Hadoop

Alexey Zinovyev, Java/BigData Trainer in EPAM

With IT since 2007

With Java since 2009

With Hadoop since 2012

With Spark since 2014

With EPAM since 2015

About

Contacts

E-mail : Alexey_Zinovyev@epam.com

Twitter : @zaleslaw @BigDataRussia

vk.com/big_data_russia **Big Data Russia**

+ Telegram @bigdatarussia

vk.com/java_jvm **Java & JVM langs**

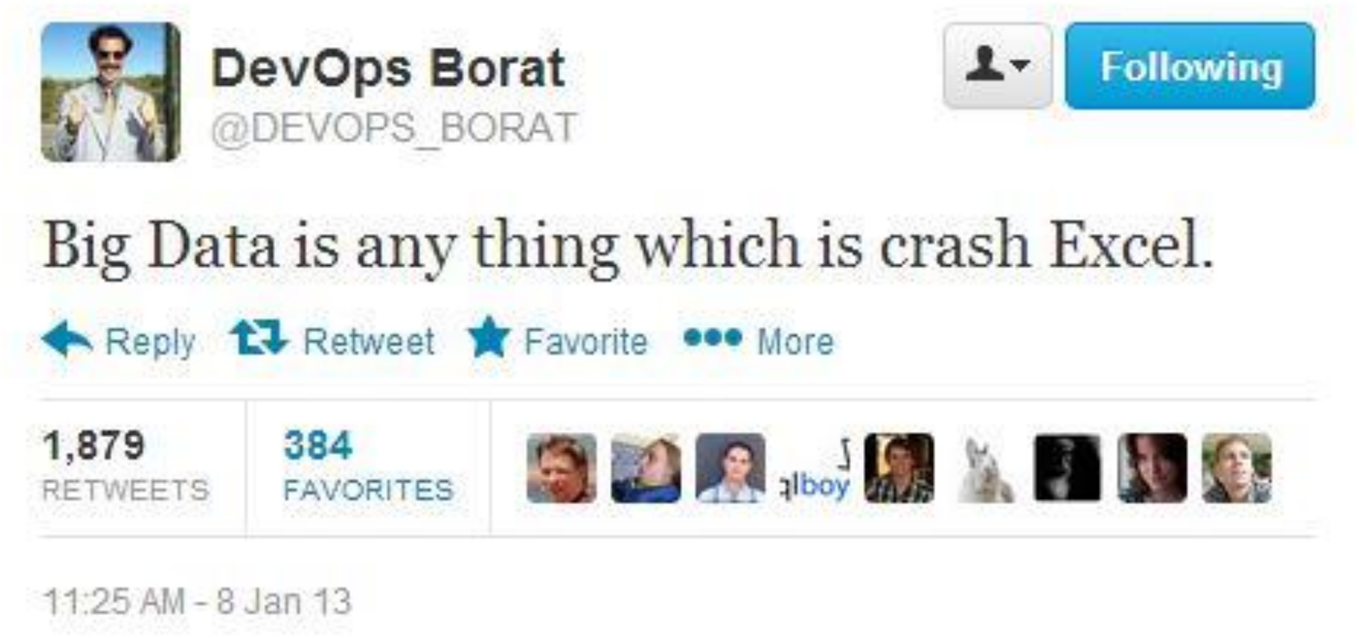
+ Telegram @javajvmlangs

Main parts

- What is BIG DATA?
- Intro in Hadoop
- HDFS & YARN
- MapReduce Java API
- JOINs techniques*
- JVM Settings*
- File formats*

WHAT IS BIG DATA?

Joke about Excel



DevOps Borat
@DEVOPS_BORAT

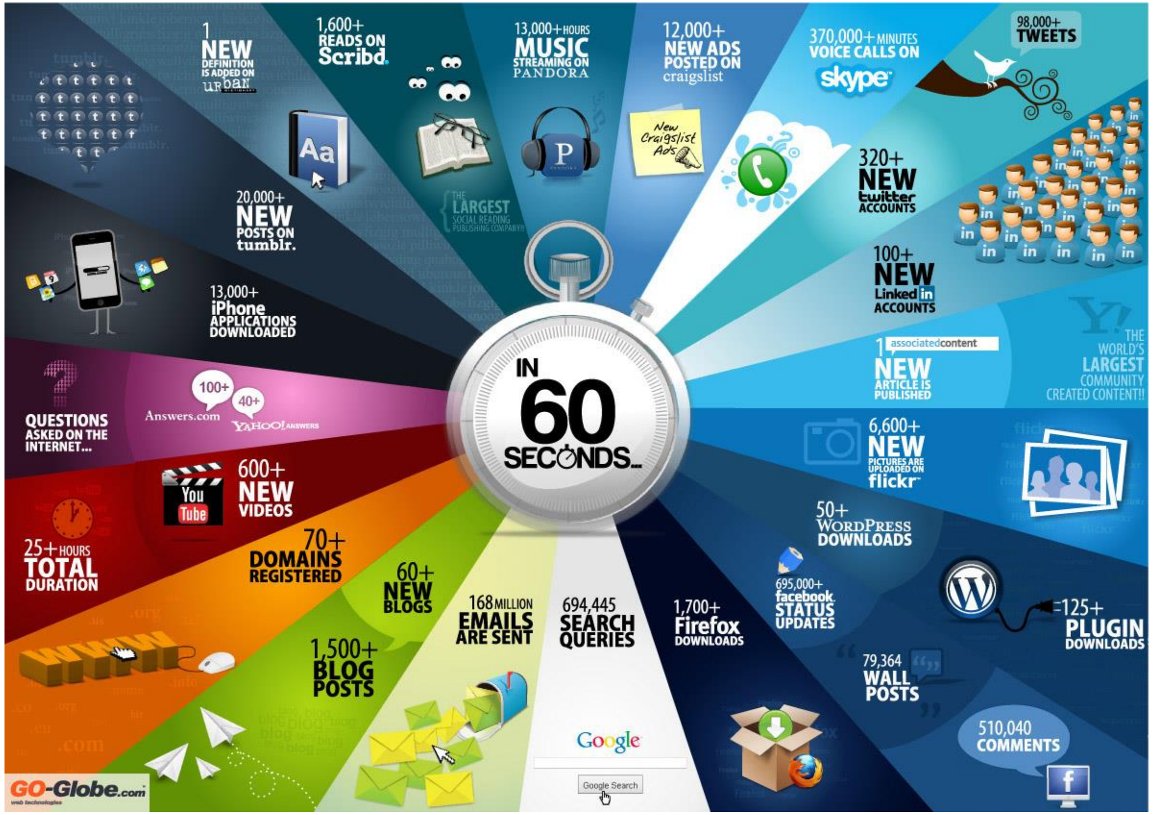
Big Data is any thing which is crash Excel.

1,879 RETWEETS

384 FAVORITES

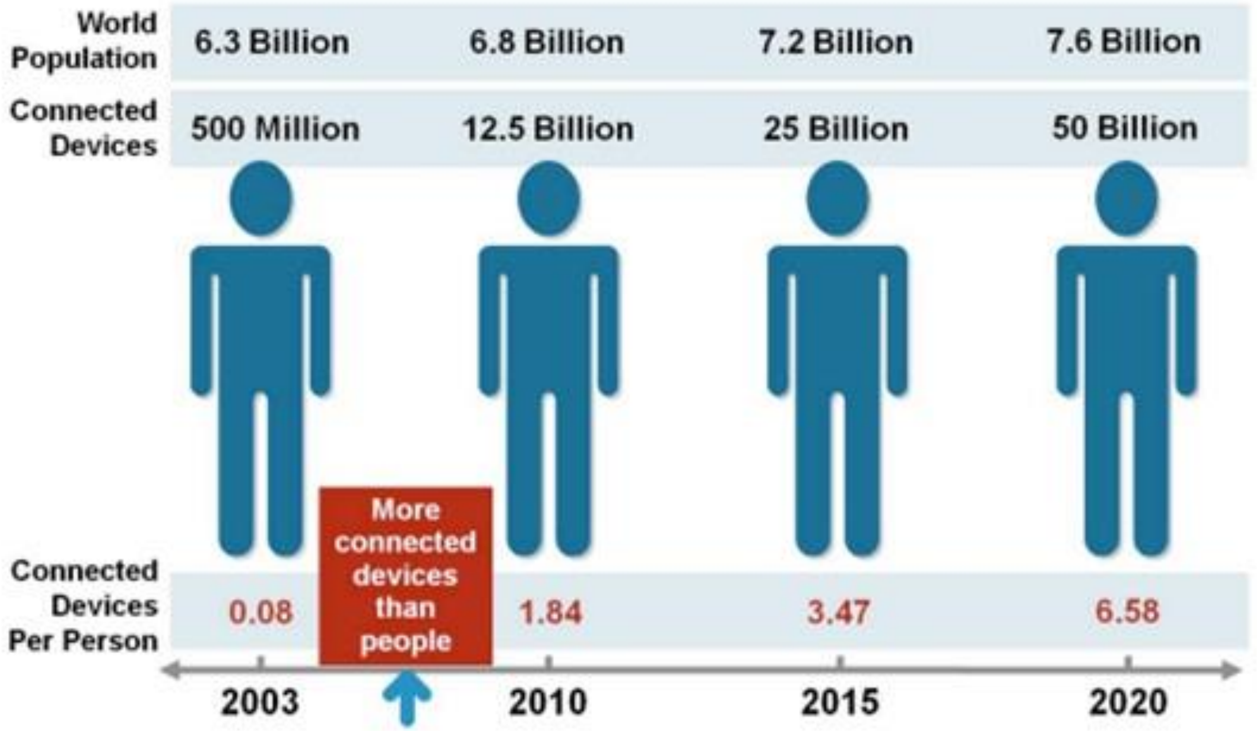
11:25 AM - 8 Jan 13

Every 60 seconds…

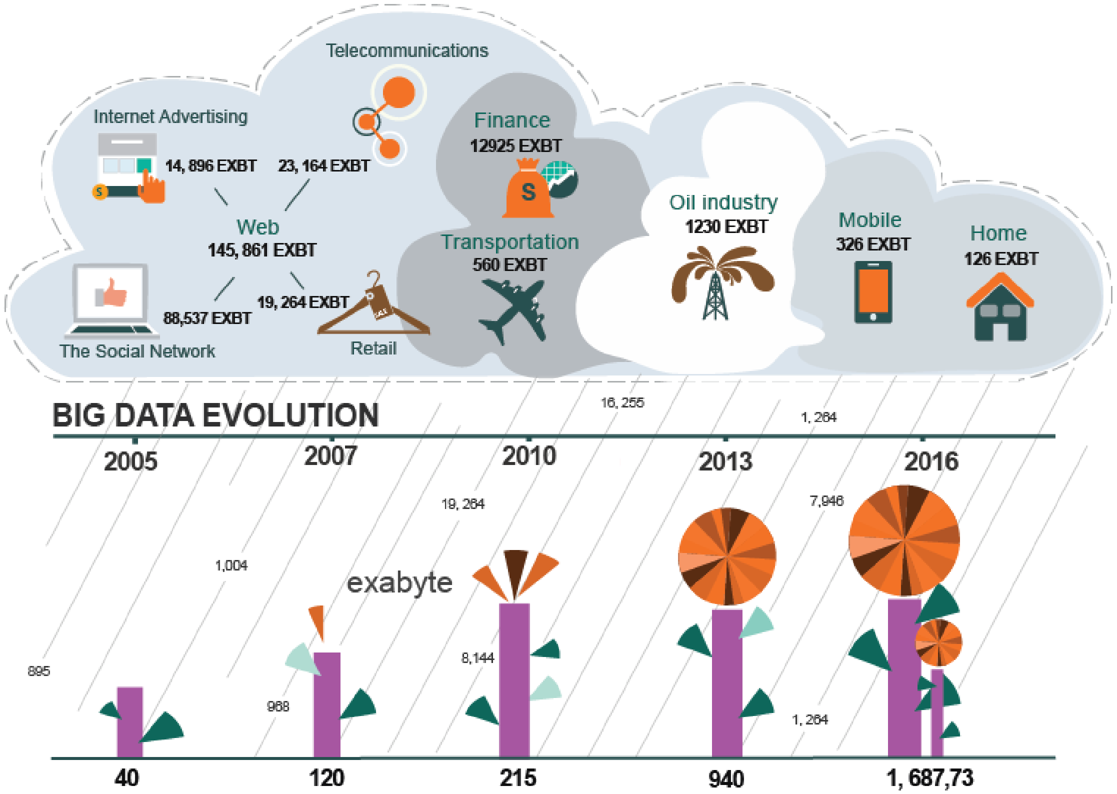


IN 60 SECONDS…
1 NEW DEFINITION IS ADDED ON urban dictionary
1,600+ READS ON Scribd.
THE LARGEST SOCIAL READING PUBLISHING COMPANY!!
13,000+ HOURS MUSIC STREAMING ON PANDORA
12,000+ NEW ADS POSTED ON craigslist
New Craigslist Ads
370,000+ MINUTES VOICE CALLS ON skype
98,000+ TWEETS
320+ NEW twitter ACCOUNTS
100+ NEW LinkedIn ACCOUNTS
1 associatedcontent NEW ARTICLE IS PUBLISHED
THE WORLD'S LARGEST COMMUNITY CREATED CONTENT!!
6,600+ NEW PICTURES ARE UPLOADED ON flickr
50+ WORDPRESS DOWNLOADS
125+ PLUGIN DOWNLOADS
695,000+ facebook STATUS UPDATES
79,364 WALL POSTS
510,040 COMMENTS
1,700+ Firefox DOWNLOADS
694,445 SEARCH QUERIES
Google
Google Search
168 MILLION EMAILS ARE SENT
60+ NEW BLOGS
1,500+ BLOG POSTS
70+ DOMAINS REGISTERED
600+ NEW VIDEOS
YouTube
25+ HOURS TOTAL DURATION
QUESTIONS ASKED ON THE INTERNET…
100+ Answers.com
40+ YAHOO! ANSWERS
13,000+ iPhone APPLICATIONS DOWNLOADED
20,000+ NEW POSTS ON tumblr.
GO-Globe.com web technologies

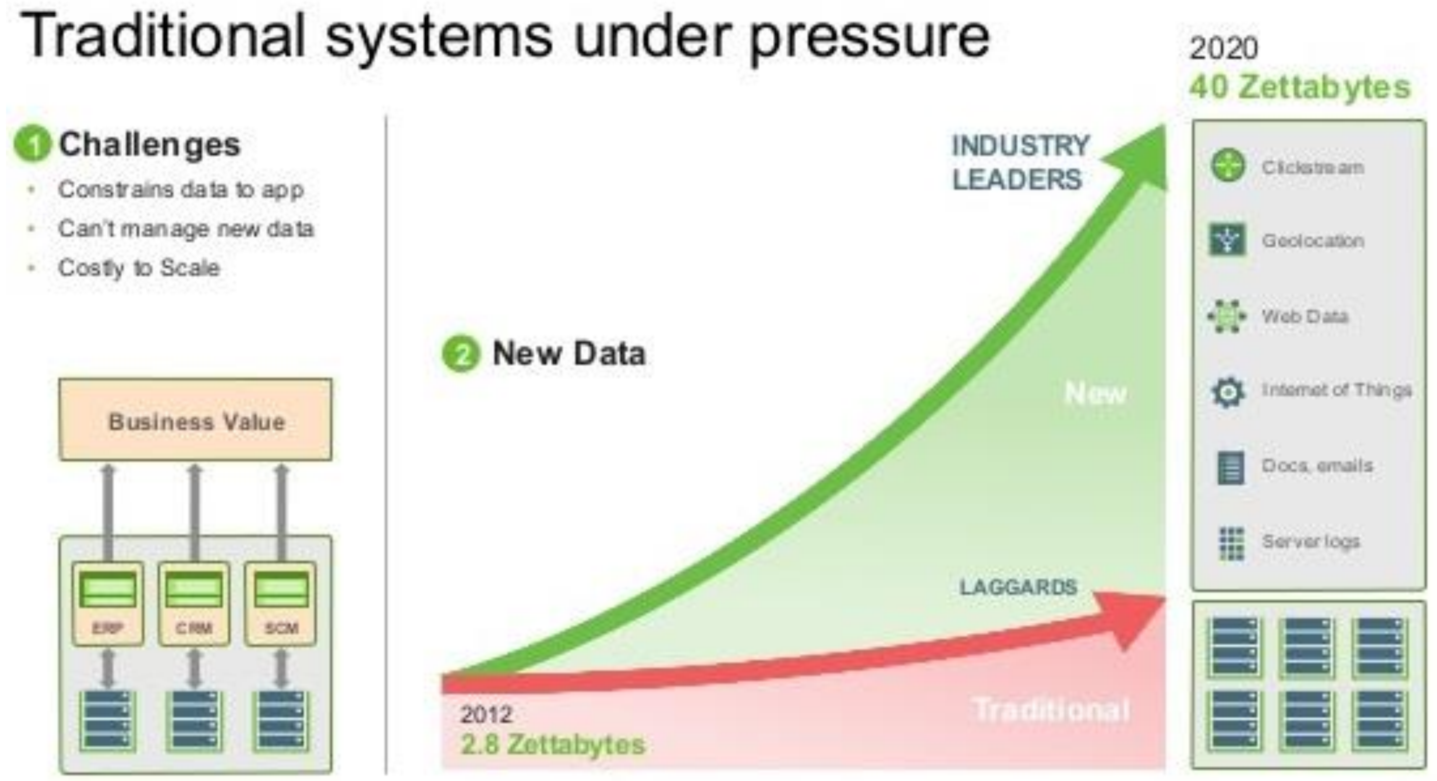
From Mobile Devices

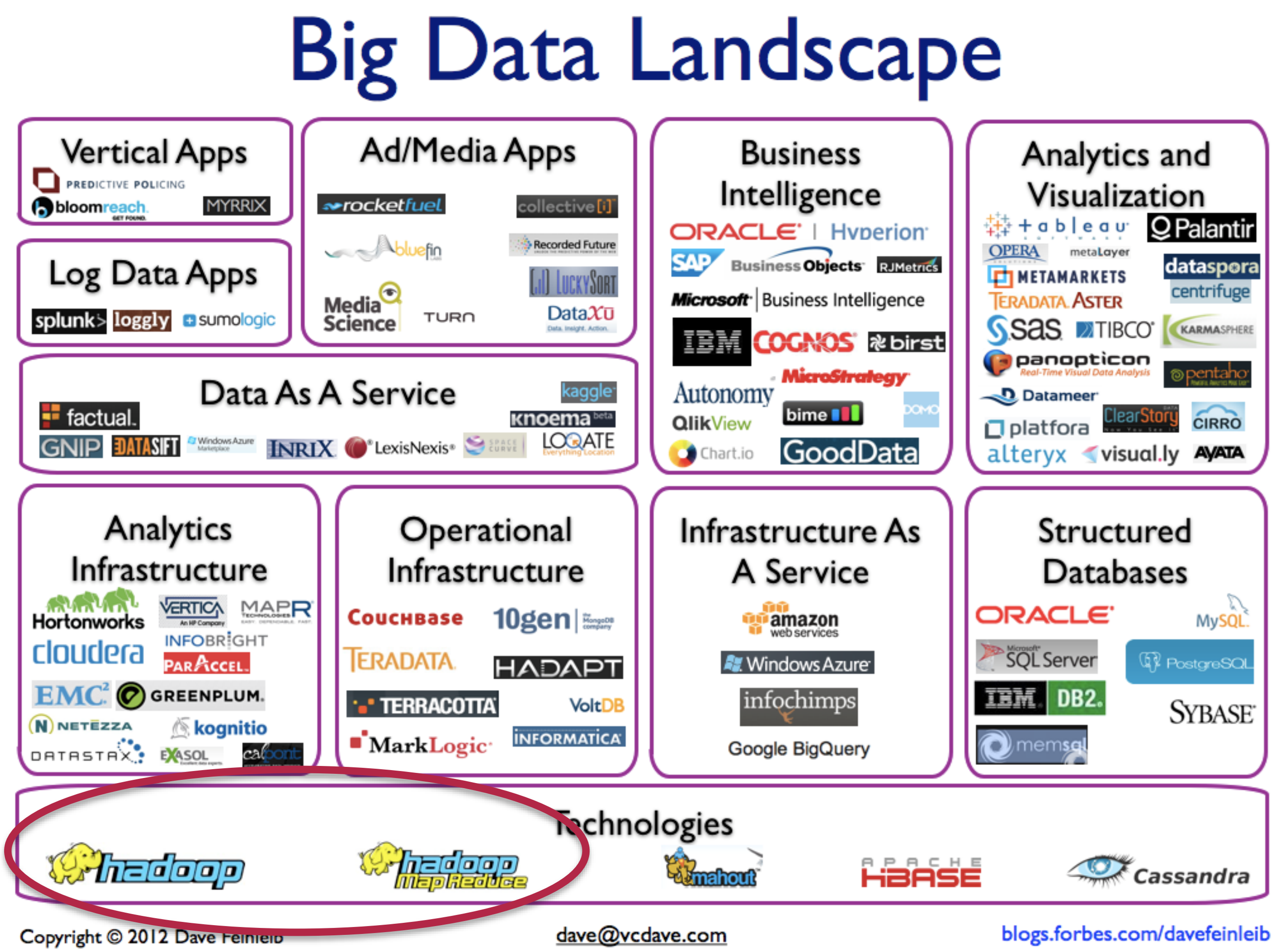


From Industry



We started to keep and handle ~~stupid~~ new things!





Big Data Landscape

Vertical Apps
PREDICTIVE POLICING
bloomreach
MYRRIX

Log Data Apps
splunk>
loggly
sumologic

Ad/Media Apps
rocketfuel
collective[i]
bluefin
Recorded Future
LuckySort
Media Science
TURN
DataXu

Data As A Service
kaggle
factual.
knoema beta
GNIP
DATASIFT
Windows Azure Marketplace
INRIX
LexisNexis
SPACE CURVE
LOQATE

Business Intelligence
ORACLE | Hyperion
SAP Business Objects
RJMetrics
Microsoft Business Intelligence
IBM
COGNOS
birst
MicroStrategy
Autonomy
QlikView
bime
DOMO
Chart.io
GoodData

Analytics and Visualization
tableau
Palantir
OPERA
metaLayer
dataspora
METAMARKETS
centrifuge
TERADATA ASTER
sas
TIBCO
KARMASPHERE
panopticon
pentaho
Datameer
platfora
ClearStory
CIRRO
alteryx
visual.ly
AYATA

Analytics Infrastructure
Hortonworks
VERTICA
MAPR
cloudera
INFOBRIGHT
PARACCEL
EMC²
GREENPLUM
NETEZZA
kognitio
DATASTAX
EXASOL
calpont

Operational Infrastructure
COUCHBASE
10gen | the MongoDB company
TERADATA
HADAPT
TERRACOTTA
VoltDB
MarkLogic
INFORMATICA

Infrastructure As A Service
amazon web services
Windows Azure
infochimps
Google BigQuery

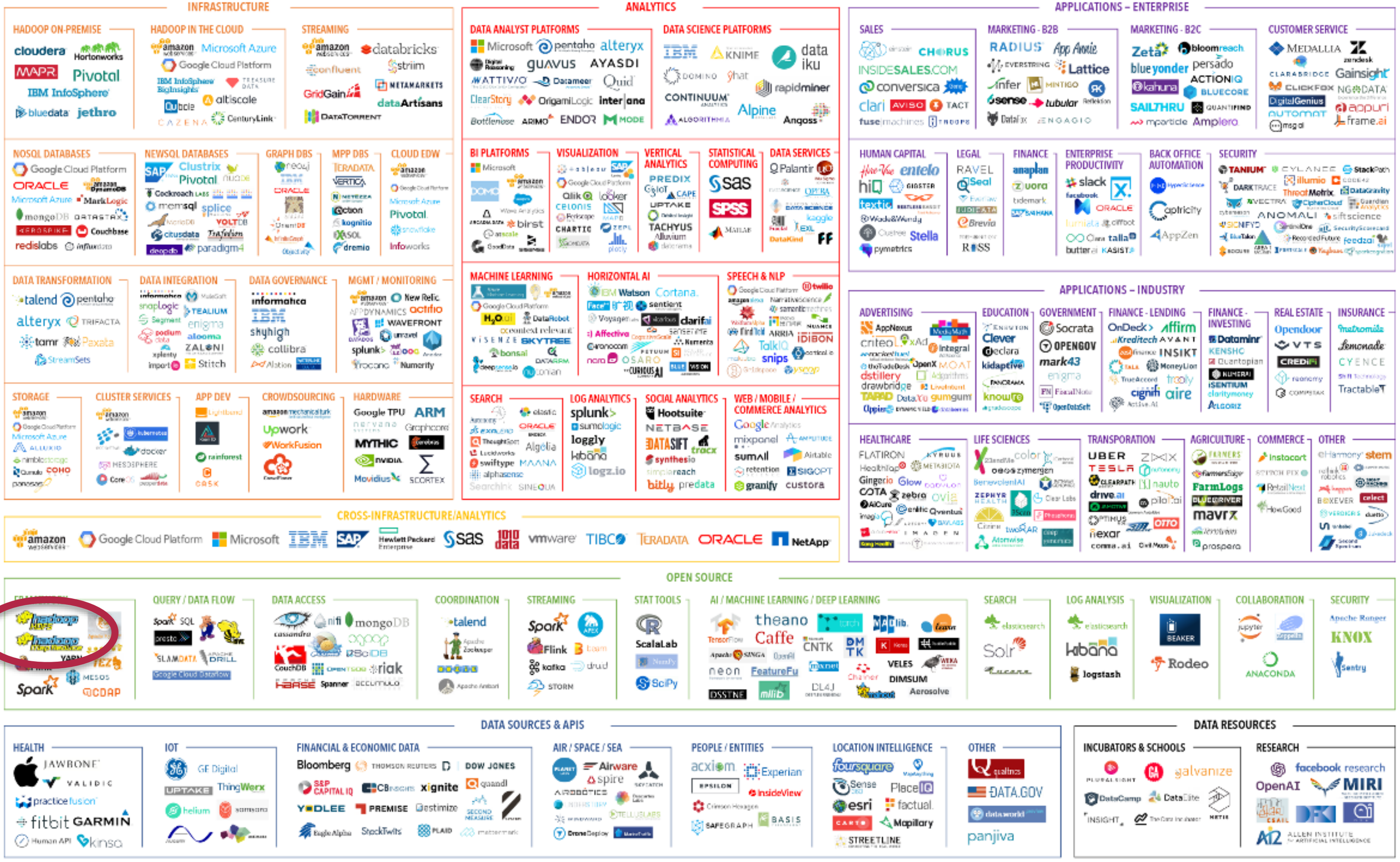
Structured Databases
ORACLE
MySQL
Microsoft SQL Server
PostgreSQL
IBM DB2
SYBASE
memsql

Technologies
hadoop
hadoop MapReduce
mahout
APACHE HBASE
Cassandra

Copyright © 2012 Dave Feinleib
dave@vcdave.com
blogs.forbes.com/davefeinleib

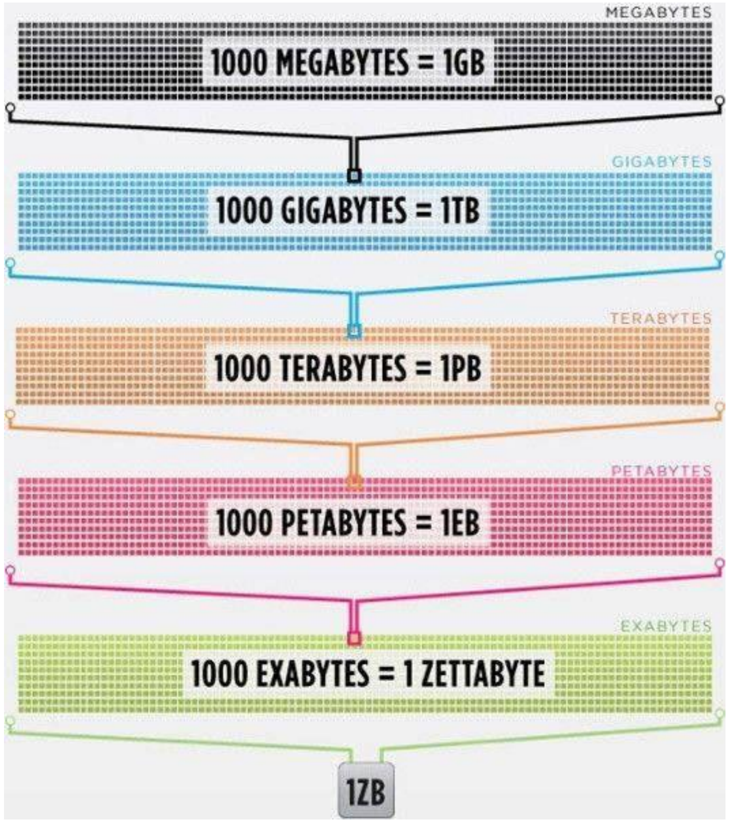
Crazy Zoo 2012

BIG DATA LANDSCAPE 2017



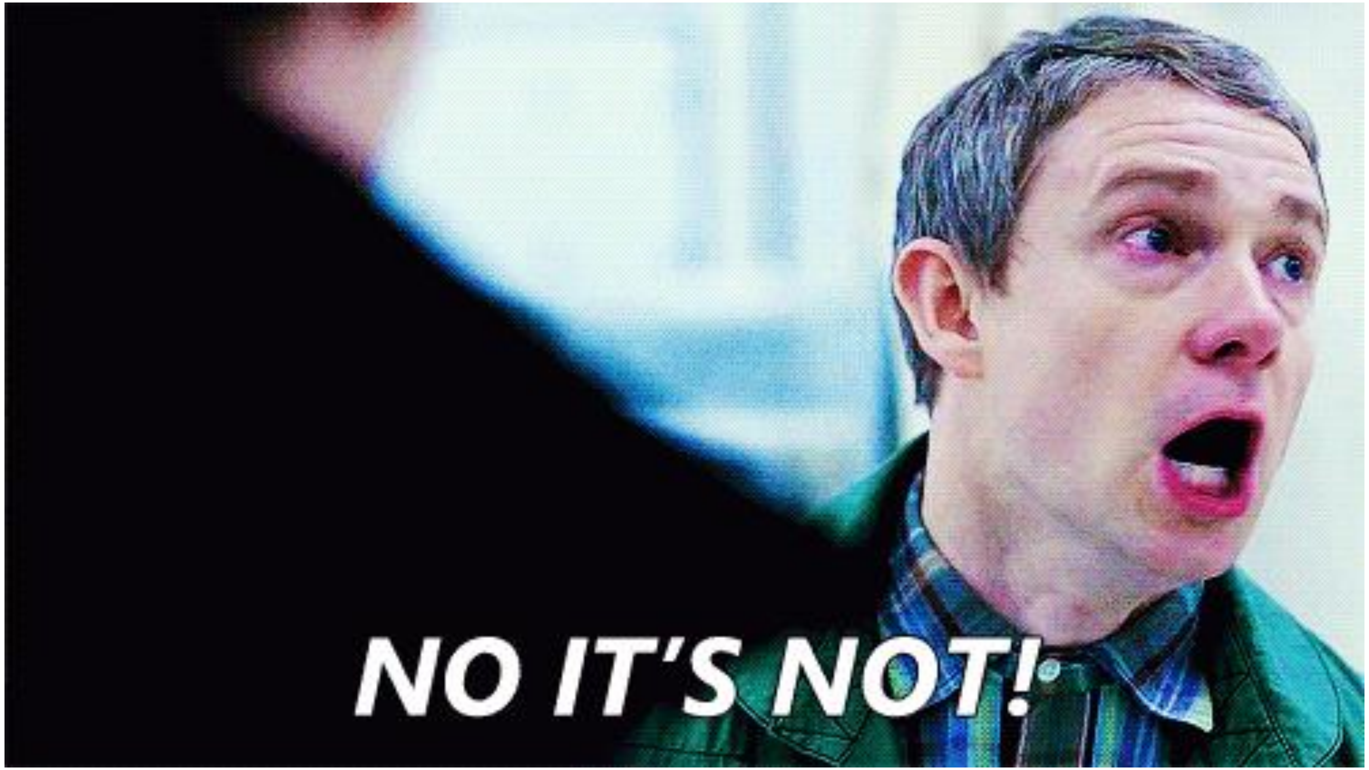
10^6 rows in MySQL

GB->TB->PB->?


MEGABYTES
1000 MEGABYTES = 1GB
GIGABYTES
1000 GIGABYTES = 1TB
TERABYTES
1000 TERABYTES = 1PB
PETABYTES
1000 PETABYTES = 1EB
EXABYTES
1000 EXABYTES = 1 ZETTABYTE
1ZB

Is BigData about PBs?

Is BigData about PBs?

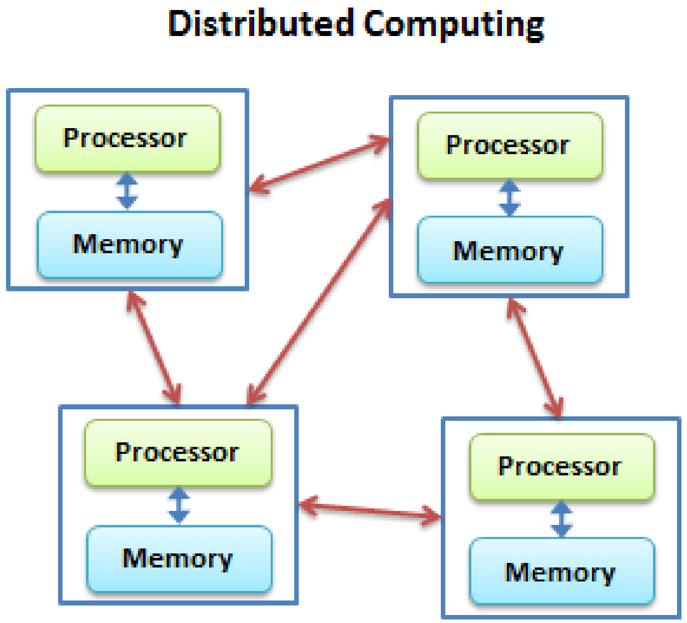


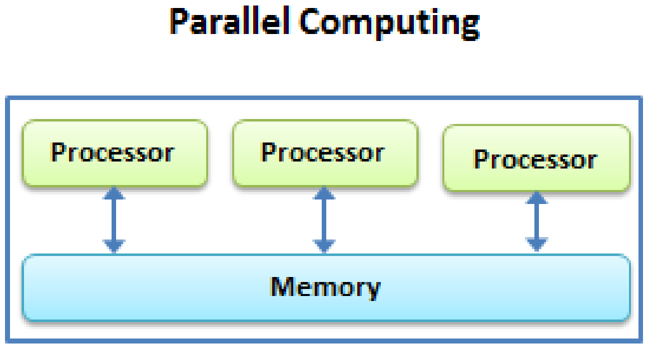
It’s hard to …

- .. store
- .. handle
- .. search in
- .. visualize
- .. send in network

Just do it … in parallel

Parallel Computing vs Distributed Computing





You need to develop

- .. distributed on-disk storage
- .. in-memory storage (or shared memory buffer)
- .. thread pool to run hundreds of threads
- .. synchronize all components
- .. provide API for reusing by other developers

All we love reinvent bicycles, but…

HADOOP

Hadoop

Disks Performance

The main concept

Let’s read data in parallel

“Cheap” cluster

Das Ist Musst surviven!

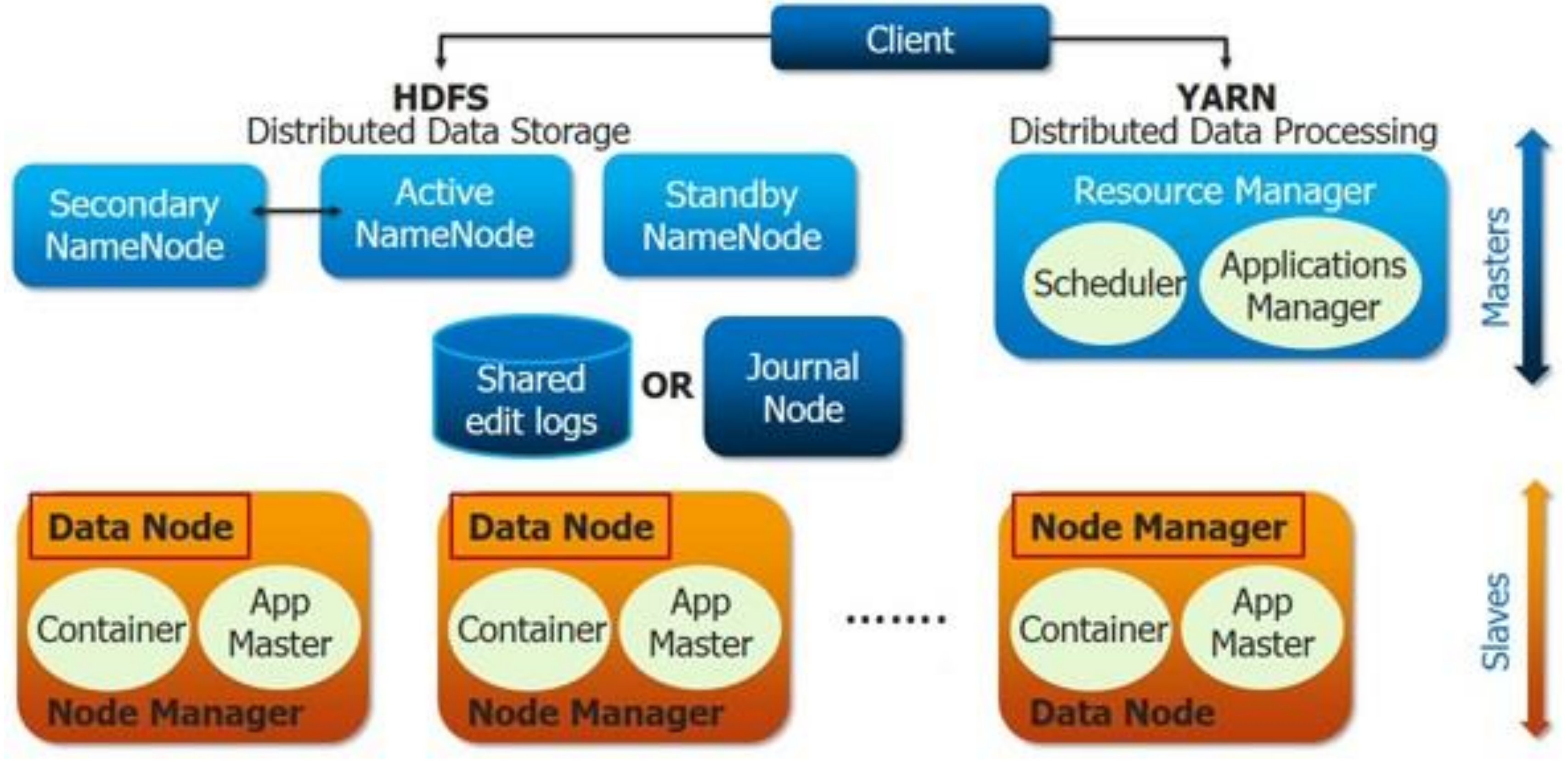
Main components

- Hadoop Commons
- Hadoop Clients
- HDFS
- YARN
- MapReduce

Hadoop frameworks

- Universal (MapReduce, Tez, RDD in Spark)
- Abstract (Pig, Pipeline Spark)
- SQL - like (Hive, Impala, Spark SQL)
- Processing graph (Giraph, GraphX)
- Machine Learning (Mahout, MLib)
- Stream processing (Spark Streaming, Storm)

Hadoop Architecture



Key features

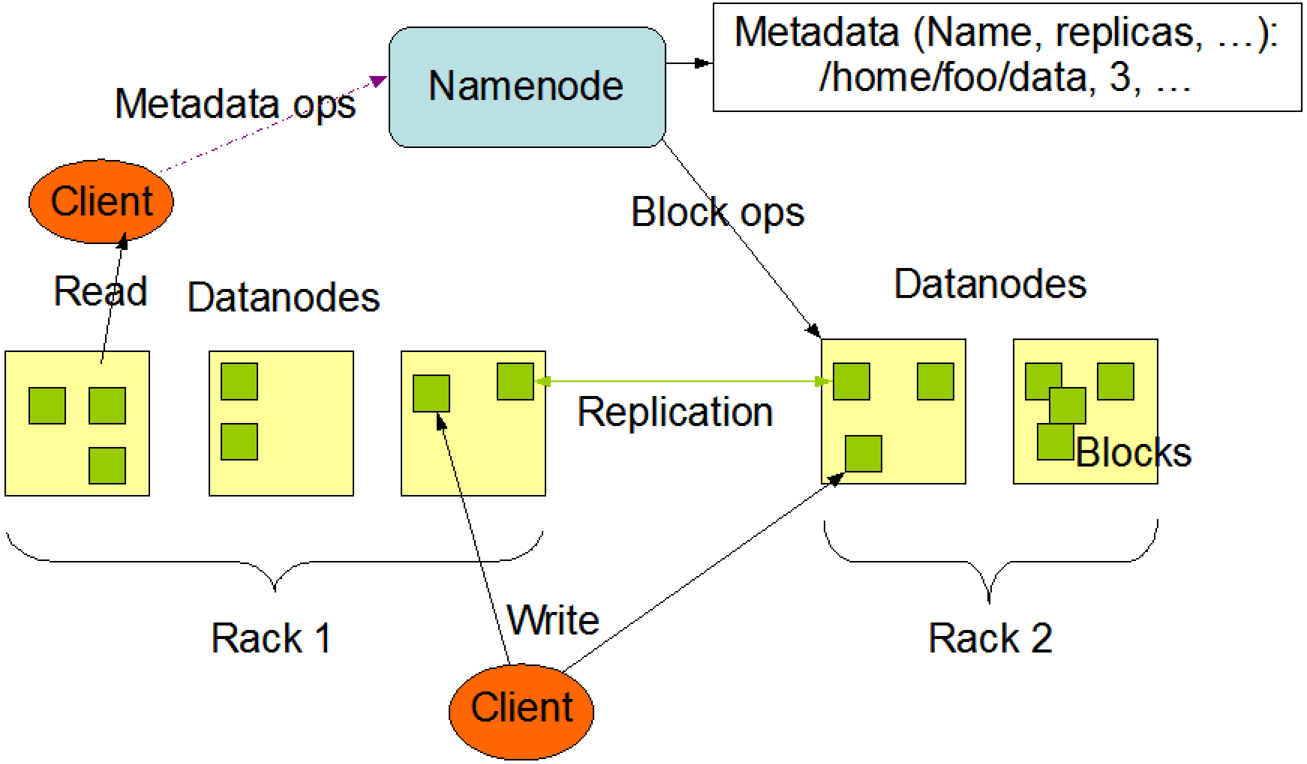
- Automatic parallelization and distribution
- Fault-tolerance
- Data Locality
- Writing the Map and Reduce functions only
- Single-threaded model

HDFS DAEMONS

The main idea

'Time to transfer' > 'Time to seek'

Main idea


Metadata ops
Namenode
Metadata (Name, replicas, …):
/home/foo/data, 3, …
Client
Block ops
Read
Datanodes
Datanodes
Replication
Blocks
Rack 1
Write
Rack 2
Client

HDFS node types

- NameNode
- DataNode
- SecondaryNode
- StandbyNode
- Checkpoint Node
- Backup Node

The main thought about HDFS

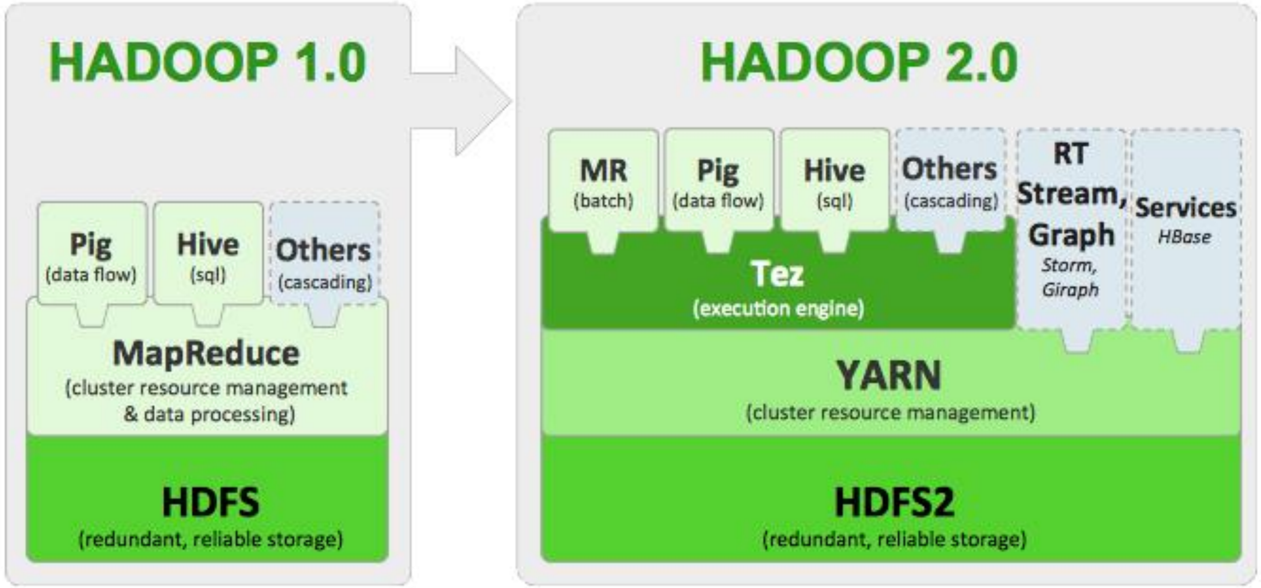
HDFS node is JVM daemon

You can do it with HDFS node

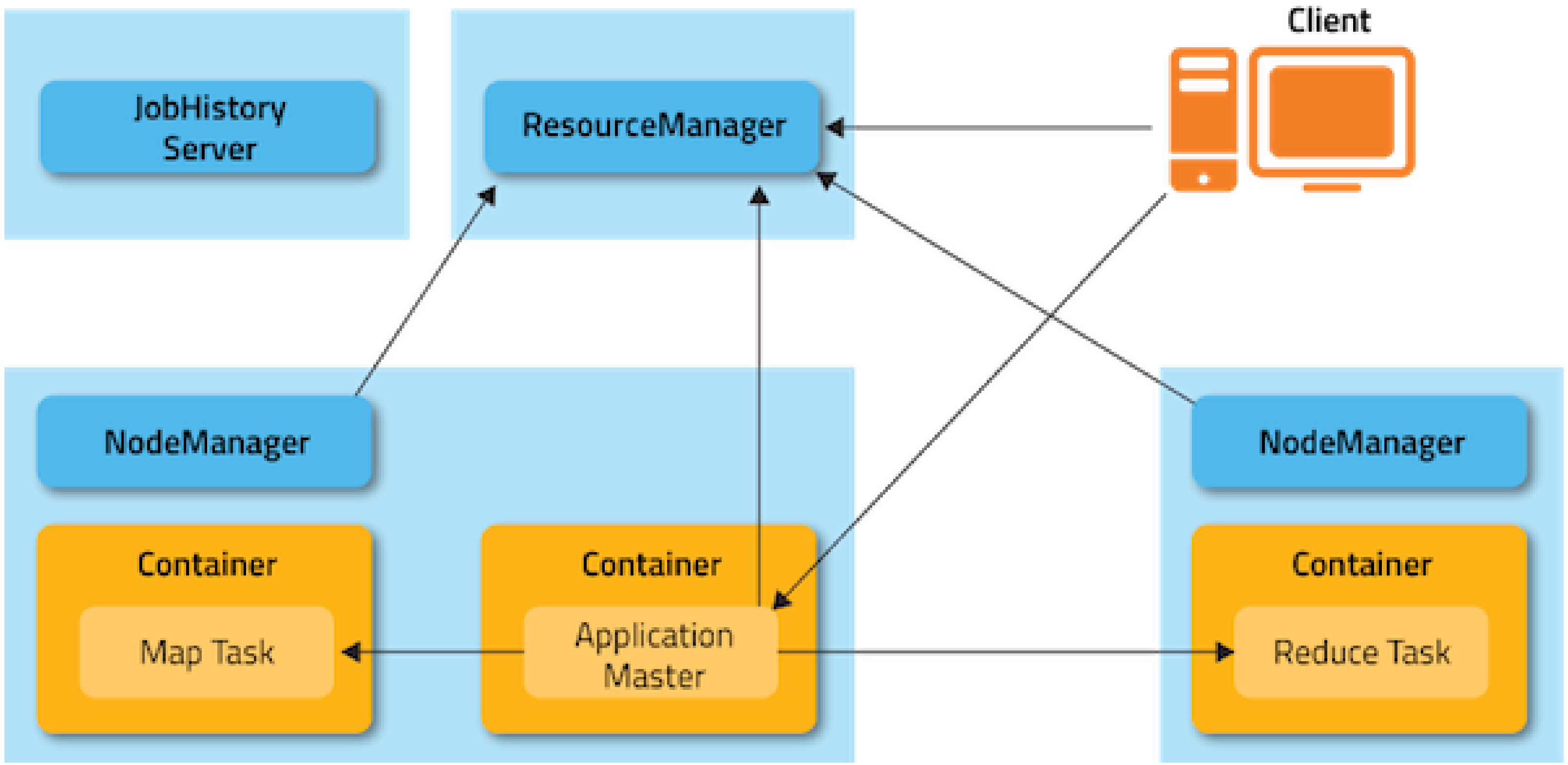
- monitor with JMX
- use jmap, jps and so on..
- configure NameNode Heap Size
- use power of JVM flags

YARN

From Hadoop 1 to Hadoop 2

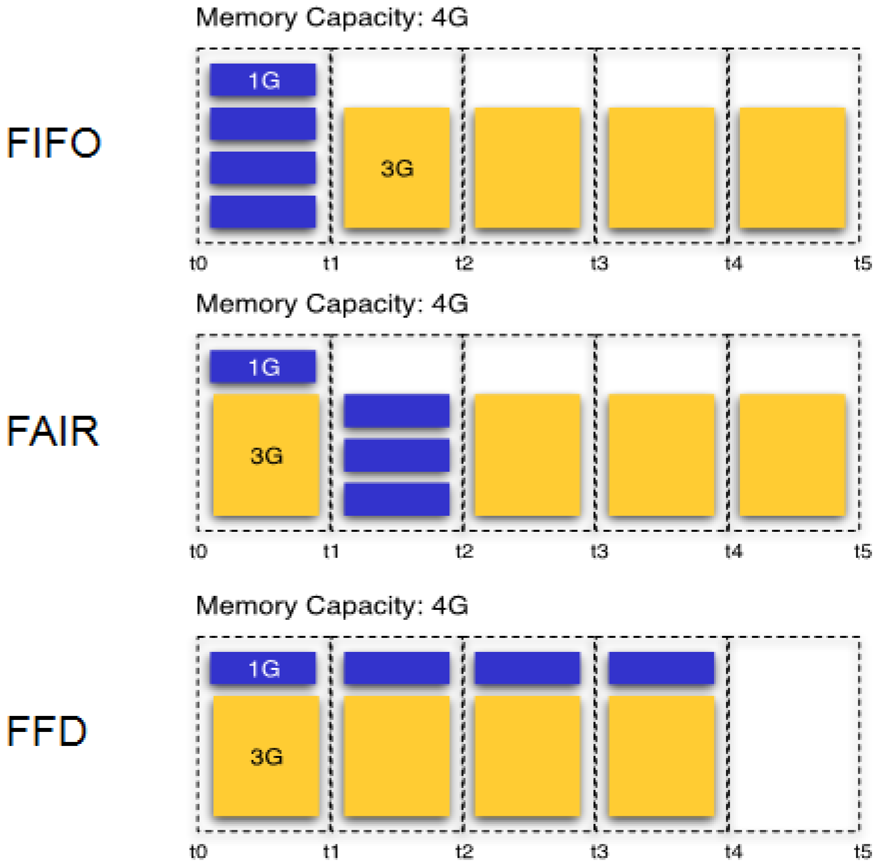


Daemons in YARN



Different scheduling algorithms

Job 1: 4 tasks, 1G demand
Job 2: 4 tasks, 3G demand

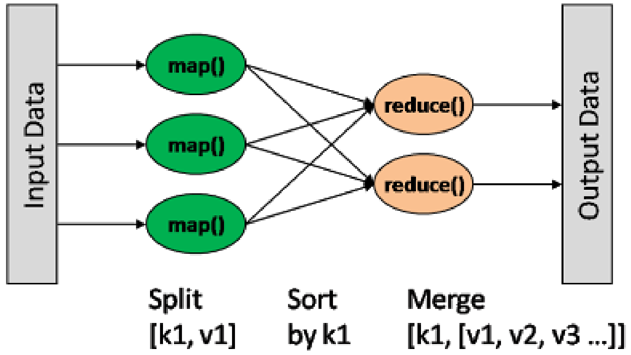


MAPREDUCE THEORY

MapReduce in different languages

Language	Code sample
Java 8	`Integer totalAge = persons` `.stream()` `.map(Person::getAge)` `.reduce( 0, (a, b) -> a + b);`
Scala	`val totalAge = persons` `.map( (p: Person) => p.getAge )` `.reduce( _ + _ )`
Python	`totalAge = reduce(` `(lambda a, b: a + b),` `list( map(lambda p: p.getAge, persons) )` `)`

Think in Key-Value style



map (k1, v1) → list(k2, v2)

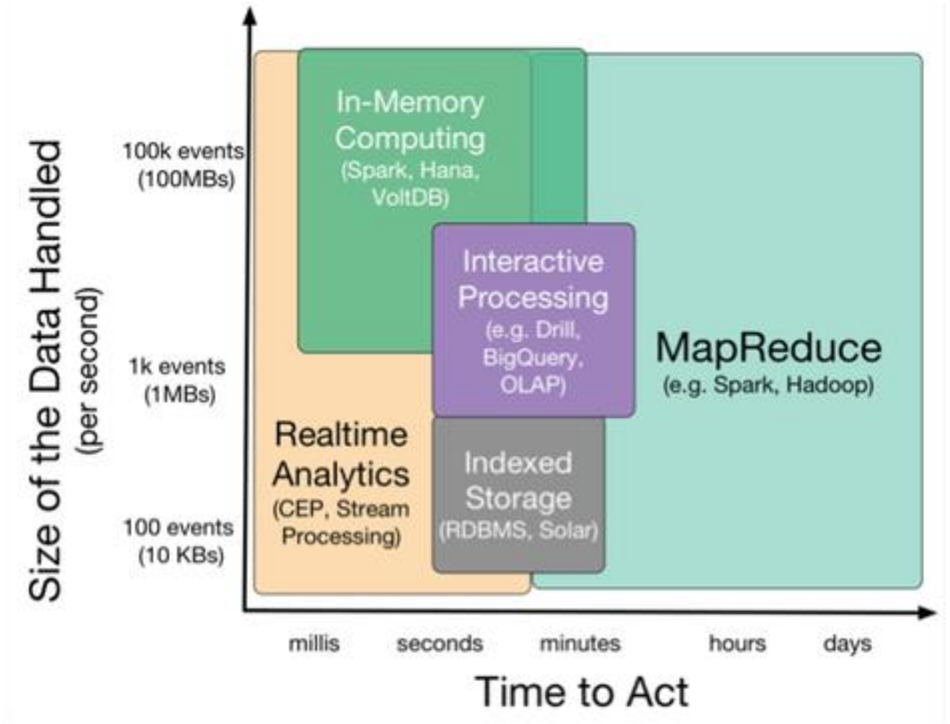
reduce (k2, list(v2*))→ list(k3, v3)

MR Typical Tasks

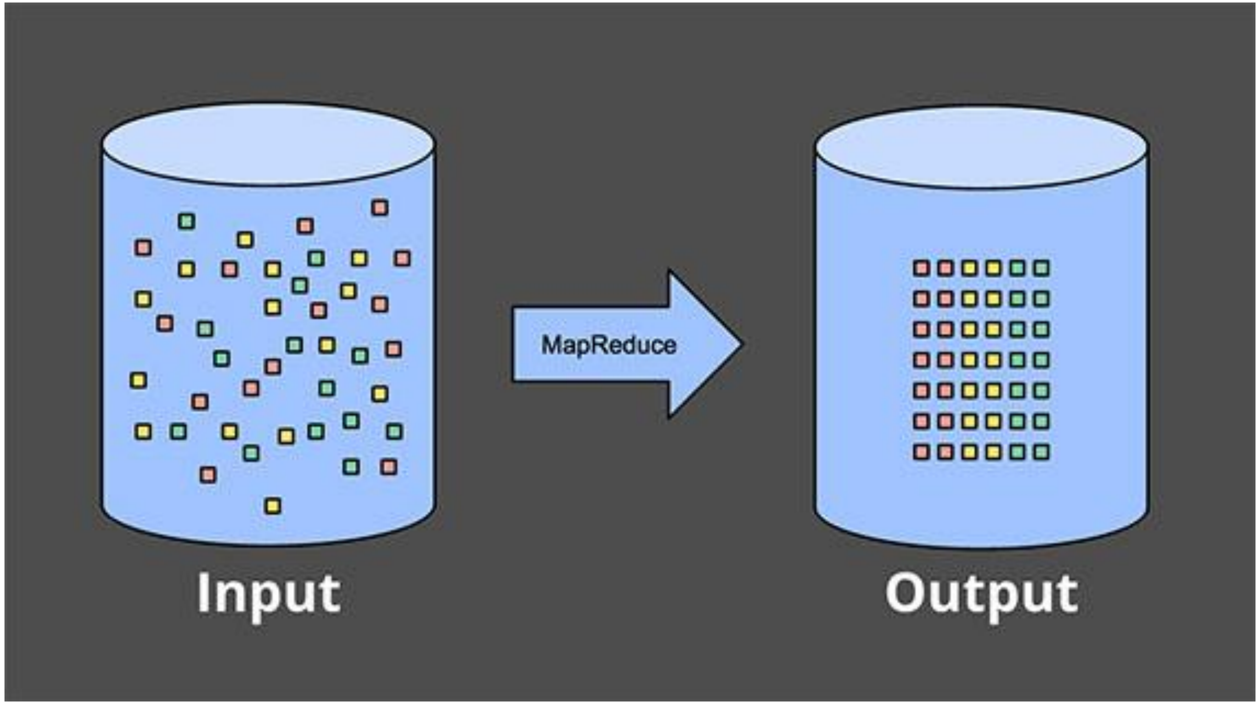
- WordCount
- Log handling
- Filtering
- Reporting Preparation

Why should we use MapReduce?

We try to reduce ‘time to act’ but keep BigData

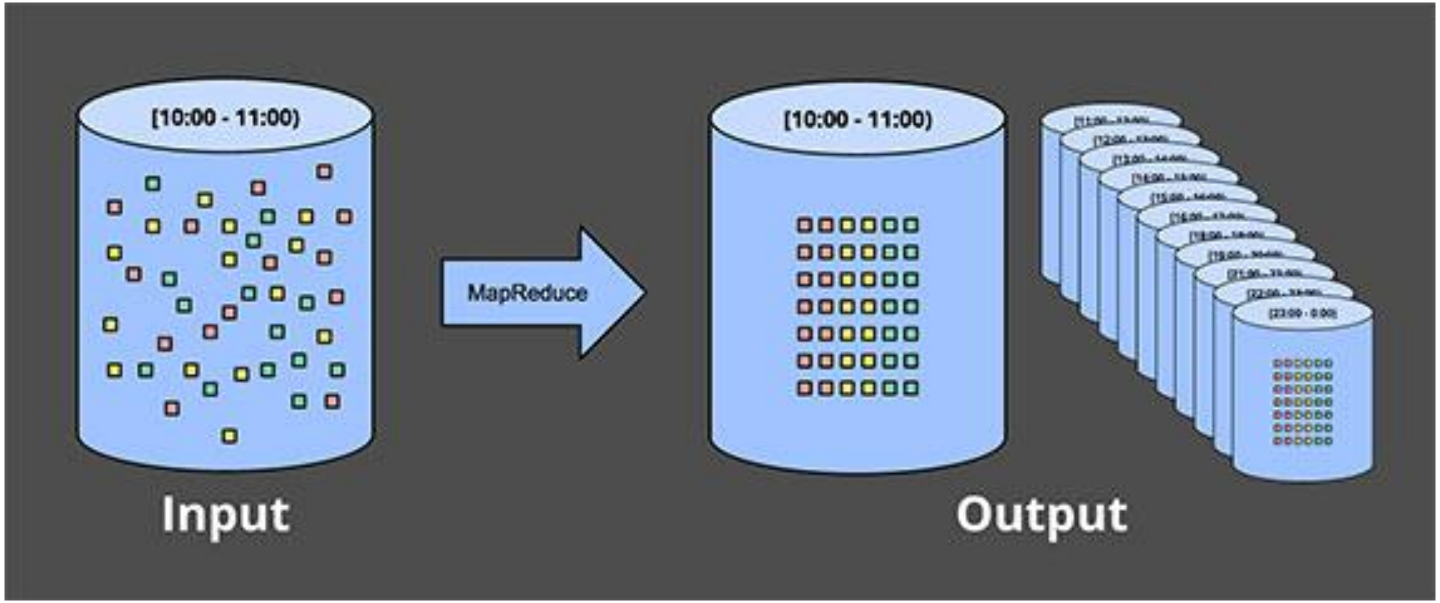

Size of the Data Handled (per second)
100k events (100MBs)
1k events (1MBs)
100 events (10 KBs)
In-Memory Computing (Spark, Hana, VoltDB)
Interactive Processing (e.g. Drill, BigQuery, OLAP)
MapReduce (e.g. Spark, Hadoop)
Realtime Analytics (CEP, Stream Processing)
Indexed Storage (RDBMS, Solar)
millis
seconds
minutes
hours
days
Time to Act

Classic Batch

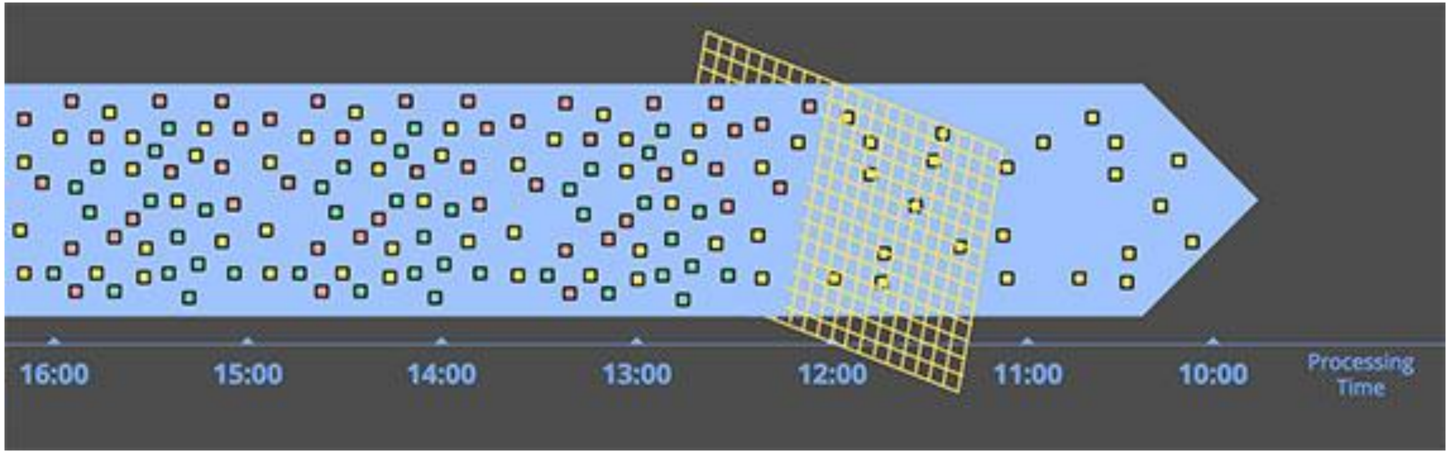


Do you like batches?

Fixed Windows



Filter all elements

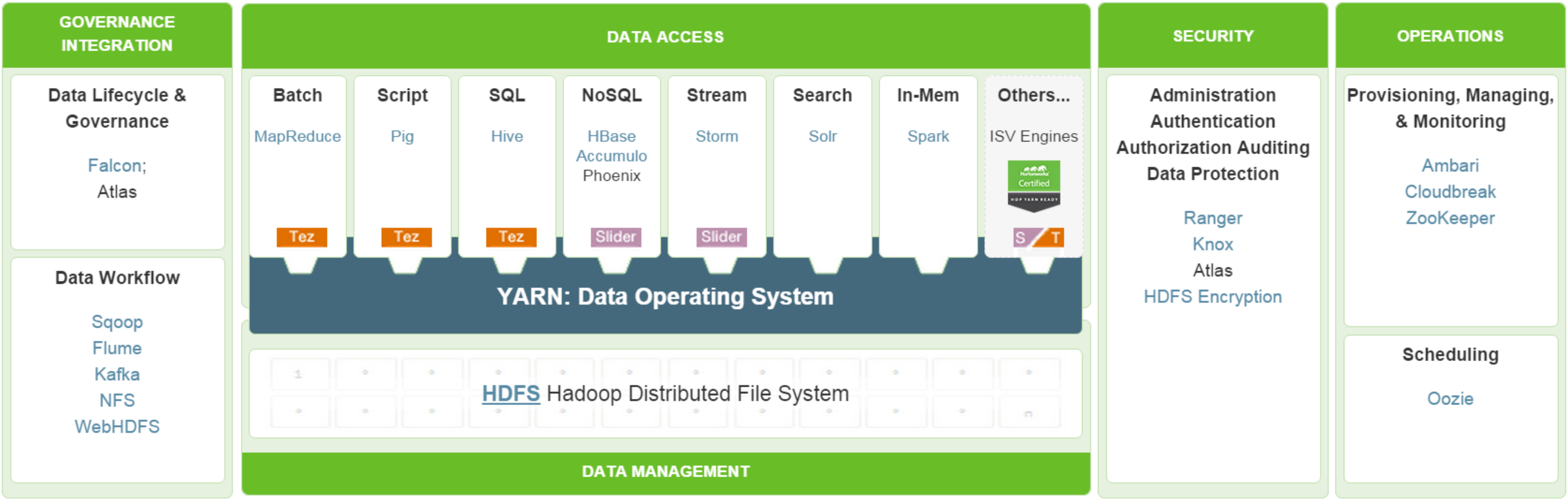


MAPREDUCE FRAMEWORK

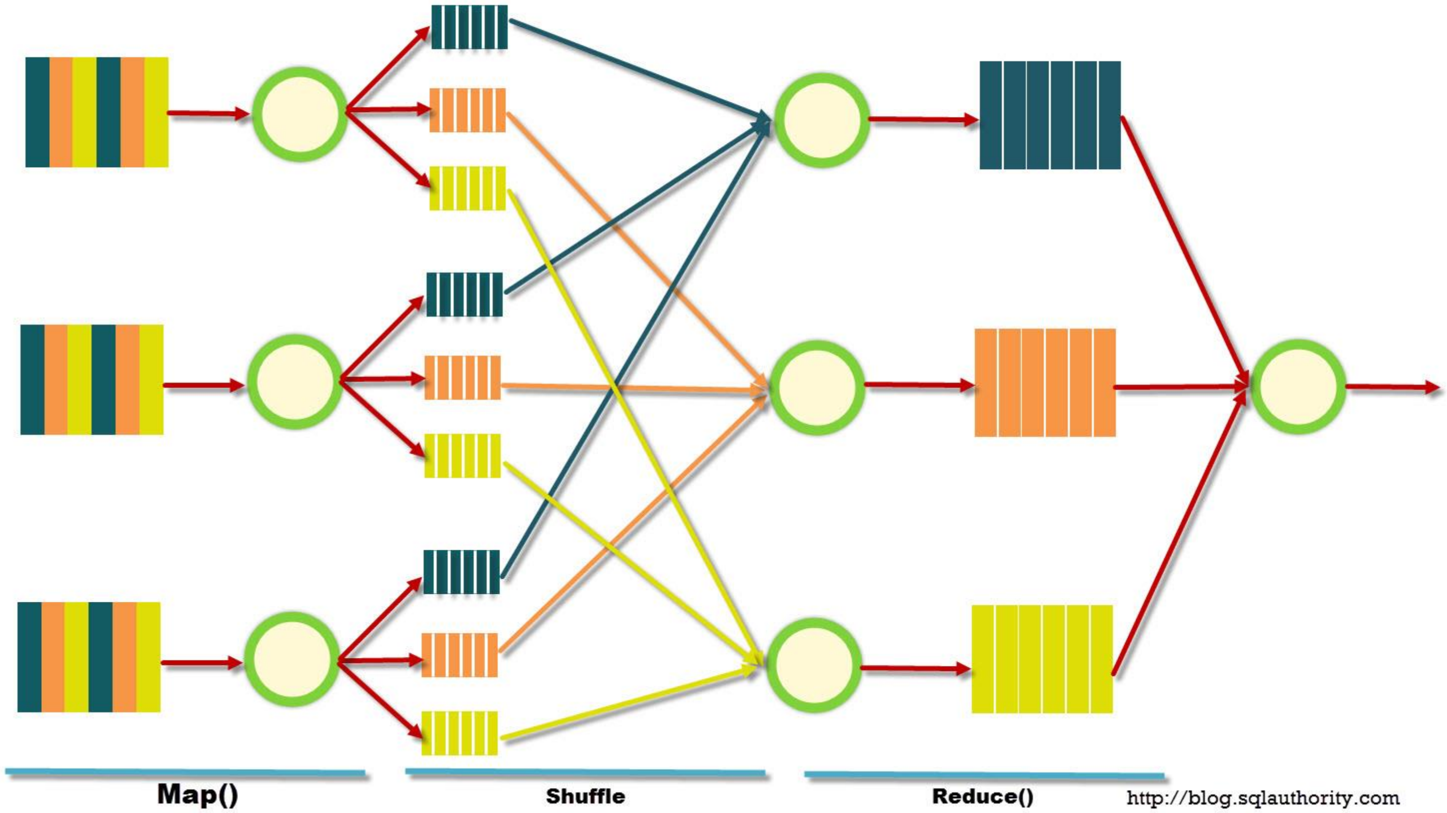
Pay attention!

Hadoop != MapReduce

Yet One YARN's lover



How MapReduce Works?



Main steps

- Map
- Shuffle
- Reduce

Minimal Runner

```
public static void main(String[] args) throws Exception {

  int exitCode = ToolRunner.run(new MinimalMapReduce(), args);
  System.exit(exitCode);

 }
}
```

Minimal Runner

```
public class MinimalMapReduce extends Configured implements Tool {
 @Override
 public int run(String[] args) throws Exception {









 }

 public static void main(String[] args) throws Exception {

  int exitCode = ToolRunner.run(new MinimalMapReduce(), args);
  System.exit(exitCode);

 }
}
```

```
public class MinimalMapReduce extends Configured implements Tool {
 @Override
 public int run(String[] args) throws Exception {


  Job job = new Job(getConf());
  job.setJarByClass(getClass());
  FileInputFormat.addInputPath(job, new Path(args[0]));
  FileOutputFormat.setOutputPath(job, new Path(args[1]));
  return job.waitForCompletion(true) ? 0 : 1;

 }

 public static void main(String[] args) throws Exception {

  int exitCode = ToolRunner.run(new MinimalMapReduce(), args);
  System.exit(exitCode);

 }
}
```

Minimal Runner

Job Config

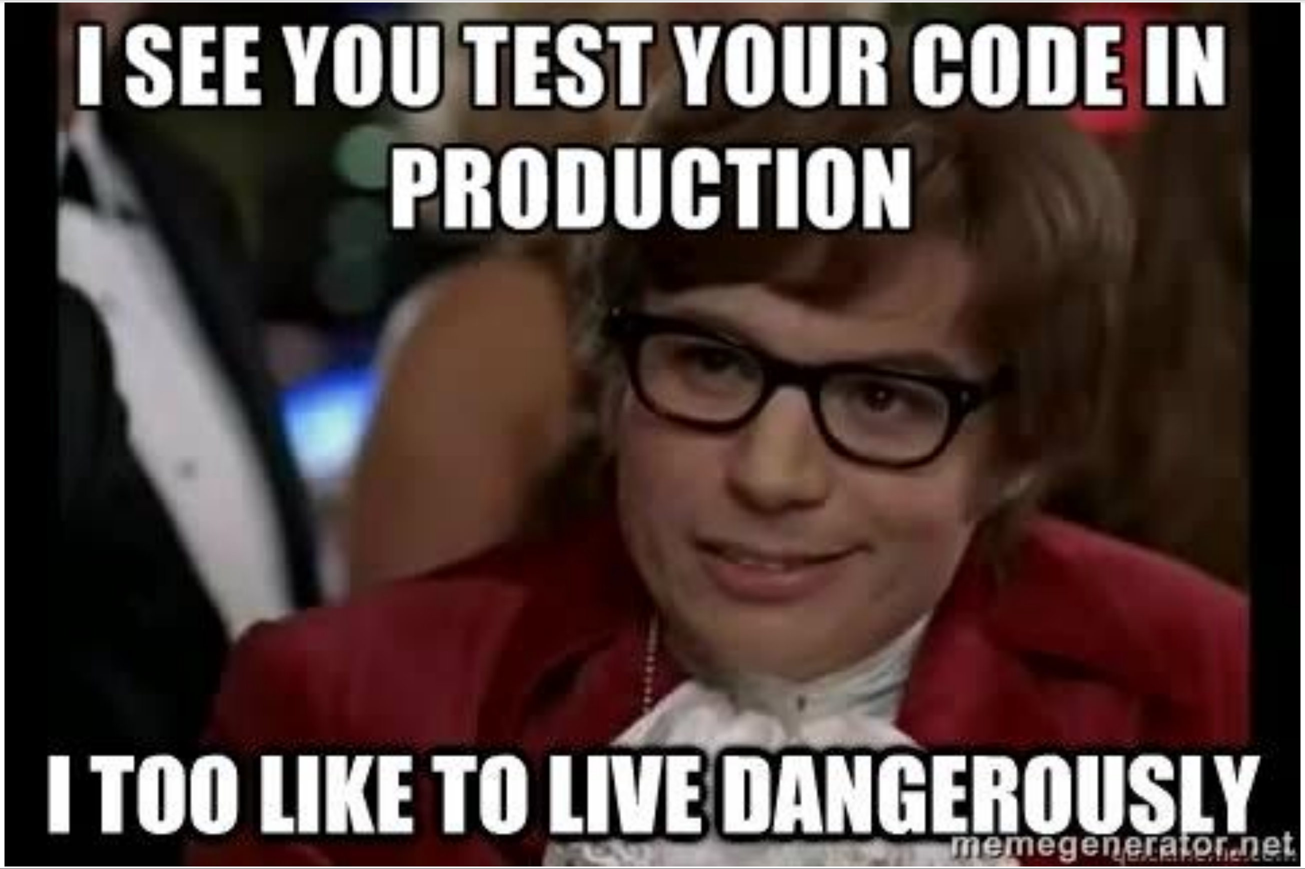
```
job.setInputFormatClass(TextInputFormat.class);

job.setMapperClass(Mapper.class);

job.setMapOutputKeyClass(LongWritable.class);

job.setMapOutputValueClass(Text.class);

job.setPartitionerClass(HashPartitioner.class);

job.setNumReduceTasks(1);

job.setReducerClass(Reducer.class);

job.setOutputKeyClass(LongWritable.class);

job.setOutputValueClass(Text.class);

job.setOutputFormatClass(TextOutputFormat.class);
```

Would you like to config in Java?

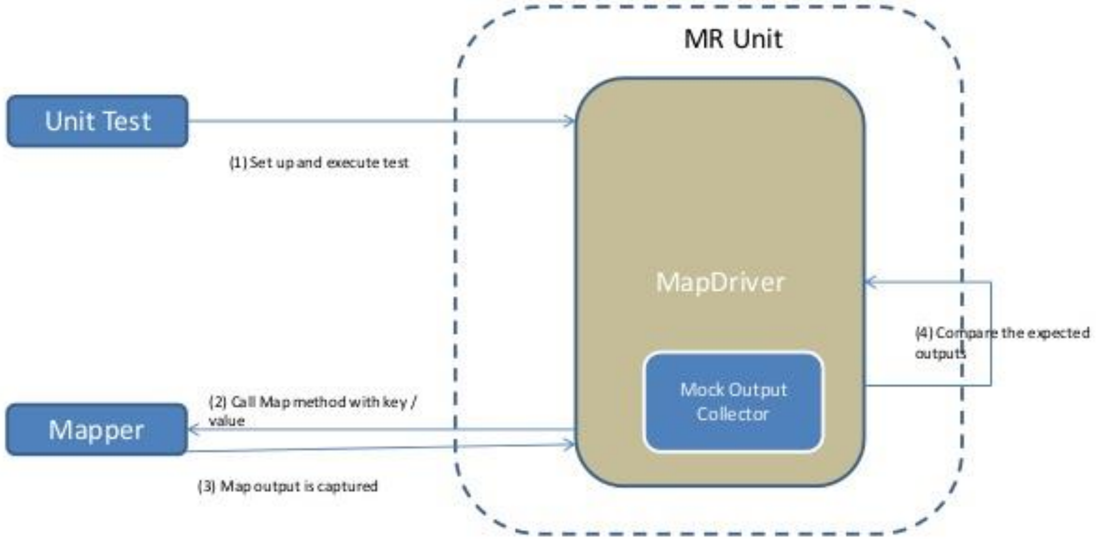
WordCount

TESTING & DEVELOPMENT

MR Unit idea


I SEE YOU TEST YOUR CODE IN PRODUCTION
I TOO LIKE TO LIVE DANGEROUSLY
memegenerator.net

MRUnit – Testing Mapper


MR Unit
Unit Test
(1) Set up and execute test
MapDriver
Mock Output Collector
(4) Compare the expected outputs
Mapper
(2) Call Map method with key / value
(3) Map output is captured

Do it separatly

Simple Test

```
public class MRUnitHelloWorld {
    MapDriver<LongWritable, Text, Text, IntWritable> mapDriver;

    @Before
    public void setUp() {
        WordMapper mapper = new WordMapper();
        mapDriver = new MapDriver<LongWritable, Text, Text,
IntWritable>();
        mapDriver.setMapper(mapper);
    }

    @Test
    public void testMapper() {
        mapDriver.withInput(new LongWritable(1), new Text("cat
dog"));
        mapDriver.withOutput(new Text("cat"), new IntWritable(1));
        mapDriver.withOutput(new Text("dog"), new IntWritable(1));
        mapDriver.runTest();
    }
}
```

Testing strategies

- First develop/test in local mode using small amount of data
- Test in pseudo-distributed mode and more data
- Test on fully distributed mode and even more data
- Final execution: fully distributed mode & all data

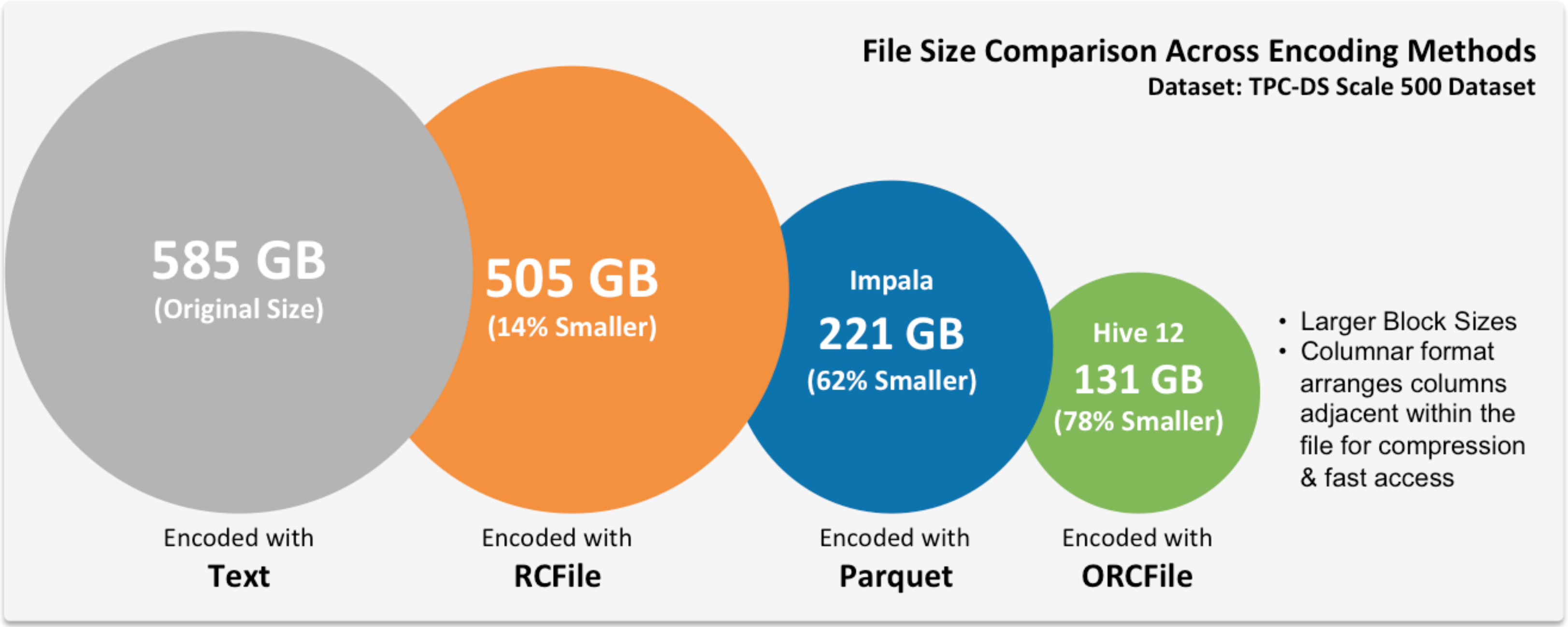
MRUnit

FILE FORMATS

Input/Output Formats

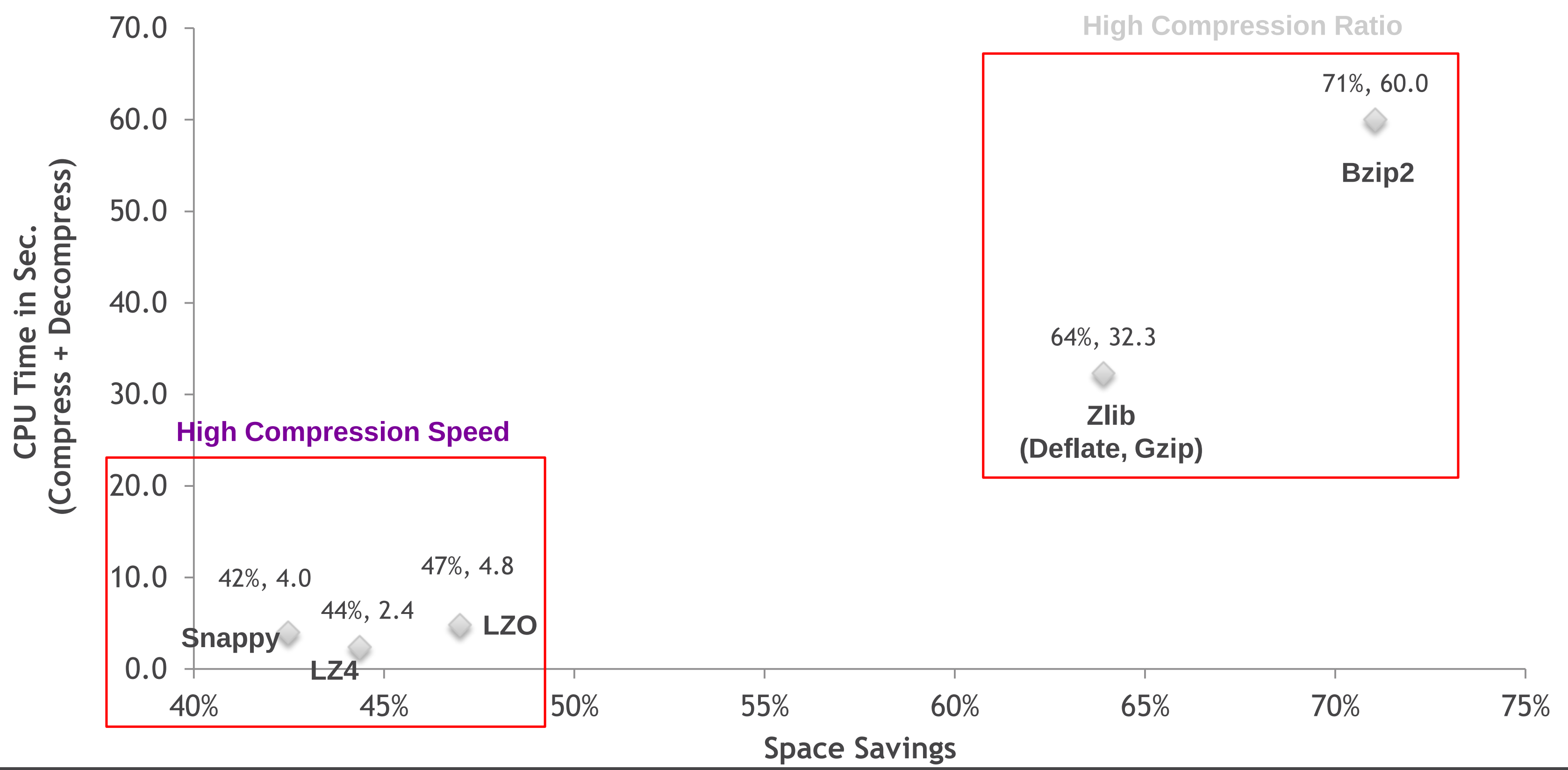
- Text based (CSV, TSV, JSON, XML)
- Sequence Files
- Column based (Parquet, RCFile, ORC)
- Avro
- HBase formats
- Custom formats

ORC vs PARQUET

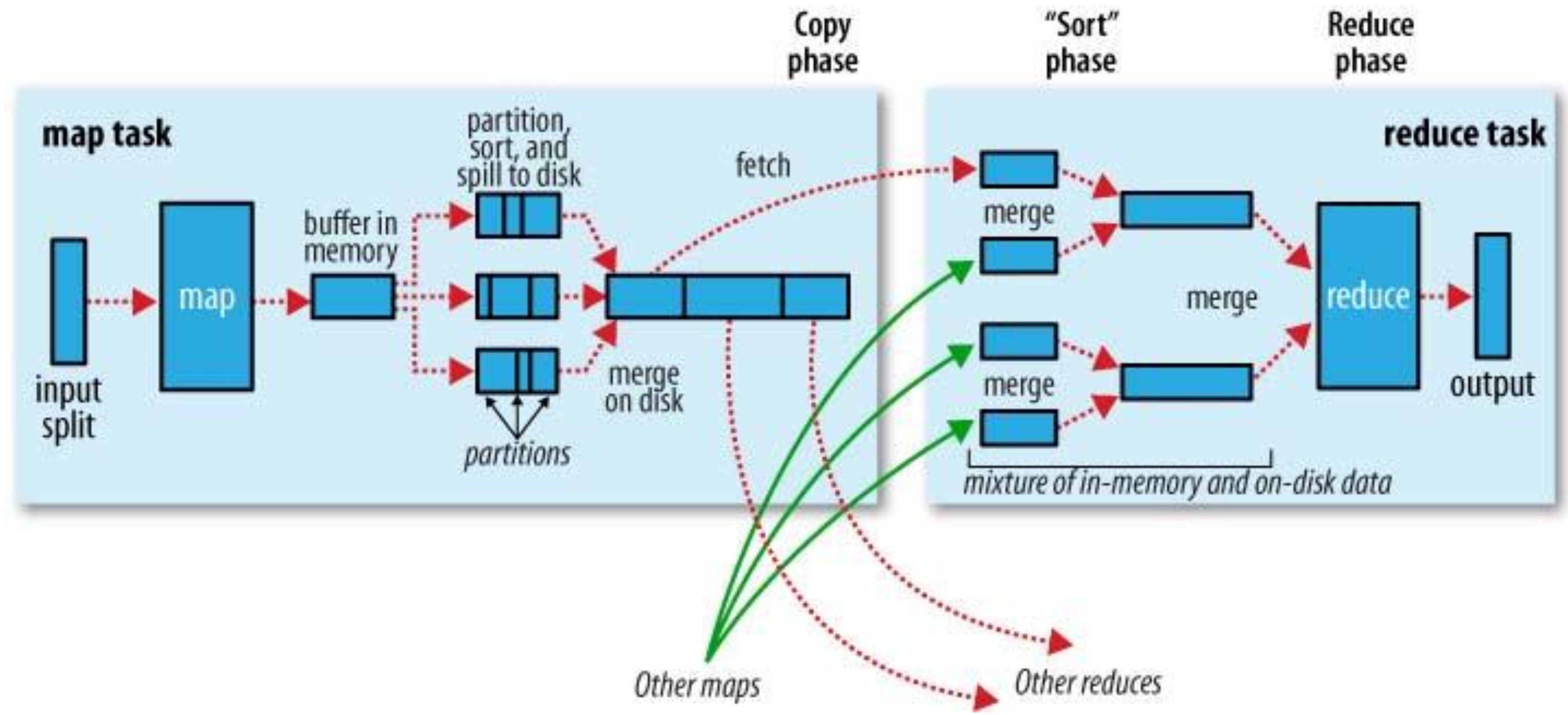


- Larger Block Sizes
- Columnar format arranges columns adjacent within the file for compression & fast access

Codec Performance on the Wikipedia Text Corpus



* LEVEL


Copy phase
"Sort" phase
Reduce phase
map task
reduce task
partition, sort, and spill to disk
fetch
buffer in memory
map
input split
partitions
merge on disk
merge
merge
merge
reduce
output
mixture of in-memory and on-disk data
Other maps
Other reduces

The main performance idea

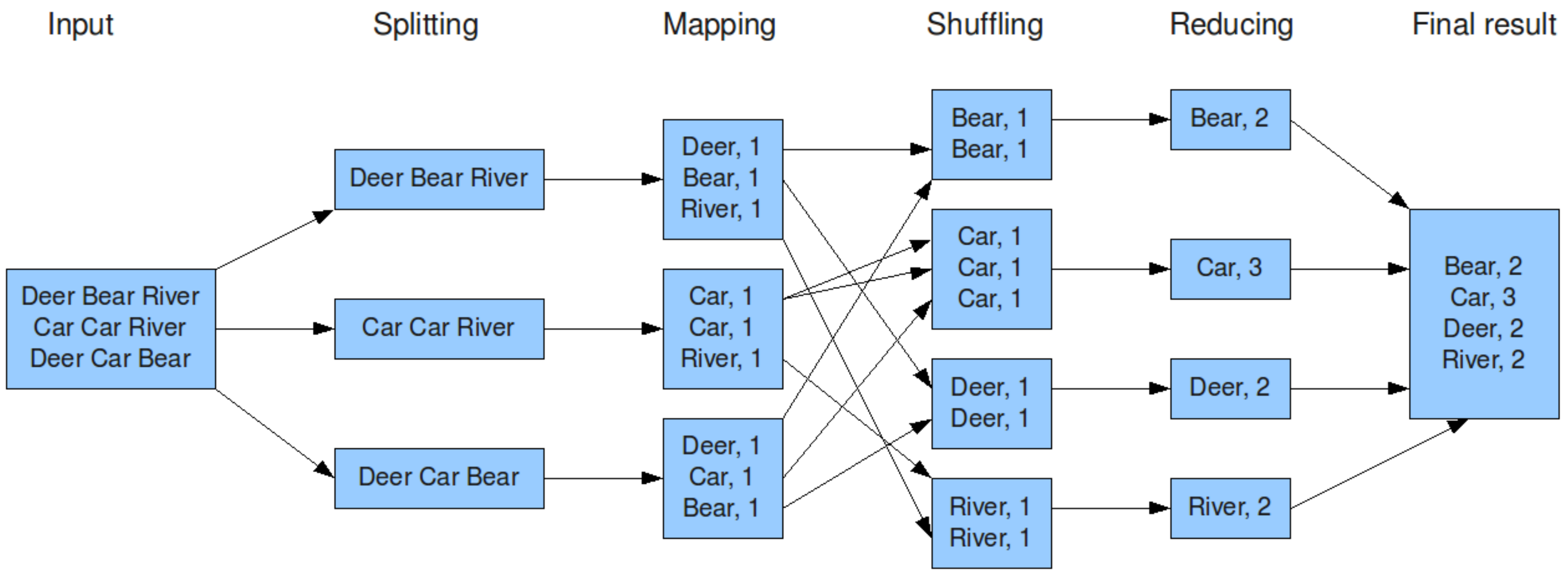
Reduce shuffle time & resources

MAPREDUCE ADVANCED

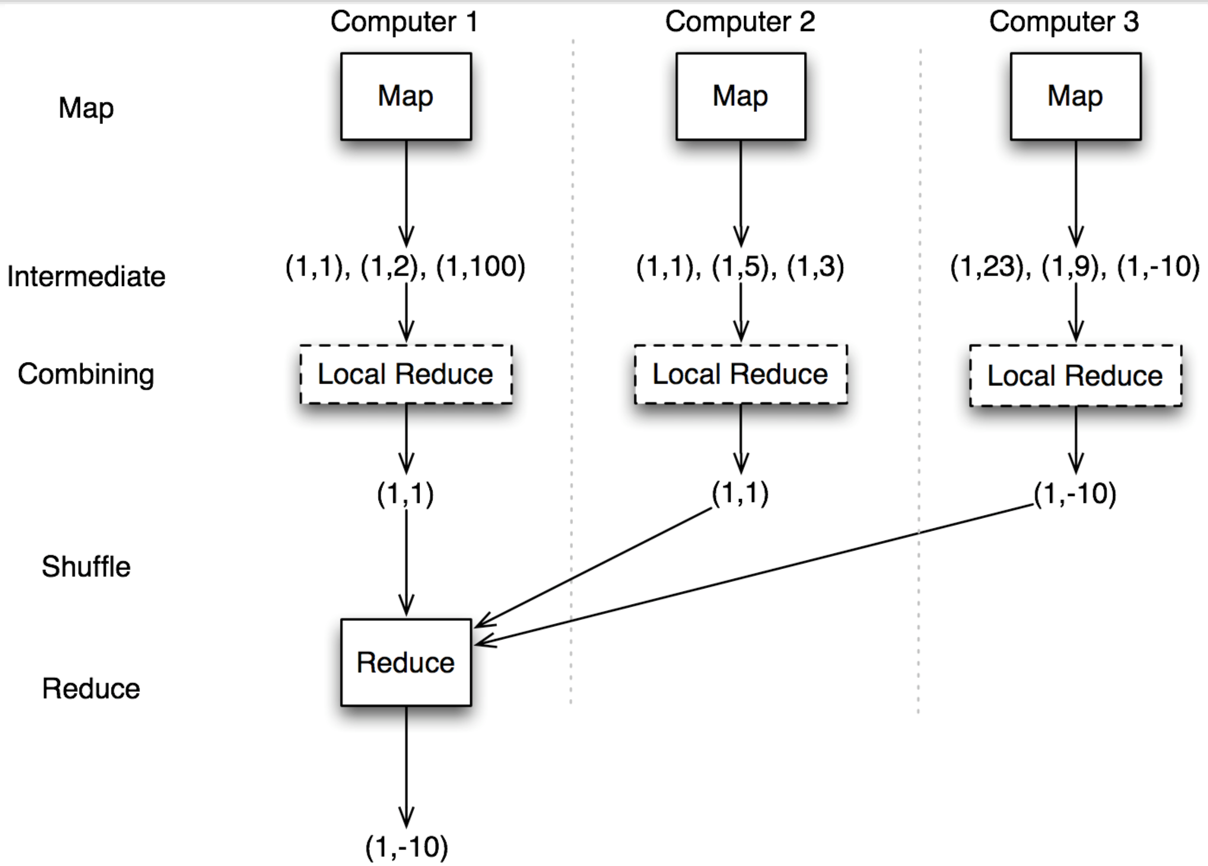
Customize MapReduce!

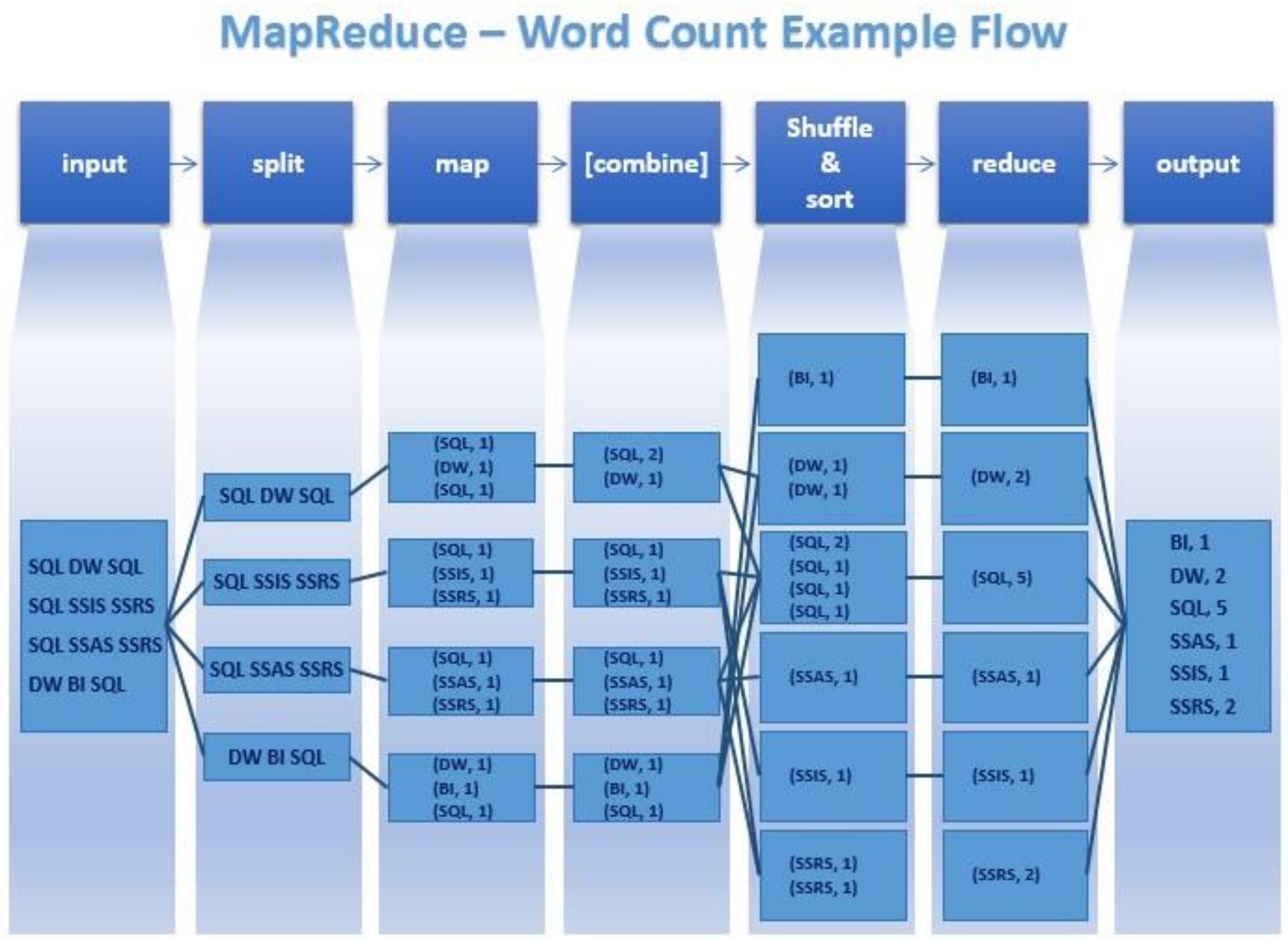
MapReduce for WordCount

The overall MapReduce word count process



WordCount Combiner




MapReduce – Word Count Example Flow
input
split
map
[combine]
Shuffle & sort
reduce
output
SQL DW SQL
SQL SSIS SSRS
SQL SSAS SSRS
DW BI SQL
SQL DW SQL
SQL SSIS SSRS
SQL SSAS SSRS
DW BI SQL
(SQL, 1) (DW, 1) (SQL, 1)
(SQL, 1) (SSIS, 1) (SSRS, 1)
(SQL, 1) (SSAS, 1) (SSRS, 1)
(DW, 1) (BI, 1) (SQL, 1)
(SQL, 2) (DW, 1)
(SQL, 1) (SSIS, 1) (SSRS, 1)
(SQL, 1) (SSAS, 1) (SSRS, 1)
(DW, 1) (BI, 1) (SQL, 1)
(BI, 1)
(DW, 1) (DW, 1)
(SQL, 2) (SQL, 1) (SQL, 1) (SQL, 1)
(SSAS, 1)
(SSIS, 1)
(SSRS, 1) (SSRS, 1)
(BI, 1)
(DW, 2)
(SQL, 5)
(SSAS, 1)
(SSIS, 1)
(SSRS, 2)
BI, 1
DW, 2
SQL, 5
SSAS, 1
SSIS, 1
SSRS, 2

Combine

Combiner

Can we make something around map(), reduce() calls?

Setup

```java
/**
 * Called once at the start of the task.
 */
protected void setup(Context context
                     ) throws IOException, InterruptedException {

   // Prepare something for each Mapper or Reducer
   // Validate external sources
}
```

```
/**
 * Called once at the end of the task.
 */
protected void cleanup(Context context
                       ) throws IOException, InterruptedException
{
  // Finish something after each Mapper or Reducer
  // Handle specific exceptions
}
```

Setup

Full control

Run Mapper

```
/**
 * Expert users can override this method for more complete control
over the
 * execution of the Mapper.
 * @param context
 * @throws IOException
 */
public void run(Context context) throws IOException,
InterruptedException {
  setup(context);
  try {
    while (context.nextKeyValue()) {
      map(context.getCurrentKey(), context.getCurrentValue(),
context);
    }
  } finally {
    cleanup(context);
  }
}
```

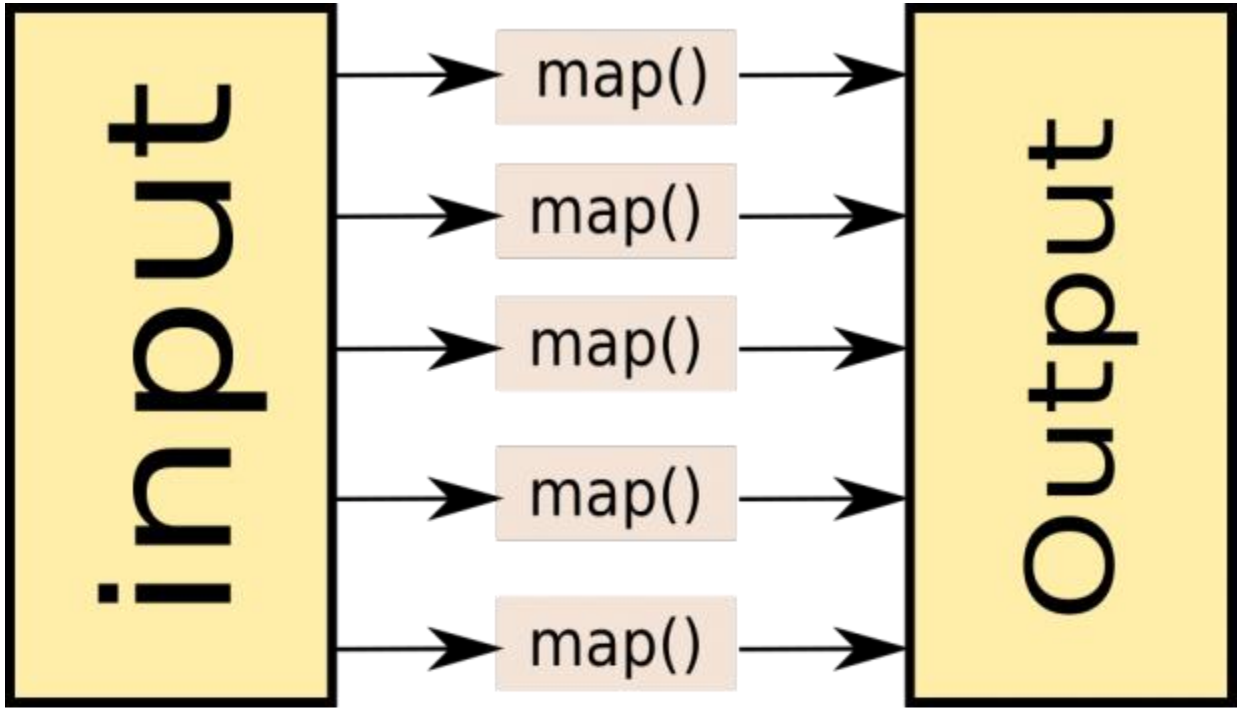
```
/**
 * Advanced application writers can use the
 * {@link #run(*.Reducer.Context)} method to
 * control how the reduce task works.
 */
public void run(Context context) throws IOException,
InterruptedException {
  setup(context);
  try {
    while (context.nextKey()) {
      reduce(context.getCurrentKey(), context.getValues(),
context);
      // If a back up store is used, reset it
      Iterator<VALUEIN> iter = context.getValues().iterator();
      if(iter instanceof ReduceContext.ValueIterator) {

((ReduceContext.ValueIterator<VALUEIN>)iter).resetBackupStore();
      }
    }
  } finally {
    cleanup(context);
  }
}
```

Run Reducer

REDUCER IS A PARTICIO OF PROBLEMO

Could we skip shuffle step?

Map-Only ‘MapReduce’ Jobs

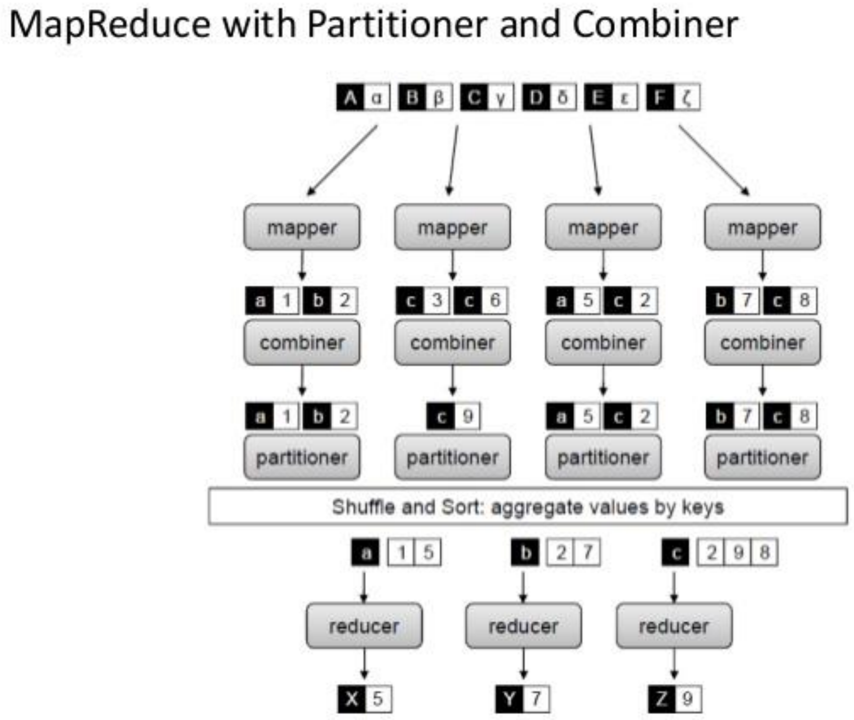


May I customize Data Flow before shuffling?

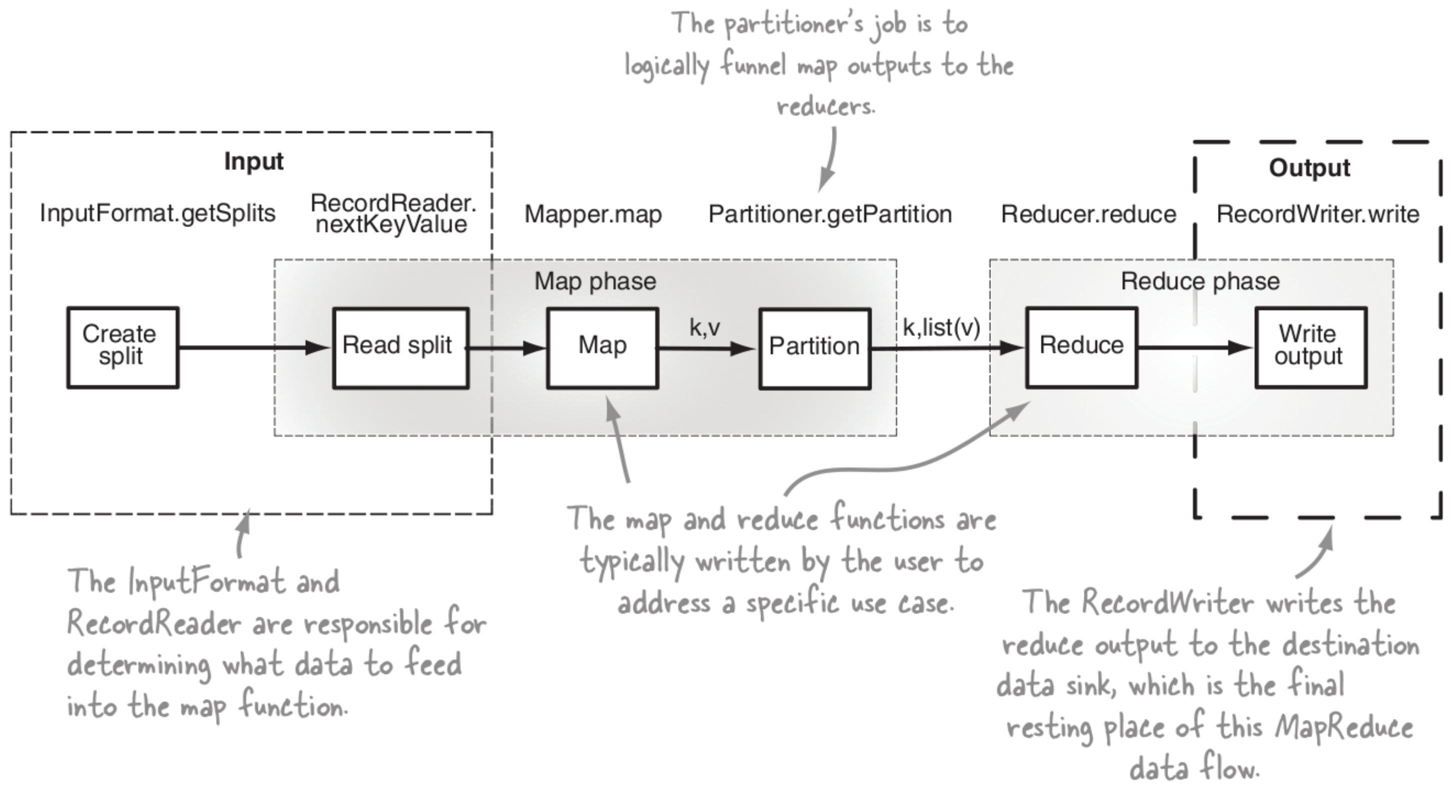
Hash Partitioner just do it..

```
public class HashPartitioner<K2, V2> extends Partitioner<K2, V2> {

public int getPartition(K2 key, V2 value, int numReduceTasks) {

        return (key.hashCode() & Integer.MAX_VALUE) % numReduceTasks;

    }
}
```

Partitioner’s Role in Shuffle and Sort



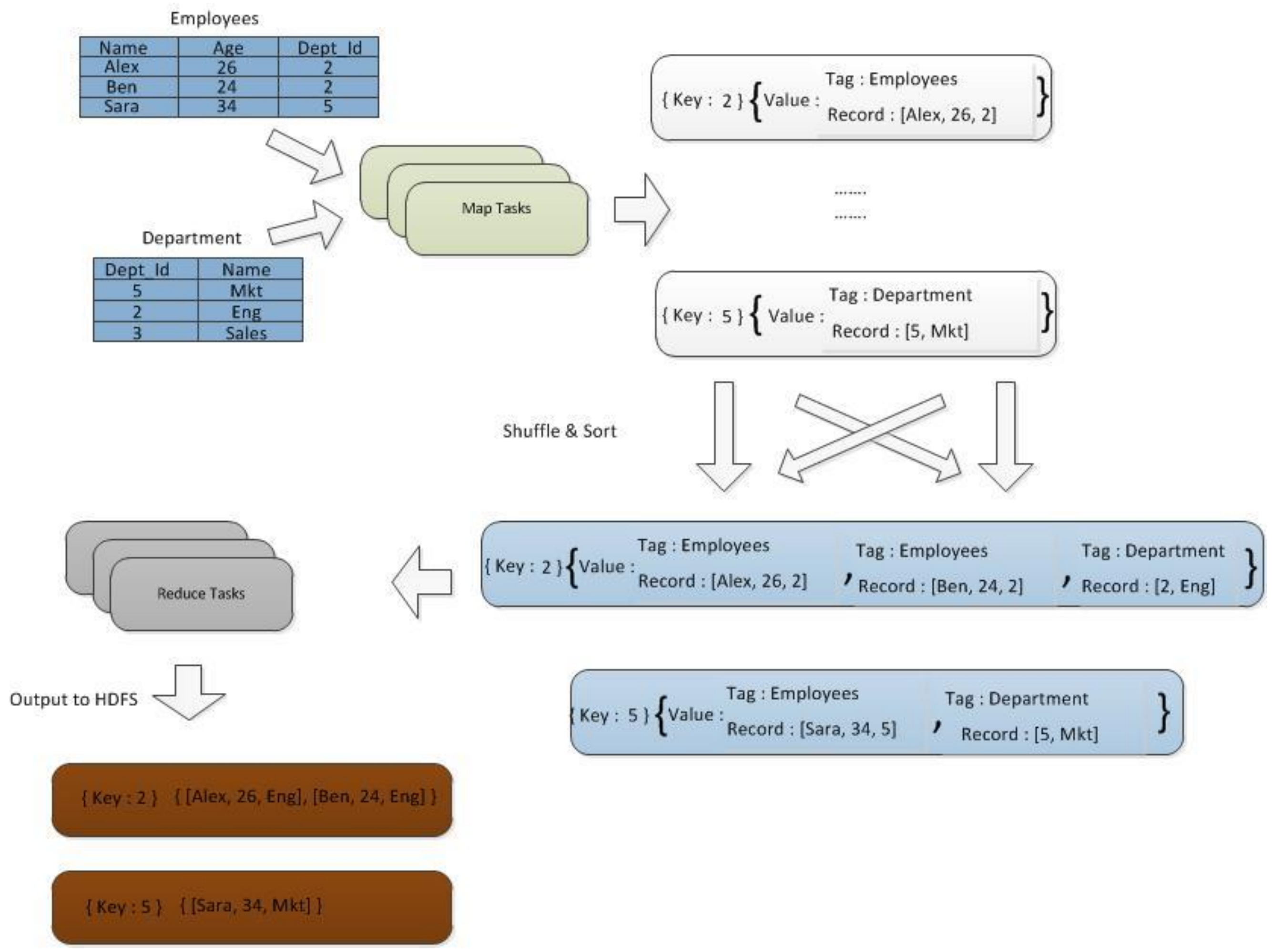
Full power


The partitioner's job is to logically funnel map outputs to the reducers.
Input
InputFormat.getSplits
RecordReader. nextKeyValue
Mapper.map
Partitioner.getPartition
Reducer.reduce
Output
RecordWriter.write
Map phase
Reduce phase
Create split
Read split
Map
k,v
Partition
k,list(v)
Reduce
Write output
The InputFormat and RecordReader are responsible for determining what data to feed into the map function.
The map and reduce functions are typically written by the user to address a specific use case.
The RecordWriter writes the reduce output to the destination data sink, which is the final resting place of this MapReduce data flow.

Custom Partitioner

JOINS

Reduce JOIN



Employees
Map Tasks
{ Key : 2 } { Value : Tag : Employees Record : [Alex, 26, 2] }
....
....
Department
{ Key : 5 } { Value : Tag : Department Record : [5, Mkt] }
Shuffle & Sort
Reduce Tasks
{ Key : 2 } { Value : Tag : Employees Record : [Alex, 26, 2] , Tag : Employees Record : [Ben, 24, 2] , Tag : Department Record : [2, Eng] }
{ Key : 5 } { Value : Tag : Employees Record : [Sara, 34, 5] , Tag : Department Record : [5, Mkt] }
Output to HDFS
{ Key : 2 } { [Alex, 26, Eng], [Ben, 24, Eng] }
{ Key : 5 } { [Sara, 34, Mkt] }

Employees

Name	Age	Dept_Id
Alex	26	2
Ben	24	2
Sara	34	5

Department

Dept_Id	Name
5	Mkt
2	Eng
3	Sales

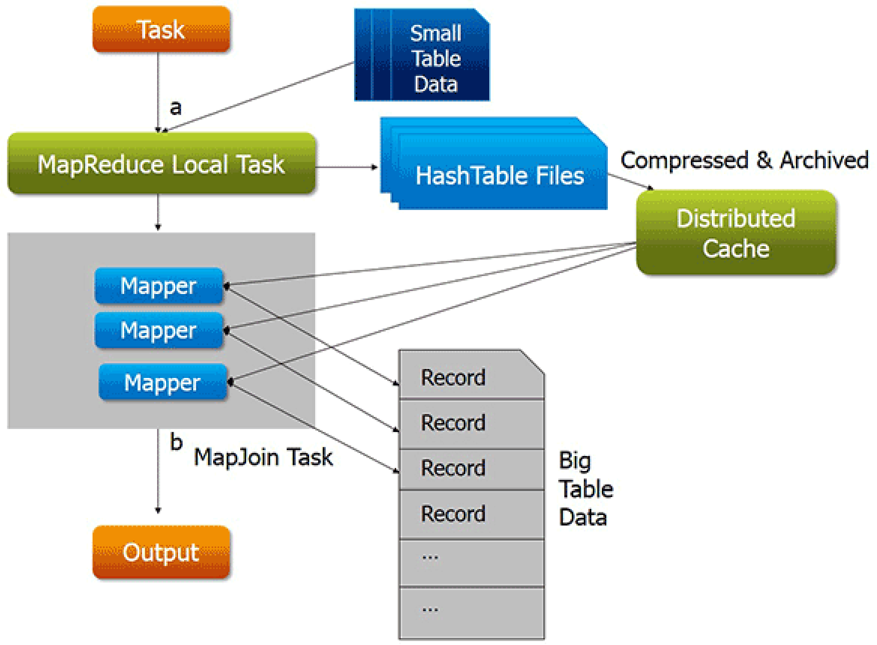
First idea

Let’s skip Reduce + Shuffle

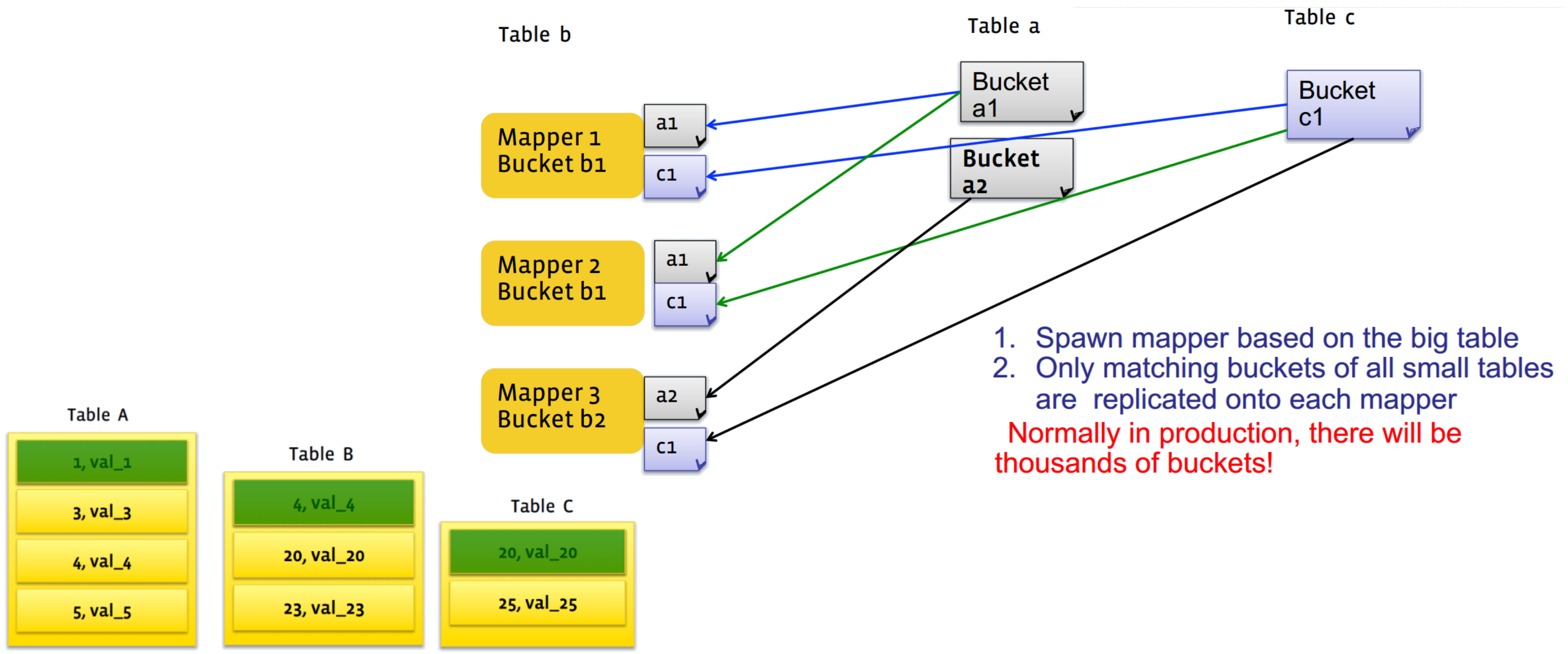
Second idea

Let’s copy small table on nodes

Map-side join



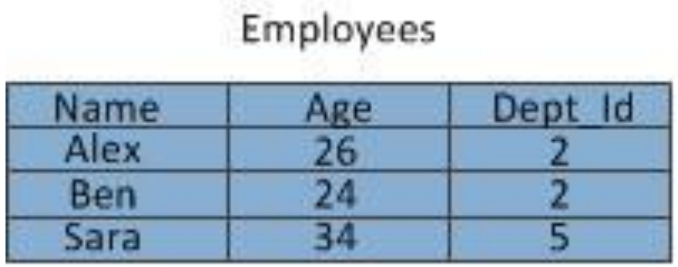
Map-side join for large datasets



1. Spawn mapper based on the big table
2. Only matching buckets of all small tables are replicated onto each mapper

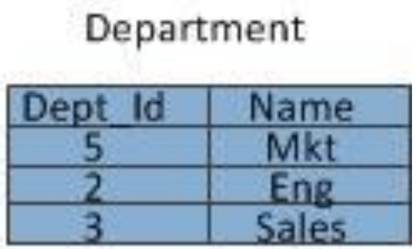
Normally in production, there will be thousands of buckets!

What about Really Large Tables?



Employees

Name	Age	Dept_Id
Alex	26	2
Ben	24	2
Sara	34	5



Department

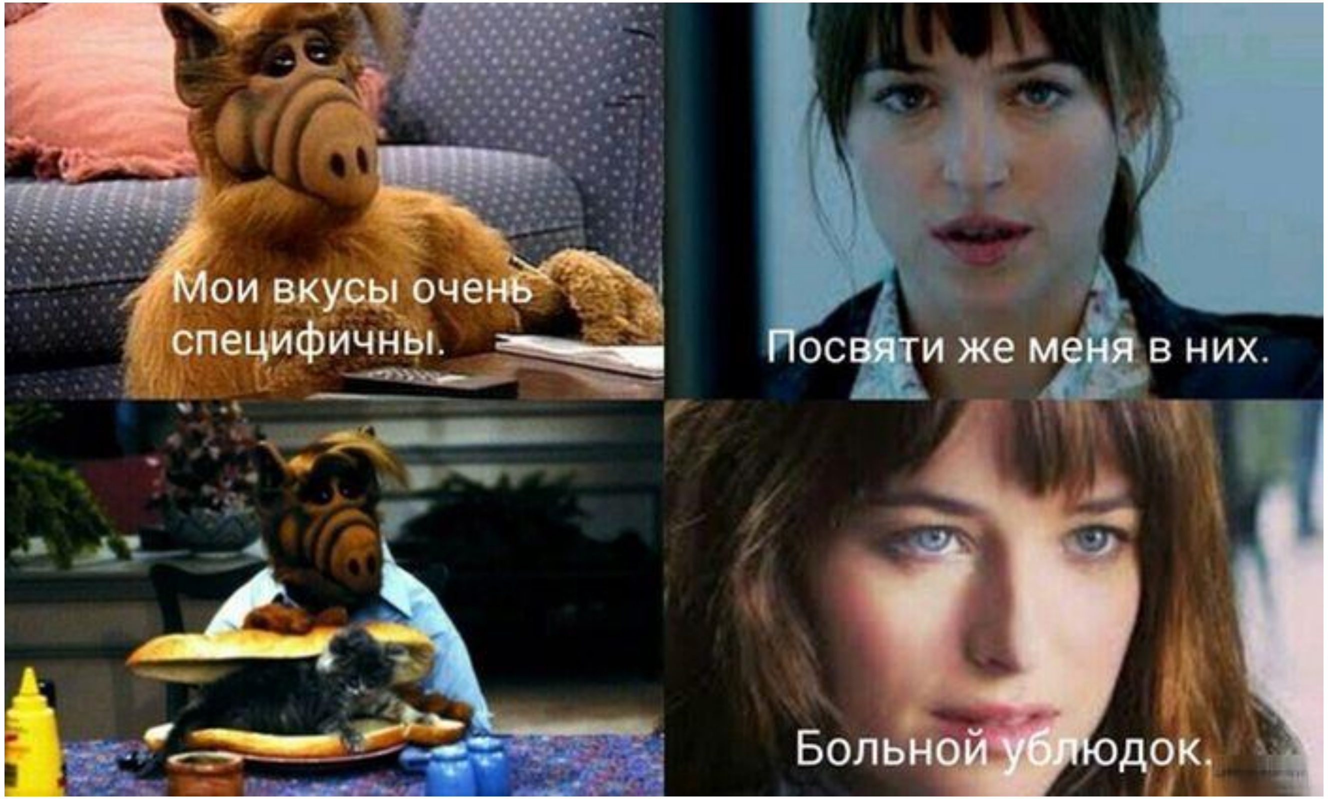
Dept_Id	Name
5	Mkt
2	Eng
3	Sales

SELECT Employees.Name, Employees.Age, Department.Name FROM Employees INNER JOIN Department ON Employees.Dept_Id=Department.Dept_Id

The main JOIN idea for large tables

Redis or Memcache cluster as Distributed Cache

PERFORMANCE



Let’s run on JVM!

Typical mistakes

- Collections are stored and sorted in memory
- Logging each input key-value pairs
- JARs hell
- Skew input: all records go to one reducer
- Forget that mapper/reduce is run on different JVM

Performance tips

- Correct data storage (on JVM ☺)
- Don’t forget about combiner
- Use appropriate Writable type
- Min required replication factor
- Tune your JVM
- Think in terms Big-O

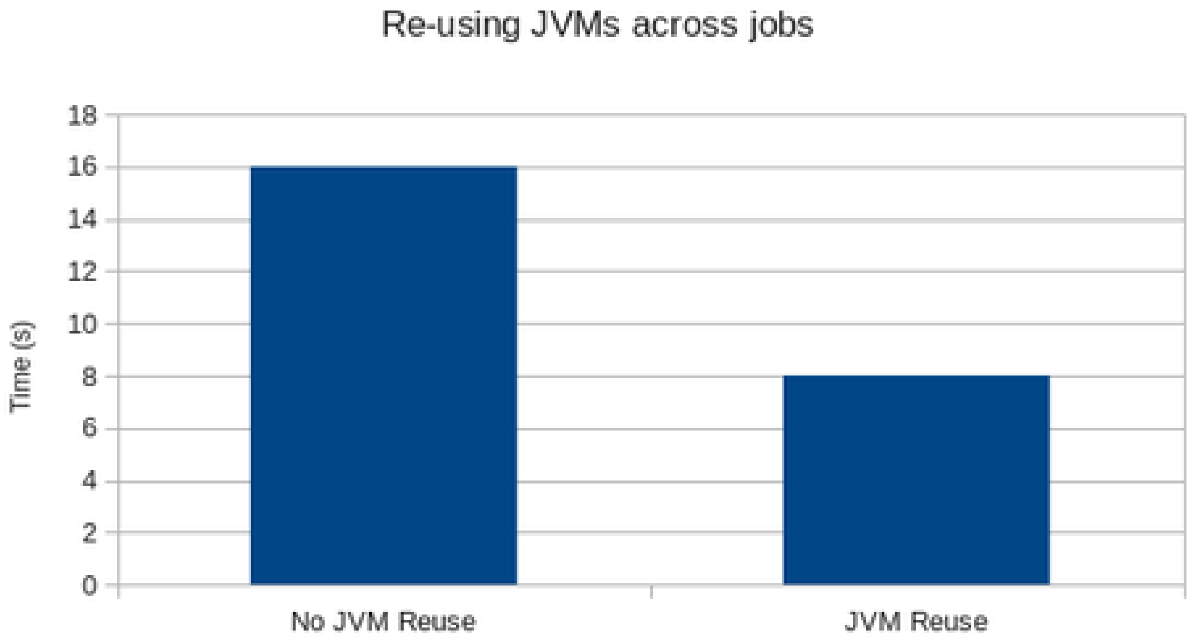
JVM tuning

- ***mapred.child.java.opts*** (heap for tasks)
- ***-XX:+PrintGCDetails -XX:+PrintGCTimeStamps***
- Low-latency GC collector ***-XX:+UseConcMarkSweepGC***,

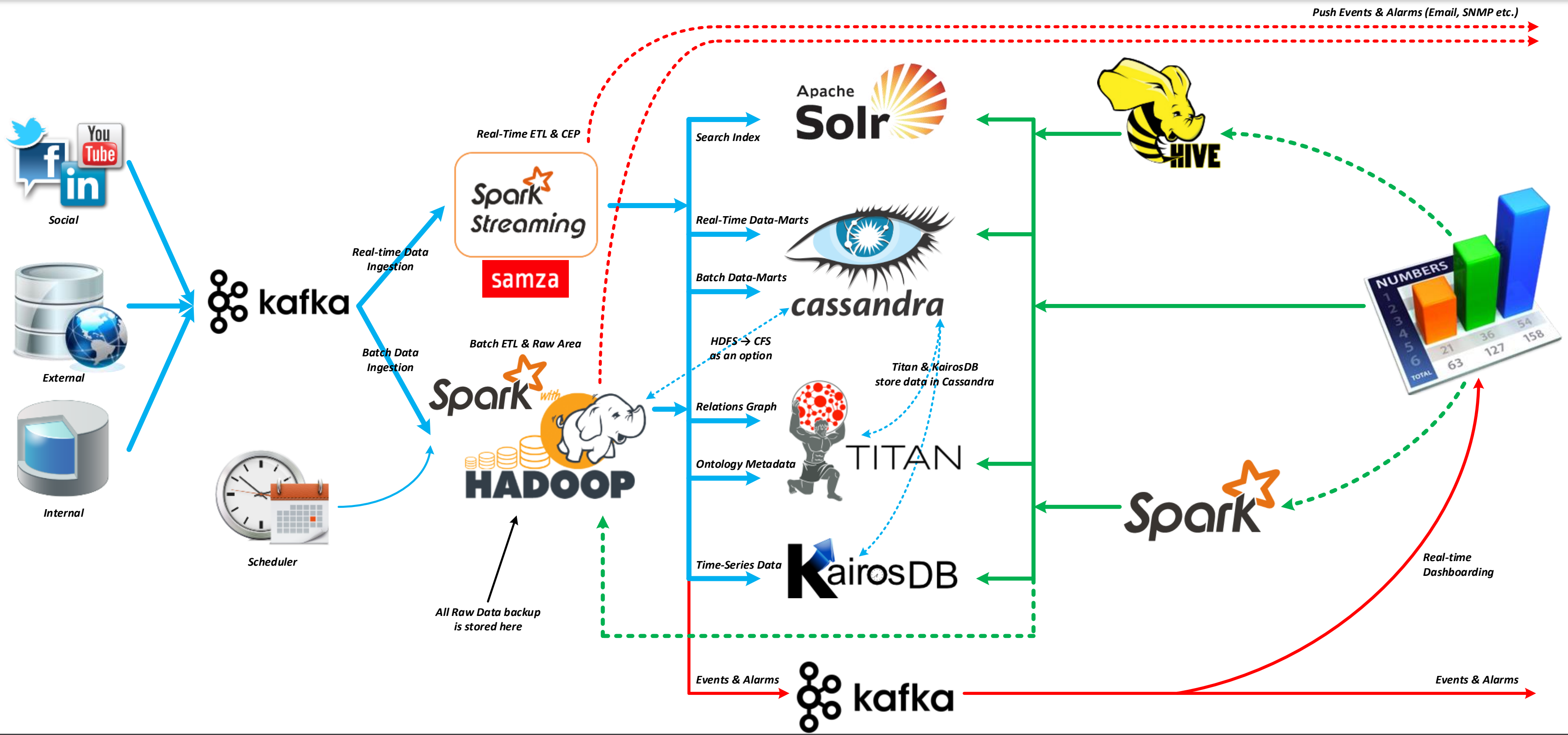
-XX:ParallelGCThreads

- Xmx == Xms (max and starting heap size)

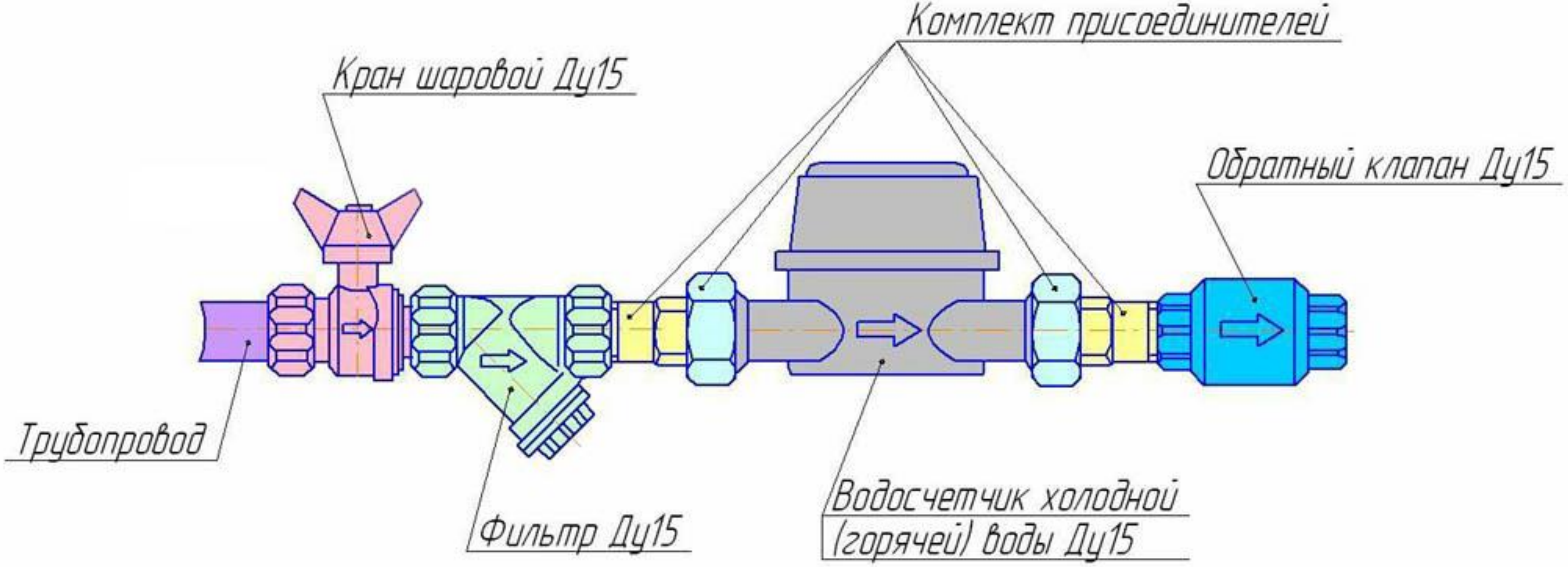
JVM Reusing: Uber Task



And we can DO IT!



It reminds me ...



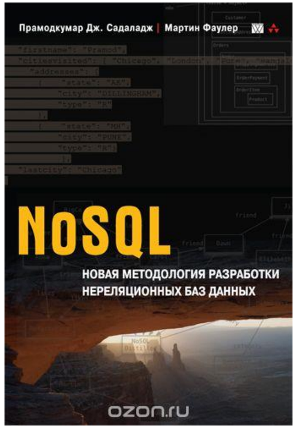
MapReduce is not a ideal approach! But it works!

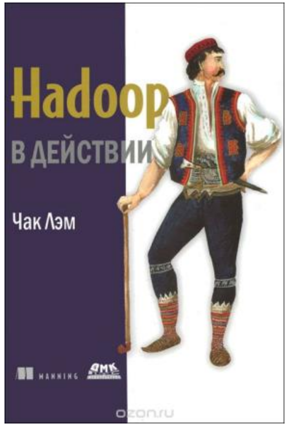
Hadoop 3: Roadmap

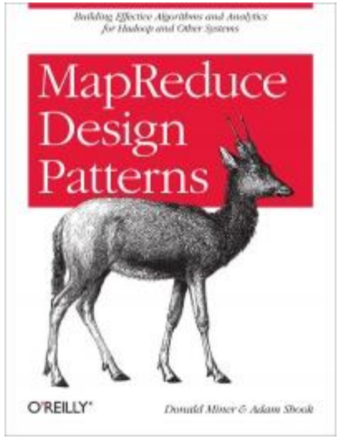
- Move to Java 8
- Support more than 2 NameNodes (multiple standby NameNodes)
- Derive heap size or mapreduce.*.memory.mb automatically
- Work with SSD, RAM, HDD, CPU as resources for YARN
- Support Docker containers

Would you like to know more?

Recommended Books


Прамодкумар Дж. Садаладж | Мартин Фаулер
NoSQL
НОВАЯ МЕТОДОЛОГИЯ РАЗРАБОТКИ НЕРЕЛЯЦИОННЫХ БАЗ ДАННЫХ
ozon.ru




Building Effective Algorithms and Analytics for Hadoop and Other Systems
MapReduce Design Patterns
O'REILLY
Donald Miner & Adam Shook

Изучаем Spark
Молниеносный анализ данных
Холден Карау, Энди Конвински, Патрик Венделл, Матей Захария
ДМК
O'REILLY

Contacts

E-mail : Alexey_Zinovyev@epam.com

Twitter : @zaleslaw @BigDataRussia

vk.com/big_data_russia **Big Data Russia**

+ Telegram @bigdatarussia

vk.com/java_jvm **Java & JVM langs**

+ Telegram @javajvmlangs

Github

Spark Tutorial: Core, Streaming, Machine Learning

https://github.com/zaleslaw/Spark-Tutorial

Gitbook

Обработка данных на Spark 2.2 и Kafka 0.10

www.gitbook.com/book/zaleslaw/data-processing-book

Any questions?