IT-Stammtisch
DARMSTADT
07.11.13

Now with
Hadoop 2.0

uweseiler

codecentric

Introduction to the Hadoop Ecosystem

IT-Stammtisch DARMSTADT 07.11.13

About me

Big Data Nerd

Hadoop Trainer

MongoDB Author

Photography Enthusiast

Travelpirate

codecentric

IT-Stammtisch DARMSTADT 07.11.13

About us

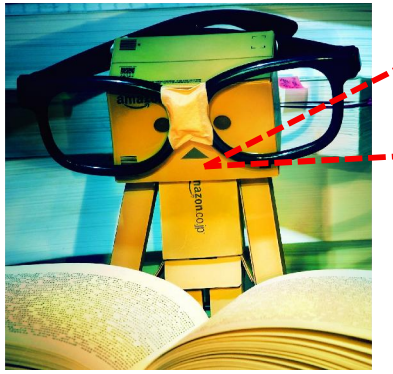
codecentric is a bunch of...

Big Data Nerds

Agile Ninjas

Continuous Delivery Gurus

Enterprise Java Specialists



Performance Geeks

Join us!

IT-Stammtisch DARMSTADT 07.11.13

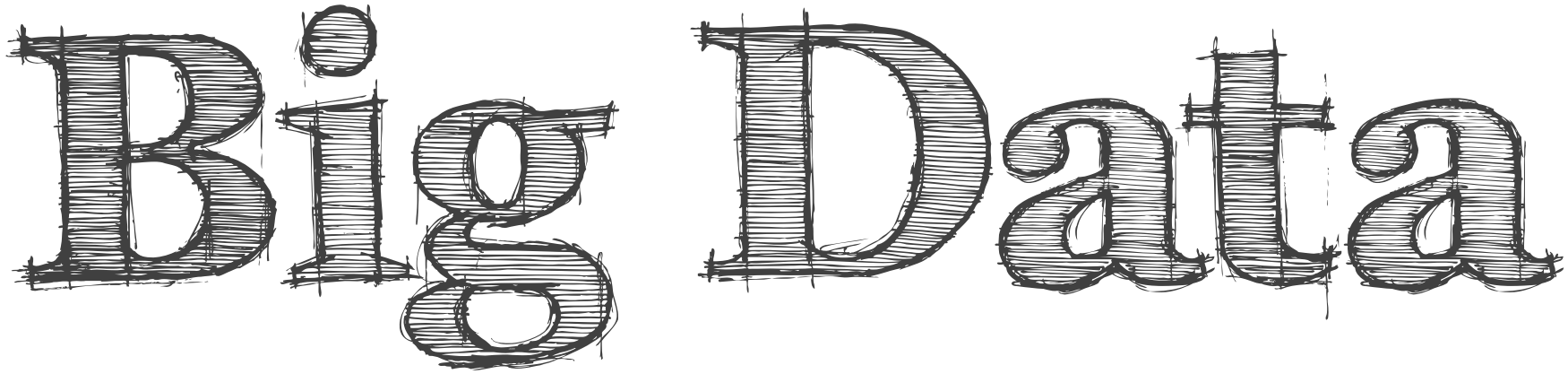
Agenda

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

codecentric

Agenda

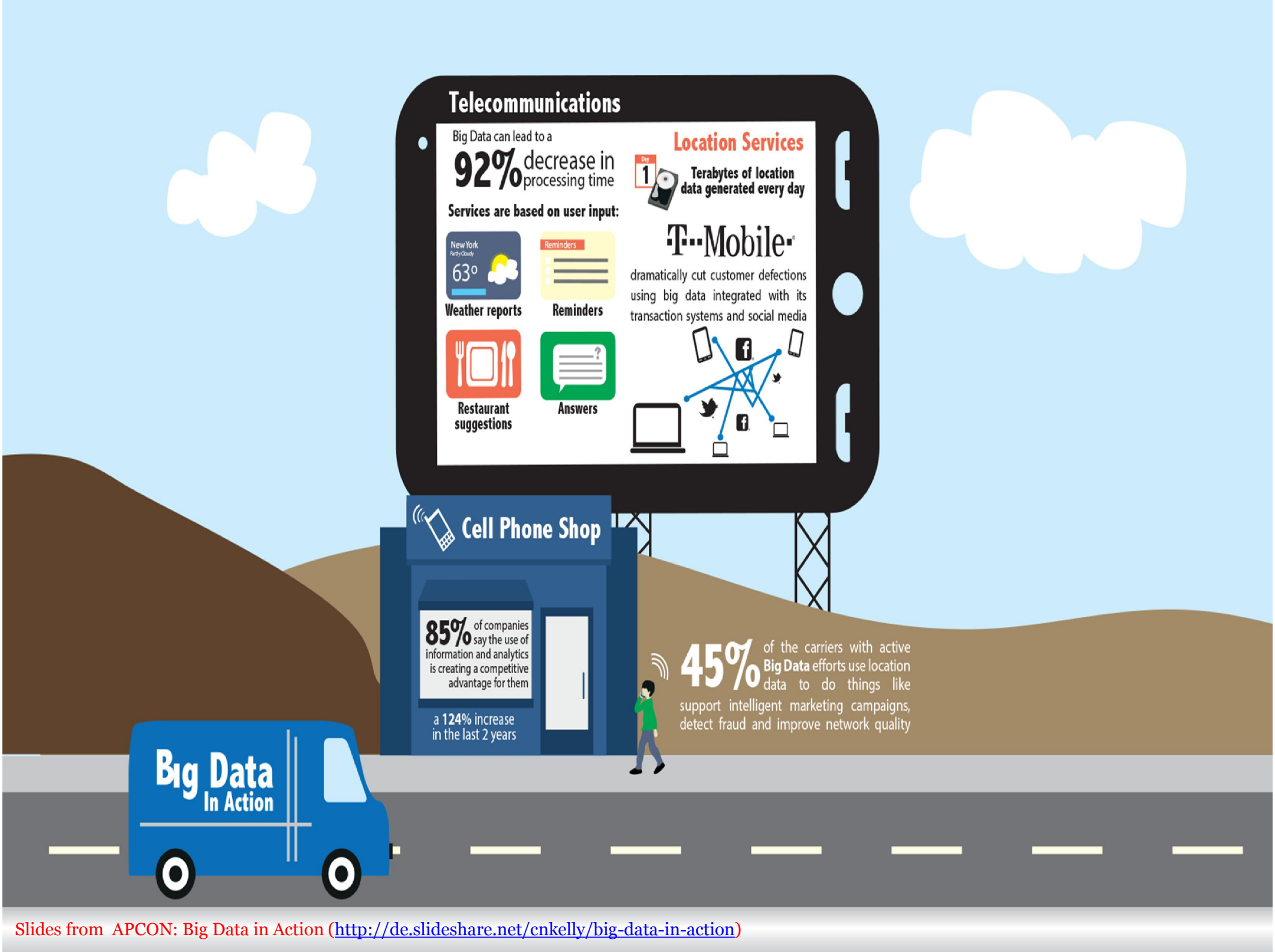
- **What is Big Data & Hadoop?**
- **Core Hadoop**
- **The Hadoop Ecosystem**
- **Use Cases**
- **What‘s next? Hadoop 2.0!**


Big Data

Big Data is like teenage sex:

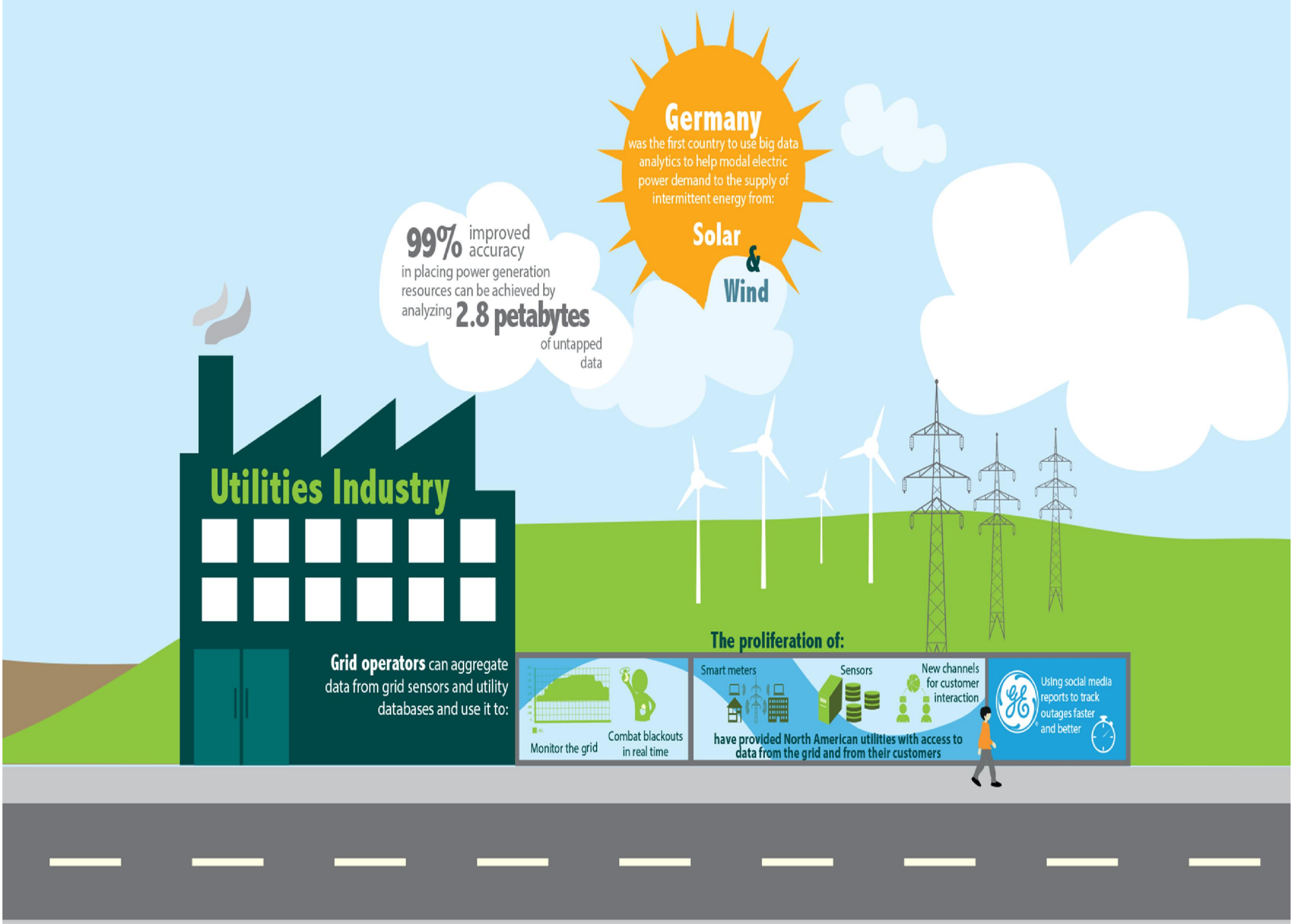
everybody talks about it, nobody really knows how to do it, everyone thinks everyone else is doing it, so everyone claims they are doing it...

Slides from APCON: Big Data in Action (http://de.slideshare.net/cnkelly/big-data-in-action)



Slides from APCON: Big Data in Action (http://de.slideshare.net/cnkelly/big-data-in-action)

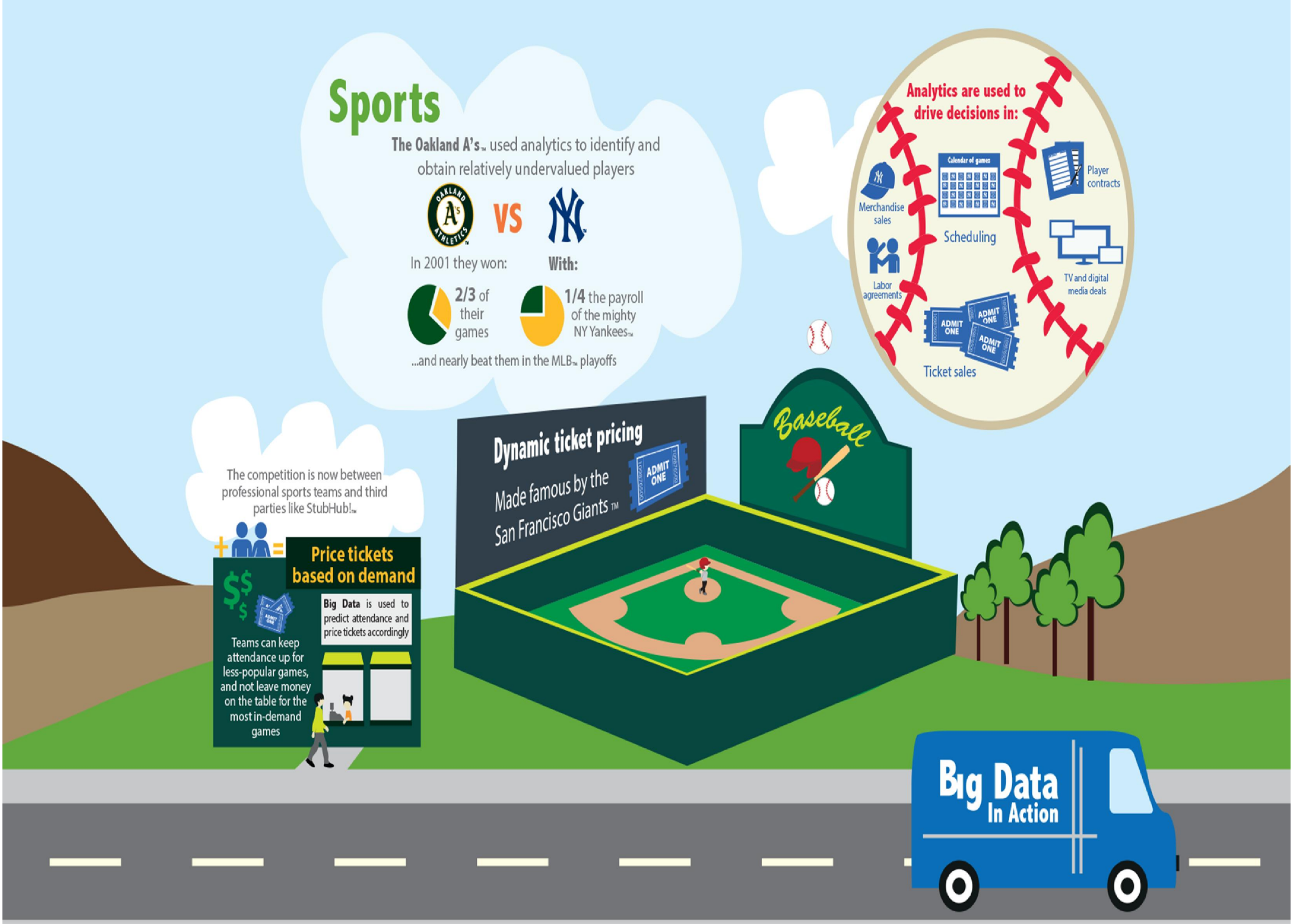
Slides from APCON: Big Data in Action (http://de.slideshare.net/cnkelly/big-data-in-action)



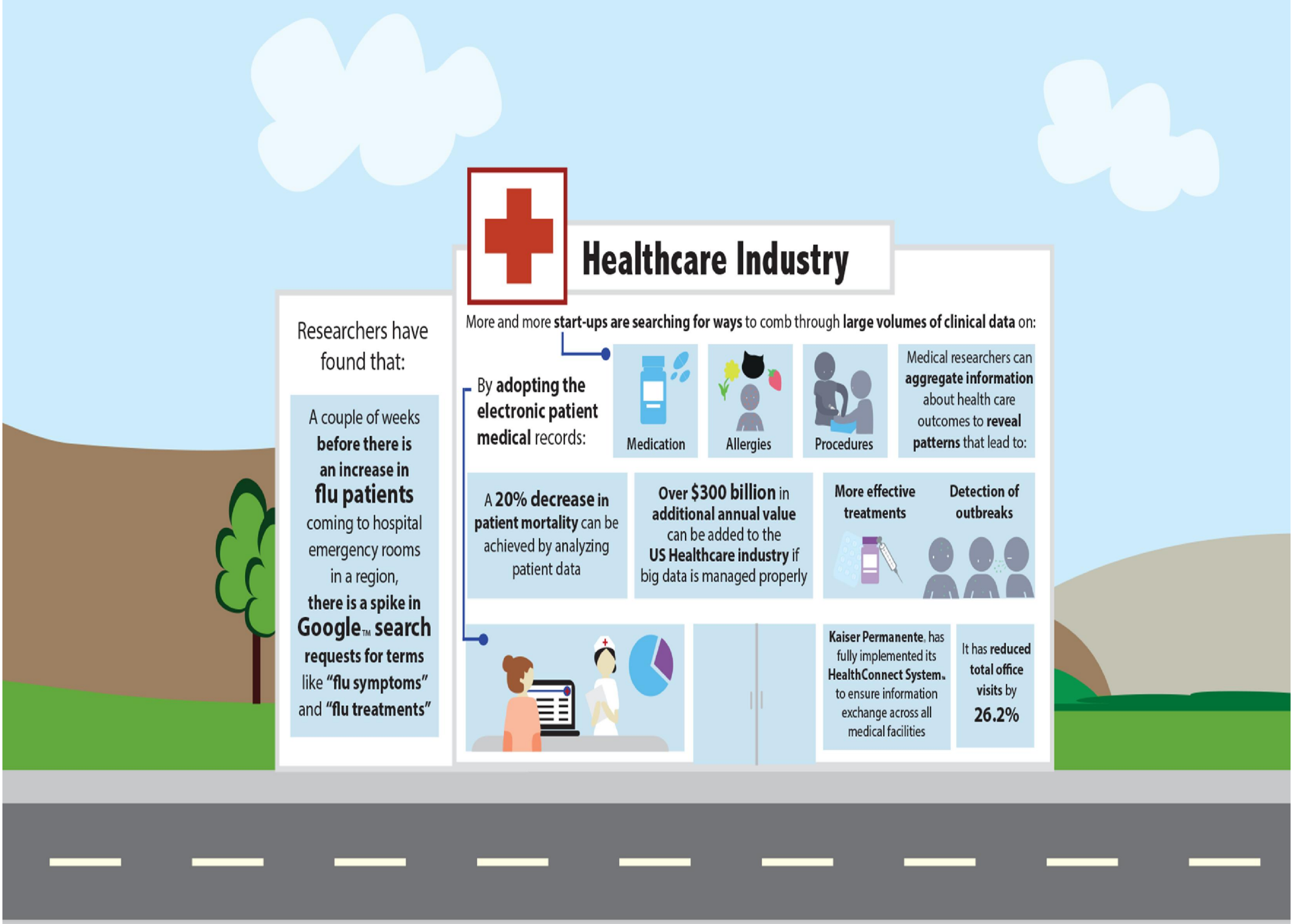
Slides from APCON: Big Data in Action (http://de.slideshare.net/cnkelly/big-data-in-action)

Slides from APCON: Big Data in Action (http://de.slideshare.net/cnkelly/big-data-in-action)

Slides from APCON: Big Data in Action (http://de.slideshare.net/cnkelly/big-data-in-action)



Slides from APCON: Big Data in Action (http://de.slideshare.net/cnkelly/big-data-in-action)



Slides from APCON: Big Data in Action (http://de.slideshare.net/cnkelly/big-data-in-action)

Slides from APCON: Big Data in Action (http://de.slideshare.net/cnkelly/big-data-in-action)

Slides from APCON: Big Data in Action (http://de.slideshare.net/cnkelly/big-data-in-action)

Slides from APCON: Big Data in Action (http://de.slideshare.net/cnkelly/big-data-in-action)

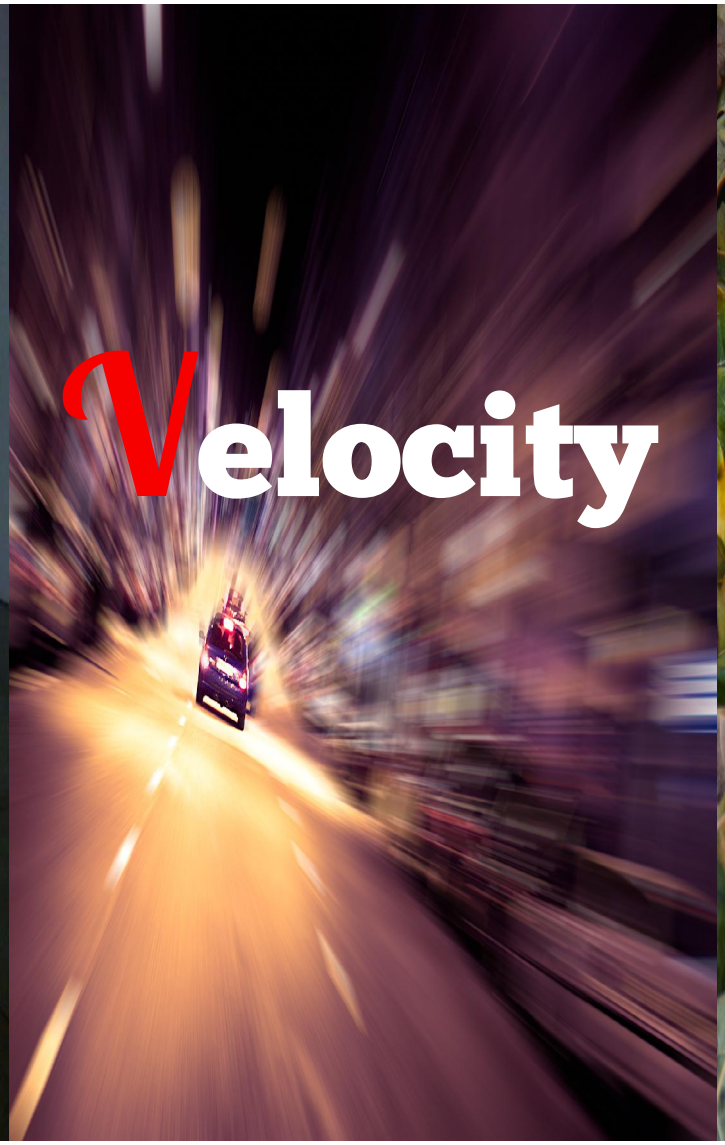
IT-Stammtisch DARMSTADT 07.11.13

My favorite definition

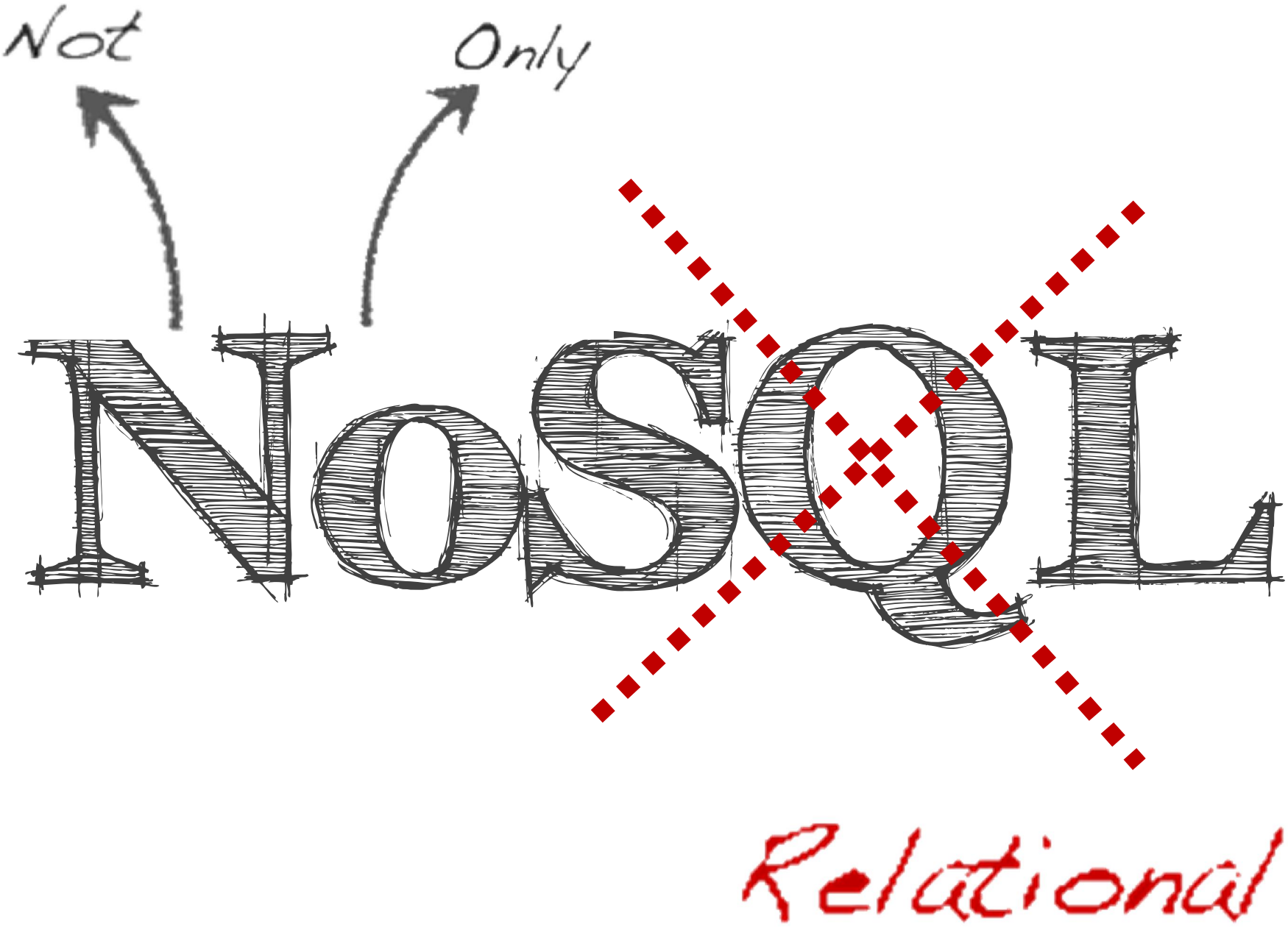
Big Data is when the data itself becomes part of the problem.

codecentric

The classic definition



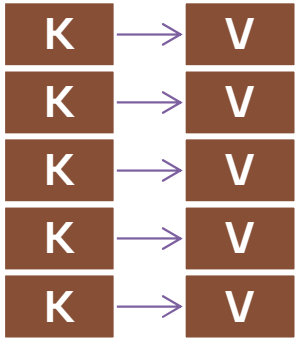
codecentric


Not
Only
NoSQL
Relational

IT-Stammtisch DARMSTADT 07.11.13

Classification of NoSQL

Key-Value Stores

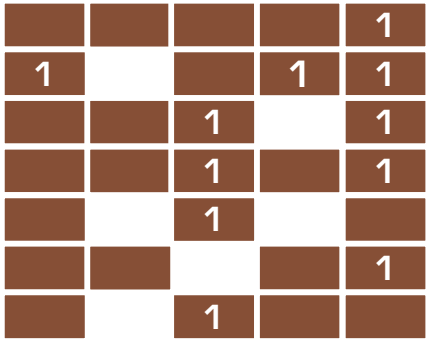


amazon web services Amazon DynamoDB

redis

riak

Column Stores

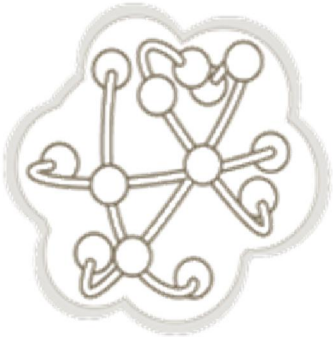


Google Bigtable

APACHE HBASE

Cassandra

Graph Databases

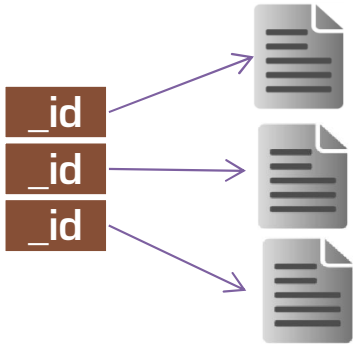


Neo4j The World's Leading Graph Database

HYPERGRAPHDB

*dex

Document Stores



mongoDB

CouchDB relax

codecentric

Horizontal Scaling

Vertical Scaling

RAM
CPU
Storage

Vertical Scaling

RAM
CPU
Storage

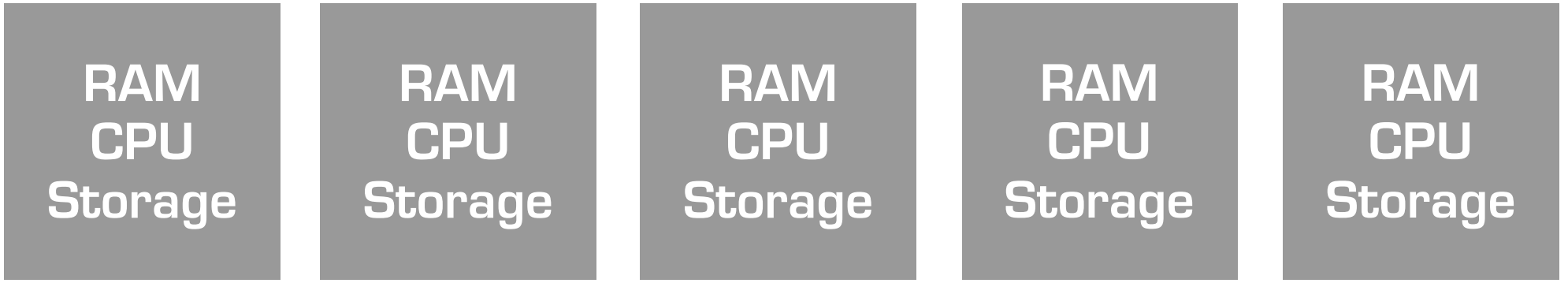
Vertical Scaling

RAM
CPU
Storage

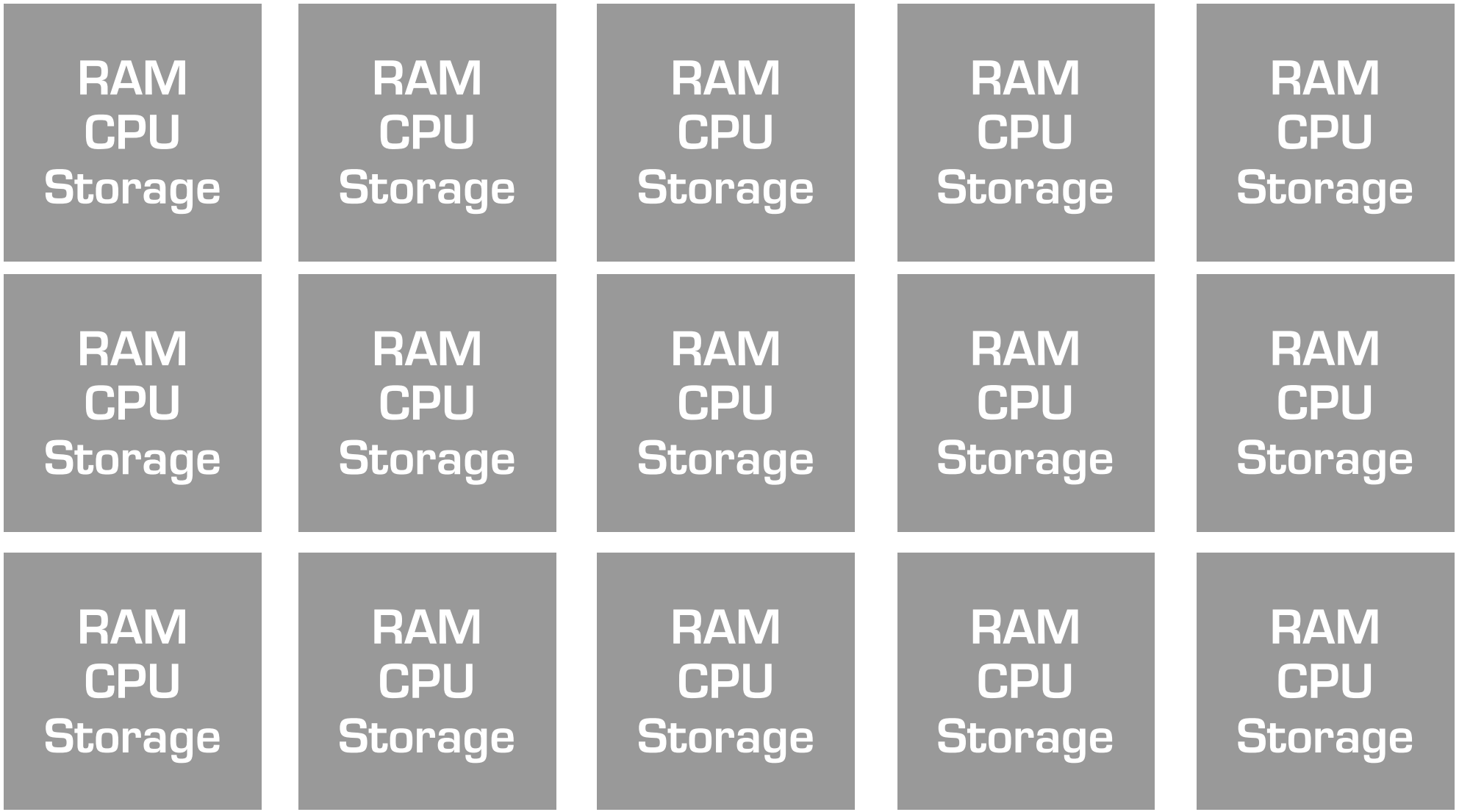
Horizontal Scaling

RAM
CPU
Storage

Horizontal Scaling



Horizontal Scaling

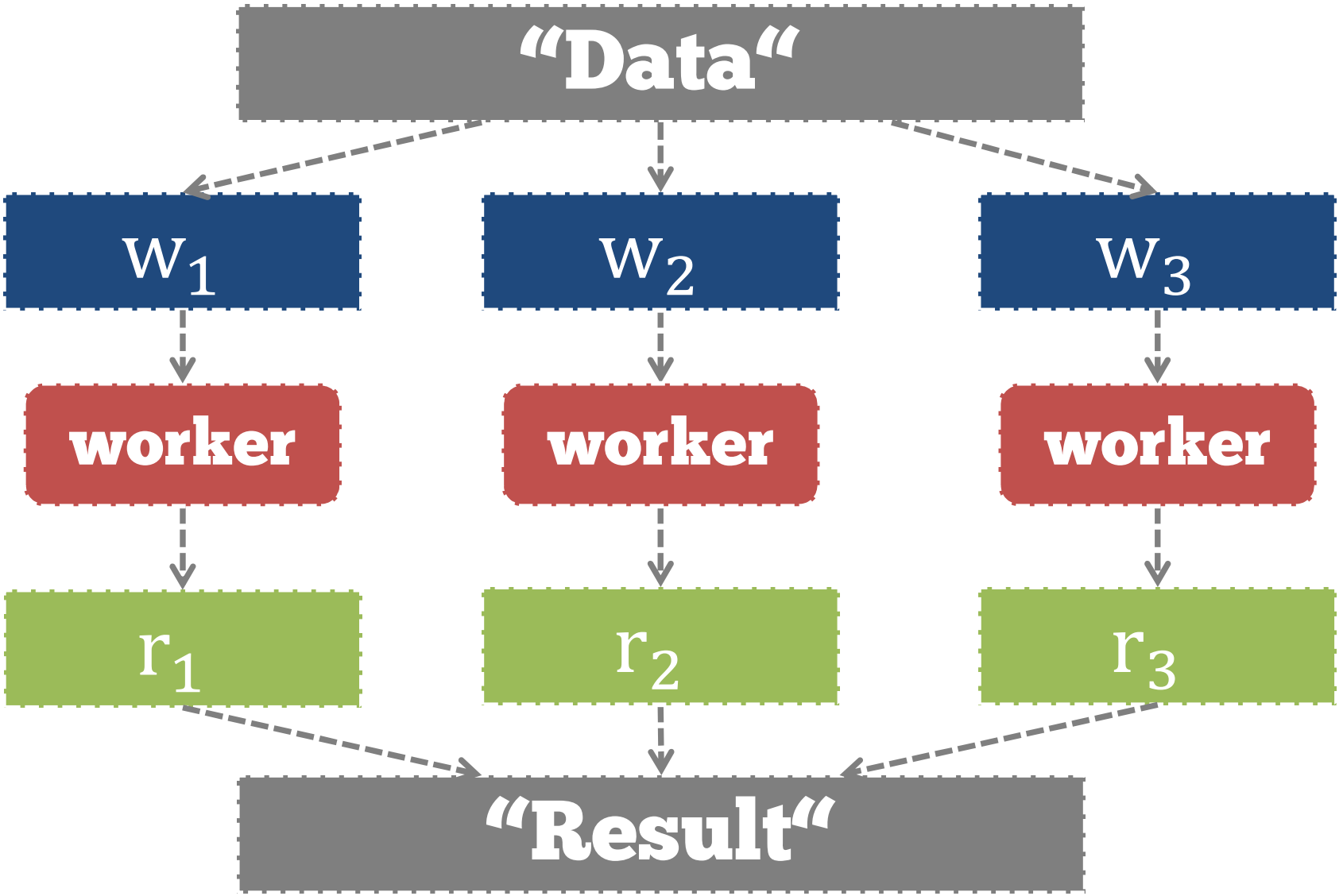


IT-Stammtisch DARMSTADT 07.11.13

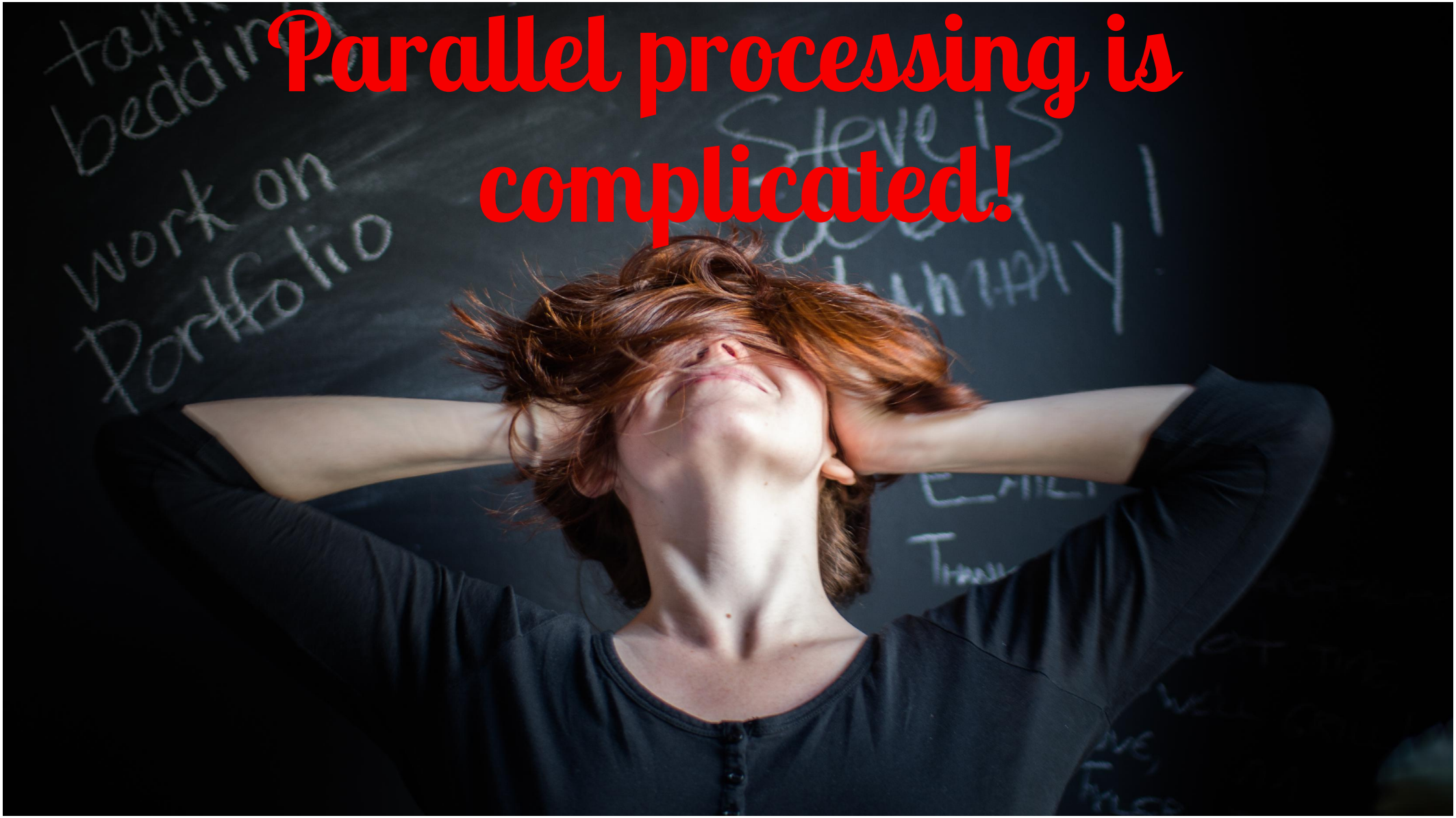
Why Hadoop?

codecentric

How to scale data?



But...



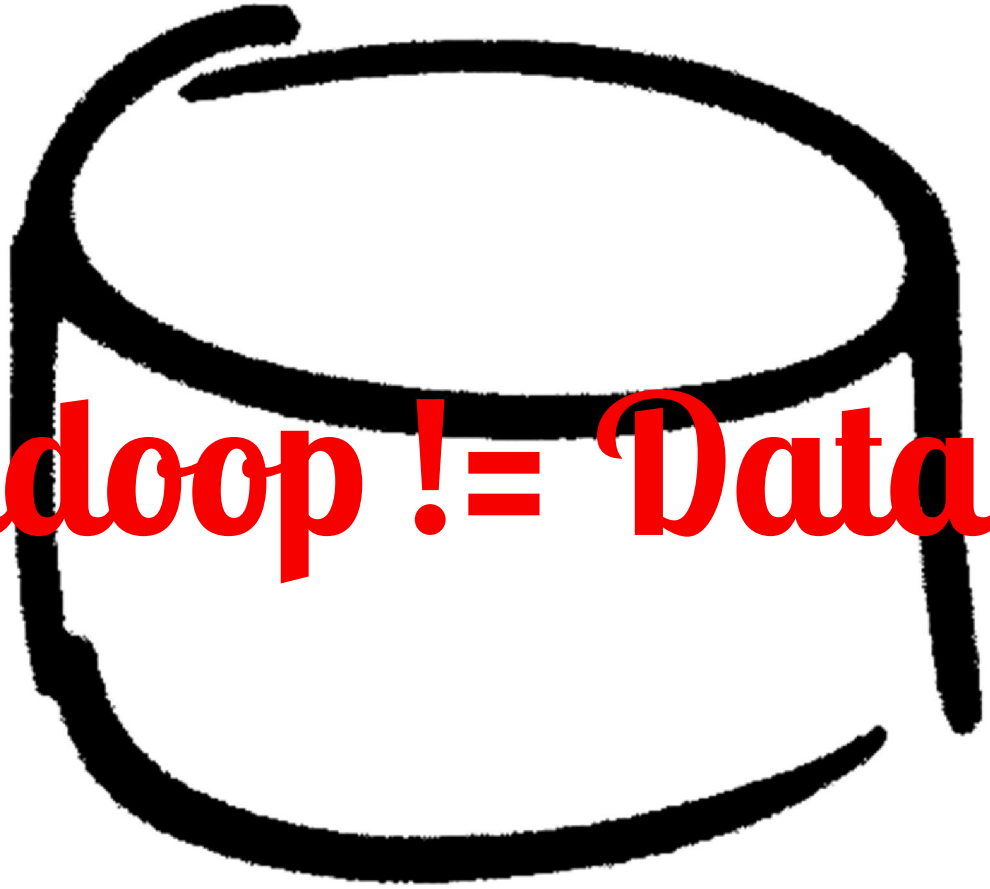
But...

IT-Stammtisch DARMSTADT 07.11.13

What is Hadoop?

codecentric

What is Hadoop?



Hadoop != Database

What is Hadoop?

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

IT-Stammtisch DARMSTADT

07.11.13

The Hadoop App Store

codecentric

IT-Stammtisch DARMSTADT 07.11.13

The Hadoop App Store

Apache Hadoop

- HDFS
- MapReduce
- Hadoop Ecosystem
- Hadoop YARN

Hadoop Distributions

+

- Test & Packaging
- Installation
- Monitoring
- Business Support

+

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

Big Data Suites

less — **Functionality** — more

codecentric

Agenda

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

OK, first things first!

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

IT-Stammtisch DARMSTADT 07.11.13

Data Storage

There is an app for that!

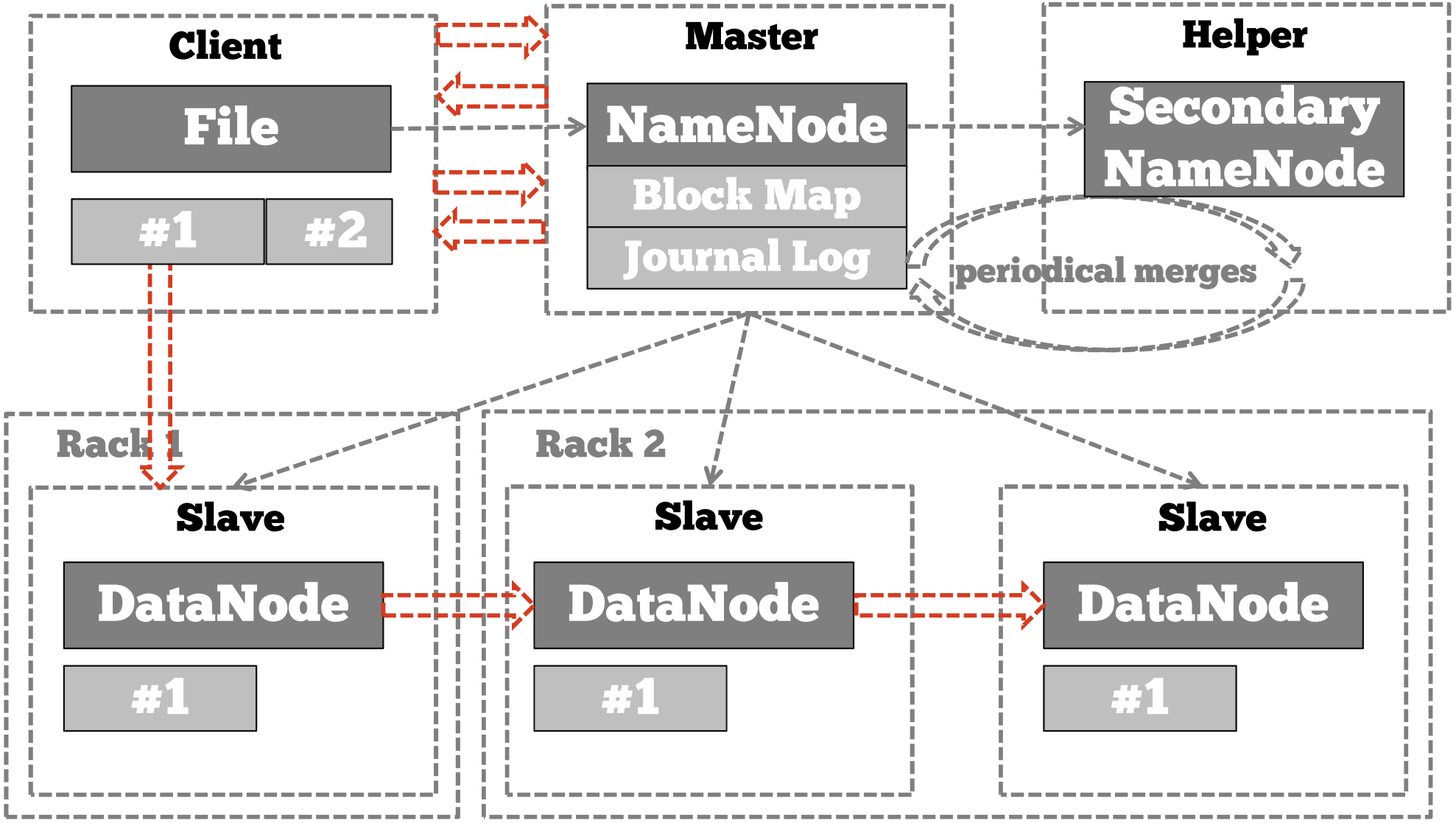
codecentric

Hadoop Distributed File System

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

Hadoop Distributed File System

HDFS Architecture



Let's have a look...

Data Processing

Data stored, check!

Now I want to create insights from my data!

IT-Stammtisch DARMSTADT 07.11.13

Data Processing

There is an app for that!

codecentric

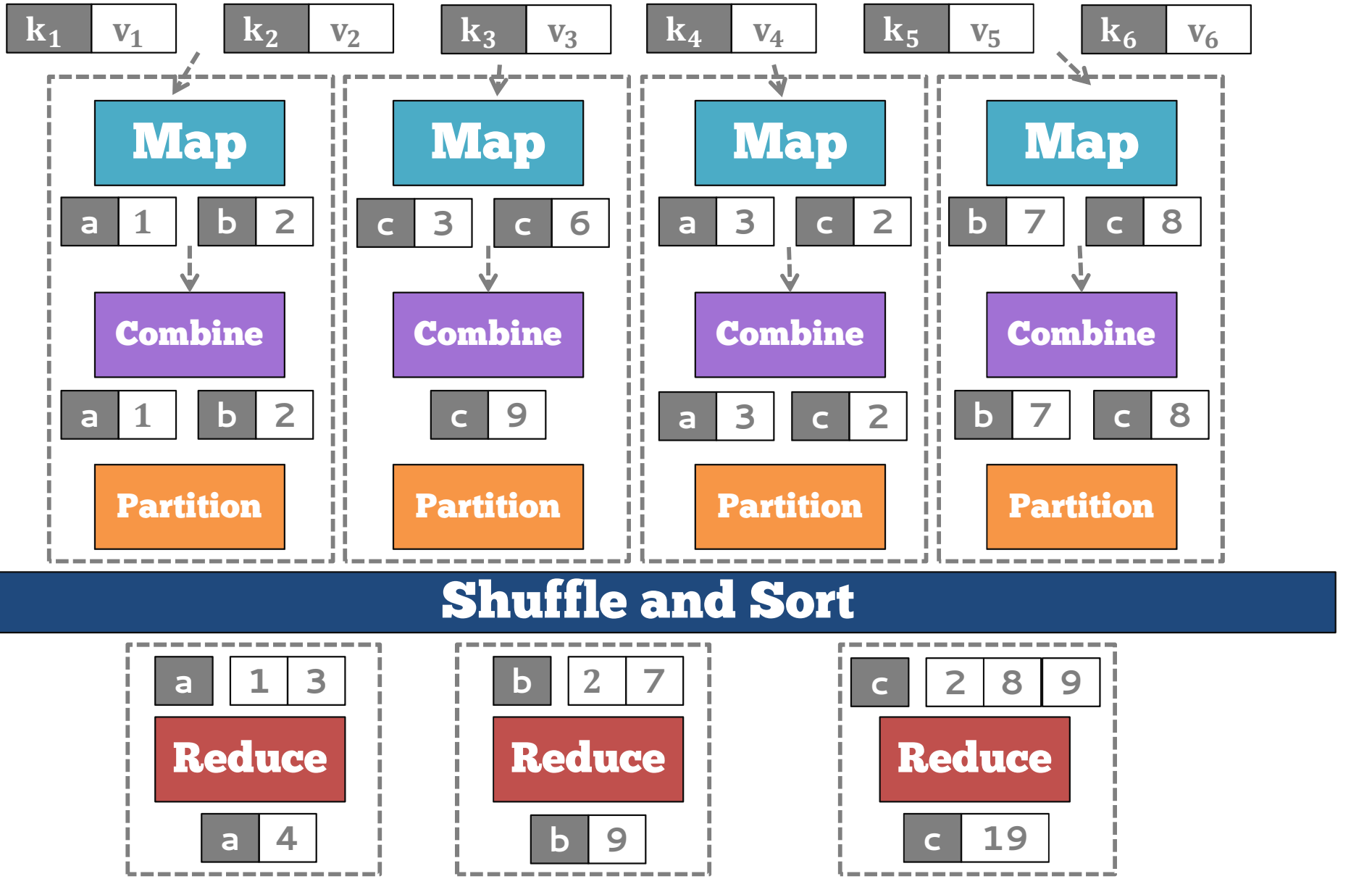
MapReduce

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

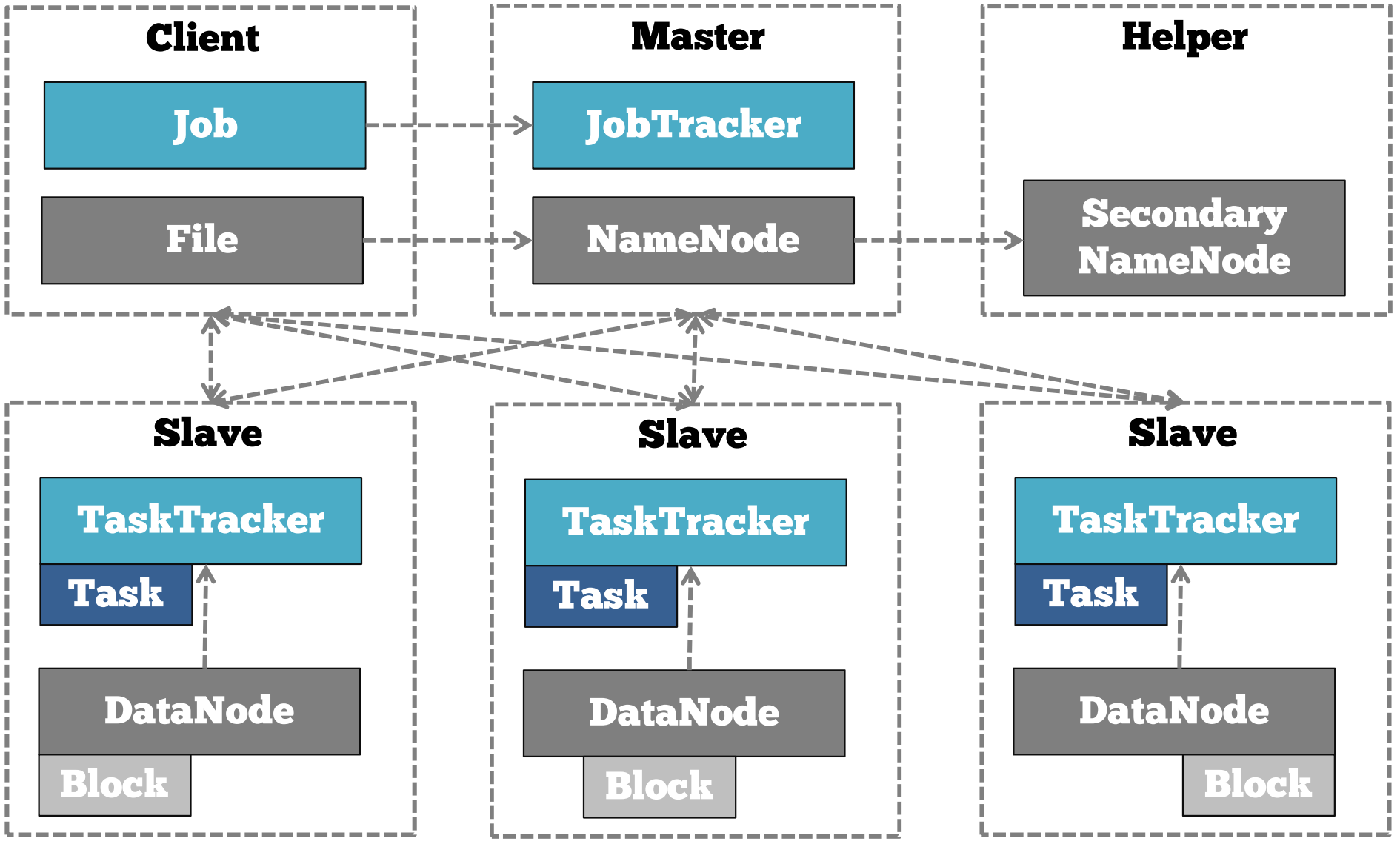
Typical large-data problem

- 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



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));
   }
}
```

Let's have a look...

Agenda

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

Scripting for Hadoop

Java for MapReduce?
I dunno, dude…

I'm more of a
scripting guy…

IT-Stammtisch DARMSTADT 07.11.13

Scripting for Hadoop

There is an app for that!

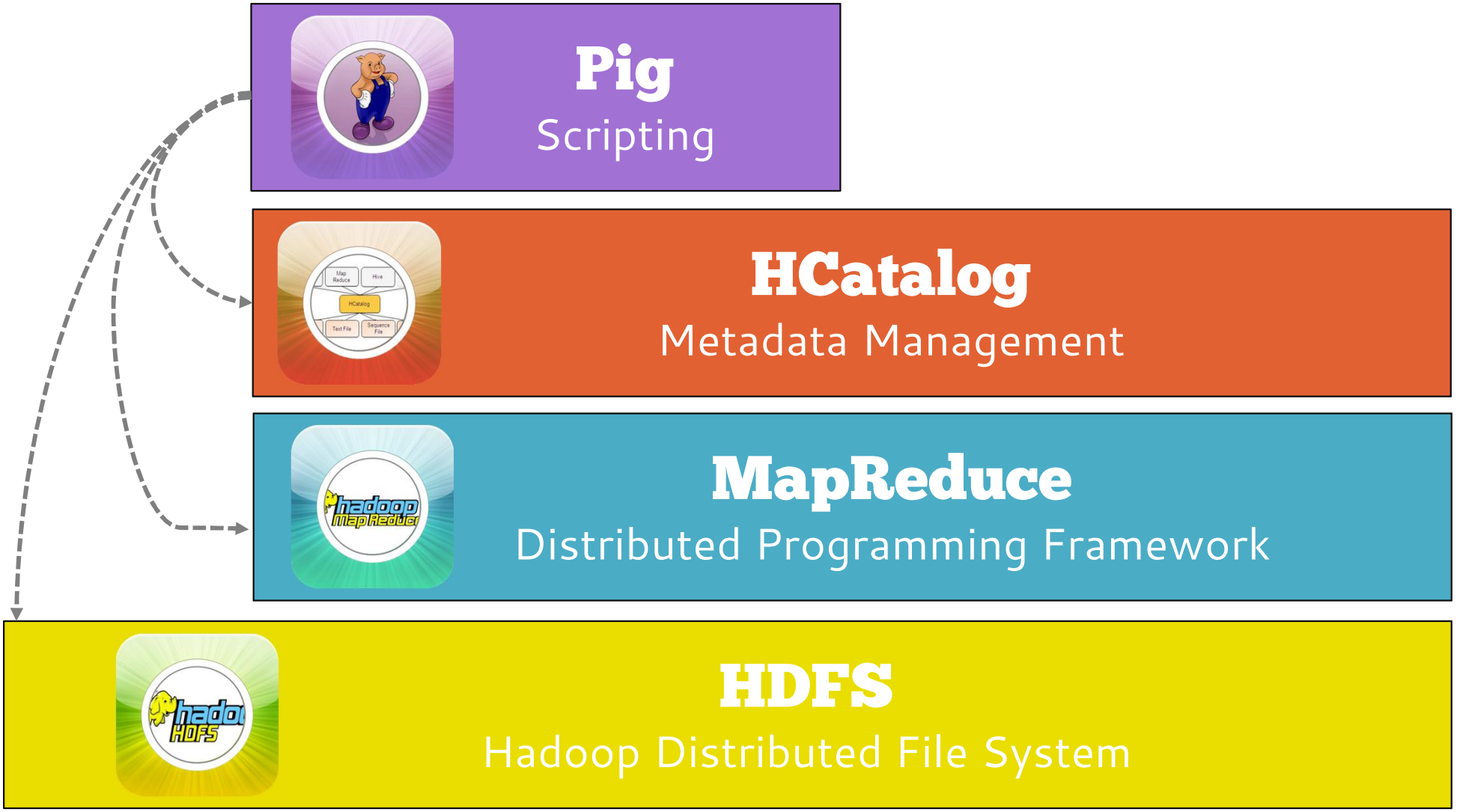
codecentric

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

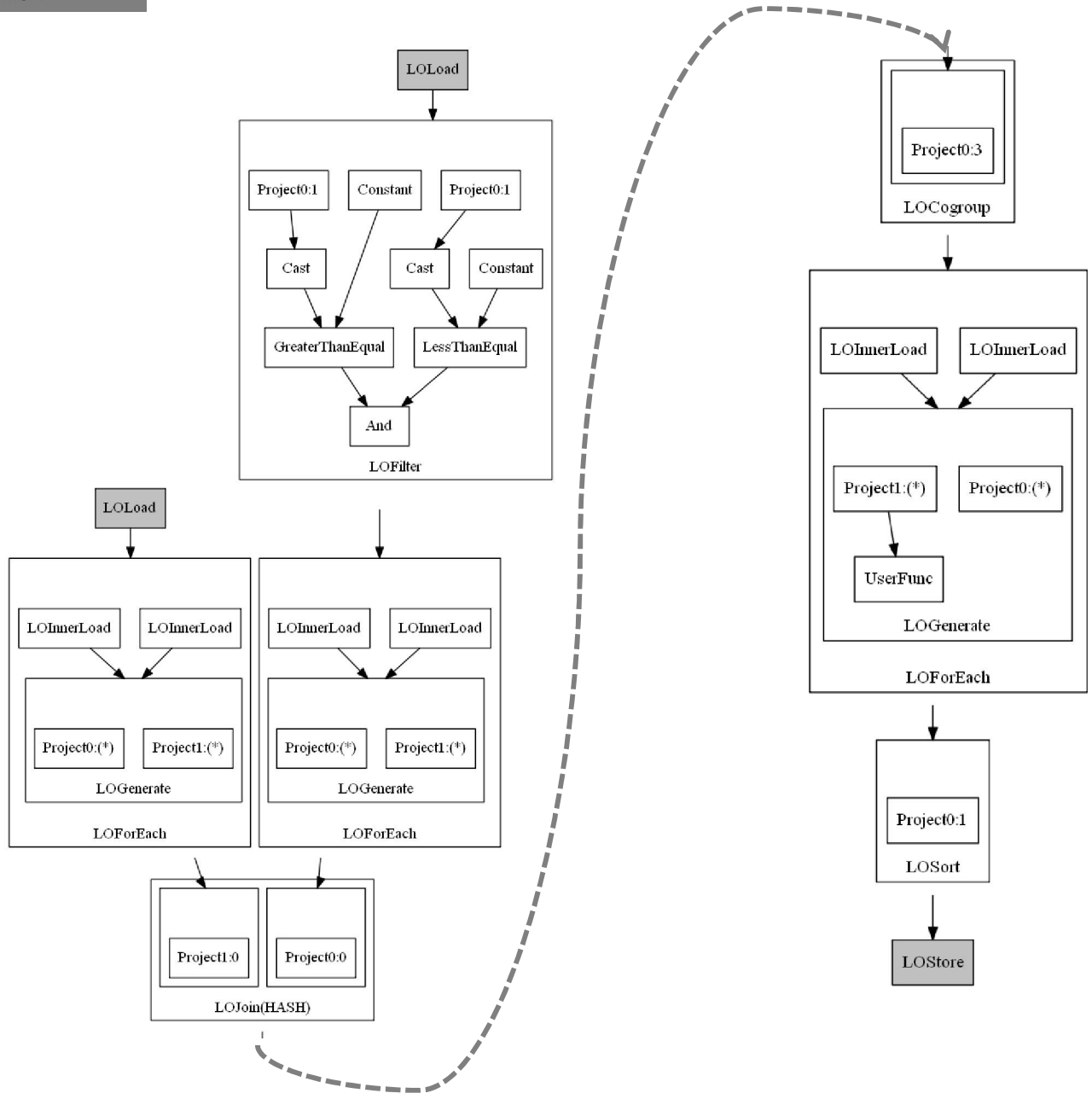
Pig in the Hadoop ecosystem



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



IT-Stammtisch DARMSTADT

07.11.13

Try that with Java...

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

import org.apache.hadoop.fs.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.MapReduceBase;
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.SequenceFileOutputFormat;
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;

public class MRExample {
    public static class LoadPages extends MapReduceBase
        implements Mapper<LongWritable, Text, Text, Text> {

        public void map(LongWritable k, Text val,
                OutputCollector<Text, Text> oc,
                Reporter reporter) throws IOException {
            // Pull the key out
            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);
            oc.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> oc,
                Reporter reporter) throws IOException {
            // Pull the key out
            String line = val.toString();
            int firstComma = line.indexOf(',');
            String value = line.substring(firstComma + 1);
            int age = Integer.parseInt(value);
            if (age < 18 || age > 25) return;
            String key = line.substring(0, firstComma);
            Text outKey = new Text(key);
            // Prepend an index to the value so we know which file
            // it came from.
            Text outVal = new Text("2" + value);
            oc.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> oc,
                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));

                reporter.setStatus("OK");
            }

            // Do the cross product and collect the values
            for (String s1 : first) {
                for (String s2 : second) {
                    String outVal = key + "," + s1 + "," + s2;
                    oc.collect(null, new Text(outVal));
                    reporter.setStatus("OK");
                }
            }
        }
    }
    public static class LoadJoined extends MapReduceBase
        implements Mapper<Text, Text, Text, LongWritable> {

        public void map(
                Text k,
                Text val,
                OutputCollector<Text, LongWritable> oc,
                Reporter reporter) throws IOException {
            // Find the url
            String line = val.toString();
            int firstComma = line.indexOf(',');
            int secondComma = line.indexOf(',', firstComma);
            String key = line.substring(firstComma, 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(key);
            oc.collect(outKey, new LongWritable(1L));
        }
    }
    public static class ReduceUrls extends MapReduceBase
        implements Reducer<Text, LongWritable, WritableComparable,
Writable> {

        public void reduce(
                Text key,
                Iterator<LongWritable> iter,
                OutputCollector<WritableComparable, Writable> oc,
                Reporter reporter) throws IOException {
            // Add up all the values we see

            long sum = 0;
            while (iter.hasNext()) {
                sum += iter.next().get();
                reporter.setStatus("OK");
            }

            oc.collect(key, new LongWritable(sum));
        }
    }
    public static class LoadClicks extends MapReduceBase
        implements Mapper<WritableComparable, Writable, LongWritable,
Text> {

        public void map(
                WritableComparable key,
                Writable val,
                OutputCollector<LongWritable, Text> oc,
                Reporter reporter) throws IOException {
            oc.collect((LongWritable)val, (Text)key);
        }
    }
    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> oc,
            Reporter reporter) throws IOException {

            // Only output the first 100 records
            while (count < 100 && iter.hasNext()) {
                oc.collect(key, iter.next());
                count++;
            }
        }
    }
    public static void main(String[] args) throws IOException {
        JobConf lp = new JobConf(MRExample.class);
        lp.setJobName("Load Pages");
        lp.setInputFormat(TextInputFormat.class);
        lp.setOutputKeyClass(Text.class);
        lp.setOutputValueClass(Text.class);
        lp.setMapperClass(LoadPages.class);
        FileInputFormat.addInputPath(lp, new
Path("/user/gates/pages"));
        FileOutputFormat.setOutputPath(lp,
                new Path("/user/gates/tmp/indexed_pages"));
        lp.setNumReduceTasks(0);
        Job loadPages = new Job(lp);

        JobConf lfu = new JobConf(MRExample.class);
        lfu.setJobName("Load and Filter Users");
        lfu.setInputFormat(TextInputFormat.class);
        lfu.setOutputKeyClass(Text.class);
        lfu.setOutputValueClass(Text.class);
        lfu.setMapperClass(LoadAndFilterUsers.class);
        FileInputFormat.addInputPath(lfu, new
Path("/user/gates/users"));
        FileOutputFormat.setOutputPath(lfu,
                new Path("/user/gates/tmp/filtered_users"));
        lfu.setNumReduceTasks(0);
        Job loadUsers = new Job(lfu);

        JobConf join = new JobConf(MRExample.class);
        join.setJobName("Join Users and Pages");
        join.setInputFormat(KeyValueTextInputFormat.class);
        join.setOutputKeyClass(Text.class);
        join.setOutputValueClass(Text.class);
        join.setMapperClass(IdentityMapper.class);
        join.setReducerClass(Join.class);
        FileInputFormat.addInputPath(join, new
Path("/user/gates/tmp/indexed_pages"));
        FileInputFormat.addInputPath(join, new
Path("/user/gates/tmp/filtered_users"));
        FileOutputFormat.setOutputPath(join, new
Path("/user/gates/tmp/joined"));
        join.setNumReduceTasks(50);
        Job joinJob = new Job(join);
        joinJob.addDependingJob(loadPages);
        joinJob.addDependingJob(loadUsers);

        JobConf group = new JobConf(MRExample.class);
        group.setJobName("Group URLs");
        group.setInputFormat(KeyValueTextInputFormat.class);
        group.setOutputKeyClass(Text.class);
        group.setOutputValueClass(LongWritable.class);
        group.setOutputFormat(SequenceFileOutputFormat.class);
        group.setMapperClass(LoadJoined.class);
        group.setCombinerClass(ReduceUrls.class);
        group.setReducerClass(ReduceUrls.class);
        FileInputFormat.addInputPath(group, new
Path("/user/gates/tmp/joined"));
        FileOutputFormat.setOutputPath(group, new
Path("/user/gates/tmp/grouped"));
        group.setNumReduceTasks(50);
        Job groupJob = new Job(group);
        groupJob.addDependingJob(joinJob);

        JobConf top100 = new JobConf(MRExample.class);
        top100.setJobName("Top 100 sites");
        top100.setInputFormat(SequenceFileInputFormat.class);
        top100.setOutputKeyClass(LongWritable.class);
        top100.setOutputValueClass(Text.class);
        top100.setOutputFormat(SequenceFileOutputFormat.class);
        top100.setMapperClass(LoadClicks.class);
        top100.setCombinerClass(LimitClicks.class);
        top100.setReducerClass(LimitClicks.class);
        FileInputFormat.addInputPath(top100, new
Path("/user/gates/tmp/grouped"));
        FileOutputFormat.setOutputPath(top100, new
Path("/user/gates/top100sitesforusers18to25"));
        top100.setNumReduceTasks(1);
        Job limit = new Job(top100);
        limit.addDependingJob(groupJob);

        JobControl jc = new JobControl("Find top 100 sites for users
18 to 25");
        jc.addJob(loadPages);
        jc.addJob(loadUsers);
        jc.addJob(joinJob);
        jc.addJob(groupJob);
        jc.addJob(limit);
        jc.run();
    }
}
```

codecentric

Pig

Let's have a look...

OK, Pig seems quite useful…

But I'm more of a SQL person…

IT-Stammtisch DARMSTADT 07.11.13

SQL for Hadoop

There is an app for that!

codecentric

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

IT-Stammtisch DARMSTADT 07.11.13

Hive in the Hadoop ecosystem

Pig
Scripting

Hive
Query

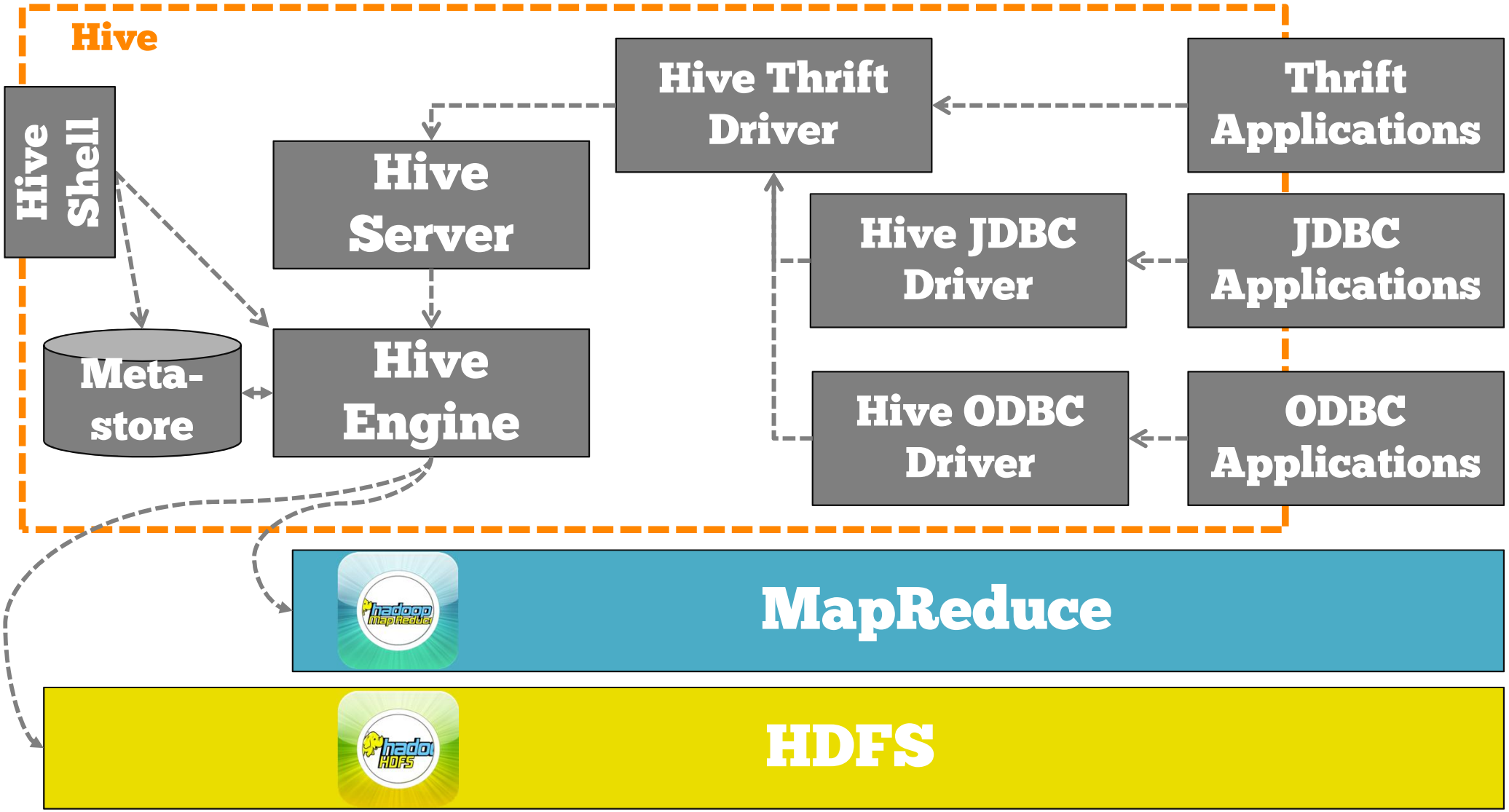
HCatalog
Metadata Management

MapReduce
Distributed Programming Framework

HDFS
Hadoop Distributed File System

codecentric

Hive Architecture



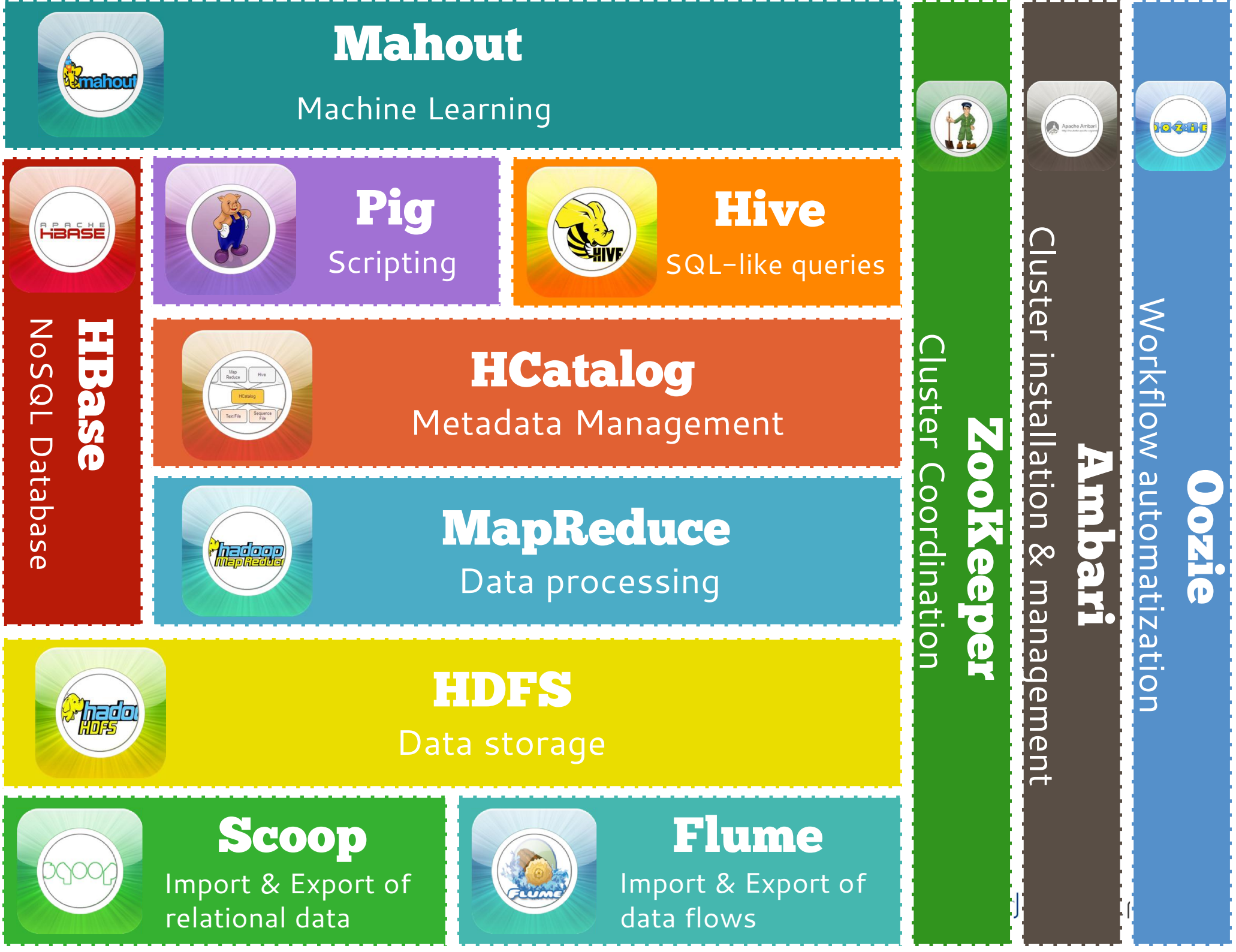
Hive Example

```
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!

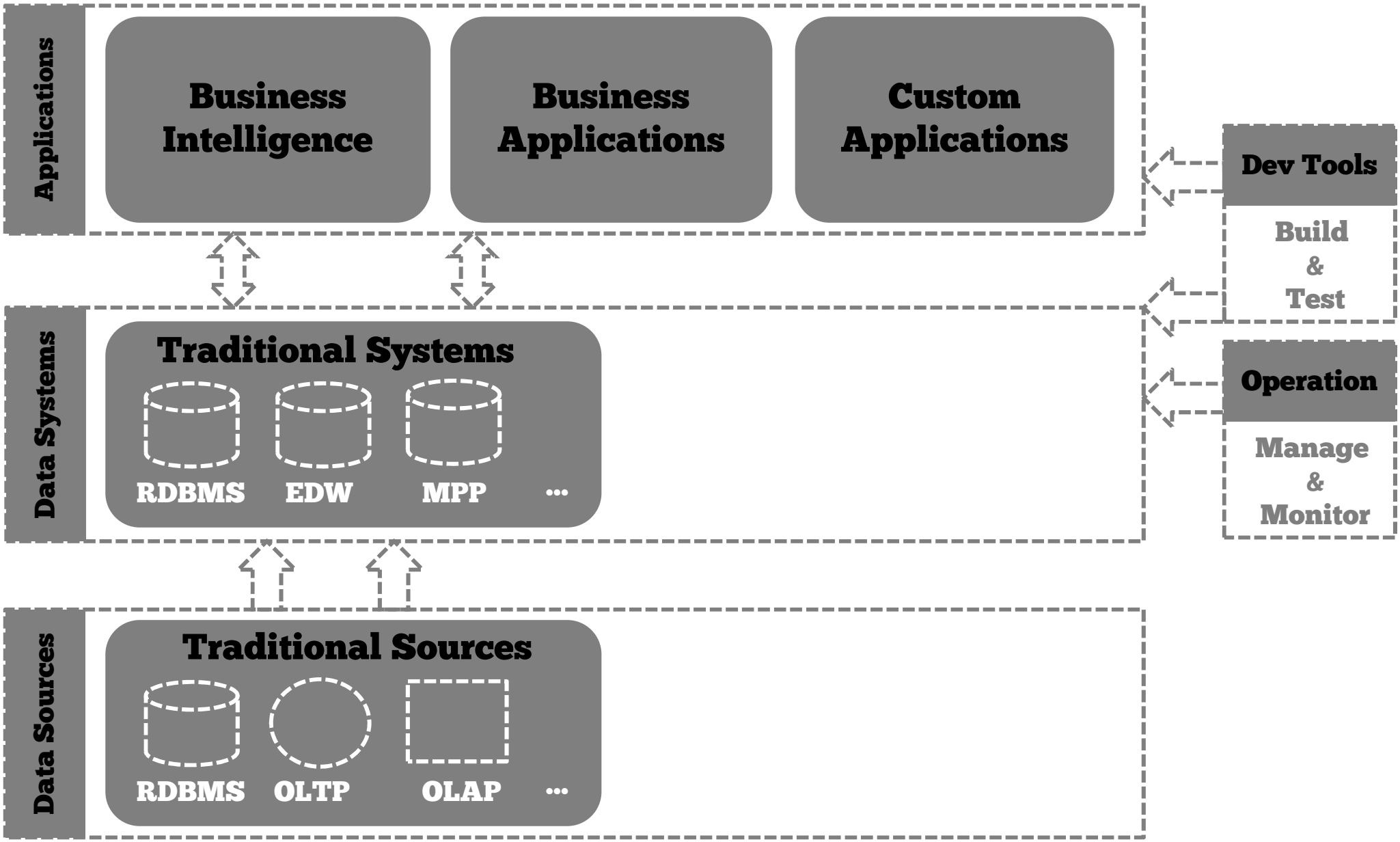
More components of the Hadoop Ecosystem


Mahout
Machine Learning
Pig
Scripting
Hive
SQL-like queries
HBase
NoSQL Database
HCatalog
Metadata Management
MapReduce
Data processing
HDFS
Data storage
Scoop
Import & Export of relational data
Flume
Import & Export of data flows
ZooKeeper
Cluster Coordination
Ambari
Cluster installation & management
Oozie
Workflow automatization

Agenda

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

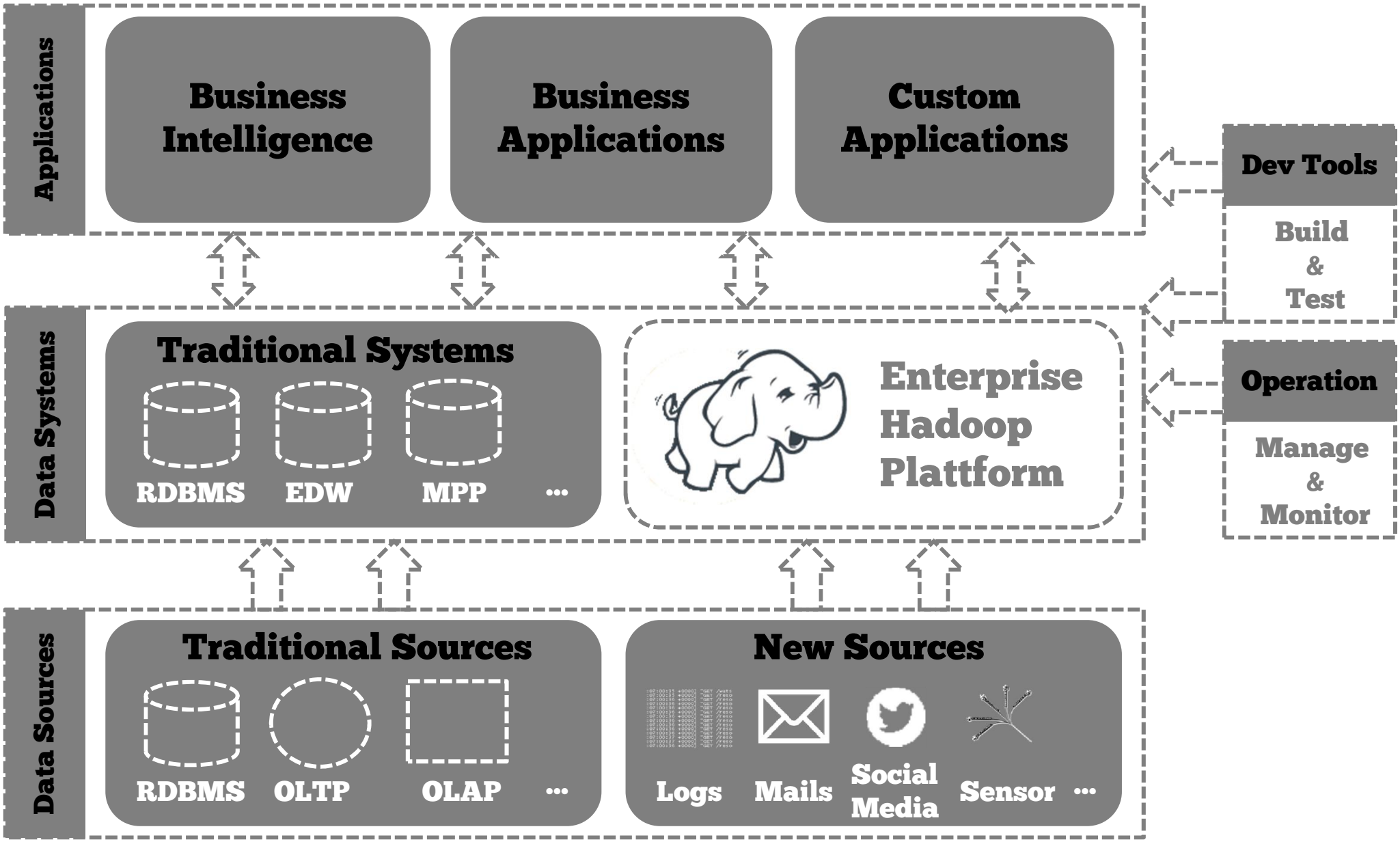
Classical enterprise platform



IT-Stammtisch DARMSTADT

07.11.13

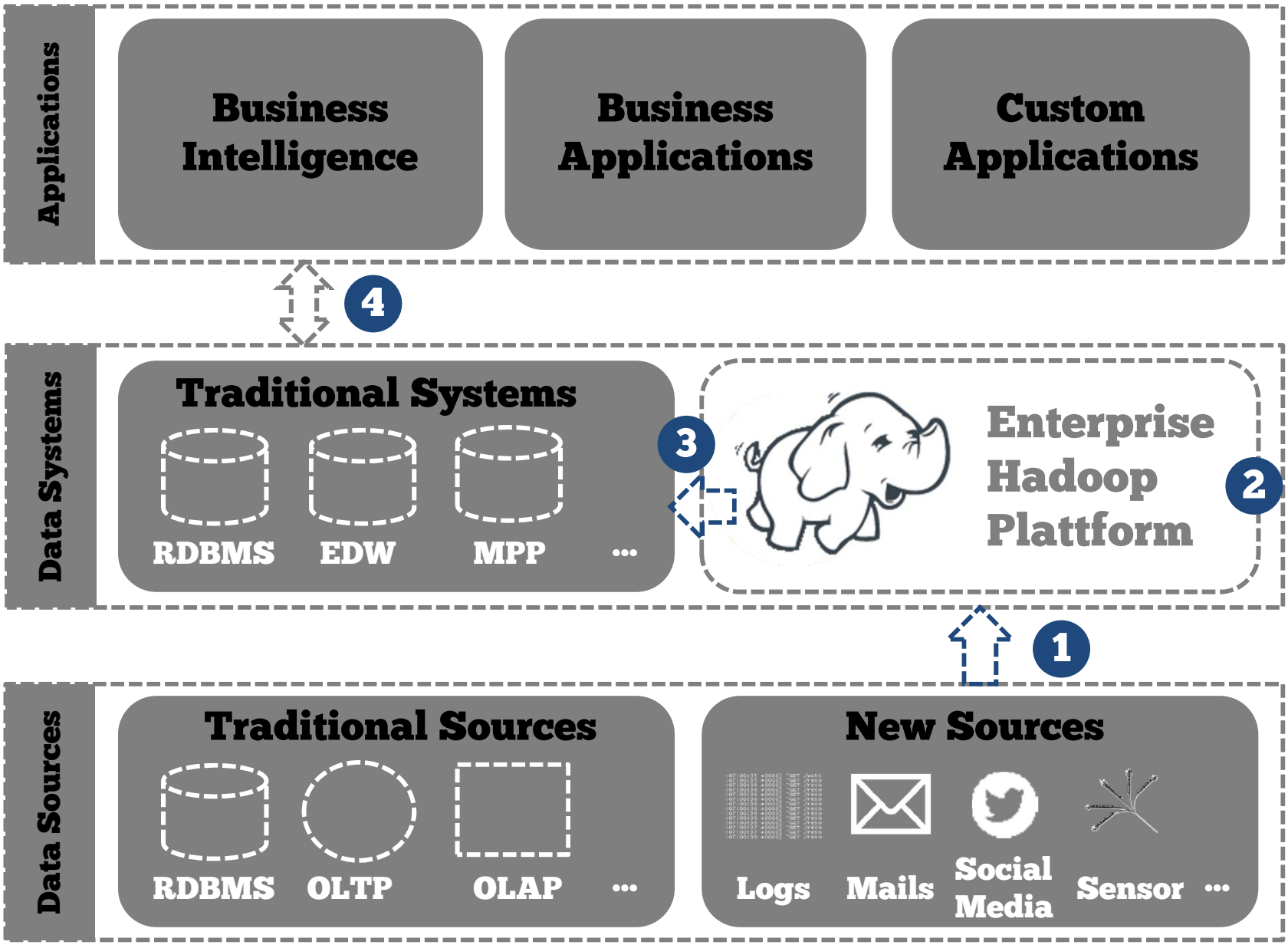
Big Data Platform



codecentric

IT-Stammtisch DARMSTADT
07.11.13

Pattern #1: Refine data

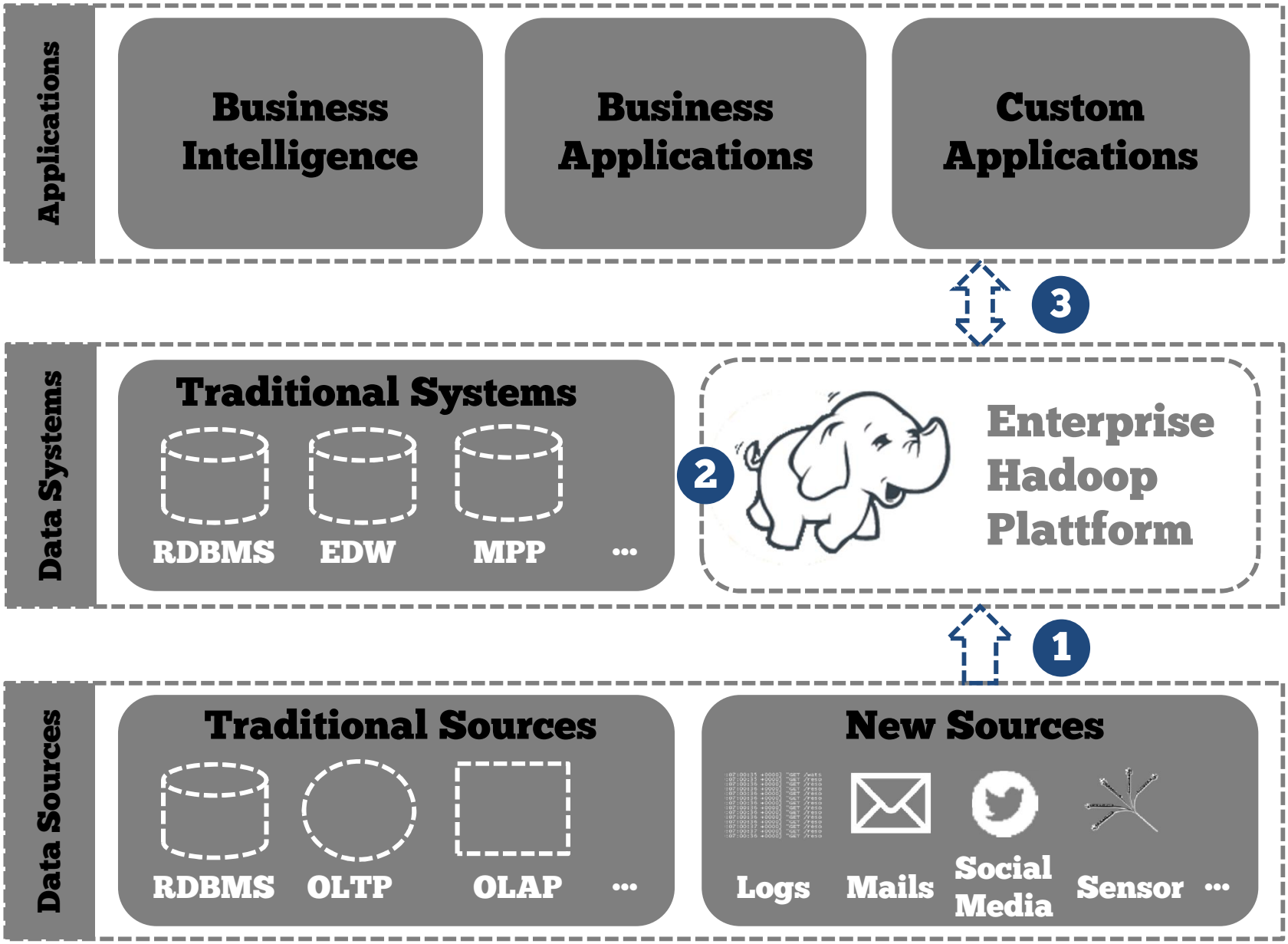


1. Capture all data
2. Process the data
3. Exchange using traditional systems
4. Process & Visualize with traditional applications

codecentric

IT-Stammtisch DARMSTADT
07.11.13

Pattern #2: Explore data

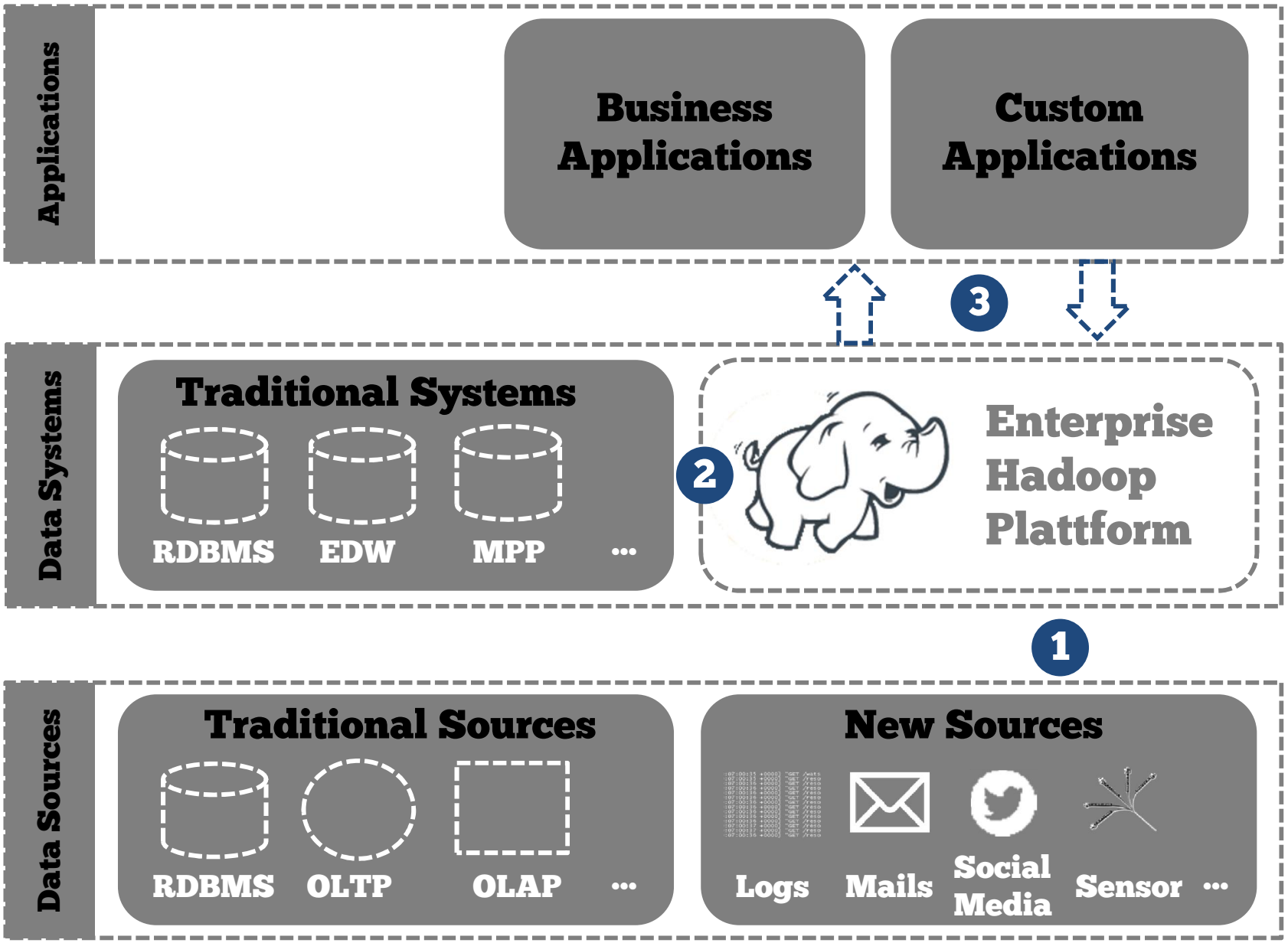


1. **Capture all data**
2. **Process the data**
3. **Explore the data using applications with support for Hadoop**

codecentric

IT-Stammtisch DARMSTADT
07.11.13

Pattern #3: Enrich data



1. Capture all data
2. Process the data
3. Directly ingest the data

codecentric

Bringing it all together...

One example...

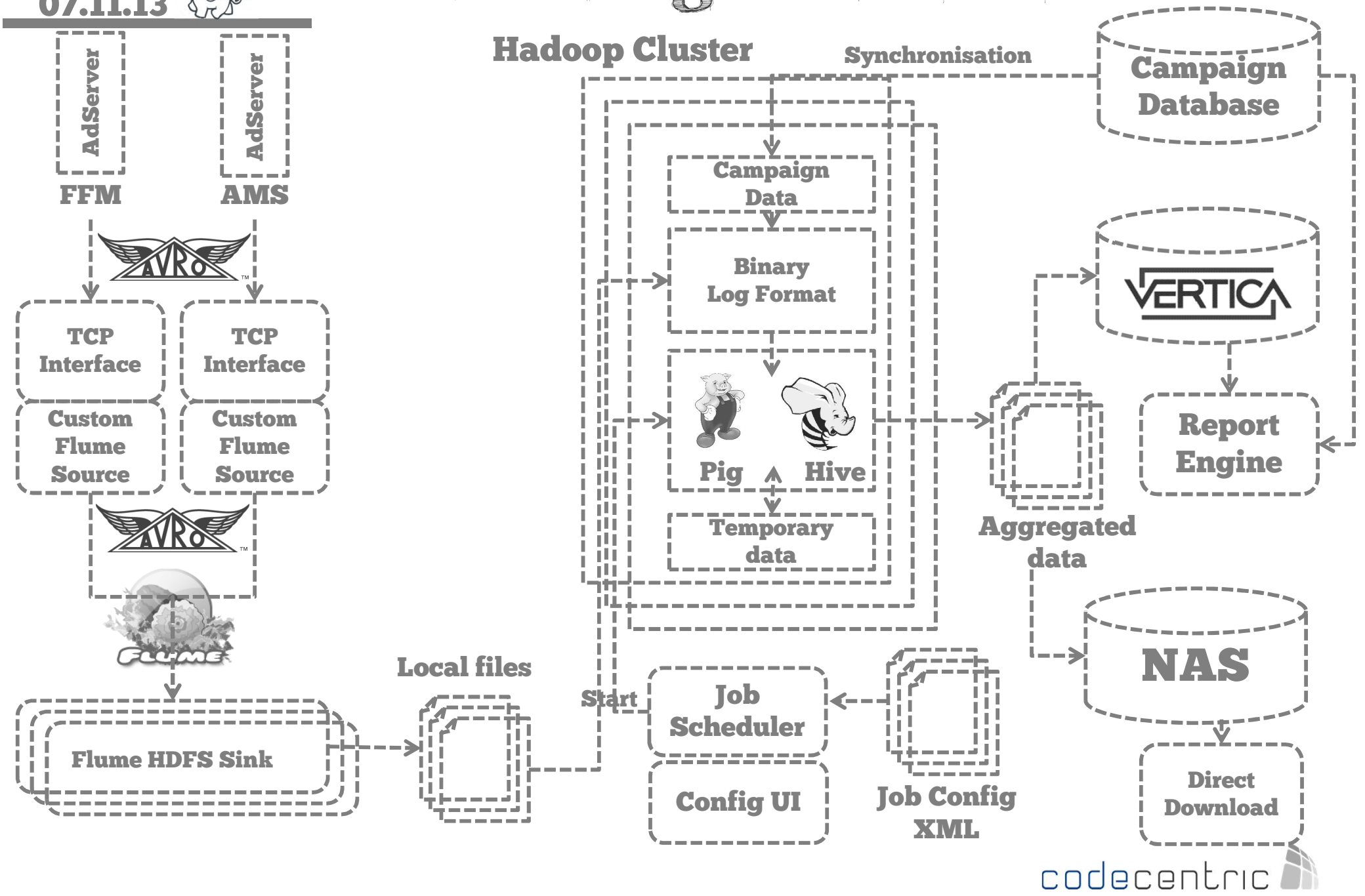
Digital Advertising

- 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

IT-Stammtisch DARMSTADT

07.11.13

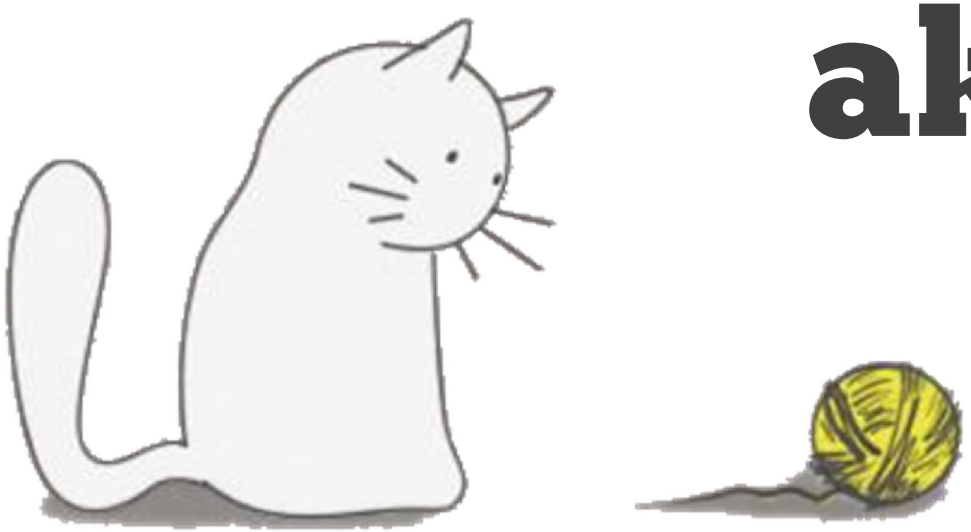
AdServing Architecture



codecentric

What's next?

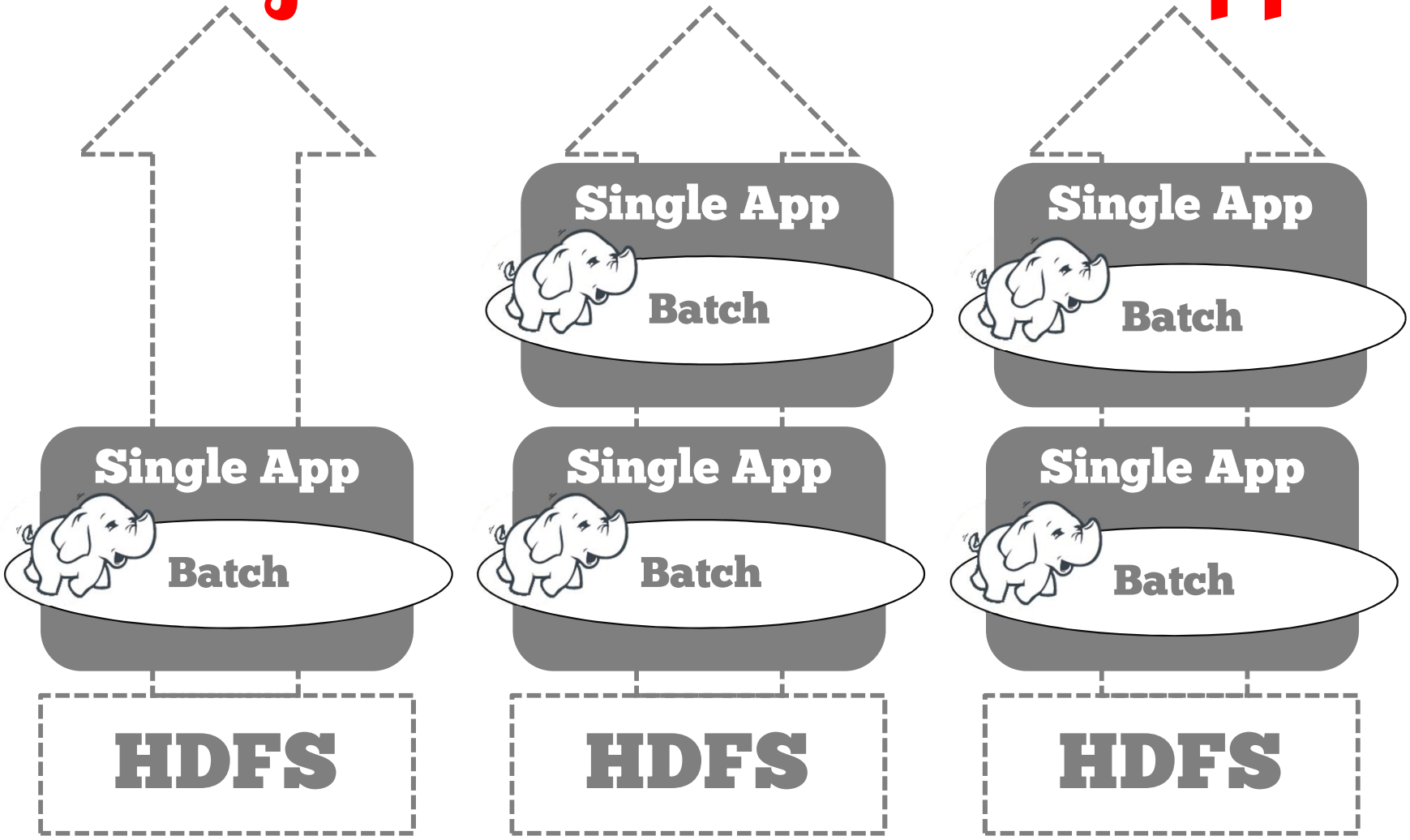
Hadoop 2.0 aka YARN



codecentric

Hadoop 1.0

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

Hadoop 1.0

MapReduce
Cluster resource mgmt.
+ data processing

HDFS
Redundant, reliable storage

Multi-purpose platform

Batch, Interactive, Streaming, ...

Hadoop 2.0

MapReduce
Data processing

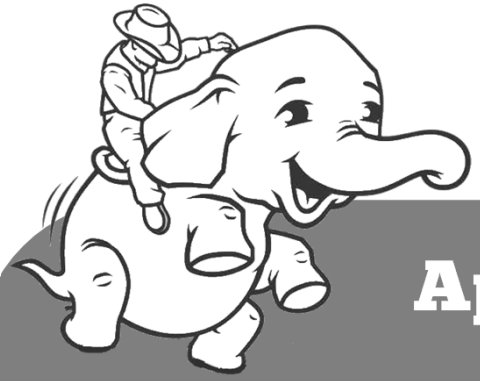
Others
Data processing

YARN
Cluster resource management

HDFS 2.0
Redundant, reliable storage

IT-Stammtisch DARMSTADT 07.11.13

Taking Hadoop beyond batch



Store all data in one place

Interact with data in multiple ways

Applications run natively in Hadoop

Batch	Interactive	Online	Streaming	Graph	In-Memory	Other
MapReduce	Tez	HOYA	Storm, ...	Giraph	Spark	Search, ...

YARN

Cluster resource management

HDFS 2.0

Redundant, reliable storage

codecentric

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**

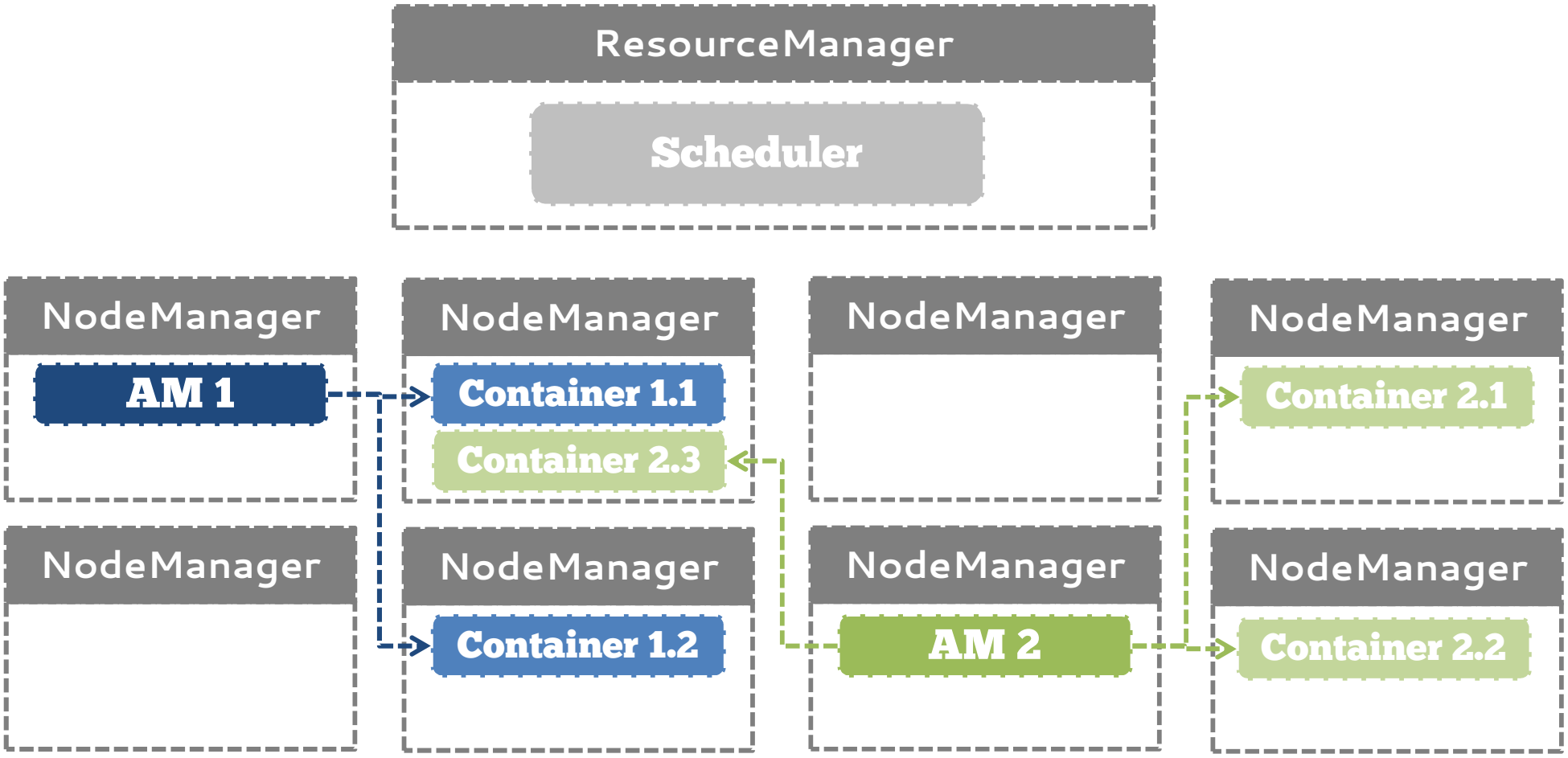
Hadoop 2.0 Projects

- **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

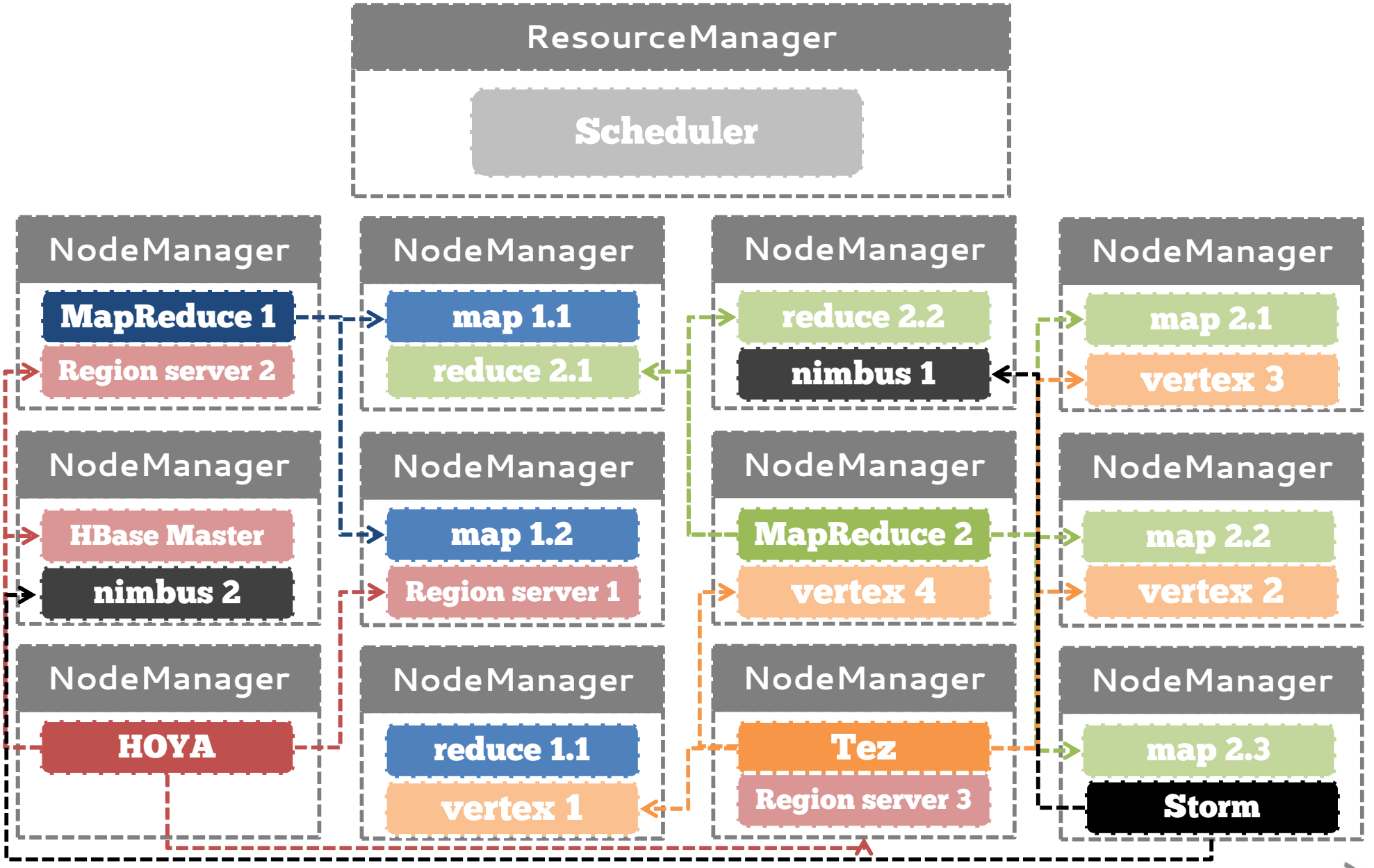


YARN: Architecture

- **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

IT-Stammtisch DARMSTADT 07.11.13

YARN: Architecture



codecentric

Hadoop 2.0 Projects

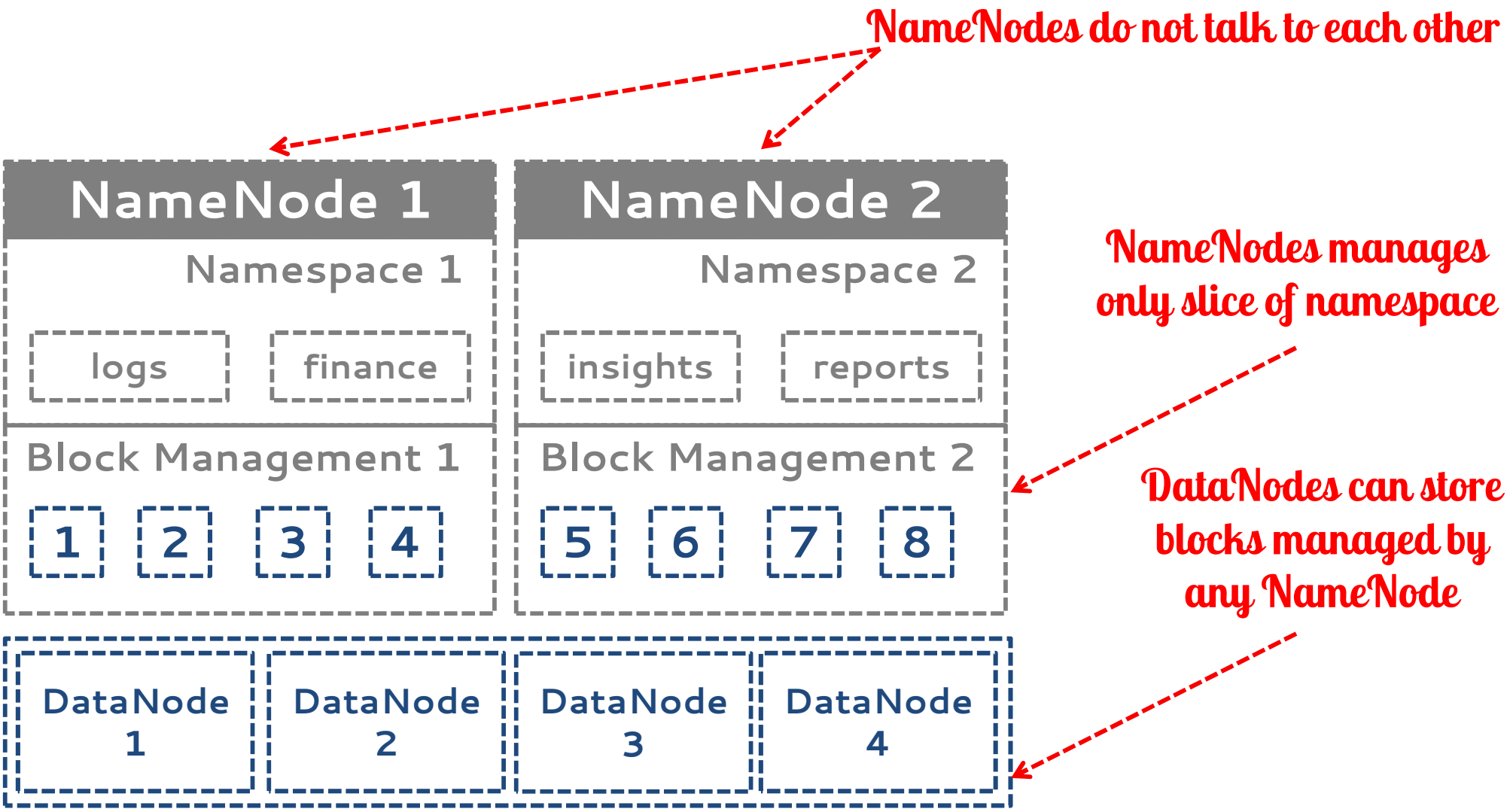
- **YARN**
- **HDFS Federation aka HDFS 2.0**
- **Stinger & Tez aka Hive 2.0**

HDFS Federation

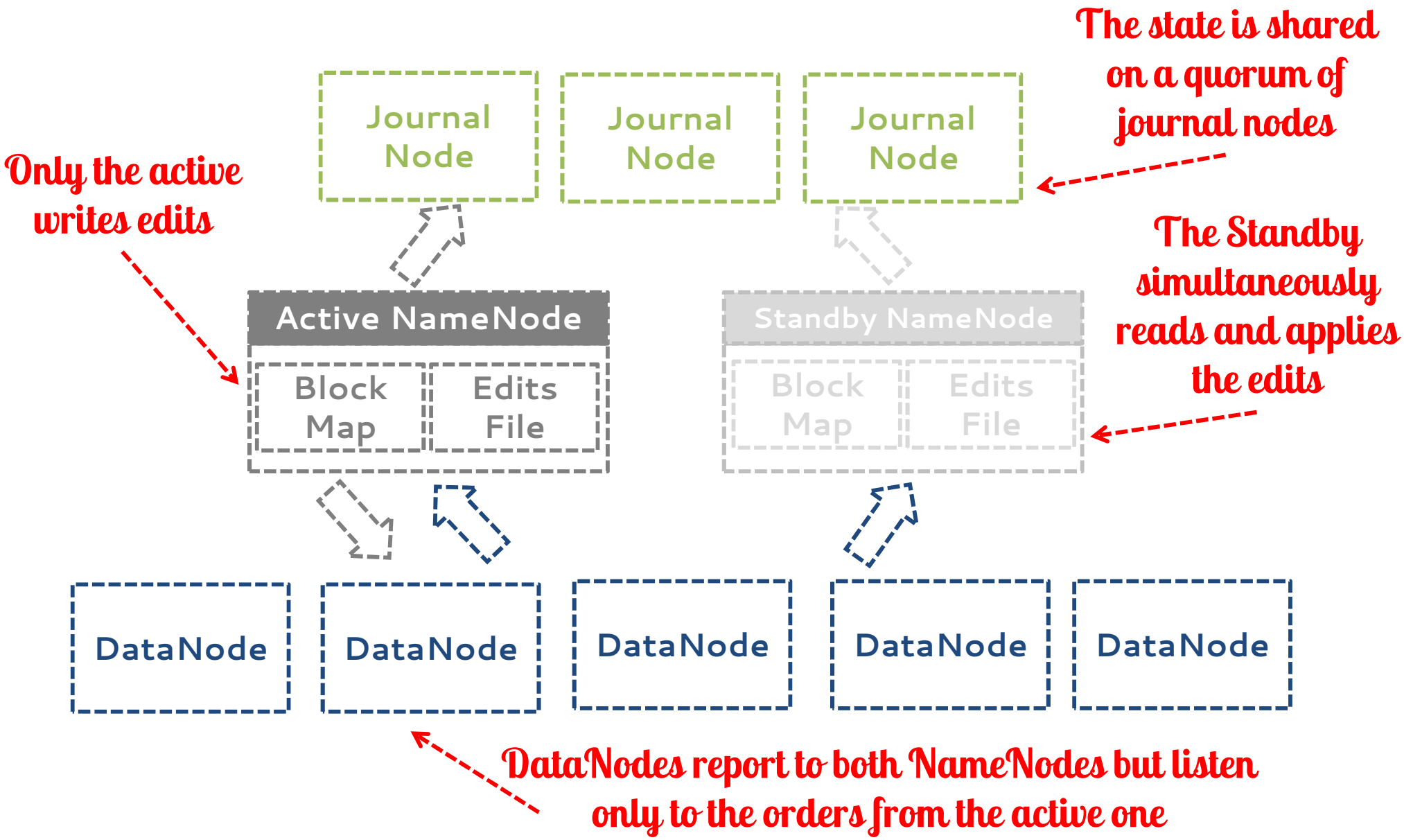
- **Removes tight coupling of Block Storage and Namespace**
- **Scalability & Isolation**
- **High Availability**
- **Increased performance**

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

HDFS Federation: Architecture



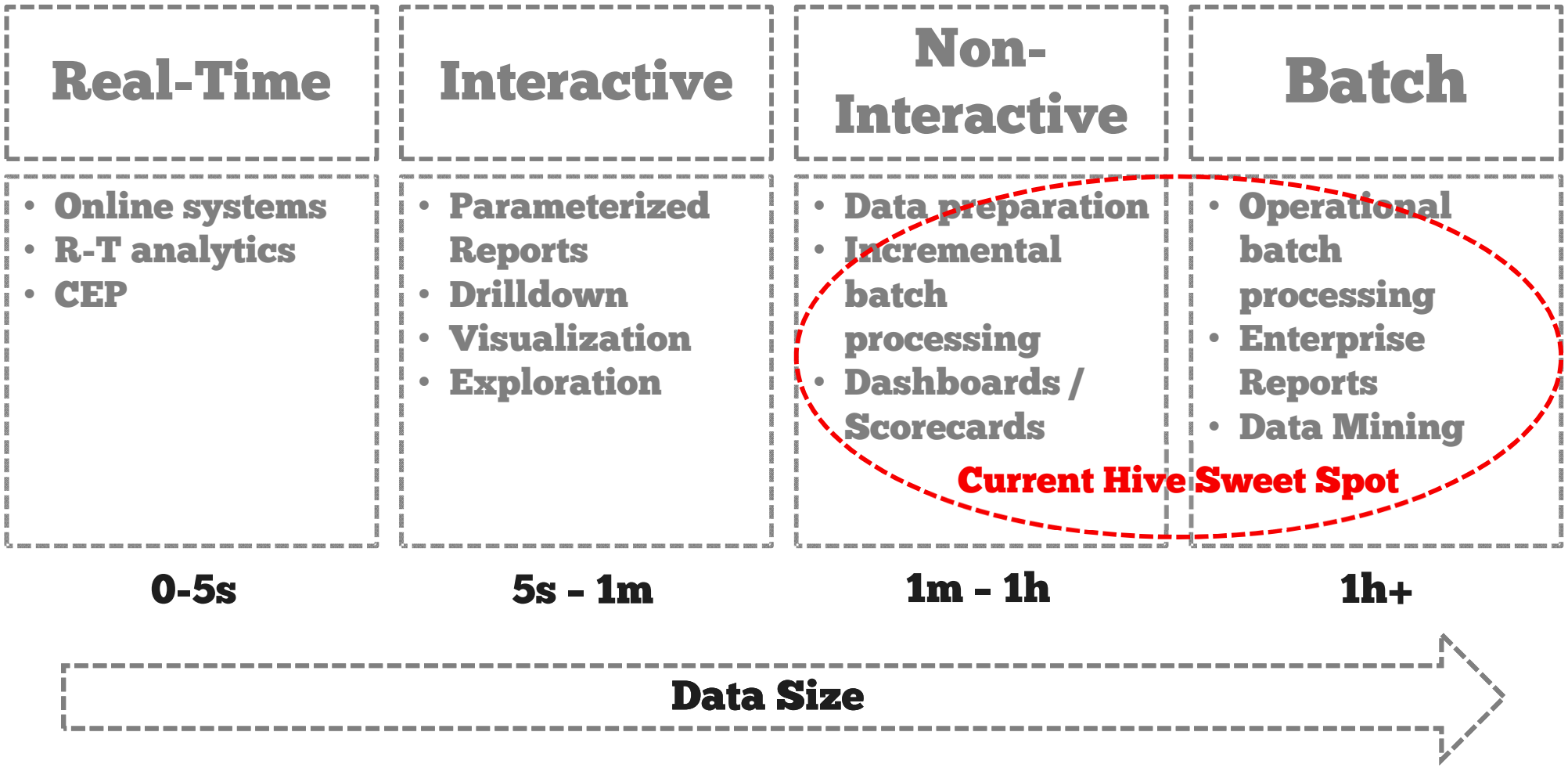
HDFS: Quorum based storage



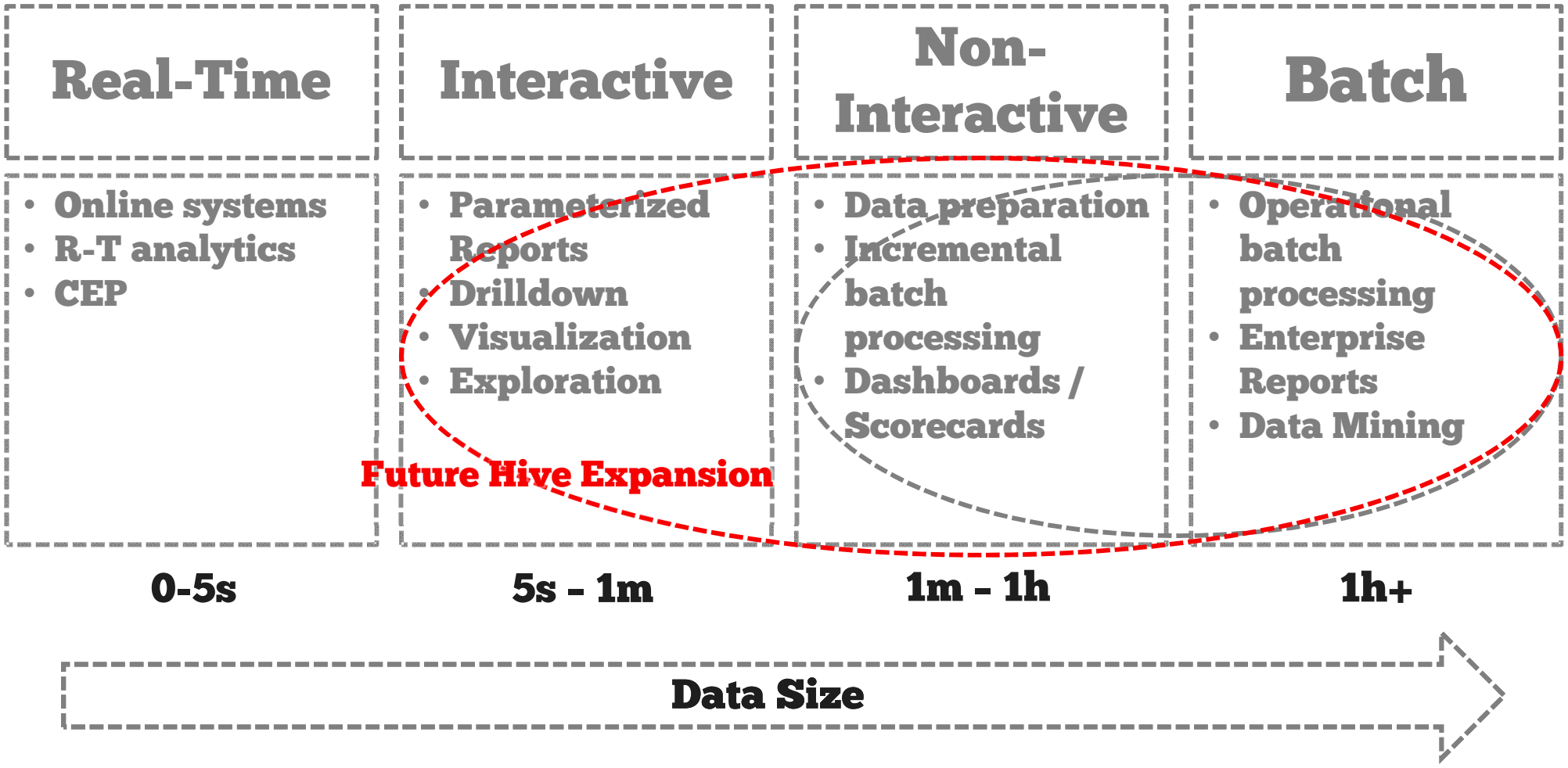
Hadoop 2.0 Projects

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

Hive: Current Focus Area



Stinger: Extending the sweet spot



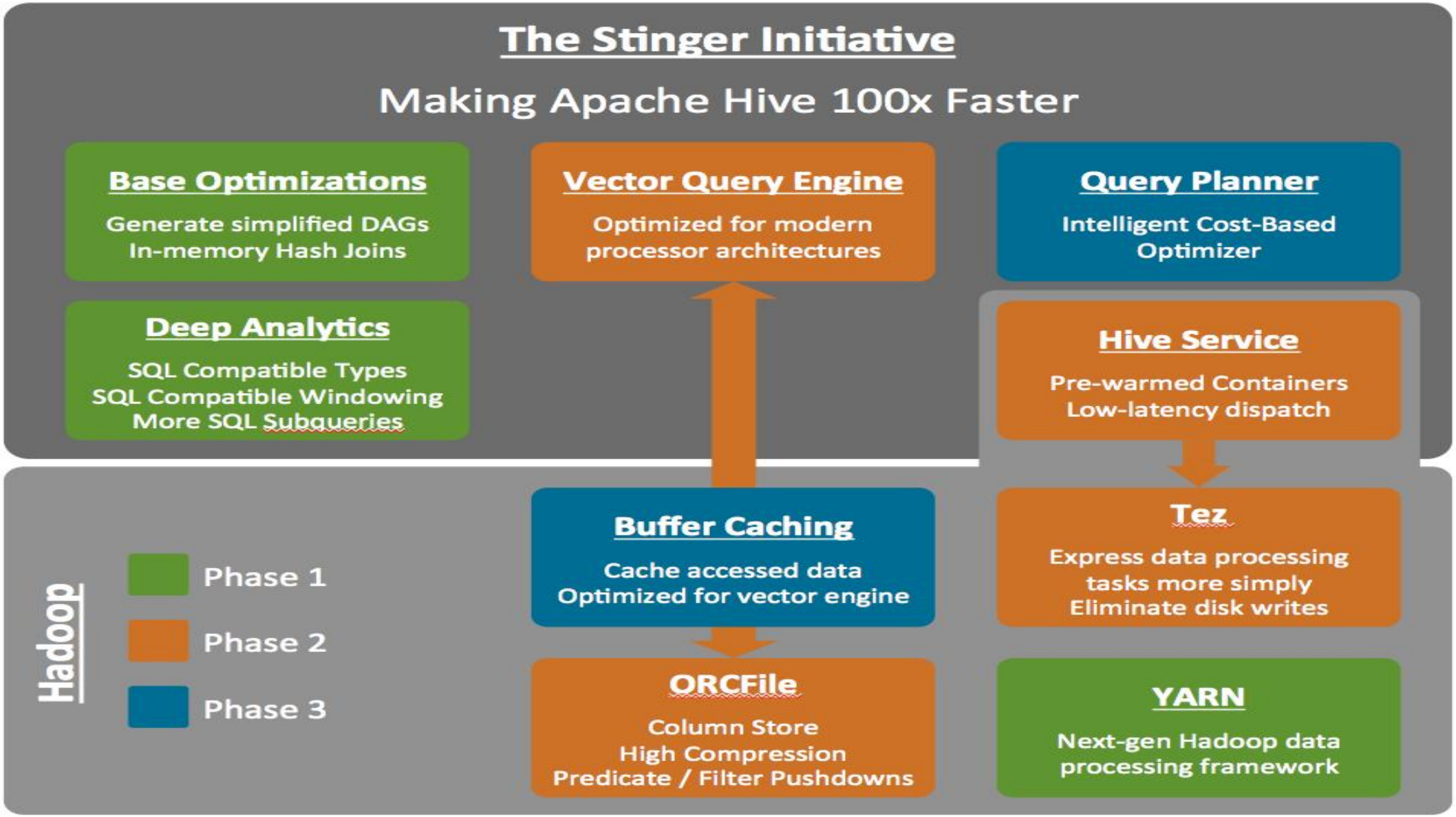
Improve Latency & Throughput

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

Extend Deep Analytical Ability

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

Stinger Initiative at a glance

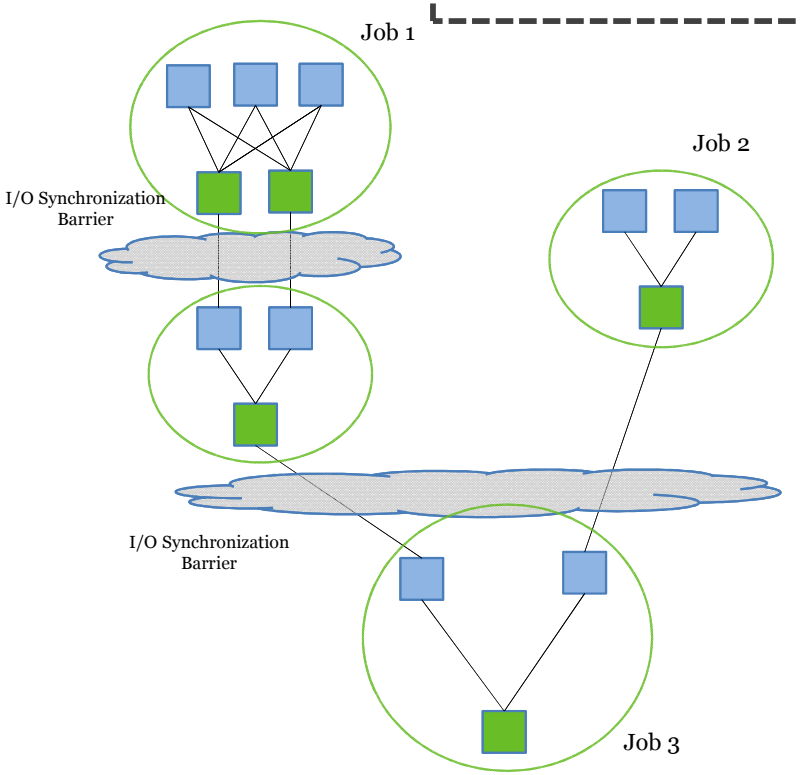


Tez: The Execution Engine

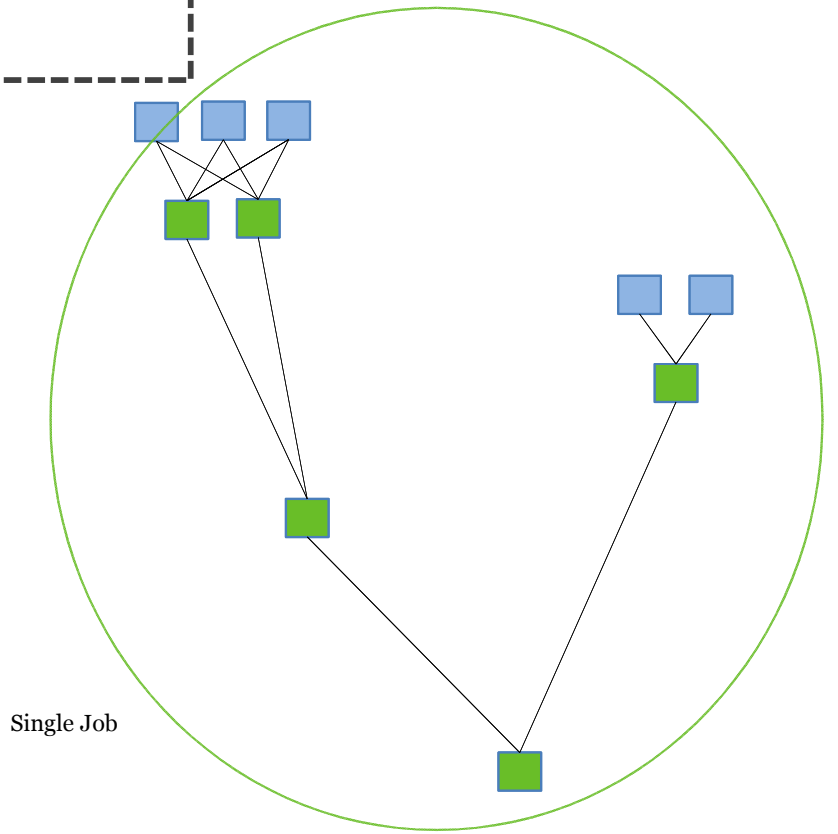
- Low level data-processing **execution engine**
- Use it for the base of MapReduce, Hive, Pig, etc.
- Enables pipelining of jobs
- Removes task and job launch times
- Hive and Pig jobs no longer need to move to the end of the queue between steps in the pipeline
- Does not write intermediate output to 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	
Parse Query	0.5s
Create Plan	0.5s
Launch Map-Reduce	20s
Process Map-Reduce	10s
Total	31s

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

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

*** 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
ORC/Hive 11/ Partitioned/ Bucketed	**1**	**687 MB**	**27**	**79.62**

*** 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

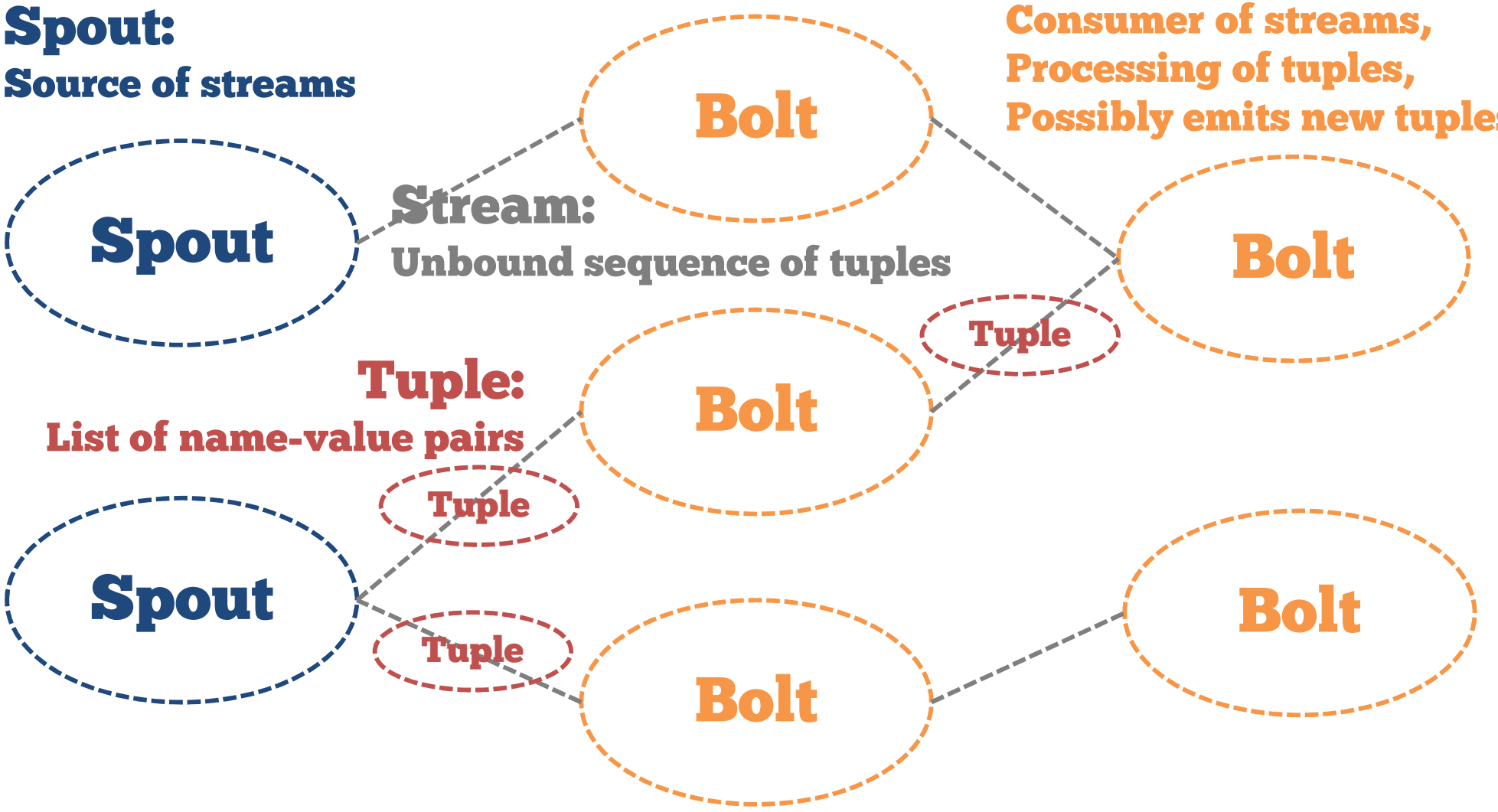
Twitter Storm

- **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

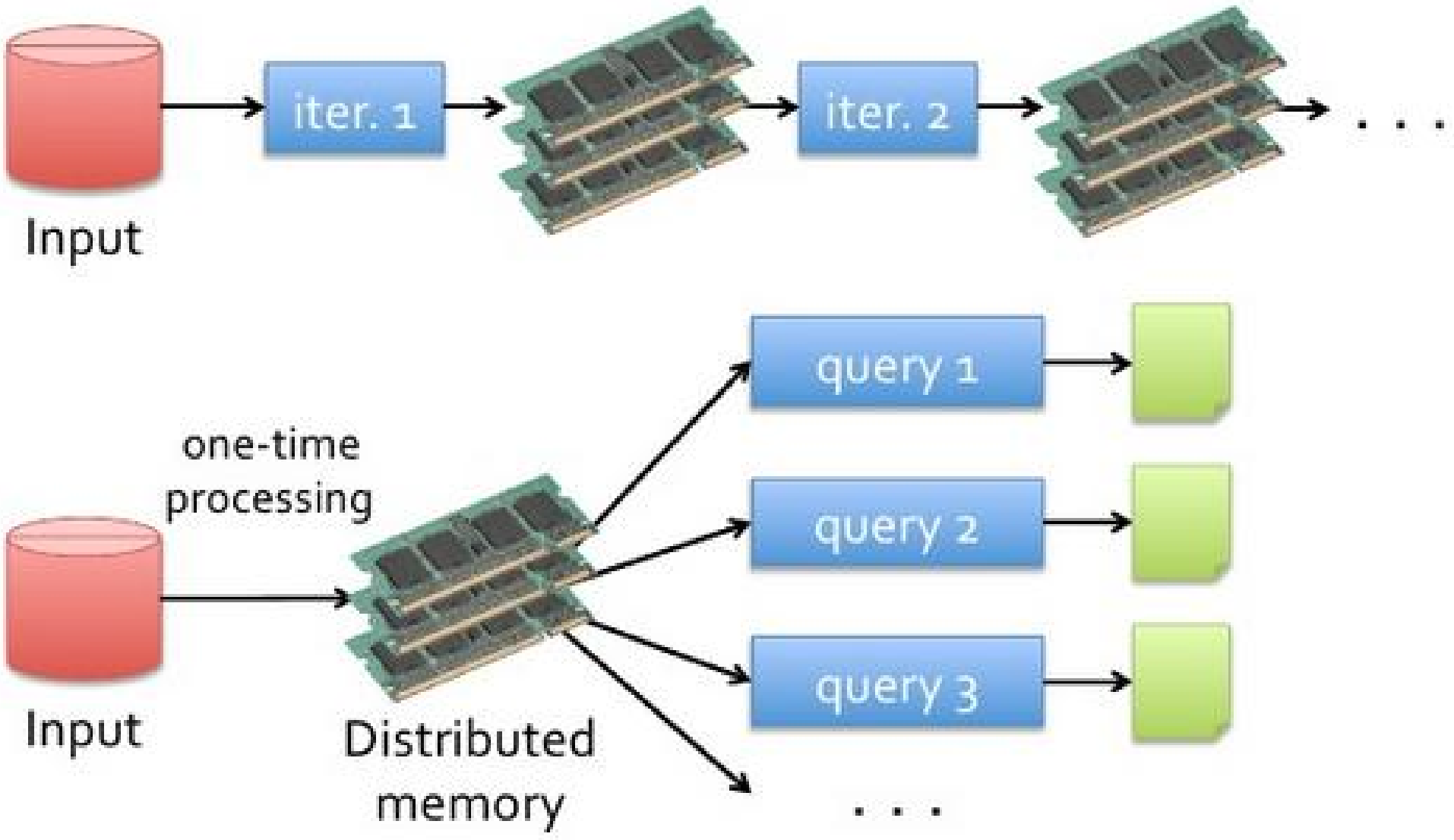
Hadoop 2.0 Applications

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

Spark

- **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

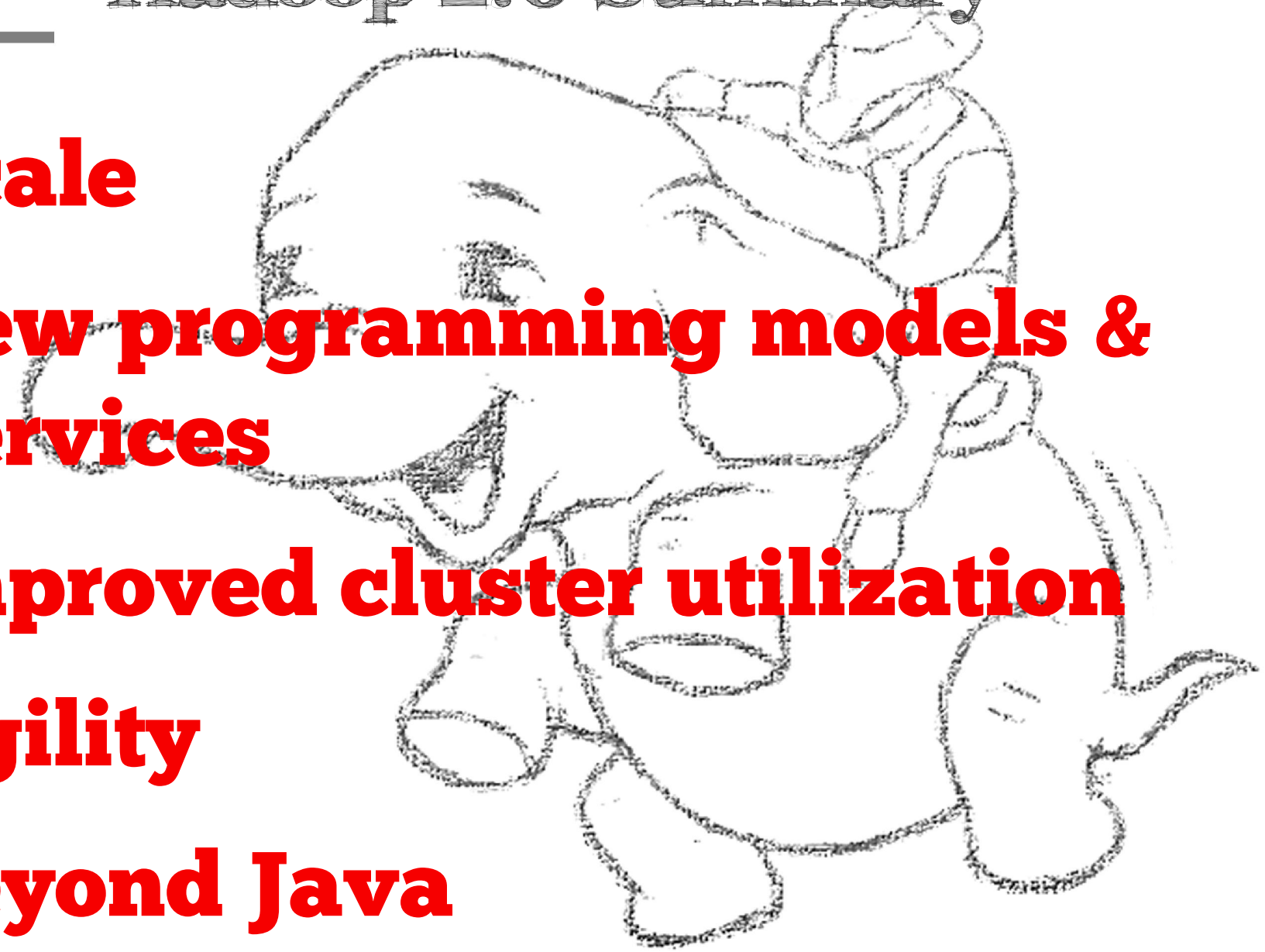
- MapReduce 2.0
- HOYA - HBase on YARN
- Storm, Spark, Apache S4
- Hamster (MPI on Hadoop)
- **Apache Giraph**
- Apache Hama
- Distributed Shell
- Tez

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**

IT-Stammtisch DARMSTADT 07.11.13

Hadoop 2.0 Summary



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

codecentric

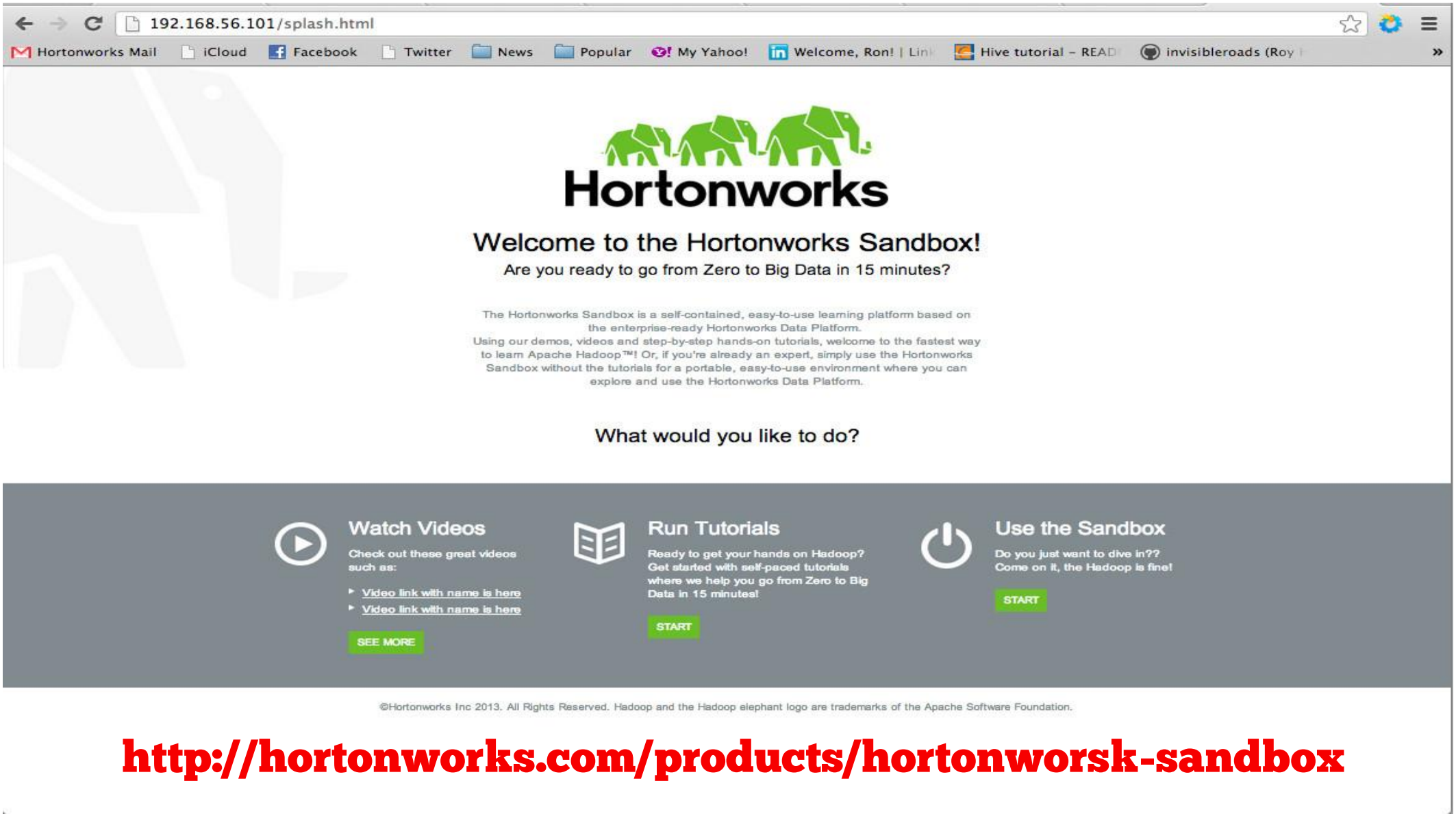
Getting started...

One more thing...

IT-Stammtisch DARMSTADT

07.11.13

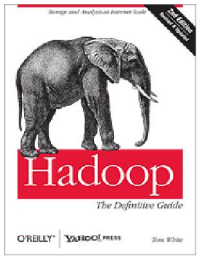
Hortonworks Sandbox



http://hortonworks.com/products/hortonworsk-sandbox

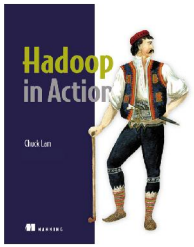
codecentric

Books about Hadoop



Hadoop - The Definite Guide, Tom White,

3rd ed., O'Reilly, 2012.

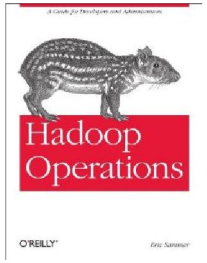


Hadoop in Action, Chuck Lam,

Manning, 2011

Programming Pig, Alan Gates

O'Reilly, 2011



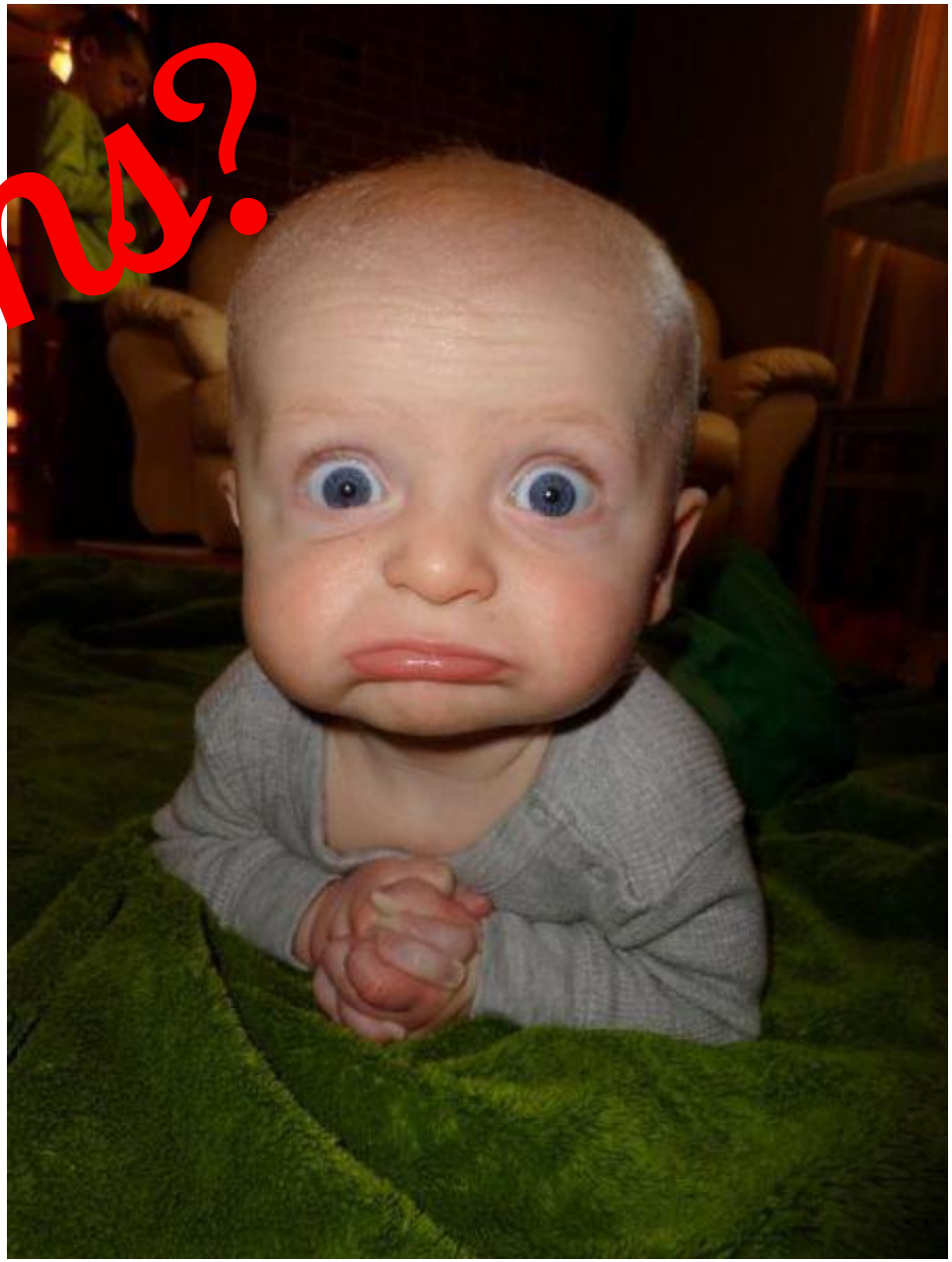
Hadoop Operations, Eric Sammer,

O'Reilly, 2012

IT-Stammtisch DARMSTADT 07.11.13

The end... or the beginning?

Questions?



codecentric