MC851 - Projeto em Computação I
MC855 - Projeto em Sistemas de Computação
MC857 - Projeto em Sistemas de Informação

HDFS, MapReduce e YARN

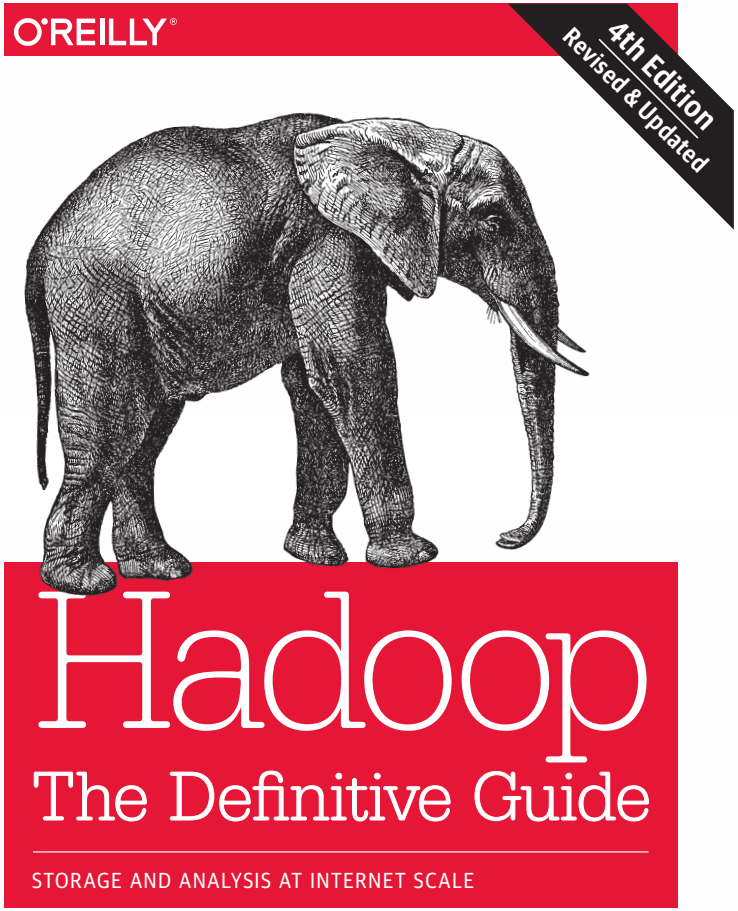
Islene Calciolari Garcia

Instituto de Computação - Unicamp

Segundo Semestre de 2016

Sumário

Motivação



Exemplo retirado do livro *Hadoop—The Definitive Guide*

- Achar a temperatura máxima por ano em um conjunto de arquivos texto
- Codificar todo o trabalho em Unix...
- Entender a importância de um *framework*

Weather dataset

Dados crus

```
0057
332130   # USAF weather station identifier
99999    # WBAN weather station identifier
19500101 # observation date
0300     # observation time
4
+51317   # latitude (degrees x 1000)
+028783  # longitude (degrees x 1000)
FM-12
+0171    # elevation (meters)
99999
V020
320      # wind direction (degrees)
1        # quality code
N
0072
1
00450    # sky ceiling height (meters)
1        # quality code
C
N
010000   # visibility distance (meters)
1        # quality code
N
9
-0128    # air temperature (degrees Celsius x 10)
1        # quality code
-0139    # dew point temperature (degrees Celsius x 10)
```

Fonte: Hadoop—The Definitive Guide, *Tom White*

Weather dataset

Organização dos arquivos

```
% ls raw/1990 | head
010010-99999-1990.gz
010014-99999-1990.gz
010015-99999-1990.gz
010016-99999-1990.gz
010017-99999-1990.gz
010030-99999-1990.gz
010040-99999-1990.gz
010080-99999-1990.gz
010100-99999-1990.gz
010150-99999-1990.gz
```

Fonte: Hadoop—The Definitive Guide, *Tom White*

Weather dataset

Código em `awk` e saída

```
#!/usr/bin/env bash
for year in all/*
do
  echo -ne `basename $year .gz`"\t"
  gunzip -c $year | \
    awk '{ temp = substr($0, 88, 5) + 0;
           q = substr($0, 93, 1);
           if (temp !=9999 && q ~ /[01459]/ && temp > max) max = temp }
         END { print max }'
done
```

```
% ./max_temperature.sh
1901    317
1902    244
1903    289
1904    256
1905    283
...
```

Fonte: Hadoop—The Definitive Guide, *Tom White*

Weather dataset

Como paralelizar?

- Múltiplas threads?
- Um computador por ano?
- Como atribuir trabalho igual para todos?
- Como juntar os resultados parcias?
- Como lidar com as falhas?

Weather dataset

Como paralelizar de maneira mais simples?

- Criar uma infraestrutura que gerencie
  - distribuição
  - escalabilidade
  - tolerância a falhas
- Criar um modelo genérico para *big data*
  - Conjuntos chave-valor
  - Operações *map* e *reduce*

Weather dataset

Dados crus e conjuntos chave-valor

```
0067011990999991950051507004...9999999N9+00001+99999999999...
0043011990999991950051512004...9999999N9+00221+99999999999...
0043011990999991950051518004...9999999N9-00111+99999999999...
0043012650999991949032412004...0500001N9+01111+99999999999...
0043012650999991949032418004...0500001N9+00781+99999999999...
```

```
(0, 0067011990999991950051507004...9999999N9+00001+99999999999...)
(106, 0043011990999991950051512004...9999999N9+00221+99999999999...)
(212, 0043011990999991950051518004...9999999N9-00111+99999999999...)
(318, 0043012650999991949032412004...0500001N9+01111+99999999999...)
(424, 0043012650999991949032418004...0500001N9+00781+99999999999...)
```

Fonte: Hadoop—The Definitive Guide, *Tom White*

Weather dataset

Função map

```
(0, 0067011990999991950051507004...9999999N9+00001+99999999999...)
(106, 0043011990999991950051512004...9999999N9+00221+99999999999...)
(212, 0043011990999991950051518004...9999999N9-00111+99999999999...)
(318, 0043012650999991949032412004...0500001N9+01111+99999999999...)
(424, 0043012650999991949032418004...0500001N9+00781+99999999999...)

(1950, 0)
(1950, 22)
(1950, -11)
(1949, 111)
(1949, 78)
```

Fonte: Hadoop—The Definitive Guide, *Tom White*

Weather dataset

Pré-processamento e função reduce

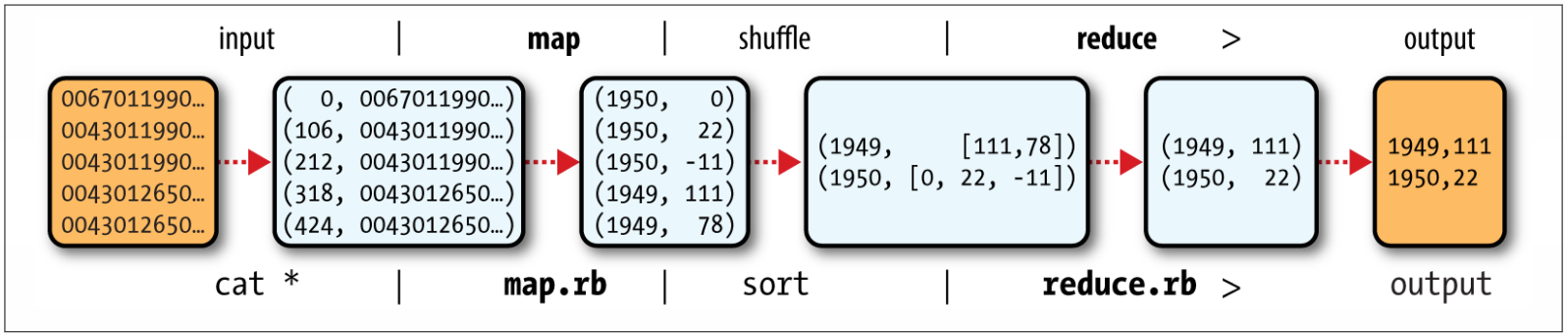
```
(1949, [111, 78])
(1950, [0, 22, -11])

(1949, 111)
(1950, 22)
```

Fonte: Hadoop—The Definitive Guide, *Tom White*

Weather dataset

Fluxo de dados



Fonte: Hadoop—The Definitive Guide, *Tom White*

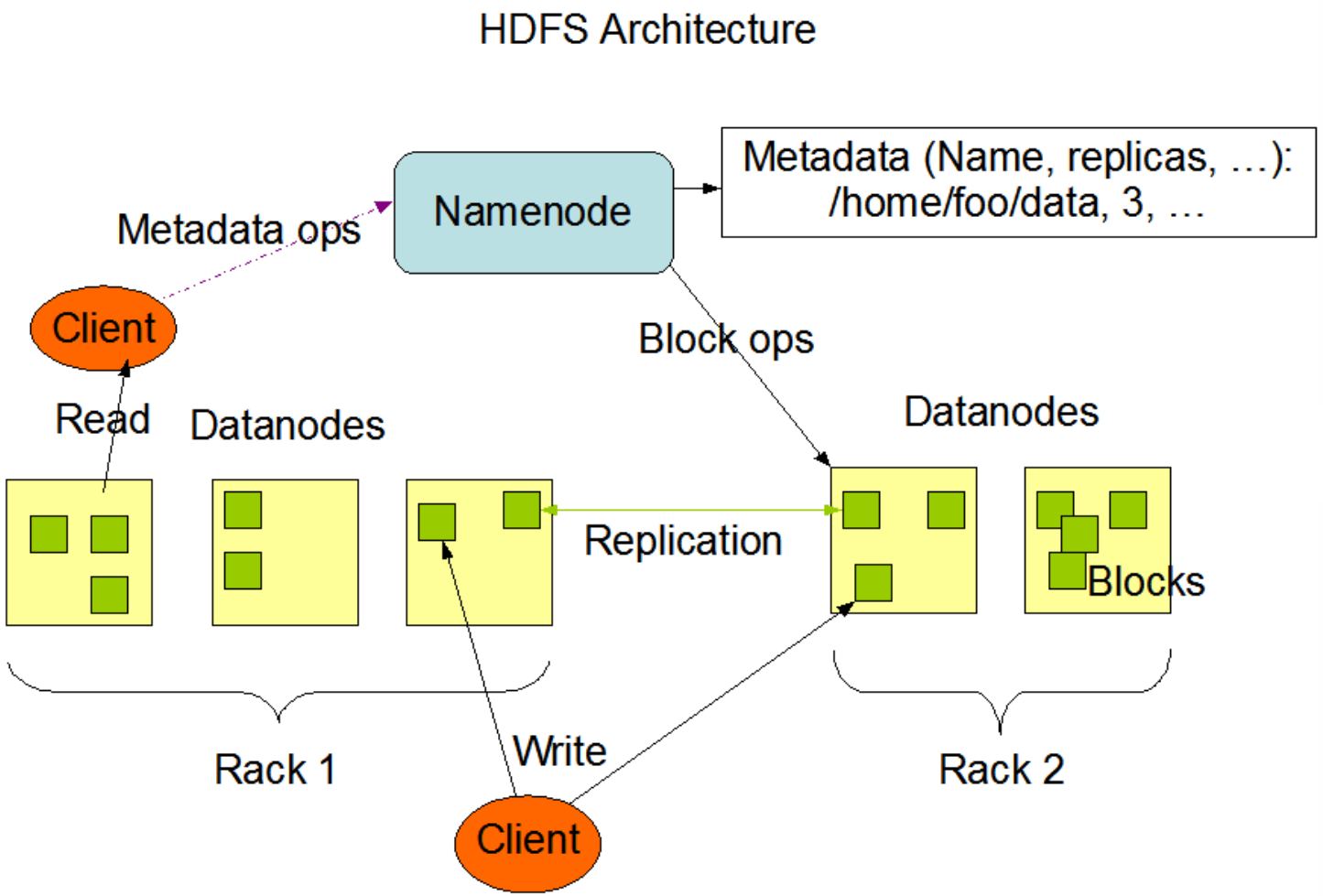
Projeto Apache Hadoop

- Sistema real! Software livre!
- Big Data: *Volume*, *Velocity*, *Variety*, *Veracity*
- Computação distribuída escalável e confiável
- Altamente relevante: usado por empresas como Amazon, Facebook, LinkedIn e Yahoo! Veja mais em Who uses Hadoop?

Um pouco da história do projeto Hadoop

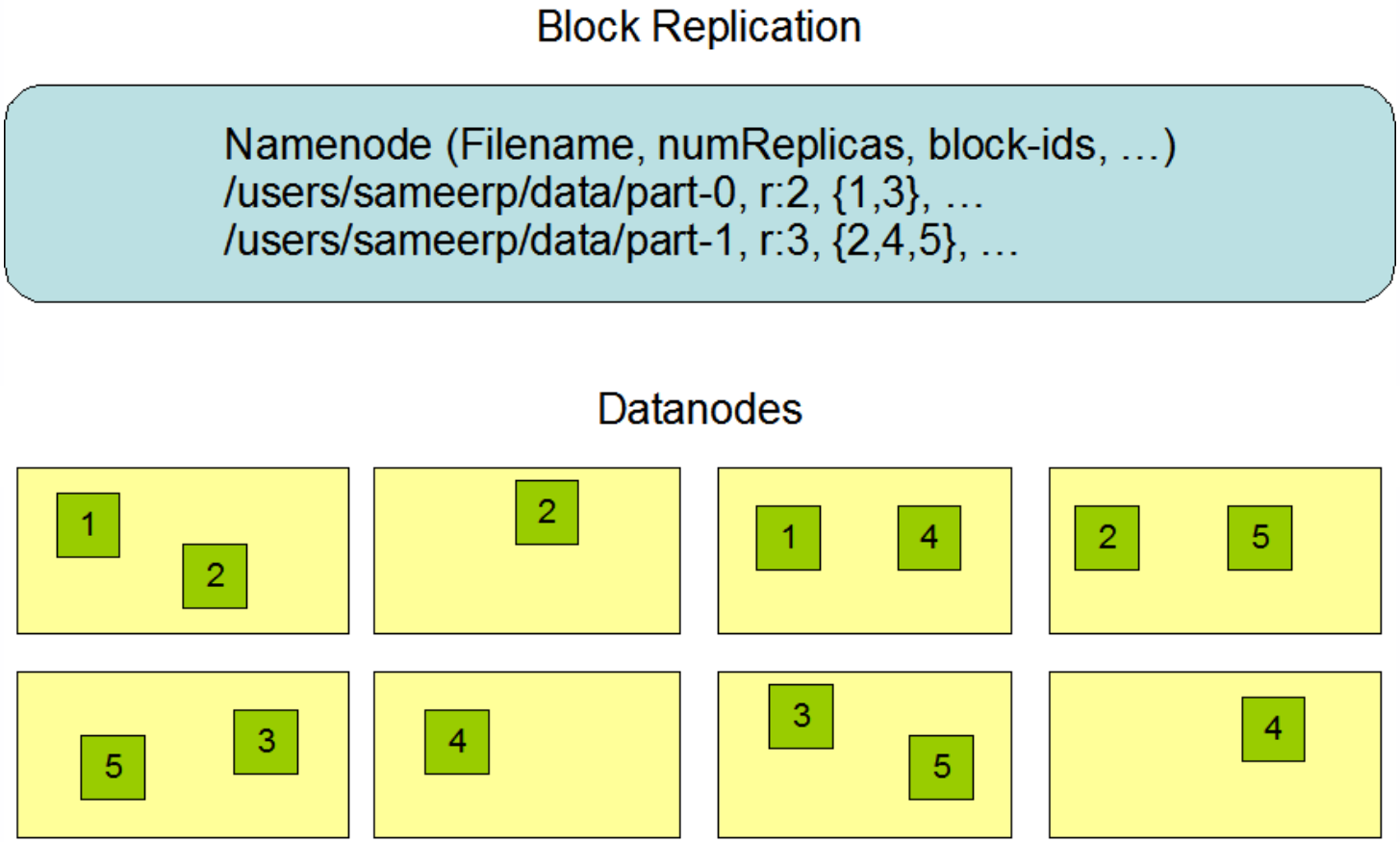
- 2002-2004: Doug Cutting e Mike Cafarella trabalham no projeto Nutch.
  - Nutch deveria indexar a web e permitir buscas
  - Alternativa livre ao Google
- 2003-2004: Google publica artigo sobre o Google File System e MapReduce
- 2004: Doug Cutting adiciona o DFS e MapReduce ao projeto Nutch
- 2006: Doug Cutting começa a trabalhar no Yahoo!
- 2008: Hadoop se torna um projeto Apache

Arquitetura do HDFS



Fonte: `http://hadoop.apache.org`

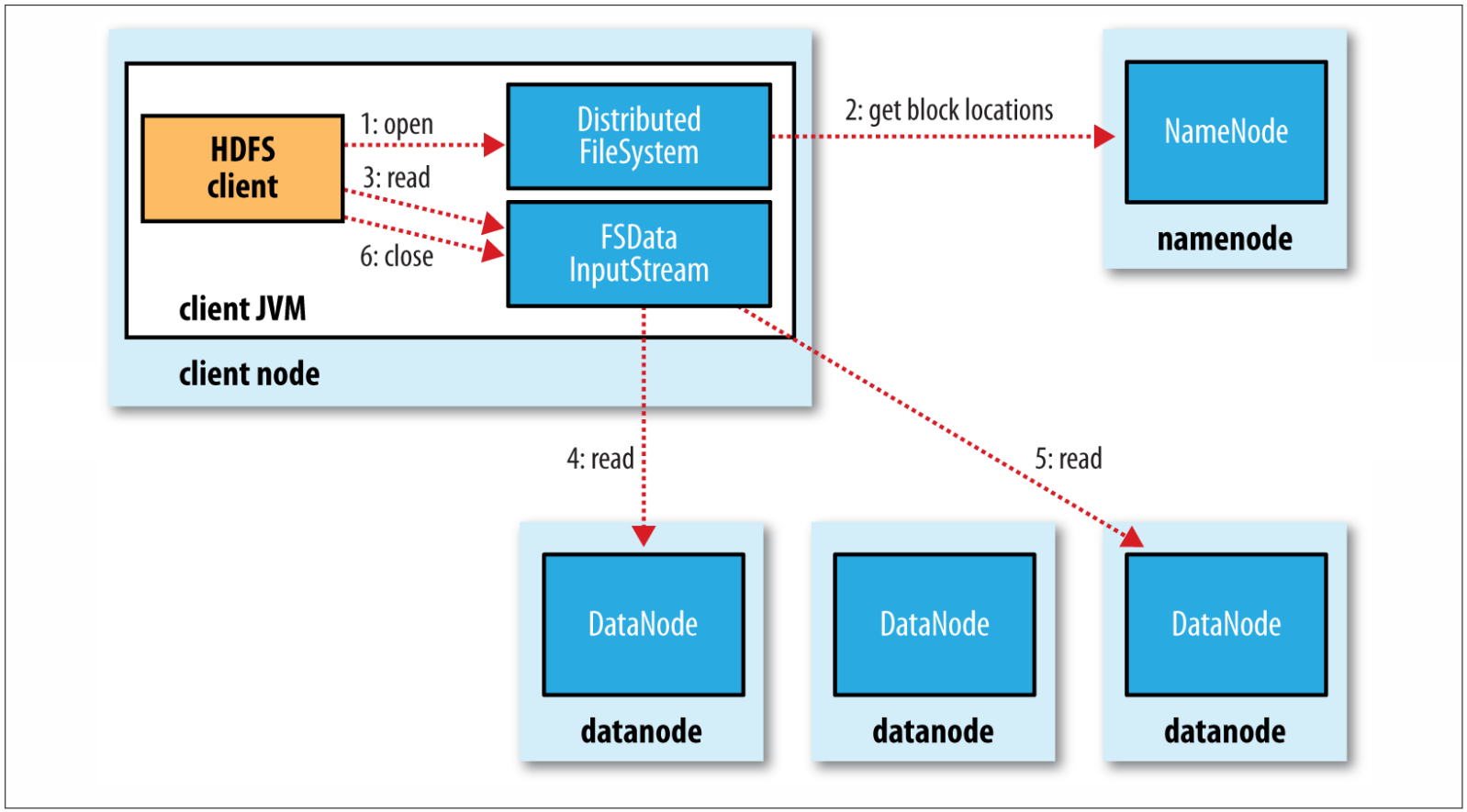
HDFS e réplicas



Fonte: `http://hadoop.apache.org`

HDFS

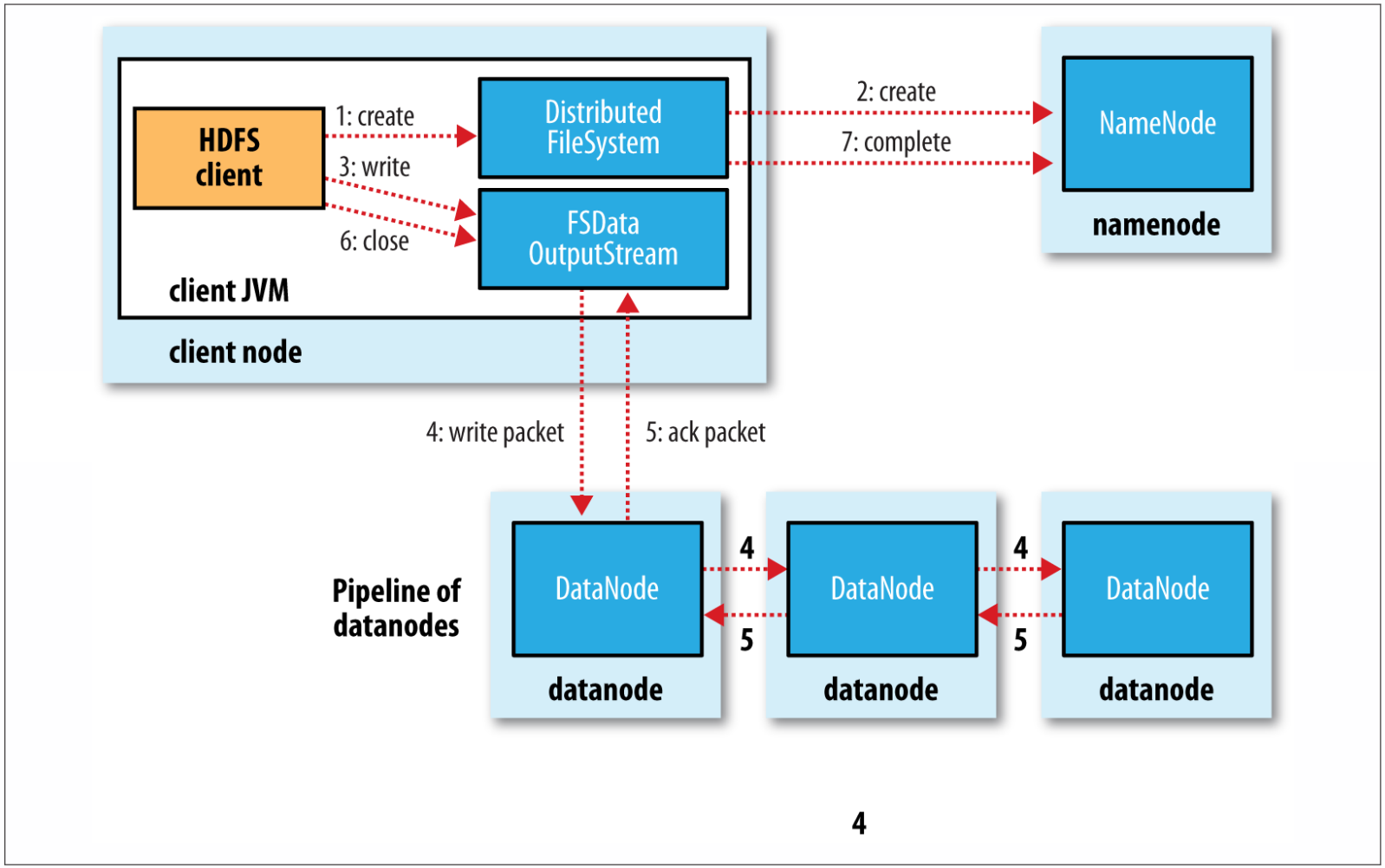
Leitura de arquivo



Fonte: Hadoop—The Definitive Guide, *Tom White*

HDFS

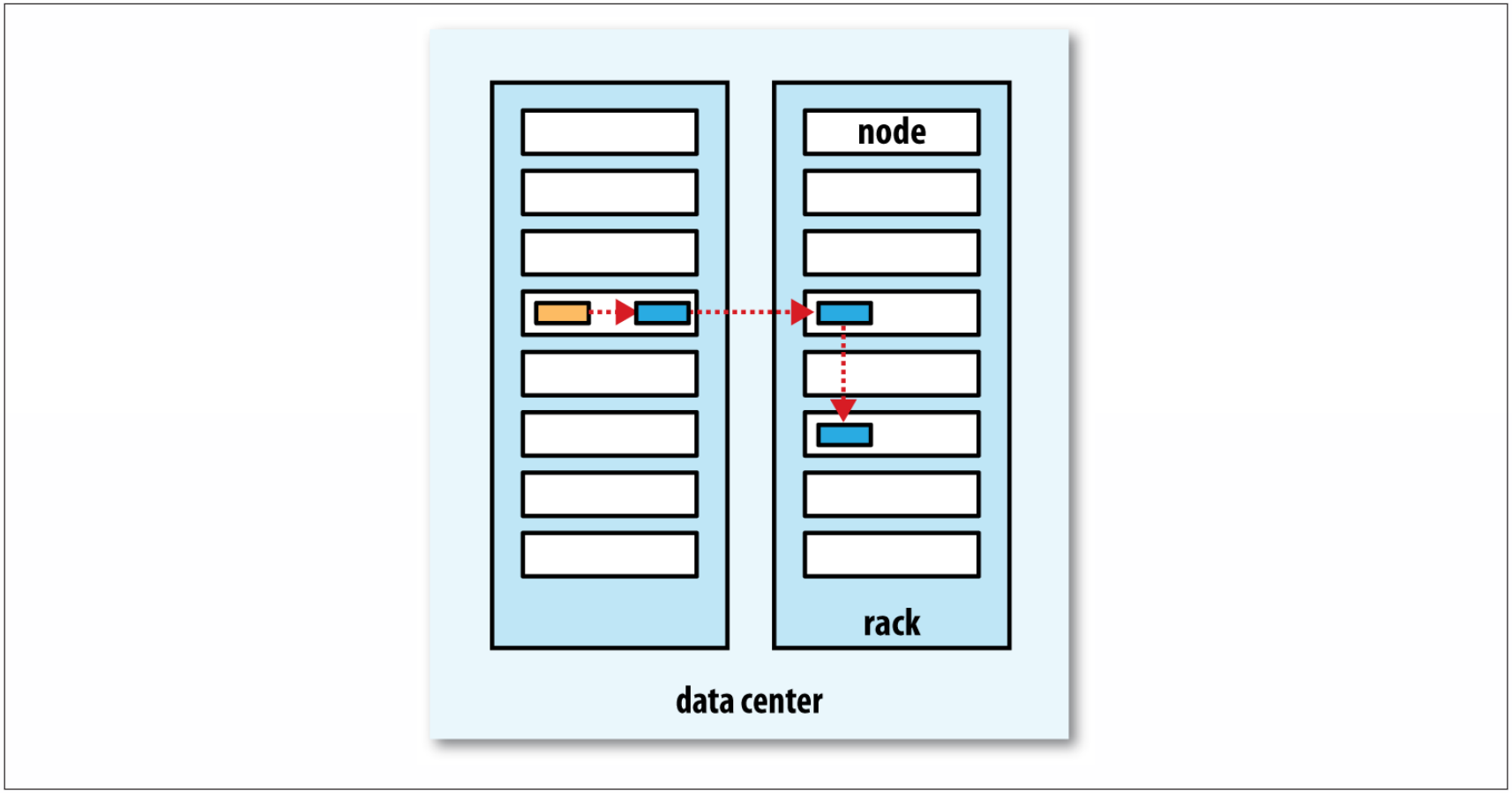
Escrita em arquivo



Fonte: Hadoop—The Definitive Guide, *Tom White*

HDFS

Pipeline



Fonte: Hadoop—The Definitive Guide, *Tom White*

HDFS

Tolerância a falhas

- Heartbeats
- Block reports
- Alta disponibilidade do NameNode
  - [HDFS-1623] High Availability Framework for HDFS NN
- Réplicas ou Erasure Coding?

  Arquivo: | A | B |

  Réplicas simples: | A | | A | | B | | B |

  Erasure coding: | A | | B | | A+B | | A+2*B |
  - [HDFS-7285] Erasure Coding inside HDFS

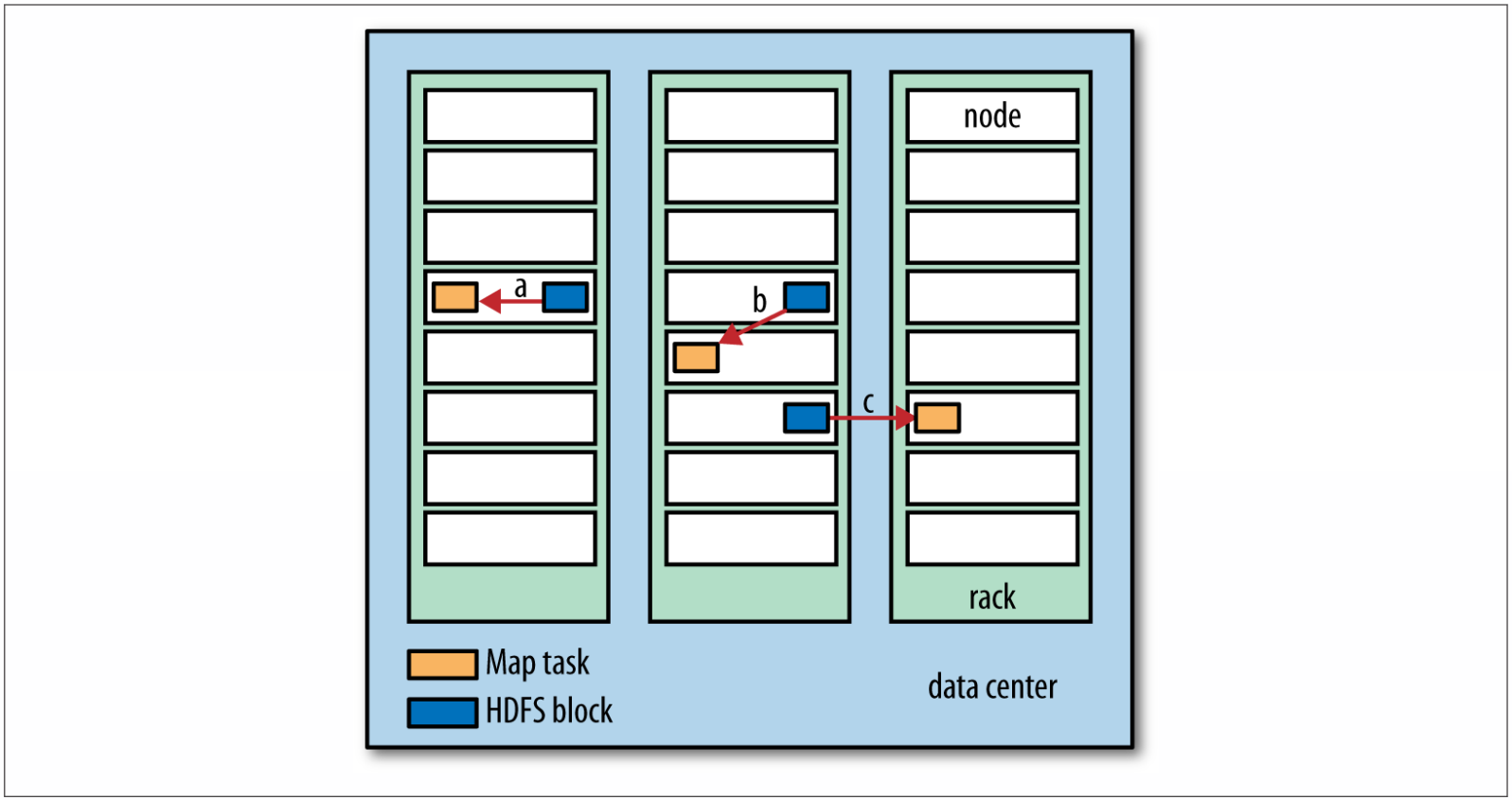
Testando o HDFS

- Hadoop: Setting up a Single Node Cluster
- Interface web: `http://localhost:50070/`
- Alguns comandos

```
$ bin/hdfs namenode -format
$ sbin/start-dfs.sh
$ bin/hdfs dfs -put <arquivo_local> <arquivo_no_hdfs>
$ bin/hdfs dfs -get <arquivo_no_hdfs> <arquivo_local>
$ bin/hdfs dfs -ls <diretorio_no_hdfs>
$ bin/hdfs dfs -rm <arquivo_no_hdfs>
$ bin/hdfs dfs -rm -r <diretorio_no_hdfs>
$ sbin/stop-dfs.sh
```

MapReduce

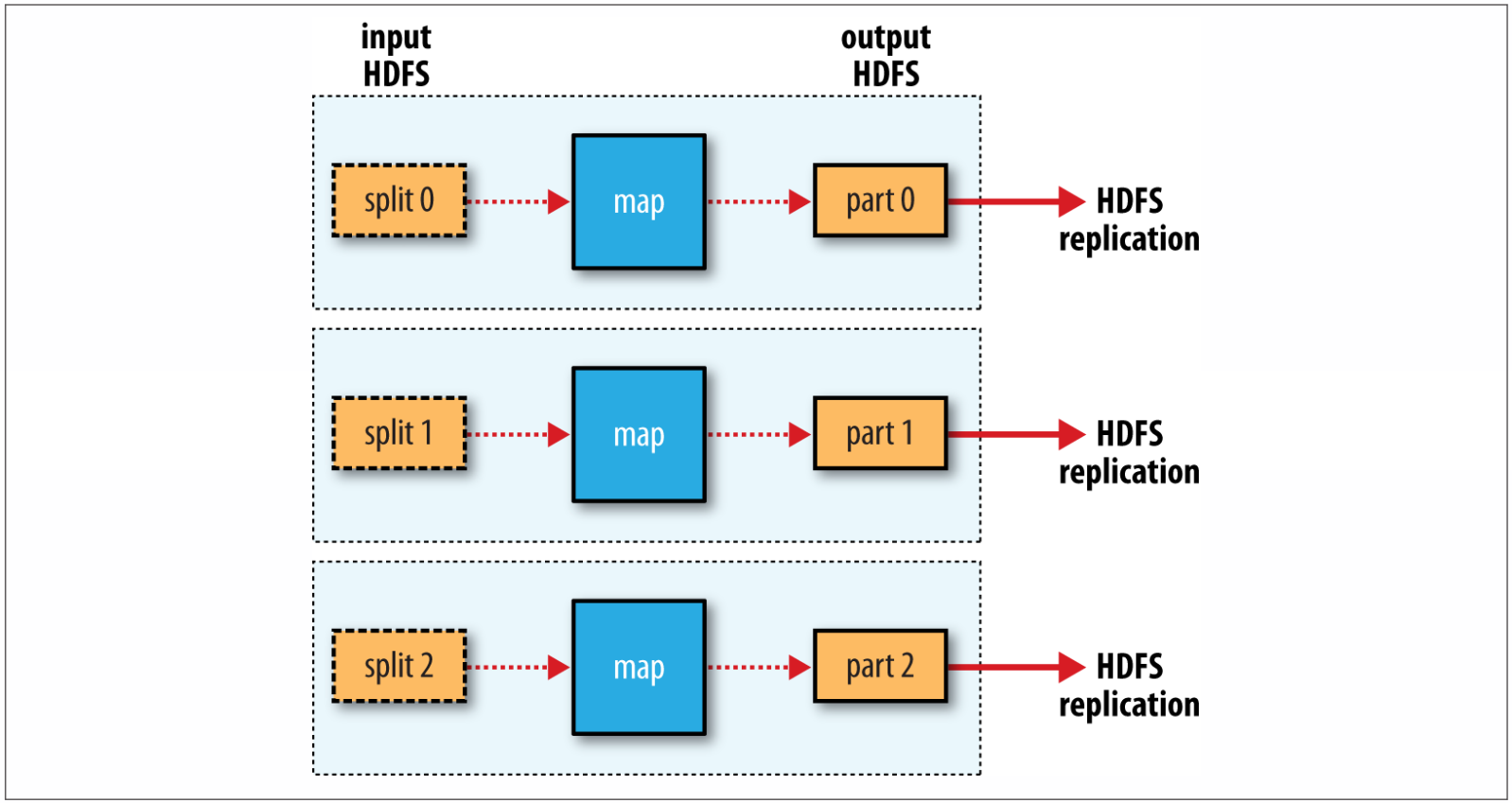
Processamento deve ficar perto dos dados...



Fonte: Hadoop—The Definitive Guide, *Tom White*

MapReduce

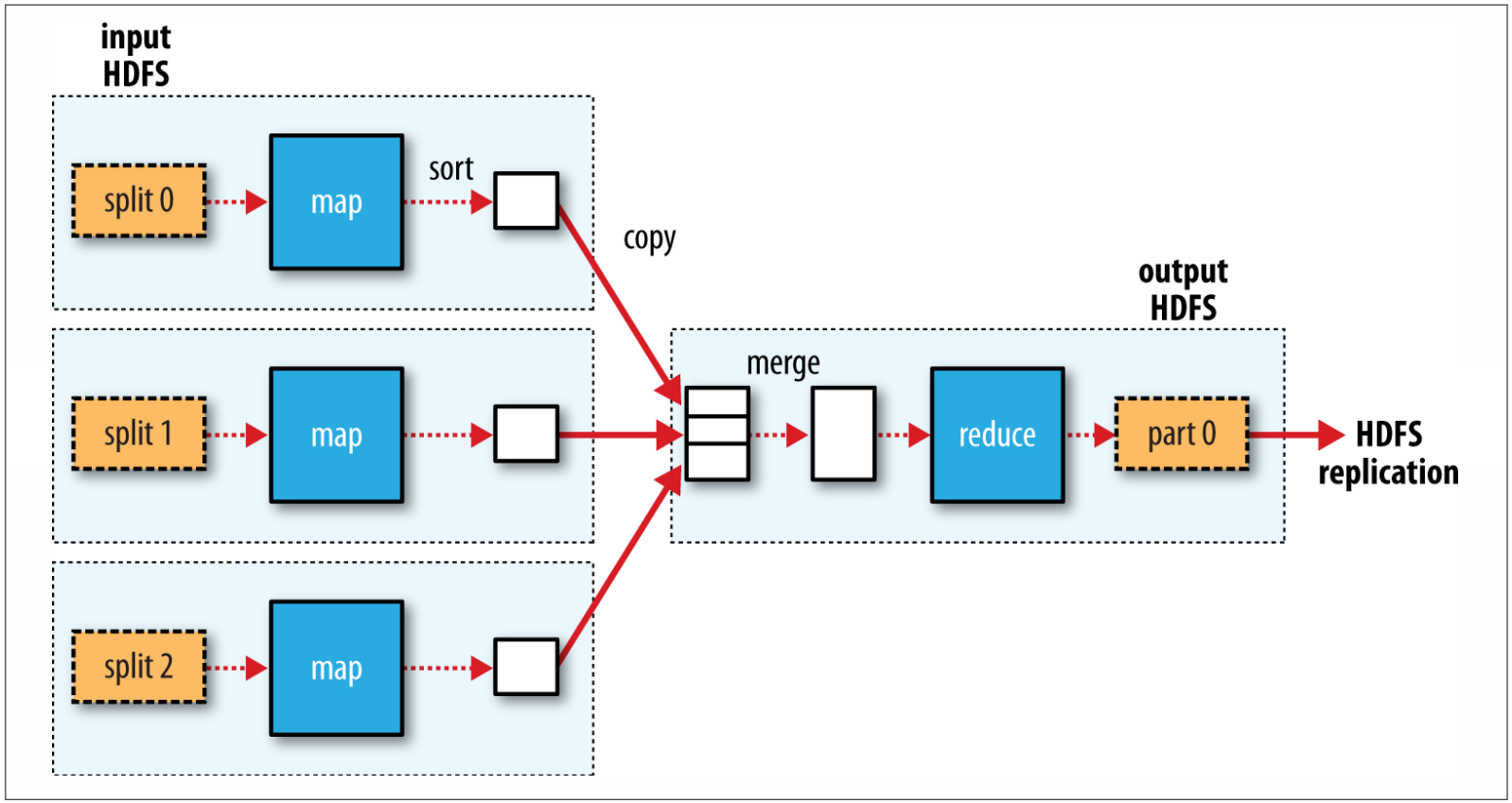
Zero *reduceres*



Fonte: Hadoop—The Definitive Guide, *Tom White*

MapReduce

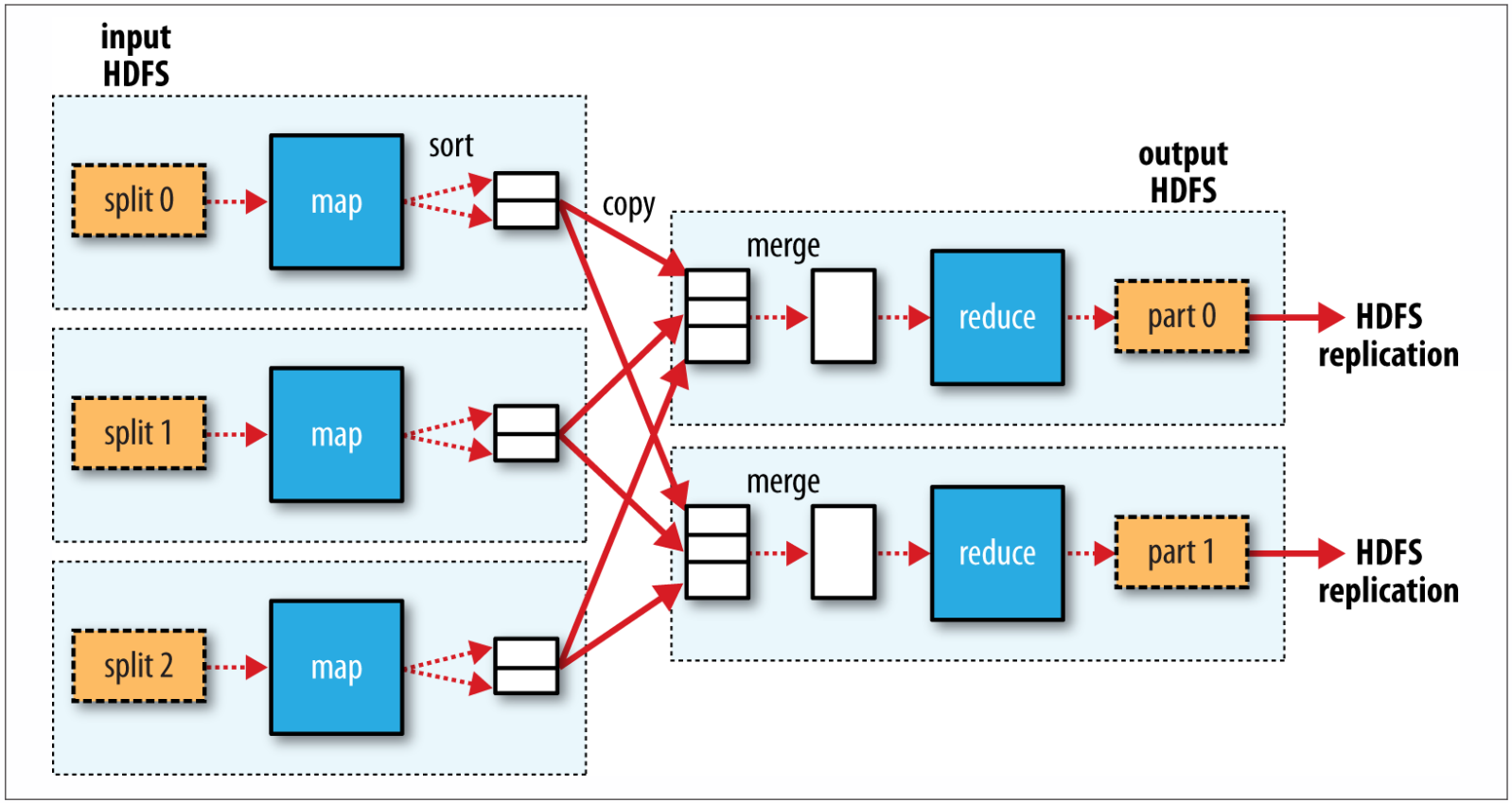
Único *reducer*



Fonte: Hadoop—The Definitive Guide, *Tom White*

MapReduce

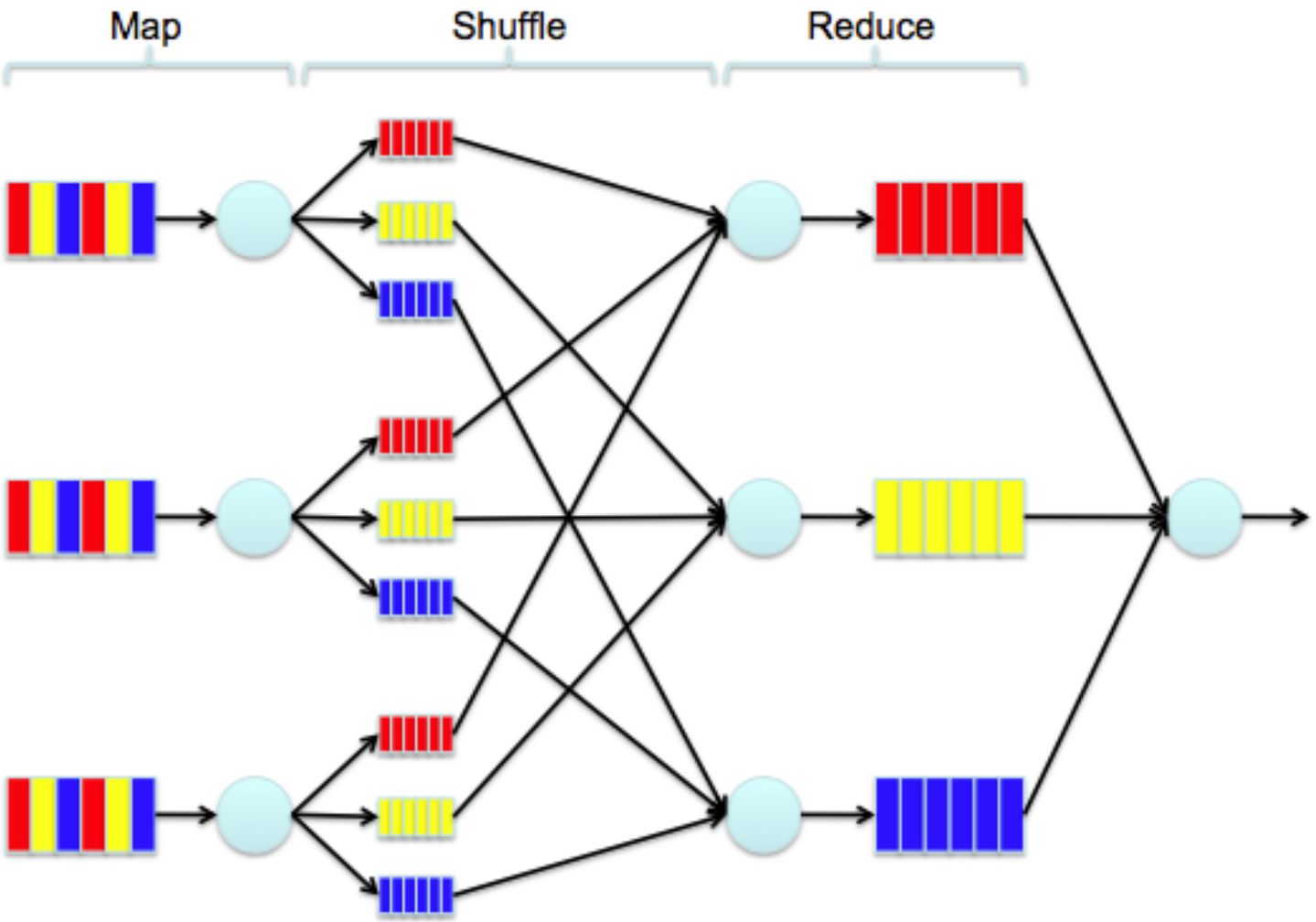
Vários *reduceres*



Fonte: Hadoop—The Definitive Guide, *Tom White*

MapReduce

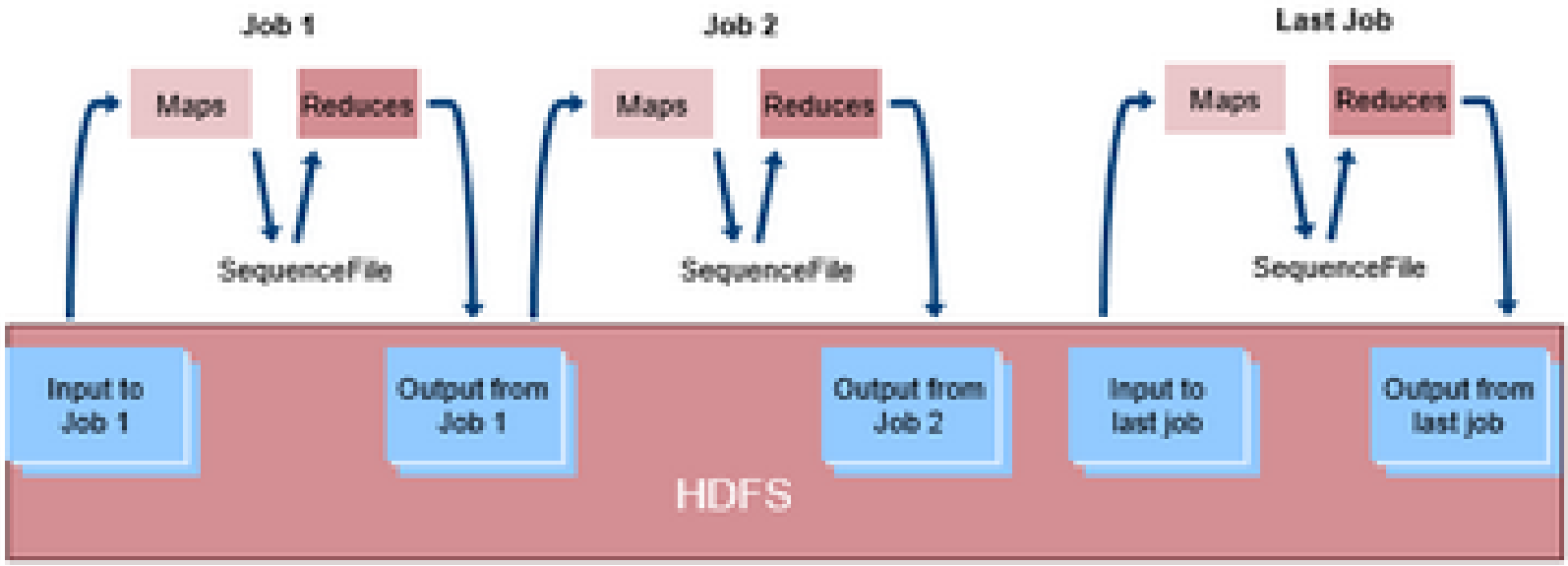
Visão colorida



http://www.cs.uml.edu/~jlu1/doc/source/report/MapReduce.html

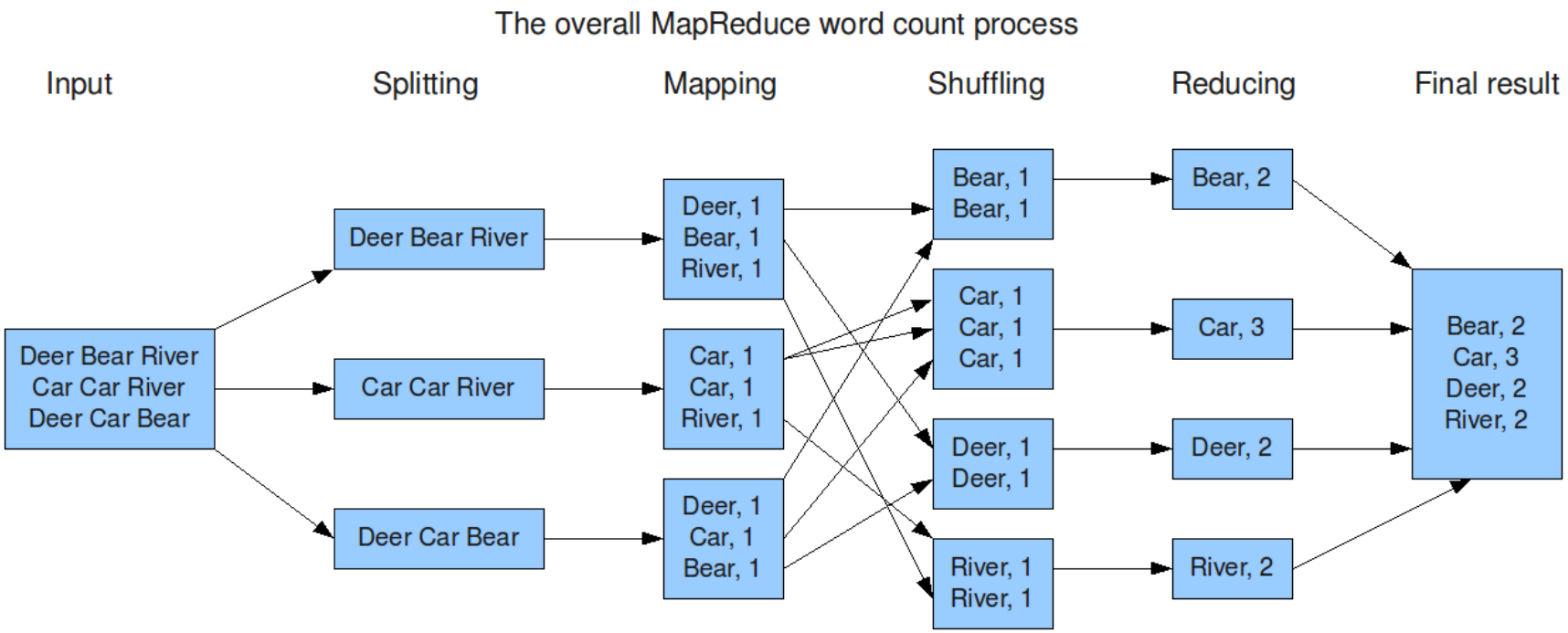
Mapreduce

Várias fases



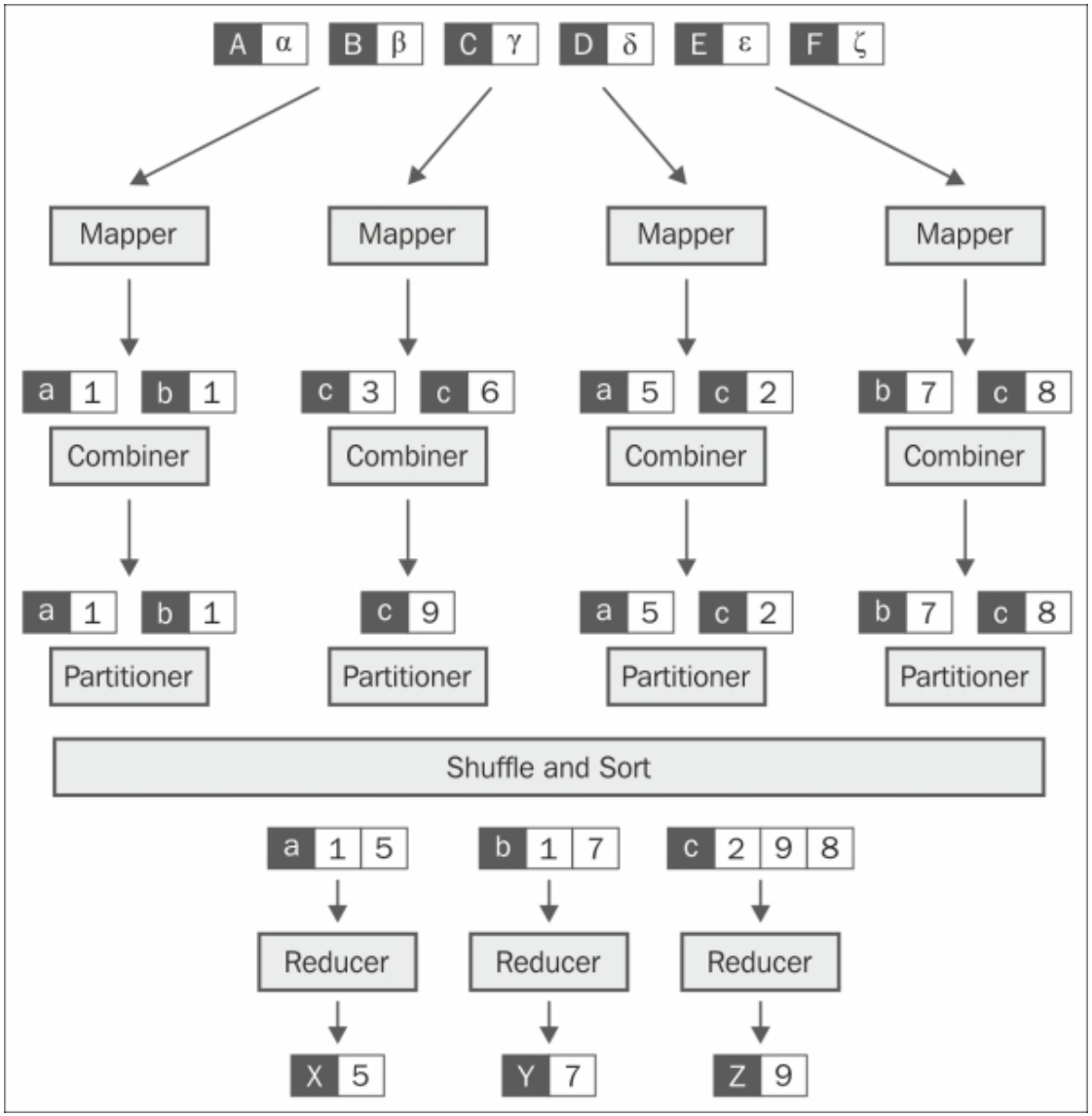
https://www.mapr.com/blog/parallel-and-iterative-processing-machine-learning-recommendations-spark

Word Count



http://www.cs.uml.edu/~jlu1/doc/source/report/img/MapReduceExample.png

Combiners

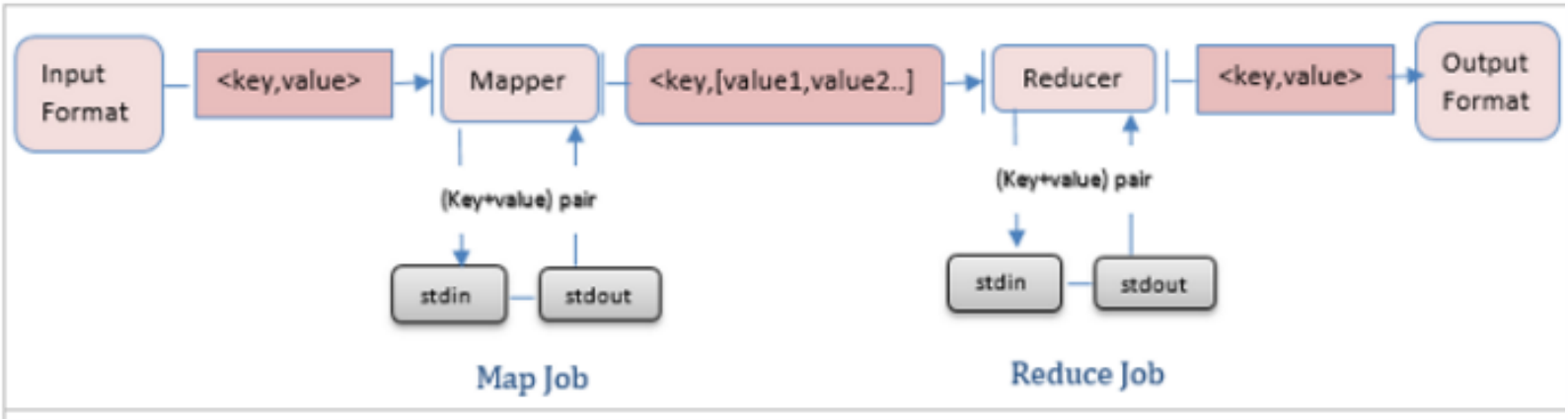


Learning Big Data with Amazon Elastic MapReduce, Vijay Rayapati and Amarkant Singh

Testando o MapReduce

```
$ bin/hadoop dfs -put input /input
$ bin/hadoop jar \
  share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.2.jar \
  wordcount /input /output
$ bin/hadoop dfs -get /output output
```

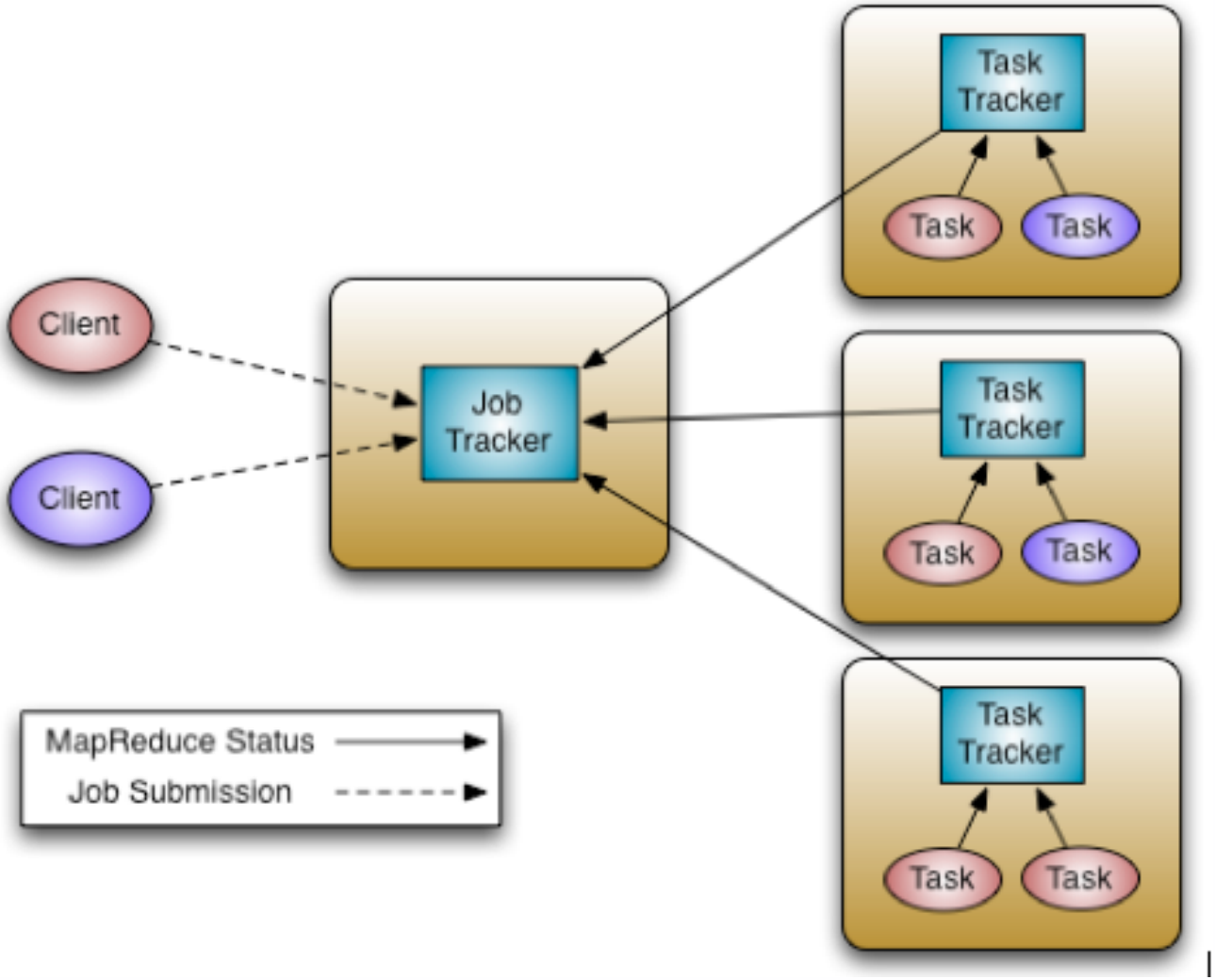
Hadoop Streaming



Fonte: https://acadgild.com/blog/writing-mapreduce-in-python-using-hadoop-streaming/

```
$ bin/hadoop dfs -put input /input
$ bin/hadoop jar \
./share/hadoop/tools/lib/hadoop-streaming-2.7.2.jar \
-input /input  \
-output /output \
-mapper mapper.py \
-reducer reducer.py
```

Hadoop 1.0



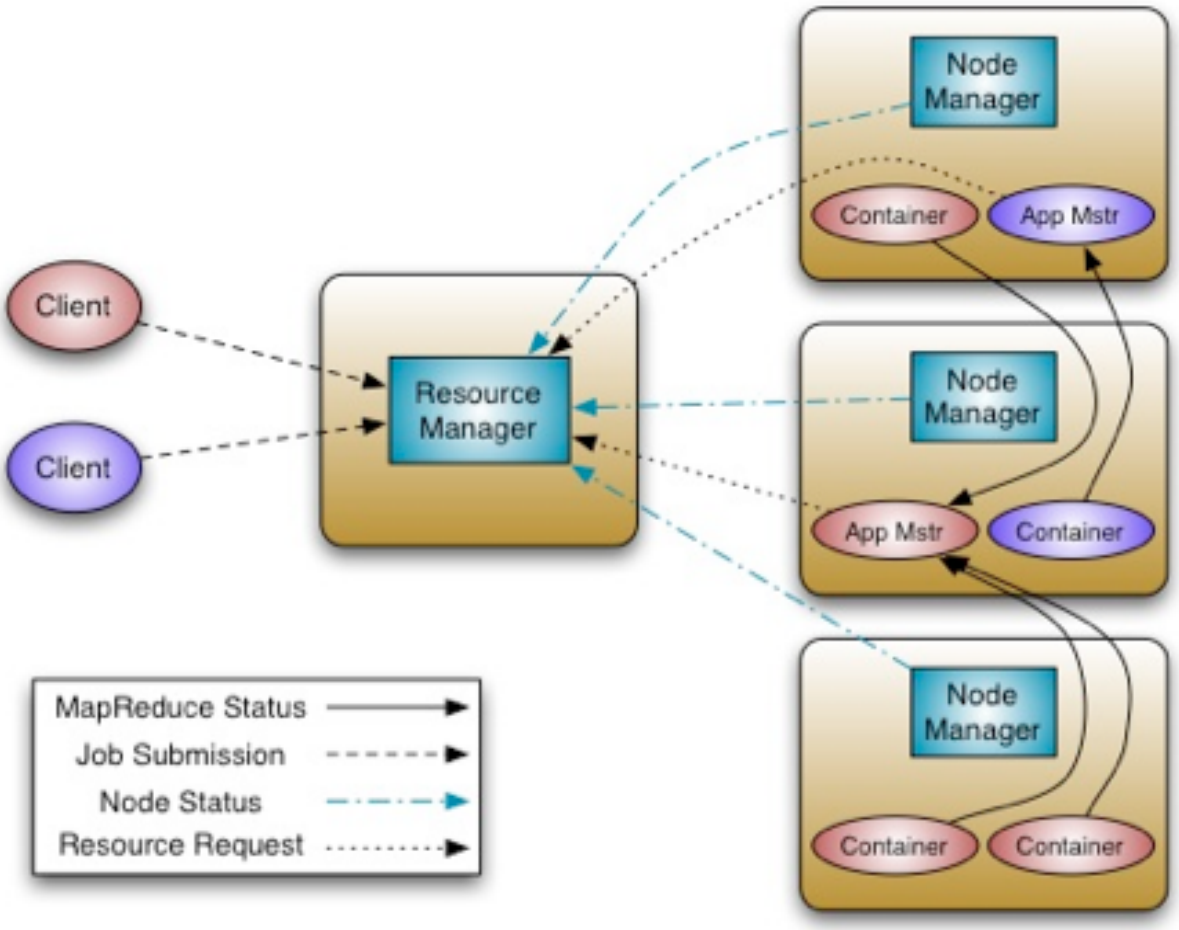
Fonte: `http://hadoop.apache.org`

Hadoop 1.0

- JobTracker
  - Gerenciamento dos Task Trackers (recursos e falhas)
  - Gerenciamento do ciclo de vida dos *jobs*
- TaskTracker
  - iniciar e encerrar *task*
  - enviar *status* para o JobTracker
- Escalabilidade?
- Outros modelos de programação?

YARN

Yet Another Resource Negotiator



Fonte: `http://hadoop.apache.org`

YARN

- JobTracker estava sobrecarregado
  - Gerenciamento de recursos
  - Gerenciamento de aplicações
- Container: abstração que incorpora recursos como cpu, memória, disco, rede...
- ResourceManager
  - Escalonador de recursos
- NodeManager
- ApplicationMaster

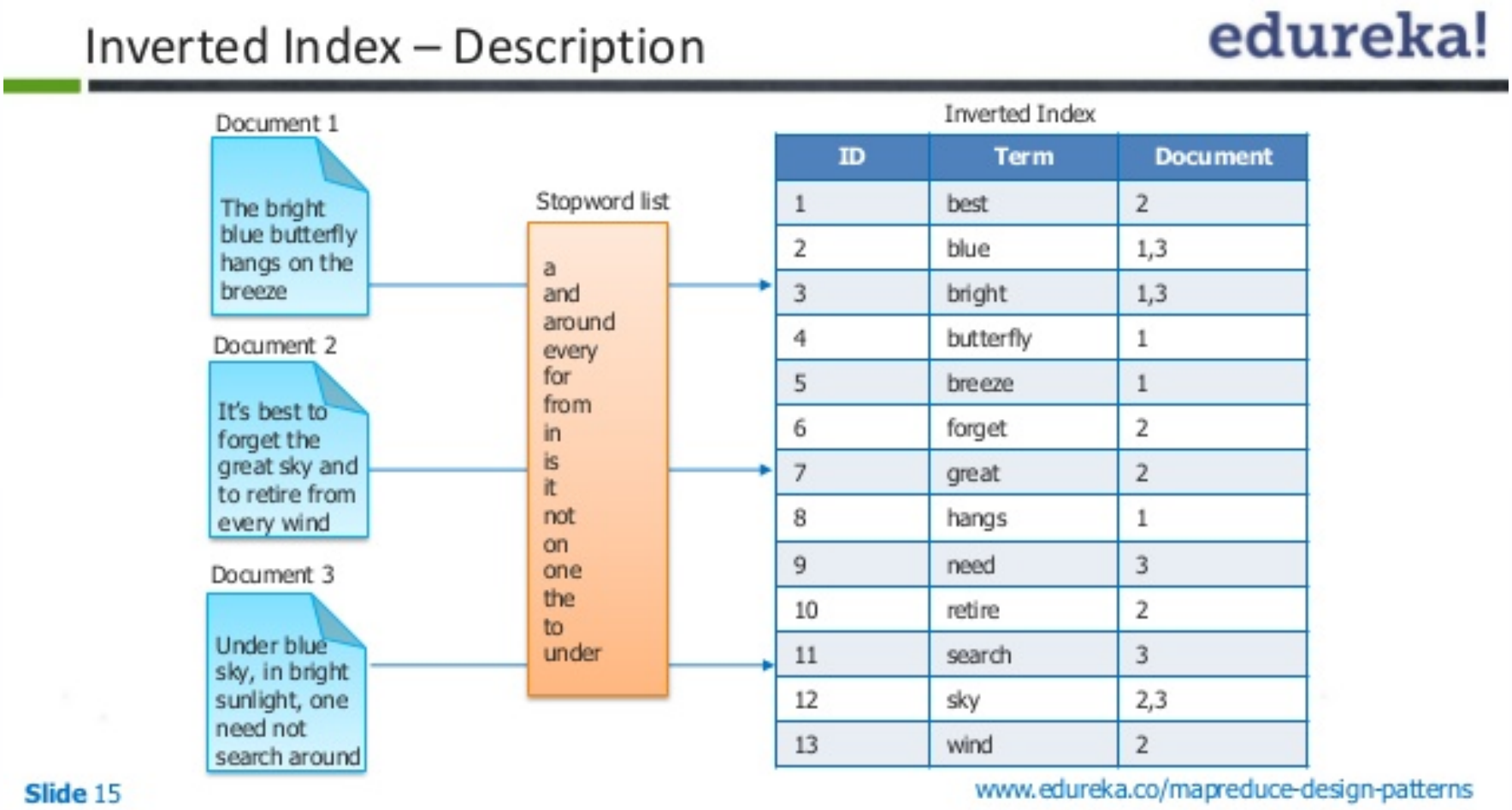
Testando o YARN

```
$ sbin/start-yarn.sh
$ bin/hadoop dfs -put input /input
$ bin/yarn jar \
   share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.2.jar \
   wordcount /input /output
$ bin/hadoop dfs -get /output output
```

- Verifique os *jobs* em `http://localhost:8088/`
- Quando terminar de usar

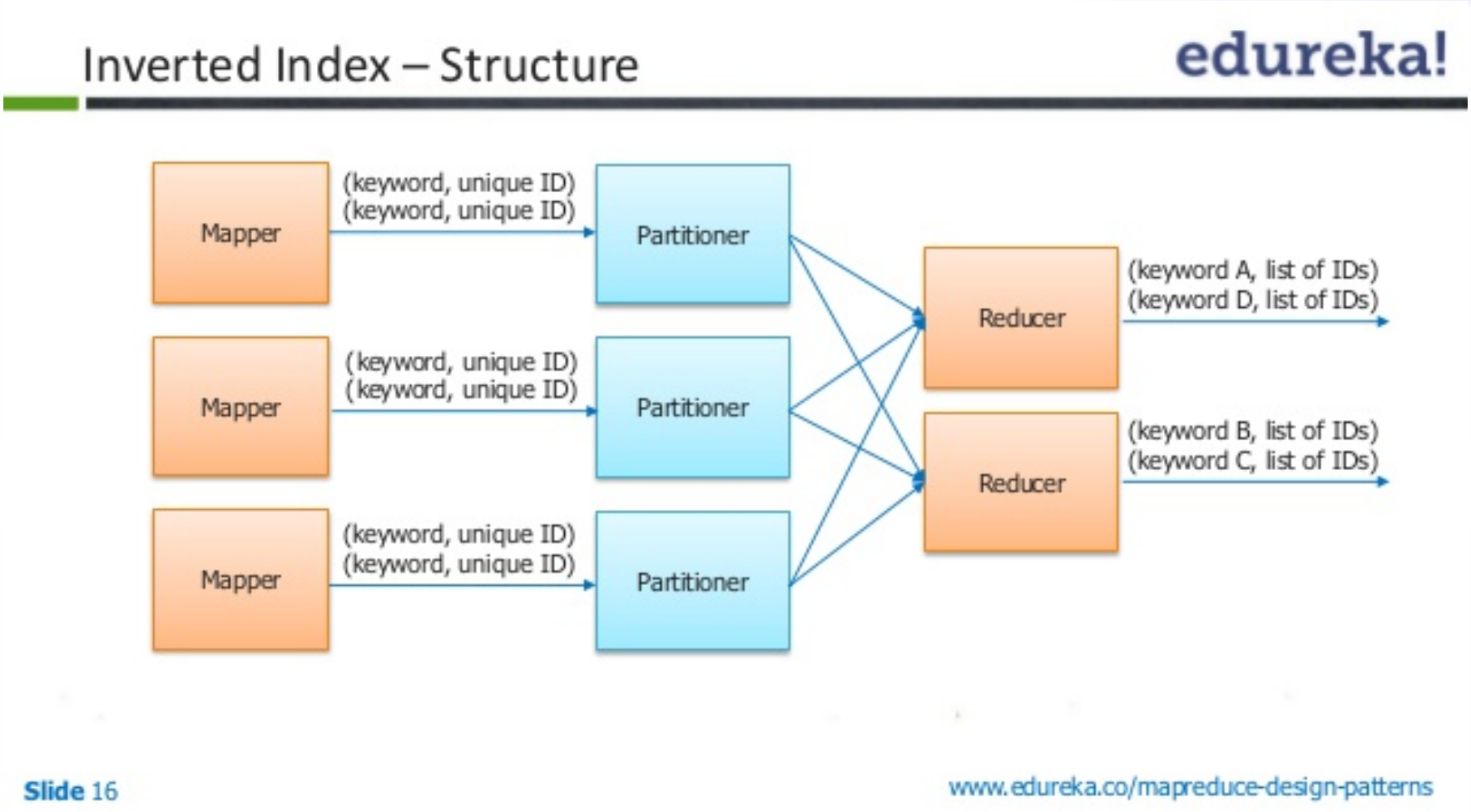
```
$ sbin/stop-yarn.sh
```

Inverted Index



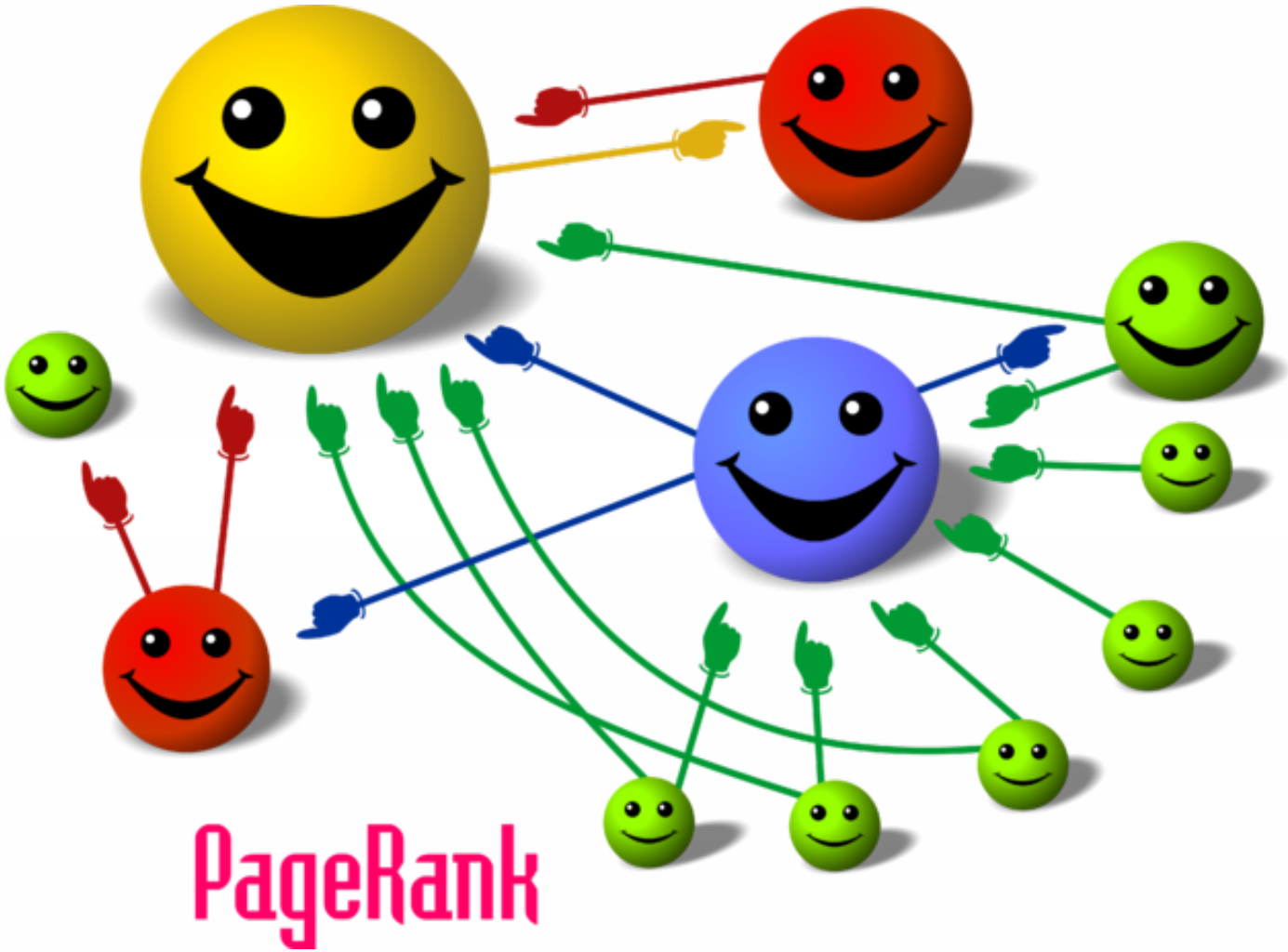
- Máquinas de busca (Google, Wikipedia, Stack Overflow, . . .)

Inverted Index



- Como fazer a distribuição eficiente entre os *reducers*?
- Cuidado com *hot spots* (palavras comuns)

Page Rank



https://commons.wikimedia.org/w/index.php?curid=2776582

Page Rank

- Simula o comportamento de um *random sufer*
  - digita urls de tempos em tempos
  - segue *links* aleatoriamente

$$PR(x) = (1 - d) + d \sum_{i=1}^{N} PR(t_i)/L(t_i)$$

- $PR(x)$ page rank da página x
- d fator de amortecimento
- N número de páginas que apontam para a página x
- $L(t_i)$ número de links distintos que uma página aponta
- Várias iterações até a convergência

Amigos em comum

Lista inicial de amigos

Paulo	João, Juliana, Carlos
João	Paulo, Carlos, Igor, Amanda
Juliana	Paulo, Igor
Carlos	Paulo, João, Igor
Igor	João, Juliana, Carlos
Amanda	João, Maria
Maria	Amanda

Amigos em comum

Mapper: marcando quem já é amigo

Entrada:

Paulo João, Juliana, Carlos

Saída:

<chave, valor>

<Paulo, (João, *)>

<Paulo, (Juliana, *)>

<Paulo, (Carlos, *)>

Amigos em comum

Mapper: novas possibilidades

Entrada:

Paulo João, Juliana, Carlos

Saída:

<chave, valor>
<João, (Juliana, Paulo)>
<João, (Carlos, Paulo)>
<Juliana, (João, Paulo)>
<Juliana, (Carlos, Paulo)>
<Carlos, (João, Paulo)>
<Carlos, (Juliana, Paulo)>

Amigos em comum

Reducer: desconsidera quem já é amigo

<Paulo, (João, Carlos)>
<Paulo, (João, *)>

Amigos em comum

Reducer: verifica possibilidades mais promissoras

<Paulo, (Igor, João)>
<Paulo, (Igor, Juliana)>
<Paulo, (Igor, Carlos)>

Paulo poderia ser amigo de Igor, com três recomendações

Conclusão

- MapReduce
  - Grande revolução
  - Pontos fracos foram surgindo
- Spark: busca por melhor desempenho
- Necessidade de camadas mais altas de abstração

Principais referências

- Projeto Apache Hadoop
- Hadoop: The Definitive Guide, Tom White, 4th Edition, O'Reilly Media