Uma introdução ao processamento de dados com Hadoop MapReduce

Alan Cortes

Objetivo:

- Uma introdução

Objetivo:

- Uma introdução
- Uma visão simplificada

Objetivo:

- Uma introdução
- Uma visão simplificada
- Omissão de vários detalhes

Objetivo:

- Uma introdução
- Uma visão simplificada
- Omissão de vários detalhes
- Ao final ter uma idéia de como fazer BY
- Não tratarei de como instalar...

What Is Apache Hadoop?

- The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models.

fonte: http://hadoop.apache.org/

Versões:

Version	Release Date
3.0.0-beta1	03 October, 2017
2.8.2	24 Oct, 2017
2.7.4	04 August, 2017
2.6.5	08 October, 2016

Stable: 2.8.2

Módulos do projeto:

- **Hadoop Common**: The common utilities that support the other Hadoop modules.

Módulos do projeto:

- **Hadoop Common**: The common utilities that support the other Hadoop modules.
- **Hadoop Distributed File System (HDFS™)**: A distributed file system that provides high-throughput access to application data.

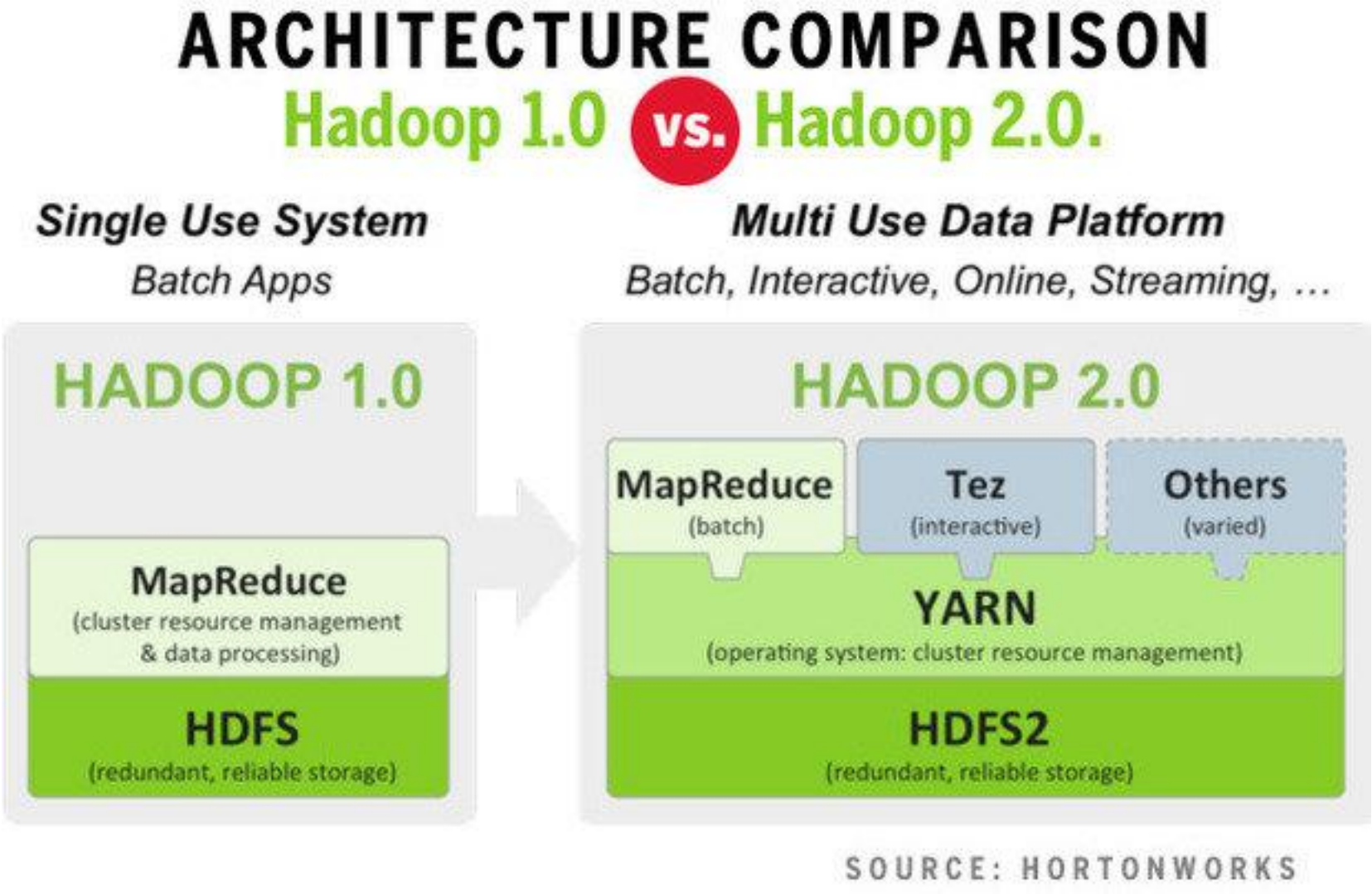
Módulos do projeto:

- **Hadoop Common**: The common utilities that support the other Hadoop modules.
- **Hadoop Distributed File System (HDFS™)**: A distributed file system that provides high-throughput access to application data.
- **Hadoop YARN**: A framework for job scheduling and cluster resource management.

Módulos do projeto:

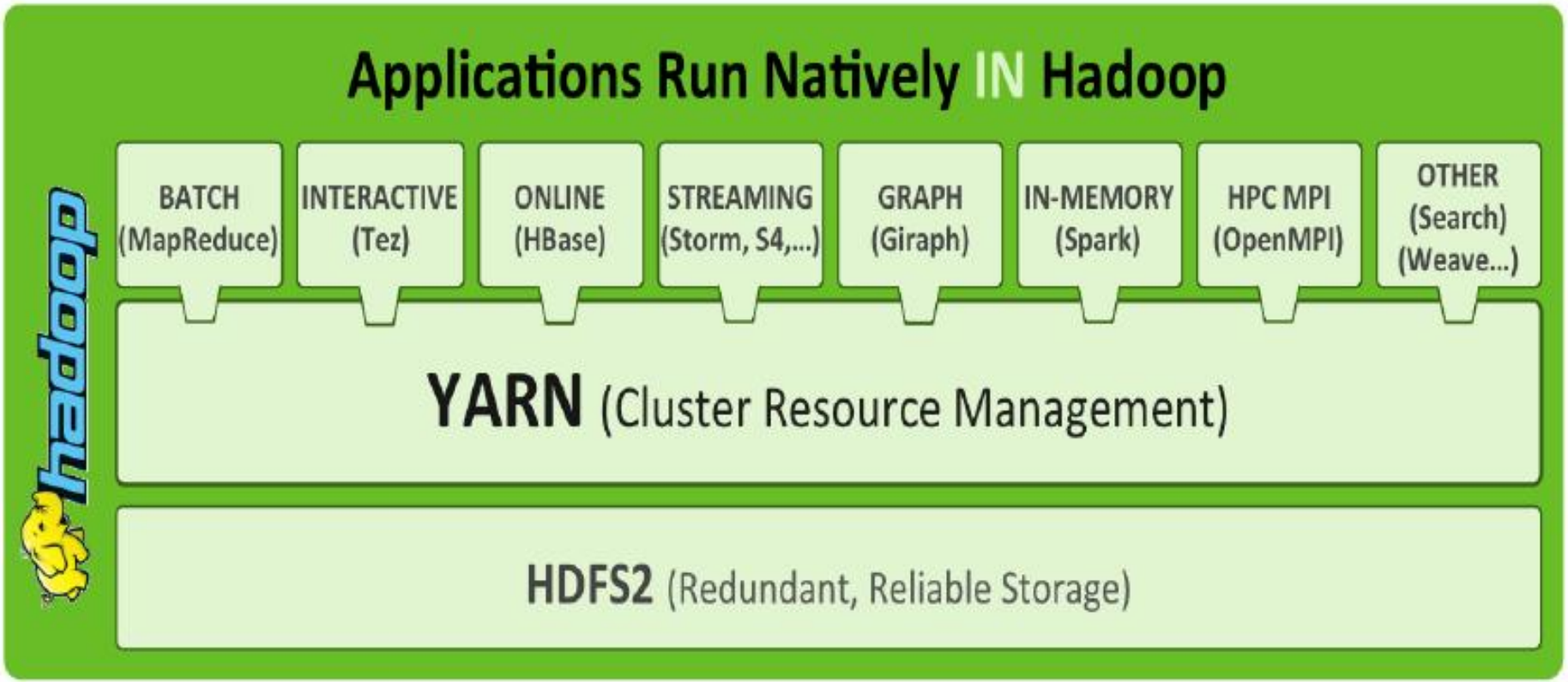
- **Hadoop Common**: The common utilities that support the other Hadoop modules.
- **Hadoop Distributed File System (HDFS™)**: A distributed file system that provides high-throughput access to application data.
- **Hadoop YARN (Yet Another Resource Negotiator)**: A framework for job scheduling and cluster resource management.
- **Hadoop MapReduce**: A YARN-based system for parallel processing of large data sets.

Módulos do projeto:



fonte: https://images.techhive.com/images/article/2014/06/hadoop-distrib-2-100341943-large.idge.jpg

Módulos do projeto:



fonte: https://weidongzhou.files.wordpress.com/2015/09/hadoop.jpg

HDFS:

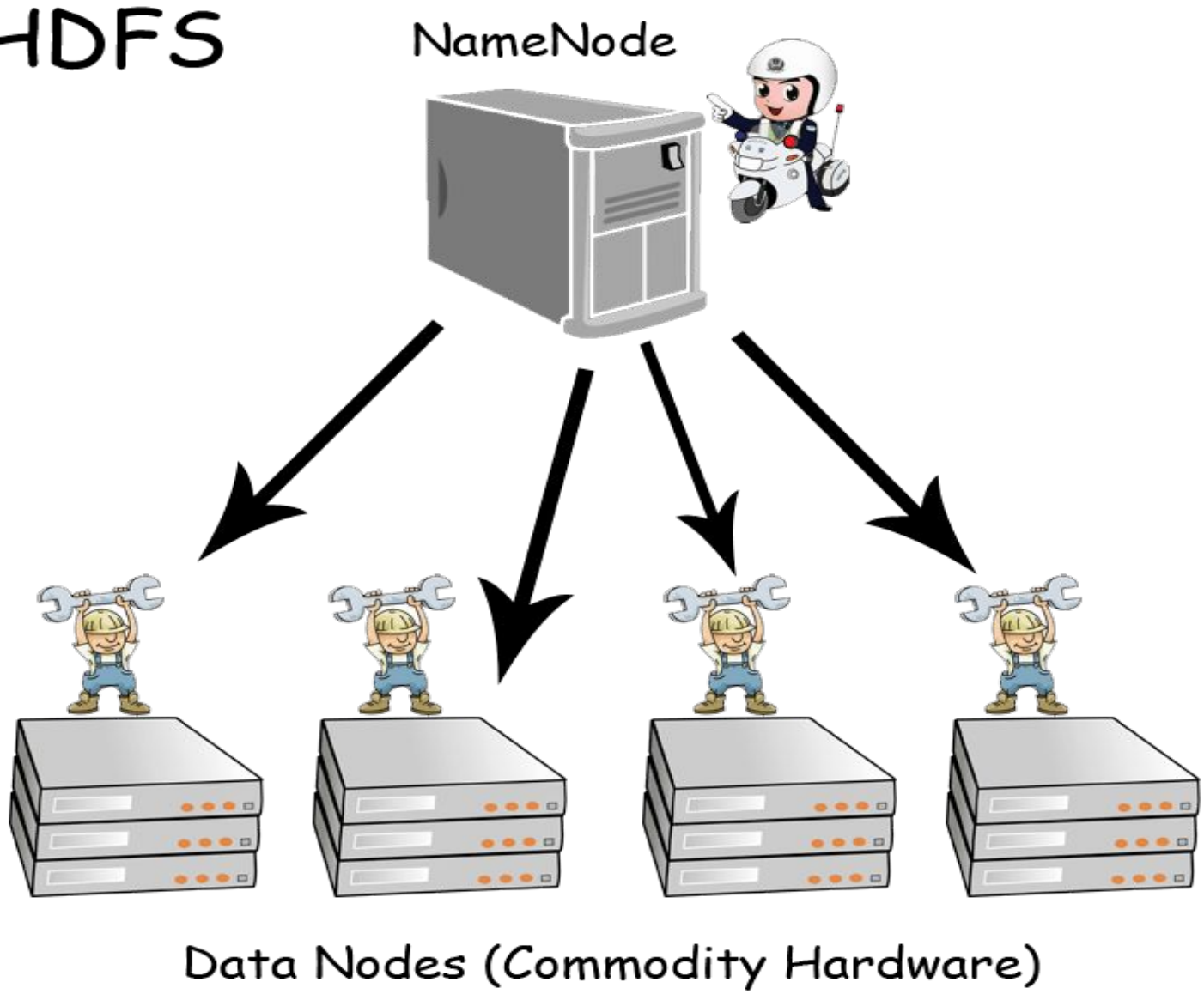
- Sistema De Arquivos Distribuído
  - Hardware Failure
  - Large Data Sets
  - Designed to be deployed on low-cost hardware

HDFS:

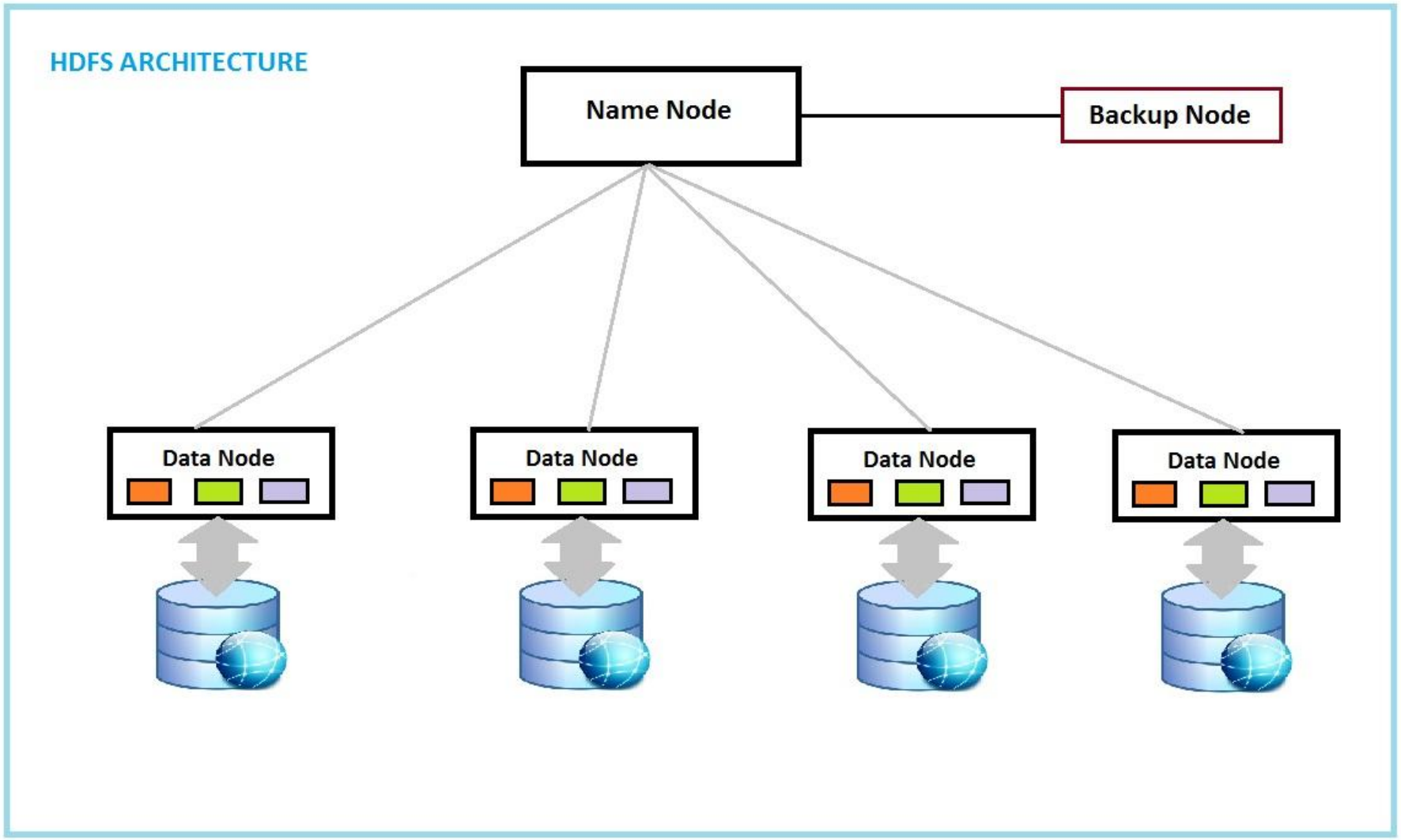
- Sistema De Arquivos Distribuído
  - Hardware Failure
  - Large Data Sets
  - Designed to be deployed on low-cost hardware
- Arquitetura Master/Slave (NameNode/DataNodes)
  - “um” mestre
  - Muitos trabalhadores

HDFS:

- Sistema De Arquivos Distribuído
  - Hardware Failure
  - Large Data Sets
  - Designed to be deployed on low-cost hardware
- Arquitetura Master/Slave (NameNode/DataNodes)
  - “um” mestre
  - Muitos trabalhadores
- Particionamento dos dados em blocos (256MB)
  - Replicação (Redundância)



fonte: http://www.wingnity.com/blog/wp-content/uploads/2014/06/hdfs-hadoop.png



fonte: https://codemphasis.files.wordpress.com/2012/09/hdfs-arch.jpg

YARN:

- Dividir Gerenciamento de Recursos do gerenciamento dos Jobs (tarefas).
  - Lidar com as Falhas

fonte: https://hadoop.apache.org/docs/stable/hadoop-yarn/hadoop-yarn-site/YARN.html

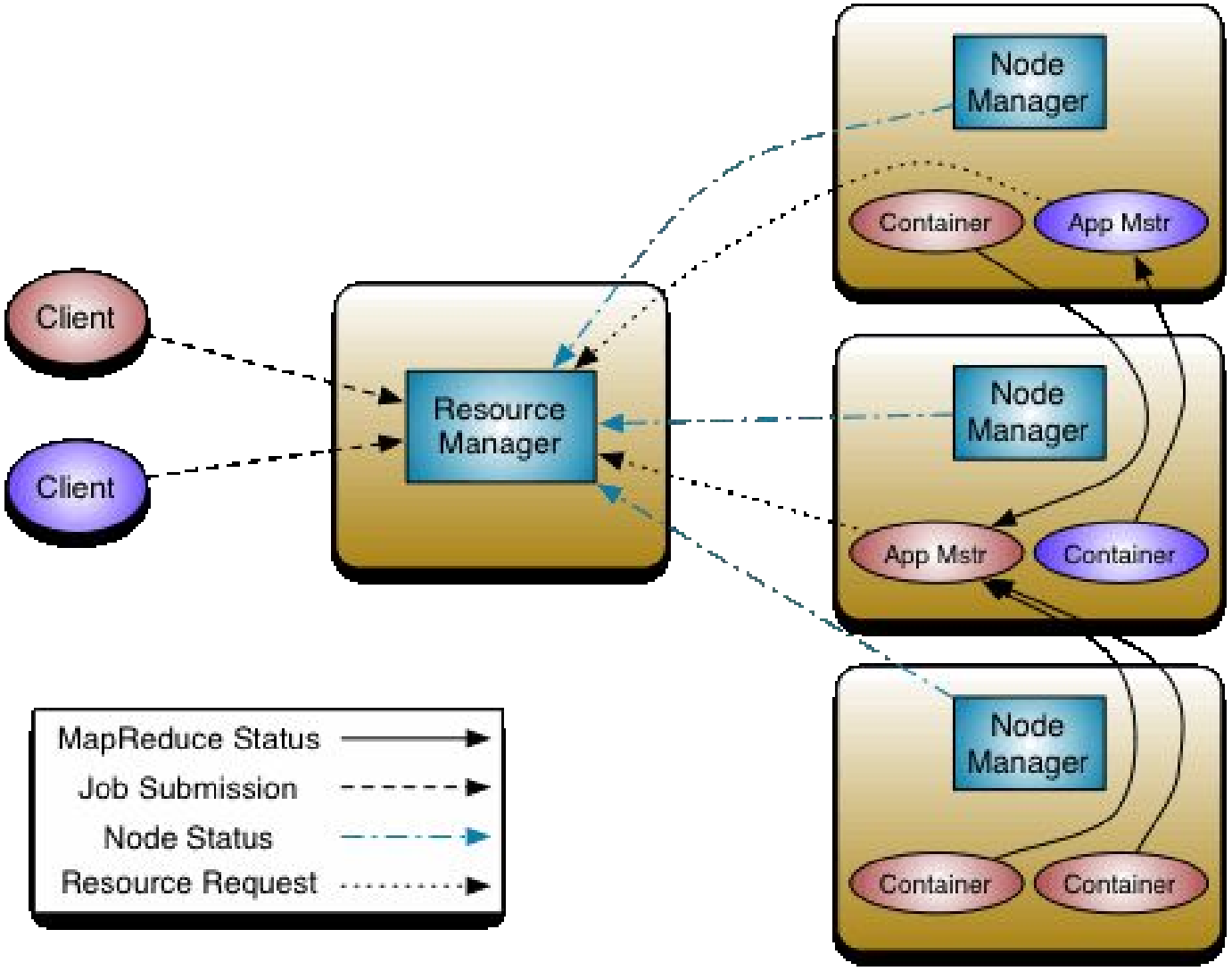
YARN:

- Dividir Gerenciamento de Recursos do gerenciamento dos Jobs (tarefas).
  - Lidar com as Falhas

- Arquitetura Master/Slave
  - ResourceManager (Master)
  - NodeManager (Cada nó)
  - ApplicationMaster (Um por aplicação)

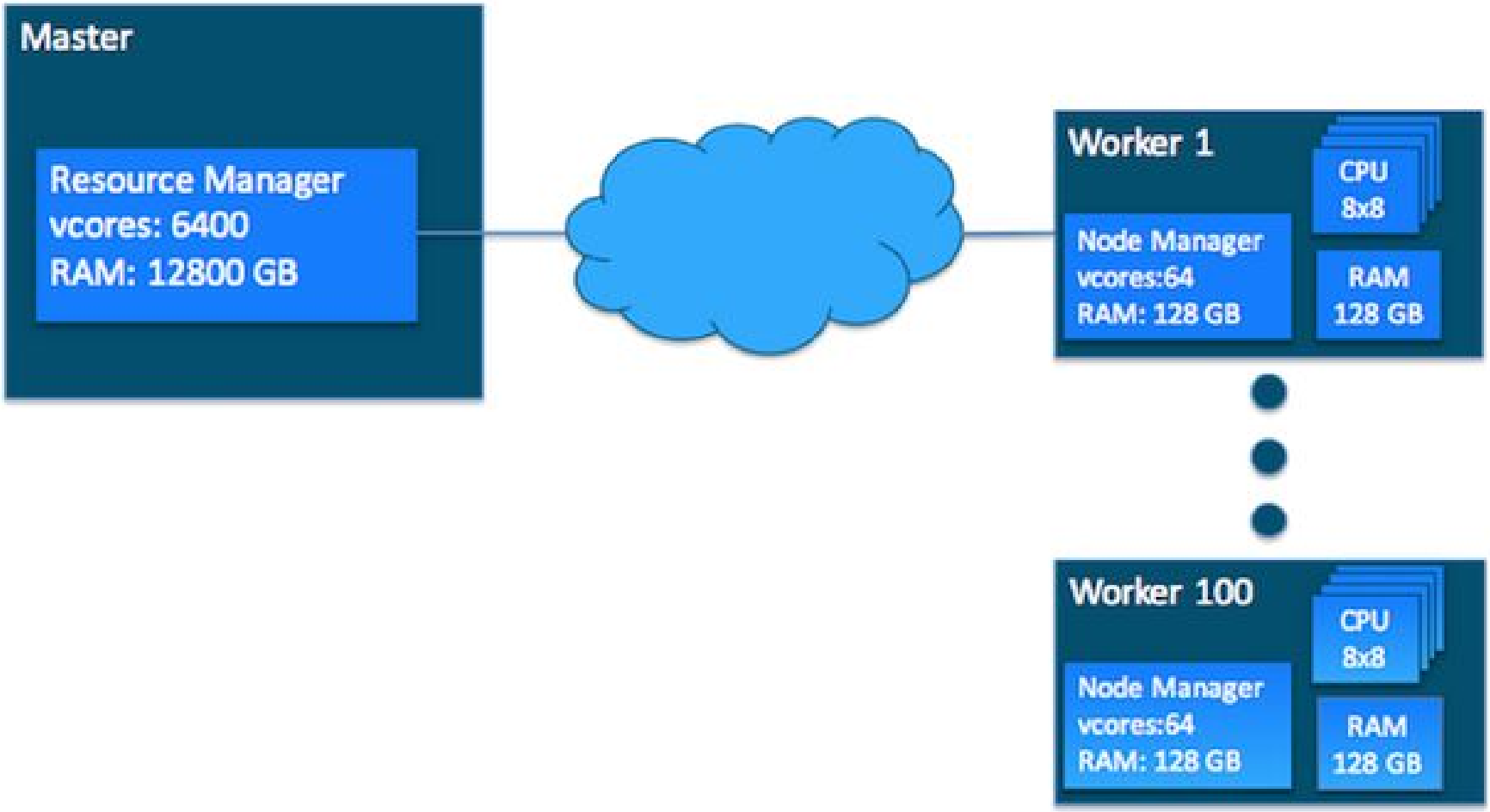
fonte: https://hadoop.apache.org/docs/stable/hadoop-yarn/hadoop-yarn-site/YARN.html

YARN:



fonte : https://hadoop.apache.org/docs/stable/hadoop-yarn/hadoop-yarn-site/YARN.html

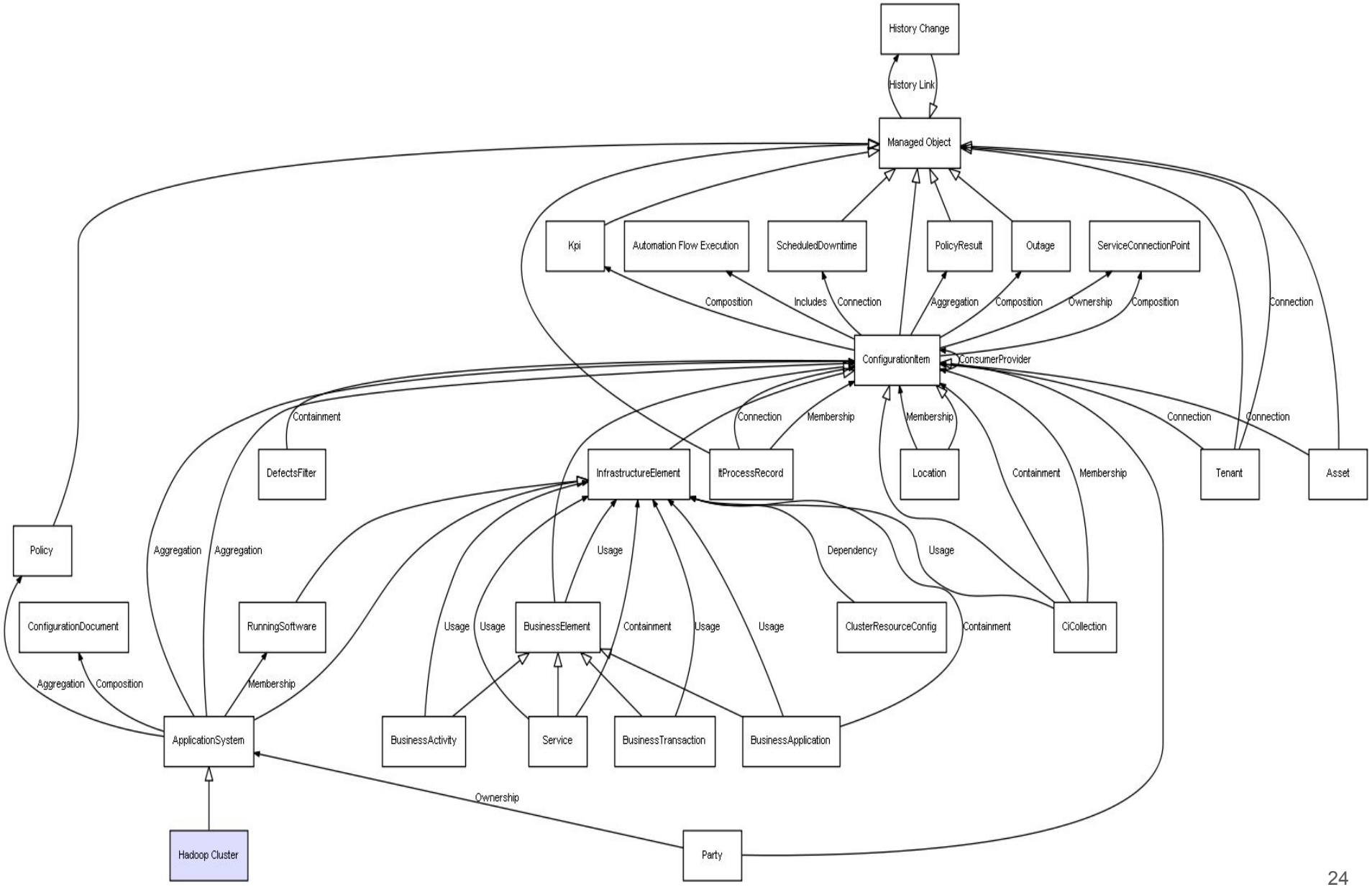
YARN:



fonte: http://blog.cloudera.com/wp-content/uploads/2015/09/untangling-yarn-1-f3.png

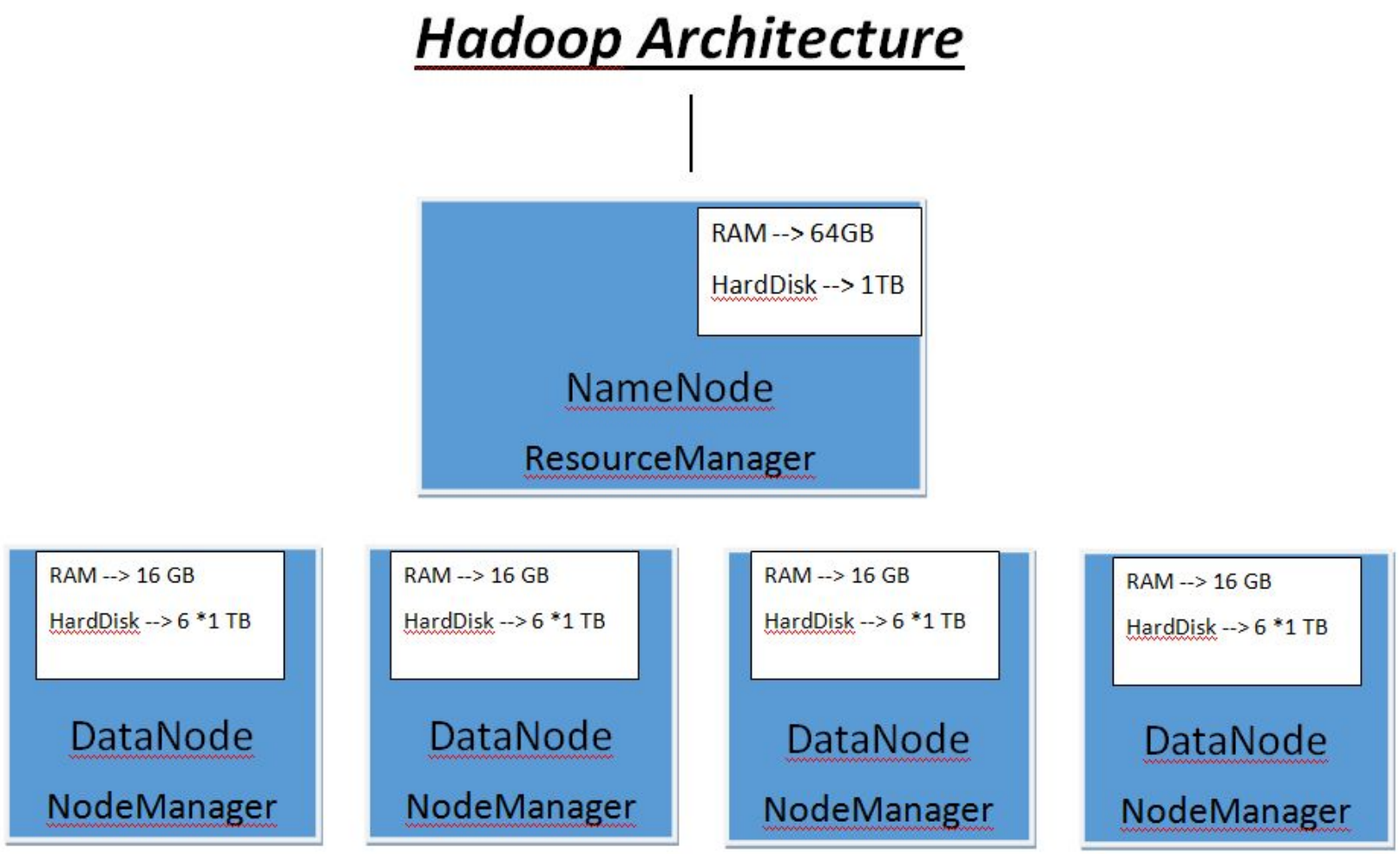
Os dois juntos?

Os dois juntos?



fonte: https://docs.software.hpe.com/UCMDB/10.33/cp-docs/docs/eng/class_model/html/hadoop_cluster_rel_full.png

Os dois juntos?



TIP: processa localmente os blocos disponíveis naquele nó. (mover a computação em vez dos dados)

fonte: https://www.linkedin.com/pulse/hadoop-architecture-nikunj-aggarwal

Map Reduce:

- Paradigma de programação...

Map Reduce:

- Paradigma de programação...
- Não é nada novo...
  - Já existe a muito tempo (linguagens funcionais [Lisp])

Map Reduce:

- Paradigma de programação...
- Não é nada novo...
  - Já existe a muito tempo (linguagens funcionais [Lisp])
- Se baseia em duas coisas.
  - Funções Map(<Chave , Valor>)
    e Reduce(<Chave , Valor>)
  - Map processa dados, e o Reduce processa o resultado do Map

  Map(k1,v1) → list(k2,v2)
  Reduce(k2, list (v2)) → list(k3, v3)

WordCount...

```
function map(String name, String document):
  // name: document name
  // document: document contents
  for each word w in document:
    emit (w, 1)

function reduce(String word, Iterator partialCounts):
  // word: a word
  // partialCounts: a list of aggregated partial counts
  sum = 0
  for each pc in partialCounts:
    sum += pc
  emit (word, sum)
```


fonte: https://en.wikipedia.org/wiki/MapReduce

WordCount...

Texto: José, João, Maria, João, Maria, Felipe, Alan

Map → {<José, 1>, <João, 1>, <Maria, 1>, <José, 1>,
<João, 1>, <Maria, 1>, <Felipe, 1>, <Alan, 1>}

Mágica → <José, (1,1)>, <João, (1,1)>, <Maria, (1,1),
<Felipe, (1)>, <Alan, (1)>

Reduce(<José, (1,1)>) → <José, 2>
Reduce(<João, (1,1)>) → <João, 2>
Reduce(<Maria, (1,1)>) → <Maria, 2>
Reduce(<Felipe, (1)>) → <Felipe, 1>
Reduce(<Alan, (1)>) → <Alan, 1>

E o Hadoop MapReduce?

E o Hadoop MapReduce?

- Framework para escrever facilmente aplicações MapReduce que processam grandes quantidades de dados, de forma paralela, distribuída em grandes clusters de hardware commodity com tolerância a falhas.

fonte:http://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html

E o Hadoop MapReduce?

- Uma aplicação (*Job*) MapReduce geralmente divide os dados de entrada em *Blocos* independentes que são processados pelos Maps paralelamente.
- O framework organiza (sort/shuffle) as saídas dos Maps que serão as entradas dos Reducers (Mágica).
- Geralmente a entrada e a saída dos *Jobs* estão/são armazenadas no HDFS.
- O framework se encarrega de organizar e monitorar as tarefas, e reexecutar as que falharam.

fonte:http://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html

E o Hadoop MapReduce?

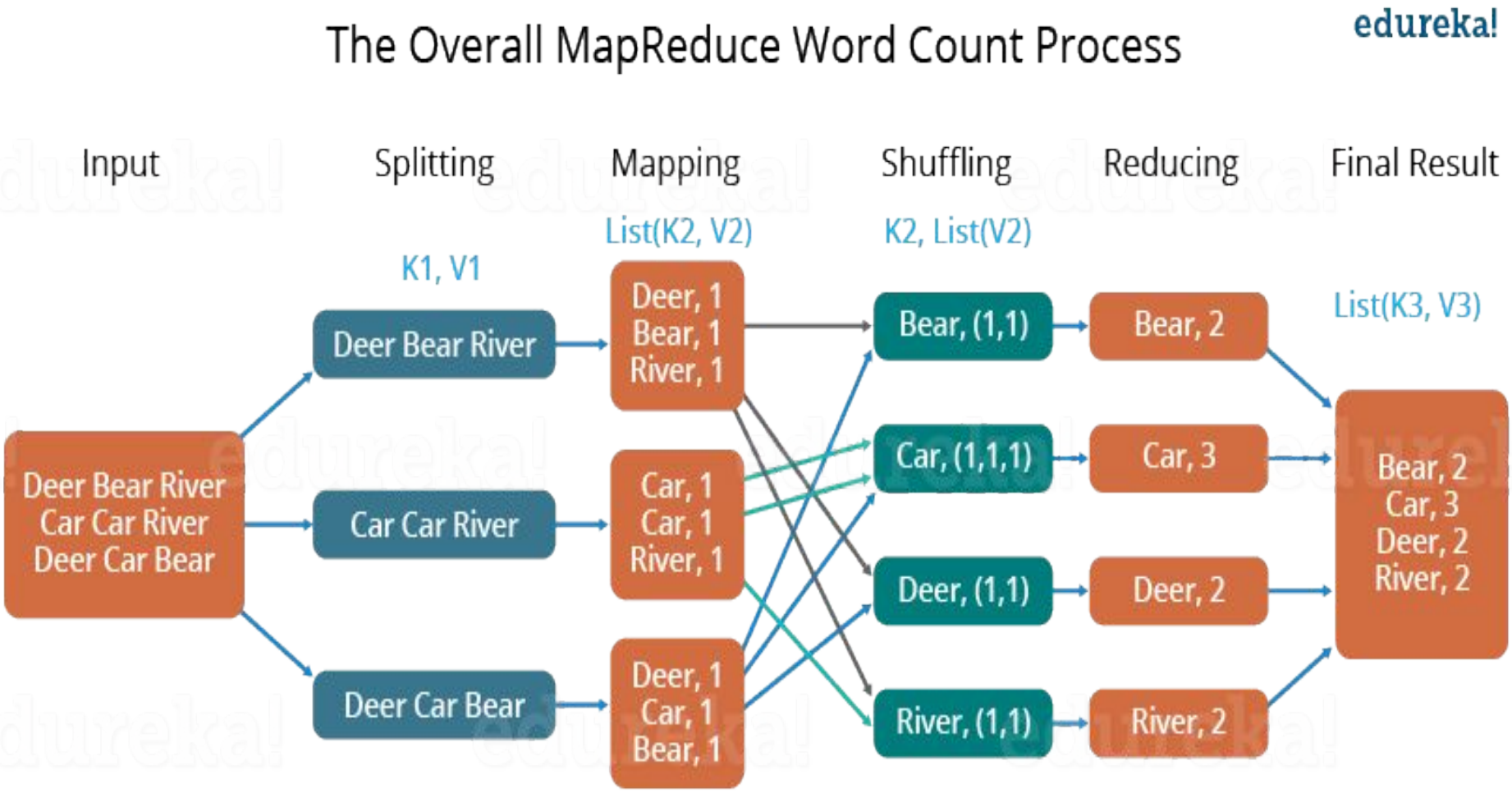
- Uma aplicação (*Job*) MapReduce geralmente divide os dados de entrada em *Blocos* independentes que são processados pelos Maps paralelamente.
- O framework organiza (sort/shuffle) as saídas dos Maps que serão as entradas dos Reducers (Mágica).
- Geralmente a entrada e a saída dos *Jobs* estão/são armazenadas no HDFS.
- O framework se encarrega de organizar e monitorar as tarefas, e reexecutar as que falharam.

❖ Minimamente um Job MapReduce precisa:
  - ➢ Especificar a localização da entrada e da saída;
  - ➢ Prover as funções Map e Reduce;
    - ■ Implementação das interfaces apropriadas/classes abstratas
  - ➢ Esses e outros parâmetros compõem a configuração do *Job*;
❖ O cliente então submete o Jar/Executável para o ResourceManager
  - ➢ O ResourceManager assume dai;
    - ■ Ele se vira com o resto;

fonte:http://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html

WordCount...

WordCount...



fonte: https://www.edureka.co/blog/mapreduce-tutorial/

Agora um pouco de Java...

origem:
http://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html

Agora um pouco de Java...

```
import java.io.IOException;
import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;


public class WordCount {
     .
     .
     .
```

Agora um pouco de Java...

```
public class WordCount {

  public static class TokenizerMapper
    extends Mapper<Object, Text, Text, IntWritable>{

    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(Object key, Text value, Context context
                 ) throws IOException, InterruptedException {
        StringTokenizer itr = new StringTokenizer(value.toString());

        while (itr.hasMoreTokens()) {
            word.set(itr.nextToken());
            context.write(word, one);
        }
    }
  }
    .
    .
    .
```

Agora um pouco de Java...

```
public class WordCount {
     .
     .
     .
   public static class IntSumReducer
       extends Reducer<Text,IntWritable,Text,IntWritable> {
    private IntWritable result = new IntWritable();

    public void reduce(Text key, Iterable<IntWritable> values,Context context
                       ) throws IOException, InterruptedException {
      int sum = 0;
      for (IntWritable val : values) {
         sum += val.get();
      }
      result.set(sum);
      context.write(key, result);
    }
  }
     .
     .
     .
```

Agora um pouco de Java...

```
public class WordCount {
     .
     .
     .

  public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "word-count");
    job.setJarByClass(WordCount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }

}//fim WordCount
```

Vamos rodar de verdade...

DESAFIO:

desafio.txt

Cidade	Dia da Semana	Temp. Mínima	Temp. Máxima	Elevação
Goiânia	1	22	33	223
Goiânia	2	23	25	223
-	-	-	-	-
-	-	-	-	-
Goiânia	7	29	31	223