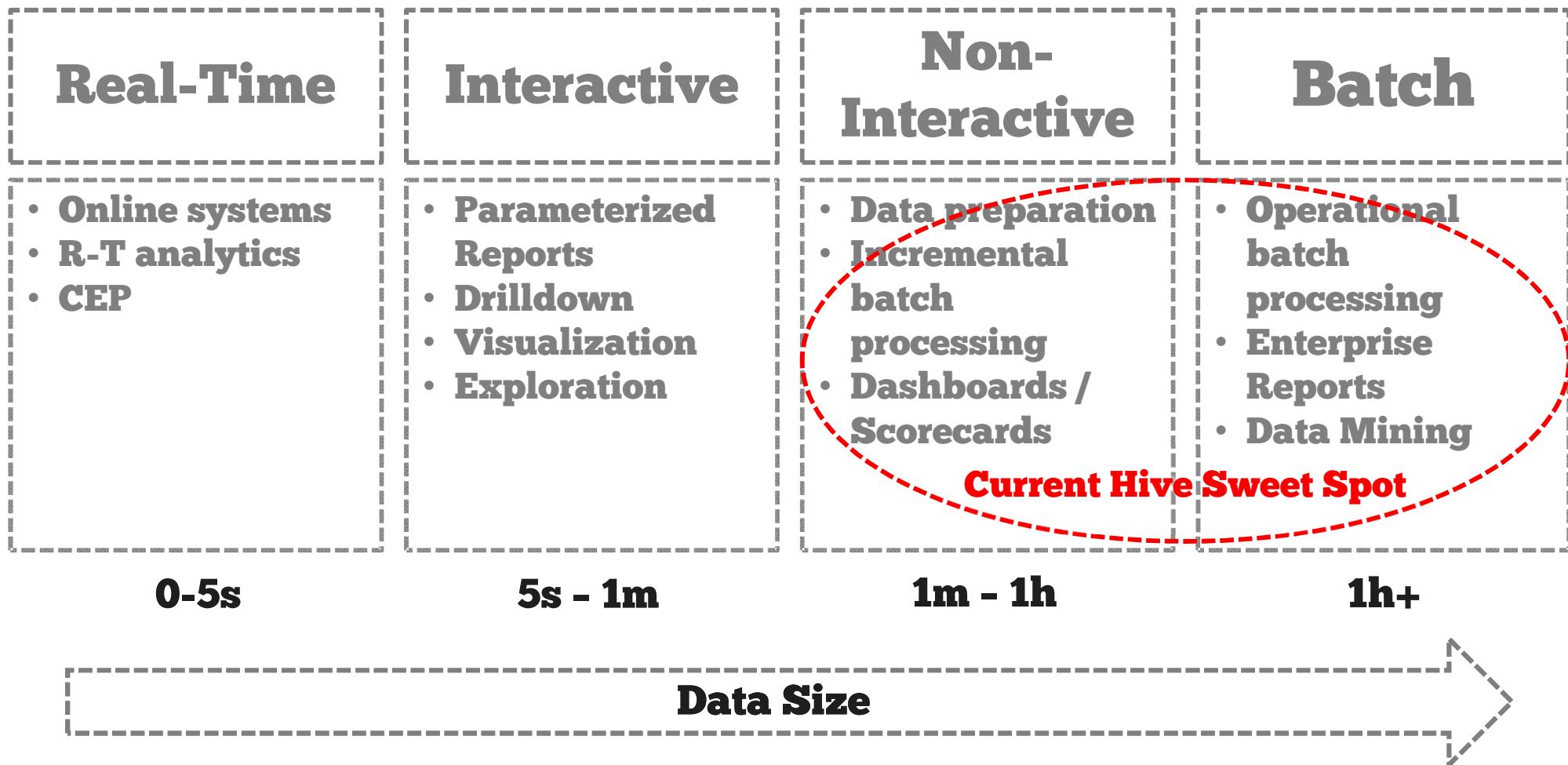
The Standby simultaneously reads and applies the edits

DataNodes report to both NameNodes but listen only to the orders from the active one

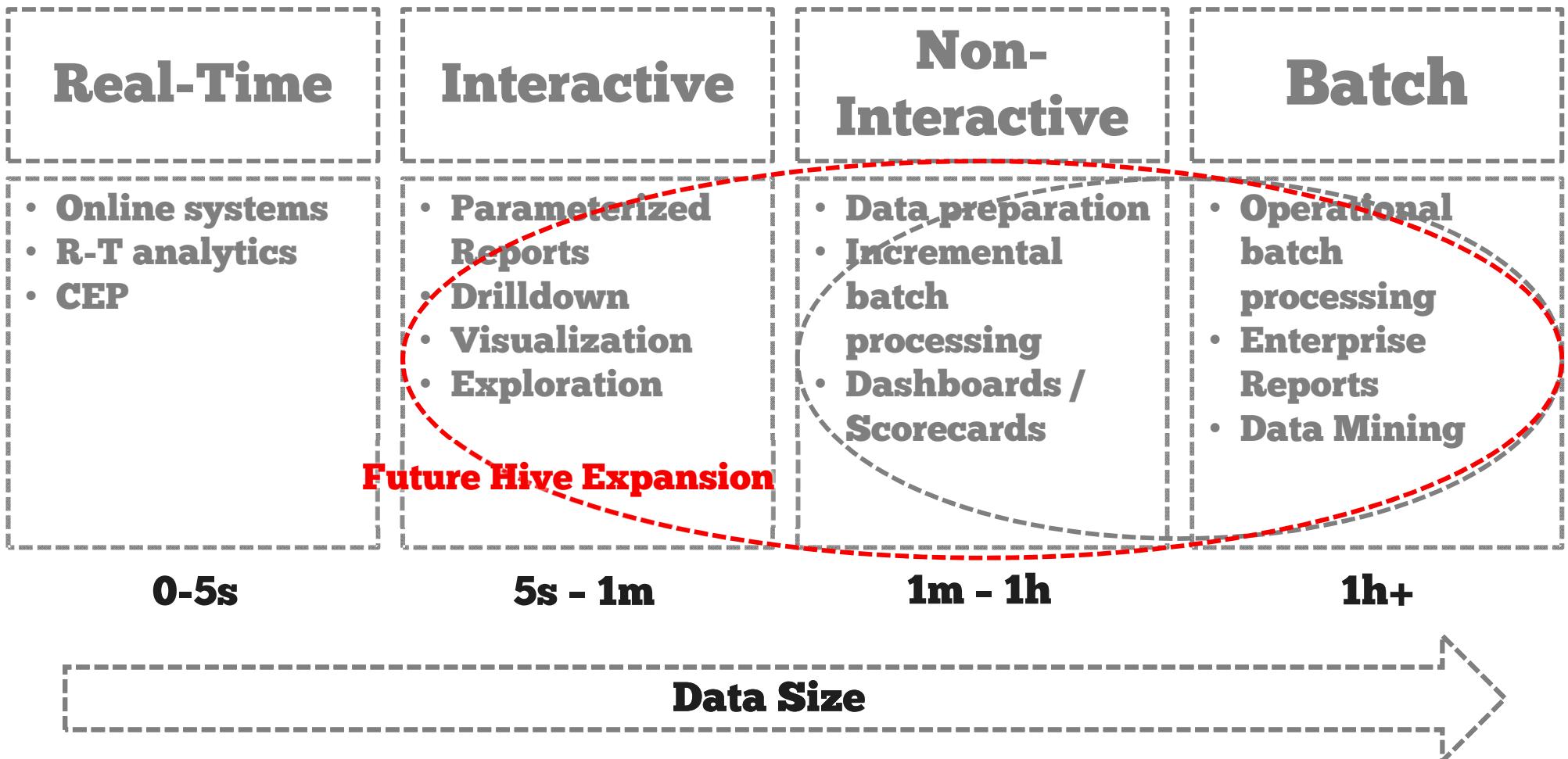
# Hadoop 2.0 Projects

- **YARN**
- **HDFS Federation aka HDFS 2.0**
- **Stinger & Tez aka Hive 2.0**

# Hive: Current Focus Area



# Stinger: Extending the sweet spot



## Improve Latency & Throughput

- Query engine improvements
- New “Optimized RCFile” column store
- Next-gen runtime (elim’s M/R latency)

## Data Size

## Extend Deep Analytical Ability

- Analytics functions
- Improved SQL coverage
- Continued focus on core Hive use cases

# Stinger Initiative at a glance

## The Stinger Initiative

### Making Apache Hive 100x Faster

#### **Base Optimizations**

Generate simplified DAGs  
In-memory Hash Joins

#### **Vector Query Engine**

Optimized for modern  
processor architectures

#### **Query Planner**

Intelligent Cost-Based  
Optimizer

#### **Deep Analytics**

SQL Compatible Types  
SQL Compatible Windowing  
More SQL Subqueries

#### **Hive Service**

Pre-warmed Containers  
Low-latency dispatch

Hadoop

Phase 1

Phase 2

Phase 3

#### **Buffer Caching**

Cache accessed data  
Optimized for vector engine

#### **Tez**

Express data processing  
tasks more simply  
Eliminate disk writes

#### **ORCFile**

Column Store  
High Compression  
Predicate / Filter Pushdowns

#### **YARN**

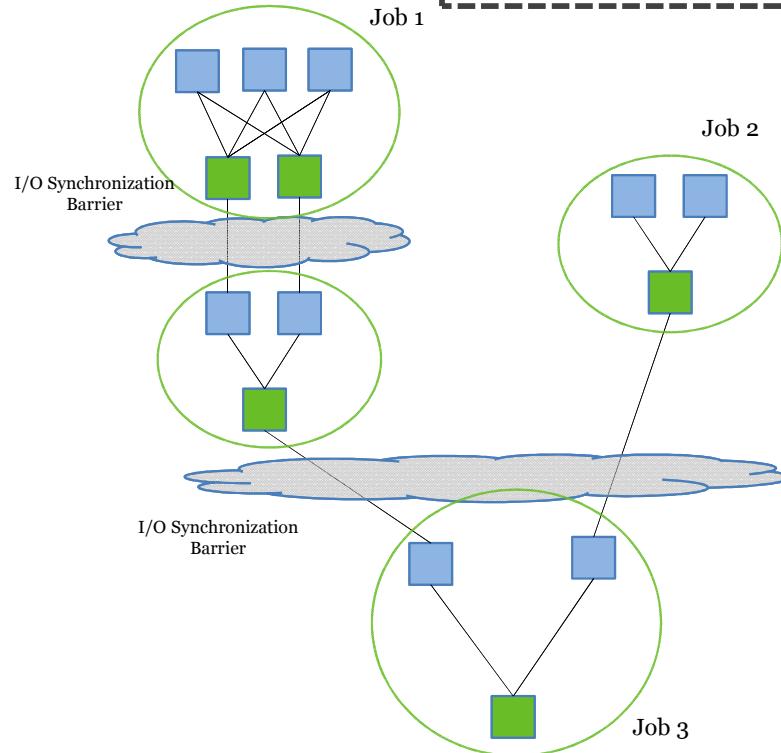
Next-gen Hadoop data  
processing framework

# Tez: The Execution Engine

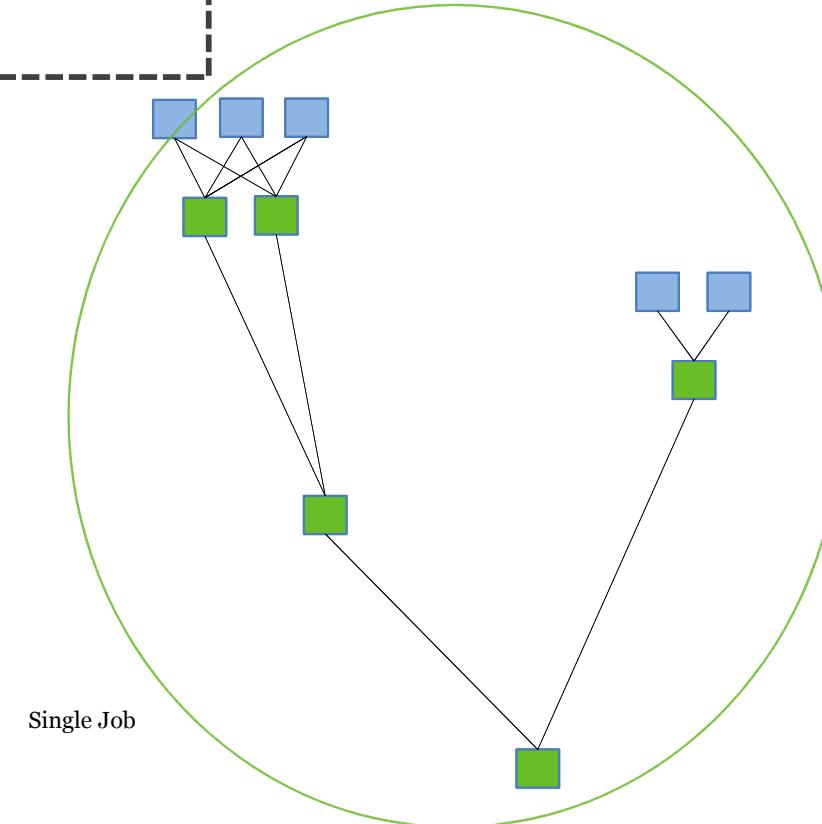
- **Low level data-processing execution engine**
- **Use it for the base of MapReduce, Hive, Pig, etc.**
- **Enables pipelining of jobs**
- **Removes task and job launch times**
- **Hive and Pig jobs no longer need to move to the end of the queue between steps in the pipeline**
- **Does not write intermediate output to HDFS**
  - **Much lighter disk and network usage**
- **Built on YARN**

# Pig/Hive MR vs. Pig/Hive Tez

```
SELECT a.state, COUNT(*),  
       AVERAGE(c.price)  
  FROM a  
 JOIN b ON (a.id = b.id)  
 JOIN c ON (a.itemId = c.itemId)  
 GROUP BY a.state
```



**Pig/Hive - MR**



**Pig/Hive - Tez**

# Tez Service

- **MapReduce Query Startup is expensive:**
  - Job launch & task-launch latencies are fatal for short queries (in order of 5s to 30s)
- **Solution:**
  - **Tez Service (= Preallocated Application Master)**
    - Removes job-launch overhead (Application Master)
    - Removes task-launch overhead (Pre-warmed Containers)
  - **Hive/Pig**
    - Submit query-plan to Tez Service
  - **Native Hadoop service, not ad-hoc**

# Tez: Low latency

```
SELECT a.state, COUNT(*),
       AVERAGE(c.price)
    FROM a
   JOIN b ON (a.id = b.id)
   JOIN c ON (a.itemId = c.itemId)
 GROUP BY a.state
```

Existing Hive	
<b>Parse Query</b>	<b>0.5s</b>
<b>Create Plan</b>	<b>0.5s</b>
<b>Launch Map-Reduce</b>	<b>20s</b>
<b>Process Map-Reduce</b>	<b>10s</b>
<b>Total</b>	<b>31s</b>

Hive/Tez	
<b>Parse Query</b>	<b>0.5s</b>
<b>Create Plan</b>	<b>0.5s</b>
<b>Launch Map-Reduce</b>	<b>20s</b>
<b>Process Map-Reduce</b>	<b>2s</b>
<b>Total</b>	<b>23s</b>

Tez & Tez Service	
<b>Parse Query</b>	<b>0.5s</b>
<b>Create Plan</b>	<b>0.5s</b>
<b>Submit to Tez Service</b>	<b>0.5s</b>
<b>Process Map-Reduce</b>	<b>2s</b>
<b>Total</b>	<b>3.5s</b>

\* No exact numbers, for illustration only

# Stinger: Summary

File Type	Number of MR Jobs	Input Size	Mappers	Time
Text/Hive 10	5	43.1 GB	179	21 minutes
Text/Hive 11	1	38 GB	151	246 seconds
RC/Hive 11	1	8.21 GB	76	136 seconds
ORC/Hive 11	1	2.83 GB	38	104 seconds
RC/Hive 11/ Partitioned/ Bucketed	1	1.73 GB	19	104 seconds
<b>ORC/Hive 11/ Partitioned/ Bucketed</b>	<b>1</b>	<b>687 MB</b>	<b>27</b>	<b>79.62</b>

\* Real numbers, but handle with care!

# Hadoop 2.0 Applications

- **MapReduce 2.0**
- **HOYA - HBase on YARN**
- **Storm, Spark, Apache S4**
- **Hamster (MPI on Hadoop)**
- **Apache Giraph**
- **Apache Hama**
- **Distributed Shell**
- **Tez**

# Hadoop 2.0 Applications

- **MapReduce 2.0**
- **HOYA - HBase on YARN**
- **Storm, Spark, Apache S4**
- **Hamster (MPI on Hadoop)**
- **Apache Giraph**
- **Apache Hama**
- **Distributed Shell**
- **Tez**

# MapReduce 2.0

- **Basically a porting to the YARN architecture**
- **MapReduce becomes a user-land library**
- **No need to rewrite MapReduce jobs**
- **Increased scalability & availability**
- **Better cluster utilization**

# Hadoop 2.0 Applications

- **MapReduce 2.0**
- **HOYA - HBase on YARN**
- **Storm, Spark, Apache S4**
- **Hamster (MPI on Hadoop)**
- **Apache Giraph**
- **Apache Hama**
- **Distributed Shell**
- **Tez**

# HOYA: HBase on YARN

- **Create on-demand HBase clusters**
- **Configure different HBase instances differently**
- **Better isolation**
- **Create (transient) HBase clusters from MapReduce jobs**
- **Elasticity of clusters for analytic / batch workload processing**
- **Better cluster resources utilization**

# Hadoop 2.0 Applications

- **MapReduce 2.0**
- **HOYA - HBase on YARN**
- **Storm, Spark, Apache S4**
- **Hamster (MPI on Hadoop)**
- **Apache Giraph**
- **Apache Hama**
- **Distributed Shell**
- **Tez**

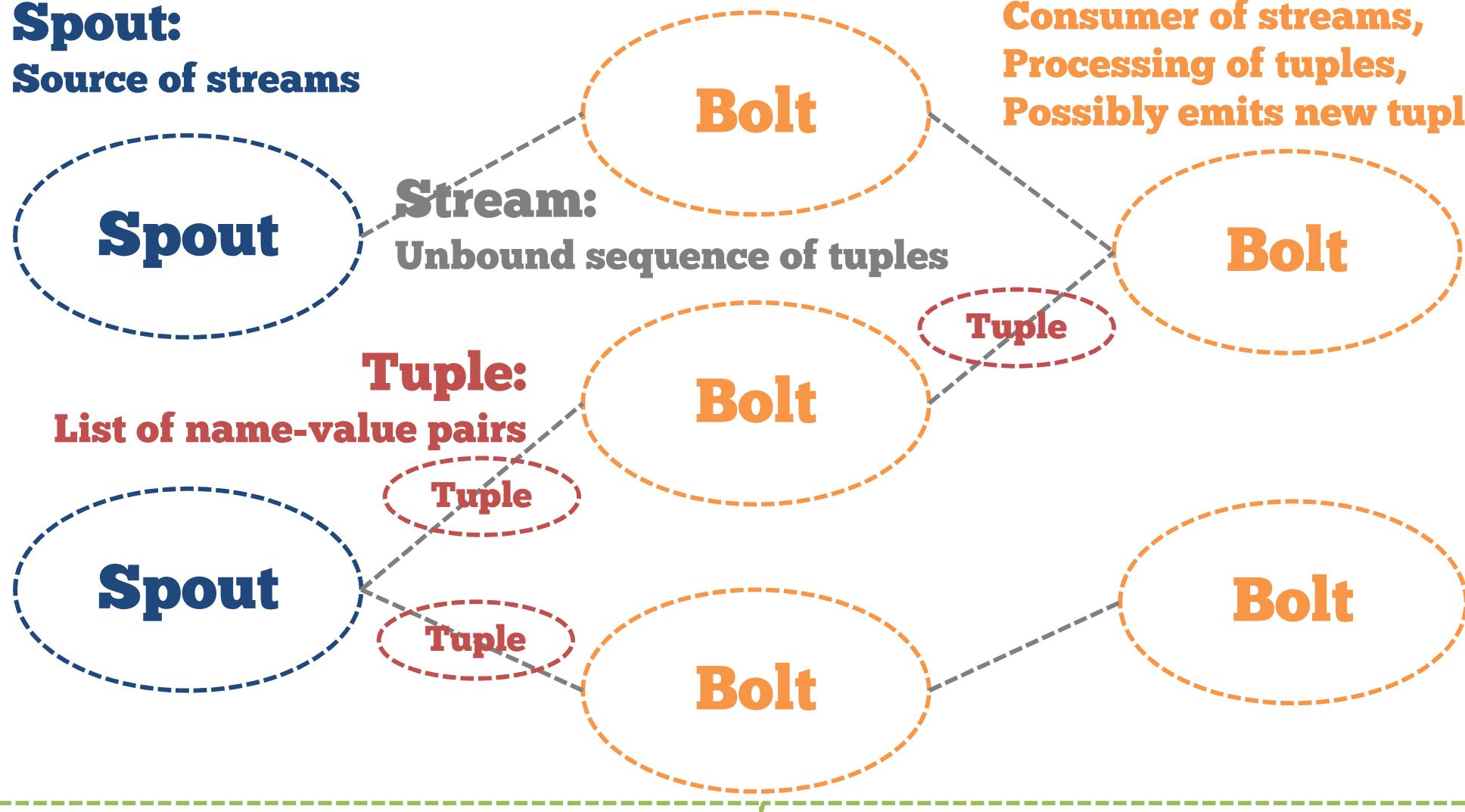
# Twitter Storm

- **Stream-processing**
- **Real-time processing**
- **Developed as standalone application**
  - <https://github.com/nathanmarz/storm>
- **Ported on YARN**
  - <https://github.com/yahoo/storm-yarn>

# Storm: Conceptual view

**Spout:**  
Source of streams

**Bolt:**  
Consumer of streams,  
Processing of tuples,  
Possibly emits new tuples



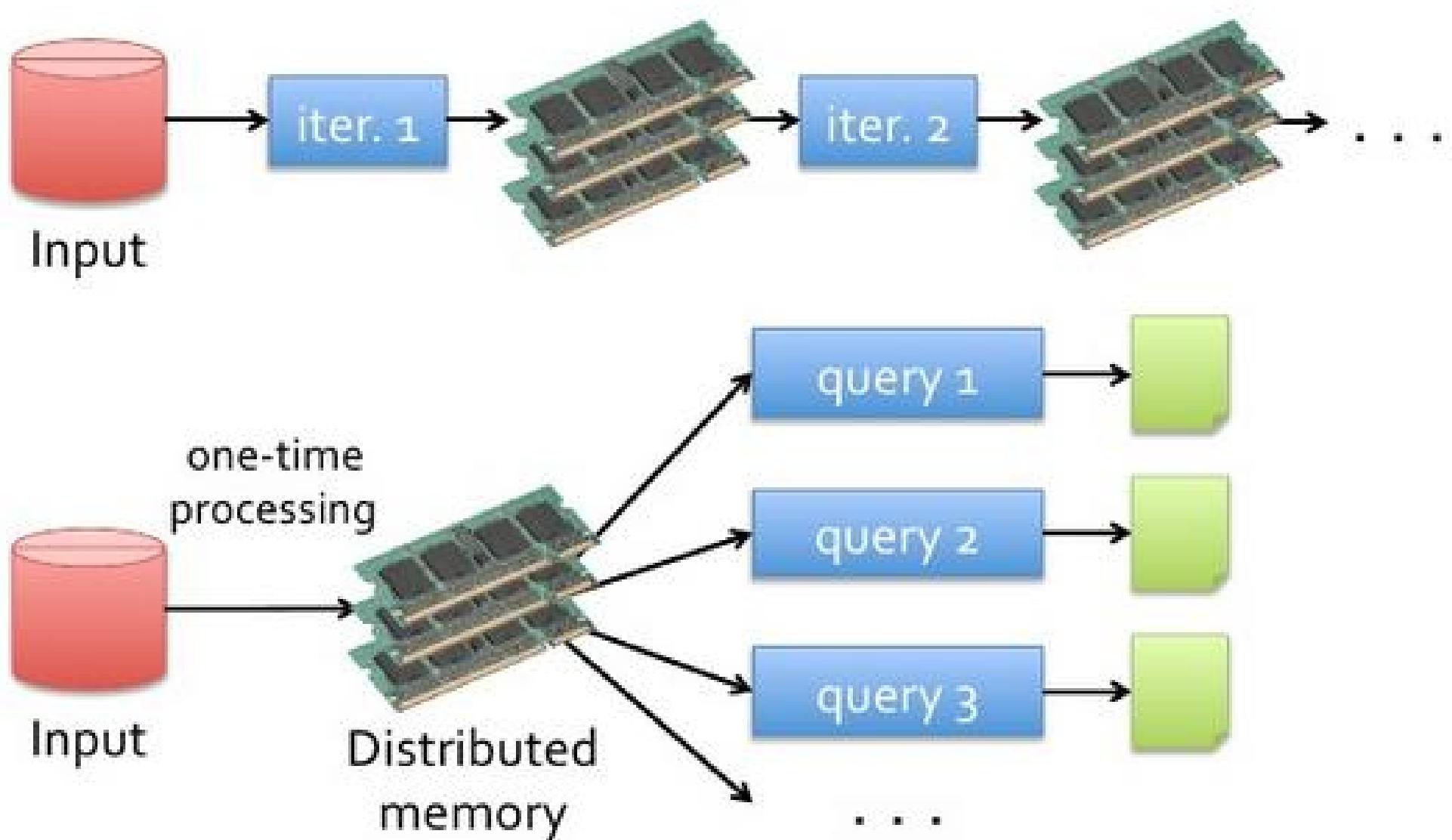
**Topology:** Network of Spouts & Bolts as the nodes and stream as the edge

# Hadoop 2.0 Applications

- **MapReduce 2.0**
- **HOYA - HBase on YARN**
- **Storm, Spark, Apache S4**
- **Hamster (MPI on Hadoop)**
- **Apache Giraph**
- **Apache Hama**
- **Distributed Shell**
- **Tez**

- **High-speed **in-memory** analytics over Hadoop and Hive**
- **Separate MapReduce-like engine**
  - Speedup of up to 100x
  - On-disk queries 5-10x faster
- **Compatible with Hadoop's Storage API**
- **Available as standalone application**
  - <https://github.com/mesos/spark>
- **Experimental support for YARN since 0.6**
  - <http://spark.incubator.apache.org/docs/0.6.0/running-on-yarn.html>

# Data Sharing in Spark



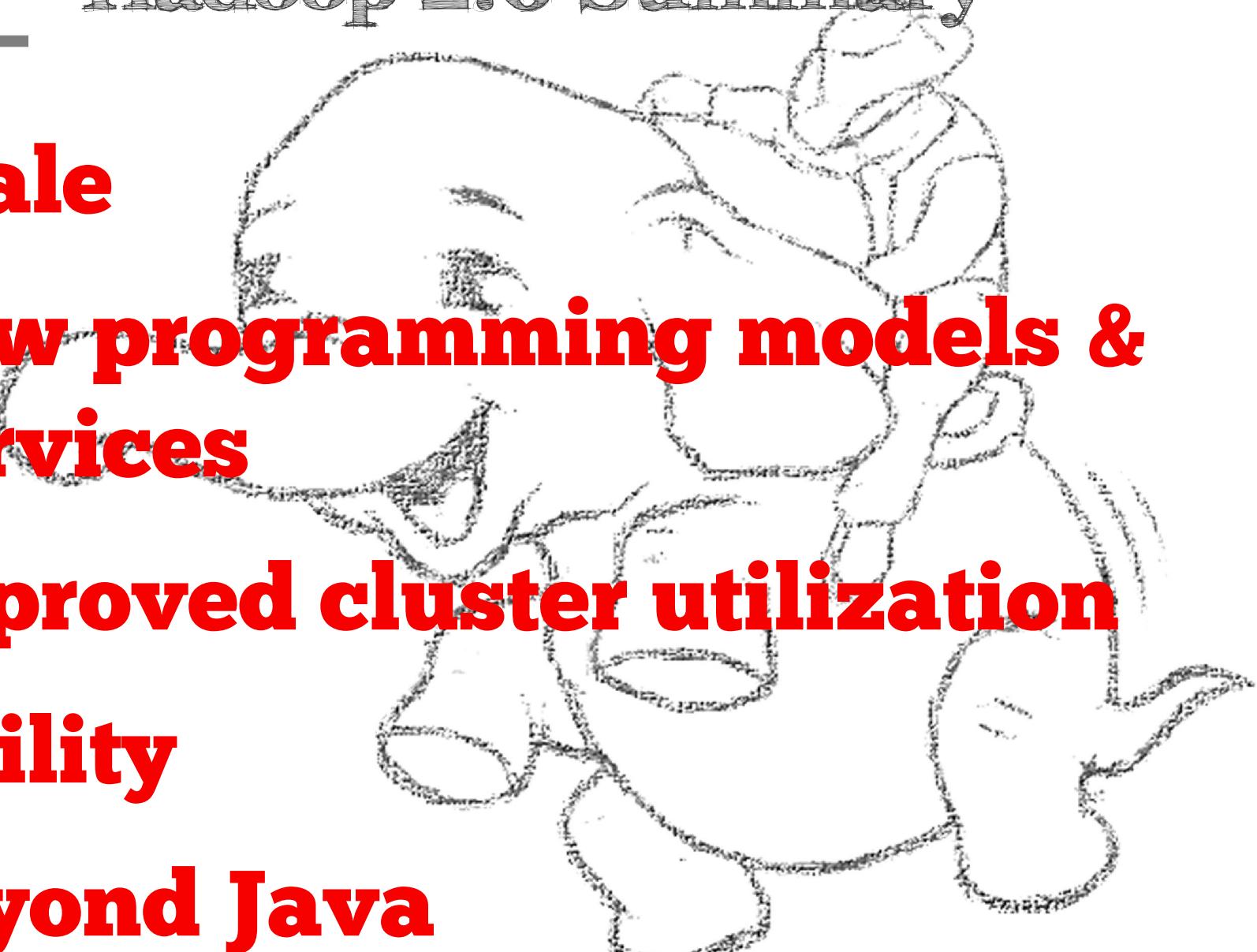
# Hadoop 2.0 Applications

- **MapReduce 2.0**
- **HOYA - HBase on YARN**
- **Storm, Spark, Apache S4**
- **Hamster (MPI on Hadoop)**
- **Apache Giraph**
- **Apache Hama**
- **Distributed Shell**
- **Tez**

# Apache Giraph

- **Giraph is a framework for processing semi-structured **graph data** on a massive scale.**
- **Giraph is loosely based upon Google's Pregel**
- **Giraph performs **iterative** calculations on top of an existing Hadoop cluster.**
- **Available on GitHub**
  - <https://github.com/apache/giraph>

# Hadoop 2.0 Summary

- 
- 1. Scale**
  - 2. New programming models & Services**
  - 3. Improved cluster utilization**
  - 4. Agility**
  - 5. Beyond Java**

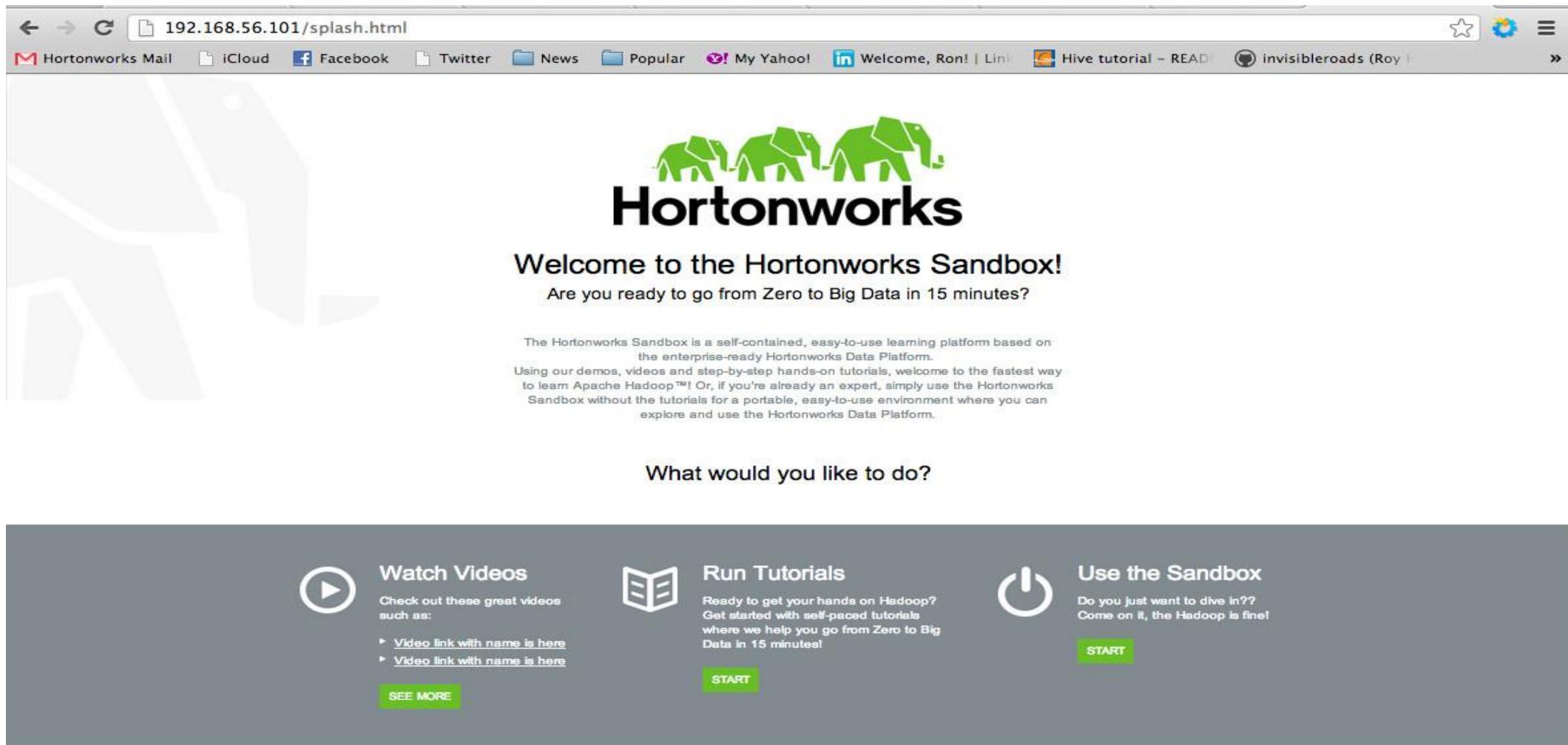
# Getting started...

# One more thing...

# Hortonworks Sandbox

192.168.56.101/splash.html

Hortonworks Mail iCloud Facebook Twitter News Popular My Yahoo! Welcome, Ron! | Link Hive tutorial – READ invisibleroads (Roy) »



The screenshot shows the Hortonworks Sandbox landing page. At the top is the Hortonworks logo (three green elephants) and the text "Welcome to the Hortonworks Sandbox!". Below it says "Are you ready to go from Zero to Big Data in 15 minutes?". A paragraph explains the purpose of the sandbox. The main section asks "What would you like to do?" with three options: "Watch Videos", "Run Tutorials", and "Use the Sandbox". Each option has a description and a "START" button.

The Hortonworks Sandbox is a self-contained, easy-to-use learning platform based on the enterprise-ready Hortonworks Data Platform. Using our demos, videos and step-by-step hands-on tutorials, welcome to the fastest way to learn Apache Hadoop™! Or, if you're already an expert, simply use the Hortonworks Sandbox without the tutorials for a portable, easy-to-use environment where you can explore and use the Hortonworks Data Platform.

What would you like to do?

**Watch Videos**  
Check out these great videos such as:  
▶ [Video link with name is here](#)  
▶ [Video link with name is here](#)

**Run Tutorials**  
Ready to get your hands on Hadoop? Get started with self-paced tutorials where we help you go from Zero to Big Data in 15 minutes!

**Use the Sandbox**  
Do you just want to dive in?? Come on it, the Hadoop is fine!

**SEE MORE**

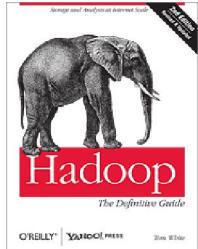
**START**

**START**

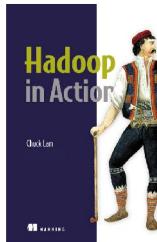
©Hortonworks Inc 2013. All Rights Reserved. Hadoop and the Hadoop elephant logo are trademarks of the Apache Software Foundation.

**<http://hortonworks.com/products/hortonworks-sandbox>**

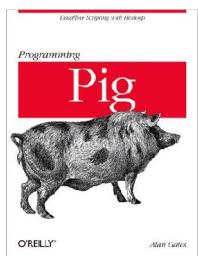
# Books about Hadoop



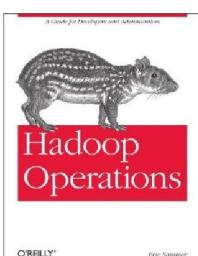
**Hadoop - The Definite Guide, Tom White,  
3rd ed., O'Reilly, 2012.**



**Hadoop in Action, Chuck Lam,  
Manning, 2011**



**Programming Pig, Alan Gates  
O'Reilly, 2011**



**Hadoop Operations, Eric Sammer,  
O'Reilly, 2012**

# The end...or the beginning?

Questions?

