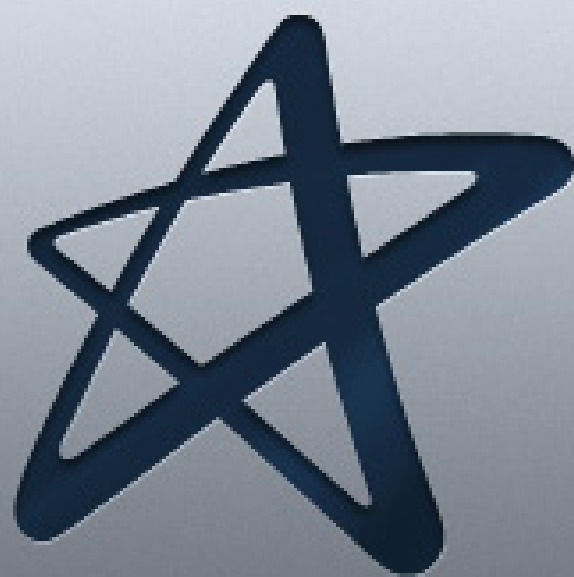


Big Data



Cruzeiro do Sul Virtual
Educação a distância

Material Teórico



**Outros Projetos Importantes do *Hadoop*,
sua Instalação e Execução**

Responsável pelo Conteúdo:

Prof. Dr. Alberto Messias

Revisão Textual:

Prof.^a Dr.^a Selma Aparecida Cesarin

UNIDADE

Outros Projetos Importantes do *Hadoop*, sua Instalação e Execução



- Projeto *Mahout*;
- Projeto *Spark*;
- Processo de Instalação do *Hadoop* em um *Single Node*;
- Iniciando com o *Hadoop*;
- Exemplo de Contagem de Palavras com o *Hadoop*.



OBJETIVO DE APRENDIZADO

- Conhecer outros dois importantes projetos agregados ao *Hadoop*;
- Adquirir conhecimento para a instalação e a configuração de uma instância *Hadoop*;
- Fazer experimentos iniciais de contagem de palavras em bases de Dados reais.



Orientações de estudo

Para que o conteúdo desta Disciplina seja bem aproveitado e haja maior aplicabilidade na sua formação acadêmica e atuação profissional, siga algumas recomendações básicas:



Assim:

- ✓ Organize seus estudos de maneira que passem a fazer parte da sua rotina. Por exemplo, você poderá determinar um dia e horário fixos como seu “momento do estudo”;
- ✓ Procure se alimentar e se hidratar quando for estudar; lembre-se de que uma alimentação saudável pode proporcionar melhor aproveitamento do estudo;
- ✓ No material de cada Unidade, há leituras indicadas e, entre elas, artigos científicos, livros, vídeos e *sites* para aprofundar os conhecimentos adquiridos ao longo da Unidade. Além disso, você também encontrará sugestões de conteúdo extra no item **Material Complementar**, que ampliarão sua interpretação e auxiliarão no pleno entendimento dos temas abordados;
- ✓ Após o contato com o conteúdo proposto, participe dos debates mediados em fóruns de discussão, pois irão auxiliar a verificar o quanto você absorveu de conhecimento, além de propiciar o contato com seus colegas e tutores, o que se apresenta como rico espaço de troca de ideias e de aprendizagem.

Projeto *Mahout*

O Projeto *Mahout* é um Projeto de Código Livre da Fundação *Apache* que possui uma biblioteca de implementação de algoritmos para aprendizagem de máquina.

De acordo com Giacomelli (2013), o objetivo do Projeto *Mahout* é ser uma escolha de ferramenta para aprendizado por máquina para Processamento de conjuntos de Dados extremamente grandes, para execução em *clusters* de instâncias de *Hadoop* ou numa única máquina.

O *Mahout* é uma ferramenta desenvolvida em Linguagem de Programação *Java* dentro do projeto de computação distribuída *Hadoop*.

O Projeto *Mahout* possui implementações de algoritmos de classificação e *clustering*, como o algoritmo *K-means*, que possui grande relevância para o curso.

O Projeto *Mahout* possui implementações de diversos algoritmos em sua execução direta no *Hadoop*, com o *Mapreduce*, algoritmos em *Spark*, além de poder usar *frameworks* *H2O* e *Flink*.

Usando o *Mapreduce*, há os seguintes algoritmos implementados:

- **Classificação:**
 - » *Naive Bayes*;
 - » *Hidden Markov Models*;
 - » *Logistic Regression*;
 - » *Random Forest*.
- **Clustering:**
 - » *k-Means*;
 - » *Canopy*;
 - » *Fuzzy k-Means*;
 - » *Streaming KMeans*;
 - » *Spectral Clustering*.
- **Processamento pós-clustering:**
 - » *Cluster Dumper tool*;
 - » *Cluster visualisation*.
- **Redução de dimensionalidade:**
 - » *Singular value decomposition*;
 - » *Algoritmo Lanczos*;
 - » *Stochastic SVD*;
 - » *PCA (via Stochastic SVD)*;
 - » *QR Decomposition*.

- **Outras implementações de suporte:**

- » *Row Similarity Job;*
- » *Collocations;*
- » *Sparse TF-IDF Vectors em Textos;*
- » *XML Parsing;*
- » *Email Archive Parsing;*
- » *Evolutionary Processes.*

Não se preocupe nesse momento com os algoritmos aqui citados, pois são ilustrações do que se tem disponível no projeto *Mahout*. No momento adequado, você terá contato com alguns desses algoritmos.

Projeto *Spark*

O Projeto *Spark* foi desenvolvido inicialmente pelo AMPLab da Universidade de Berkeley, usando os mesmos conceitos do projeto *Mesos*, proposto pela mesma Universidade. Como característica principal, o *Spark* otimiza os processos executados por meio do *Hadoop MapReduce* em memória, o que permite um desempenho em torno de 100 vezes mais rápido que o *MapReduce* com *Hadoop* tradicional.

Em geral, o *Spark* propõe uma plataforma de processamento ao tradicional oferecido pelo *Hadoop*. Ele oferece ou possibilita implementações em linguagens *Casa*, *Java*, *R* e *Tom*, ainda permite interface com as plataformas tradicionais *Hadoop* como: *HDFS*, *HBase*, *Cassandra*, *MapR-DB*, *MongoDB* e Amazonas S3.

Segue o diagrama *Spark*, que ilustra graficamente sua estrutura, com seus principais componentes:

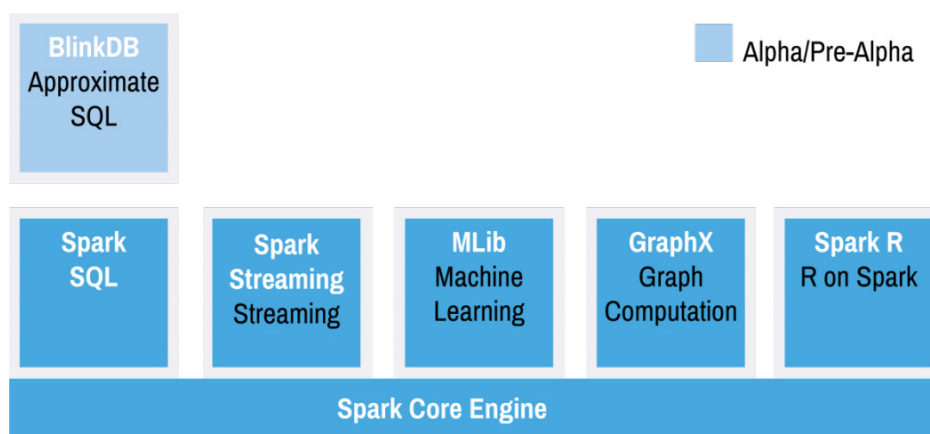


Figura 1 – Diagrama *Spark*

Fonte: Adaptado de Scott, 2015

Segue a descrição simples de cada um dos componentes principais:

Apache Spark Core

O *Spark Core* é a plataforma principal de execução; é base para todas as outras funcionalidades. Ela fornece a execução em memória e faz referência a conjuntos de dados em Sistemas de Armazenamento externos e distribuídos.

Spark SQL

O *Spark SQL* é um componente importante do *Spark Core*, que permite a abstração de dados chamada *Schema RDD*, que fornece suporte para dados estruturados e semi-estruturados.

Spark Streaming

Spark Streaming aproveita a capacidade de processamento rápido da *Spark Core* para executar os processamentos. Permite a execução e a transformações *RDD (Solids Distributed Datasets)* nesses pedaços menores de Dados e lotes de Processamento.

MLlib (Biblioteca de Machine Learning)

MLlib é uma biblioteca de aprendizagem por máquina distribuída que roda sobre a arquitetura de *Spark* baseada em memória distribuída. O *SparkMLlib* é nove vezes mais rápido que a versão baseada em HDFS e disco do *Hadoop* do *Apache Mahout* (antes do *framework Mahout* implementar a interface com *Spark*). Cabe destacar que a biblioteca *MLib* possui uma grande quantidade de funções e algoritmos para aprendizado por máquina.

GraphX

GraphX é uma estrutura distribuída de processamento de gráficos sob a arquitetura *Spark*. Ele fornece uma *API* para computação gráfica que pode modelar os gráficos definidos pelo usuário usando a *API* de abstração *Pregel*. Ele também fornece um tempo de execução otimizado para essa abstração.

Spark R

Essa funcionalidade permite que usuários da plataforma ou *software R* utilizem funções ou funcionalidades *Spark* de maneira mais usual.

Resilient Distributed Datasets

O *Resilient Distributed Datasets (RDD)* é o conceito central da plataforma *Spark*. Foi desenvolvido para suportar o armazenamento de dados na memória e distribuído ou num *cluster*, que implementa sua tolerância a falhas, devido, em parte, ao seu rastreamento de dados brutos ou processamentos.

Segue um trecho de código de exemplo de *Word Count* em *Scala* para execução em *Spark*:

```
import org.apache.spark.SparkContext
import org.apache.spark.SparkContext._
val txtFile = "/ola.txt"
val txtData = sc.textFile(txtFile)
txtData.cache()
txtData.count()

val wcData = txtData.flatMap(l => l.split(" ")).map(word => (word, 1)).
  reduceByKey(_ + _)

wcData.collect().foreach(println)
```

Esse código executa a contagem de palavras do artigo de entrada "ola.txt", presente na terceira linha.



Caso queira se aprofundar no tema de *Spark*, segue o *link* com importantes definições em inglês e o *link* para introdução à Linguagem *Scala*, bastante utilizada no *Spark*. Disponível em:

- <https://goo.gl/9EMkrY>
- <https://goo.gl/KoZM6v>

Processo de Instalação do *Hadoop* em um *Single Node*

Aqui será descrito o processo de instalação e configuração do *Hadoop* para um único nó utilizando o *Linux Debian*:

- Atualizar o Sistema, trocar o perfil do usuário no Sistema para administrador (*root*) e digitar os comandos:

```
apt-get update
```

```
apt-get upgrade
```

- Como administrador, adicionar um grupo *hadoop* e um usuário *hadoop-user* a esse grupo; há possibilidade de deixar as informações solicitadas por *adduser* em branco, com exceção da senha:

```
add group hadoop
```

```
adduser -ingroup hadoop hadoop-user
```

- Instalar o *Java* 8, ainda como administrador do Sistema. Remover o *openjdk*; configurar o *Java* 8:

```
apt-purge openjdk*

echo "deb http://ppa.launchpad.net/webupd8team/java/ubuntu trusty/main"
| tee /etc/apt/sources.list.d/webupd8team-java.list

echo "deb-src http://ppa.launchpad.net/webupd8team/java/ubuntu trusty-
main" | tee -a /etc/apt/sources.list.d/webupd8team-java.list

apt-key adv --keyserver keyserver.ubuntu.com --recv-keys EEA14886
```

- Mais uma vez, atualizar o Sistema e instalar:

```
apt-get update

apt-get install -y oracle-java8-installer
```

- Aceitar a licença de *software* e verificar se a versão correta foi instalada:

```
java -version
```

Configurar a variável de ambiente `$JAVA_HOME`, ainda usando o usuário administrador do Sistema:

```
sh -c 'echo "export JAVA_HOME=/usr/local/JDK..." >> /etc/profile'

source /etc/profile
```

O trecho "**JDK...**" devem ser trocada pelo local onde foi instalado o **JDK**.

- Comunicações do *Hadoop* são criptografadas via SSH; portanto, o servidor foi instalado e configurado:

```
apt-get install openssh-server
```

- O *hadoop-user* foi associado a um par de chaves e, subsequentemente, concedeu o seu acesso à máquina local. Para configurar o SSH, há a necessidade de mudar para o *hadoop-user*:

```
su - hadoop-user

ssh-keygen -t rsa -P ""

cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
```

- O *hadoop-user* será capaz de acessar via **ssh** o *localhost* sem fornecer senha:

```
ssh localhost
```

- Voltar ao administrador da máquina, fazer o *download* do *Hadoop*, descompactar e mover para “/usr/local”; adicionar o *link* usando o nome *hadoop* alterando do conteúdo para o *hadoop-user*:

```
wget http://mirror.nohup.it/apache/hadoop/common/hadoop-2.7.1/hadoop-2.7.1.tar.gz

tar xzf hadoop-2.7.1.tar.gz

rm hadoop-2.7.1.tar.gz

mv hadoop-2.7.1 /usr/local

ln -sf /usr/local/hadoop-2.7.1/ /usr/local/hadoop

chown -R hadoop-user:hadoop /usr/local/hadoop-2.7.1/
```

- Mudar para o *hadoop-user* e adicionar as seguintes linhas no final de ~/.*bashrc*:

```
# Set Hadoop-related environment variables
export HADOOP_PREFIX=/usr/local/hadoop
export HADOOP_HOME=/usr/local/hadoop
export HADOOP_MAPRED_HOME=${HADOOP_HOME}
export HADOOP_COMMON_HOME=${HADOOP_HOME}
export HADOOP_HDFS_HOME=${HADOOP_HOME}
export YARN_HOME=${HADOOP_HOME}
export HADOOP_CONF_DIR=${HADOOP_HOME}/etc/hadoop
# Native path
export HADOOP_COMMON_LIB_NATIVE_DIR=${HADOOP_PREFIX}/lib/native
export HADOOP_OPTS="-Djava.library.path=${HADOOP_PREFIX}/lib/native"
# Java path
export JAVA_HOME="/usr"
# Add Hadoop bin/ directory to PATH
export PATH=$PATH:$HADOOP_HOME/bin:$JAVA_PATH/bin:$HADOOP_HOME/sbin
```

Figura 2

- Para que as novas variáveis de ambiente estejam em vigor, é preciso recarregar o arquivo **.bashrc**:

```
source .bashrc
```

- Modificar arquivos de configurações dentro /usr/local/hadoop/etc/hadoop. Todos eles seguem o formato XML e *configuration*. As atualizações referem-se ao nó de nível superior. Especificamente:

- » No arquivo em *yarn-site.xml*:

```
<configuration>
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
</configuration>
```

Figura 3

- » No arquivo em *core-site.xml*:

```
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://localhost:9000</value>
  </property>
</configuration>
```

Figura 4

- Em *mapred-site.xml* (likey para ser criado através de *cmapred-site.xml*.
template mapred-site.xml):

```
<configuration>
  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>
</configuration>
```

Figura 5

» No arquivo em *hdfs-site.xml*:

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
  <property>
    <name>dfs.namenode.name.dir</name>
    <value>file:/usr/local/hadoop/yarn_data/hdfs/namenode</value>
  </property>
  <property>
    <name>dfs.datanode.data.dir</name>
    <value>file:/usr/local/hadoop/yarn_data/hdfs/datanode</value>
  </property>
</configuration>
```

Figura 6

- Isso requer a criação manual dos dois diretórios especificados nos dois últimos value nos XML:

```
mkdir -p /usr/local/hadoop/yarn_data/hdfs/namenode
```

```
mkdir -p /usr/local/hadoop/yarn_data/hdfs/datanode
```

- Insira o “/usr/local/java/JDK...” na variável *JAVA_HOME*, no arquivo */usr/local/hadoop/etc/hadoop/hadoop-env.sh*.
- Agora você poderá iniciar o serviço do *Hadoop*. Para isso, use o comando:

```
/usr/local/hadoop/sbin/start-all.sh
```

- Formatar o sistema de arquivos, operação a ser executada como *hadoop-user*:

```
hdfsnamenode -format
```

Note que a instalação aqui descrita foi para um único nó; porém, pode-se encontrar facilmente na documentação do *Hadoop* a instalação e a configuração para *clusters* de *hadoop*.

Por exemplo, no *link*:



Hadoop Cluster Setup. Disponível em: <https://goo.gl/KbH724>.

O processo de instalação aqui descrito é importante como referência para um administrador; porém, há, inúmeras máquinas virtuais que podem ser baixadas com todas as funcionalidades prontas para o uso, como, por exemplo, da *Cloudera* ou *Hortonworks*. Uma alternativa é usar as máquinas prontas nas plataformas de computação em nuvem, pois o *hadoop* está presente em todas as grandes plataformas, como Google, IBM, Amazon e Microsoft Azure.

Iniciando com o *Hadoop*

O *Hadoop* possui algumas ferramentas administrativas interessantes, que podem auxiliar no processo de administração diária.

← → ↻ 192.168.1.124:50070/dfshealth.html#tab-overview

Hadoop Overview Datanodes Datanode Volume Failures Snapshot Startup Progress Utilities ▾

Browse the file system
Logs

Overview 'localhost:9000' (active)

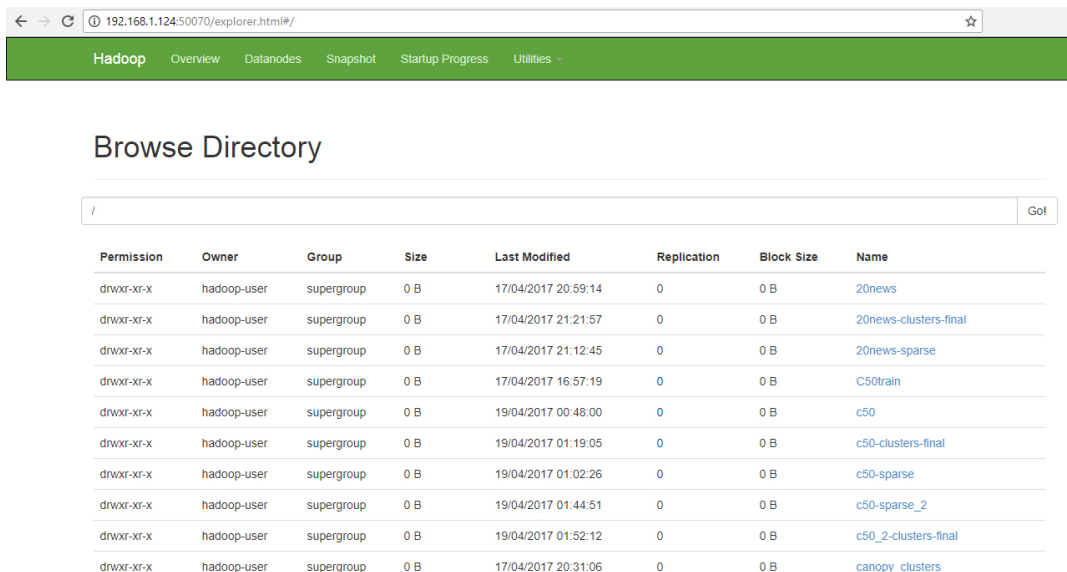
Started:	Thu Apr 13 12:23:43 BRT 2017
Version:	2.7.1, r15ecc87ccf4a0228f35af08fc56de536e6ce657a
Compiled:	2015-06-29T06:04Z by jenkins from (detached from 15ecc87)
Cluster ID:	CID-073ea671-03bb-4231-a757-9858b31b3c04
Block Pool ID:	BP-7182094-127.0.1.1-1488475392793

Summary

Security is off.
Safemode is off.
27160 files and directories, 26601 blocks = 53761 total filesystem object(s).
Heap Memory used 54.28 MB of 70.8 MB Heap Memory. Max Heap Memory is 966.69 MB.
Non Heap Memory used 47.45 MB of 47.63 MB Committed Non Heap Memory. Max Non Heap Memory is 214 MB.

Figura 7

Há uma URL que permitirá ao administrador visualizar todos os arquivos presentes no HDFS, indo em “*Utilities/Browse the file system*”, conforme se observa na Figura a seguir.



Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	hadoop-user	supergroup	0 B	17/04/2017 20:59:14	0	0 B	20news
drwxr-xr-x	hadoop-user	supergroup	0 B	17/04/2017 21:21:57	0	0 B	20news-clusters-final
drwxr-xr-x	hadoop-user	supergroup	0 B	17/04/2017 21:12:45	0	0 B	20news-sparse
drwxr-xr-x	hadoop-user	supergroup	0 B	17/04/2017 16:57:19	0	0 B	C50train
drwxr-xr-x	hadoop-user	supergroup	0 B	19/04/2017 00:48:00	0	0 B	c50
drwxr-xr-x	hadoop-user	supergroup	0 B	19/04/2017 01:19:05	0	0 B	c50-clusters-final
drwxr-xr-x	hadoop-user	supergroup	0 B	19/04/2017 01:02:26	0	0 B	c50-sparse
drwxr-xr-x	hadoop-user	supergroup	0 B	19/04/2017 01:44:51	0	0 B	c50-sparse_2
drwxr-xr-x	hadoop-user	supergroup	0 B	19/04/2017 01:52:12	0	0 B	c50_2-clusters-final
drwxr-xr-x	hadoop-user	supergroup	0 B	17/04/2017 20:31:06	0	0 B	canopy_clusters

Figura 8

Por meio dos *links* presentes nessa ferramenta, é possível observar, por exemplo, o processo de inicialização do *hadoop*, *LOGs*, os datanodes e seus respectivos tamanhos no Sistema de Arquivos, dentre outras informações.

Há, ainda, uma página de administração que exibe uma série de outras informações a respeito das execuções de processamentos no *hadoop*.

<

Figura 9

A interface exibe as aplicações em execução e seu respectivo tempo e porcentagem de conclusão do processamento.

É importante o administrador conhecer alguns comandos para o gerenciamento do HDFS, grande parte herdados de comandos Unix/Linux, tais como:

- **ls** – para listagem de arquivos;
- **rm** – para remover arquivos;

- **rmdir** – para remover diretórios;
- **cat** – para exibir conteúdos de arquivos;
- **cp** – para copiar arquivos;
- **copyFromLocal** – para enviar arquivos para o HDFS;
- **find** – para buscar arquivos;
- **chmod** – para editar permissões sobre um arquivo;
- **chow** – para alterar o dono de um arquivo;
- **mv** – para mover um arquivo entre diretórios;
- **moveFromLocal** – para mover um arquivo para o HDFS;
- **appendToFile** – para inserir conteúdos no final do arquivo.



Esses estão dentre uma lista completa, que se encontra facilmente nos documentos presentes na *site* do fabricante. Disponível em: <https://goo.gl/BBdAzK>.

Exemplo de Contagem de Palavras com o *Hadoop*

Vamos a um exemplo trivial de contagem de palavras. Assume-se que deve-se estar autenticado com o usuário *hadoop-user* em nossa instância de *Hadoop*.

O primeiro passo será copiar para o HDFS da instância de *Hadoop* que estamos executando; nesse caso, com o seguinte comando:

```
hadoopfs -copyFromLocal ola.txt /
```

O comando copia o arquivo **ola.txt** para a raiz do Sistema de Arquivos HDFS. Pela interface *WEB*, você consegue se certificar que a cópia do arquivo ocorreu, conforme se pode observar na Figura a seguir.

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	hadoop-user	supergroup	0 B	22/09/2017 02:17:35	0	0 B	C50
drwxr-xr-x	hadoop-user	supergroup	0 B	22/09/2017 02:03:08	0	0 B	ola.output
-rw-r--r--	hadoop-user	supergroup	110 B	22/09/2017 02:01:15	1	128 MB	ola.txt
drwx-----	hadoop-user	supergroup	0 B	22/09/2017 02:02:41	0	0 B	tmp

Figura 10

O Conteúdo do arquivo é:

“Olá mundo hadoop.
Cruzeiro do Sul Virtual Educação a distância.
Professor Alberto.”

Para se certificar que o arquivo foi copiado, você pode acessar a página de navegação em arquivos

Nos diretórios de exemplos do *hadoop*, há uma implementação de contagem de palavras no arquivo “*hadoop-mapreduce-examples-2.7.4.jar*”, que está no diretório “*hadoop/share/hadoop/mapreduce/*”.

O comando pode ser executado da seguinte maneira:

```
hadoopjar hadoop-mapreduce-examples-2.7.4.jar wordcount /ola.txt /ola_output
```

O comando é dividido da seguinte maneira: “*hadoopjar*” informando que será executado um arquivo no formato jar. Nesse caso, o “*hadoop-mapreduce-examples-2.7.4.jar*”.

Os parâmetros seguintes são, executar a função “*wordcount*” passando como parâmetros o arquivo ou diretório de entrada e arquivo de saída.

Para resgatar o arquivo de saída, pode-se copiá-lo por comando no HDFS ou fazer o *download* através da interface *WEB*.

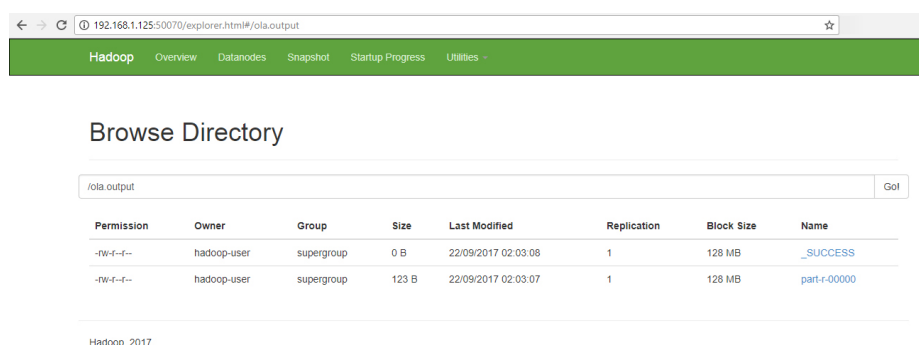


Figura 11

O arquivo de saída o “*part-r-00000*”. Segue o arquivo de saída do comando executado:

Tabela 1

Alberto.	1
Cruzeiro	1
Educação	1
Olá	1
Professor	1
Sul	1

a	1
distância	1
do	1
hadoop	1
mundo	1
Virtual	1

Na primeira coluna, o arquivo de saída possui a palavra e na segunda coluna a frequência com que a palavra ocorreu no texto. Note que o comando não faz nenhuma limpeza ou pré-processamento na Base de Dados. Nesse caso, ele entendeu como uma única palavra o “*hadoop*”.

Vamos a um segundo exemplo, com uma base de dados real, o primeiro passo será baixar a base de dados, a qual iremos utilizar na execução de exemplo.



A base de exemplo se trata de uma base de postagem de 50 autores em *Blogs*. Ela pode ser baixada no link: <https://goo.gl/RWrNAR>.

Descompacte o arquivo *C50.zip* baixado do site por meio do comando:

```
unzip C50.zip
```

```
hadoopfs -copyFromLocal C50/ /
```

Verifique, pelo navegador, se os arquivos foram copiados, se agora existem as duas pastas *C50test* e *C50train*, conforme se observa na figura a seguir:

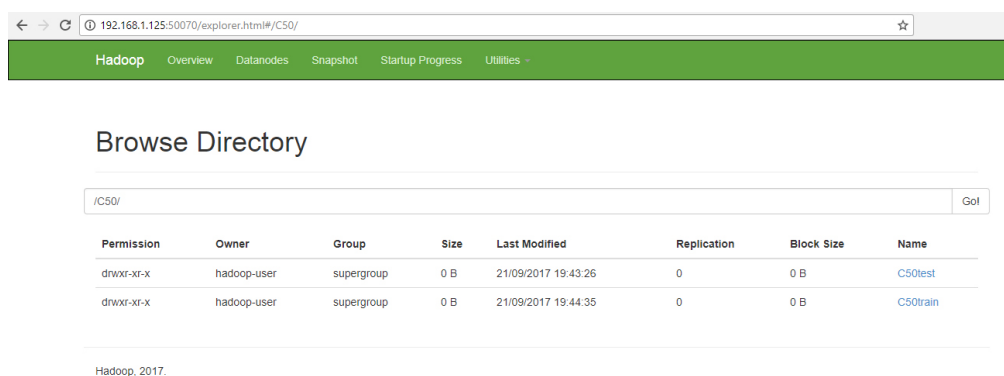


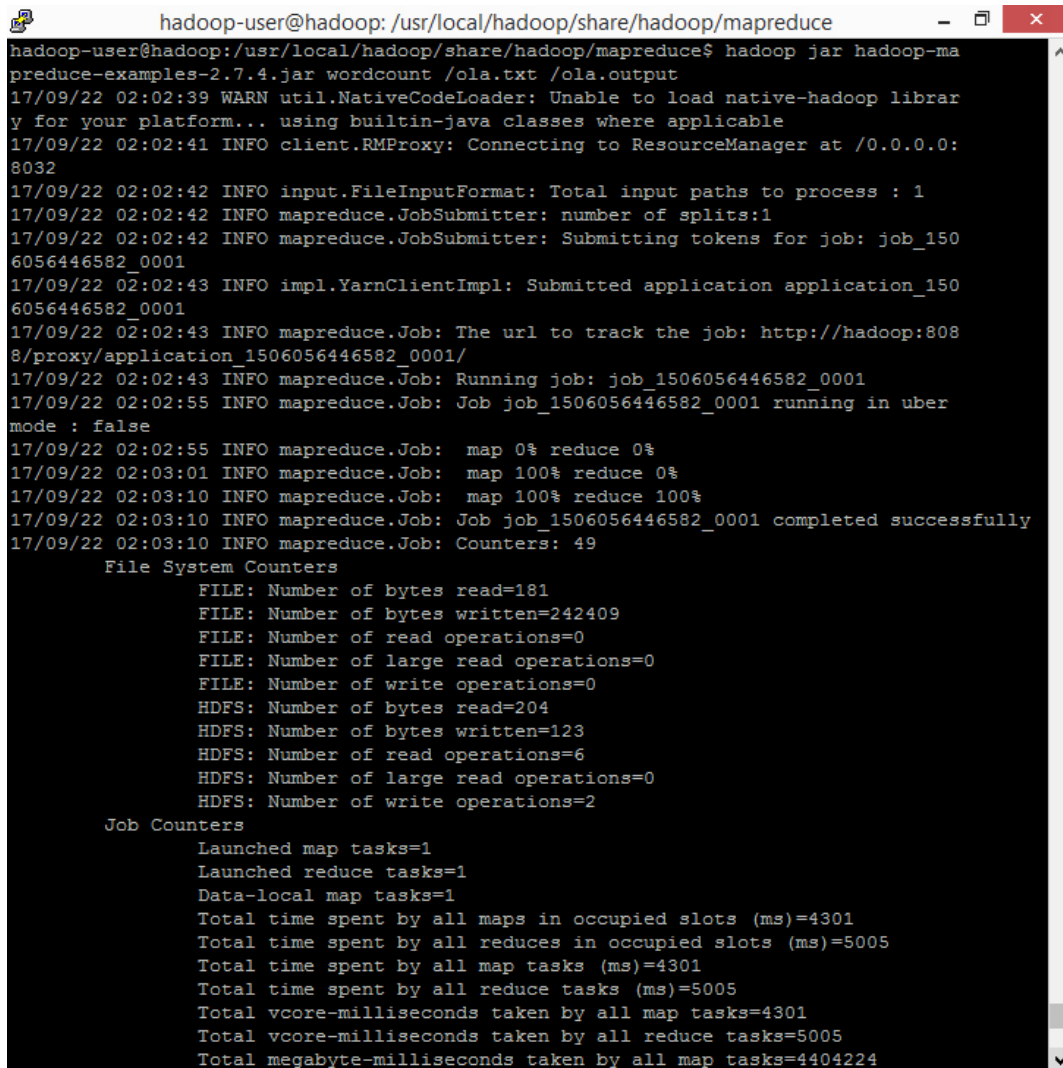
Figura 12

Vamos executar o comando de contagem de palavras para um diretório de um dos autores e na base de testes.

```
hadoopjar hadoop-mapreduce-examples-2.7.4.jar wordcount /C50/C50test/  
AaronPressman/ C50/output
```

Ou, caso se queira, executar para a base inteira:

```
hadoopjar hadoop-mapreduce-examples-2.7.4.jar wordcount /C50/C50test/*/*
/C50/output
```



```
hadoop-user@hadoop: /usr/local/hadoop/share/hadoop/mapreduce
hadoop-user@hadoop: /usr/local/hadoop/share/hadoop/mapreduce$ hadoop jar hadoop-mapreduce-examples-2.7.4.jar wordcount /ola.txt /ola.output
17/09/22 02:02:39 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
17/09/22 02:02:41 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
17/09/22 02:02:42 INFO input.FileInputFormat: Total input paths to process : 1
17/09/22 02:02:42 INFO mapreduce.JobSubmitter: number of splits:1
17/09/22 02:02:42 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1506056446582_0001
17/09/22 02:02:43 INFO impl.YarnClientImpl: Submitted application application_1506056446582_0001
17/09/22 02:02:43 INFO mapreduce.Job: The url to track the job: http://hadoop:8088/proxy/application_1506056446582_0001/
17/09/22 02:02:43 INFO mapreduce.Job: Running job: job_1506056446582_0001
17/09/22 02:02:55 INFO mapreduce.Job: Job job_1506056446582_0001 running in uber mode : false
17/09/22 02:02:55 INFO mapreduce.Job:  map 0% reduce 0%
17/09/22 02:03:01 INFO mapreduce.Job:  map 100% reduce 0%
17/09/22 02:03:10 INFO mapreduce.Job:  map 100% reduce 100%
17/09/22 02:03:10 INFO mapreduce.Job: Job job_1506056446582_0001 completed successfully
17/09/22 02:03:10 INFO mapreduce.Job: Counters: 49
    File System Counters
      FILE: Number of bytes read=181
      FILE: Number of bytes written=242409
      FILE: Number of read operations=0
      FILE: Number of large read operations=0
      FILE: Number of write operations=0
      HDFS: Number of bytes read=204
      HDFS: Number of bytes written=123
      HDFS: Number of read operations=6
      HDFS: Number of large read operations=0
      HDFS: Number of write operations=2
    Job Counters
      Launched map tasks=1
      Launched reduce tasks=1
      Data-local map tasks=1
      Total time spent by all maps in occupied slots (ms)=4301
      Total time spent by all reduces in occupied slots (ms)=5005
      Total time spent by all map tasks (ms)=4301
      Total time spent by all reduce tasks (ms)=5005
      Total vcore-milliseconds taken by all map tasks=4301
      Total vcore-milliseconds taken by all reduce tasks=5005
      Total megabyte-milliseconds taken by all map tasks=4404224
```

Figura 13

Essa execução deve demorar, tendo em vista o tamanho da base e que o comando está sendo executado em uma única instância de *Hadoop*.

O arquivo de saída poderá ser baixado através da interface *WEB* do *Hadoop*.

Material Complementar

Indicações para saber mais sobre os assuntos abordados nesta Unidade:



Sites

Apache Hadoop

Referência com os diversos comandos existentes para o HDFS.

<https://goo.gl/BBdAzK>

UCI Machine Learning Repository

Download de bases de dados para experimentos *BI/Big Data*.

<https://goo.gl/xvrDyL>

IBM

Sobre o *framework mahout* desenvolvido pela IBM.

<https://goo.gl/MrxUue>

Microsoft

Artigo interessante da MSDN quanto ao uso e o processo de instalação do *Spark*.

<https://goo.gl/Zy9HFN>

DevMedia

Uma breve introdução ao projeto *Mahout*.

<https://goo.gl/CU64WG>

Referências

GIACOMELLI, P. **Apache Mahout Cookbook**. [S.l.]: Packt Publishing, 2013.

OWEN, Sean *et al.* **Mahout in Action**. Greenwich: Manning Publications Co., 2011.

QUINTERO, D. *et al.* **IBM Data Engine for Hadoop and Spark**. IBM RedBooks. 2016. Disponível em: <<http://www.redbooks.ibm.com/redbooks/pdfs/sg248359.pdf>>. Acesso em: 12 set. 2017.

SCOTT, J. A. **Get started with apache spark**. MapR Technologies. 2015. Disponível em: <http://info.mapr.com/rs/mapr/images/Getting_Started_With_Apache_Spark.pdf>. Acesso em: 20 ago. 2017.

SHENOY, Aravind. **Hadoop Explained**. Mumbai: Packt Publishing, 2014. Disponível em: <<https://www.packtpub.com/packt/free-ebook/hadoop-explained>>. Acesso em: 12 set. 2017.

TUTORIALS Point Simply easy learning. Disponível em: <https://www.tutorialspoint.com/apache_spark/index.htm>. Acesso em: 20 ago 2017.

WHITE, T. **Hadoop: The Definitive Guide**. 4. ed. [S.l.]: O'Reilly Media, Inc., 2015.

ZECEVICAND, Petar; BONACI, Marko. **Spark in Action**. Greenwich: Manning Publications Co., 2016.



Cruzeiro do Sul
Educatonal