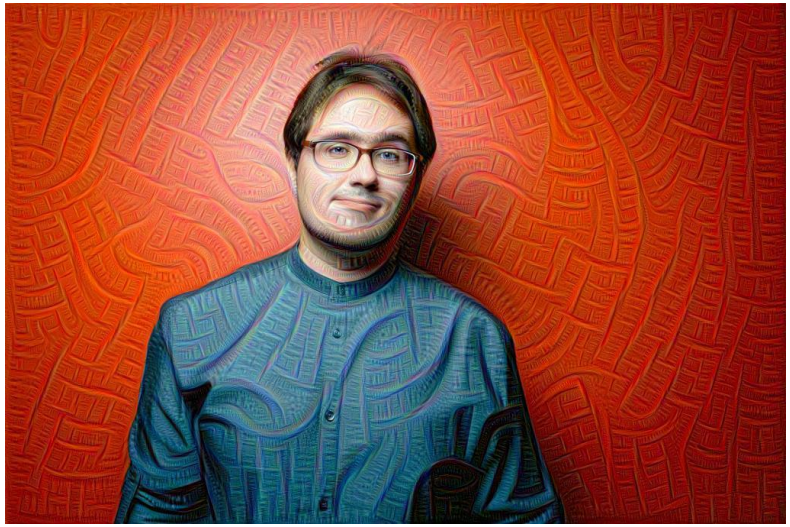

epam
Hadoop Jungle
Alexey Zinovyev, Java/BigData Trainer in EPAM
hadoop



About

I am a <graph theory, machine learning, traffic jams prediction, BigData algorithms> scientist

But I'm a <Java, Scala, NoSQL, Hadoop, Spark> programmer and trainer

Contacts

E-mail : Alexey_Zinovyev@epam.com

Twitter : @zaleslaw @BigDataRussia

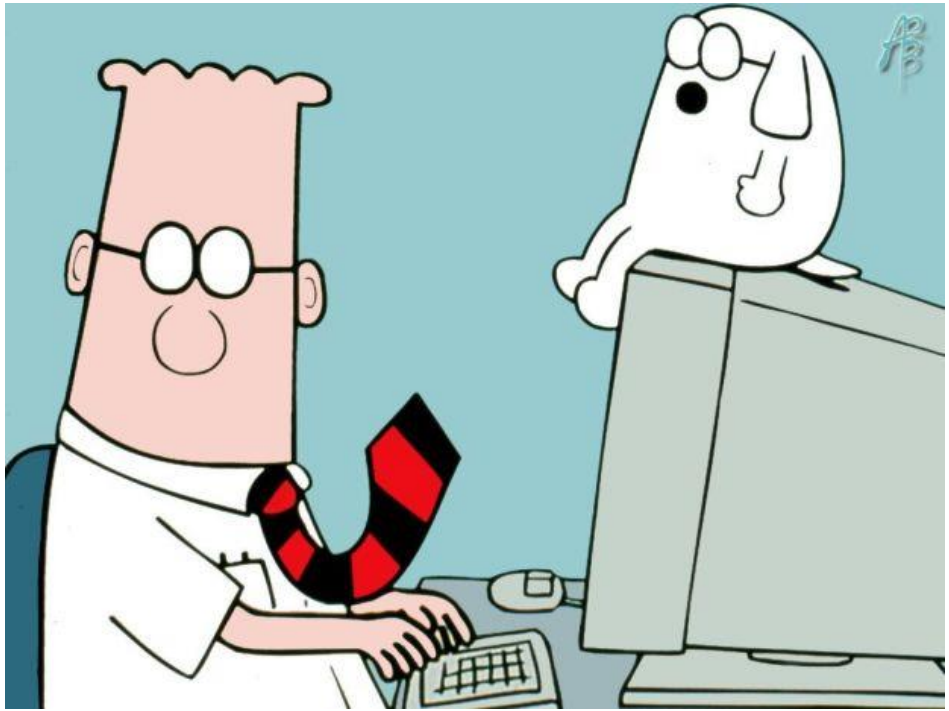
vk.com/big_data_russia **Big Data Russia**

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

Main parts

- What is BIG DATA?
- Intro in Hadoop
- HDFS
- Classic MapReduce and modern customization
- JOINs techniques*
- Development strategies*
- Compression and File Formats*

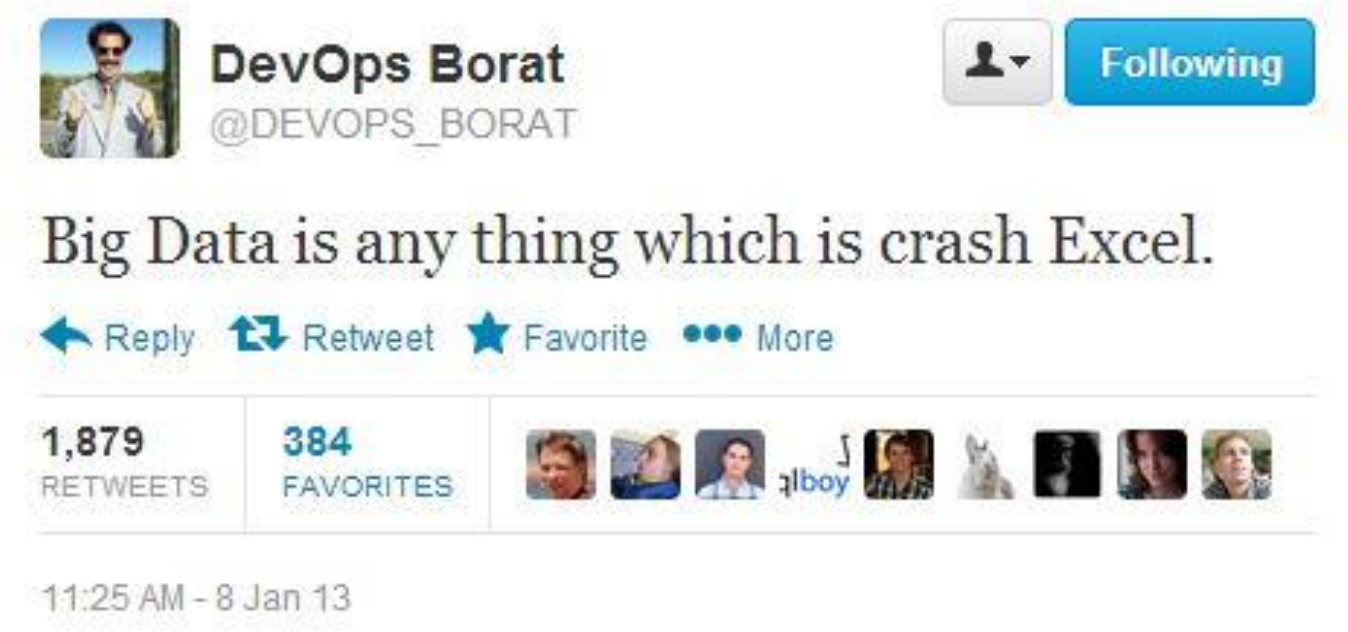
Are you a Hadoop developer?



Let’s do THIS!

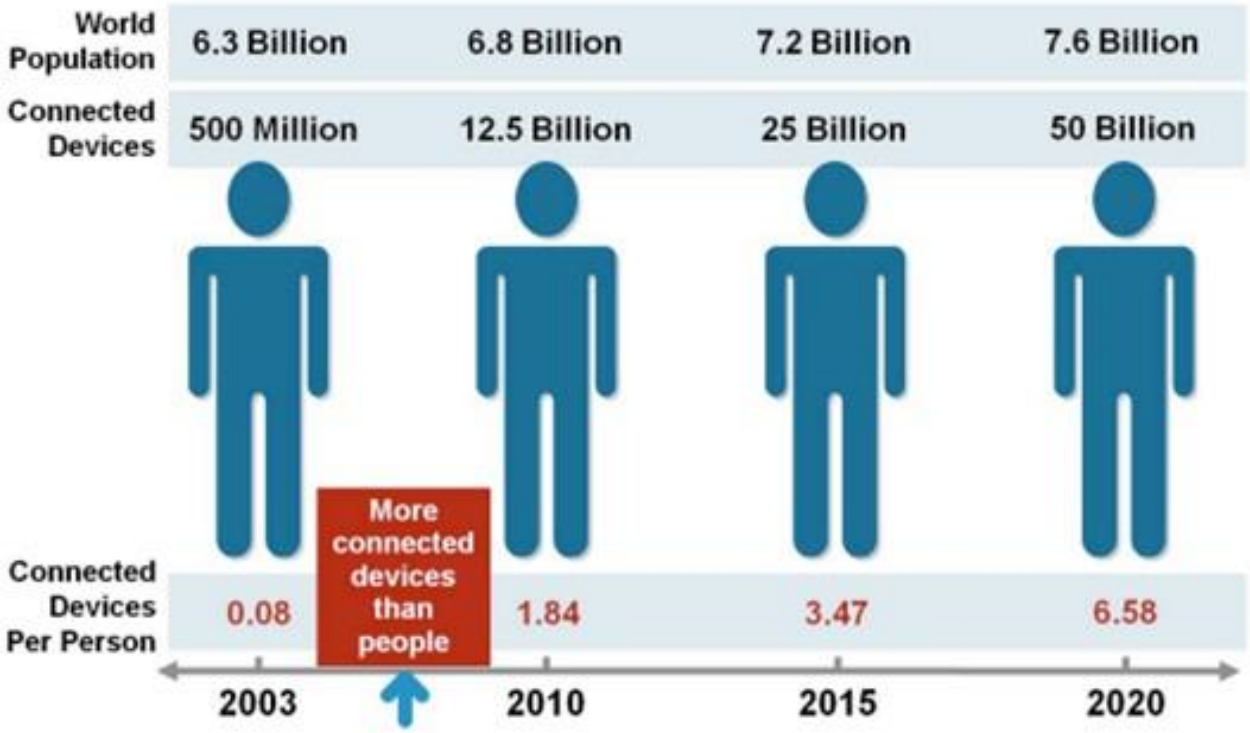
WHAT IS BIG DATA?

Joke about Excel

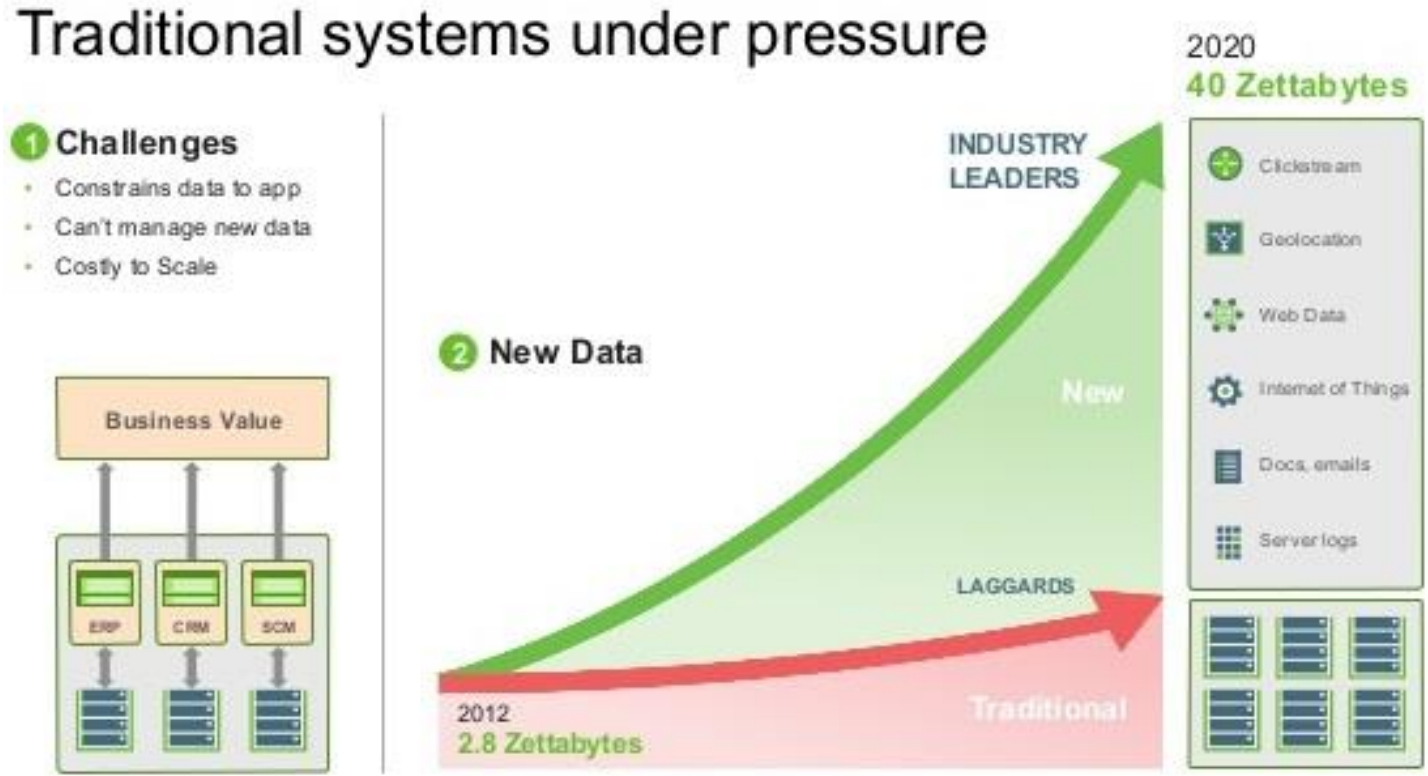


Every 60 seconds...

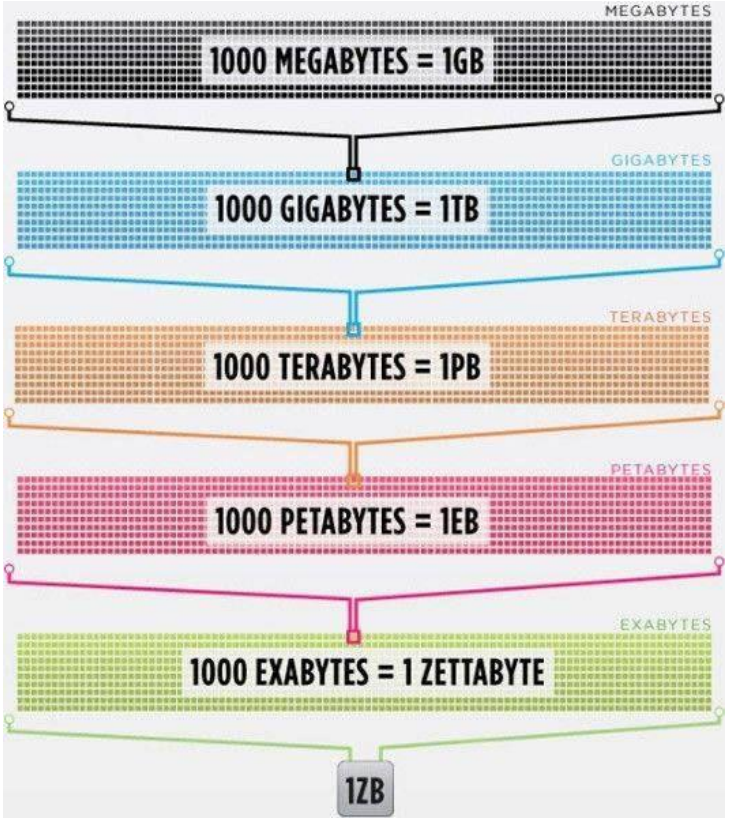
From Mobile Devices



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

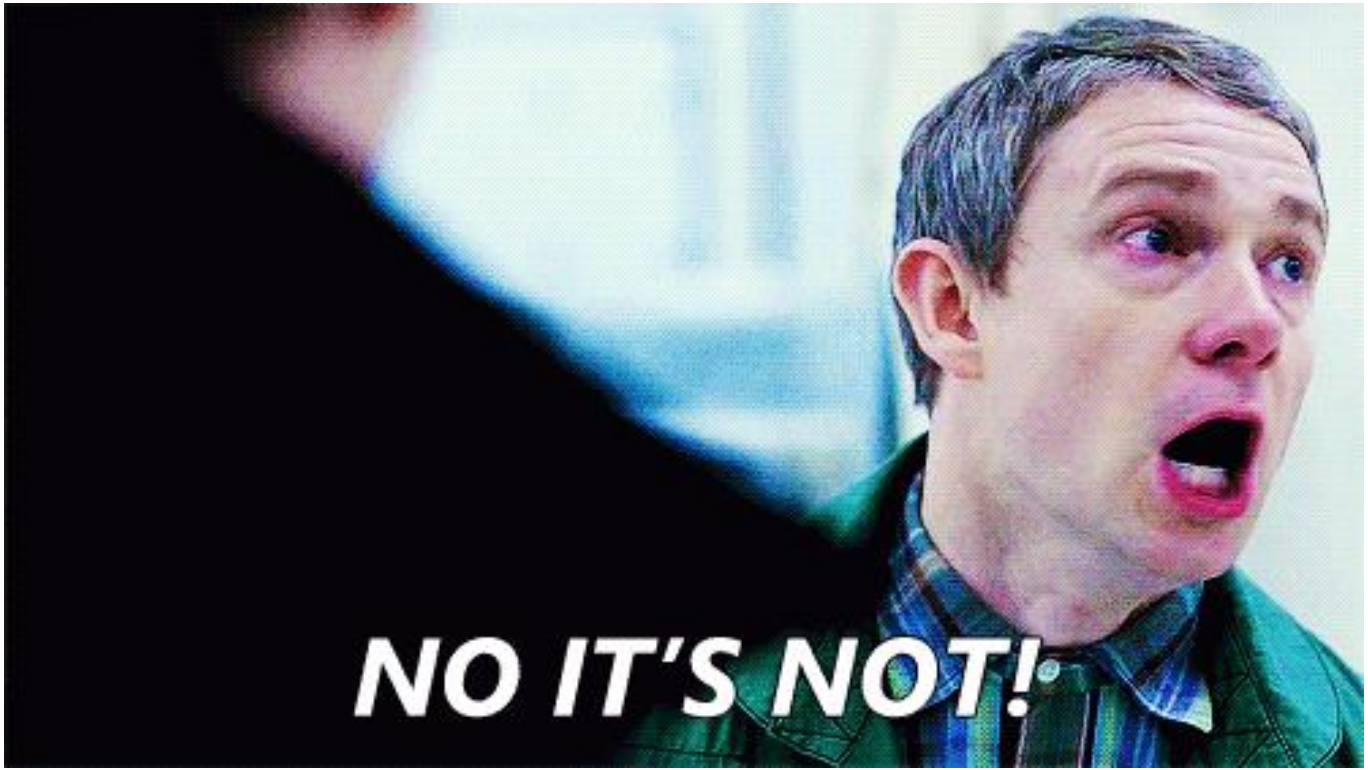


GB->TB->PB->?


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?

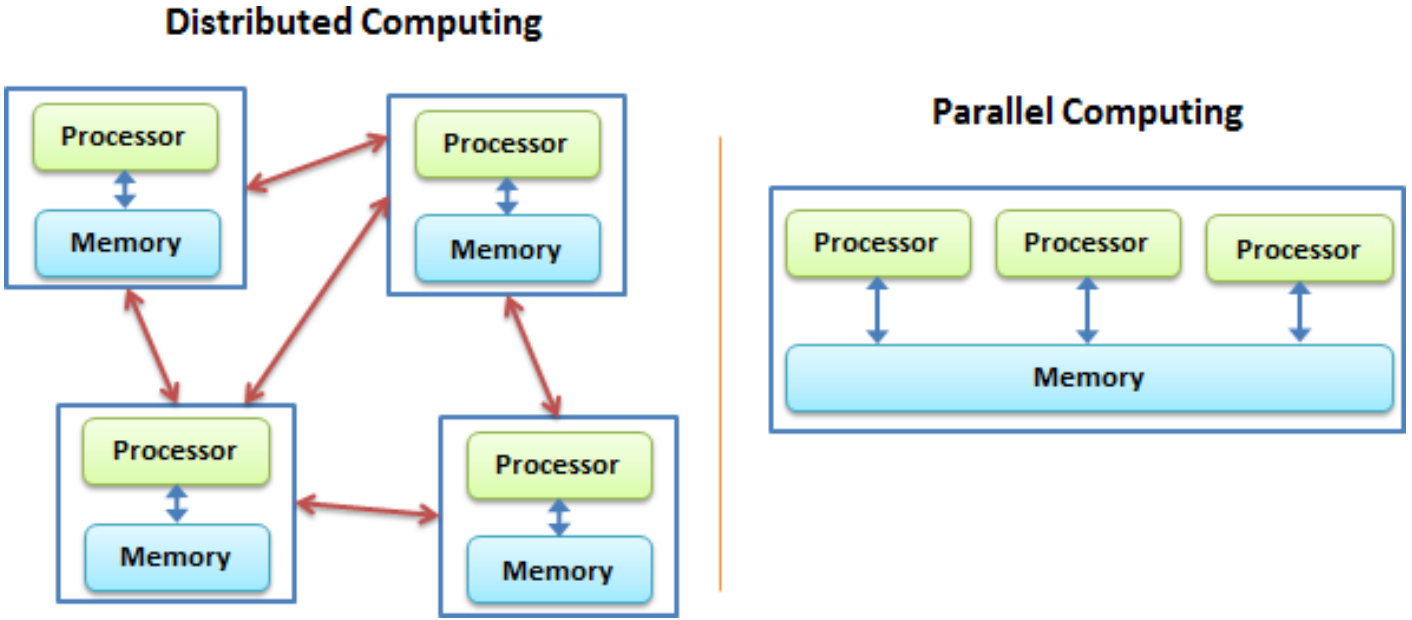


It’s hard to …

- .. 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

You need to develop

- .. distributed on-disk storage
- .. in-memory storage (or shared memory buffer)

You need to develop

- .. distributed on-disk storage
- .. in-memory storage (or shared memory buffer)
- .. thread pool to run hundreds of threads

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

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!

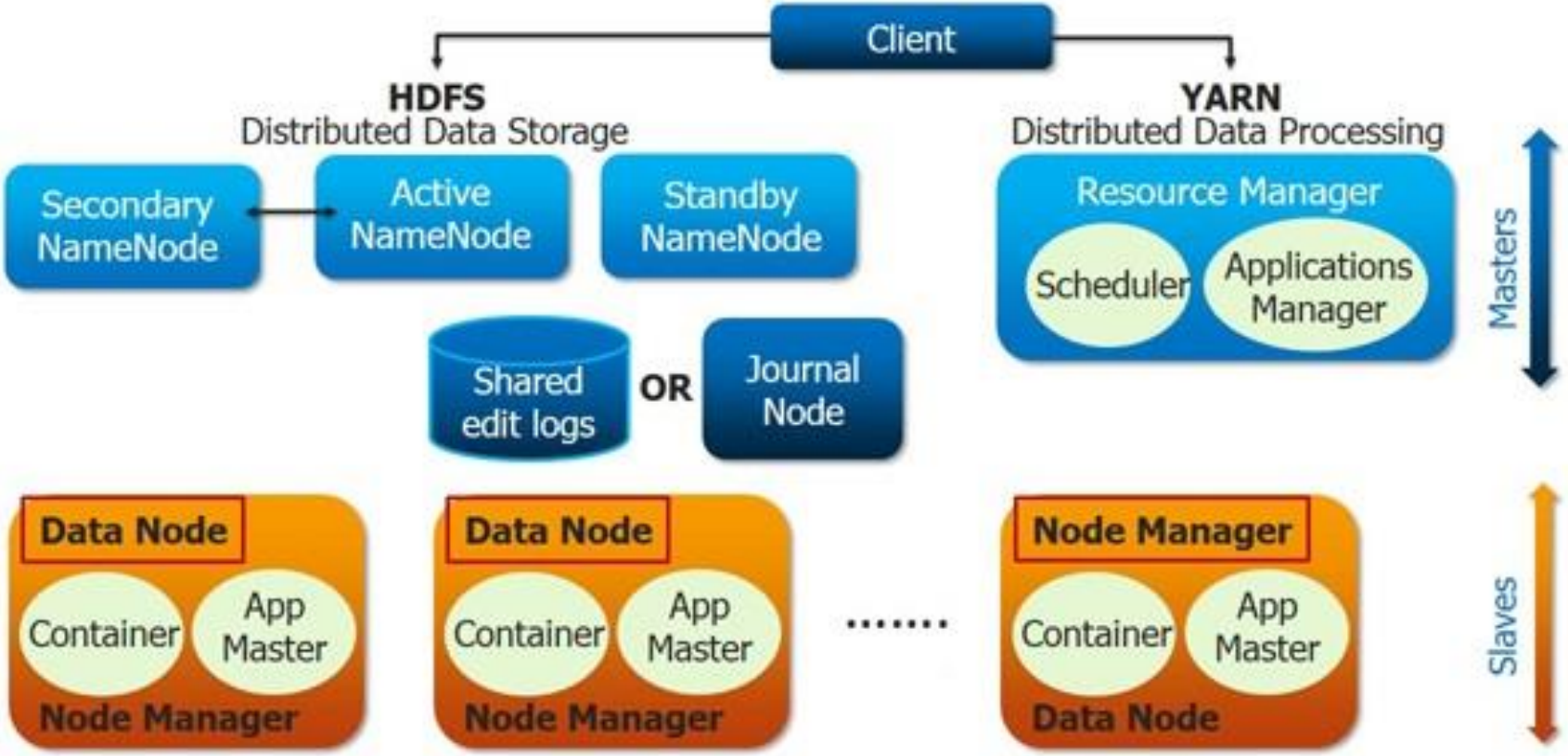
Main components

- 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

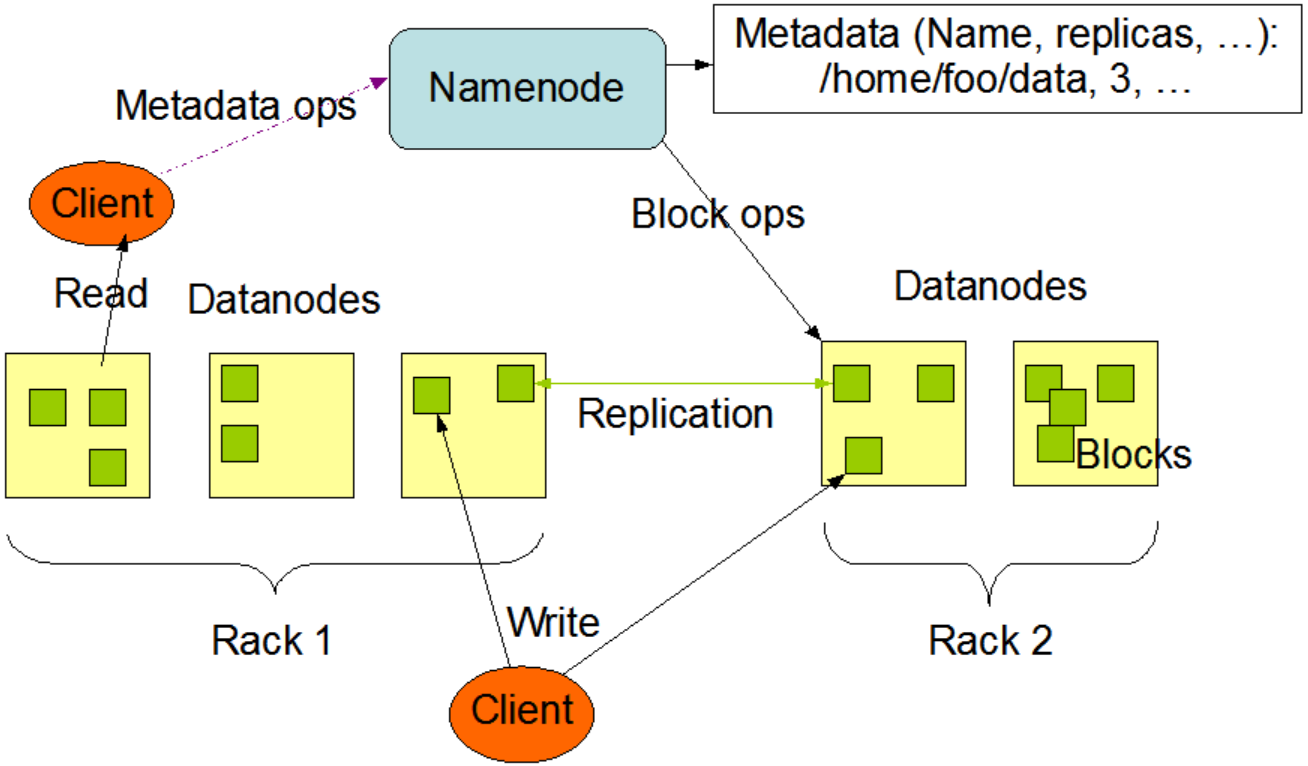
- Automatic parallelization and distribution
- Fault-tolerance
- Data Locality
- Writing the Map and Reduce functions only
- Single-threaded model

HDFS DAEMONS

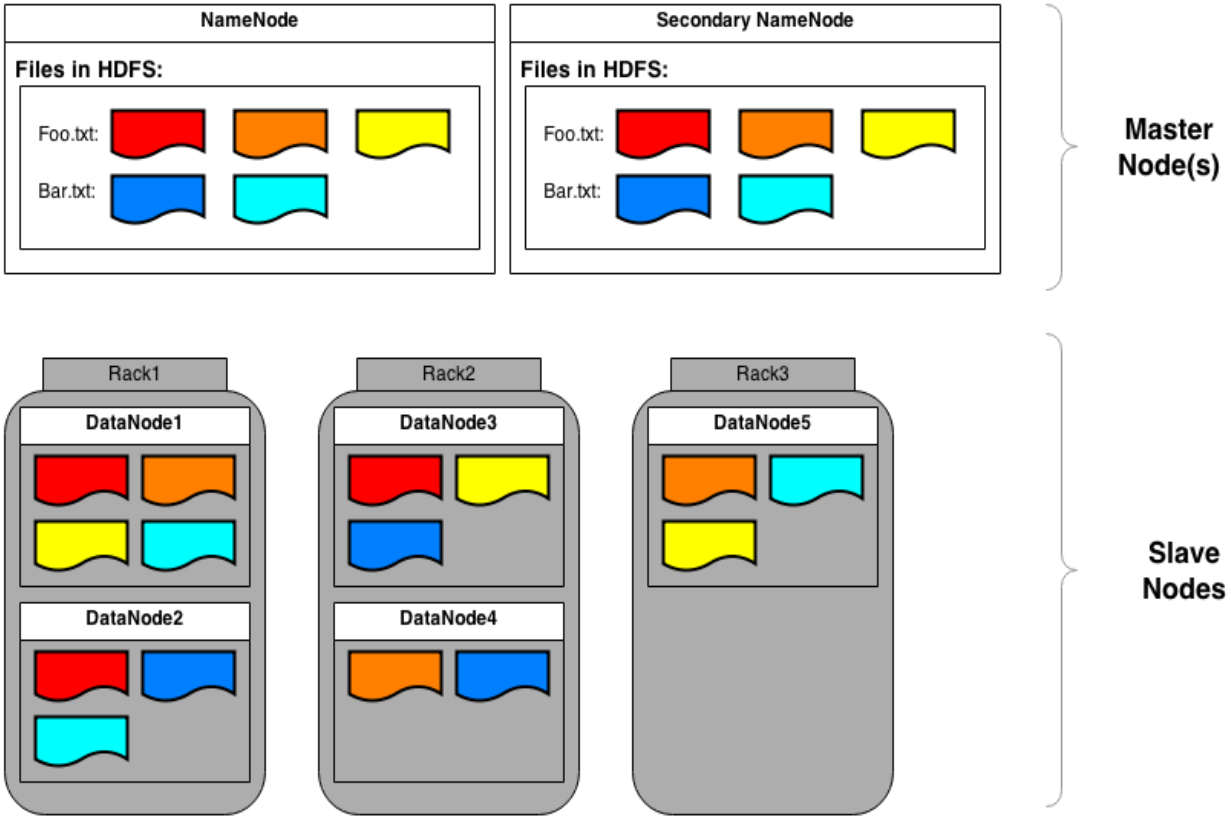
The main idea

'Time to transfer' > 'Time to seek'

Main idea


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

Files in HDFS



HDFS node types

- NameNode

HDFS node types

- NameNode
- DataNode

HDFS node types

- NameNode
- DataNode
- SecondaryNode (not for HA)

HDFS node types

- NameNode
- DataNode
- SecondaryNode
- StandbyNode

HDFS node types

- NameNode
- DataNode
- SecondaryNode
- StandbyNode
- Checkpoint Node

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

<epam>

MAPREDUCE THEORY

MapReduce in different languages

Language	Code sample

Java 8	`Integer totalAge = persons`
	`  .stream()`
	`  .map(Person::getAge)`
	`  .reduce( 0, (a, b) -> a + b);`

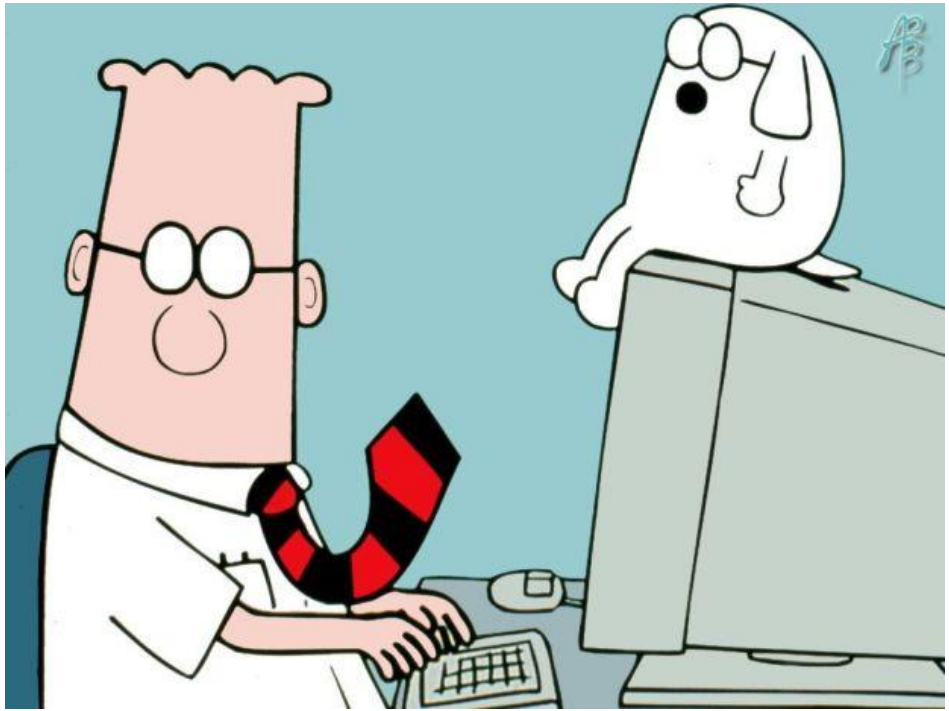
Scala	`val totalAge = persons`
	`  .map( (p: Person) => p.getAge )`
	`  .reduce( _ + _ )`

Python	`totalAge = reduce(`
	`  (lambda a, b: a + b),`
	`  list( map(lambda p: p.getAge, persons) )`
	`)`

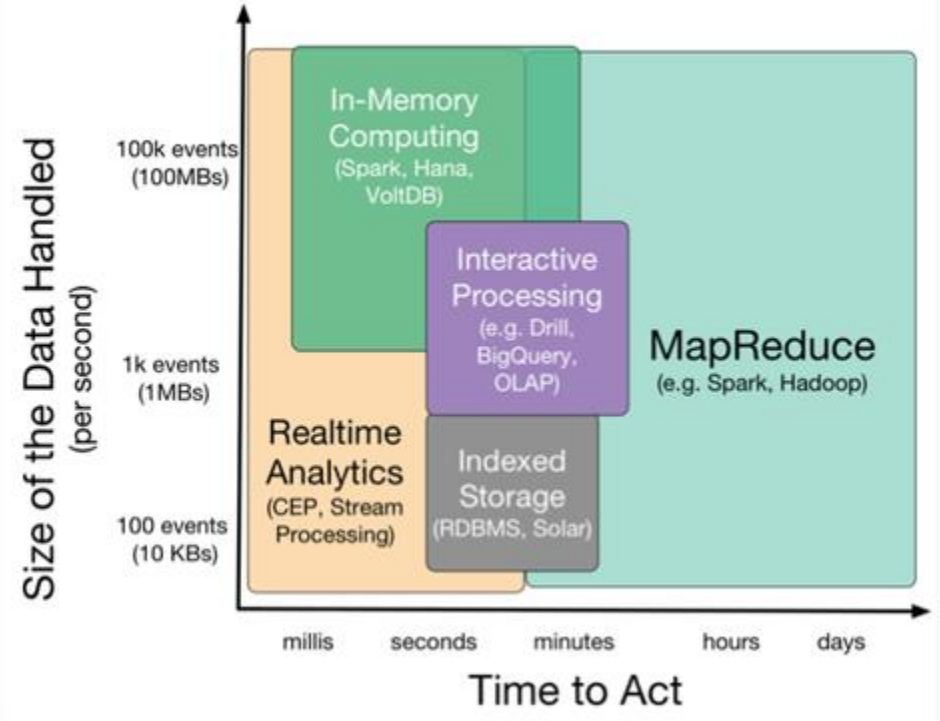
MR Typical Tasks

- WordCount
- Log handling
- Filtering
- Reporting Preparation

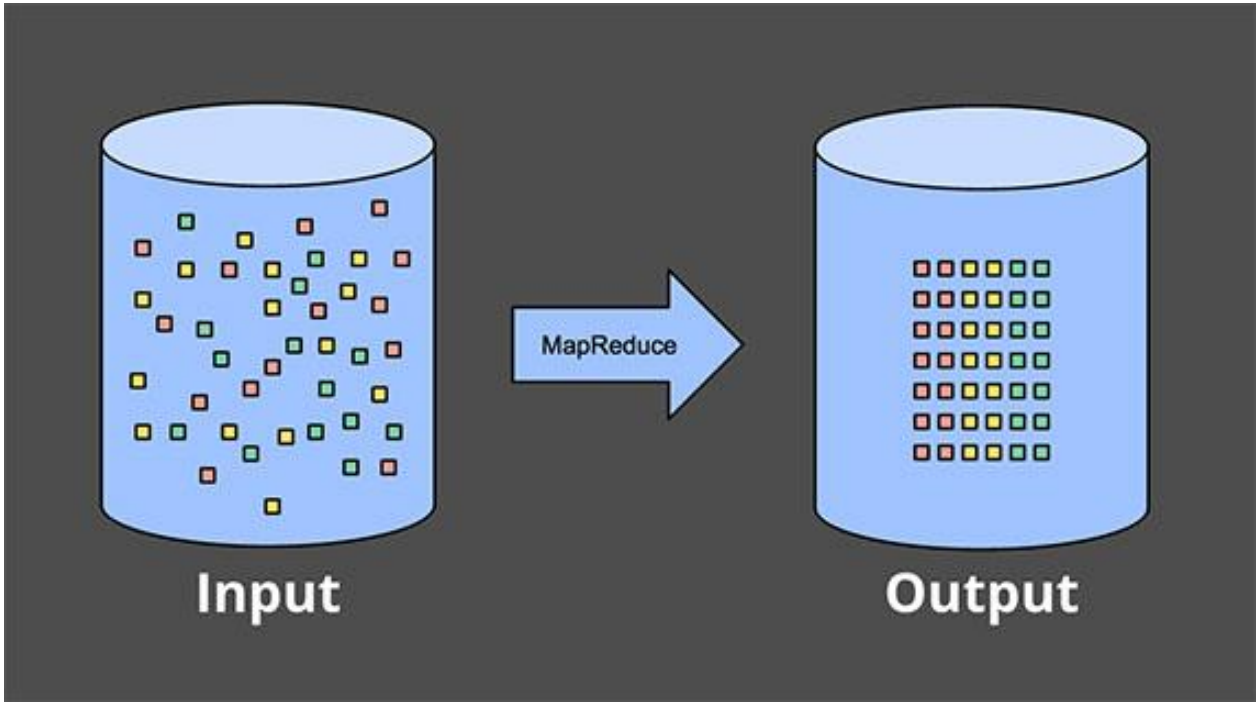
Why should we use MapReduce?



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

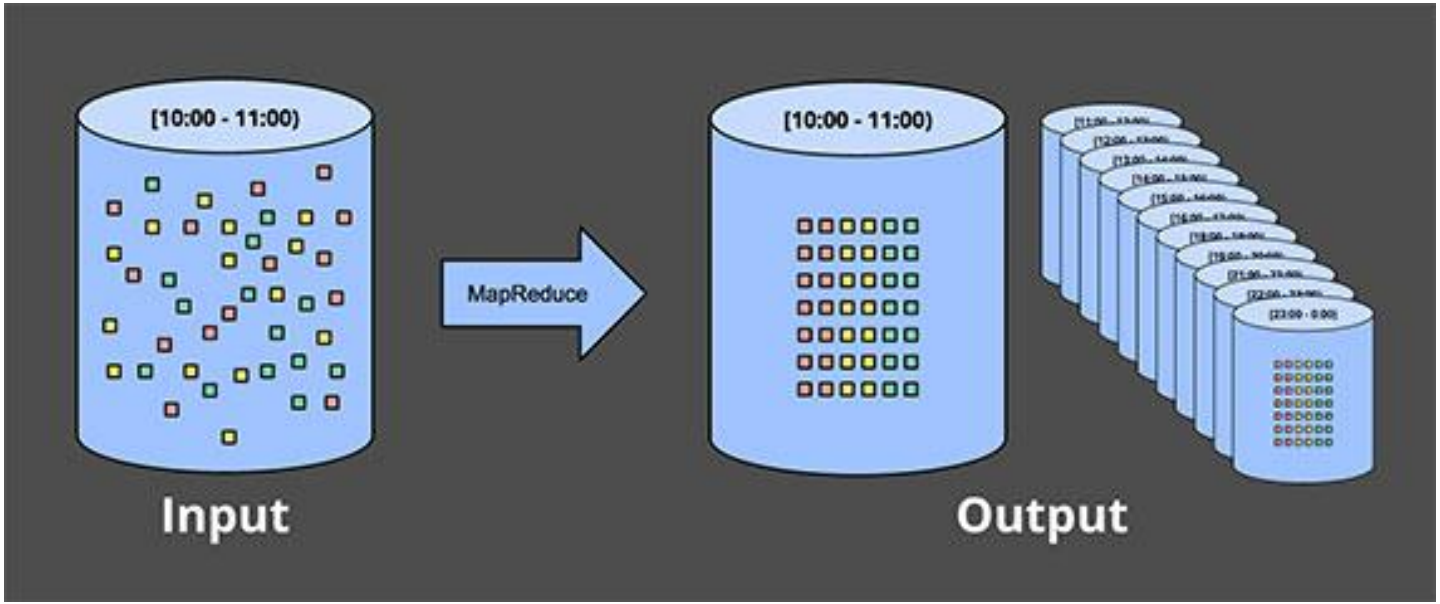


Classic Batch

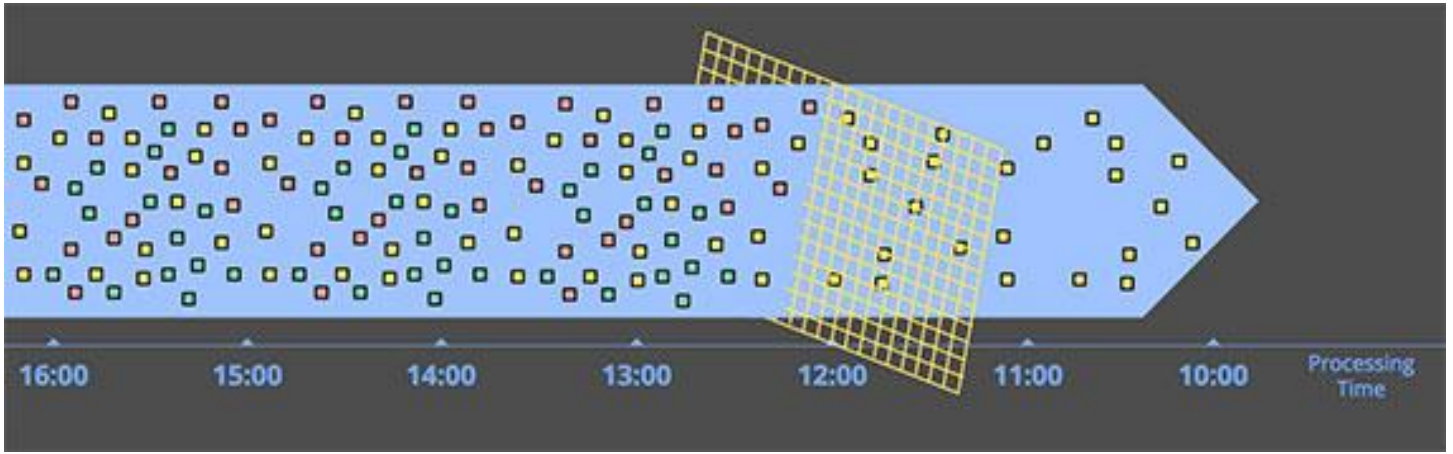


Do you like batches?

Fixed Windows



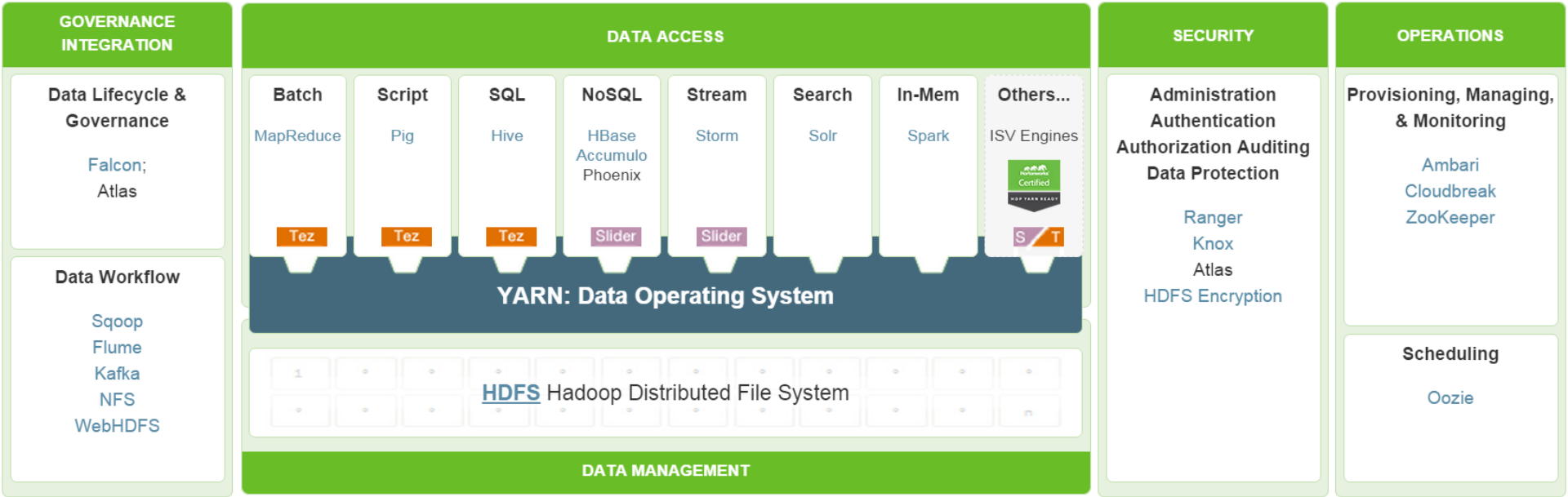
Filter all elements



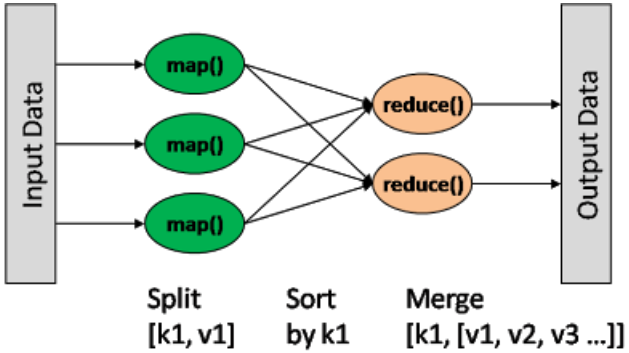
Pay attention!

Hadoop != MapReduce

Yet One YARN's lover



Think in Key-Value style



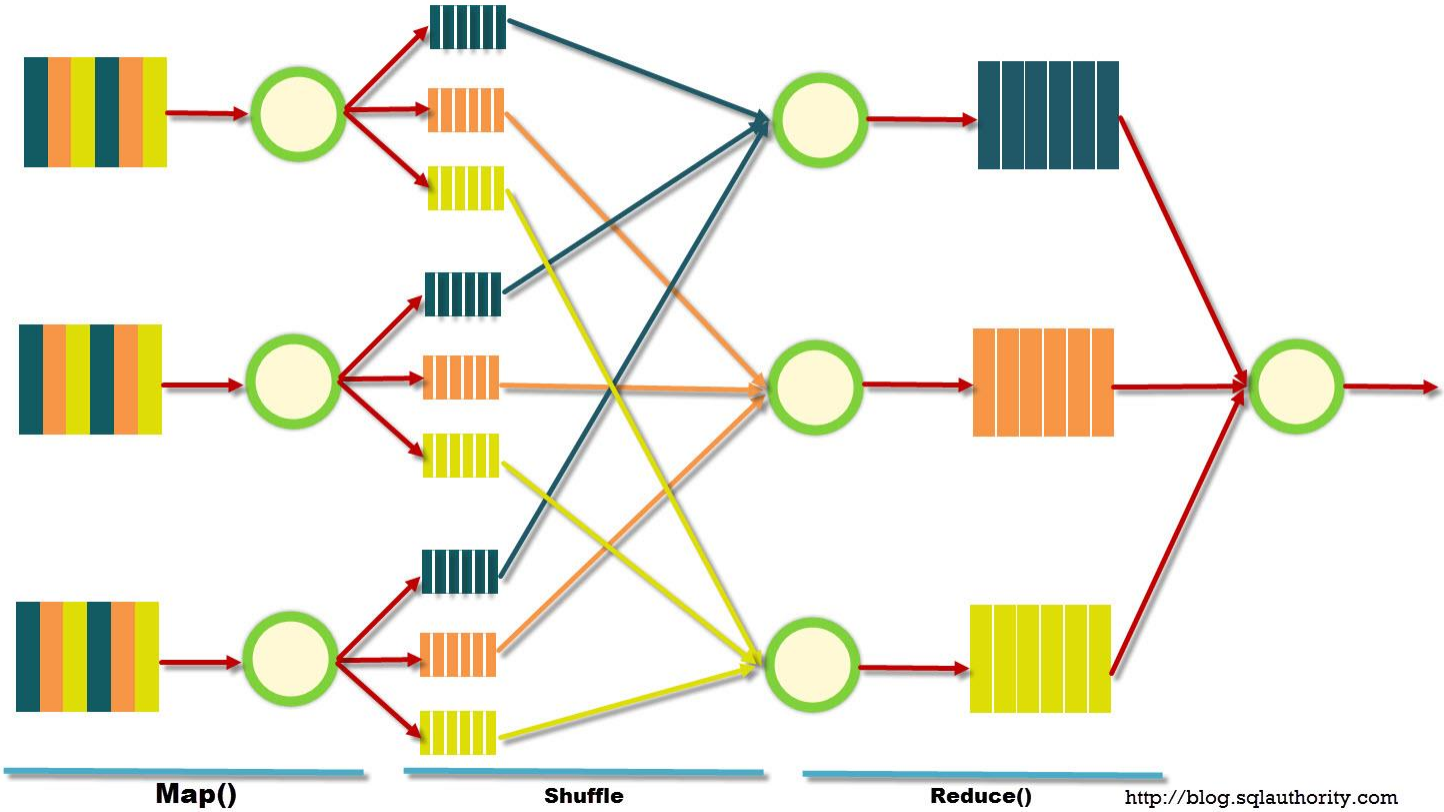
map (k1, v1) → list(k2, v2)

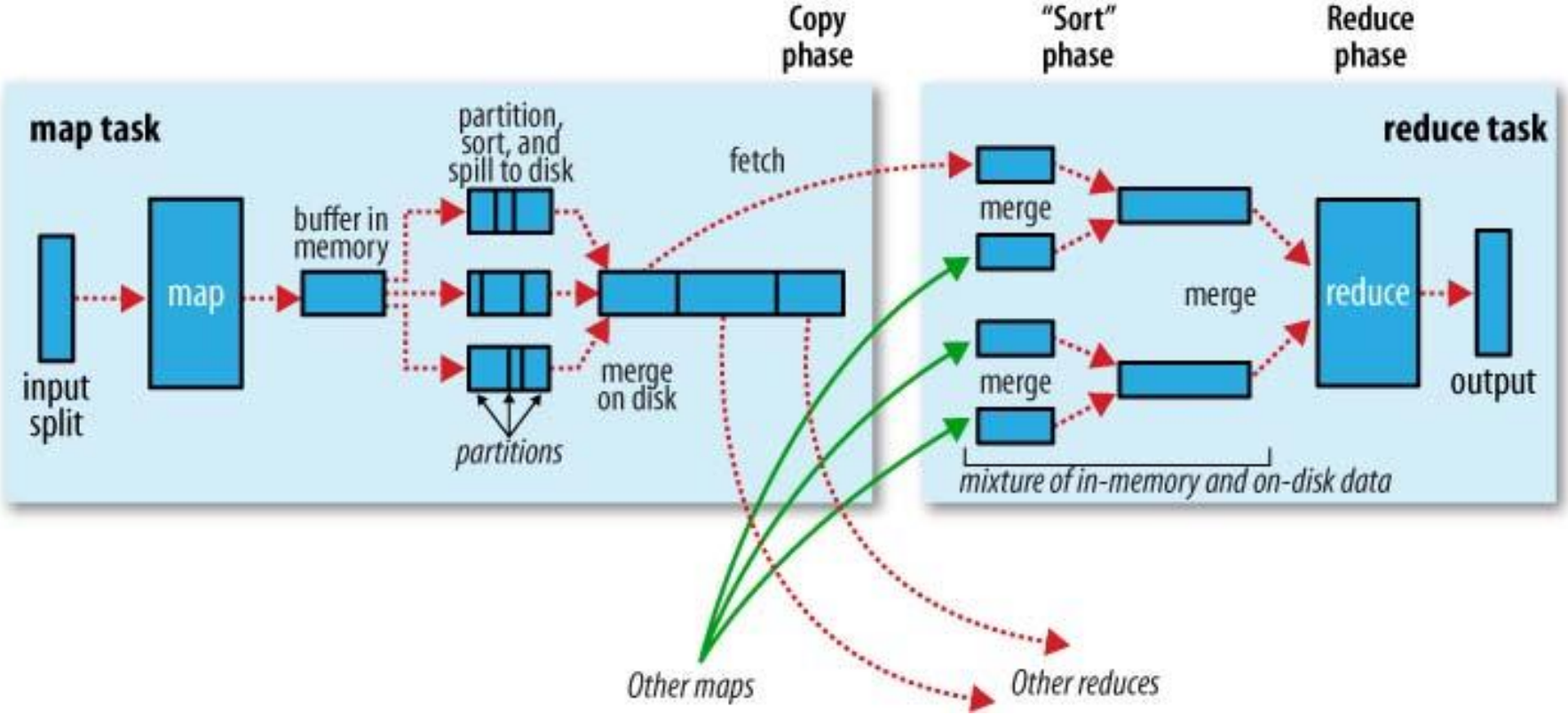
reduce (k2, list(v2*))→ list(k3, v3)

Main steps

- Map
- Shuffle
- Reduce

How MapReduce Works?




Copy phase
"Sort" phase
Reduce phase
map task
reduce task
input split
map
buffer in memory
partition, sort, and spill to disk
partitions
merge on disk
fetch
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

```
 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);

 }
}
```

Minimal Runner

```
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.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);
```

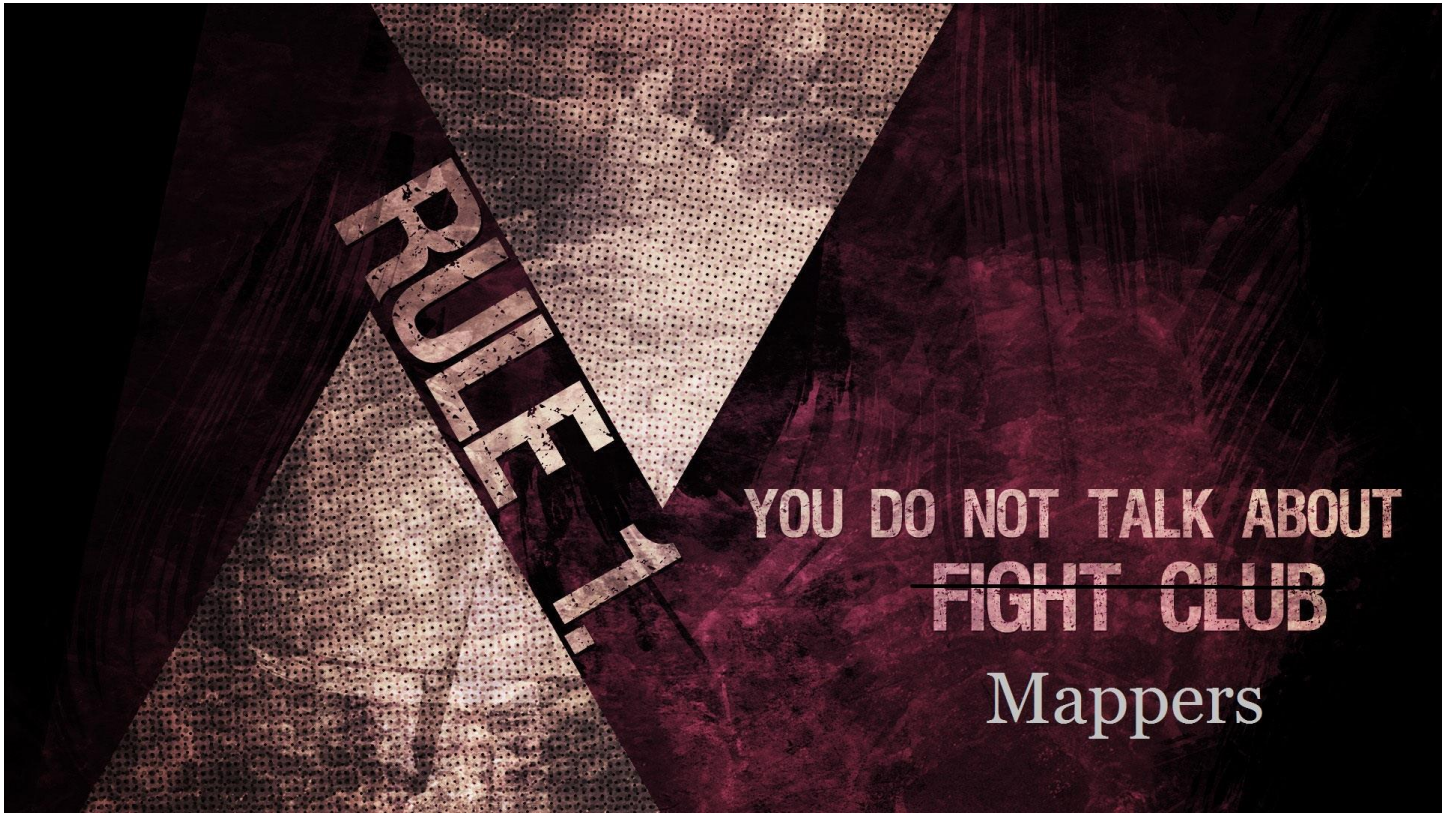
Job Config

Would you like to config in Java?

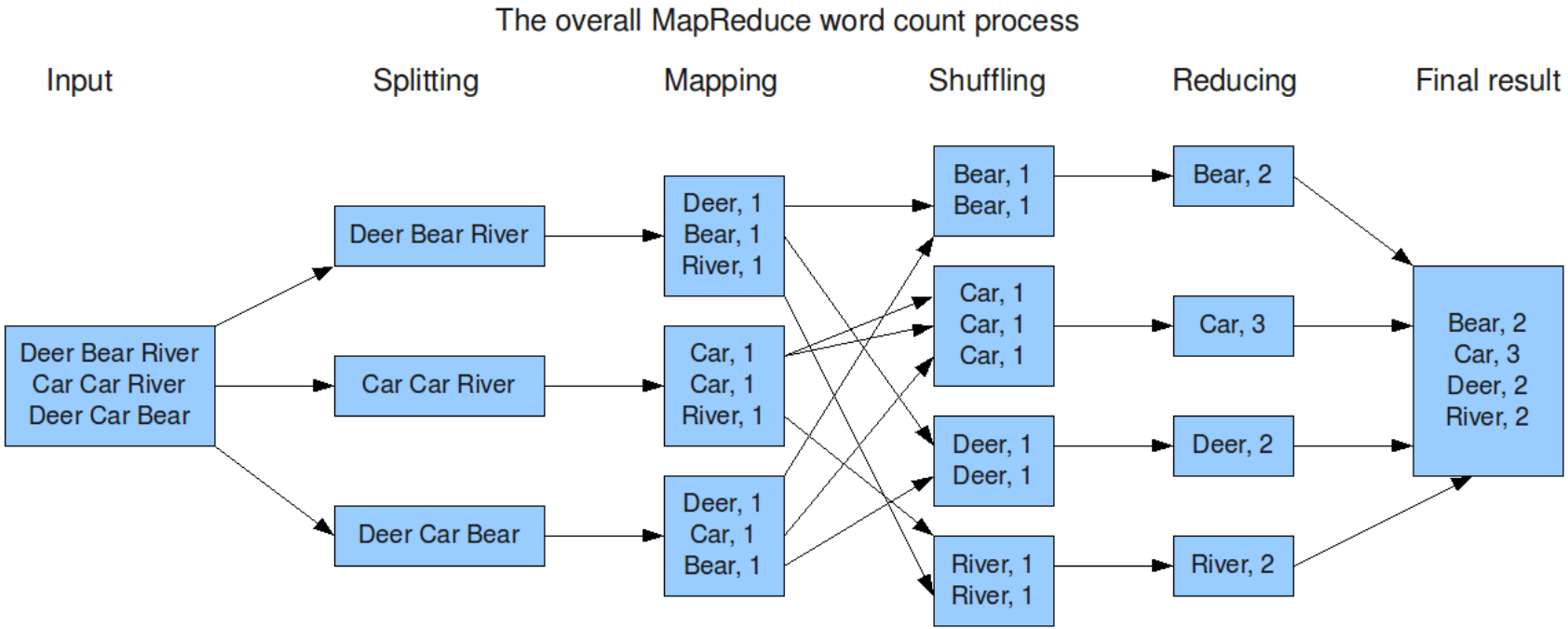
MAPREDUCE ADVANCED

Customize MapReduce!

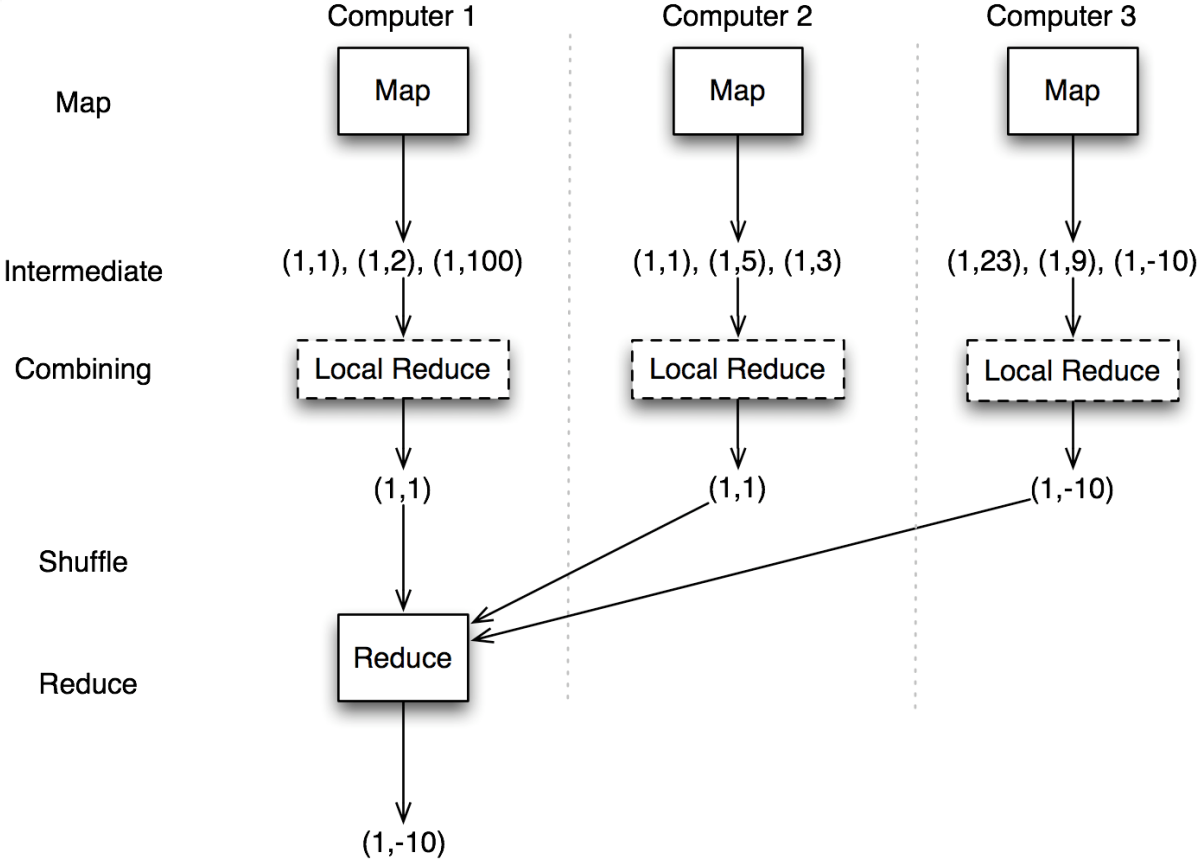
Hadoop club rules

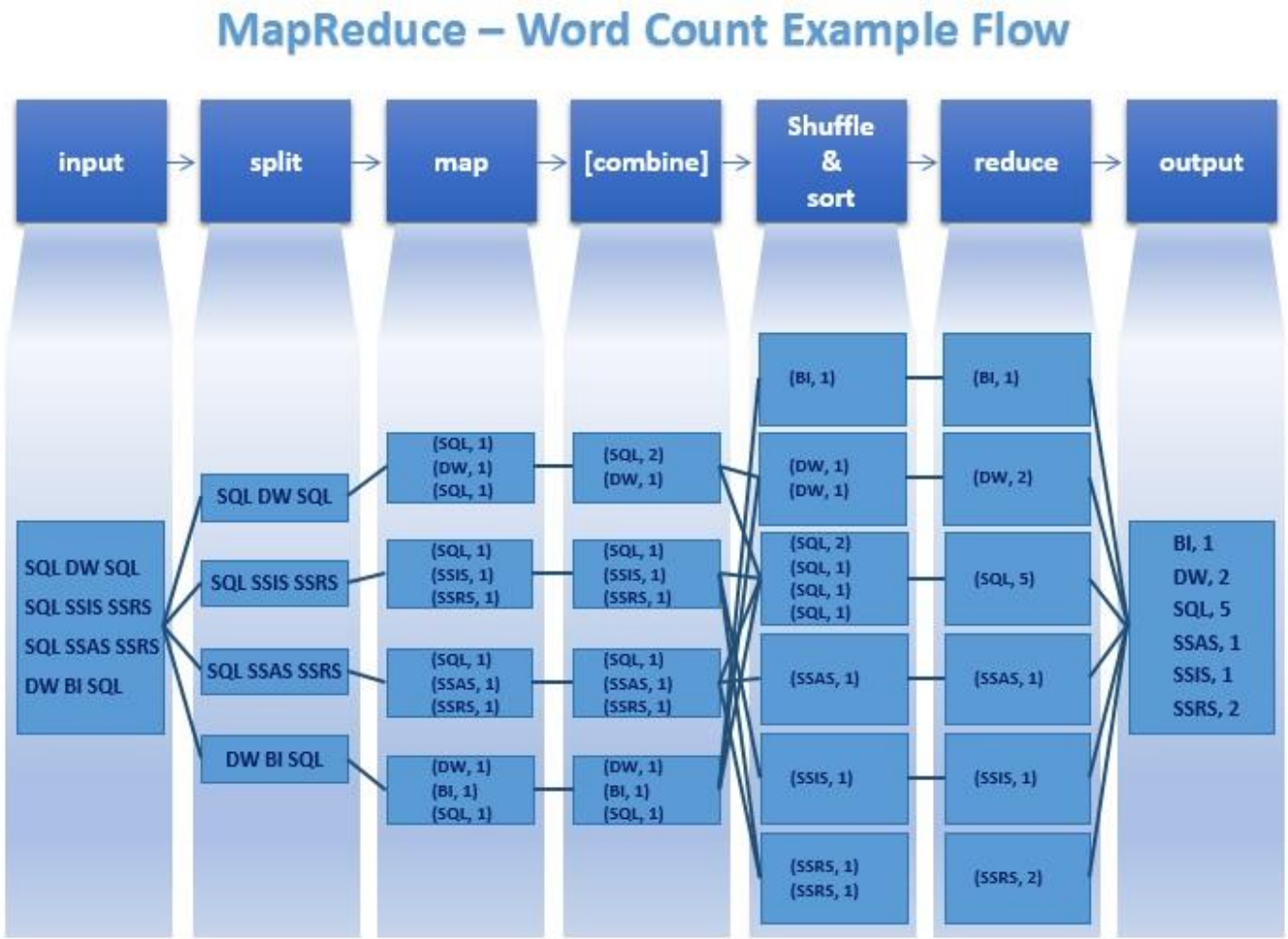


MapReduce for WordCount



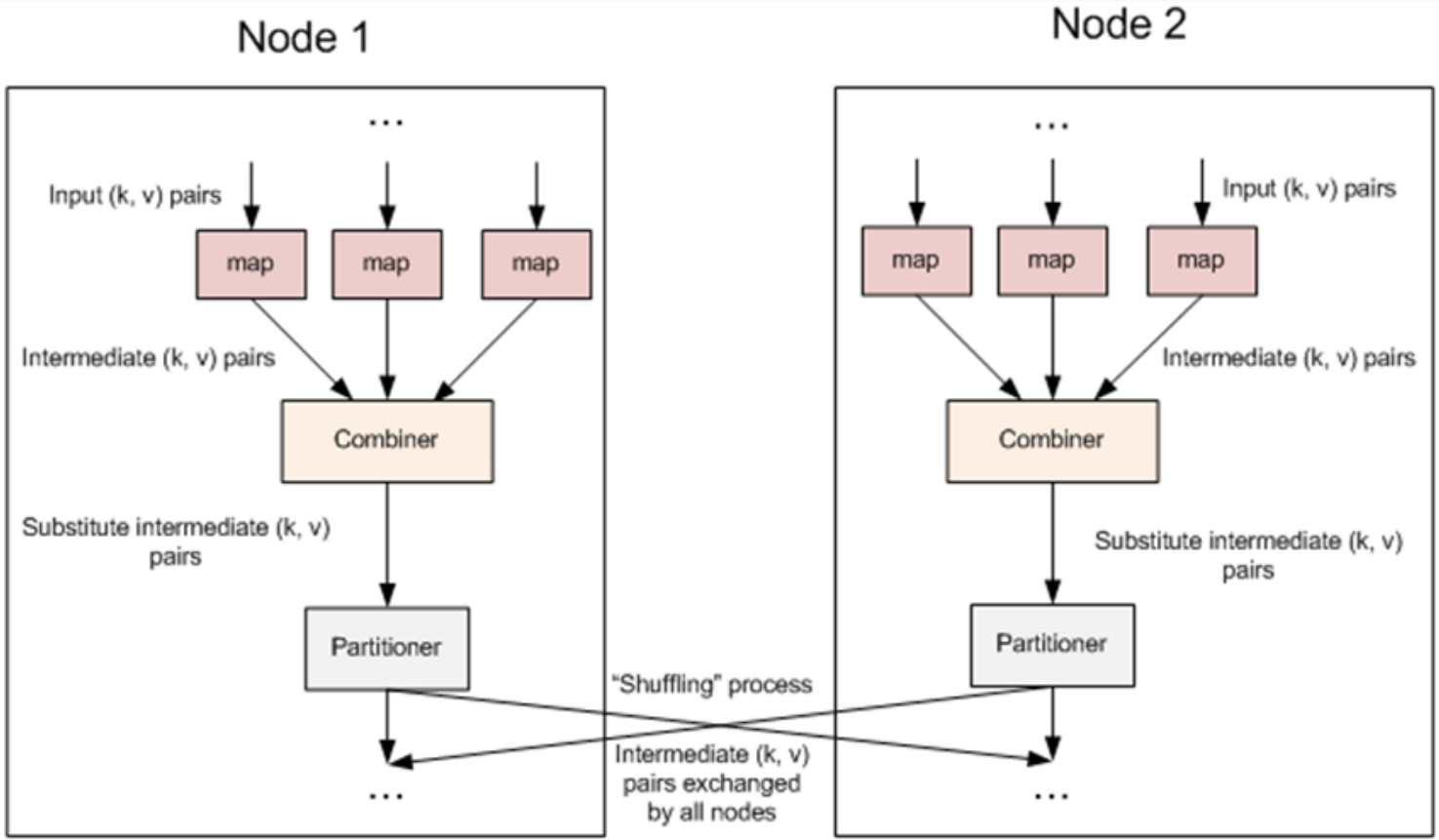
WordCount Combiner



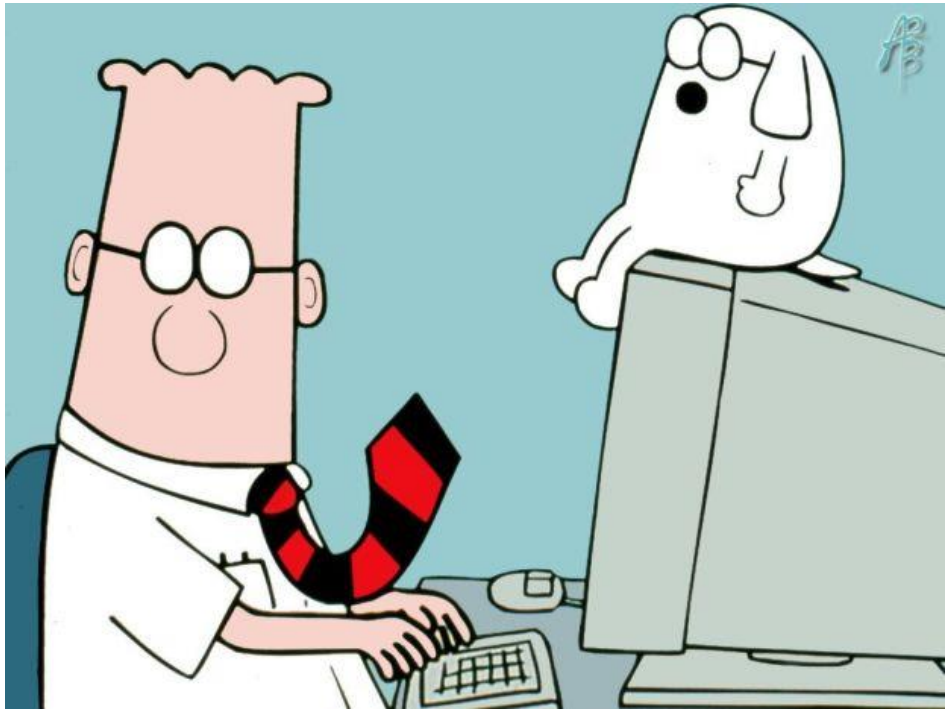

MapReduce – Word Count Example Flow
input
split
map
[combine]
Shuffle & sort
reduce
output
SQL DW SQL
SQL SSIS SSRS
SQL SSAS SSRS
DW BI SQL
SQL DW SQL
SQL SSIS SSRS
SQL SSAS SSRS
DW BI SQL
(SQL, 1) (DW, 1) (SQL, 1)
(SQL, 1) (SSIS, 1) (SSRS, 1)
(SQL, 1) (SSAS, 1) (SSRS, 1)
(DW, 1) (BI, 1) (SQL, 1)
(SQL,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



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



```
/**
 * 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
}
```

Setup

```
/**
 * 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

```
/**
 * 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);
  }
}
```

Run Mapper

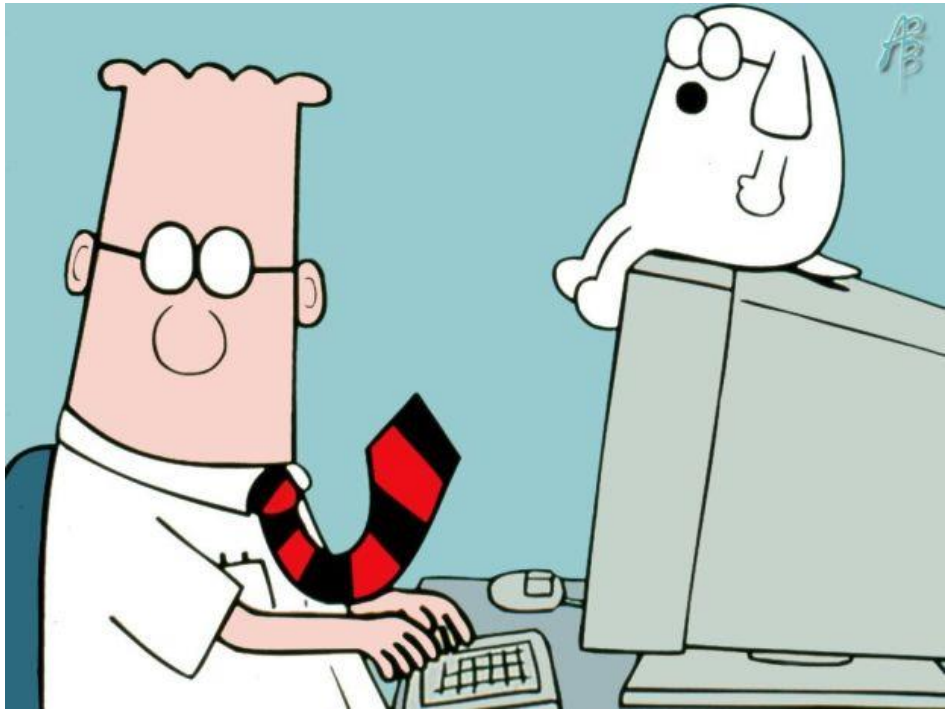
```
/**
 * Advanced application writers can use the
 * {@link #run(*.Reducer.Context)} method to
 * control how the reduce task works.
 */
public void run(Context context) throws IOException,
InterruptedException {
  setup(context);
  try {
    while (context.nextKey()) {
      reduce(context.getCurrentKey(), 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

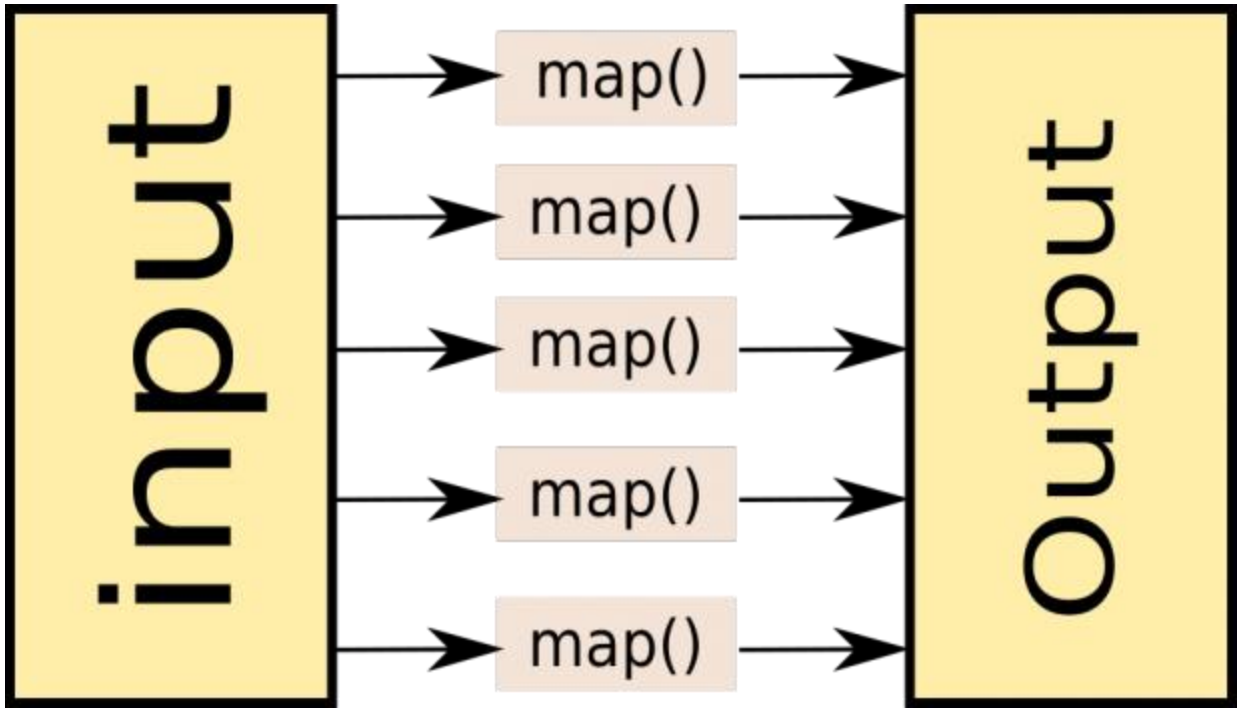
<epam>

REDUCER IS A PARTICIO OF PROBLEMO

Could we skip shuffle step?



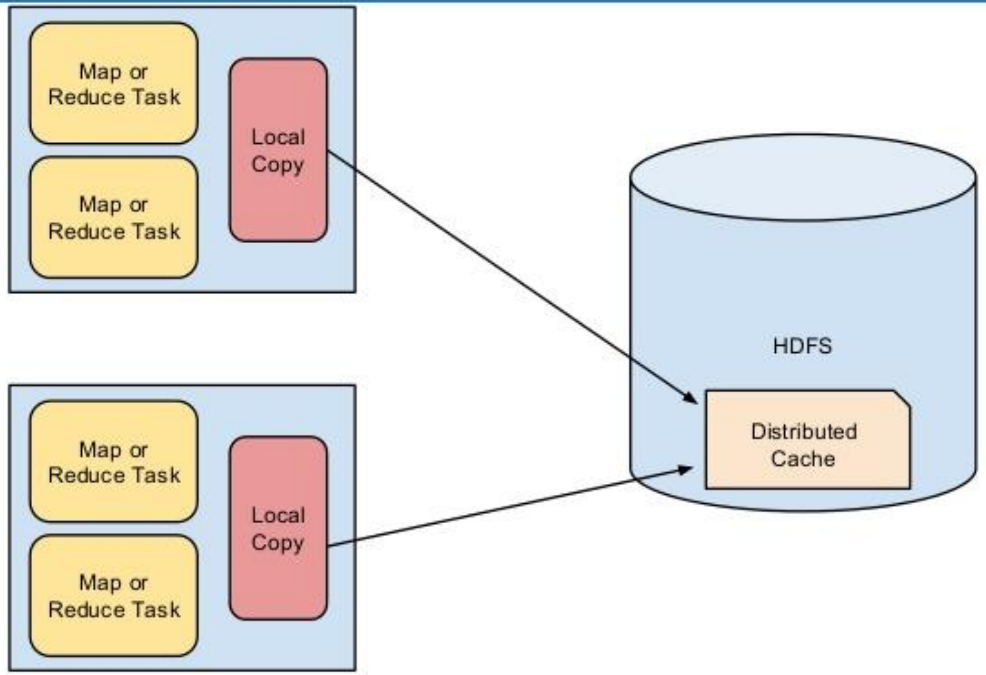
Map-Only ‘MapReduce’ Jobs



<epam>

* LEVEL

Distributed Cache



The main concept of Distributed Cache

Share common data☺

How to implement?

For ToolRunner in command line with ***-files*** option

- ***- archives***
- ***- libjars***

In code ***Job.addCacheFile(URI uriToFileInHdfs);***

- ***Job.addCacheArchive(URI uriToArchiveFileInHdfs);***
- ***Job.addArchiveToClassPath(Path pathToJarInHdfs);***

How to get data from Cache?

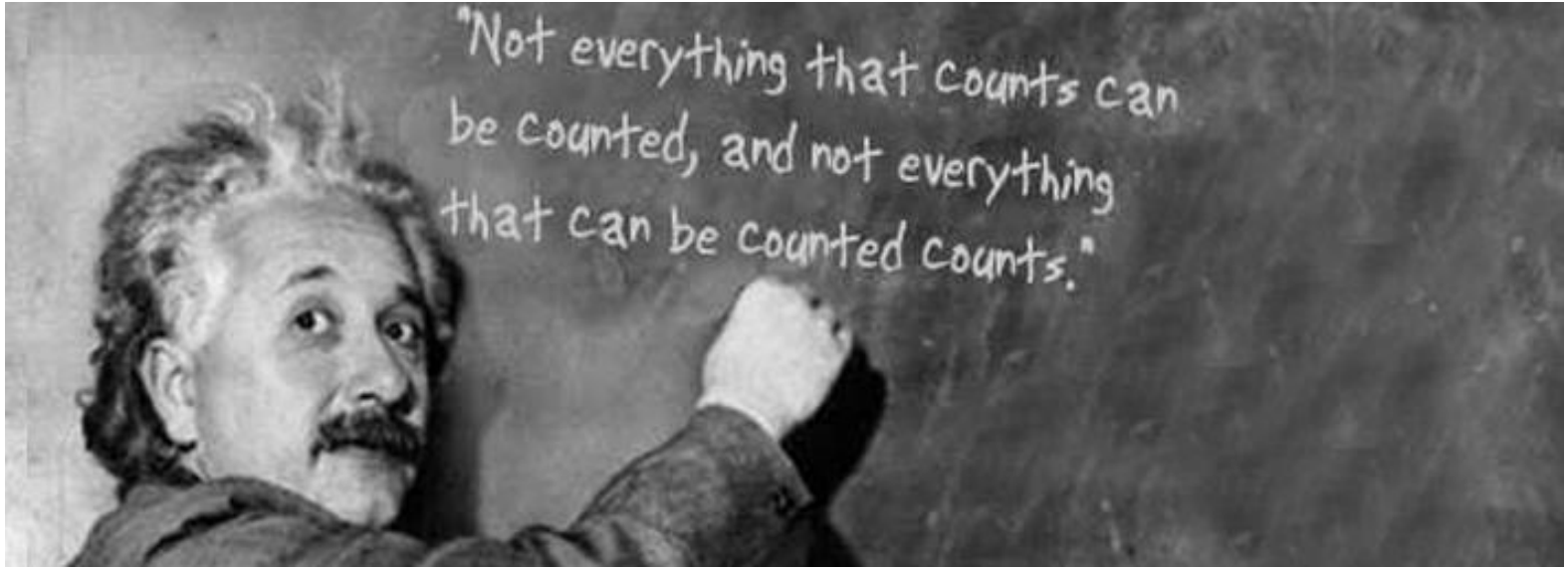
To list the content of Distributed Cache:

```
Path[] pathes = JobContext.getLocalCacheFiles();
```

Then you have to check this URI to pick up your file

```
for(Path u : pathes) {
   if(u.getName().toUpperCase.contains("TARGET")) {
      ....
  }
}
```

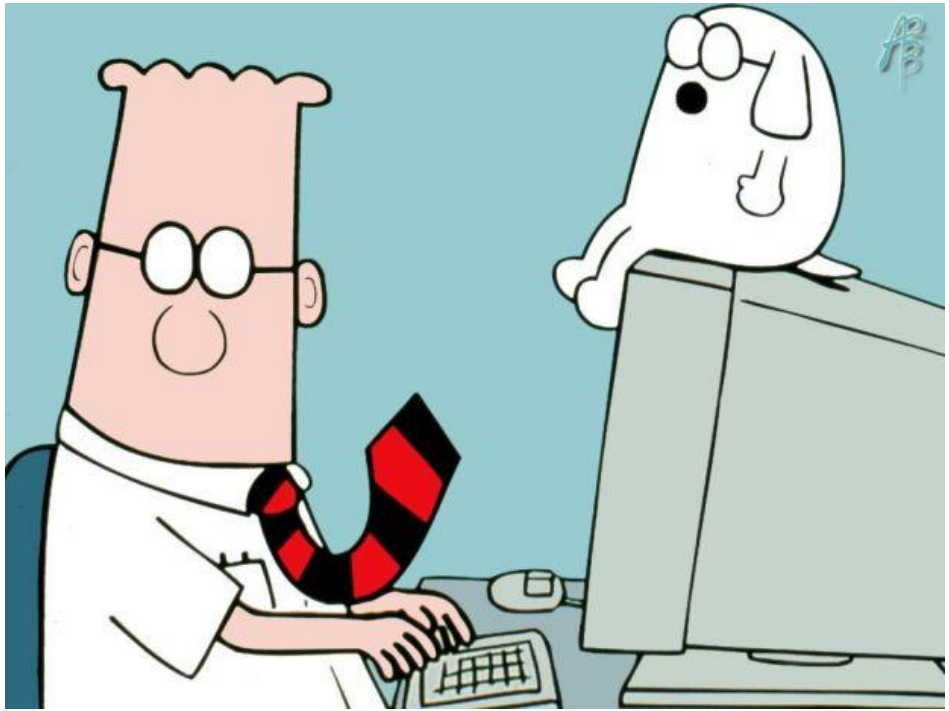
Counters usage



```
14/11/24 09:15:50 INFO mapreduce.Job: Counters: 32
        File System Counters
                FILE: Number of bytes read=0
                FILE: Number of bytes written=100843
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=350
                HDFS: Number of bytes written=189
                HDFS: Number of read operations=5
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=17661
                Total time spent by all reduces in occupied slots (ms)=0
                Total time spent by all map tasks (ms)=5887
                Total vcore-seconds taken by all map tasks=5887
                Total megabyte-seconds taken by all map tasks=4521216
        Map-Reduce Framework
                Map input records=2
                Map output records=30
                Input split bytes=118
                Spilled Records=0
                Failed Shuffles=0
                Merged Map outputs=0
                GC time elapsed (ms)=64
                CPU time spent (ms)=1210
                Physical memory (bytes) snapshot=184856576
                Virtual memory (bytes) snapshot=1254182912
                Total committed heap usage (bytes)=136478656
        com.hadoop.skipper.StopWordSkipper$COUNTERS
                GOODWORDS=30
                STOPWORDS=12
        File Input Format Counters
                Bytes Read=232
        File Output Format Counters
                Bytes Written=189
```

Counters

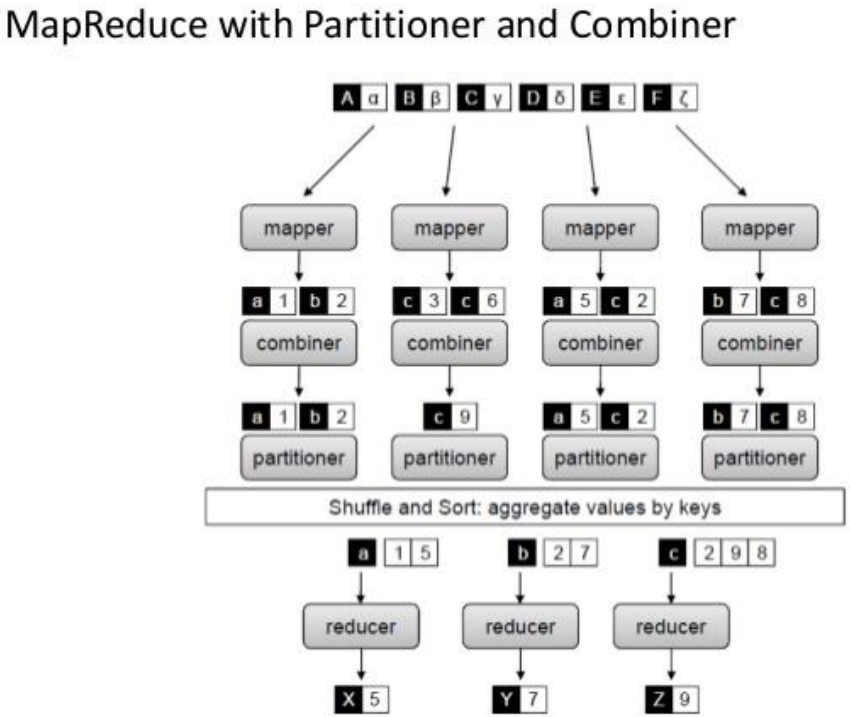
May I customize DataFlow before shuffling?



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;

    }
}
```

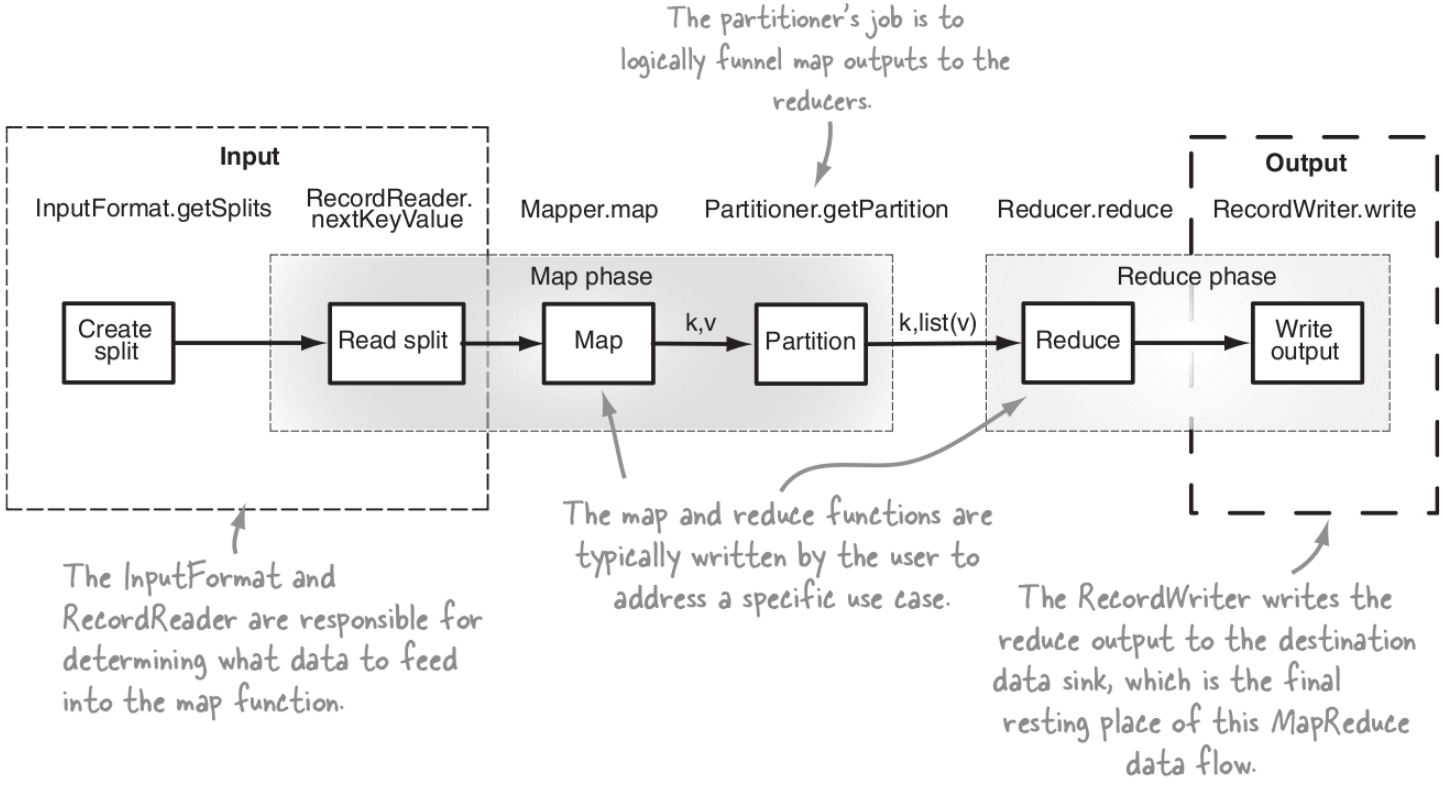
Partitioner's Role in Shuffle and Sort



How to customize?

- You want to send by other rule (hash to value)
- Need know more about secondary sort
- Extend Partitioner abstract class
- getPartition(key, value, number of Reducers)
- Return int [0, number of Reducers – 1]
- Don’t forget *job.setPartitionerClass(MyPartitioner.class);*

Full power

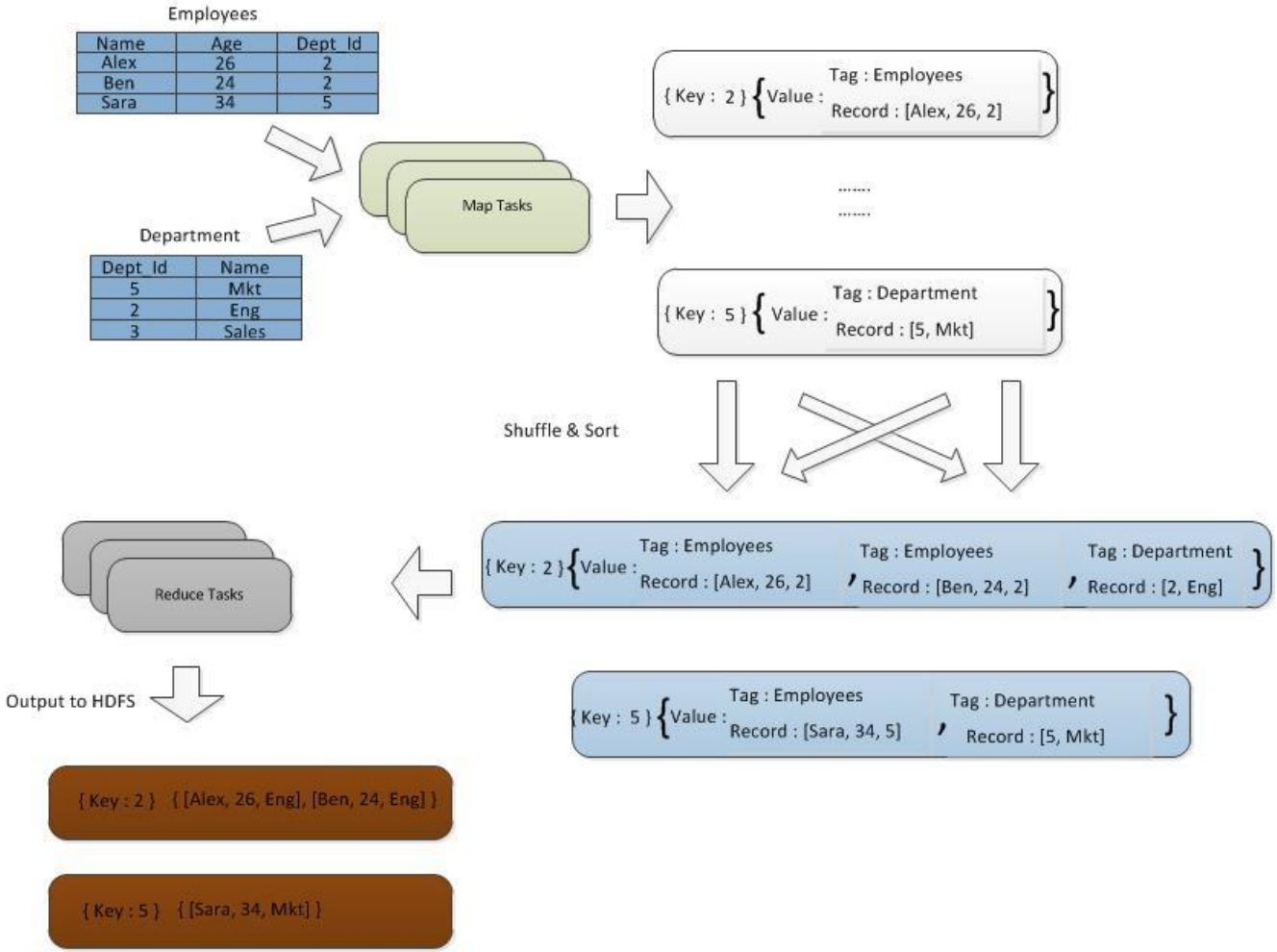

The partitioner's job is to logically funnel map outputs to the reducers.
Input
InputFormat.getSplits
RecordReader.nextKeyValue
Mapper.map
Partitioner.getPartition
Reducer.reduce
Output
RecordWriter.write
Map phase
Reduce phase
Create split
Read split
Map
k,v
Partition
k,list(v)
Reduce
Write output
The 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.

<epam>

JOINS

First idea

Let’s copy small table on nodes

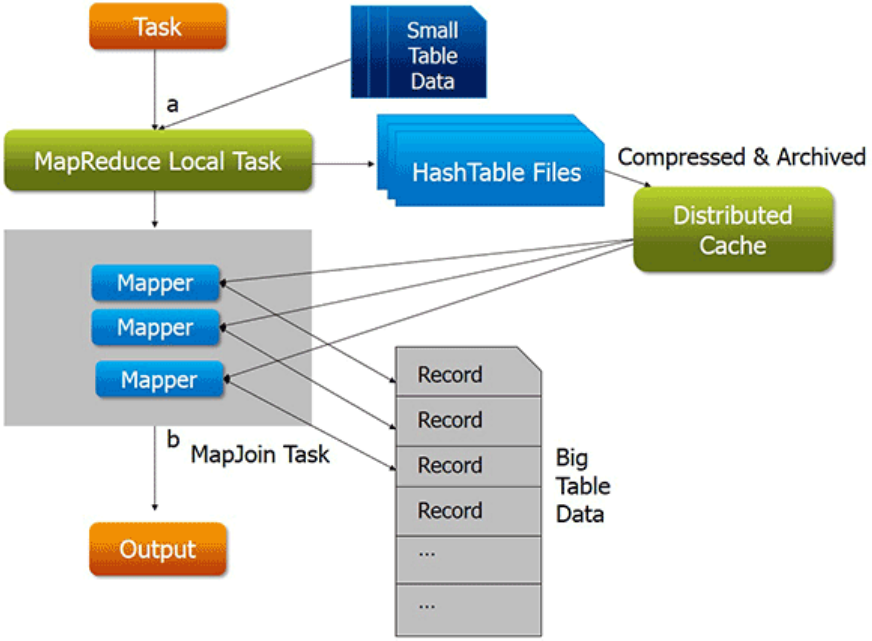

Employees
Name Age Dept_Id
Alex 26 2
Ben 24 2
Sara 34 5
Department
Dept_Id Name
5 Mkt
2 Eng
3 Sales
Map Tasks
{ Key : 2 } { Value : Tag : Employees Record : [Alex, 26, 2] }
.....
.....
{ Key : 5 } { Value : Tag : Department Record : [5, Mkt] }
Shuffle & Sort
{ Key : 2 } { Value : Tag : Employees Record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

Reduce JOIN

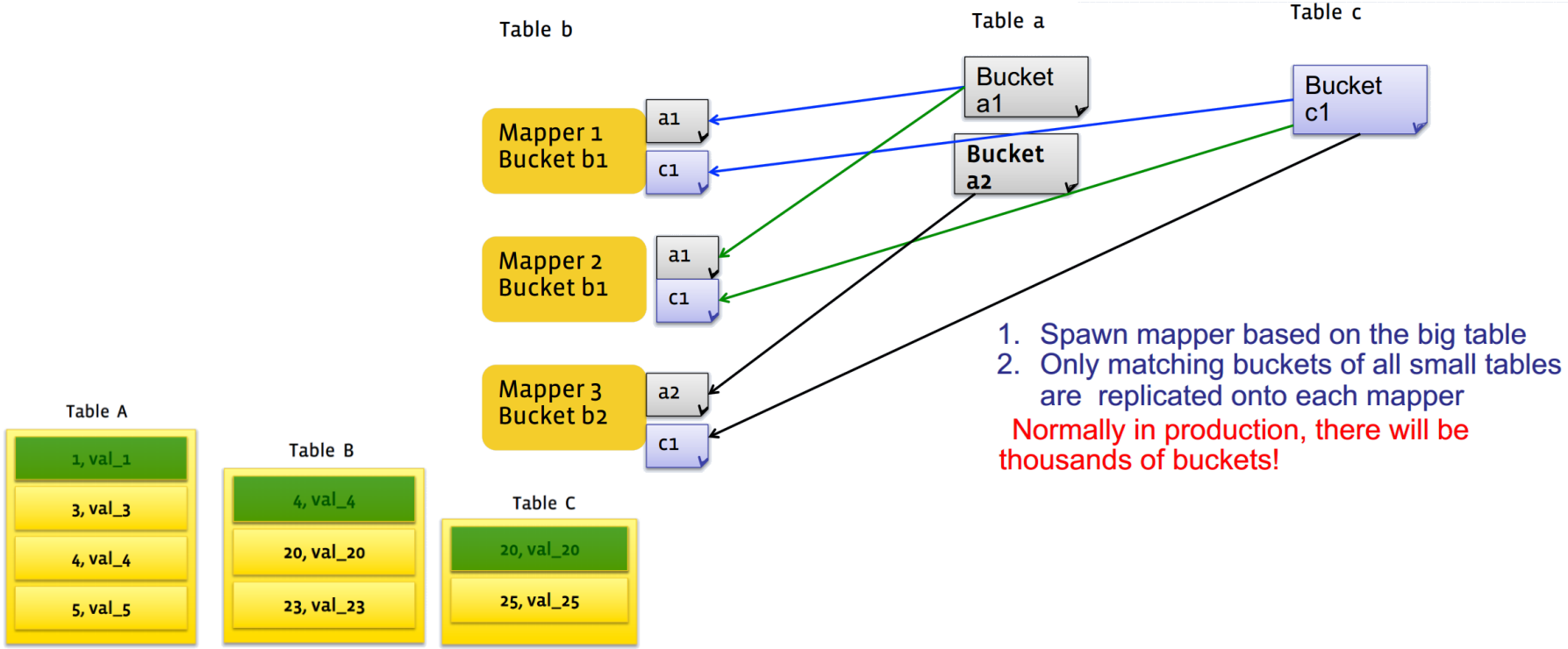
The main concept of JOIN

Let’s skip Reduce + Shuffle

Map-side join



Map-side join for large datasets



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
```

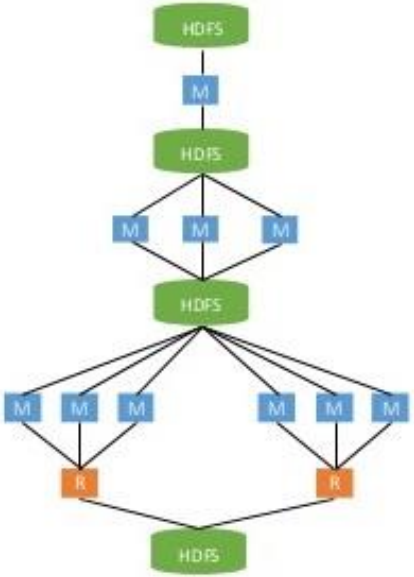
The main JOIN idea fro large tables

Redis or Memcache cluster as Distributed Cache

PERFORMANCE

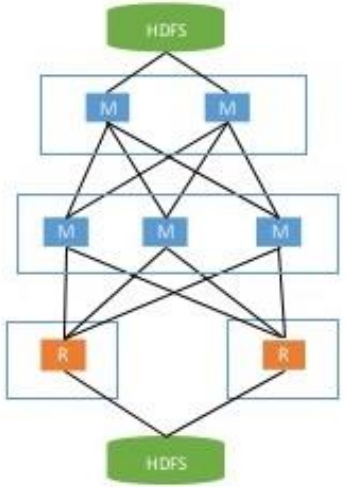
Dig to DAG (Tez means “speed” in Hindi)

MapReduce



DAG engine

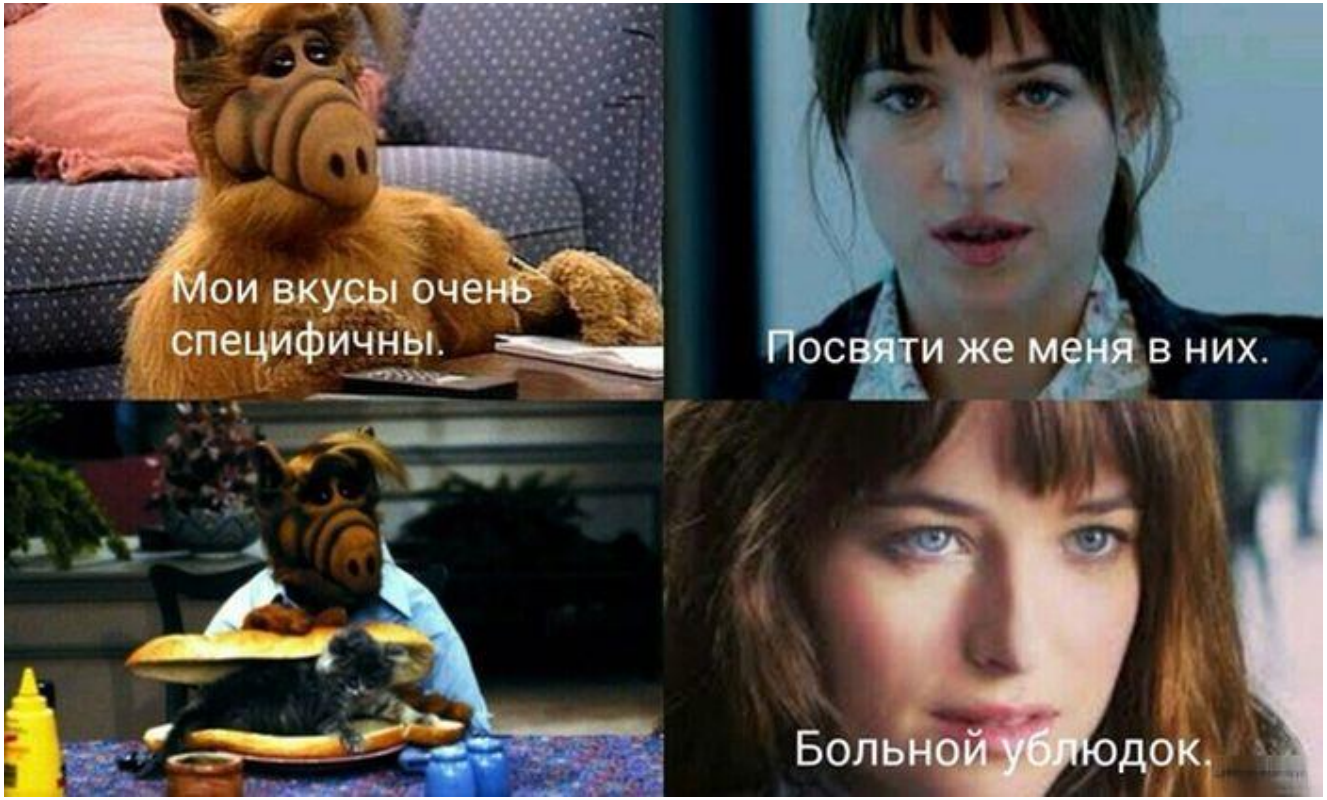
(Tez / Spark)



No intermediate DFS reads/writes!

The main concept of DAG

Decompose you problem, build DAG, skip HDFS



Let’s run on JVM!

<epam>

JVM SETTINGS

Typical mistakes

- Collections are stored and sorted in memory
- Sorting all data in memory
- Logging each input key-value pairs
- JARs hell
- Skew input: all records go to one reducer

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

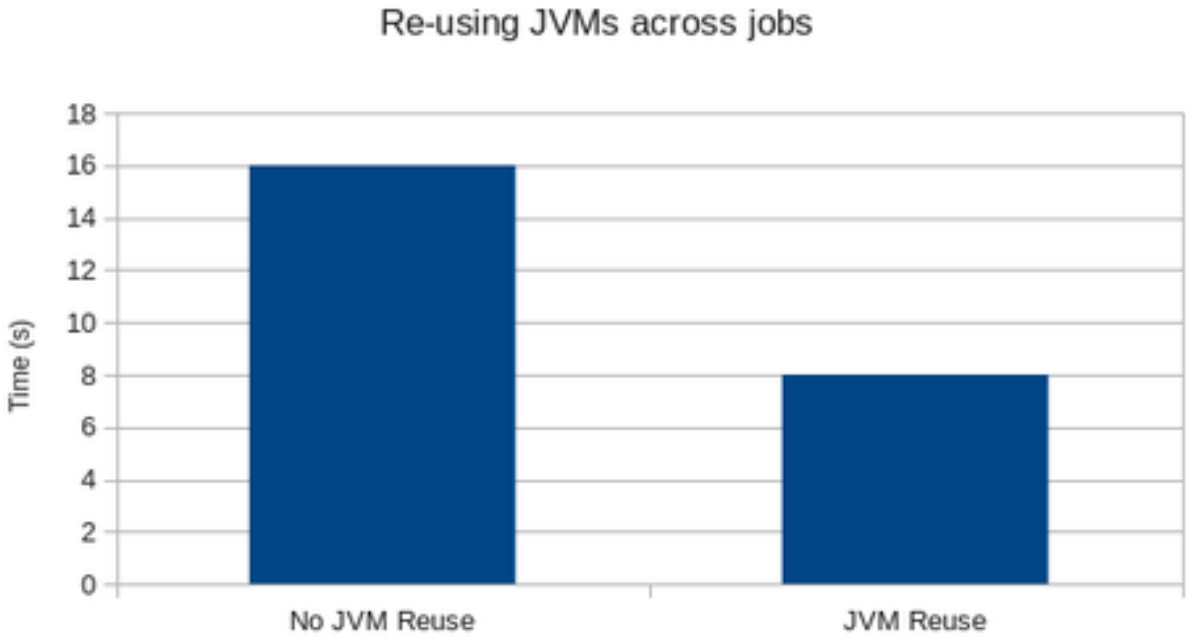
JVM tuning

- ***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

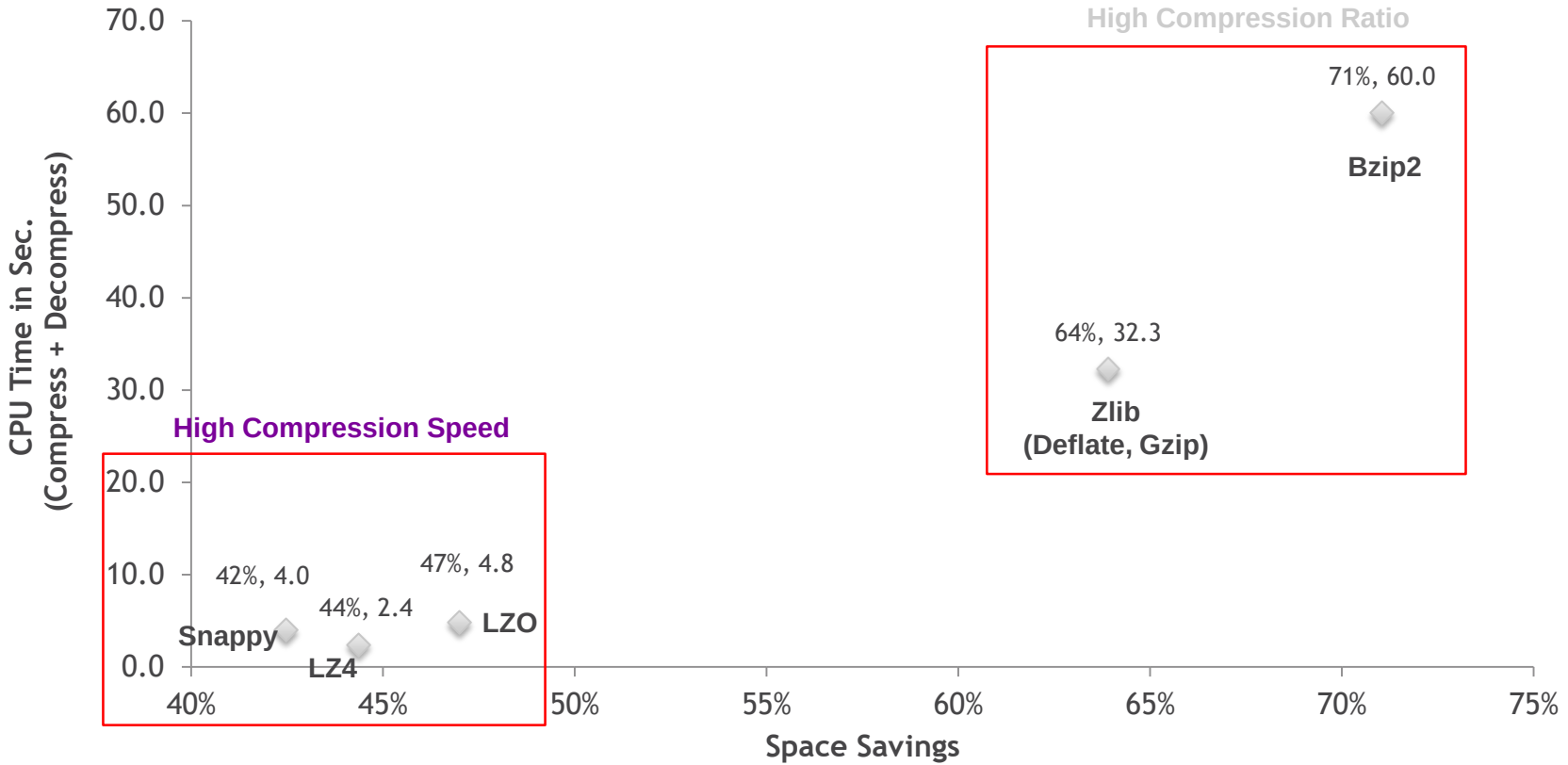


FILE FORMATS

Input/Output Formats

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

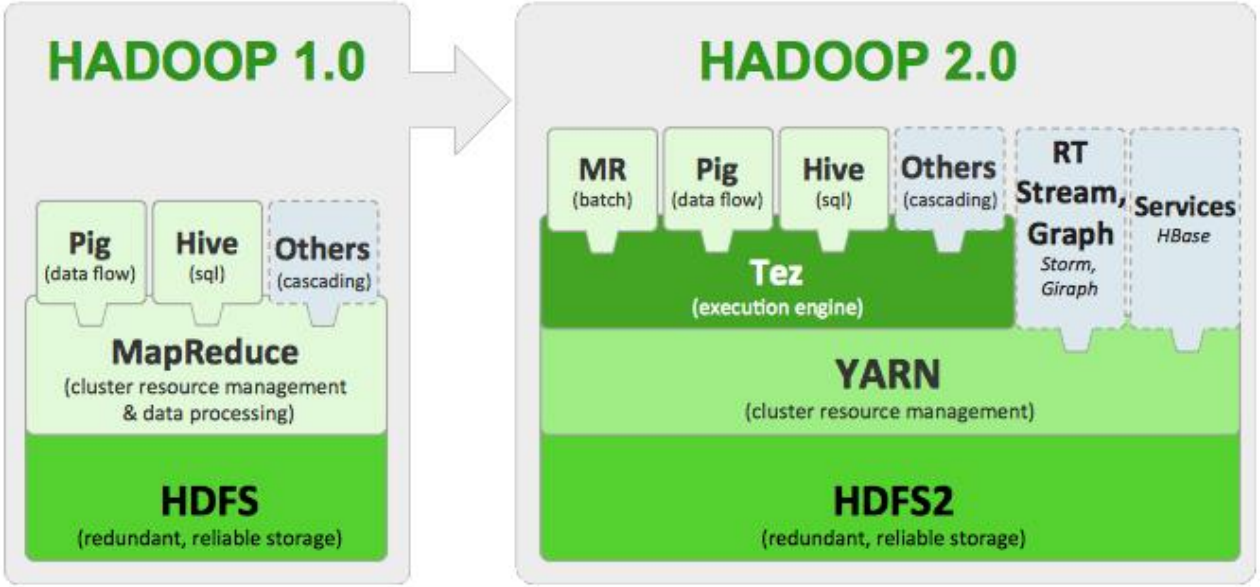
Codec Performance on the Wikipedia Text Corpus



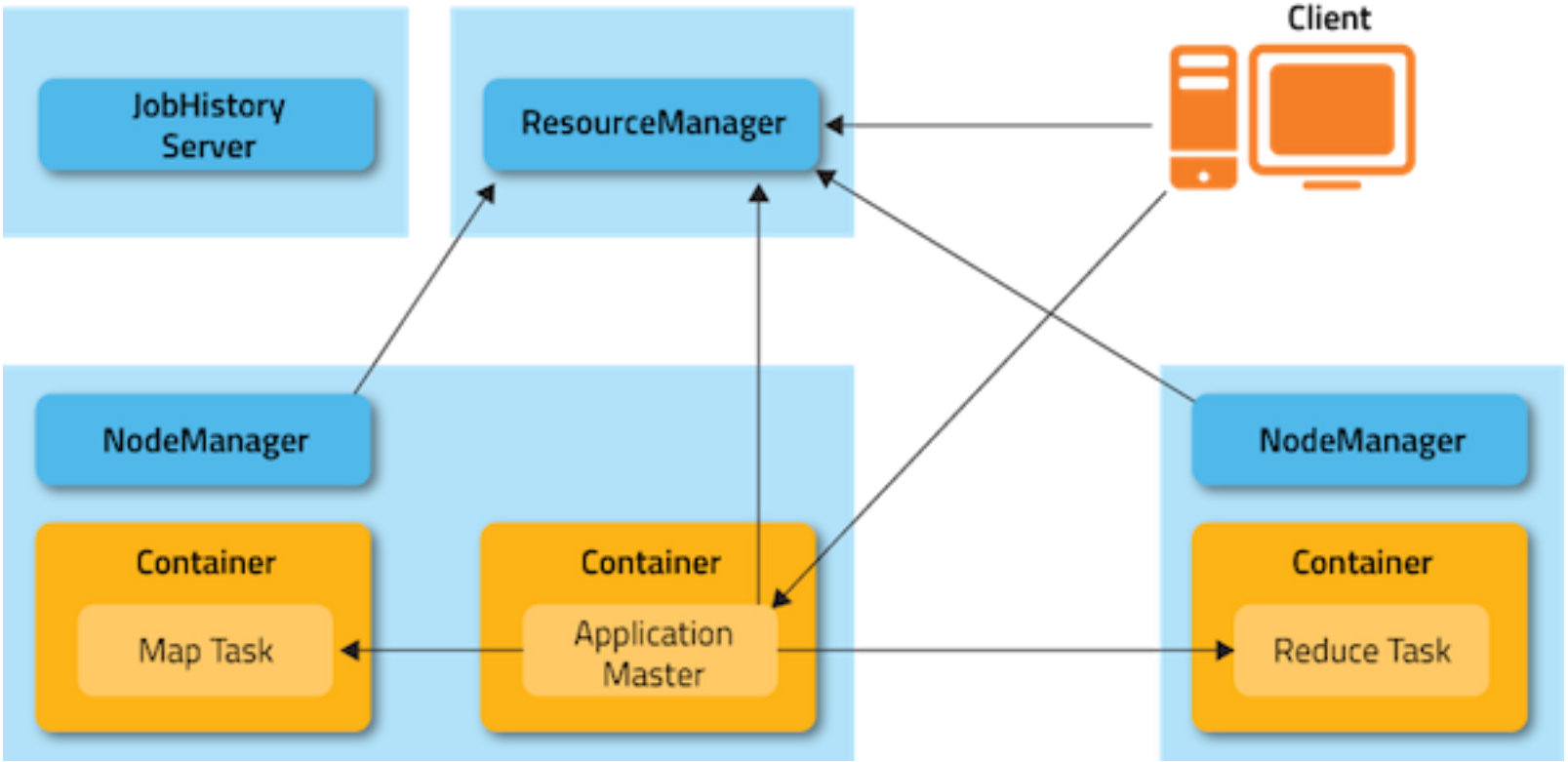
<epam>

YARN

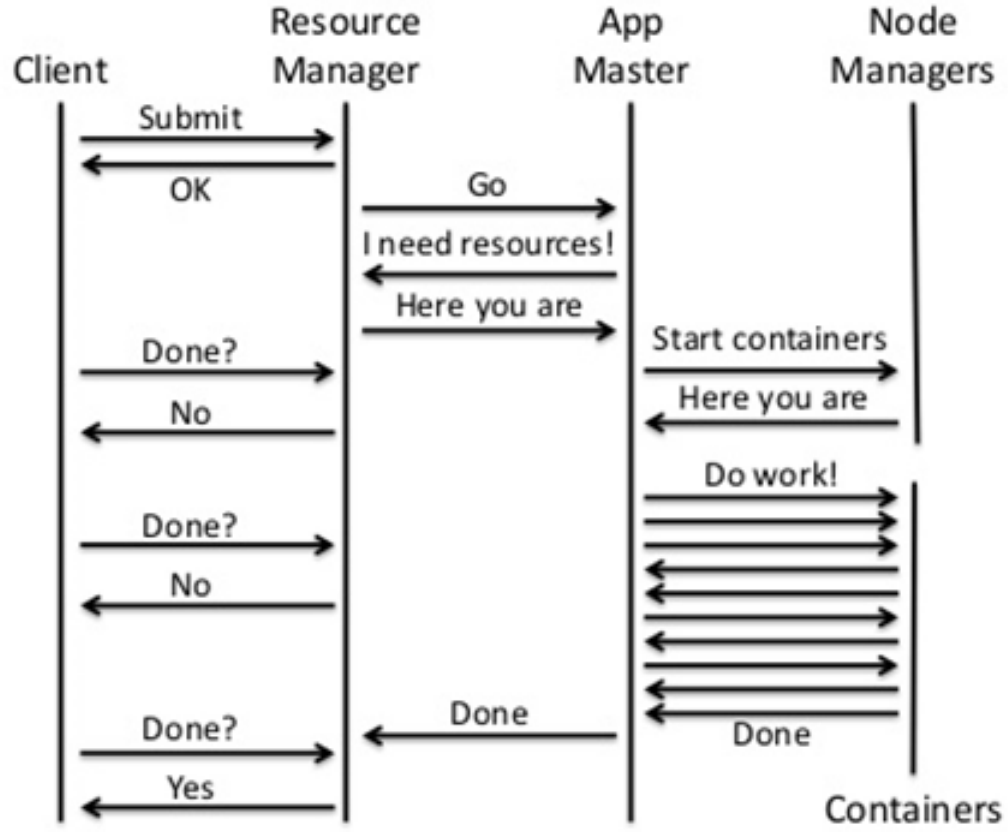
From Hadoop 1 to Hadoop 2



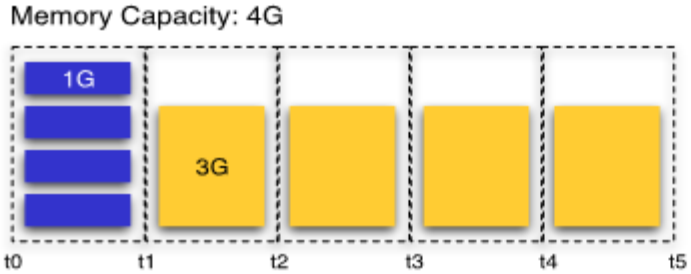
Daemons in YARN



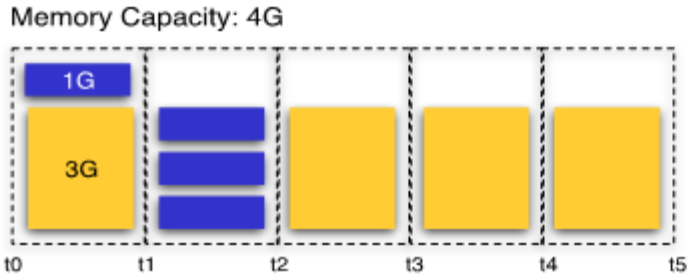
Full lifecycle



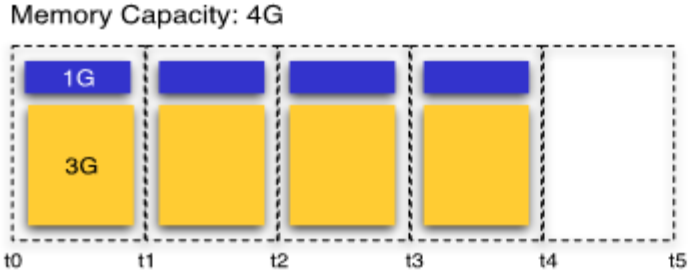
FIFO



FAIR



FFD



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

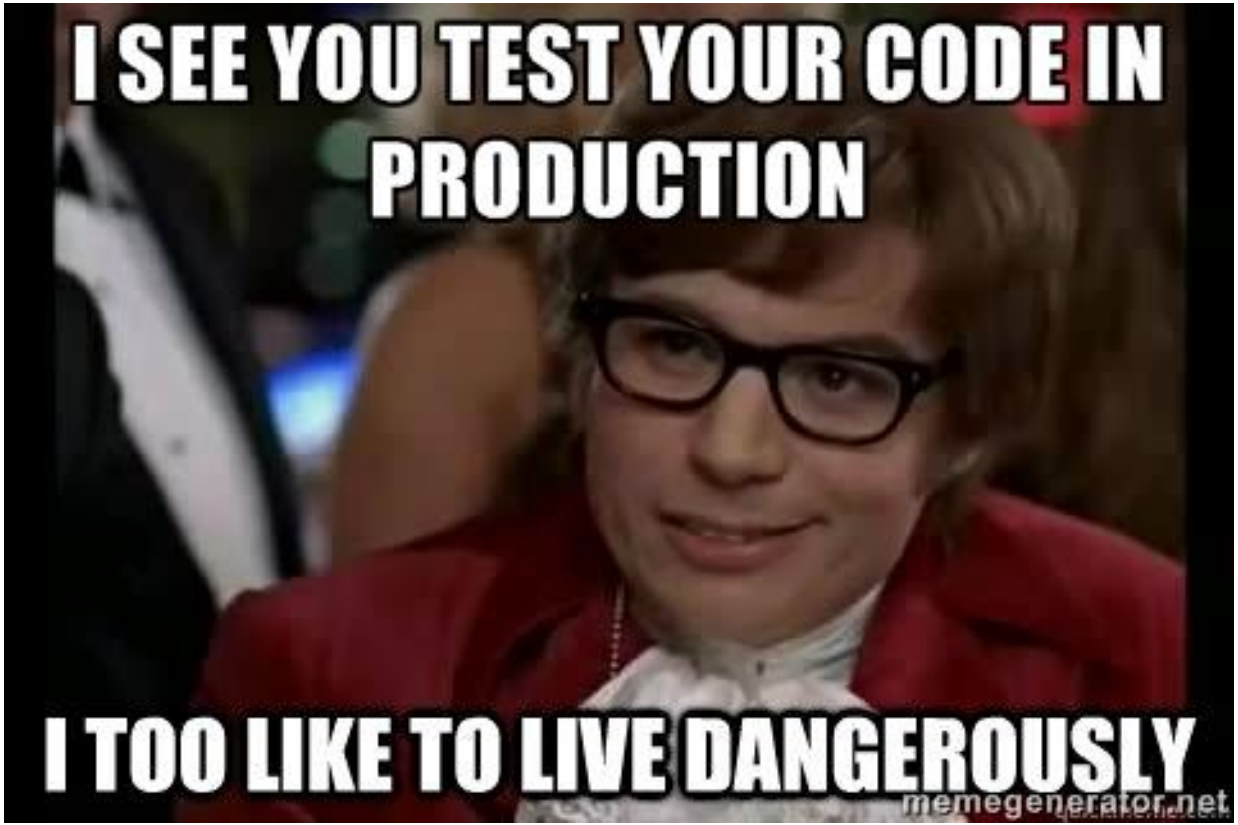
Different scheduling algorithms

New features

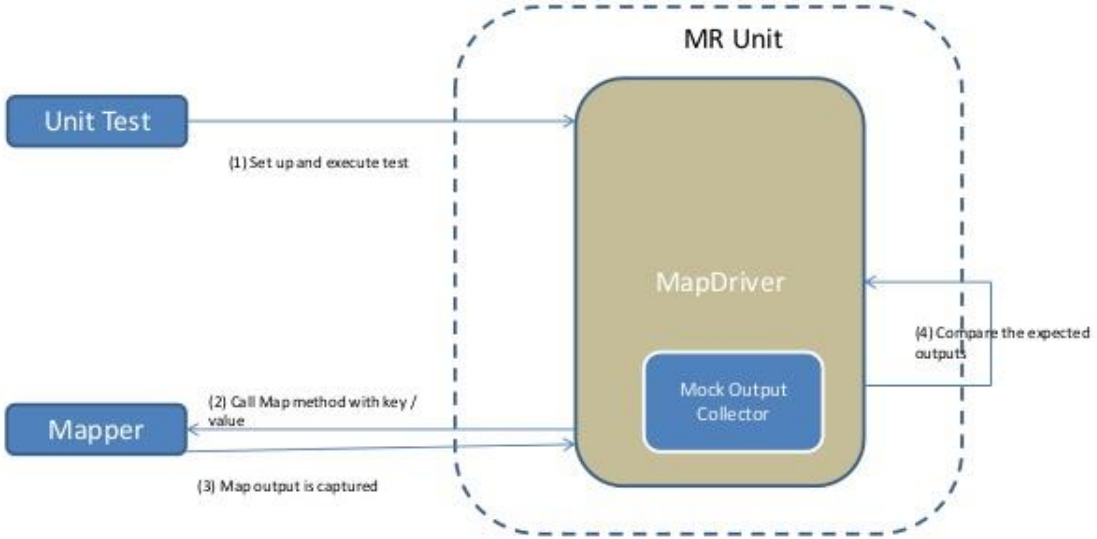
- Dynamic Resource Configuration
- Support disk as a resource in YARN
- Support for Docker containers in YARN
- Resource Manager High Availability
- Support NodeGroup layer topology on YARN

TESTING & DEVELOPMENT

MR Unit idea


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

MRUnit – Testing Mapper



Do it separatly

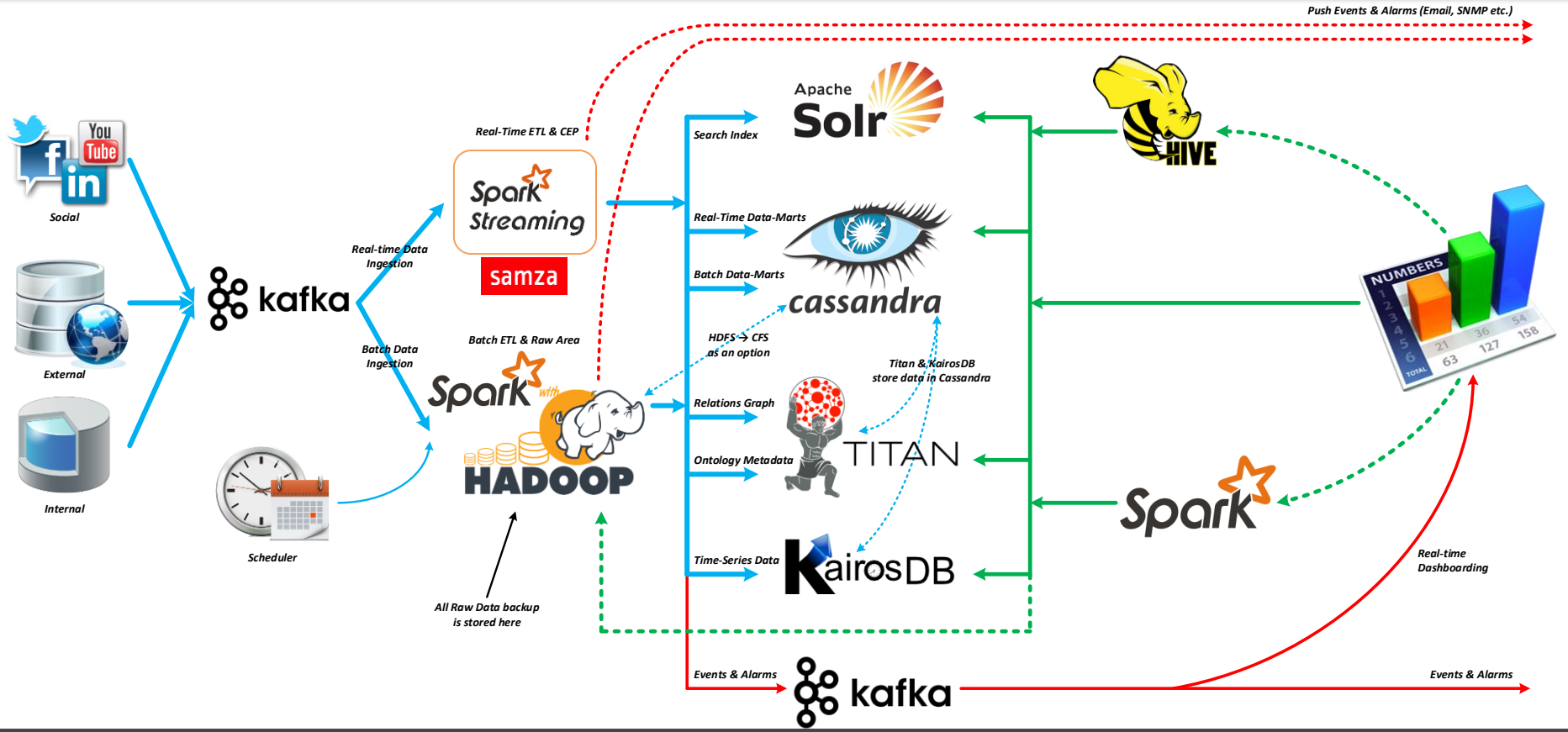
```
public class MRUnitHelloWorld {
    MapDriver<LongWritable, Text, Text, IntWritable> mapDriver;

    @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();
    }
}
```

Simple Test

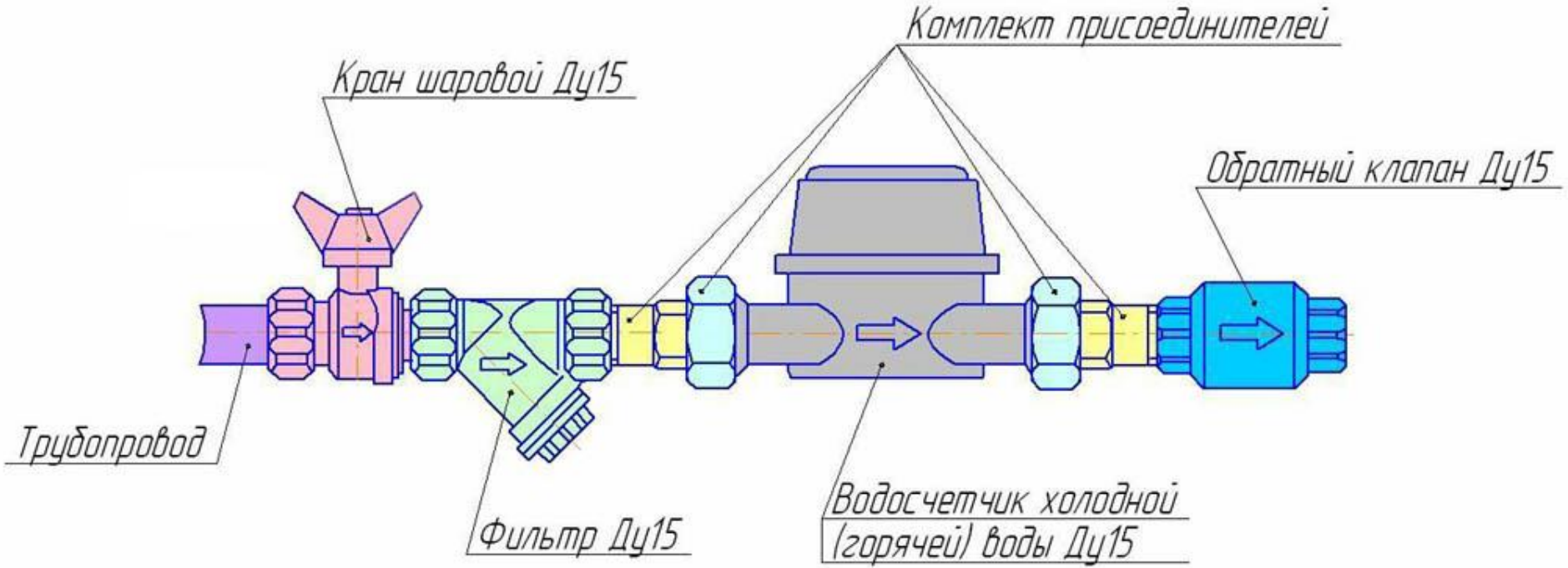
Testing strategies

- 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

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

Hadoop's friend

- Batch and Streaming in the same system
- Full Hadoop compatibility
- Automatic, language independent optimizer
- Not near-realtime like Spark, but really REALTIME, like STORM

Hadoop’s friend

Contacts

E-mail : Alexey_Zinovyev@epam.com

Twitter : @zaleslaw @BigDataRussia

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

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