



Desenvolvimento de Soluções com MapReduce utilizando Hadoop

Capítulo 1. Modelo MapReduce

Prof. João Paulo Barbosa Nascimento



Aula 1.1. História e motivação

Nesta aula

- Modelo MapReduce.
- Motivação para criação.

O modelo MapReduce.

- Desafio: a grande quantidade de dados.
- Dificuldade em escrever programas distribuídos.
- Programas:
 - Não escaláveis.
 - Não tolerantes a falhas.

O modelo MapReduce

- Quem é responsável por toda essa produção de dados?
 - Redes sociais (Facebook, Twitter, etc.).
 - Experimentos científicos (LHC, Projeto Genoma, etc.).
 - Blogs, notícias, fotos, vídeos, etc.
 - Sensores, câmeras de monitoramento, etc.
 - Aumento massivo de espaço nos Hard Disks.
 - Aumento da velocidade de acesso aos dados.
 - Queda no custo do armazenamento.



O modelo MapReduce

- Grande quantidade de dados (Big Data).
 - Necessário uma infraestrutura para:
 - Gerenciar e manipular esses dados.
 - Tolerante a falhas.
 - Adaptável aos diversos problemas rotineiros.
 - Com alta disponibilidade e escalável.
 - Fácil de programar.
 - Paralelizado e distribuído.



O modelo MapReduce

- Em 2003 o Google propôs uma ferramenta:
 - Altamente escalável e tolerante a falhas.
 - Com sistema próprio de arquivos distribuído (Google File System – GFS).
- Composto por duas funções principais (Map e Reduce).
- Trazendo simplicidade no desenvolvimento.
- Modelo MapReduce.

O modelo MapReduce

- “Novo paradigma de programação”.
- O usuário programa duas funções: Map e Reduce.
- O sistema paralelizará a execução dessas funções.
- Com tolerância a falhas e balanceamento de carga.
- Com sistema próprio de armazenamento distribuído.
- Necessário modelar o problema nesse formato.

Conclusão

- Criação do modelo MapReduce.
- Solução para a distribuição, escalabilidade e tolerância a falhas.

■ Próxima aula

- ❑ Funcionamento do modelo MapReduce.



Aula 1.2. Exemplo de funcionamento

- ❑ Funcionamento do modelo MapReduce.
- ❑ Exemplo de execução.
- ❑ Funções Map e Reduce.

Exemplo de funcionamento do MapReduce

- Contexto do exemplo:
 - Rede de sensores.
 - Monitora o nível de CO₂ (dióxido de carbono) na atmosfera.
 - Sensores em diversos pontos do globo terrestre.
 - Coletando dados o tempo todo.
 - Gerando grande massa de dados.
 - Com o crescimento dos dados a pesquisa sequencial fica inviável.



Exemplo de funcionamento do MapReduce

- Modelo MapReduce torna-se um aliado para auxiliar nesse processamento.
- Sensores espalhados fazem as medições e armazenam os resultados em arquivos texto.
- Esse problema é candidato a ser resolvido pelo modelo MR.
- Dados estruturados dentro do arquivo em posições específicas.
- Cada medição consiste em uma linha do arquivo.

Exemplo de funcionamento do MapReduce

- Dados do site CO2 Now (<http://www.co2now.org>).
- Focaremos na data da medição e no nível do CO2.

```
1988 04121402 35707
1988 04251405 35727
1988 05031922 35823
1988 05091357 35719
1987 02231400 35412
1987 03021349 35267
1987 03091400 35350
1987 03161359 35418
1986 01201523 35129
1986 01271518 35520
1986 02031455 35328
1986 02101622 35257
1985 05061943 35234
1985 06241908 35056
1985 07011900 34967
1985 08261528 33733
```

Exemplo de funcionamento do MapReduce

- Queremos saber o maior nível de CO2 por ano.
- Função Map recebe as linhas e extrai ano e nível de CO2.

(1988, 357.07)

(1988, 357.27)

(1988, 358.23)

(1988, 357.19)

(1987, 354.12)

(1987, 352.67)

(1987, 353.50)

(1987, 354.18)

(1986, 351.29)

(1986, 355.20)

(1986, 353.28)

(1986, 352.57)

(1985, 352.34)

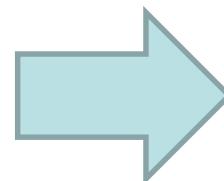
(1985, 350.56)

(1985, 349.67)

(1985, 337.33)

Exemplo de funcionamento do MapReduce

1988 04121402 35707
1988 04251405 35727
1988 05031922 35823
1988 05091357 35719
1987 02231400 35412
1987 03021349 35267
1987 03091400 35350
1987 03161359 35418
1986 01201523 35129
1986 01271518 35520
1986 02031455 35328
1986 02101622 35257
1985 05061943 35234
1985 06241908 35056
1985 07011900 34967
1985 08261528 33733



(1988, 357.07)
(1988, 357.27)
(1988, 358.23)
(1988, 357.19)
(1987, 354.12)
(1987, 352.67)
(1987, 353.50)
(1987, 354.18)
(1986, 351.29)
(1986, 355.20)
(1986, 353.28)
(1986, 352.57)
(1985, 352.34)
(1985, 350.56)
(1985, 349.67)
(1985, 337.33)

Resultado da função Map.

Exemplo de funcionamento do MapReduce

- Map ordena os dados pelo par chave-valor.
- A função Reduce receberá o seguinte conjunto de dados:

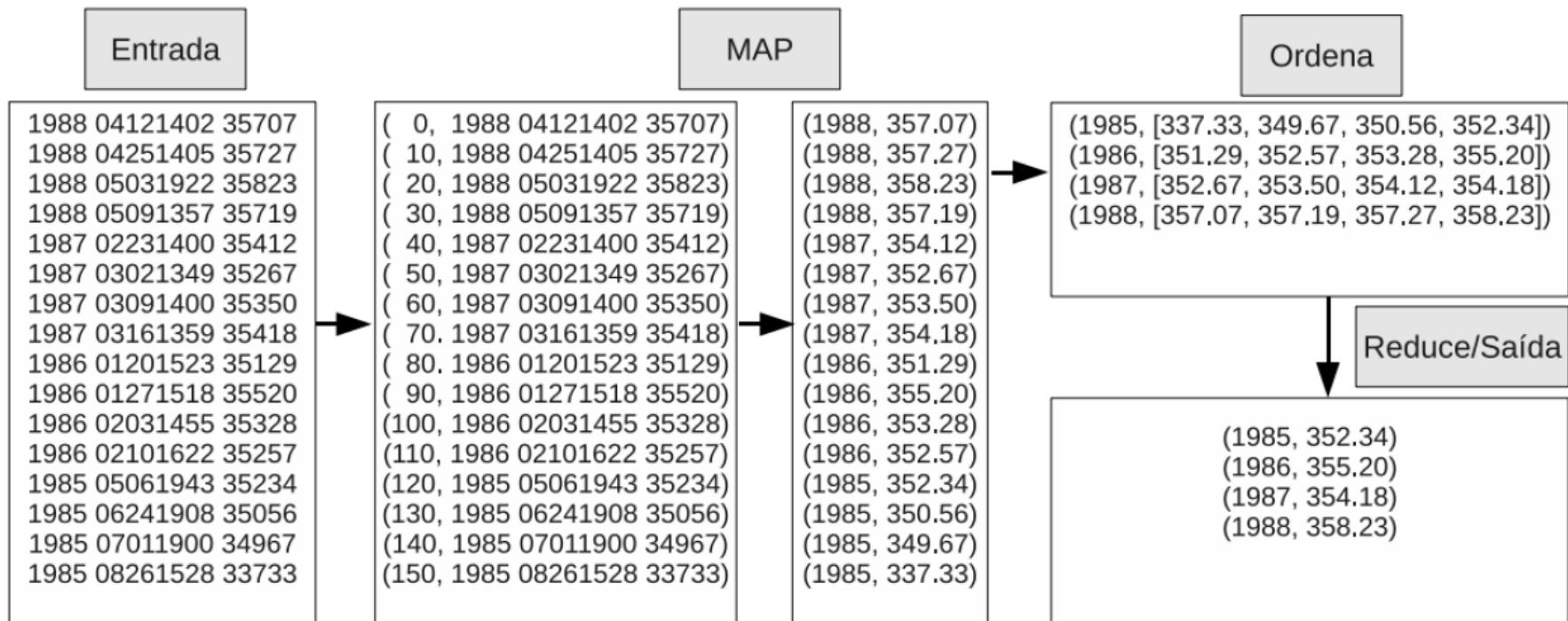
(1985, [337.33, 349.67, 350.56, 352.34])
(1986, [351.29, 352.57, 353.28, 355.20])
(1987, [352.67, 353.50, 354.12, 354.18])
(1988, [357.07, 357.19, 357.27, 358.23])

Exemplo de funcionamento do Hadoop

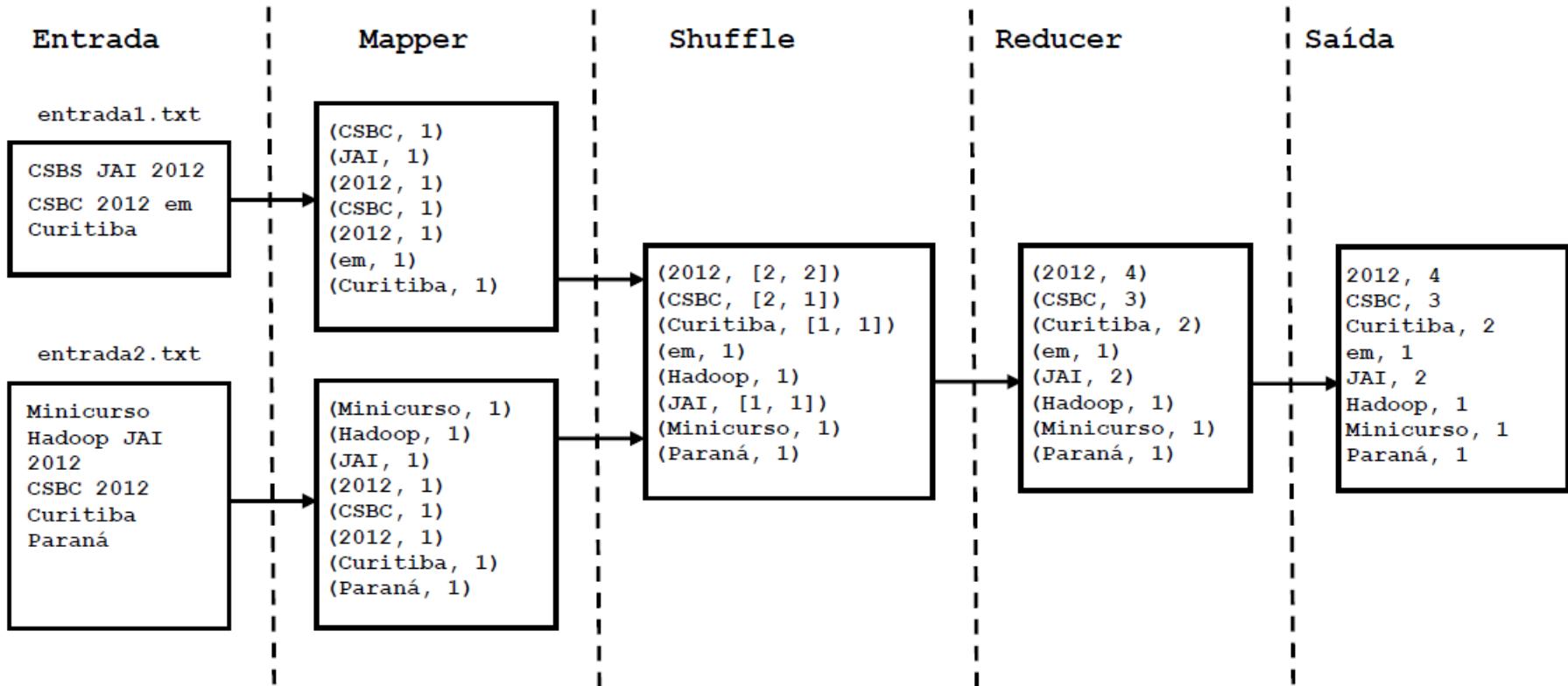
- Cada ano pode ser processado por uma função Reduce.
- Ao final, a seguinte saída será produzida pela função Reduce:

(1985, 352.34)
(1986, 355.20)
(1987, 354.18)
(1988, 358.23)

Exemplo de funcionamento do MapReduce



Exemplo de funcionamento do Hadoop



Conclusão

- Funcionamento do modelo.
- Entrada / Saída.
- Funções Map e Reduce.



Desenvolvimento de Soluções com MapReduce utilizando Hadoop

Capítulo 2. Conhecendo o Hadoop

Prof. João Paulo Barbosa Nascimento



Aula 2.1. História

Nesta aula

- ❑ Criação do Hadoop.

- Foi criado por Doug Cutting, criador do Apache Lucene.
- Lucene: uma biblioteca de busca textual amplamente utilizada.
- O Hadoop teve suas origens no Apache Nutch.
- Um motor de busca na web open-source.
- O Apache Nutch é uma parte do Apache Lucene.



- O Apache Nutch começou em 2002, juntamente com um crawler e um sistema de busca.
- Percebeu-se que a arquitetura implementada não escalaria para bilhões de páginas.
- A ajuda para isso estava em um artigo publicado em 2003, pelo Google.



- Esse artigo descrevia a arquitetura do Google's Distributed FileSystem (GFS).
- GFS resolveria a necessidade de armazenamento de arquivos muito grandes, gerado pela coleta e indexação da web.
- Em 2004, começaram a escrever uma implementação em código aberto do GFS.



- Surgiu aí o Nutch Distributed FileSystem (NDFS).
- Em 2004 o Google publicou o documento que apresentou o modelo MapReduce ao mundo.
- Em 2005, os desenvolvedores implementaram o modelo MapReduce no Nutch.



- No meio de 2005, todos os principais algoritmos do Nutch foram portados para executar usando o MapReduce e o NDFS.
- NDFS e o MapReduce no Nutch foram além do ambiente acadêmico.
- Em 2006, eles saíram do Nutch para formarem um projeto independente.
- Surgiu o projeto Hadoop.



- Na mesma época, Doug Cutting foi para o Yahoo.
- O Yahoo forneceu uma equipe para transformar o Hadoop em um sistema, para funcionar na escala da web.
- Em 2008 o Yahoo anunciou que sua pesquisa contava com um cluster Hadoop com 10.000 núcleos.



- Ainda em 2008, o Hadoop ganhou um projeto de alto nível na Apache.
- Muitas outras empresas passaram a utilizar.
- Yahoo!, Facebook e New York Times.
- Em abril de 2008, o Hadoop quebrou o recorde mundial, se tornando o sistema mais rápido para classificar 1TB de dados.
- 910 nós em 209 segundos.



- Em novembro de 2008, o Google informou que a sua implementação do MapReduce ordenou 1TB em 68 segundos.
- Em abril de 2009, foi anunciado pelo Yahoo a ordenação de 1TB em 62 segundos.
- Em uma competição em 2014, uma equipe da Databricks usou um cluster Spark com 207 nós para classificar 1TB em 1406 minutos.
- 4,27 TB por minuto.



- Hoje, o Hadoop é amplamente utilizado por diversas organizações.
- Ferramenta para análise e armazenamento de Big Data.
- Grandes empresas e produtos dão suporte ao Hadoop.
- IBM, Oracle, Microsoft, etc.
- Cloudera, Hortonworks, etc.



Conclusão

- Surgimento e evolução do Hadoop.
- Armazenamento e processamento distribuído.
- Atualmente é amplamente utilizado.

- Ecossistema Hadoop.



Aula 2.2. Ecossistema

- Ecossistema.
- Componentes.

- Arcabouço (framework) que permite processamento:
 - Paralelo.
 - Distribuído.
- Grandes massas de dados.
- Utiliza clusters de computadores.



- Proporciona escalabilidade.
- Modelo de programação simplificado:
 - Paralelismo e distribuição automática (por conta do framework).
 - Desenvolvedor pode focar seus esforços no negócio.



- Hadoop Common:
 - Aplicativos comuns à todos os outros módulos do ambiente.
- Hadoop Distributed File System (HDFS):
 - Sistema de arquivos distribuído.

Ecosistema

IGTI



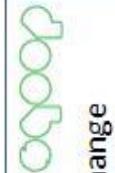
Apache Hadoop Ecosystem



Provisioning, Managing and Monitoring Hadoop Clusters

Ambari

- Ambari: gerenciamento e monitoramento do cluster.
- Flume: coletor de logs.
- Sqoop: transferência de dados.
- PIG: Script.
- Hive: Data Warehouse.
- Hbase: Banco de dados não-relacional, distribuído e orientado a colunas.
- Yarn: gerenciador de execução.
- Oozie: agendamento de tarefas Hadoop.



Sqoop
Data Exchange



Zookeeper
Coordination



Oozie
Workflow



Pig
Scripting



Mahout
Machine Learning

R Connectors

Statistics



Hive
SQL Query



Hbase
Columnar Store



HDFS
Hadoop Distributed File System



YARN Map Reduce v2
Distributed Processing Framework



- Hadoop Yarn:
 - Framework para distribuição de tarefas.
 - Gerenciamento dos recursos do cluster.
 - Inserido a partir da versão 2.
- Hadoop MapReduce:
 - Framework para escrita de aplicações para processamento de grandes massas de dados.
 - Aplicações paralelas, distribuídas e tolerantes a falhas.

Ecossistema

- Ambari:
 - Ferramenta para gerenciamento e monitoramento do cluster.
 - Consumo de CPU, memória e rede.
 - Máquina por máquina.
- Hbase:
 - Banco de dados não-relacional e distribuído.
 - Suporta o armazenamento estruturado em grandes tabelas (BigTable).



Apache Ambari



- Hive:

- Estrutura para Data Warehouse.
 - Suporta consultas SQL, indexação e grandes conjuntos de dados.



- Mahout:

- Biblioteca que fornece algoritmos de aprendizado de máquina.
 - Mineração de dados (classificação, agrupamento, etc.).





- Spark:
 - Estrutura do Hadoop que prioriza o processamento em memória.
 - Possui algoritmos de aprendizado de máquina, grafos e streaming.
- Zookeeper:
 - Coordenador de serviços altamente distribuído.
 - Coordena locks, sincronização e grupos de serviços para aplicações distribuídas.



Apache ZooKeeper™

Conclusão

- Componentes.
- Ecossistema Hadoop.

YARN.

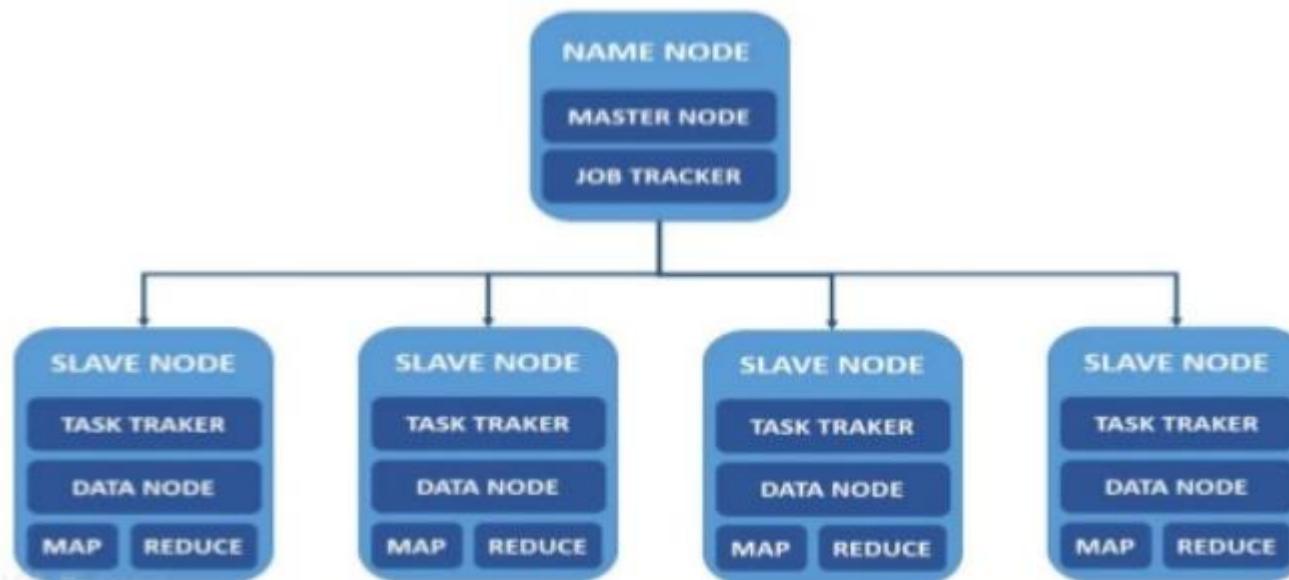


Aula 2.3. YARN

Nesta aula

- ❑ YARN.
- ❑ Arquitetura, importância e funcionamento.

- Arquitetura mestre/escravo.



- YARN (**Y**et **A**nother **R**esource **N**avigator).
- No Hadoop versão 1.0 (versão 1 do MapReduce), o MapReduce desempenhava as funções de processamento e gerenciamento de recursos.
- O mestre alocava os recursos, agendava e monitorava o processamento.
- Esse desenho resultou em um gargalo de escalabilidade.
- O Yahoo! Encontrou o limite em 5000 nós e 40.000 tarefas simultâneas.

- A utilização de recursos computacionais era ineficiente.
- Para superar esses problemas, o YARN foi introduzido na versão 2.0 do Hadoop (2012).
- A ideia por trás do YARN é aliviar o MapReduce.
- O YARN assume a responsabilidade do gerenciamento de recursos e agendamento de tarefas.

- O YARN começou a fornecer ao Hadoop a capacidade de executar tarefas não MapReduce dentro da estrutura.
- Revolucionou o ecossistema Hadoop.
- O Hadoop tornou-se muito mais flexível, eficiente e escalável.
- Em 2013, o Yahoo implantou o YARN.
- Reduziu o tamanho de seu cluster de 40.000 para 32.000 nós.
- O número de trabalhos dobrou para 26 milhões por mês.



Hadoop v1.0

MapReduce

Data Processing
& Resource Management

HDFS

Distributed File Storage



Hadoop v2.0

MapReduce

Other Data
Processing
Frameworks

YARN

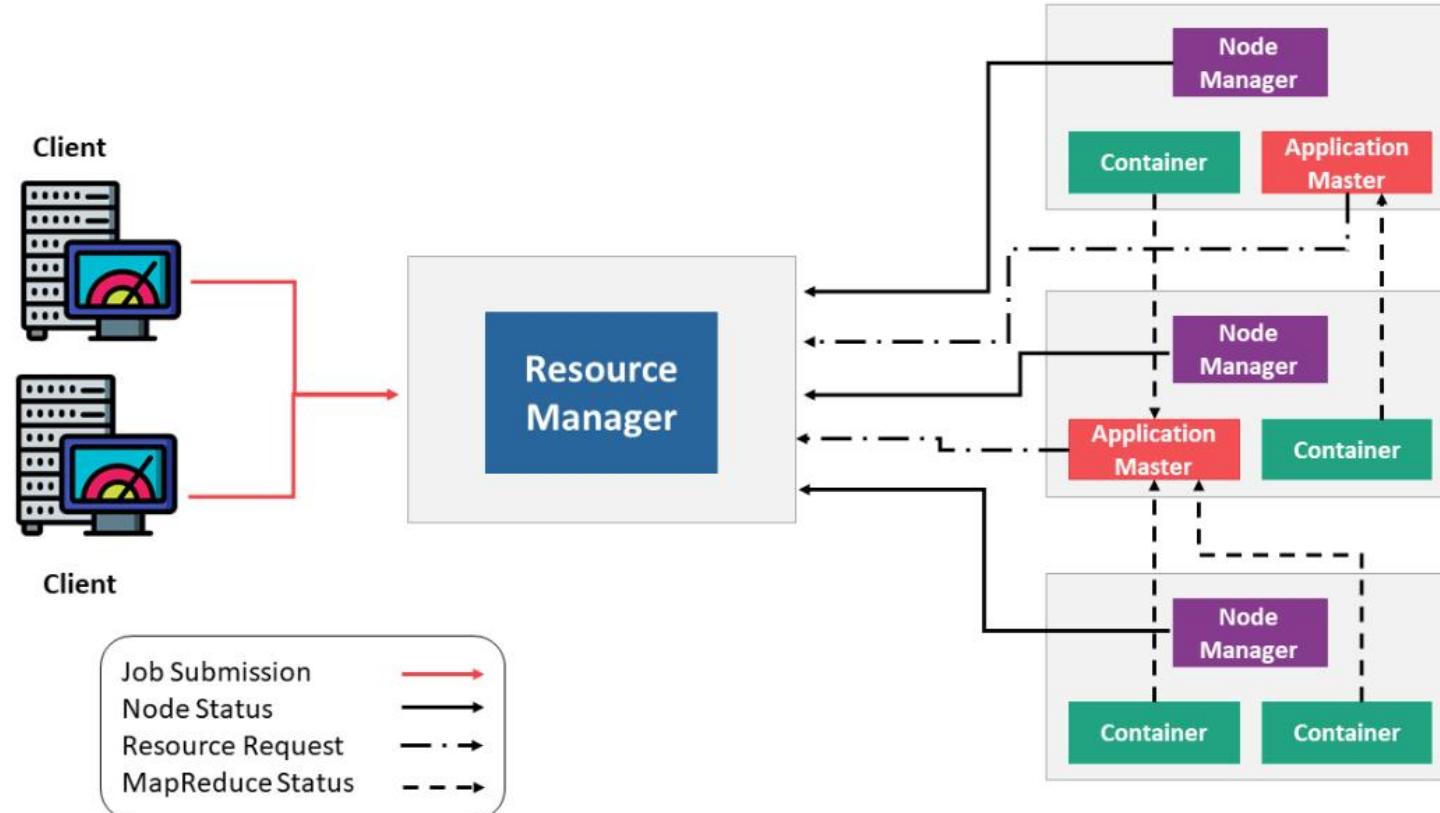
Resource Management

HDFS

Distributed File Storage

- O YARN permitiu a execução de operações usando uma maior variedade de ferramentas:
 - Spark (processamento em tempo real).
 - Hive (para consultas SQL).
 - Hbase (para noSQL).

- A arquitetura YARN consiste nos seguintes componentes principais:
 - **Resource Manager:** executa no mestre e gerencia a alocação de recursos no cluster.
 - **Node Manager:** executa nos escravos e são responsáveis pela execução das tarefas em cada um dos nós.
 - **Application Master:** gerencia o ciclo de vida do job e os recursos necessários a cada aplicação. Monitora as execuções das tarefas.
 - **Container:** pacote de recursos, incluindo memória RAM, CPU, rede, etc.
(em um nó)



Conclusão

- YARN – Gerenciador de recursos.
- MapReduce x YARN.

■ Próxima aula

- ❑ Funcionamento dos componentes do YARN.

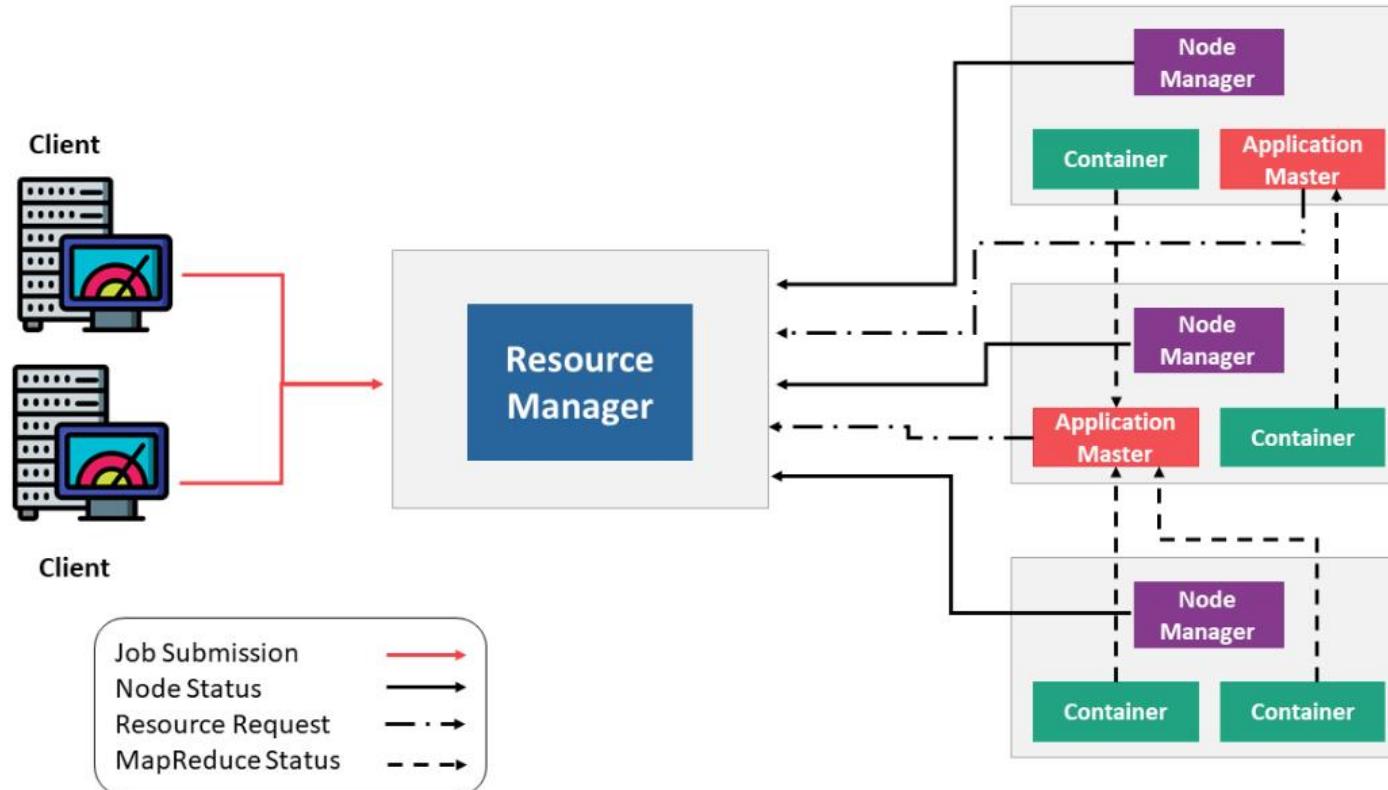


Aula 2.4. Componentes do YARN

Nesta aula

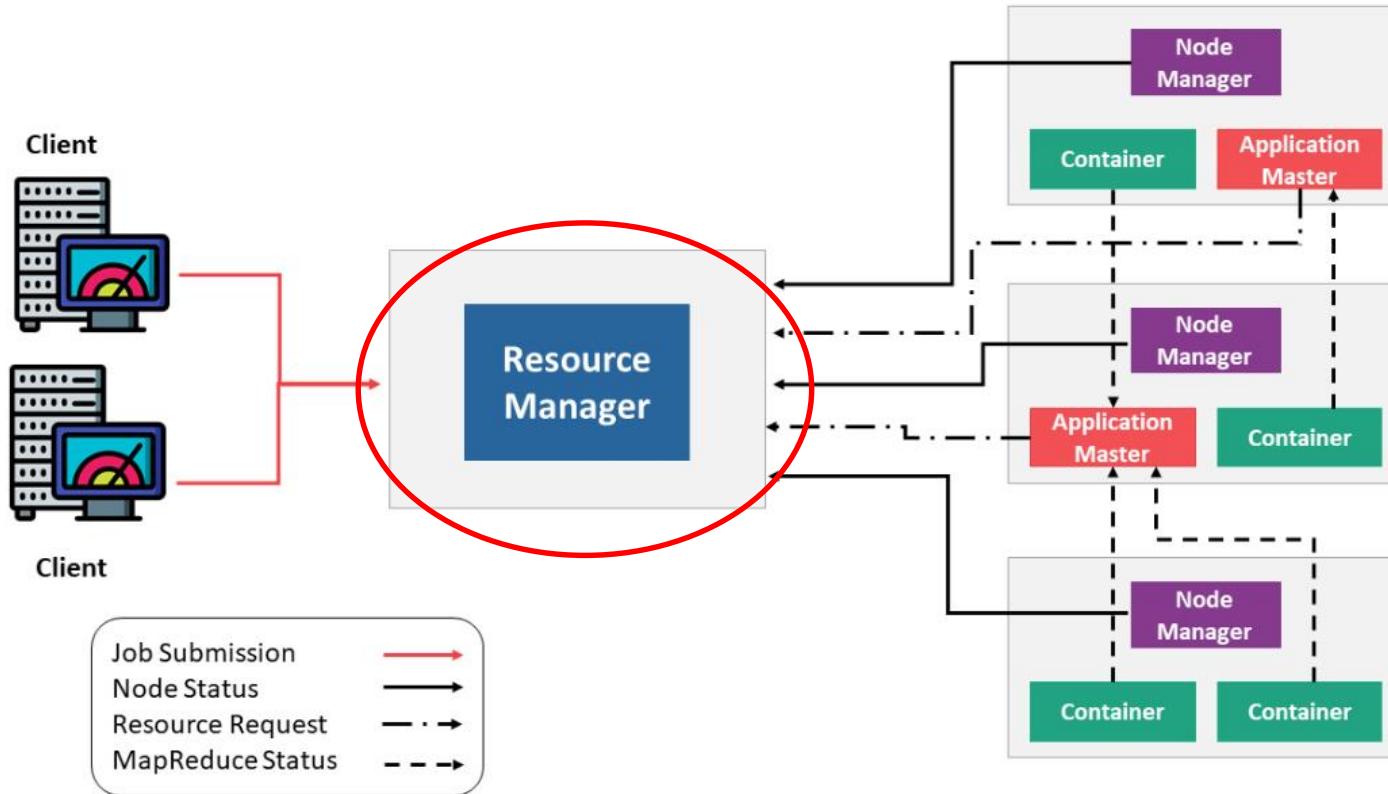
- ❑ Arquitetura YARN.
- ❑ Detalhe dos componentes do YARN.

Componentes do YARN



- Resource Manager:
 - Autoridade final na alocação de recursos.
 - Árbitro dos recursos do cluster. Decide pela alocação dos recursos.
 - Otimiza a utilização do cluster.
 - Possui dois componentes: o Agendador e o Gerenciador de Aplicativos.

Componentes do YARN



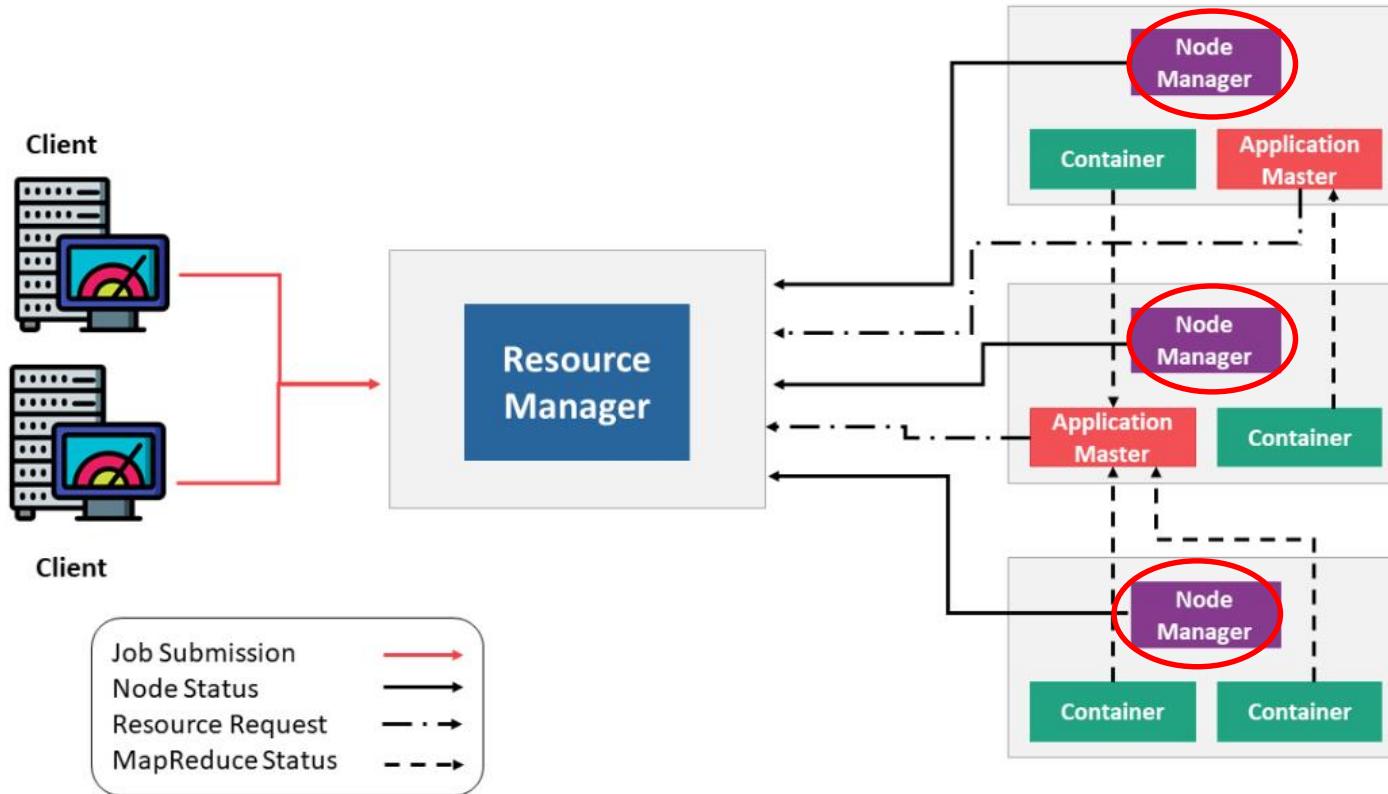
Componentes do YARN

- Resource Manager – Agendador:
 - Responsável por alocar recursos para os vários aplicativos em execução, sujeito a restrições de capacidades, filas, etc.
 - Não executa nenhum monitoramento ou rastreamento de status dos aplicativos (Agendador Puro).
 - Não garante a reinicialização das tarefas com falhas.

Componentes do YARN

- Resource Manager - Gerenciador de Aplicativos:
 - Responsável por aceitar os trabalhos.
 - Negocia a execução.

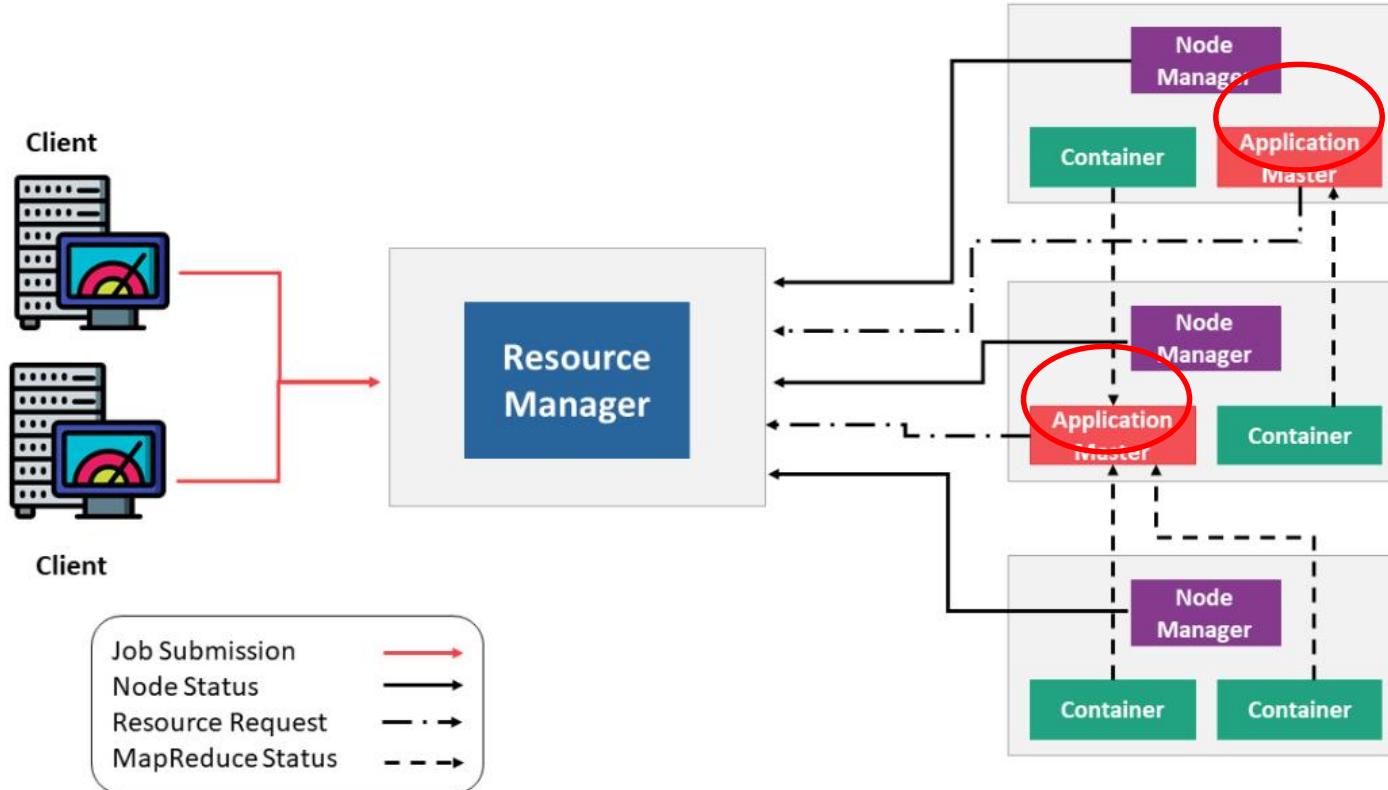
Componentes do YARN



Componentes do YARN

- Node Manager:
 - Cuida dos nós individualmente no cluster.
 - Envia “pulsos” sobre a integridade do nó.
 - Seu objetivo principal é gerenciar contêineres de aplicativos.
 - Executa o gerenciamento de logs.
 - Monitora o uso de recursos (CPU e memória) de contêineres individuais.

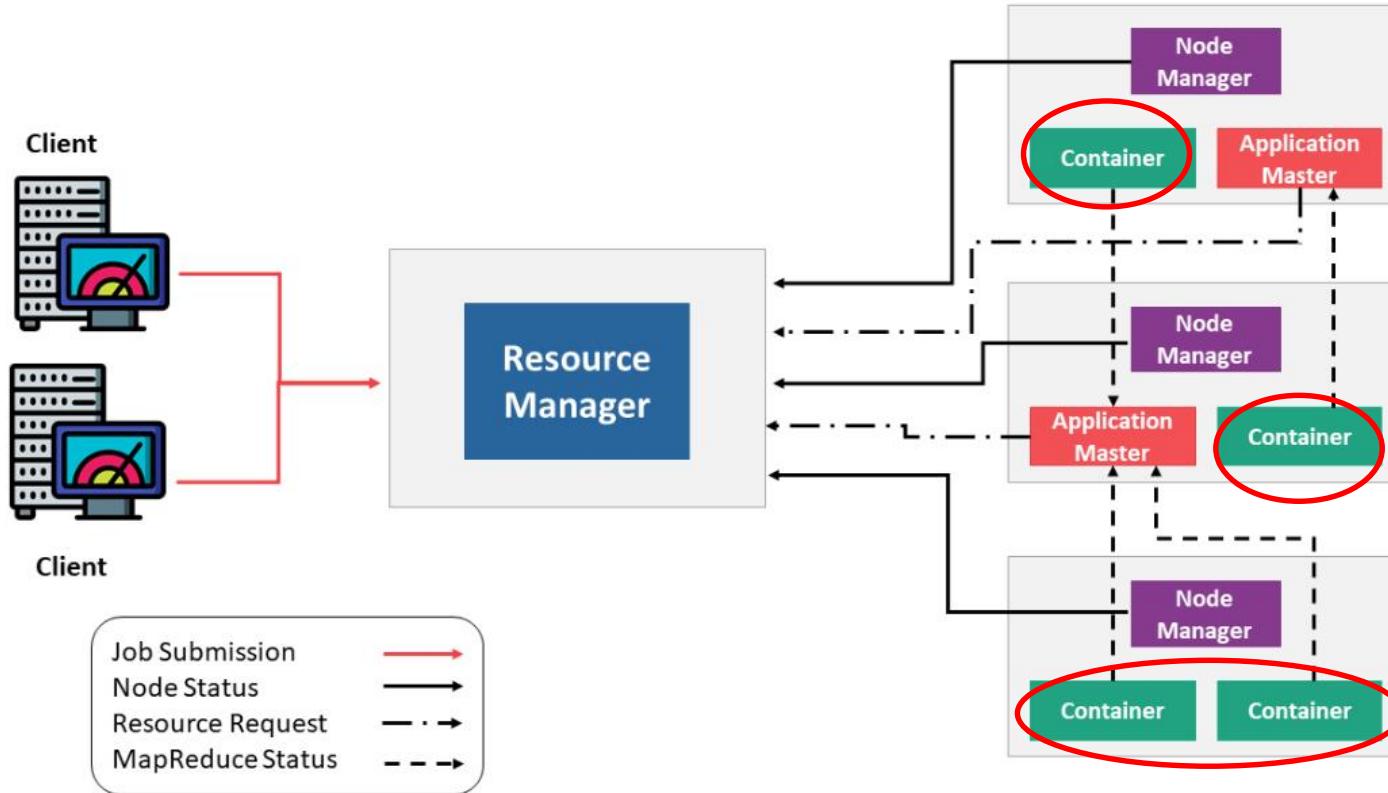
Componentes do YARN



Componentes do YARN

- Application Master:
 - Cada trabalho enviado ao framework possui um único AM associado a ele.
 - É o processo que coordena a execução de um aplicativo no cluster e também gerencia as falhas.
 - Sua tarefa é negociar recursos com o Resource Manager.
 - Responsável por negociar os contêineres de recursos apropriados.

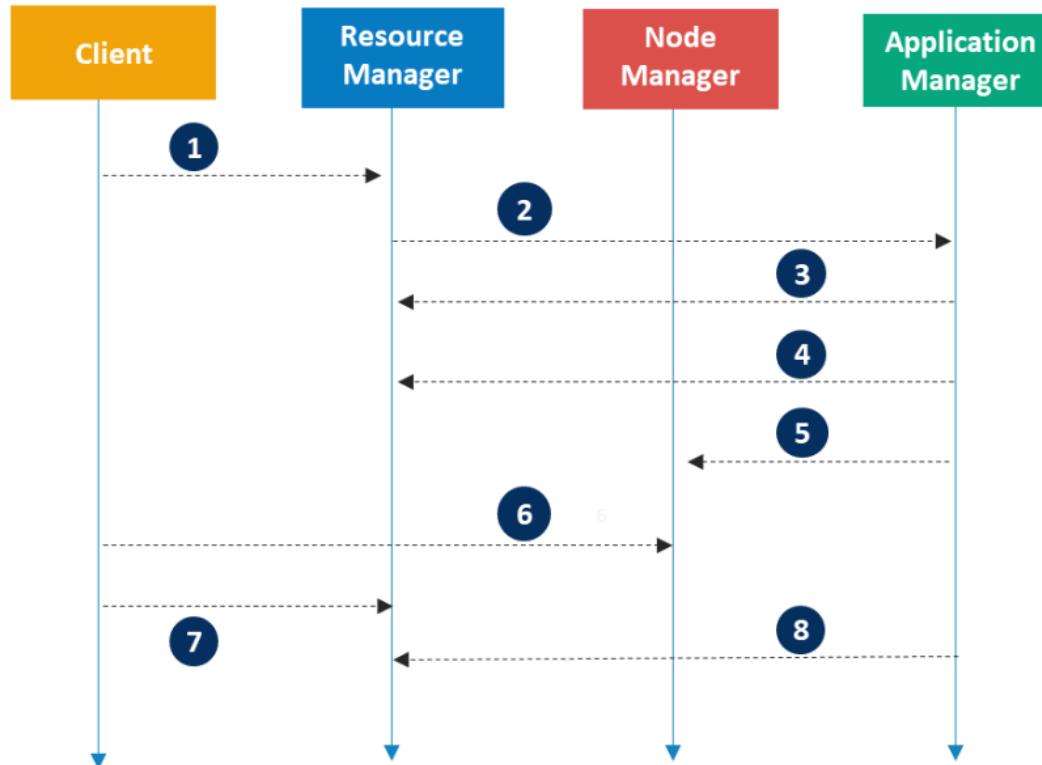
Componentes do YARN



Componentes do YARN

- Contêiner:
 - É uma coleção de recursos físicos (RAM, núcleos de CPU e discos em um único nó).
 - Gerenciado pelo ciclo de vida do contêiner (CLC).

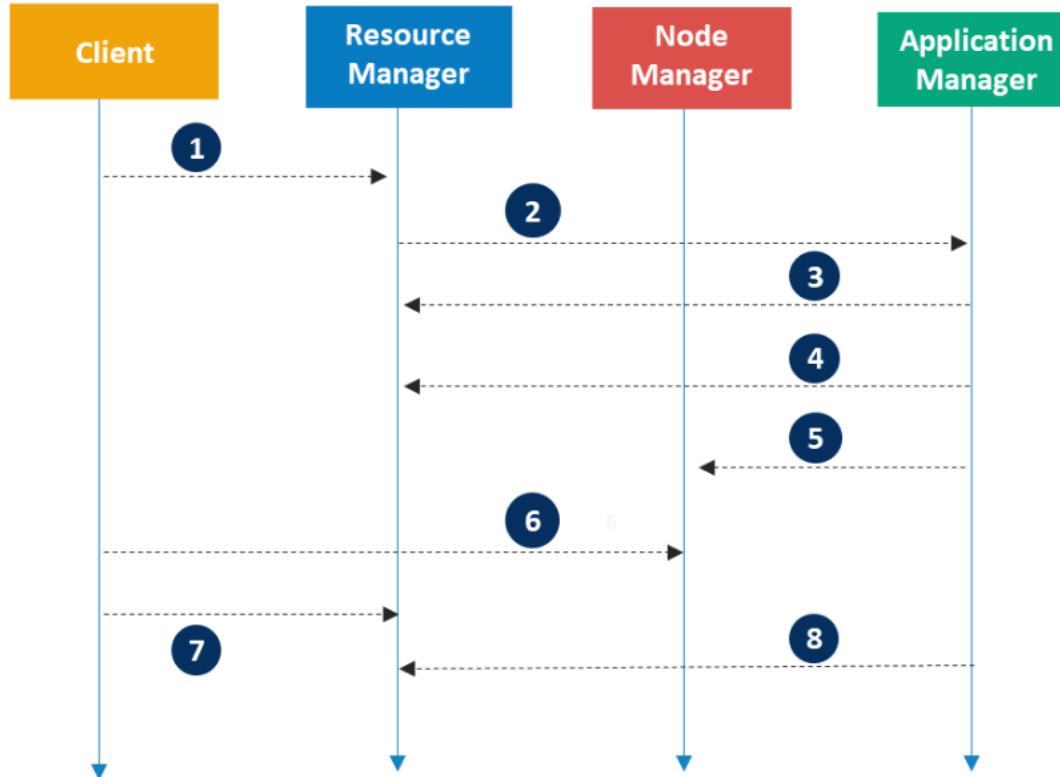
Componentes do YARN



▪ Fluxo de execução:

1. O Client submete uma aplicação.
2. O Resource Manager aloca um contêiner para iniciar o Application Manager.
3. O Application Manager registra-se no Resource Manager.
4. O Application Manager solicita contêiners ao Resource Manager.

Componentes do YARN



- **Fluxo de execução:**

5. O Application Manager notifica o Node Manager para que os contêineres sejam disponibilizados.
6. O código da aplicação é executado nos contêineres.
7. O Client entra em contato com o Resource Manager para que ele monitore o status da aplicação.
8. O Application Manager encerra seu registro no Resource Manager.

- Detalhes sobre os componentes do YARN.
- Fluxo.

■ Próxima aula

- Instalação do Hadoop.



Aula 2.5. Instalação

- ❑ Instalação do Hadoop.

Instalação do Hadoop

- Primeiro passo:
 - Pré-requisitos.



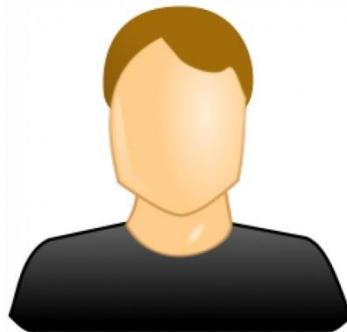
Hadoop – Instalação

- Pré-requisitos para a instalação:
 - Máquina Virtual JAVA.
 - Use o comando java: –version.
 - SSH.



Instalação do Hadoop

- Segundo passo:
 - Usuário hduser.



Hadoop – Instalação

- Criar um usuário exclusivo para trabalhar com o Hadoop.
- Adicionando o usuário hduser ao grupo hadoop.
- Essa etapa é opcional.

```
1 $ sudo addgroup hadoop  
2 $ sudo adduser --ingroup hadoop hduser
```

- Terceiro passo:
 - Configurando o SSH.



- Hadoop utiliza SSH para gerenciar os nós.
- Mesmo para localhost.
- Devemos configurar o acesso ao localhost para o usuário hduser sem senha.
- Iremos considerar que o ssh está executando na máquina.

Hadoop – Instalação

- Conecte com o usuário hduser. (1)
- Gere uma chave SSH para o usuário hduser. (2)
- Senha vazia.

```
1 user@ubuntu:~$ su - hduser (1)
2 hduser@ubuntu:~$ ssh-keygen -t rsa -P ""
3 Generating public/private rsa key pair.
4 Enter file in which to save the key (/home/hduser/.ssh/id_rsa):
5 Created directory '/home/hduser/.ssh'.
6 Your identification has been saved in /home/hduser/.ssh/id_rsa.
7 Your public key has been saved in /home/hduser/.ssh/id_rsa.pub.
8 The key fingerprint is:
9 9b:82:ea:58:b4:e0:35:d7:ff:19:66:a6:ef:ae:0e:d2 hduser@ubuntu
10 The key's randomart image is:
11 [...snipp...]
12 hduser@ubuntu:~$
```

- Habilitar o SSH para acessar a máquina local com a chave recentemente criada.

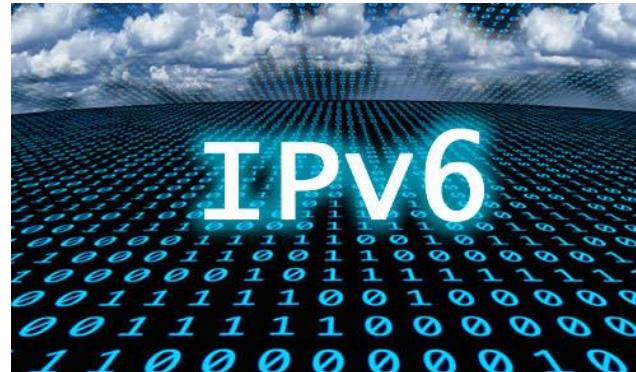
```
1 hduser@ubuntu:~$ cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
```

- Vamos testar a conexão para a máquina local com o usuário hduser!

```
1 hduser@ubuntu:~$ ssh localhost
2 The authenticity of host 'localhost (::1)' can't be established.
3 RSA key fingerprint is d7:87:25:47:ae:02:00:eb:1d:75:4f:bb:44:f9:36:26.
4 Are you sure you want to continue connecting (yes/no)? yes
5 Warning: Permanently added 'localhost' (RSA) to the list of known hosts.
6 Linux ubuntu 2.6.32-22-generic #33-Ubuntu SMP Wed Apr 28 13:27:30 UTC 2010 i686 GNU/Linux
7 Ubuntu 10.04 LTS
8 [...snipp...]
9 hduser@ubuntu:~$
```

Instalação do Hadoop

- Quarto passo:
 - Desabilitar o IPv6.



- Abra o arquivo /etc/sysctl.conf em um editor de texto e adicione as seguintes linhas no fim do arquivo:

/etc/sysctl.conf

```
1 # disable ipv6
2 net.ipv6.conf.all.disable_ipv6 = 1
3 net.ipv6.conf.default.disable_ipv6 = 1
4 net.ipv6.conf.lo.disable_ipv6 = 1
```

- Reinicie a máquina.

- Verifique se o IPv6 está realmente desabilitado:

```
1 $ cat /proc/sys/net/ipv6/conf/all/disable_ipv6
```

- Se o comando retornar 0 (zero), o IPv6 está habilitado.
- **Se retornar 1, o IPv6 está desabilitado (é o que queremos).**

- Quinto passo:
 - Hadoop.



Hadoop – Instalação

- Faça o download do Hadoop:

- <http://www-eu.apache.org/dist/hadoop/core/hadoop-2.7.3/>

Name	Last modified	Size	Description
Parent Directory		-	
hadoop-2.7.3-src.tar.gz	2016-08-25 19:25	17M	
hadoop-2.7.3-src.tar.gz.asc	2016-08-25 19:25	521	
hadoop-2.7.3-src.tar.gz.mds	2016-08-25 19:25	1.1K	
hadoop-2.7.3.tar.gz	2016-08-25 19:25	204M	
hadoop-2.7.3.tar.gz.asc	2016-08-25 19:25	521	
hadoop-2.7.3.tar.gz.mds	2016-08-25 19:25	958	

Hadoop – Instalação

- Sugestão: descompactar no diretório /usr/local.

```
1 $ cd /usr/local  
2 $ sudo tar xzf hadoop- 2.7.3 .tar.gz  
3 $ sudo mv hadoop- 2.7.3 hadoop  
4 $ sudo chown -R hduser:hadoop hadoop
```

Instalação do Hadoop

- Sexto passo:
 - Variáveis de ambiente.



Hadoop – Instalação

- Precisamos editar o arquivo .bashrc para o usuário hduser.
- Incluir variáveis de ambiente.
- HADOOP_HOME.
- JAVA_HOME.

```
$HOME/.bashrc
1 # Set Hadoop-related environment variables
2 export HADOOP_HOME=/usr/local/hadoop ← yellow arrow
3
4 # Set JAVA_HOME (we will also configure JAVA_HOME directly for Hadoop later on)
5 export JAVA_HOME=/usr/lib/jvm/java-6-sun ← blue arrow
6
7 # Some convenient aliases and functions for running Hadoop-related commands
8 unalias fs &> /dev/null
9 alias fs="hadoop fs"
10 unalias hls &> /dev/null
11 alias hls="fs -ls"
12
13 # If you have LZO compression enabled in your Hadoop cluster and
14 # compress job outputs with LZOP (not covered in this tutorial):
15 # Conveniently inspect an LZOP compressed file from the command
16 # line; run via:
17 #
18 # $ lzohead /hdfs/path/to/lzop/compressed/file.lzo
19 #
20 # Requires installed 'lzop' command.
21 #
22 lzohead () {
23     hadoop fs -cat $1 | lzop -dc | head -1000 | less
24 }
25
26 # Add Hadoop bin/ directory to PATH
27 export PATH=$PATH:$HADOOP_HOME/bin ← red arrow
```

Instalação do Hadoop

- Sétimo passo:
 - Configurando o Hadoop.



Hadoop – Instalação

- Crie um diretório para armazenamento de dados temporários do Hadoop.
- Sugestão: criar no endereço: /usr/local/tmp
 - sudo mkdir –p /usr/local/tmp
- Esse diretório vai ser utilizado posteriormente.
- Atribua permissões ao novo diretório:

```
sudo chown hduser:hadoop /usr/local/hadoop/tmp
```

```
sudo chmod /usr/local/hadoop/tmp
```

Hadoop – Instalação

- Altere o arquivo core-site.xml.
- /usr/local/hadoop/etc/hadoop/core-site.xml
- Inserir entre a tag: <configuration> </configuration>:

conf/core-site.xml

```
1 <property>
2   <value>/usr/local/hadoop/tmp</name>
3   <value>/app/hadoop/tmp</value>
4   <description>A base for other temporary directories.</description>
5 </property>
6
7 <property>
8   <name>fs.default.name</name>
9   <value>hdfs://localhost:54310</value>
10  <description>The name of the default file system. A URI whose
11    scheme and authority determine the Filesystem implementation. The
12    uri's scheme determines the config property (fs.SCHEME.impl) naming
13    the FileSystem implementation class. The uri's authority is used to
14    determine the host, port, etc. for a filesystem.</description>
15 </property>
```

- Altere o arquivo mapred-site.xml.
- /usr/local/hadoop/etc/hadoop/mapred-site.xml.
- Inserir entre a tag: <configuration> <\configuration>:

conf/mapred-site.xml

```
1 <property>
2   <name>mapred.job.tracker</name>
3   <value>localhost:54311</value>
4   <description>The host and port that the MapReduce job tracker runs
5     at. If "local", then jobs are run in-process as a single map
6     and reduce task.
7   </description>
8 </property>
```

- Altere o arquivo hdfs-site.xml.
- /usr/local/hadoop/etc/hadoop/hdfs-site.xml.
- Inserir entre as tags: <configuration> </configuration>:

```
conf/hdfs-site.xml

1 <property>
2   <name>dfs.replication</name>
3   <value>1</value>
4   <description>Default block replication.
5   The actual number of replications can be specified when the file is created.
6   The default is used if replication is not specified in create time.
7   </description>
8 </property>
```

Instalação do Hadoop

- Oitavo passo:
 - Formatando o HDFS.



Hadoop – Instalação

- Formatando o sistema de arquivos distribuído – HDFS:

```
1 hduser@ubuntu:~$ /usr/local/hadoop/bin/hadoop namenode -format
```

- A saída do comando acima deve ser a seguinte:

```
1 hduser@ubuntu:/usr/local/hadoop$ bin/hadoop namenode -format
2 10/05/08 16:59:56 INFO namenode.Namenode: STARTUP_MSG:
3 ****
4 STARTUP_MSG: Starting NameNode
5 STARTUP_MSG:   host = ubuntu/127.0.1.1
6 STARTUP_MSG:   args = [-format]
7 STARTUP_MSG:   version = 0.20.2
8 STARTUP_MSG:   build = https://svn.apache.org/repos/asf/hadoop/common/branches/branch-0.20 -i
9 ****
10 10/05/08 16:59:56 INFO namenode.FSNamesystem: fsOwner=hduser,hadoop
11 10/05/08 16:59:56 INFO namenode.FSNamesystem: supergroup=supergroup
12 10/05/08 16:59:56 INFO namenode.FSNamesystem: isPermissionEnabled=true
13 10/05/08 16:59:56 INFO common.Storage: Image file of size 96 saved in 0 seconds.
14 10/05/08 16:59:57 INFO common.Storage: Storage directory .../hadoop-hduser/dfs/name has been
15 10/05/08 16:59:57 INFO namenode.Namenode: SHUTDOWN_MSG:
16 ****
17 SHUTDOWN_MSG: Shutting down NameNode at ubuntu/127.0.1.1
18 ****
19 hduser@ubuntu:/usr/local/hadoop$
```

Instalação do Hadoop

- Nono passo:
 - Iniciando os serviços Hadoop.



- Execute o comando:
 - hduser@ubuntu:~\$ /usr/local/hadoop/sbin/start-all.sh
- A saída do comando acima deve ser a seguinte:

```
1  hduser@ubuntu:~$ /usr/local/hadoop/sbin/start-all.sh
2  starting namenode, logging to /usr/local/hadoop/bin/../logs/hadoop-hduser-namenode-ubuntu.out
3  localhost: starting datanode, logging to /usr/local/hadoop/bin/../logs/hadoop-hduser-datanode-
4  localhost: starting secondarynamenode, logging to /usr/local/hadoop/bin/../logs/hadoop-hduser-
5  starting jobtracker, logging to /usr/local/hadoop/bin/../logs/hadoop-hduser-jobtracker-ubuntu.
6  localhost: starting tasktracker, logging to /usr/local/hadoop/bin/../logs/hadoop-hduser-tasktr
7  hduser@ubuntu:/usr/local/hadoop$
```

- Por meio do comando jps, verifique se os serviços estão ativos:
 - Datanode.
 - ResourceManager.
 - NameNode.
 - SecondaryNameNode.
 - NodeManager.

Instalação do Hadoop

- Décimo passo:
 - Executando um teste.



- Executando o cálculo do PI.
- Dentro da pasta /usr/local/hadoop.

```
bin/hadoop jar /usr/local/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-  
2.7.3.jar pi 16 1000
```

Hadoop – Instalação

```
HDFS: Number of write operations=309
Map-Reduce Framework
  Map input records=16
  Map output records=32
  Map output bytes=288
  Map output materialized bytes=448
  Input split bytes=2390
  Combine input records=0
  Combine output records=0
  Reduce input groups=2
  Reduce shuffle bytes=448
  Reduce input records=32
  Reduce output records=0
  Spilled Records=64
  Shuffled Maps =16
  Failed Shuffles=0
  Merged Map outputs=16
  GC time elapsed (ms)=828
  Total committed heap usage (bytes)=2527027200
Shuffle Errors
  BAD_ID=0
  CONNECTION=0
  IO_ERROR=0
  WRONG_LENGTH=0
  WRONG_MAP=0
  WRONG_REDUCE=0
File Input Format Counters
  Bytes Read=1888
File Output Format Counters
  Bytes Written=97
Job Finished in 9.169 seconds
Estimated value of Pi is 3.14250000000000000000
hduser@InovaZ hadoop$
```

Conclusão

Instalação do Hadoop.

■ Próxima aula

- Mecanismo e Estrutura da Ferramenta.



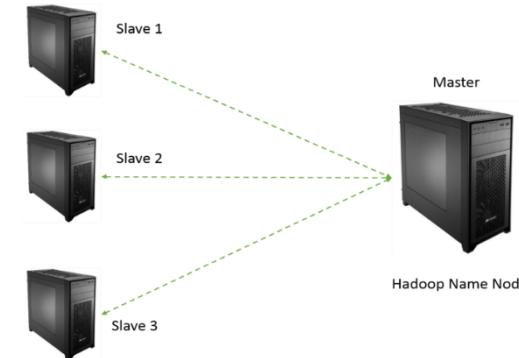
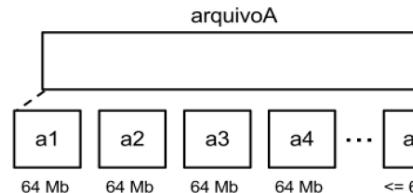
Aula 2.6. Mecanismo e estrutura da ferramenta

Nesta aula

- ❑ Mecanismos e estrutura da ferramenta.
- ❑ Arquivos de configuração.

Mecanismo e estrutura da ferramenta

- O processo começa com o arquivo de entrada.
- MapReduce divide o(s) arquivo(s) de entrada em M partições.
- Esse processo inicia cópias do programa no cluster.
- Uma das cópias é o Master e o resto Workers.



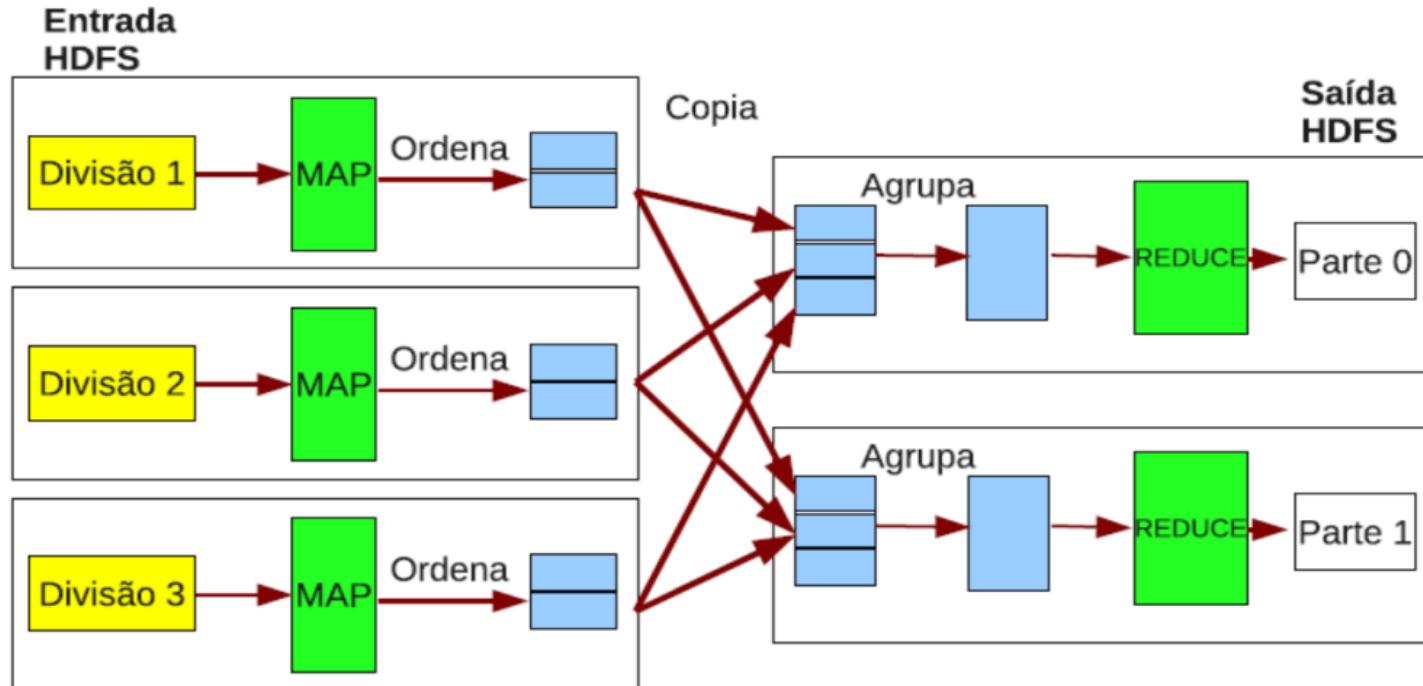
Mecanismo e estrutura da ferramenta

- Master escolhe Workers inativos e atribui tarefas.
- O Worker analisa os dados de entrada e envia à função Map.
- O resultado da função Map é armazenado em buffer.
- Periodicamente os dados do buffer são escritos em disco.

Mecanismo e estrutura da ferramenta

- A localização do resultado do Map é enviada ao master.
- Os dados são ordenados e agrupados.
- O Worker Reduce interage sobre os dados.
- Quando todas as tarefas Map e Reduce são completadas, o master é notificado.
- O resultado fica disponível em arquivos de saída.

Mecanismo e estrutura da ferramenta



Arquivos de configuração

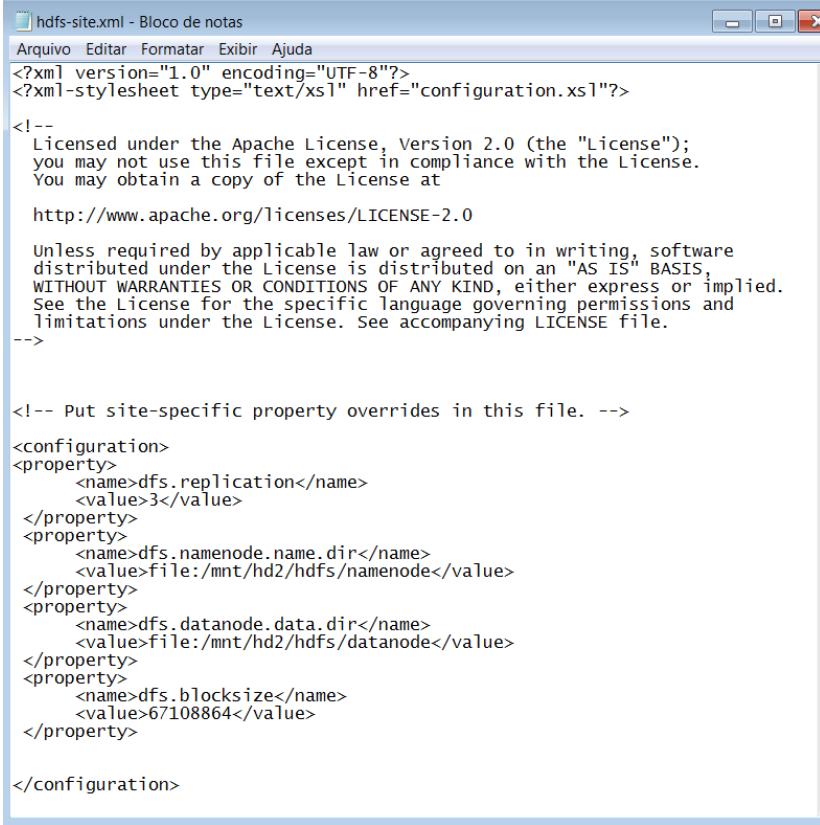
- Hadoop possui diversos arquivos de configuração.
- XML.
- Hadoop-env.sh: variáveis de ambiente do Hadoop.
- Core-site.xml: parâmetros importantes para o núcleo do Hadoop.
- Hdfs-site.xml: configurações do sistema de arquivos (HDFS).
- Mapred-site.xml: configurações do MapReduce.



Arquivos de configuração

- Slaves:
 - Lista de máquinas do cluster (não é XML).
 - Cada linha é uma máquina Worker (Datanode).
- Masters:
 - Lista de máquinas Master do cluster (não é XML).
 - Cada linha é uma máquina Master (Namenode).

Arquivos de configuração



The screenshot shows a Windows Notepad window with the title "hdfs-site.xml - Bloco de notas". The content of the file is an XML configuration for HDFS. It includes a license notice, a general disclaimer, and specific property overrides for replication, namenode, datanode, and blocksize.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!--
Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

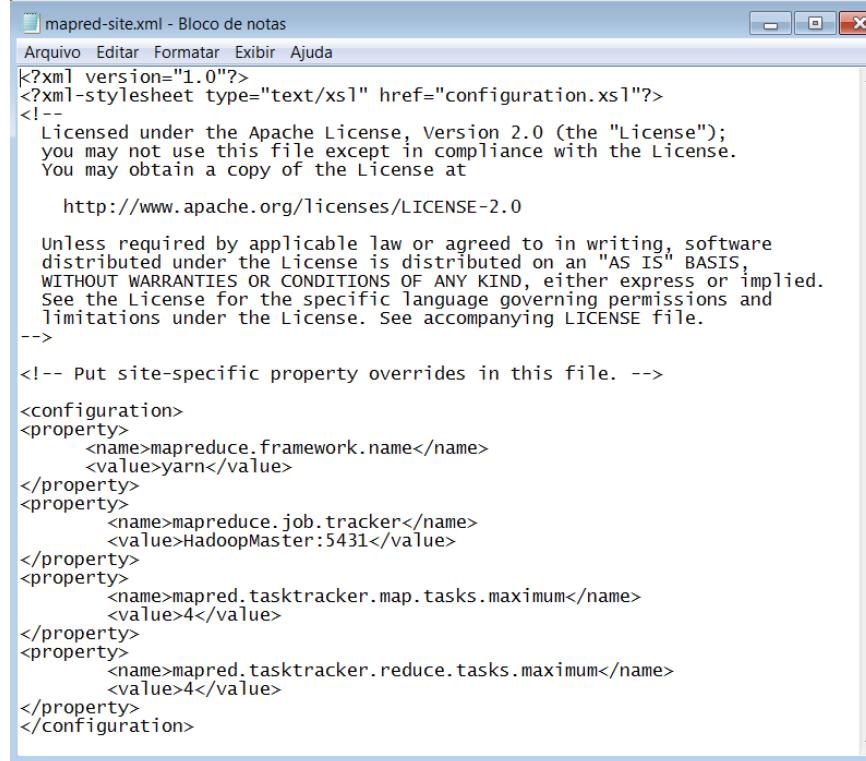
Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
<property>
    <name>dfs.replication</name>
    <value>3</value>
</property>
<property>
    <name>dfs.namenode.name.dir</name>
    <value>file:/mnt/hd2/hdfs/namenode</value>
</property>
<property>
    <name>dfs.datanode.data.dir</name>
    <value>file:/mnt/hd2/hdfs/datanode</value>
</property>
<property>
    <name>dfs.blocksize</name>
    <value>67108864</value>
</property>

</configuration>
```

Arquivos de configuração



```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
<property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
</property>
<property>
    <name>mapreduce.job.tracker</name>
    <value>HadoopMaster:5431</value>
</property>
<property>
    <name>mapred.tasktracker.map.tasks.maximum</name>
    <value>4</value>
</property>
<property>
    <name>mapred.tasktracker.reduce.tasks.maximum</name>
    <value>4</value>
</property>
</configuration>
```

Conclusão

Mecanismos e estrutura da ferramenta:

- Map, reduce, workers e master.

Arquivos de configuração:

- mapred-site.xml, mapred-site.xml, core-site.xml, slaves e masters.

❑ Funções:

- Map.
- Reduce.
- Combine.



Aula 2.7. Funções map, reduce e combine

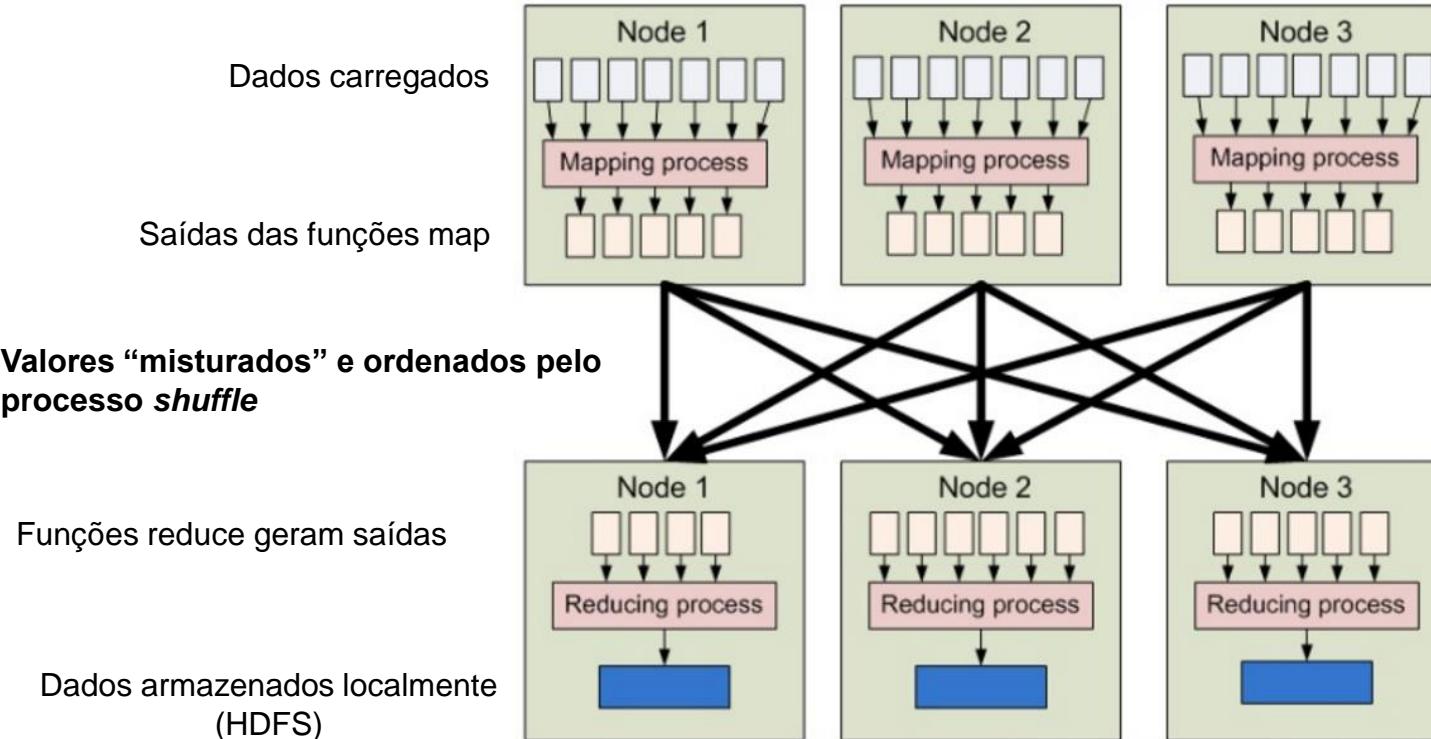
Funções:

- Map.
- Reduce.
- Combine.

Funções map, reduce e combine

- Hadoop trabalha dividindo os trabalhos em duas funções distintas e principais: map e reduce.
- A forma de trabalho de cada função é definida pelo programador.
- Recebem como entrada pares chave-valor.
- Os tipos de dados desses pares são definidos pelo programador.
- O modelo fornece uma função intermediária: combine.

Funções map, reduce e combine



Funções map, reduce e combine

- Dado um arquivo.
- Grande volume.
- Com vendas realizadas por uma empresa.

Funções map, reduce e combine

IGTI

- Dados de entrada. Arquivo(s) no HDFS:

```
050534564201403201645464564000084665202114205213037897435120000500004546654564564564564631231230003586453210
050534564201403201645464564564000084665202114205213037897435120000302004546654564564564564631231230003586453210
050534564200507201645464564564000084665202114205213037897435120000507754546654564564564564631231230003586453210
050534564200705201645464564564000084665202114205213037897435120000599244546654564564564564564631231230003586453210
050534564201403201645464564564000084665202114205213037897435120000512234546654564564564564631231230003586453210
05053456420084201645464564564000084665202114205213037897435120000637224546654564564564564631231230003586453210
050534564202001201645464564564000084665202114205213037897435120001419254546654564564564564631231230003586453210
050534564201403201645464564564000084665202114205213037897435120002329224546654564564564564631231230003586453210
050534564050804201645464564564000084665202114205213037897435120006258804546654564564564631231230003586453210
050534564200705201645464564564000084665202114205213037897435120000080244546654564564564564631231230003586453210
050534564200507201645464564564000084665202114205213037897435120000637884546654564564564564631231230003586453210
050534564200804201645464564564000084665202114205213037897435120001255594546654564564564564631231230003586453210
050534564201403201645464564564000084665202114205213037897435120000832694546654564564564564631231230003586453210
050534564020804201645464564564000084665202114205213037897435120000369694546654564564564564631231230003586453210
050534564201305201645464564564000084665202114205213037897435120000333694546654564564564564631231230003586453210
050534564200705201645464564564000084665202114205213037897435120000635854546654564564564564631231230003586453210
050534564200804201645464564564000084665202114205213037897435120000120004546654564564564564631231230003586453210
050534564200507201645464564564000084665202114205213037897435120000825504546654564564564564631231230003586453210
050534564201403201645464564564000084665202114205213037897435120000637504546654564564564564631231230003586453210
050534564200705201645464564564000084665202114205213037897435120000612124546654564564564564564631231230003586453210
050534564201305201645464564564000084665202114205213037897435120000137694546654564564564564631231230003586453210
0505345642008112016454645645640000846652021142052130378974351200001819364546654564564564631231230003586453210
0505345642008112016454645645640000846652021142052130378974351200001369524546654564564564564631231230003586453210
0505345642008112016454645645640000846652021142052130378974351200008520694546654564564564564631231230003586453210
0505345642008112016454645645640000846652021142052130378974351200006352524546654564564564564631231230003586453210
050534564201403201645464564564000084665202114205213037897435120000368854546654564564564564631231230003586453210
05053456420001201645464564564000084665202114205213037897435120000237854546654564564564564631231230003586453210
05053456420001201645464564564000084665202114205213037897435120000100254546654564564564564631231230003586453210
```

050534564200705201645464564564000084665202114205213037897435120000888964546654564564564631231230003586453210

Funções map, reduce e combine

- Posições 12 até 19 referem-se às datas das vendas.
- Posições 63 até 72 referem-se aos valores das vendas.

Funções map, reduce e combine

- Map:

05053456420**14032016**4546456456400008466520211420521303789743512**0000050000**4546654564564564564631231230003586453210
05053456420**14032016**4546456456400008466520211420521303789743512**0000030200**4546654564564564564631231230003586453210
05053456420**05072016**4546456456400008466520211420521303789743512**0000050775**4546654564564564564631231230003586453210
05053456420**07052016**4546456456400008466520211420521303789743512**0000059924**4546654564564564564631231230003586453210
05053456420**14032016**4546456456400008466520211420521303789743512**0000051223**4546654564564564564631231230003586453210
05053456420**08042016**4546456456400008466520211420521303789743512**0000063722**4546654564564564564631231230003586453210
05053456420**20012016**4546456456400008466520211420521303789743512**0000141925**4546654564564564564631231230003586453210
05053456420**14032016**4546456456400008466520211420521303789743512**0000232922**4546654564564564564631231230003586453210
05053456405**08042016**4546456456400008466520211420521303789743512**0000625880**4546654564564564564631231230003586453210
05053456420**07052016**4546456456400008466520211420521303789743512**0000008024**4546654564564564564631231230003586453210
05053456420**05072016**4546456456400008466520211420521303789743512**0000063788**4546654564564564564631231230003586453210
05053456420**08042016**4546456456400008466520211420521303789743512**0000125559**4546654564564564564631231230003586453210
05053456420**14032016**4546456456400008466520211420521303789743512**0000083269**4546654564564564564631231230003586453210
05053456402**08042016**4546456456400008466520211420521303789743512**0000036969**4546654564564564564631231230003586453210
05053456420**13052016**4546456456400008466520211420521303789743512**0000033369**4546654564564564564631231230003586453210
05053456420**07052016**4546456456400008466520211420521303789743512**0000063585**4546654564564564564631231230003586453210
05053456420**08042016**4546456456400008466520211420521303789743512**0000012000**4546654564564564564631231230003586453210
05053456420**05072016**4546456456400008466520211420521303789743512**0000082550**4546654564564564564631231230003586453210
05053456420**14032016**4546456456400008466520211420521303789743512**0000063750**4546654564564564564631231230003586453210
05053456420**07052016**4546456456400008466520211420521303789743512**0000061212**4546654564564564564631231230003586453210
05053456420**13052016**4546456456400008466520211420521303789743512**0000013769**4546654564564564564631231230003586453210
05053456420**08112016**4546456456400008466520211420521303789743512**0000181936**4546654564564564564631231230003586453210
05053456420**08112016**4546456456400008466520211420521303789743512**0000136952**4546654564564564564631231230003586453210
05053456420**08112016**4546456456400008466520211420521303789743512**0000852069**4546654564564564564631231230003586453210
05053456420**08112016**4546456456400008466520211420521303789743512**0000635252**4546654564564564564631231230003586453210
05053456420**14032016**4546456456400008466520211420521303789743512**0000036885**4546654564564564564631231230003586453210
05053456420**20012016**4546456456400008466520211420521303789743512**0000023785**4546654564564564564631231230003586453210
05053456420**20012016**4546456456400008466520211420521303789743512**0000010025**4546654564564564564631231230003586453210

05053456420**07052016**4546456456400008466520211420521303789743512**0000088896**4546654564564564564631231230003586453210

Funções map, reduce e combine

- Map.
- Combine.
- Agrupamento e ordenação (Shuffle).
- Reduce.

Funções map, reduce e combine

IGTI

■ Map:

05053456420**14032016**4546456456400008466520211420521303789743512**0000050000**4546654564564564564564631231230003586453210
05053456420**14032016**4546456456400008466520211420521303789743512**0000030200**4546654564564564564564631231230003586453210
05053456420**05072016**4546456456400008466520211420521303789743512**0000050775**4546654564564564564564631231230003586453210
05053456420**07052016**4546456456400008466520211420521303789743512**0000059924**4546654564564564564564631231230003586453210
05053456420**14032016**4546456456400008466520211420521303789743512**0000051223**4546654564564564564564631231230003586453210
05053456420**08042016**4546456456400008466520211420521303789743512**0000063722**4546654564564564564564631231230003586453210
05053456420**20012016**4546456456400008466520211420521303789743512**0000141925**4546654564564564564564631231230003586453210
05053456420**14032016**4546456456400008466520211420521303789743512**0000232922**4546654564564564564564631231230003586453210
05053456405**08042016**4546456456400008466520211420521303789743512**0000625880**4546654564564564564564631231230003586453210
05053456420**07052016**4546456456400008466520211420521303789743512**0000008024**4546654564564564564564631231230003586453210
05053456420**05072016**4546456456400008466520211420521303789743512**0000063788**4546654564564564564564631231230003586453210
05053456420**08042016**4546456456400008466520211420521303789743512**0000125559**4546654564564564564564631231230003586453210
05053456420**14032016**4546456456400008466520211420521303789743512**0000083269**4546654564564564564564631231230003586453210
05053456402**08042016**4546456456400008466520211420521303789743512**0000036969**4546654564564564564564631231230003586453210
05053456420**13052016**4546456456400008466520211420521303789743512**0000033369**4546654564564564564564631231230003586453210
05053456420**07052016**4546456456400008466520211420521303789743512**0000063585**4546654564564564564564631231230003586453210
05053456420**08042016**4546456456400008466520211420521303789743512**0000012000**4546654564564564564564631231230003586453210
05053456420**05072016**4546456456400008466520211420521303789743512**0000082550**4546654564564564564564631231230003586453210
05053456420**14032016**4546456456400008466520211420521303789743512**0000063750**4546654564564564564564631231230003586453210
05053456420**07052016**4546456456400008466520211420521303789743512**0000061212**4546654564564564564564631231230003586453210
05053456420**13052016**4546456456400008466520211420521303789743512**0000013769**4546654564564564564564631231230003586453210
05053456420**08112016**4546456456400008466520211420521303789743512**0000181936**4546654564564564564564631231230003586453210
05053456420**08112016**4546456456400008466520211420521303789743512**0000136952**4546654564564564564564631231230003586453210
05053456420**08112016**4546456456400008466520211420521303789743512**0000852069**4546654564564564564564631231230003586453210
05053456420**08112016**4546456456400008466520211420521303789743512**0000635252**4546654564564564564564631231230003586453210
05053456420**14032016**4546456456400008466520211420521303789743512**0000036885**4546654564564564564564631231230003586453210
05053456420**20012016**4546456456400008466520211420521303789743512**0000023785**4546654564564564564564631231230003586453210
05053456420**20012016**4546456456400008466520211420521303789743512**0000010025**4546654564564564564564631231230003586453210
05053456420**07052016**4546456456400008466520211420521303789743512**0000088896**4546654564564564564564631231230003586453210

Funções map, reduce e combine

```

050534564201403201645465456456400084665202114205213037897435120000500004546654564564564564631231230003586453210
05053456420140320164546545645640008466520211420521303789743512000030204546654564564564564564631231230003586453210
0505345642005072016454645645645640008466520211420521303789743512000050924546654564564564564631231230003586453210
05053456420070520164546456456456400084665202114205213037897435120000059924546654564564564564631231230003586453210
05053456420140320164546456456456400084665202114205213037897435120000512234546654564564564564631231230003586453210
05053456420080420164546456456456400084665202114205213037897435120000637224546654564564564564564631231230003586453210
05053456420200120164546456456456400084665202114205213037897435120000119254546654564564564564631231230003586453210
0505345642004032016454645645645640008466520211420521303789743512000002324546654564564564564564631231230003586453210
050534564200509720164546456456456400084665202114205213037897435120000059924546654564564564564631231230003586453210
05053456420070520164546456456456400084665202114205213037897435120000080044546654564564564564564631231230003586453210
05053456420080420164546456456456400084665202114205213037897435120000125594546654564564564564564631231230003586453210
0505345642004032016454645645645640008466520211420521303789743512000013294546654564564564564564631231230003586453210
05053456420080420164546456456456400084665202114205213037897435120000036984546654564564564564564631231230003586453210
05053456420040320164546456456456400084665202114205213037897435120000033984546654564564564564564631231230003586453210
05053456420070520164546456456456400084665202114205213037897435120000012004546654564564564564564631231230003586453210
05053456420050720164546456456456400084665202114205213037897435120000825504546654564564564564564631231230003586453210
05053456420070520164546456456456400084665202114205213037897435120000637504546654564564564564564631231230003586453210
05053456420040320164546456456456400084665202114205213037897435120000112142546654564564564564564631231230003586453210
05053456420070520164546456456456400084665202114205213037897435120000137694546654564564564564564631231230003586453210
05053456420080420164546456456456400084665202114205213037897435120000118964546654564564564564564631231230003586453210
05053456420080420164546456456456400084665202114205213037897435120000120084546654564564564564564631231230003586453210
05053456420080420164546456456456400084665202114205213037897435120000820684546654564564564564564631231230003586453210
05053456420080420164546456456456400084665202114205213037897435120000368854546654564564564564564631231230003586453210
05053456420200120164546456456456400084665202114205213037897435120000237854546654564564564564564631231230003586453210
05053456420070520164546456456456400084665202114205213037897435120000110254546654564564564564564631231230003586453210

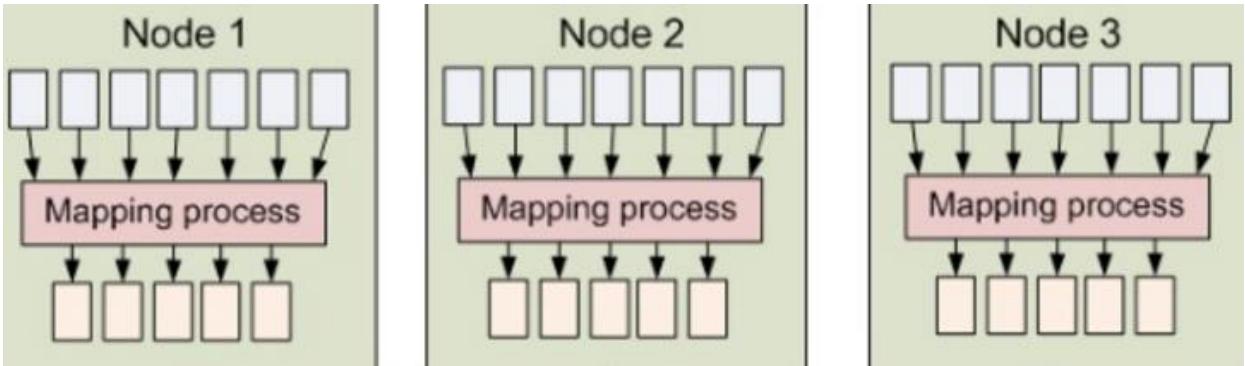
```

Entrada para os Mappers

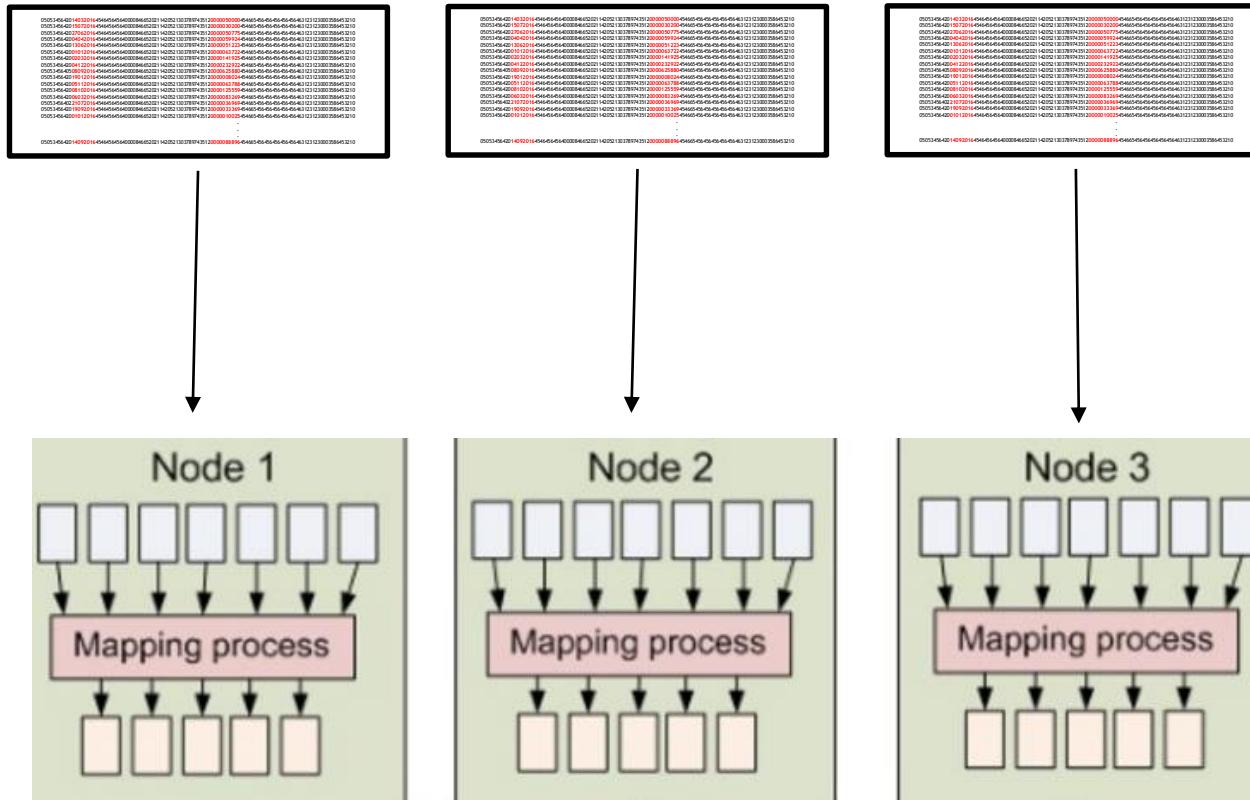
Dataset particionado

0505345642007052016454645645640008466520211420521303789743512000088964546654564564564564631231230003586453210

⋮



Funções map, reduce e combine



Funções map, reduce e combine

- O que a função **map** vai fazer?
- Cada linha do arquivo é uma execução do Map.
- Retirar de cada linha somente o que é útil.

Funções map, reduce e combine

<CHAVE, VALOR>

14032016	0000050000
14032016	0000030200
05072016	0000050775
07052016	0000059924
14032016	0000051223
08042016	0000063722
20012016	0000141925
14032016	0000232922
08042016	0000625880
07052016	0000008024
05072016	0000063788
08042016	0000125559
14032016	0000083269
08042016	0000036969
13052016	0000033369
07052016	0000063585
08042016	0000012000
05072016	0000082550
14032016	0000063750
07052016	0000061212
13052016	0000013769
08112016	0000181936
08112016	0000136952
08112016	0000852069
08112016	0000635252
14032016	0000036885
20012016	0000023785
20012016	0000010025
	.
	.
	.
07052016	0000088896

Funções map, reduce e combine

- Parâmetros da função Map:
 - **Key**: gerada automaticamente pelo framework.
 - **Value**: toda a linha do arquivo:
 - 05053456420**14032016**4546456456400008466520211420521303789743512**0000050000**4546654564564564564631231230003586453210

```
public static class MapStageIGTI extends MapReduceBase implements Mapper<LongWritable, Text, Text, Text> {  
    public void map(LongWritable key, Text value, OutputCollector<Text, Text> output, Reporter reporter) throws  
    IOException  
    {  
        Text txtKey = new Text();  
        Text txtValue = new Text();  
  
        String Data = value.substring(12, 19);  
        txtKey.set(Data);  
  
        String Valor = value.substring(63,72);  
        txtValue.set(Valor);  
  
        output.collect(txtKey, txtValue);  
    }  
}
```

Funções map, reduce e combine

-

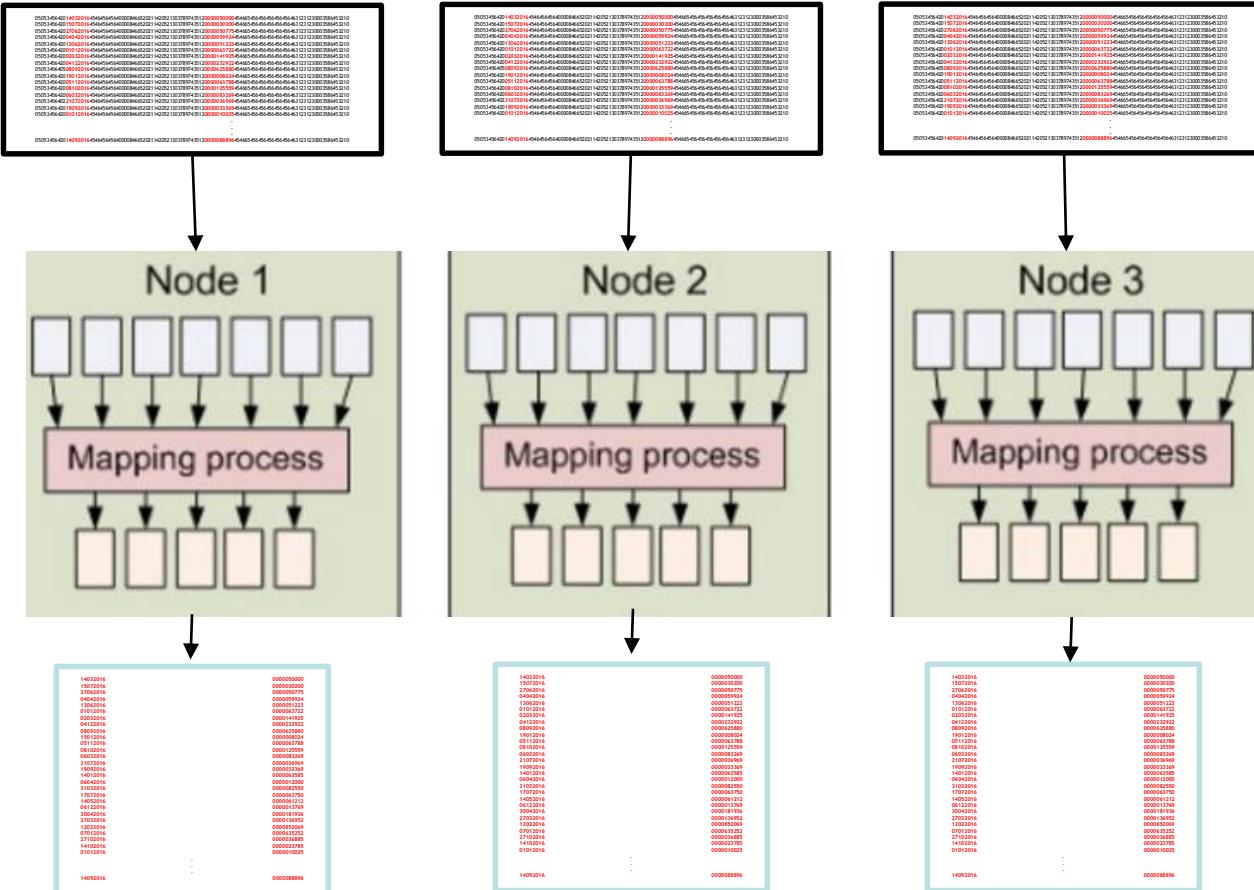
Parâmetros da função Map:

- **Key**: gerada automaticamente pelo framework.
- **Value**: toda a linha do arquivo:
 - 05053456420**14032016**4546456456400008466520211420521303789743512**0000050000**4546654564564564564631231230003
586453210

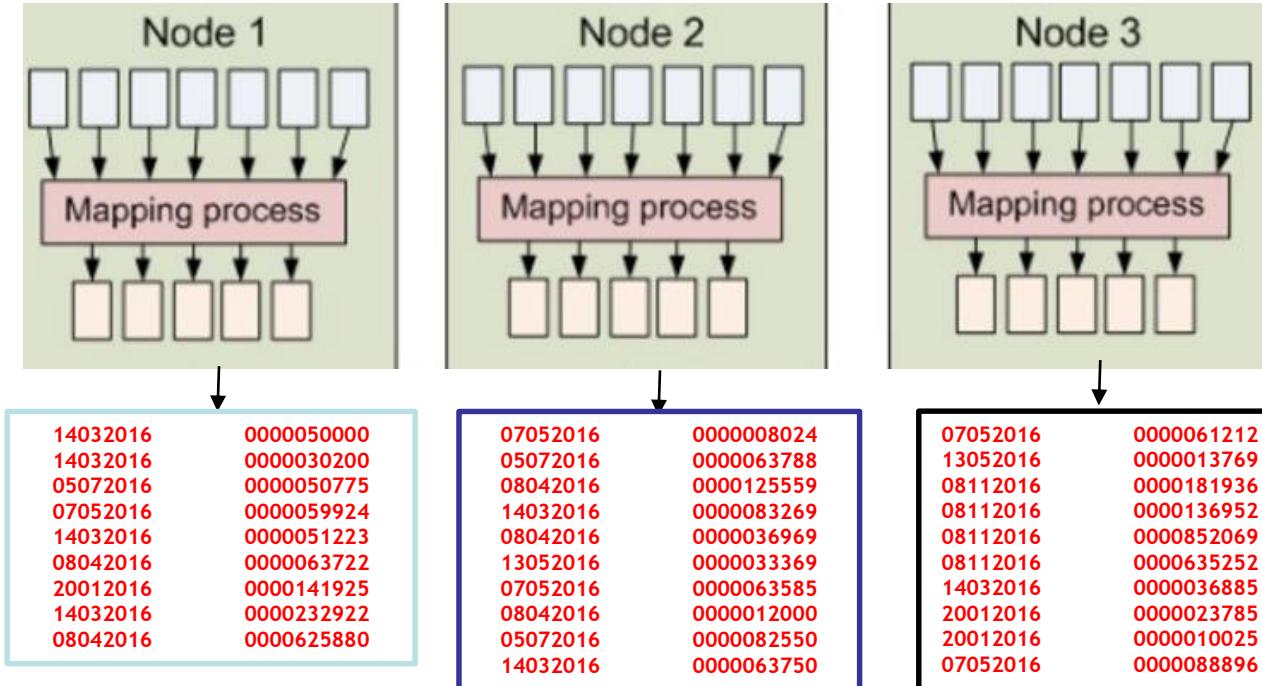
```
public static class MapStageIGTI extends MapReduceBase implements Mapper<LongWritable, Text, Text, Text> {  
    public void map(LongWritable key, Text value, OutputCollector<Text, Text> output, Reporter reporter) throws IOException  
    {  
        Text txtKey = new Text();  
        Text txtValue = new Text();  
  
        String Data = value.substring(12, 19);  
        txtKey.set(Data);  
        14032016  
        String Valor = value.substring(63,72);  
        txtValue.set(Valor);  
        0000050000  
        output.collect(txtKey, txtValue);  
    }  
}  
      14032016 0000050000
```



Funções map, reduce e combine



Funções map, reduce e combine

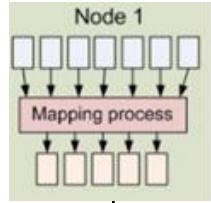


Saídas das funções map

Funções map, reduce e combine

- Map.
- **Combine.**
- Agrupamento e ordenação (Shuffle).
- Reduce.

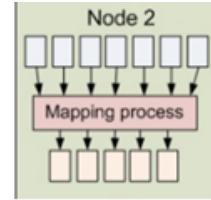
Funções map, reduce e combine



14032016	0000050000
14032016	0000030200
05072016	0000050775
07052016	0000059924
14032016	0000051223
08042016	0000063722
20012016	0000141925
14032016	0000232922
08042016	0000625880

Combine

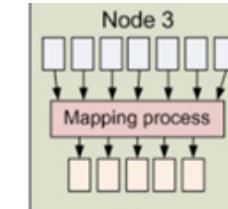
05072016	0000050775
07052016	0000059924
20012016	0000141925
14032016	0000232922
08042016	0000625880



07052016	0000008024
05072016	0000063788
08042016	0000125559
14032016	0000083269
08042016	0000036969
13052016	0000033369
07052016	0000063585
08042016	0000012000
05072016	0000082550
14032016	0000063750

Combine

08042016	0000125559
14032016	0000083269
13052016	0000033369
07052016	0000063585
05072016	0000082550



07052016	0000061212
13052016	0000013769
08112016	0000181936
08112016	0000136952
08112016	0000852069
08112016	0000635252
14032016	0000036885
20012016	0000023785
20012016	0000010025
07052016	0000088896

Combine

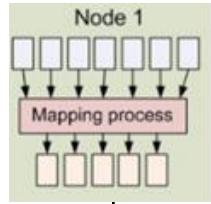
13052016	0000013769
08112016	0000852069
14032016	0000036885
20012016	0000023785
07052016	0000088896

Cada função combine é executada no próprio datanode

Funções map, reduce e combine

- Map.
- Combine.
- Agrupamento e ordenação (Shuffle).
- Reduce.

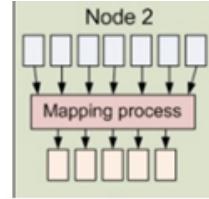
Funções map, reduce e combine



05072016	0000050775
07052016	0000059924
20012016	0000141925
14032016	0000232922
08042016	0000625880

Ordenação

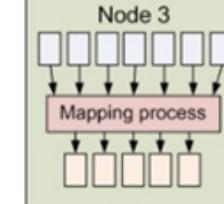
20012016	0000141925
14032016	0000232922
08042016	0000625880
07052016	0000059924
05072016	0000050775



08042016	0000125559
14032016	0000083269
13052016	0000033369
07052016	0000063585
05072016	0000082550

Ordenação

14032016	0000083269
08042016	0000125559
07052016	0000063585
13052016	0000033369
05072016	0000082550

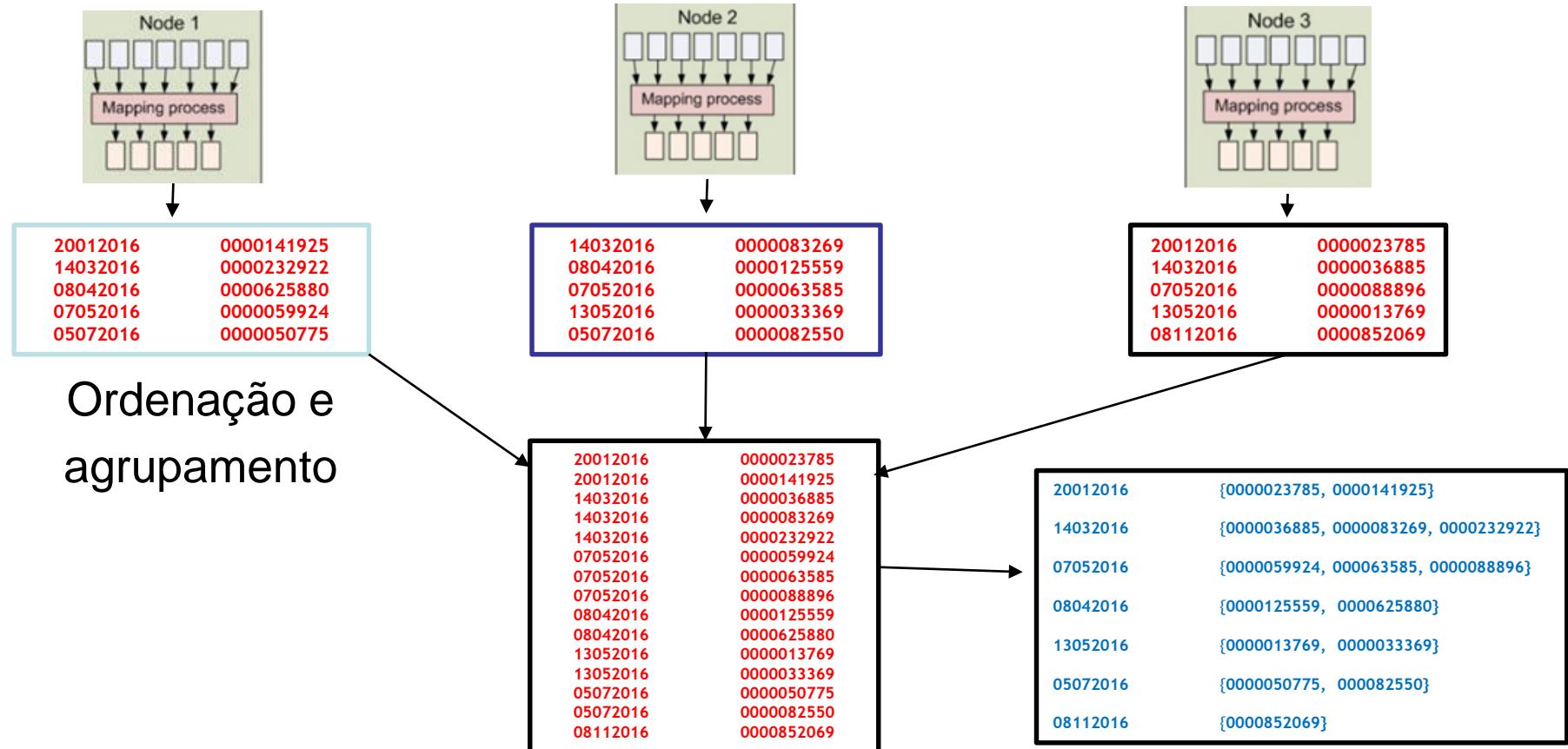


13052016	0000013769
08112016	0000852069
14032016	0000036885
20012016	0000023785
07052016	0000088896

Ordenação

20012016	0000023785
14032016	0000036885
07052016	0000088896
13052016	0000013769
08112016	0000852069

Funções map, reduce e combine



Funções map, reduce e combine

- Map.
- Combine.
- Agrupamento e ordenação (Shuffle).
- Reduce.

Funções map, reduce e combine

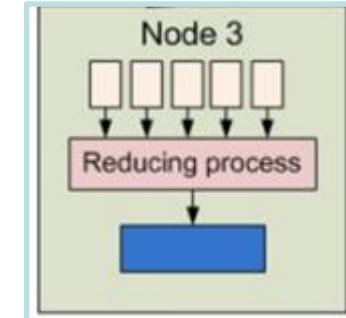
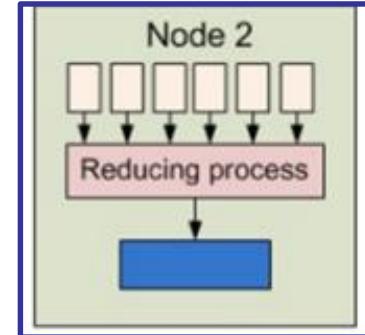
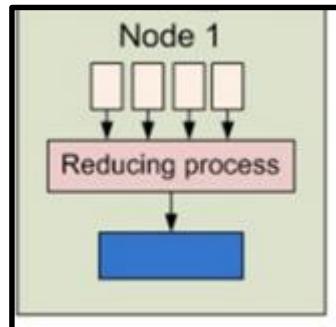
20012016	{0000023785, 0000141925}
14032016	{0000036885, 0000083269, 0000232922}
07052016	{0000059924, 000063585, 0000088896}
08042016	{0000125559, 0000625880}
13052016	{0000013769, 0000033369}
05072016	{0000050775, 000082550}
08112016	{0000852069}

Entrada de cada fase reduce

05072016	{0000050775, 000082550}
08112016	{0000852069}

08042016	{0000125559, 0000625880}
13052016	{0000013769, 0000033369}

20012016	{0000023785, 0000141925}
14032016	{0000036885, 0000083269, 0000232922}
07052016	{0000059924, 000063585, 0000088896}



Funções map, reduce e combine

- Como encontrar a maior venda do dia?

Funções map, reduce e combine

- Parâmetros da função Reduce:
 - Key: **14032016 (A Data)**
 - Value: Uma lista com todos os values (vendas)
 - **{0000036885, 0000083269, 0000232922}**

```
public static class RedStageIGTI extends MapReduceBase implements Reducer<Text, Text, Text, Text>{  
    public void reduce (Text key, Iterator<Text> values, OutputCollector<Text, Text> output, Reporter reporter) throws IOException {  
        Text vendaCorrente = new Text();  
        Double maiorVenda = 0;  
        Text txtMaiorVenda = new Text();  
  
        while (values.hasNext()) {  
            vendaCorrente = values.next();  
            if (vendaCorrente.toDouble() > maiorVenda )  
                maiorVenda = vendaCorrente;  
        }  
  
        txtMaiorVenda.set(maiorVenda.toString());  
        output.collect(key, txtMaiorVenda);  
    }  
}
```

Funções map, reduce e combine

- Parâmetros da função Reduce:
 - Key: **14032016 (A Data)**
 - Value: Uma lista com todos os values (vendas)
 - **{0000036885, 0000083269, 0000232922}**

```
public static class RedStageIGTI extends MapReduceBase implements Reducer<Text, Text, Text, Text>{  
    public void reduce (Text key, Iterator<Text> values, OutputCollector<Text, Text> output, Reporter reporter) throws IOException {  
        Text vendaCorrente = new Text();  
        Double maiorVenda = 0;  
        Text txtValue = new Text();  
  
        {0000036885, 0000083269, 0000232922}  
        while (values.hasNext()) {  
            vendaCorrente = values.next();  
            if (vendaCorrente.toDouble() > maiorVenda )  
                maiorVenda = vendaCorrente;  
        }  
  
        txtValue.set(maiorVenda.toString());  
        output.collect(key, txtValue);  
    }  
    14032016  
}
```



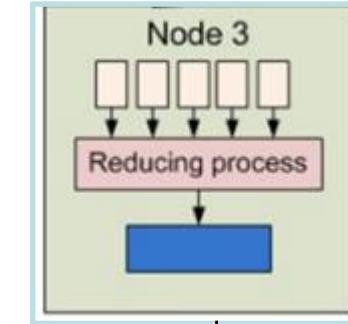
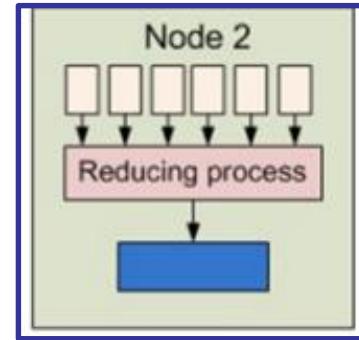
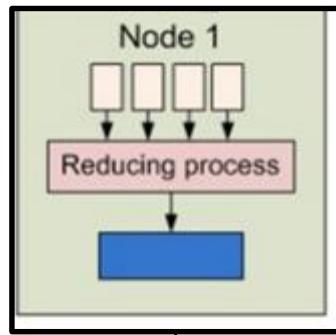
Funções map, reduce e combine

Entrada de cada fase reduce

05072016 {0000050775, 000082550}
08112016 {0000852069}

08042016 {0000125559, 0000625880}
13052016 {0000013769, 0000033369}

20012016 {0000023785, 0000141925}
14032016 {0000036885, 0000083269, 0000232922}
07052016 {0000059924, 000063585, 0000088896}



05072016 {000082550}
08112016 {0000852069}

08042016 {0000625880}
13052016 {0000033369}

20012016 {0000141925}
14032016 {0000232922}
07052016 {0000088896}

Saída das fases reduce para o HDFS

Conclusão

- Map.
- Reduce.
- Combine.

■ Próxima aula

- Tolerância a falhas.

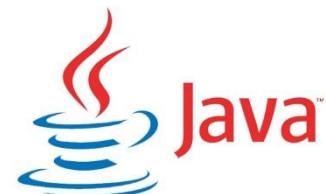


Aula 2.8. Tolerância a falhas

- Tolerância a falhas.

- Aplicações podem se deparar com problemas:
 - Erros no código.
 - Falhas de máquinas (memória, disco, etc.).
 - Processos inesperadamente encerrados.
 - Falhas na rede.
- Um dos maiores benefícios do Hadoop é sua habilidade em lidar com falhas.
- Habilidade para garantir a conclusão dos trabalhos já iniciados.

- Quando um erro acontece a JVM comunica o fato ao TaskTracker (processo responsável pela execução de tarefas MapReduce).
- TaskTracker executa uma tarefa Map ou Reduce atribuída.
- O erro é registrado em um arquivo de log.
- O TaskTracker marca a tarefa com o status de falha.
- Libera o slot para execução de nova tarefa.



- Quando o TaskTracker fica muito tempo sem receber informações de um nó, a tarefa é considerada nula.
- Será identificado que aquele nó falhou.
- Nesse momento a tarefa será novamente agendada.
- TaskTracker tenta evitar novo agendamento para o mesmo nó.

- Caso uma tarefa falhe sucessivas vezes (parâmetro).
- Isso poderá resultar no cancelamento da tarefa.
- Nem sempre do trabalho total.
- Existe um parâmetro para definir qual o percentual aceito.

- Um dos maiores problemas em um cluster Hadoop é a falha do nó mestre.
- Por se tratar do nó controlador do cluster.
- Se não existir redundância do nó mestre, o Hadoop não conseguirá se recuperar dessa falha.

Tolerância a falhas:

- Falhas no código ou estrutura.
- Recuperação de falhas.
- Nó mestre e nó escravo.

■ Próxima aula

- Formatos de entrada e saída.



Aula 2.9. Formatos de entrada e saída

- Formatos de entrada e saída do Hadoop.

Formatos de entrada e saída

- Formatos mais comuns: csv e txt.
- Mas não são somente esses.
- O Hadoop pode processar dados não-estruturados, como vídeo, áudio, texto, etc.
- A maioria dos dados são estruturados ou semiestruturados.

Formatos de entrada e saída

- A importância do formato do arquivo para a operação de split.
- Os dados que o Hadoop processa são divididos em blocos.
- Precisamos estar disponíveis para ler os dados em qualquer ponto do arquivo, mesmo fora da ordem. O processamento distribuído exige isso.

Formatos de entrada e saída

- Os arquivos csv têm facilidade para a realização do processo de split.
- Podemos começar a ler os csv's de qualquer ponto.
- Um arquivo XML é diferente. Não podemos começar a ler o arquivo no meio de uma tag.
- Se nossos arquivos forem menores que o bloco (blocksize) do HDFS, pode ser que não tenhamos problemas com o Split.
- Entretanto, arquivos pequenos são a exceção no Hadoop.

Formatos de entrada e saída

- Muitos pequenos arquivos pode ocasionar problemas de desempenho.
- Arquivos csv são muito comuns para troca de informações entre o Hadoop e sistemas externos.
- Registros JSON são diferentes de arquivos JSON. Eles tornam os arquivos divisíveis.

Formatos de entrada e saída

- Sequence files.
- Armazena dados em formato binário.
- Utilizado para armazenamento de dados complexos.
- Muito utilizado para processamento de áudio e vídeo.

Conclusão

- Formatos de entrada e saída.



Desenvolvimento de Soluções com MapReduce utilizando Hadoop

Capítulo 3. O HDFS (Hadoop Distributed Filesystem)

Prof. João Paulo Barbosa Nascimento



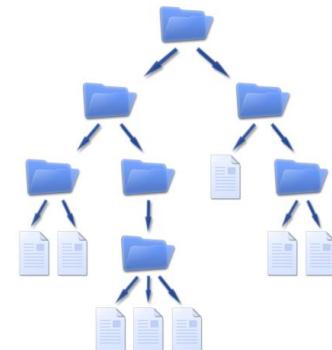
Aula 3.1. Formato, conceitos e comandos básicos do HDFS

Nesta aula

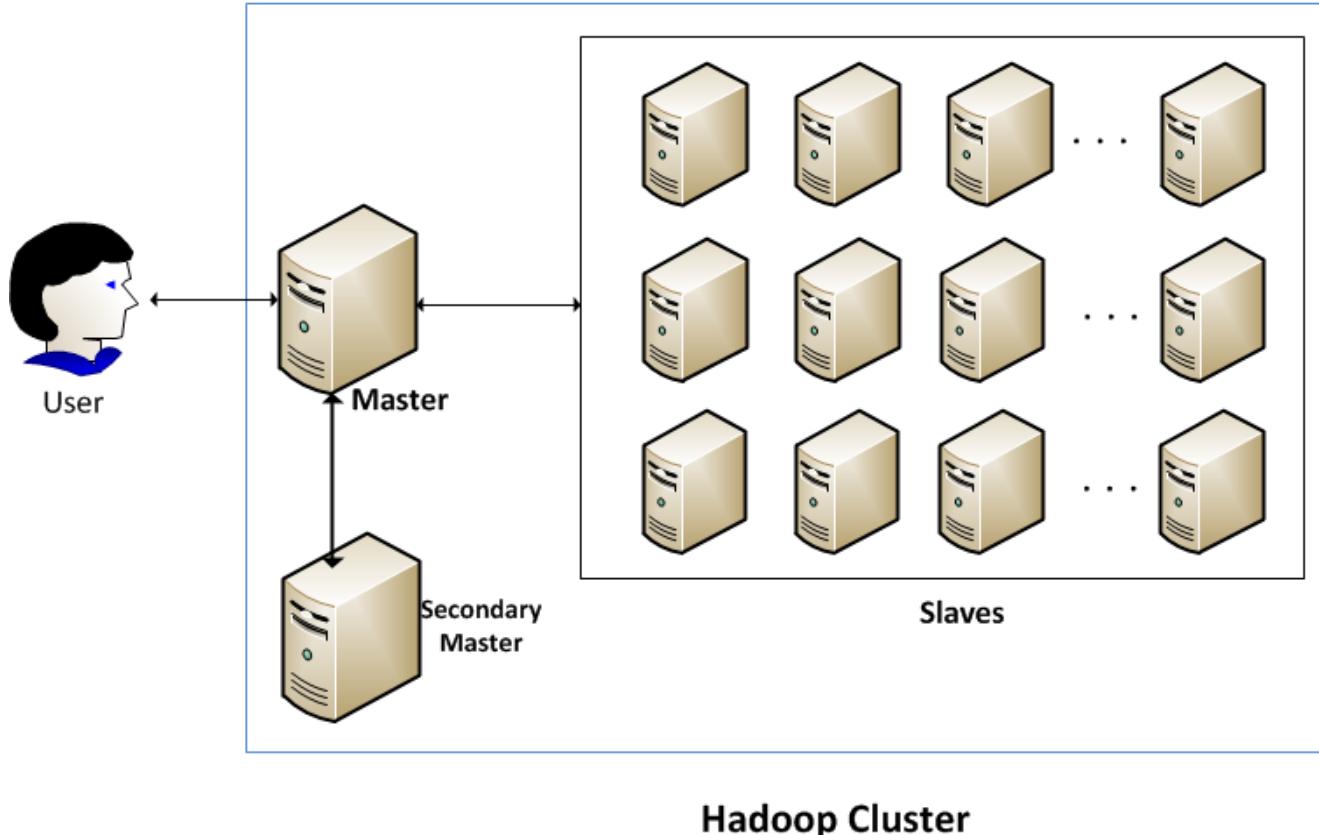
- ❑ Formato do HDFS.
- ❑ Conceitos e Comandos Básicos do HDFS.

Formato do HDFS

- Sistema de arquivos distribuídos:
 - Conjunto de dados (*dataset*) extrapola a capacidade de armazenamento de uma máquina.
 - Necessário realizar o particionamento desse *dataset* através de outras máquinas.
 - Interligadas em cluster.
 - Sistemas de arquivos distribuídos: gerenciam e armazenam dados através de uma rede.
 - Sistema de arquivos mais complexo que sistemas que atuam localmente.
 - Maior desafio: gerenciar e tolerar falhas. Evitar a perda de dados.
 - Hadoop possui seu próprio sistema de arquivos distribuído.



Formato do HDFS



Formato do HDFS

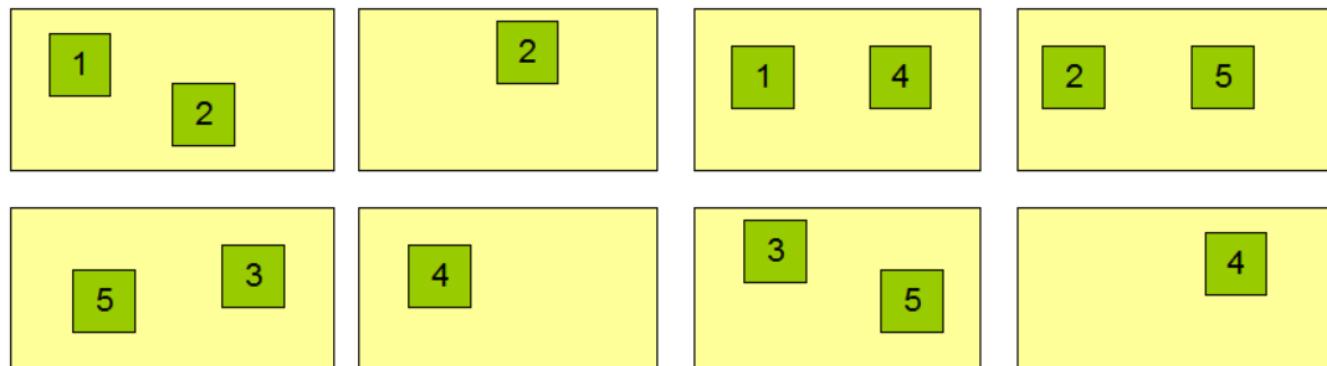
- HDFS:
 - Sistema de arquivos distribuídos.
 - Projetado para armazenar arquivos grandes.
 - Distribuído em grandes clusters.
- Grandes Arquivos:
 - Arquivos com centenas de Megabytes, Gigabytes ou Terabytes de tamanho.
 - Alguns clusters Hadoop já trabalham com Petabytes.
 - HDFS está preparado também para trabalhar com vários pequenos arquivos.
 - Diversos estudos comparam o comportamento da ferramenta nas duas situações.
 - Granularidade.

Formato do HDFS

Block Replication

```
Namenode (Filename, numReplicas, block-ids, ...)  
/users/sameerp/data/part-0, r:2, {1,3}, ...  
/users/sameerp/data/part-1, r:3, {2,4,5}, ...
```

Datanodes



Formato do HDFS

- Acesso aos Dados
 - Baseado em um padrão eficiente de processamento.
 - Escreva uma vez e leia muitas vezes.
 - Dados copiados de uma fonte e diversos tipos de análises são feitas.
 - Essas análises podem envolver todo o conjunto ou apenas uma parte.
- Hardware
 - HDFS não exige um grande e caro cluster.
 - Hadoop foi projetado para executar em clusters homogêneos ou heterogêneos.



Conceitos e comandos básicos do HDFS

- Para interagir com o HDFS o usuário utiliza linhas de comando.
- Existem outras formas de interação.
- Linhas de comando é a forma mais eficiente.
- Mais familiar aos desenvolvedores.

```
[root@sandbox ~]# hadoop fs -help
Usage: hadoop fs [generic options]
  [-appendToFile <localsrc> ... <dst>]
  [-cat [-ignoreCrc] <src> ...]
  [-checksum <src> ...]
  [-chgrp [-R] GROUP PATH...]
  [-chmod [-R] <MODE[,MODE]... | OCTALMODE> PATH...]
  [-chown [-R] [OWNER][:GROUP]] PATH...
  [-copyFromLocal [-f] [-p] [-l] <localsrc> ... <dst>]
  [-copyToLocal [-p] [-ignorecrc] [-crc] <src> ... <localdst>]
  [-count [-q] [-h] [-t [<storage type>]] <path> ...]
  [-cp [-f] [-p | -p[topax]] <src> ... <dst>]
  [-createSnapshot <snapshotDir> [<snapshotName>]]
  [-deleteSnapshot <snapshotDir> <snapshotName>]
  [-df [-h] [<path> ...]]
  [-du [-s] [-h] <path> ...]
  [-expunge]
  [-find <path> ... <expression> ...]
```

Conceitos e comandos básicos do HDFS

- HDFS fornece comandos para
 - Criar diretórios
 - Adicionar, mover ou excluir arquivos
 - Listar o conteúdo de diretórios
 - Formatar o sistema de arquivos
- Todos os comandos podem ser consultados usando:
 - `hadoop fs -help`



Conceitos e comandos básicos do HDFS

- Sintaxe dos comandos:
 - Similar aos comandos do Linux.
 - Iniciados por “hadoop dfs” ou “hdfs dfs”.
 - hadoop dfs -comando [argumentos].

Conceitos e Comandos Básicos do HDFS

- Alguns comandos úteis:
 - Formatação do HDFS: `bin/hdfs namenode -format`
 - Criação de diretórios: `hdfs dfs -mkdir arquivos_hadoop`
 - Inserir arquivos: `hdfs dfs -put meuarquivo.txt /user/hadoop_user`
 - Listar arquivos: `hdfs dfs -ls`
 - Remover arquivos: `hdfs dfs rm <path>`
 - Remover diretórios: `hdfs dfs rmr <path>`

```
[cris@crislinn|hadoop]$ hadoop
Usage: hadoop [--config confdir] COMMAND
where COMMAND is one of:
namenode -format          format the DFS filesystem
secondarynamenode
namenode
datanode
dradmin
dradmin
fsck
fs
balancer
jobtracker
pipes
tasktracker
job
queue
net.informations.johnbunes
```



Conceitos e comandos básicos do HDFS

```
% hadoop fs -mkdir books
% hadoop fs -ls .
Found 2 items
drwxr-xr-x    - tom supergroup          0 2009-04-02 22:41 /user/tom/books
-rw-r--r--    1 tom supergroup        118 2009-04-02 22:29 /user/tom/quangle.txt
```

- Informação retornada similar ao ls – l do Linux.
- Primeira coluna: permissões do arquivo ou diretório.
- Segunda coluna: fator de replicação.
- Terceira e quarta colunas: proprietário do arquivo (usuário e grupo).
- Quinta coluna: tamanho do arquivo.
- Sexta e sétima colunas: data e hora de última modificação.
- Última coluna: caminho e nome completo do arquivo ou diretório.

Conceitos e comandos básicos do HDFS

- Linhas de comando podem ser usadas sem muita dificuldade.
- Principalmente para conhecedores de Linux.
- Existem outras formas de acesso e manipulação.
- Providas pelo framework.

Conclusão

- Sistema de arquivos distribuídos.
- Sistema de arquivos distribuídos do Hadoop.
- Comandos úteis.
 - Linhas de comando.

■ Próxima aula

- Namenodes.
- Datanodes.

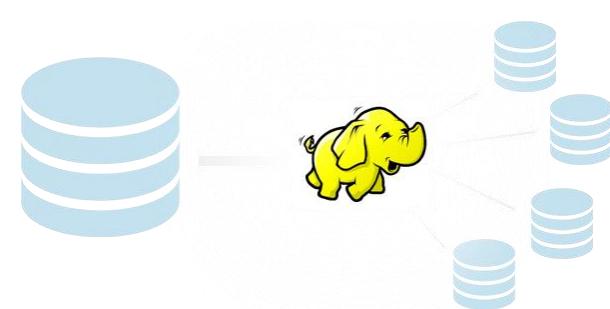


Aula 3.2. Namenodes e Datanodes

- Namenodes.
- Datanodes.

Namenodes e Datanodes

- Um cluster Hadoop possui dois tipos de nós:
 - Namenode.
 - Datanode.
- Realizam operações no padrão mestre-escravo:
 - Namenode (nó mestre) – geralmente 1. Algumas vezes 2.
 - Datanode (nó escravo) – número indefinido.



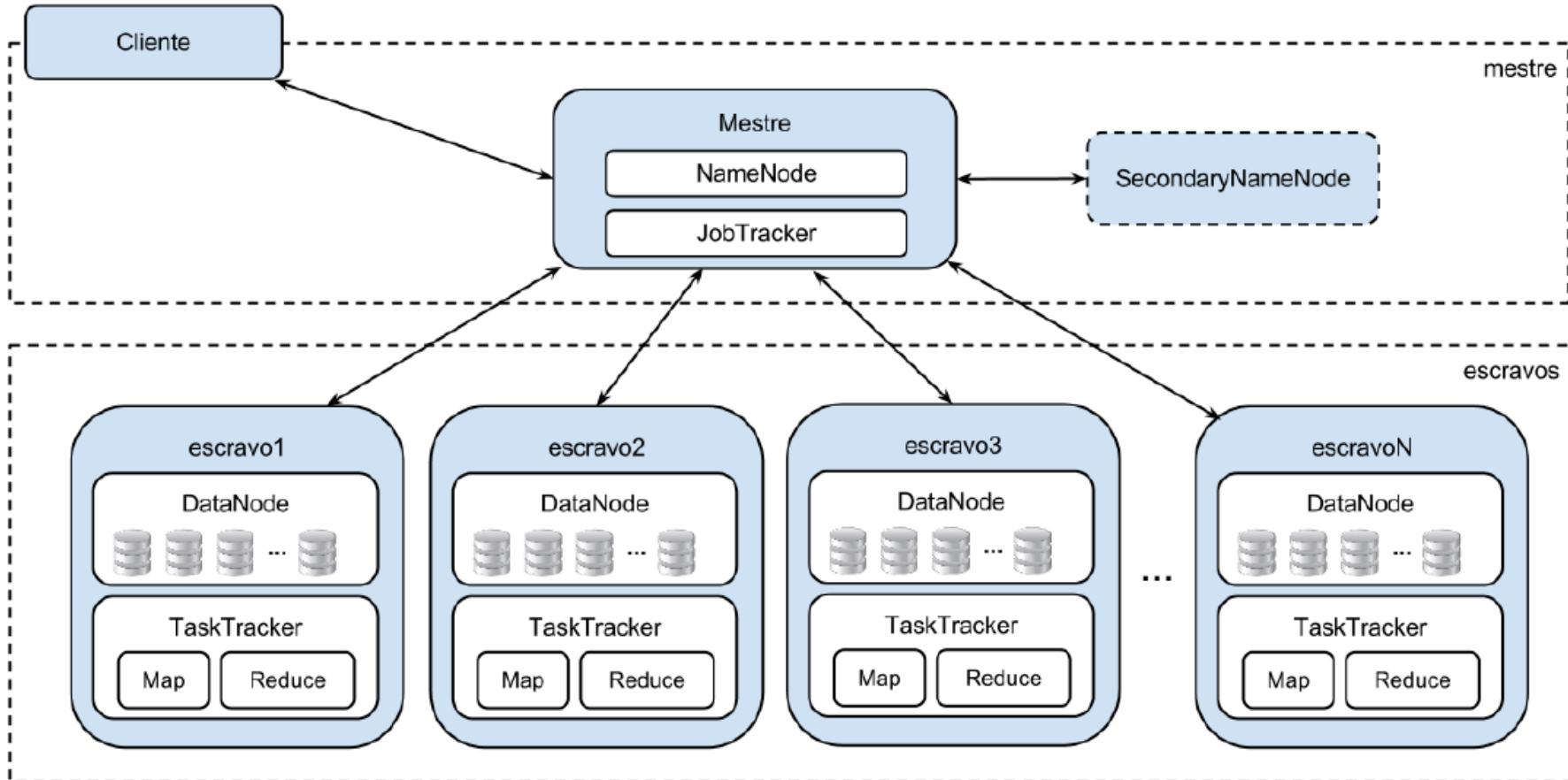
- O Namenode gerencia o sistema de arquivos (HDFS).
- Mantém a árvore de diretórios e arquivos.
- Possui acesso a todos os Datanodes do cluster, onde os blocos de dados serão enviados.
- Sem o Namenode não há como usar o sistema de arquivos.

- Funções:
 - Realizar a divisão dos arquivos em blocos.
 - Mapear a localização dos blocos de dados.
 - Encaminhar blocos aos nós escravos.
 - Fica localizado no nó-mestre da aplicação.

- Se o Namenode sofrer algum problema grave, todos os arquivos do HDFS serão perdidos.
- Caso as informações para sua reconstrução não esteja nos Datanodes.
- Dois tipos de recuperação:
 - Backup dos dados nos Datanodes.
 - Segundo Namenode.

- Os Datanodes são os “trabalhadores” do sistema.
- São instruídos pelo Namenode.
- Armazena e recuperam blocos de dados.
- Periodicamente enviam ao Namenode uma lista com os blocos de dados que estão armazenando.

Datanode



Conclusão

Namenode:

- Responsável pelo HDFS.
- Ligado ao nó mestre.
- Distribui as tarefas.

Datanode:

- Nó escravo.
- Executa os trabalhos.
- Armazena blocos de dados do HDFS.

■ Próxima aula

- Operações básicas do HDFS.



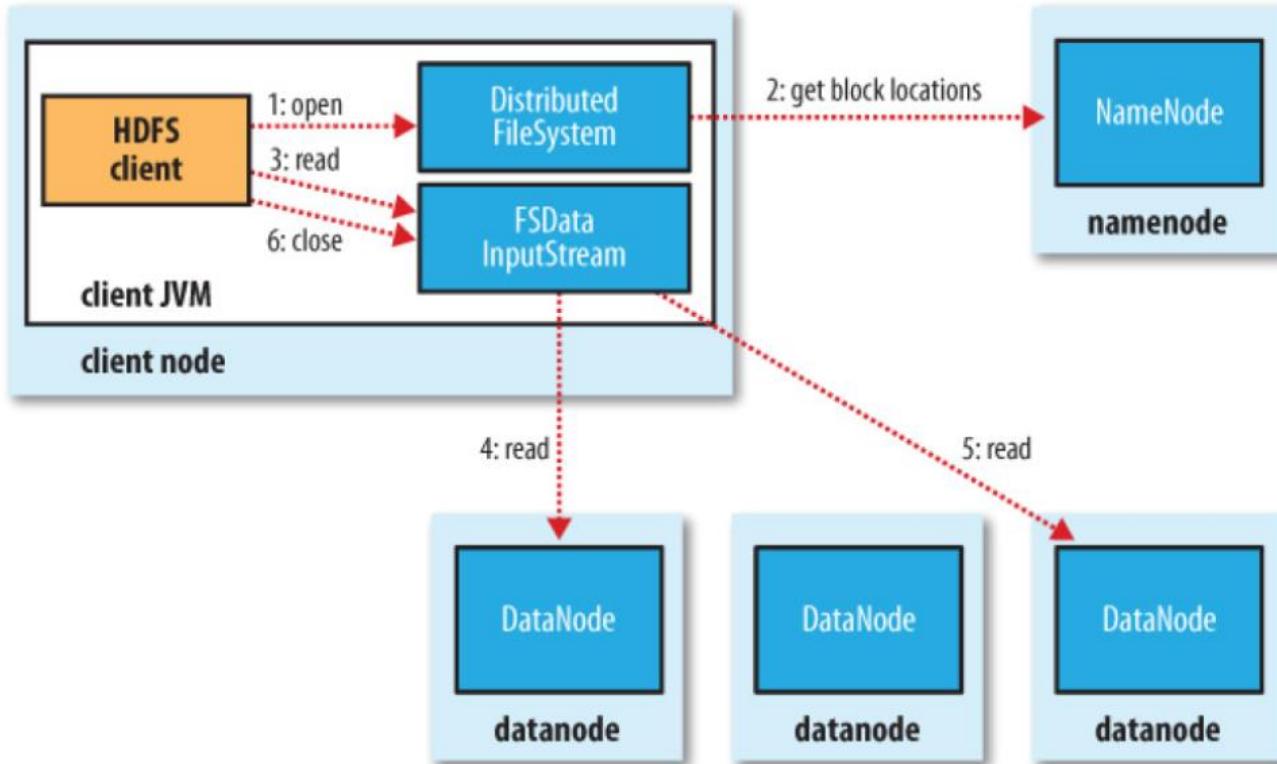
Aula 3.3. Operações básicas do HDFS

- ❑ Fluxo de dados entre nó-mestre e nó-escravo.
- ❑ Processo de leitura.
- ❑ Processo de escrita.

Operações Básicas do HDFS

- Processo de leitura e processo de escrita são efetuadas o tempo todo no HDFS.
- Grande troca de dados entre os nós mestre e escravos.
- Processo definido.
- Começaremos pelo processo de leitura.

Processo de Leitura



1 - O HDFS Client abre o arquivo desejado chamando a função `open()` no objeto `FileSystem`

1.1 – `FileSystem` é um objeto do tipo `DistributedFileSystem`

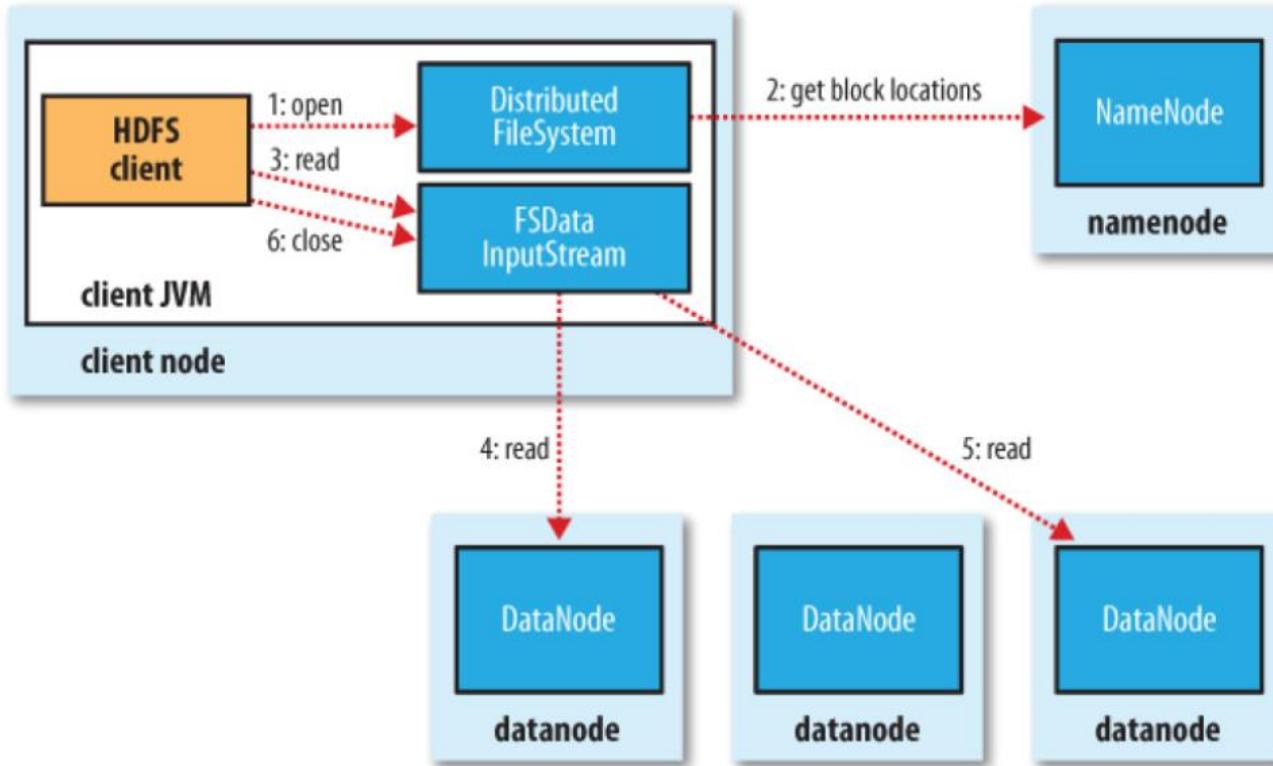
2 – `DistributedFileSystem` chama o `Namenode` e solicita informações sobre a localização dos arquivos

2.1 – O `Namenode` retorna os nomes dos `Datanodes` que possuem cópias do bloco

2.2 – Os `Datanodes` estão ordenados de acordo com sua proximidade com o cliente

3.3 – `DistributedFileSystem` retorna um `FSDataInputStream` para o HDFS Cliente para que ele possa ler os dados

Processo de Leitura



3 – O cliente HDFS chama a função Read() para a leitura dos dados

4 – O DFSInputStream, que possui armazenado os endereços dos Datanodes, conecta-se ao Datanode mais próximo

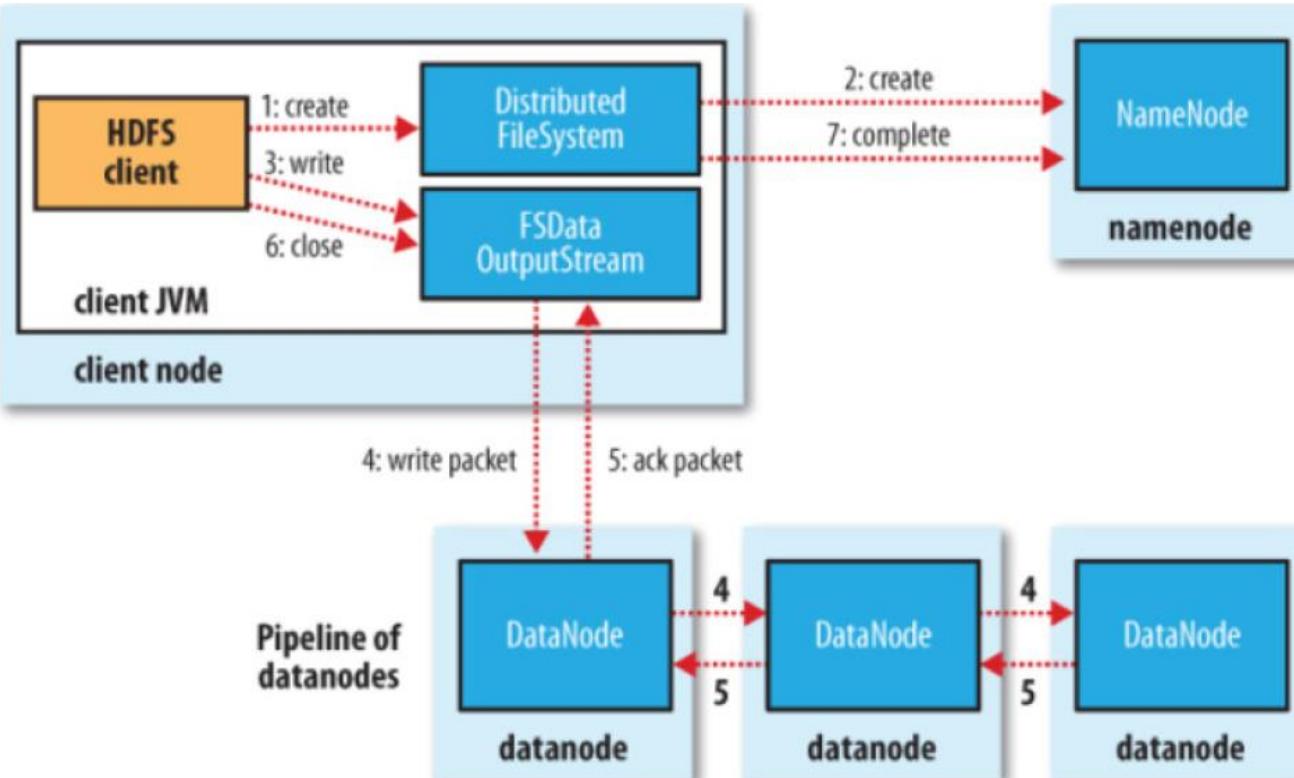
4.1 – Os dados são enviados do Datanode de volta ao cliente chamando repetidas vezes a função Read()

5 – Quando o fim do bloco é alcançado, DFSInputStream fechará a conexão com o Datanode

■ Operações Básicas do HDFS

- Em seguida temos o processo de escrita de arquivos no HDFS.

Processo de Escrita



1 – O HDFS Cliente solicita a criação do arquivo chamando a função Create() do DistributedFileSystem

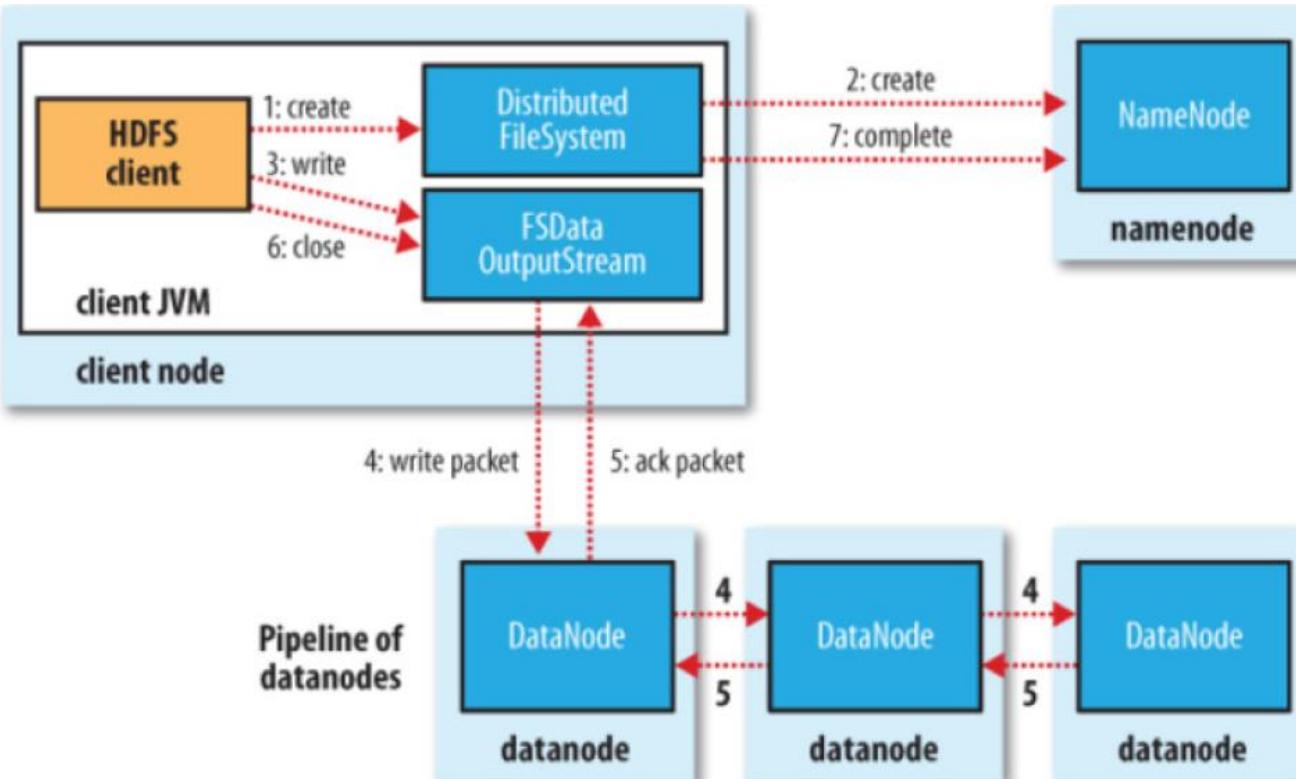
2 – DistributedFileSystem faz uma chamada ao Namenode para que o arquivo seja criado

2.1 – Namenode faz várias verificações para descobrir se o arquivo já existe e se existe permissão para a criação

2.3 – Se existir permissão o Namenode cria o arquivo, senão a criação falha e o cliente recebe uma exceção

2.4 – DistributedFileSystem retorna um FsDataOutputStream para o início da escrita

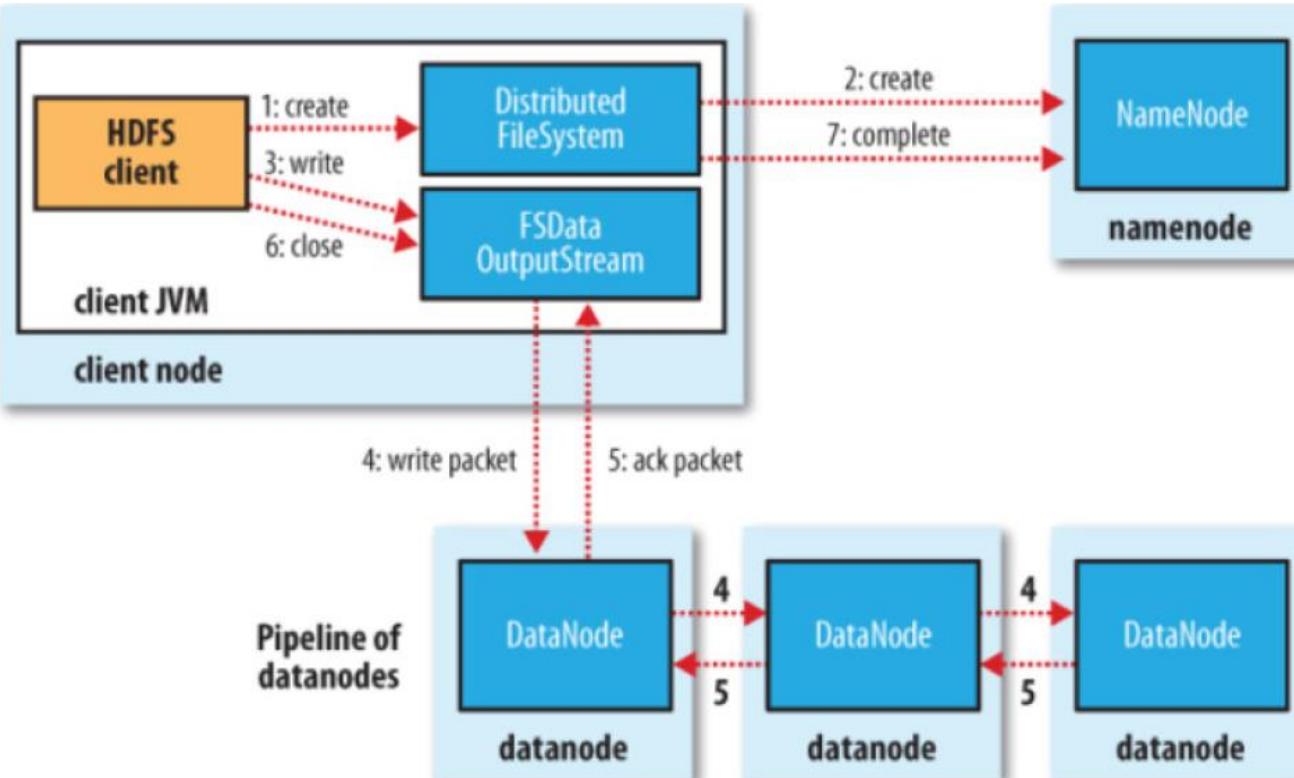
Processo de Escrita



3 – Cliente começa a escrever os dados e o `DFSOutputStream` particiona esses dados em uma fila interna (Data Queue)

3.1 – Objeto `DataStream` consome a fila, solicitando ao Namenode a alocação de novos blocos

Processo de Escrita

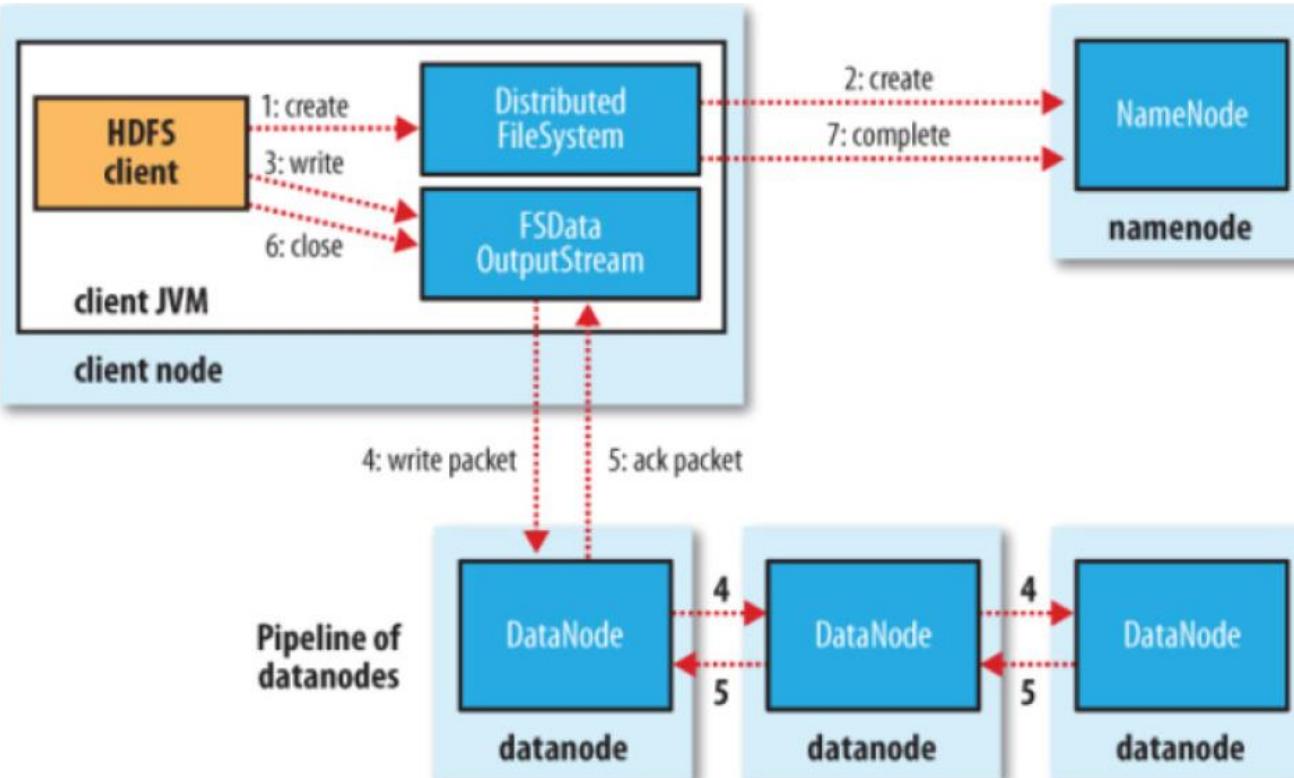


4 – DataStreamer escolhe em uma lista de Datanodes os mais adequados para armazenar as réplicas

4.1 – A lista de Datanodes forma um pipeline. Caso o número de réplicas seja 3, teremos 3 nós na pipeline

4.2 – DataStreamer divide os pacotes para o primeiro Datanode e esse Datanode repassa aos demais

Processo de Escrita



5 – O DFSOutputStream mantém uma fila interna de pacotes que está aguardando para serem conhecidos (Ack Queue)

5.1 – Um pacote é removido da Ack Queue apenas quando ele foi conhecido por todos os Datanodes

6 – Quando o cliente finaliza a escrita dos dados ele chama a função Close()

6.1 – Essa ação descarrega todos os pacotes restantes para a fila e aguarda que os Datanodes reconheçam

7 – Entra em contato com o Namenode e informa que a escrita do arquivo foi completa

Conclusão

Operações de Escrita.

Operações de Leitura.

■ Próxima aula

- Interfaces gráficas para acesso ao HDFS.



Aula 3.4. Interface gráfica para gerenciamento do HDFS

- Interfaces do Hadoop.
- Consultando o HDFS.
- Consultando os Jobs.

Interface gráfica para acesso ao HDFS

- Hadoop traz algumas interfaces gráficas.
- Utilizadas para gerenciamento dos serviços.
- Podem ser acessadas pelo *browser*.
- Todos os passos de um trabalho podem ser acompanhados pela interface gráfica.

Interface gráfica para acesso ao HDFS

- Falhas podem ser acompanhadas.
- Uma das interfaces pode ser acessada pela porta 9870.
- <http://localhost:9870>
- Fornece informações sobre o HDFS e sobre os Datanodes.

Interface gráfica para acesso ao HDFS

IGTI

Hadoop

Overview

Datanodes

Datanode Volume Failures

Snapshot

Startup Progress

Utilities ▾

Overview 'HadoopMaster:9000' (active)

Started:	Wed May 18 12:37:44 UTC 2016
Version:	2.7.1, r15ecc87ccf4a0228f35af08fc56de536e6ce657a
Compiled:	2015-06-29T06:04Z by jenkins from (detached from 15ecc87)
Cluster ID:	CID-ed310e79-b1a1-4c78-93b8-d35821a97577
Block Pool ID:	BP-652330868-10.54.8.12-1463575059827

Summary

Security is off.

Safemode is off.

102 files and directories, 86 blocks = 188 total filesystem object(s).

Heap Memory used 162.23 MB of 332.5 MB Heap Memory. Max Heap Memory is 889 MB.

Non Heap Memory used 61.08 MB of 63.27 MB Committed Non Heap Memory. Max Non Heap Memory is -1 B.

Interface gráfica para acesso ao HDFS

IGTI

Hadoop

Overview

Datanodes

Datanode Volume Failures

Snapshot

Startup Progress

Utilities ▾

Datanode Information

In operation

Node	Last contact	Admin State	Capacity	Used	Non DFS Used	Remaining	Blocks	Block pool used	Failed Volumes	Version
metrica1-7:50010 (10.54.8.18:50010)	0	In Service	393.6 GB	11.56 MB	20.07 GB	373.52 GB	29	11.56 MB (0%)	0	2.7.1
metrica1-4:50010 (10.54.8.13:50010)	0	In Service	393.6 GB	11.47 MB	20.07 GB	373.52 GB	37	11.47 MB (0%)	0	2.7.1
metrica1-8:50010 (10.54.8.16:50010)	0	In Service	393.6 GB	11.29 MB	20.07 GB	373.52 GB	26	11.29 MB (0%)	0	2.7.1
metrica1-2:50010 (10.54.8.17:50010)	0	In Service	393.6 GB	13.93 MB	20.07 GB	373.52 GB	28	13.93 MB (0%)	0	2.7.1
metrica1-5:50010 (10.54.8.15:50010)	0	In Service	393.6 GB	12.96 MB	20.07 GB	373.52 GB	36	12.96 MB (0%)	0	2.7.1
metrica1-3:50010 (10.54.8.14:50010)	2	In Service	393.6 GB	6.98 MB	20.07 GB	373.52 GB	24	6.98 MB (0%)	0	2.7.1
metrica1-6:50010 (10.54.8.19:50010)	2	In Service	393.6 GB	15.16 MB	20.07 GB	373.52 GB	32	15.16 MB (0%)	0	2.7.1
HadoopMaster:50010 (10.54.8.12:50010)	0	In Service	393.6 GB	18.9 MB	20.07 GB	373.51 GB	46	18.9 MB (0%)	0	2.7.1

[Hadoop](#)[Overview](#)[Datanodes](#)[Datanode Volume Failures](#)[Snapshot](#)[Startup Progress](#)[Utilities ▾](#)

Datanode Volume Failures

There are no reported volume failures.

Hadoop, 2015.

Interface gráfica para acesso ao HDFS

IGTI

The screenshot shows the Hadoop Web UI interface. At the top, there is a navigation bar with tabs: Hadoop (selected), Overview, Datanodes, Snapshot, Startup Progress, and Utilities (with a dropdown menu). The Utilities menu has two options: 'Browse the file system' (selected) and 'Logs'. Below the navigation bar, the title 'Browse Directory' is displayed. In the center, there is a table listing directory contents. The table has columns: Permission, Owner, Group, Size, Last Modified, Replication, Block Size, and Name. Two entries are listed:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwx-----	hduser	supergroup	0 B	18/05/2016 09:38:21	0	0 B	tmp
drwxr-xr-x	hduser	supergroup	0 B	18/05/2016 09:38:06	0	0 B	user

Interface gráfica para acesso ao HDFS

IGTI

Hadoop Overview Datanodes Snapshot Startup Progress Utilities ▾

Browse Directory

/user/hduser Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	hduser	supergroup	684.65 KB	18/05/2016 09:38:10	3	128 MB	Grafo_Internet
-rw-r--r--	hduser	supergroup	2.16 KB	18/05/2016 09:38:12	3	128 MB	Grafo_Internet200
-rw-r--r--	hduser	supergroup	532 B	18/05/2016 09:38:19	3	128 MB	Grafo_Internet50
-rw-r--r--	hduser	supergroup	15.33 KB	18/05/2016 09:38:14	3	128 MB	Grafo_Twitter
-rw-r--r--	hduser	supergroup	2.97 KB	18/05/2016 09:38:17	3	128 MB	Grafo_Twitter200
drwxr-xr-x	hduser	supergroup	0 B	18/05/2016 10:47:12	0	0 B	excentricidade
drwxr-xr-x	hduser	supergroup	0 B	18/05/2016 10:40:47	0	0 B	parada
drwxr-xr-x	hduser	supergroup	0 B	18/05/2016 10:47:31	0	0 B	resultado

Directory: /logs/

[SecurityAuth-hduser.audit](#)

[hadoop-hduser-datanode-HadoopMaster.log](#)

[hadoop-hduser-datanode-HadoopMaster.out](#)

[hadoop-hduser-namenode-HadoopMaster.log](#)

[hadoop-hduser-namenode-HadoopMaster.out](#)

[hadoop-hduser-secondarynamenode-HadoopMaster.log](#)

[hadoop-hduser-secondarynamenode-HadoopMaster.out](#)

[userlogs/](#)

[yarn-hduser-nodemanager-HadoopMaster.log](#)

[yarn-hduser-nodemanager-HadoopMaster.out](#)

[yarn-hduser-resourcemanager-HadoopMaster.log](#)

[yarn-hduser-resourcemanager-HadoopMaster.out](#)

0 bytes May 18, 2016 12:37:43 PM

295935 bytes May 18, 2016 3:29:20 PM

718 bytes May 18, 2016 12:37:48 PM

1199096 bytes May 18, 2016 3:37:20 PM

4960 bytes May 18, 2016 1:51:32 PM

26986 bytes May 18, 2016 2:38:57 PM

718 bytes May 18, 2016 12:37:54 PM

4096 bytes May 18, 2016 3:36:03 PM

557962 bytes May 18, 2016 1:47:39 PM

702 bytes May 18, 2016 12:38:02 PM

1574389 bytes May 18, 2016 1:47:39 PM

2086 bytes May 18, 2016 12:41:34 PM

Interface gráfica para acesso ao HDFS

```
2016-05-18 12:37:43,379 INFO org.apache.hadoop.hdfs.server.namenode.NameNode: STARTUP_MSG:  
*****  
STARTUP_MSG: Starting NameNode  
STARTUP_MSG: host = HadoopMaster/10.54.8.12  
STARTUP_MSG: args = []  
STARTUP_MSG: version = 2.7.1  
STARTUP_MSG: classpath = /usr/local/hadoop/etc/hadoop:/usr/local/hadoop/share/hadoop/common/lib/paranamer-2.3.jar:/usr/local/hadoop/share/hadoop/common/lib/mockito-all-1.8.5.jar:/usr/local/hadoop/share/hadoop/common/lib/httpcore-4.2.5.jar:/usr/local/hadoop/share/hadoop/common/lib/commons-beanutils-1.7.0.jar:/usr/local/hadoop/share/hadoop/common/lib/servlet-api-2.5.jar:/usr/local/hadoop/share/hadoop/common/lib/httpclient-4.2.5.jar:/usr/local/hadoop/share/hadoop/common/lib/api-asn1-api-1.0.0-M20.jar:/usr/local/hadoop/share/hadoop/common/lib/commons-logging-1.1.3.jar:/usr/local/hadoop/share/hadoop/common/lib/commons-configuration-1.6.jar:/usr/local/hadoop/share/hadoop/common/lib/stax-api-1.0-2.jar:/usr/local/hadoop/share/hadoop/common/lib/jettison-1.1.jar:/usr/local/hadoop/share/hadoop/common/lib/jackson-mapper-asl-1.9.13.jar:/usr/local/hadoop/share/hadoop/common/lib/jets3t-0.9.0.jar:/usr/local/hadoop/share/hadoop/common/lib/gson-2.2.4.jar:/usr/local/hadoop/share/hadoop/common/lib/jetty-util-6.1.26.jar:/usr/local/hadoop/share/hadoop/common/lib/api-util-1.0.0-M20.jar:/usr/local/hadoop/share/hadoop/common/lib/jersey-json-1.9.jar:/usr/local/hadoop/share/hadoop/common/lib/protobuf-java-2.5.0.jar:/usr/local/hadoop/share/hadoop/common/lib/jsp-api-2.1.jar:/usr/local/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar:/usr/local/hadoop/share/hadoop/common/lib/zookeeper-3.4.6.jar:/usr/local/hadoop/share/hadoop/common/lib/snappy-java-1.0.4.1.jar:/usr/local/hadoop/share/hadoop/common/lib/hadoop-auth-2.7.1.jar:/usr/local/hadoop/share/hadoop/common/lib/avro-1.7.4.jar:/usr/local/hadoop/share/hadoop/common/lib/apacheds-kerberos-codec-2.0.0-M15.jar:/usr/local/hadoop/share/hadoop/common/lib/commons-digester-1.8.jar:/usr/local/hadoop/share/hadoop/common/lib/jaxb-api-2.2.2.jar:/usr/local/hadoop/share/hadoop/common/lib/jersey-server-1.9.jar:/usr/local/hadoop/share/hadoop/common/lib/xmlenc-0.52.jar:/usr/local/hadoop/share/hadoop/common/lib/commons-cli-1.2.jar:/usr/local/hadoop/share/hadoop/common/lib/guava-11.0.2.jar:/usr/local/hadoop/share/hadoop/common/lib/junit-4.11.jar:/usr/local/hadoop/share/hadoop/common/lib/commons-lang-1.8.5.jar:/usr/local/hadoop/share/hadoop/common/lib/apachesnmp-2.0.0-M15.jar:/usr/local/hadoop/share/hadoop/common/lib/commons-lang-2.6.jar:/usr/local/hadoop/share/hadoop/common/lib/jsr305-3.0.0.jar:/usr/local/hadoop/share/hadoop/common/lib/jaxb-impl-2.2.3-1.jar:/usr/local/hadoop/share/hadoop/common/lib/jackson-jaxrs-1.9.13.jar:/usr/local/hadoop/share/hadoop/common/lib/curator-recipes-2.7.1.jar:/usr/local/hadoop/share/hadoop/common/lib/commons-compress-1.4.1.jar:/usr/local/hadoop/share/hadoop/common/lib/htrace-core-3.1.0-incubating.jar:/usr/local/hadoop/share/hadoop/common/lib/commons-collections-3.2.1.jar:/usr/local/hadoop/share/hadoop/common/lib/jersey-core-1.9.jar:/usr/local/hadoop/share/hadoop/common/lib/commons-net-3.1.jar:/usr/local/hadoop/share/hadoop/common/lib/hamcrest-core-1.3.jar:/usr/local/hadoop/share/hadoop/common/lib/jscsh-0.1.42.jar:/usr/local/hadoop/share/hadoop/common/lib/netty-3.6.2.Final.jar:/usr/local/hadoop/share/hadoop/common/lib/commons-beanutils-core-1.8.0.jar:/usr/local/hadoop/share/hadoop/common/lib/curator-framework-2.7.1.jar:/usr/local/hadoop/share/hadoop/common/lib/commons-math3-3.1.1.jar:/usr/local/hadoop/share/hadoop/common/lib/activity-1.1.jar:/usr/local/hadoop/share/hadoop/common/lib/java-xmlobuilder-0.4.jar:/usr/local/hadoop/share/hadoop/common/lib/commons-io-2.4.jar:/usr/local/hadoop/share/hadoop/common/lib/commons-codec-1.4.jar:/usr/local/hadoop/share/hadoop/common/lib/curator-client-2.7.1.jar:/usr/local/hadoop/share/hadoop/common/lib/xz-1.0.jar:/usr/local/hadoop/share/hadoop/common/lib/log4j-1.2.17.jar:/usr/local/hadoop/share/hadoop/common/lib/jetty-6.1.26.jar:/usr/local/hadoop/share/hadoop/common/lib/asm-3.2.jar:/usr/local/hadoop/share/hadoop/common/lib/jackson-asl-1.9.13.jar:/usr/local/hadoop/share/hadoop/common/lib/jackson-xc-1.9.13.jar:/usr/local/hadoop/share/hadoop/common/lib/hadoop-annotations-2.7.1.jar:/usr/local/hadoop/share/hadoop/common/lib/commons-httpclient-3.1.jar:/usr/local/hadoop/share/hadoop/common/hadoop-common-2.7.1.jar:/usr/local/hadoop/share/hadoop/nfs-2.7.1.jar:/usr/local/hadoop/share/hadoop/common/hadoop-common-2.7.1-tests.jar:/usr/local/hadoop/share/hadoop/hdfs/usr/local/hadoop/share/hadoop/hdfs/lib/xercesImpl-2.9.1.jar:/usr/local/hadoop/share/hadoop/hdfs/lib/servlet-api-2.5.jar:/usr/local/hadoop/share/hadoop/hdfs/lib/commons-logging-1.1.3.jar:/usr/local/hadoop/share/hadoop/hdfs/lib/jackson-mapper-asl-1.9.13.jar:/usr/local/hadoop/share/hadoop/hdfs/lib/jetty-util-6.1.26.jar:/usr/local/hadoop/share/hadoop/hdfs/lib/protobuf-java-2.5.0.jar:/usr/local/hadoop/share/hadoop/hdfs/lib/commons-daemon-1.0.13.jar:/usr/local/hadoop/share/hadoop/hdfs/lib/jersey-server-1.9.jar:/usr/local/hadoop/share/hadoop/hdfs/lib/xmlenc-0.52.jar:/usr/local/hadoop/share/hadoop/hdfs/lib/commons-cli-1.2.jar:/usr/local/hadoop/share/hadoop/hdfs/lib/guava-11.0.2.jar:/usr/local/hadoop/share/hadoop/hdfs/lib/commons-lang-2.6.jar:/usr/local/hadoop/share/hadoop/hdfs/lib/jsr305-3.0.0.jar:/usr/local/hadoop/share/hadoop/hdfs/lib/htrace-core-3.1.0-incubating.jar:/usr/local/hadoop/share/hadoop/hdfs/lib/jersey-core-1.9.jar:/usr/local/hadoop/share/hadoop/hdfs/lib/xml-apis-1.3.04.jar:/usr/local/hadoop/share/hadoop/hdfs/lib/netty-3.6.2.Final.jar:/usr/local/hadoop/share/hadoop/hdfs/lib/commons-io-2.4.jar:/usr/local/hadoop/share/hadoop/hdfs/lib/commons-codec-1.4.jar:/usr/local/hadoop/share/hadoop/hdfs/lib/log4j-1.2.17.jar:/usr/local/hadoop/share/hadoop/hdfs/lib/jetty-6.1.26.jar:/usr/local/hadoop/share/hadoop/hdfs/lib/leveldbjni-all-1.8.jar:/usr/local/hadoop/share/hadoop/hdfs/lib/netty-all-4.0.23.Final.jar:/usr/local/hadoop/share/hadoop/hdfs/lib/jackson-core-asl-1.9.13.jar:/usr/local/hadoop/share/hadoop/hdfs/lib/hadoop-hdfs-2.7.1-jar:/usr/local/hadoop/share/hadoop/hdfs/lib/hadoop-hdfs-2.7.1-tests.jar:/usr/local/hadoop/share/hadoop/hdfs/lib/hadoop-hdfs-nfs-2.7.1.Final.jar:/usr/local/hadoop/share/hadoop/yarn/lib/aopalliance-1.0.jar:/usr/local/hadoop/share/hadoop/yarn/lib/guice-3.0.jar:/usr/local/hadoop/share/hadoop/yarn/lib/servlet-api-2.5.jar:/usr/local/hadoop/share/hadoop/yarn/lib/commons-logging-1.1.3.jar:/usr/local/hadoop/share/hadoop/yarn/lib/stax-api-1.0-2.jar:/usr/local/hadoop/share/hadoop/yarn/lib/jettison-1.1.jar:/usr/local/hadoop/share/hadoop/yarn/lib/jackson-mapper-asl-1.9.13.jar:/usr/local/hadoop/share/hadoop/yarn/lib/jetty-util-6.1.26.jar:/usr/local/hadoop/share/hadoop/yarn/lib/jersey-json-1.9.jar:/usr/local/hadoop/share/hadoop/yarn/lib/protobuf-java-2.5.0.jar:/usr/local/hadoop/share/hadoop/yarn/lib/guice-servlet-3.0.jar:/usr/local/hadoop/share/hadoop/yarn/lib/zookeeper-3.4.6.jar:/usr/local/hadoop/share/hadoop/yarn/lib/jersey-client-1.9.jar:/usr/local/hadoop/share/hadoop/yarn/lib/commons-clr-1.2.jar:/usr/local/hadoop/share/hadoop/yarn/lib/guava-11.0.2.jar:/usr/local/hadoop/share/hadoop/yarn/lib/commons-lang-2.6.jar:/usr/local/hadoop/share/hadoop/yarn/lib/jsr305-3.0.0.jar:/usr/local/hadoop/share/hadoop/yarn/lib/jaxb-impl-2.2.3-1.jar:/usr/local/hadoop/share/hadoop/yarn/lib/jackson-jaxrs-1.9.13.jar:/usr/local/hadoop/share/hadoop/yarn/lib/javax.inject-1.jar:/usr/local/hadoop/share/hadoop/yarn/lib/commons-compress-1.4.1.jar:/usr/local/hadoop/share/hadoop/yarn/lib/commons-collections-3.2.1.jar:/usr/local/hadoop/share/hadoop/yarn/lib/jetty-activiti-
```

Interface gráfica para acesso ao HDFS

- Os Jobs podem ser acompanhados por outra interface.
- Acessada pela porta 8088.
- <http://localhost:8088>

Interface gráfica para acesso ao HDFS



All Applications

Logged in as: dr.who

Cluster
About
Nodes
Node Labels
Applications
NEW
NEW SAVING
SUBMITTED
ACCEPTED
RUNNING
FINISHED
FAILED
KILLED
Scheduler
Tools

Cluster Metrics																													
Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	Vcores Used	Vcores Total	Vcores Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes														
26	0	0	26	0	0 B	64 GB	0 B	0	64	0	8	0	0	0	0														
Scheduler Metrics																													
Scheduler Type																													
Capacity Scheduler				Scheduling Resource Type				Minimum Allocation				Maximum Allocation																	
<memory:1024, vCores:1>				<memory:8192, vCores:8>																									
Show 20 ▾ entries																													
Search:																													
ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress	Tracking UI	History																		
application_1463575081879_0026	hduser	Resultado	MAPREDUCE	default	Wed May 18 10:47:15 -0300 2016	Wed May 18 10:47:31 -0300 2016	FINISHED	SUCCEEDED			History																		
application_1463575081879_0025	hduser	Excentricidade	MAPREDUCE	default	Wed May 18 10:44:23 -0300 2016	Wed May 18 10:47:12 -0300 2016	FINISHED	SUCCEEDED			History																		
application_1463575081879_0024	hduser	AUX	MAPREDUCE	default	Wed May 18 10:40:49 -0300 2016	Wed May 18 10:44:21 -0300 2016	FINISHED	SUCCEEDED			History																		
application_1463575081879_0023	hduser	HEDA	MAPREDUCE	default	Wed May 18 10:38:04 -0300 2016	Wed May 18 10:40:48 -0300 2016	FINISHED	SUCCEEDED			History																		
application_1463575081879_0022	hduser	HEDA	MAPREDUCE	default	Wed May 18 10:34:29 -0300 2016	Wed May 18 10:38:02 -0300 2016	FINISHED	SUCCEEDED			History																		
application_1463575081879_0021	hduser	HEDA	MAPREDUCE	default	Wed May 18 10:31:47 -0300 2016	Wed May 18 10:34:27 -0300 2016	FINISHED	SUCCEEDED			History																		
application_1463575081879_0020	hduser	HEDA	MAPREDUCE	default	Wed May 18 10:29:05 -0300 2016	Wed May 18 10:31:45 -0300 2016	FINISHED	SUCCEEDED			History																		

Interface gráfica para acesso ao HDFS

IGTI



Logged in as: dr.who

Nodes of the cluster

Cluster	
About Nodes	
Node Labels	
Applications	
NEW	
NEW_SAVING	
SUBMITTED	
ACCEPTED	
RUNNING	
FINISHED	
FAILED	
KILLED	
Scheduler	
Tools	

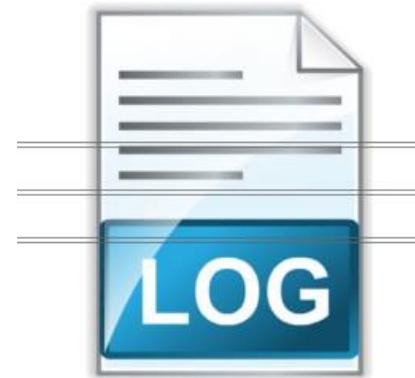
Cluster Metrics																
Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	Vcores Used	Vcores Total	Vcores Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes	
26	0	0	26	0	0 B	64 GB	0 B	0	64	0	8	0	0	0	0	0
Scheduler Metrics																
Scheduler Type				Scheduling Resource Type				Minimum Allocation				Maximum Allocation				
Capacity Scheduler				[MEMORY]				<memory:1024, vCores:1>				<memory:8192, vCores:8>				
Show 20 ▾ entries																
Node Labels	Rack	Node State	Node Address	Node HTTP Address	Last health-update	Health-report	Containers	Mem Used	Mem Avail	Vcores Used	Vcores Avail	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes	Version
/default-rack	RUNNING	metrica1-7:50706	metrica1-7:8042	metrica1-7:8042	Wed May 18 15:44:03 +0000 2016		0	0 B	8 GB	0	8	2.7.1				
/default-rack	RUNNING	metrica1-8:60808	metrica1-8:8042	metrica1-8:8042	Wed May 18 15:44:03 +0000 2016		0	0 B	8 GB	0	8	2.7.1				
/default-rack	RUNNING	HadoopMaster:59952	HadoopMaster:8042	HadoopMaster:8042	Wed May 18 15:44:03 +0000 2016		0	0 B	8 GB	0	8	2.7.1				
/default-rack	RUNNING	metrica1-5:36177	metrica1-5:8042	metrica1-5:8042	Wed May 18 15:44:03 +0000 2016		0	0 B	8 GB	0	8	2.7.1				
/default-rack	RUNNING	metrica1-6:39401	metrica1-6:8042	metrica1-6:8042	Wed May 18 15:44:03 +0000 2016		0	0 B	8 GB	0	8	2.7.1				
/default-rack	RUNNING	metrica1-2:33592	metrica1-2:8042	metrica1-2:8042	Wed May 18 15:44:03 +0000 2016		0	0 B	8 GB	0	8	2.7.1				
/default-rack	RUNNING	metrica1-3:42671	metrica1-3:8042	metrica1-3:8042	Wed May 18 15:44:03 +0000 2016		0	0 B	8 GB	0	8	2.7.1				
/default-rack	RUNNING	metrica1-4:33093	metrica1-4:8042	metrica1-4:8042	Wed May 18 15:44:03 +0000 2016		0	0 B	8 GB	0	8	2.7.1				

Showing 1 to 8 of 8 entries

First Previous 1 Next Last

Interface gráfica para acesso ao HDFS

- Além disso é possível consultar:
 - Configurações (XML)
 - Logs



Interface gráfica para acesso ao HDFS

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<configuration>
  ▼<property>
    <name>mapreduce.job.ubertask.enable</name>
    <value>false</value>
    <source>mapred-default.xml</source>
  </property>
  ▼<property>
    <name>yarn.resourcemanager.max-completed-applications</name>
    <value>10000</value>
    <source>yarn-default.xml</source>
  </property>
  ▼<property>
    <name>yarn.resourcemanager.delayed.delegation-token.removal-interval-ms</name>
    <value>30000</value>
    <source>yarn-default.xml</source>
  </property>
  ▼<property>
    <name>io.bytes.per.checksum</name>
    <value>512</value>
    <source>core-default.xml</source>
  </property>
  ▼<property>
    <name>yarn.timeline-service.leveldb-timeline-store.read-cache-size</name>
    <value>104857600</value>
    <source>yarn-default.xml</source>
  </property>
  ▼<property>
    <name>fs.s3a.connection.timeout</name>
    <value>50000</value>
    <source>core-default.xml</source>
  </property>
  ▼<property>
    <name>mapreduce.client.submit.file.replication</name>
    <value>10</value>
    <source>mapred-default.xml</source>
  </property>
  ▼<property>
```

Conclusão

- Interfaces Gráficas.
- HDFS.
- Jobs.
- Status do *cluster*.



Aula 3.5. Instalando a Máquina Virtual

- Instalando a Máquina Virtual.

Conclusão

Instalando a Máquina Virtual.

- ❑ Executando os comandos básicos do Ecossistema Hadoop e do HDFS.



Aula 3.7. Executando os comandos básicos do Ecossistema Hadoop e do HDFS

- ❑ Executando os comandos básicos do Ecossistema Hadoop e do HDFS.

- ✓ Executando os comandos básicos do Ecossistema Hadoop e do HDFS.

■ Próxima aula

- Capítulo 4.



Desenvolvimento de Soluções com MapReduce utilizando Hadoop

Capítulo 4. Criando Programas Apache Hadoop

Prof. João Paulo Barbosa Nascimento



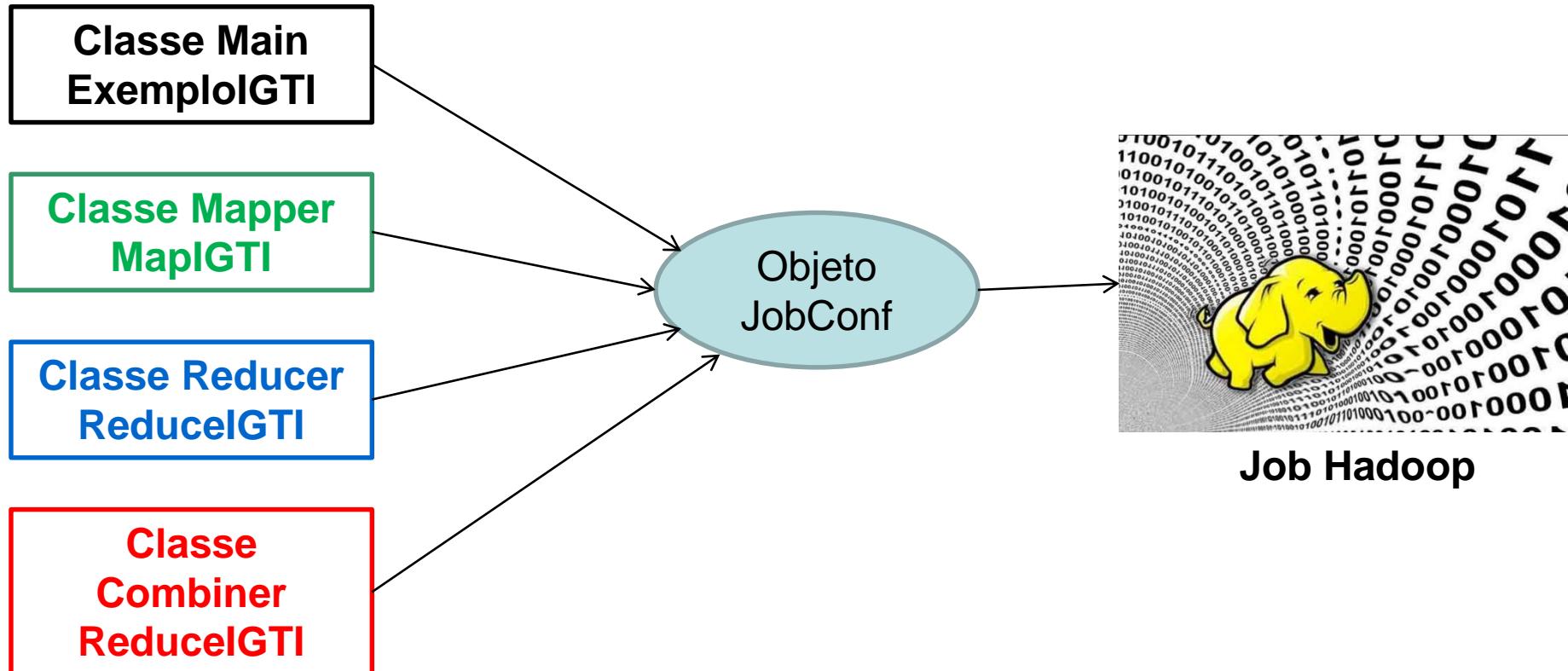
Aula 4.1. Estrutura de um programa Hadoop e a classe JobConf

- ❑ Estrutura do capítulo.
- ❑ Estruturando um programa no Apache Hadoop.
- ❑ Detalhes da classe JobConf.
- ❑ Exemplo de uso da classe JobConf.

Estruturando um programa no Hadoop

- Estrutura de um programa Java comum.
- Método main().
- Uma classe para o Mapper.
- Uma classe para o Reducer/Combiner.
- Um objeto para unir tudo.

Estrutura básica de um programa Hadoop



- A classe JobConf é a interface primária onde o usuário descreve um trabalho MapReduce (job).
- Para ser enviado ao framework Hadoop.
- Para que esse job possa ser realizado.
- O framework tenta executar o job fielmente ao que está definido em JobConf.

A classe JobConf

- Alguns parâmetros podem ser atribuídos utilizando a classe JobConf, outros não.
- Especifica o Mapper, Combiner (se houver), Reducer, etc.
- Pode ser usada para especificar outras características avançadas do Job.
- Visível dentro das funções Map e Reduce.

- Alguns métodos:
 - **setStrings**: enviar dados para as fases Map e Reduce.
 - **setNumTasks**: atribuir o número de tarefas Map para cada job.
 - **setNumReduceTasks**: atribuir o número de tarefas Reduce para cada job.
 - **setJobName**: atribuir o nome do job.

- Alguns métodos:
 - **setOutputKeyClass** ([Class](#)<?> theClass): atribui a classe para a saída dos dados do *job* (dados da chave – Key).
 - **setOutputValueClass**([Class](#)<?> theClass): atribui a classe para a saída dos dados do *job* (dados do valor - value).
 - **setMapperClass**: atribui uma classe *Mapper* para o *job* (minha classe).
 - **setReducerClass**: atribui uma classe *Reducer* para o *job* (minha classe).
 - **setCombinerClass**: atribui uma classe *Combiner* para o *job*.
 - Geralmente a mesma classe *Reducer*

A classe JobConf

- Alguns métodos:
 - setMaxMapAttempts(int n): atribui o número máximo de tentativas que serão feitas para executar uma tarefa Map.
 - setMaxReduceAttempts(int n): atribui o número máximo de tentativas que serão feitas para executar uma tarefa Reduce.

Exemplo de utilização da classe JobConf

```
1 package exemploigt;
2 import org.apache.hadoop.conf.*;
3 ...
4 public class ExemploIGTI extends Configured implements Tool {
5     public static void main (final String[] args) throws Exception {
6         int res = ToolRunner.run(new Configuration(), new ExemploIGTI, args);
7     }
8
9     public int run (final String[] args) throws Exception {
10        ...
11        JobConf conf = new JobConf(getConf(), ExemploIGTI.class);
12        conf.setJobName("Nome do Job");
13
14        conf.setOutputKeyClass(Text.class);
15        conf.setOutputValueClass(Text.class);
16
17        conf.setMapperClass(MapIGTI.class);
18        conf.setReducerClass(ReduceIGTI.class);
19        conf.setCombinerClass(ReduceIGTI.class);
20        ...
21    }
22 }
```

- Estrutura básica de um programa Hadoop.
- Estrutura Java.
- Classe principal, Mapper, Reducer e Combiner.
- Objeto JobConf.

■ Próxima aula

- A classe MapReduceBase.
- Definições e utilidade.



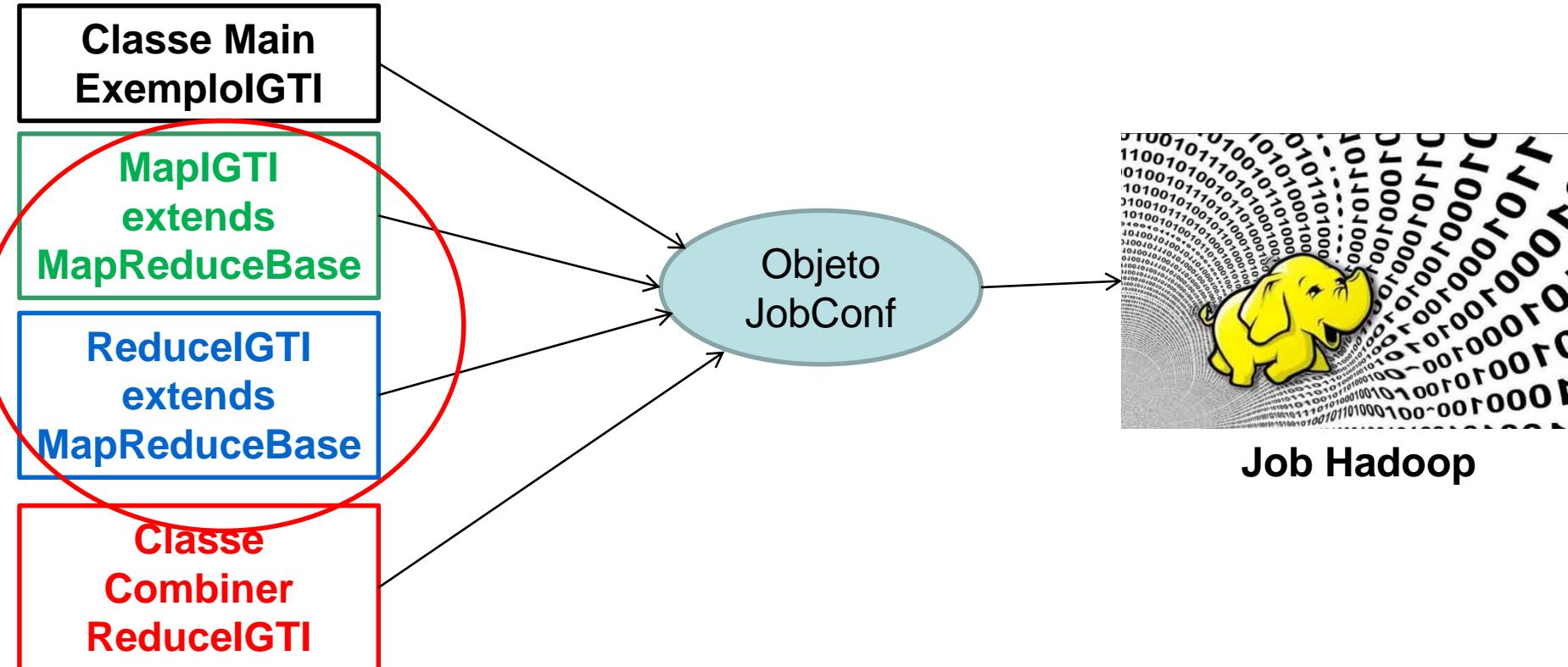
Aula 4.2. A classe MapReduceBase

- ❑ Descrição da classe MapReduceBase.
- ❑ Exemplo de uso.

A classe MapReduceBase

- É a classe base para implementação de Mappers e Reducers.
- Fica em org.apache.hadoop.mapred.MapReduceBase.
- Métodos principais:
 - Close.
 - Configure.

A classe MapReduceBase



Exemplo de utilização da classe MapReduceBase

```
1 package exemploigti;
2 import org.apache.hadoop.conf.*;
3 ...
4 public class ExemploIGTI extends Configured implements Tool {
5     public static void main (final String[] args) throws Exception {
6         int res = ToolRunner.run(new Configuration(), new ExemploIGTI, args);
7     }
8
9     public int run (final String[] args) throws Exception {
10        ...
11        JobConf conf = new JobConf(getConf(), ExemploIGTI.class);
12        conf.setJobName("Nome do Job");
13
14        conf.setOutputKeyClass(Text.class);
15        conf.setOutputValueClass(Text.class);
16
17        conf.setMapperClass(MapIGTI.class);
18        conf.setReducerClass(ReduceIGTI.class);
19        conf.setCombinerClass(ReduceIGTI.class);
20        ...
21    }
22 }
```

Exemplo de utilização da classe MapReduceBase - Mapper

```
27 ...
28 public static class MapIGTI extends MapReduceBase implements Mapper<LongWritable, Text, Text, Text> {
29     public void configure(JobConf job) {
30         ... //sua implementação
31     }
32
33     public void map(LongWritable key, Text value, OutputCollector<Text, Text> output, Reporter reporter)
34     throws IOException {
35         ... //sua implementação
36     }
37
38     public void close() {
39         ... //sua implementação
40     }
41 }
```

A classe MapReduceBase

- Mapper possui o método Map.
- Mapeia o key/value de entrada em key/value intermediário.
- Um key/value pode gerar mais de um key/value intermediário
 - Key: a chave de entrada;
 - Value: o valor de entrada;
 - Output: coleta as chaves e valores mapeados;
 - Reporter: facilita a tarefa de reportar o progresso do job (logs).

Exemplo de utilização da classe MapReduceBase - Reducer

```
44 public static class ReduceIGTI extends MapReduceBase implements Reducer<Text, Text, Text, Text>
45     public void configure (JobConf job) {
46         ... //sua implementação
47     }
48
49     public void reduce (Text key, Iterator<Text> values, OutputCollector<Text, Text> output,
50 Reporter reporter) throws IOException {
51         ... //sua implementação
52     }
53
54     public void close () {
55         ... //sua implementação
56     }
57 }
```

A classe MapReduceBase

- Interface Reducer possui o método Reduce.
- Reduz um conjunto de valores intermediários que possuem uma chave e um conjunto de valores.
- Fases primárias:
 - Shuffle:
 - Sort
- Parâmetros:
 - Key: a chave;
 - Values: a lista de valores intermediários para reduzir;
 - Output e Reporter.

Classe MapReduceBase.

Interfaces:

- Mapper;

- Reducer.

- Principais comandos de manipulação de diretórios no HDFS via aplicação.



Aula 4.3. Criando, consultando e excluindo diretórios no HDFS (via aplicação)

- Criando, consultando e excluindo diretórios no HDFS (via aplicação).

Criando e excluindo diretórios no HDFS

- O Hadoop permite manipular o HDFS via aplicação.
- Para isso utilizamos a classe *FileSystem*.
- Permite manipular sistemas de arquivos distribuídos ou local.
- Possui dezenas de métodos para manipulação do HDFS.
- Para saber mais (documentação da classe):
<https://hadoop.apache.org/docs/r2.7.1/api/index.html?org/apache/hadoop/fs/FileSystem.html>

Criando e excluindo diretórios no HDFS

- FileSystem.
- Fornece diversos métodos para manipulação do HDFS:
 - exists(Path f): verifica se um determinado caminho existe.
 - getHomeDirectory(): retorna o diretório raiz (home) no HDFS.
 - isDirectory (Path p): verifica se o caminho p é um diretório.
 - isFile(Path f): verifica se p é um arquivo.
 - mkdirs(Path p): permite a criação de um diretório.
 - delete(Path p): permite a exclusão de um diretório.

Criando e excluindo diretórios no HDFS

```
package IGTI;

import java.io.*;
import java.util.*;
import java.util.Random;
import java.text.*;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.fs.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.util.*;

public class ComandosHDFS extends Configured implements Tool {

    public static void main (final String[] args) throws Exception {
        int res = ToolRunner.run(new Configuration(), new ComandosHDFS(), args);
        System.exit(res);
    }

    public int run (final String[] args) throws Exception {
        try{
            JobConf conf = new JobConf(getConf(), ComandosHDFS.class);
            conf.setJobName("Ola Mundo");

            final FileSystem fs = FileSystem.get(conf);

            Path p = new Path("IGTI"); /* caminho no HDFS: /user/hduser/IGTI */

            if (fs.exists(p)) { /* verifica se o caminho p existe */
                System.out.println("Diretorio IGTI encontrado");
                fs.delete(p); /* como o diretório já existe, vamos exclui-lo */
            }
            else {
                System.out.println("Diretorio inexistente");
                fs.mkdirs(p); /* como o diretório não existe, vamos cria-lo */
            }
        catch ( Exception e ) {
            throw e;
        }
        return 0;
    }
}
```

- O resultados dos métodos podem ser consultados na ferramenta de gerenciamento do Hadoop.
- <http://localhost:9870>
- Ou via linha de comando.

Criando e excluindo diretórios no HDFS

Hadoop Overview Datanodes Snapshot Startup Progress Utilities ▾

Browse Directory

/user/hduser

Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	hduser	supergroup	0 B	19/03/2017 22:52:00	0	0 B	IGTI

Hadoop, 2016.

Conclusão

- Criando, consultando e excluindo diretórios no HDFS.
- Via aplicação.
- Importância para aplicações dinâmicas.

- Lendo e gravando dados no HDFS via aplicação.
- Atribuindo os diretórios padrão de entrada e saída de um job MapReduce.



Aula 4.4. Lendo e gravando dados no HDFS (via aplicação)

- Lendo e gravando dados no HDFS via aplicação.
- Atribuindo os diretórios padrão de entrada e saída de um job MapReduce.

Lendo e gravando dados no HDFS (via aplicação)



- Iteração com o HDFS é necessária durante a execução do job.
- Ler e gravar arquivos manipular diretórios.
- Classe FileSystem fornece vários métodos para manipulações de arquivos.
- FileInputFormat e FileOutputFormat: classes utilizadas para manipular arquivos atribuídos às funções Map e Reduce.

Lendo e gravando dados no HDFS (via aplicação)

IGTI

- **FileInputFormat:**
 - **static setInputPaths(JobConf conf, Path inputPaths):** atribui caminhos como entrada para o job MapReduce
 - **static setInputPaths(JobConf conf, String commaSeparatedPaths):** lista de entradas para o job MapReduce
 - **static getInputPaths(JobConf conf):** retorna uma lista de Paths para o job MapReduce

- **FileOutputFormat:**
 - **static setOutputPath(JobConf conf, Path outputDir):** atribui um caminho para o diretório de saída do *job* MapReduce.
 - Diretório onde os dados serão gravados.
 - **setCompressOutput(JobConf conf, boolean compress):** define se os dados de saída do *job* serão compactados.

Lendo e gravando dados no HDFS (via aplicação)



```
import java.io.*;
import java.util.*;
import java.util.Random;
import java.text.*;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.fs.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.util.*;

public class ExemploIGTI extends Configured implements Tool {
    public static void main (final String[] args) throws Exception {
        int res = ToolRunner.run(new Configuration(), new ExemploIGTI(), args);
        System.exit(res);
    }

    public int run (final String[] args) throws Exception {
        try{
            JobConf conf = new JobConf(getConf(), ExemploIGTI.class);
            conf.setJobName("Ola Mundo");

            String caminhoEntrada = "entrada";
            FileInputFormat.setInputPaths(conf, caminhoEntrada);
            FileOutputFormat.setOutputPath(conf, new Path("saída"));

            conf.setOutputKeyClass(Text.class);
            conf.setOutputValueClass(Text.class);
            conf.setMapperClass(MapIGTI.class);
            conf.setReducerClass(ReduceIGTI.class);
            JobClient.runJob(conf);
        }
        catch ( Exception e ) {
            throw e;
        }
        return 0;
    }
}
```

Conclusão

- ✓ Atribuindo entradas e saídas para o job.
- ✓ FileInputFormat.
- ✓ FileOutputFormat.

■ Próxima aula

- Um “Olá mundo” com Hadoop.
- Aplicar todos os conceitos vistos até agora.



Aula 4.5. Um “Olá mundo!” utilizando o Hadoop

- Um programa Hadoop utilizando todos os conceitos apresentados até aqui.

Um “Olá mundo!” utilizando o Hadoop

- Etapas de execução do programa:
 - Configurar um objeto JobConf;
 - Criar um diretório de entrada com a classe FileSystem;
 - Incluir um arquivo para processamento (do sistema de arquivos do Linux para o HDFS), com o método copyFromLocalFile;
 - Atribuir um diretório de entrada para o Job;
 - Atribuir um diretório de saída para o Job (output);
 - Atribuir as classes Mapper e Reducer;
 - Apresentar a implementação dos métodos Map e Reduce;
 - Executar o Job;
 - Apresentar e analisar os resultados.

Um “Olá mundo!” utilizando o Hadoop

- Etapas de execução do programa:
 - Configurar um objeto JobConf.

```
package IGTI;

import java.io.*;
import java.util.*;
import java.util.Random;
import java.text.*;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.fs.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.util.*;

public class ExemploIGTI extends Configured implements Tool
{
    public static void main (final String[] args) throws Exception {
        int res = ToolRunner.run(new Configuration(), new ExemploIGTI(), args);
        System.exit(res);
    }

    public int run (final String[] args) throws Exception {
        try{
            JobConf conf = new JobConf(getConf(), ExemploIGTI.class);
            conf.setJobName("Olá Mundo");
        }
    }
}
```

Um “Olá mundo!” utilizando o Hadoop

- Etapas de execução do programa:
 - Criar um diretório de entrada com a classe FileSystem.
 - Copiar um arquivo para dentro desse diretório.

```
public int run (final String[] args) throws Exception {
    try{
        JobConf conf = new JobConf(getConf(), ExemploIGTI.class);
        conf.setJobName("Olá Mundo");

        final FileSystem fs = FileSystem.get(conf);
        Path diretorioEntrada = new Path("Entrada"), diretorioSaida = new Path("Saida");

        /* Criar um diretório de entrada no HDFS */
        if (!fs.exists(diretorioEntrada))
            fs.mkdirs(diretorioEntrada);

        /* Adicionar um arquivo para ser processado */
        fs.copyFromLocalFile(new Path("/usr/local/hadoop/Dados/arquivoBigData.txt"), diretorioEntrada);
```

Um “Olá mundo!” utilizando o Hadoop

- Etapas de execução do programa:
 - Arquivo a ser processado:

```
fsdffsrrrewrejwrwerkljkj3423423k4j23kl4j23k4ljk23l4j001 fsfdssdfsdfsdfs00020090 fsdfsdffd...  
44j23lkj4k23lj4k23lj4k23lj4kl23j4kl23j4k23lj4kl32j4kl2j4ui007 fsdfsda...00010020 dsfsdfdsf...  
4i234j32oi4jio23j4io32j4i23o...001 erwerewrewrew4300067800 dsfsdfsd...0004343 dsfsdf...  
4i3o2p4iop23i4o23p4io23p4io23pi4o23i4o2p3i423poi4o23pi423p009 fsfsdfsdf...00004578 sda...  
o4pi23po4i23op4ik23o4ik23ç4k23l4kl23çk4lç32k4ç23lk4lç23k4l001 asdfsdf...00004578 sda...  
h4jl32h4j23h4jkh234jkh234kj...009 fsfsdfsdf...00003333 dsfd...00033210 dsfd...  
432423423432k4jk32lj4kl23j4kl2j34lkj234lkj234lkj34kl32j4kl23j44009 fsfsdfsdf...00033210 dsfd...  
jk32j423k14j23lk4jk23lj432kj4kl23j4k3l2j43lk2jj23kl007 fsdafsdf...00034567 sdfsdf...  
23432423432423423k4j23klj423lk4jk2l34432ftf4ty32ft32t4f34009 fsfsdfsdf...00043321 sdfsdf...  
86887686786kjk...001 fsdfsaf...00034567 sdfsdf...00034567 sdfsdf...  
423i4ui23ou4i32u4io32u4io3u4io3u4iou4io23u4io32u4io32u4oi23u4i001 fsfsfsdf...00000234 newrew...  
k4k34j23lkj4k32lj4kl23j4kl23j4kl32j4kl32j4kl32j4kl32j4kl32001 fsfsdfsdf...00003259 wererew...  
j432j4kl32j4kl32j4kl23j432kj432lkj432lkj4k23l4j32lkj4k32l4007 fsdafs...00092345 re...  
l4j32lkj432kl4j3k2l4j23kl4j3k2l4j32lkj4kl234j3kl2j44004 vvc...00000211 ewr...  
j4ç3k2k423çl4kl23k4l32k4lç23k4lç23k4l32çk4lç32k4l3ç2k43444008 vvc...00002235 fqqqqqqqqqqqqqqq  
k43k4lç32k4l23çk4lç23k4lç23k42ioriewopriweopriweopirv004 jfdgfdgdf...00002113 imm...  
4jop34k23o4io234io32p4i32po4i32po4i3poi432poi432poi432po4004 jdfgdf...00022119 dfdfhjkkkkkkkk  
4ç324324k23çl4k3l2k4lç2k4lç32k4lç23k4lç32k4lç32k4l3çk4ç4kl004 jfdgrtqqqqqqqqqc00002135 rtr...  
4kç23k4lç23k4lç32k4l4k23çlk423çlk4l23çlk432çlk44k333001 newewqewqewqewv00000123 tytyt...  
43123210909109109209330123k2j32k1çl3k21çlk3l21k3l222007 popopopopo...00012999 iuiuiuiuaaaaaaa
```

Um “Olá mundo!” utilizando o Hadoop

- Etapas de execução do programa:
 - Atribuir diretórios de entrada e saída:

```
public int run (final String[] args) throws Exception {  
    try{  
        JobConf conf = new JobConf(getConf(), ExemploIGTI.class);  
        conf.setJobName("Olá Mundo");  
  
        final FileSystem fs = FileSystem.get(conf);  
        Path diretorioEntrada = new Path("Entrada"), diretorioSaida = new Path("Saída");  
  
        /* Criar um diretório de entrada no HDFS */  
        if (!fs.exists(diretorioEntrada))  
            fs.mkdirs(diretorioEntrada);  
  
        /* Adicionar um arquivo para ser processado */  
        fs.copyFromLocalFile(new Path("/usr/local/hadoop/Dados/arquivoBigData.txt"), diretorioEntrada);  
  
        /* Atribuindo os diretórios de Entrada e Saída para o Job */  
        FileInputFormat.setInputPaths(conf, diretorioEntrada);  
        FileOutputFormat.setOutputPath(conf, diretorioSaida);  
    }  
}
```



Um “Olá mundo!” utilizando o Hadoop

- Etapas de execução do programa:
 - Atribuir as classes Mapper e Reducer:

```
conf.setOutputKeyClass(Text.class);
conf.setOutputValueClass(Text.class);
conf.setMapperClass(MapIGTI.class);
conf.setReducerClass(ReduceIGTI.class);
JobClient.runJob(conf);
```

Um “Olá mundo!” utilizando o Hadoop

- Etapas de execução do programa:
 - Implementação do método Map:

```
public static class MapIGTI extends MapReduceBase implements Mapper<LongWritable, Text, Text, Text> {  
    public void map(LongWritable key, Text value, OutputCollector<Text, Text> output, Reporter reporter) throws IOException {  
        Text txtChave = new Text();  
        Text txtValor = new Text();  
  
        String codigoCliente = value.toString().substring(58, 61);  
        String qtdeItens = value.toString().substring(76, 84);  
  
        txtChave.set(codigoCliente);  
        txtValor.set(qtdeItens);  
  
        output.collect(txtChave, txtValor);  
    }  
}
```

Um “Olá mundo!” utilizando o Hadoop

- Etapas de execução do programa:
 - Implementação do método Reduce:

```
public static class ReduceIGTI extends MapReduceBase implements Reducer<Text, Text, Text, Text> {  
    public void reduce (Text key, Iterator<Text> values, OutputCollector<Text, Text> output, Reporter reporter) throws IOException {  
  
        double media = 0.0;  
        int acumuladorItens = 0, contaVendas = 0;  
        Text value = new Text();  
  
        while (values.hasNext()) {  
            value = values.next();  
            contaVendas++;  
            acumuladorItens += Integer.parseInt(value.toString());  
        }  
  
        media = acumuladorItens / new Double(contaVendas);  
        value.set(String.valueOf(media));  
        output.collect(key, value);  
    }  
}
```

Um “Olá mundo!” utilizando o Hadoop

- Etapas de execução do programa:

- Execução do Job:

```
17/03/23 19:35:57 INFO mapreduce.JobSubmitter: number of splits:2
17/03/23 19:35:57 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1490307791670_0001
17/03/23 19:35:58 INFO impl.YarnClientImpl: Submitted application application_1490307791670_0001
17/03/23 19:35:58 INFO mapreduce.Job: The url to track the job: http://JOAOLINUX:8088/proxy/application_1490307791670_0001/
17/03/23 19:35:58 INFO mapreduce.Job: Running job: job_1490307791670_0001
17/03/23 19:36:06 INFO mapreduce.Job: Job job_1490307791670_0001 running in uber mode : false
17/03/23 19:36:06 INFO mapreduce.Job: map 0% reduce 0%
17/03/23 19:36:14 INFO mapreduce.Job: map 100% reduce 0%
17/03/23 19:36:30 INFO mapreduce.Job: map 100% reduce 100%
17/03/23 19:36:30 INFO mapreduce.Job: Job job_1490307791670_0001 completed successfully
17/03/23 19:36:30 INFO mapreduce.Job: Counters: 49
    File System Counters
        FILE: Number of bytes read=306
        FILE: Number of bytes written=355853
        FILE: Number of read operations=0
        FILE: Number of large read operations=0
        FILE: Number of write operations=0
        HDFS: Number of bytes read=3310
        HDFS: Number of bytes written=71
        HDFS: Number of read operations=9
        HDFS: Number of large read operations=0
        HDFS: Number of write operations=2
    Job Counters
        Launched map tasks=2
        Launched reduce tasks=1
        Data-local map tasks=2
        Total time spent by all maps in occupied slots (ms)=9793
        Total time spent by all reduces in occupied slots (ms)=12540
        Total time spent by all map tasks (ms)=9793
        Total time spent by all reduce tasks (ms)=12540
        Total vcore-milliseconds taken by all map tasks=9793
        Total vcore-milliseconds taken by all reduce tasks=12540
        Total megabyte-milliseconds taken by all map tasks=10028032
        Total megabyte-milliseconds taken by all reduce tasks=12840960
```

Um “Olá mundo!” utilizando o Hadoop

- Etapas de execução do programa:
 - Analisando os resultados (ferramentas de gerenciamento do Hadoop):

localhost:8088/cluster

90% Pesquisar

All Applications

hadoop

Cluster Metrics														
	Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	Vcores Used	Vcores Total	Vcores Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes
About Nodes	1	0	0	1	0	0 B	8 GB	0 B	0	8	0	1	0	0
Node Labels Applications	NEW	NEW	SAVING	SUBMITTED	ACCEPTED	RUNNING	FINISHED	FAILED	KILLED					

Scheduler Metrics		Scheduler Type		Scheduling Resource Type		Minimum Allocation		Maximum Allocation	
Capacity Scheduler		[MEMORY]				<memory:1024, vCores:1>			<memory:8192, vCores:8>

Show 20 entries															Search:
ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress	Tracking UI					
application_1490307791670_0001	hduser	Ola Mundo	MAPREDUCE	default	Thu Mar 23 19:35:58 -0300 2017	Thu Mar 23 19:36:28 -0300 2017	FINISHED	SUCCEEDED							

Showing 1 to 1 of 1 entries

Um “Olá mundo!” utilizando o Hadoop

- Etapas de execução do programa:
 - Analisando os resultados (HDFS – localhost:9870):

The screenshot shows the HDFS Web Interface. The top navigation bar has tabs: Hadoop, Overview, Datanodes, Snapshot, Startup Progress, Utilities (selected), and a dropdown menu with options: Browse the file system and Logs. Below the navigation is a search bar with the path /user/hduser and a Go! button. The main content area is titled 'Browse Directory' and displays a table of files in the /user/hduser directory. The table columns are: Permission, Owner, Group, Size, Last Modified, Replication, Block Size, and Name. There are two entries:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	hduser	supergroup	0 B	23/03/2017 19:35:54	0	0 B	Entrada
drwxr-xr-x	hduser	supergroup	0 B	23/03/2017 19:36:28	0	0 B	Saida

At the bottom left, it says 'Hadoop, 2016.'

Um “Olá mundo!” utilizando o Hadoop

- Etapas de execução do programa:
 - Analisando os resultados (HDFS – Conteúdo do diretório entrada):

The screenshot shows the Hadoop Web UI interface. At the top, there is a navigation bar with tabs: Hadoop, Overview, Datanodes, Snapshot, Startup Progress, Utilities (selected), and a dropdown menu. The Utilities dropdown is open, showing options: 'Browse the file system' and 'Logs'. Below the navigation bar, the main content area has a title 'Browse Directory' and a path input field containing '/user/hduser/Entrada'. To the right of the path input is a 'Go!' button. A table below lists the contents of the directory:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	hduser	supergroup	2.01 KB	23/03/2017 19:35:55	1	128 MB	arquivoBigData.txt

At the bottom left, there is a footer note: 'Hadoop, 2016.'

Um “Olá mundo!” utilizando o Hadoop

- Etapas de execução do programa:
 - Analisando os resultados (HDFS – Conteúdo do diretório saída):

Hadoop Overview Datanodes Snapshot Startup Progress Utilities ▾

Browse Directory

/user/hduser/Saida

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	hduser	supergroup	0 B	23/03/2017 19:36:28	1	128 MB	_SUCCESS
-rw-r--r--	hduser	supergroup	71 B	23/03/2017 19:36:28	1	128 MB	part-00000

Hadoop, 2016.

Um “Olá mundo!” utilizando o Hadoop

- Etapas de execução do programa:
 - Analisando os resultados (HDFS – Conteúdo do arquivo de saída):



A screenshot of a file browser interface. At the top, there is a dark header bar with a 'Abrir' button and a '+' icon. Below the header, a list of files is displayed in a table format:

001	18664.428
004	6644.5
007	37482.75
008	2235.0
009	21051.75

Conclusão

- Primeiro programa completo.
- Manipulação de arquivos no HDFS e processamento.

- Criando, compilando e executando um programa Hadoop/MapReduce (Parte 1).



Aula 4.6.1. Criando, compilando e executando um programa Hadoop/MapReduce (Parte 1)

- Criando, compilando e executando um programa Hadoop/MapReduce (Parte 1).

- Criando, compilando e executando um programa Hadoop/MapReduce (Parte 1).

- Criando, compilando e executando um programa Hadoop/MapReduce (Parte 2).



Aula 4.6.2. Criando, compilando e executando um programa Hadoop/MapReduce (Parte 2)

- Criando, compilando e executando um programa Hadoop/MapReduce (Parte 2).

- Criando, compilando e executando um programa Hadoop/MapReduce (Parte 2).

- Métodos configure e close (MapReduceBase).



Aula 4.7. Métodos configure e close

- Métodos MapReduceBase:

- Configure.
 - Close.

Métodos configure e close

- Métodos presentes na classe MapReduceBase:
 - close() – Não possui parâmetros.
 - configure(JobConf job) – JobConf do job MapReduce.
- Na classe MapReduceBase esses métodos não possuem nenhum código.

- Método configure(...):
 - É a única maneira de acessar o JobConf dentro de um Mapper ou Reducer.
 - É uma forma de realizar um setup.
 - É chamado pelo framework antes de iniciar os métodos Map ou Reduce.
 - Usado muitas vezes para:
 - Instanciar objetos.
 - Atribuir valores iniciais para variáveis.
 - Registrar log.

Métodos configure e close

- Pelo JobConf, que é passado como parâmetro, é possível enviar informações do método main() da classe principal para os métodos Map ou Reduce.
- SetStrings.

Métodos configure e close

```
import org.apache.hadoop.fs.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.util.*;

public class ExemploIGTI extends Configured implements Tool
{
    public static void main (final String[] args) throws Exception {
        int res = ToolRunner.run(new Configuration(), new ExemploIGTI(), args);
        System.exit(res);
    }

    public int run (final String[] args) throws Exception {
        try{
            JobConf conf = new JobConf(getConf(), ExemploIGTI.class);
            conf.setJobName("Ola Mundo");
            conf.setStrings("codigoExecucao", Integer.toString(337)); ←
        }
    }
}
```

Métodos configure e close

```
public static class ReduceIGTI extends MapReduceBase implements Reducer<Text, Text, Text, Text> {  
  
    Text value;  
    public void configure(JobConf job) { ←  
        value = new Text();  
        System.out.println(job.getStrings("codigoExecucao")); ←  
    }  
  
    public void reduce (Text key, Iterator<Text> values, OutputCollector<Text, Text> output, Reporter reporter) throws IOException {  
        double media = 0.0;  
        int acumuladorItens = 0, contaVendas = 0;  
  
        while (values.hasNext()) {  
            value = values.next();  
            contaVendas++;  
            acumuladorItens += Integer.parseInt(value.toString());  
        }  
        media = acumuladorItens / new Double(contaVendas);  
        value.set(String.valueOf(media));  
        output.collect(key, value);  
    }  
}
```

Métodos configure e close

- Método close():
 - Não possui parâmetros.
 - Chamado pelo framework quando todas as entradas tiverem sido processadas pelos métodos Map ou Reduce.
 - Pode ser usado para arrematar o processamento.
 - Fechar algum arquivo suplementar que foi aberto.
 - Efetivar alterações.
 - Gravar algum dado ou log adicional/final.

Métodos configure e close

```
public static class ReduceIGTI extends MapReduceBase implements Reducer<Text, Text, Text, Text> {  
    Text value;  
    public void configure(JobConf job) {  
        value = new Text();  
        System.out.println(job.getStrings("codigoExecucao"));  
    }  
  
    public void reduce (Text key, Iterator<Text> values, OutputCollector<Text, Text> output, Reporter reporter) throws IOException {  
        double media = 0.0;  
        int acumuladorItens = 0, contaVendas = 0;  
  
        while (values.hasNext()) {  
            value = values.next();  
            contaVendas++;  
            acumuladorItens += Integer.parseInt(value.toString());  
        }  
        media = acumuladorItens / new Double(contaVendas);  
        value.set(String.valueOf(media));  
        output.collect(key, value);  
    }  
  
    public void close() {  
        System.out.println("Processamento encerrado ");  
    }  
}
```

- Métodos configure e close.
- setString.
- Enviar dados da classe main() para os métodos Map e Reduce.

■ Próxima aula

- ❑ Definindo uma função combine.



Aula 4.8. Definindo uma função Combine

Nesta aula

- Função Combine.
- Utilidade e formas de uso.

Definindo uma função Combine

- Jobs MapReduce são limitadas pela largura de banda disponível no cluster.
- Para minimizar a transferência de dados entre tarefas Map e Reduce, existe a função Combine.
- Recebe como entrada a saída da função Map.
- A utilização da função Combine não deve alterar o resultado.

Definindo uma função Combine

- Combiner não implementa uma interface específica.
- Combiner implementa a interface Reducer e o método reduce().
- Alguns apelidam o Combiner de semi-reducer ou pré-reducer.
- É totalmente opcional.
- Pode apresentar comportamento diferenciado quando executado em cluster ou localmente.

Definindo uma função Combine

Node 1

(1950, 0)
(1950, 20)
(1950, 10)

Node 2

(1950, 25)
(1950, 15)

Entrada para Reduce

(1950, [0, 20, 10, 25, 15])

Saída de Reduce

(1950, 25)

Entrada para Reduce com Combine

(1950, [20, 25])

- Mais sucintamente podemos expressar a função Reduce na seguinte forma:

$$\max(0, 20, 10, 25, 15) = \max(\max(0, 20, 10), \max(25, 15)) = \max(20, 25) = 25$$

Definindo uma função Combine

- Cuidado! Nem todos os jobs podem utilizar a função Combine e produzir resultados corretos.
- Se ao invés do maior valor quiséssemos a média dos valores?

$$\text{mean}(0, 20, 10, 25, 15) = 14$$

- Não poderíamos usar a nossa função Combine, pois:

$$\text{mean}(\text{mean}(0, 20, 10), \text{mean}(25, 15)) = \text{mean}(10, 20) = 15$$

Definindo uma função Combine

- A função Combine não substitui a função Reduce, pois os dados são processados apenas localmente e não globalmente.
- Podem existir dados com a mesma chave em diferentes Datanodes.
- Combine pode auxiliar a diminuir a quantidade de dados mesclados entre diferentes Mappers e Reducers.
- setCombinerClass.

Definindo uma função Combine

```
conf.setOutputKeyClass(Text.class);
conf.setOutputValueClass(Text.class);
conf.setMapperClass(MapIGTI.class);
conf.setCombinerClass(ReduceIGTI.class); ←
conf.setReducerClass(ReduceIGTI.class);
JobClient.runJob(conf);
```

Definindo uma função Combine

```
Map input records=20
Map output records=20
Map output bytes=260
Map output materialized bytes=108
Input split bytes=226
Combine input records=7

Combine output records=7
Spilled Records=7
Failed Shuffles=0
Merged Map outputs=0
GC time elapsed (ms)=379
CPU time spent (ms)=2180
Physical memory (bytes) snapshot=485355520
Virtual memory (bytes) snapshot=3825672192
Total committed heap usage (bytes)=391118848
File Input Format Counters
    Bytes Read=3084
```

Combiners:

- Quando usar.
- Forma de usar.
- Benefícios.

■ Próxima aula

- ❑ Depurando um *job* e os *logs* do Hadoop.



Aula 4.9. Depurando um job e os logs do Hadoop

Hadoop:

- Depuração.
- Logs.

Depurando um job e os logs do Hadoop

- A depuração de programas Hadoop pode ser feita de diversas maneiras:
 - Registro de logs;
 - Counters;
 - Plugins.

Depurando um job e os logs do Hadoop

- Registro de logs:
 - System.out.println.
 - Registro nos logs do Hadoop.

Depurando um job e os logs do Hadoop

```
public static class ReduceIGTI extends MapReduceBase implements Reducer<Text, Text, Text, Text> {  
    public void reduce (Text key, Iterator<Text> values, OutputCollector<Text, Text> output, Reporter reporter) throws IOException {  
        double media = 0.0;  
        int acumuladorItens = 0, contaVendas = 0;  
        Text value = new Text();  
  
        while (values.hasNext()) {  
            value = values.next();  
            contaVendas++;  
            acumuladorItens += Integer.parseInt(value.toString());  
        }  
        media = acumuladorItens / new Double(contaVendas);  
  
        if (media > 15000)  
            System.out.println("Chave " + key.toString() + " possui mais de 15000 itens ( " + String.valueOf(media) + " ) ");  
        value.set(String.valueOf(media));  
        output.collect(key, value);  
    }  
}
```



Depurando um job e os logs do Hadoop

IGTI

The screenshot shows the Hadoop Startup Progress page. At the top, there is a navigation bar with tabs: Hadoop, Overview, Datanodes, Datanode Volume Failures, Snapshot, Startup Progress, and Utilities. A red arrow points from the word "Utilities" to a dropdown menu that contains "Browse the file system" and "Logs". Below the navigation bar, the page title is "Startup Progress". It displays the elapsed time and percent complete (1 sec, 100%). The main content is a table showing the startup phases and their completion status:

Phase	Completion	Elapsed Time
Loading fsimage /usr/local/hadoop/tmp/dfs/name/current/fsimage_00000000000000000000 353 B	100%	0 sec
inodes (0/0)	100%	
delegation tokens (0/0)	100%	
cache pools (0/0)	100%	
Loading edits	100%	0 sec
Saving checkpoint	100%	0 sec
Safe mode	100%	0 sec
awaiting reported blocks (0/0)	100%	

Depurando um job e os logs do Hadoop

Directory: /logs/

SecurityAuth-hduser.audit	0 bytes	19/03/2017 19:45:49
hadoop-hduser-datanode-JOAOLINUX.log	609813 bytes	26/03/2017 18:04:16
hadoop-hduser-datanode-JOAOLINUX.out	718 bytes	26/03/2017 16:46:39
hadoop-hduser-datanode-JOAOLINUX.out.1	718 bytes	23/03/2017 19:22:58
hadoop-hduser-datanode-JOAOLINUX.out.2	718 bytes	21/03/2017 23:50:03
hadoop-hduser-datanode-JOAOLINUX.out.3	718 bytes	19/03/2017 21:38:12
hadoop-hduser-datanode-JOAOLINUX.out.4	716 bytes	19/03/2017 21:23:11
hadoop-hduser-datanode-JOAOLINUX.out.5	716 bytes	19/03/2017 21:17:54
hadoop-hduser-namenode-JOAOLINUX.log	702092 bytes	26/03/2017 18:04:15
hadoop-hduser-namenode-JOAOLINUX.out	5001 bytes	26/03/2017 17:37:08
hadoop-hduser-namenode-JOAOLINUX.out.1	5001 bytes	23/03/2017 20:20:24
hadoop-hduser-namenode-JOAOLINUX.out.2	5006 bytes	21/03/2017 23:59:15
hadoop-hduser-namenode-JOAOLINUX.out.3	5006 bytes	19/03/2017 22:28:31
hadoop-hduser-namenode-JOAOLINUX.out.4	716 bytes	19/03/2017 21:23:07
hadoop-hduser-namenode-JOAOLINUX.out.5	716 bytes	19/03/2017 21:17:50
hadoop-hduser-secondarynamenode-JOAOLINUX.log	244436 bytes	26/03/2017 17:47:51
hadoop-hduser-secondarynamenode-JOAOLINUX.out	718 bytes	26/03/2017 16:46:45
hadoop-hduser-secondarynamenode-JOAOLINUX.out.1	718 bytes	23/03/2017 19:23:03
hadoop-hduser-secondarynamenode-JOAOLINUX.out.2	718 bytes	21/03/2017 23:50:09
hadoop-hduser-secondarynamenode-JOAOLINUX.out.3	718 bytes	19/03/2017 21:38:18
hadoop-hduser-secondarynamenode-JOAOLINUX.out.4	716 bytes	19/03/2017 21:23:20
hadoop-hduser-secondarynamenode-JOAOLINUX.out.5	716 bytes	19/03/2017 21:18:03
userlogs/	4096 bytes	26/03/2017 18:16:55
yarn-hduser-nodemanager-JOAOLINUX.log	1521399 bytes	26/03/2017 18:04:22
yarn-hduser-nodemanager-JOAOLINUX.out	1571 bytes	26/03/2017 16:46:59
yarn-hduser-nodemanager-JOAOLINUX.out.1	1571 bytes	23/03/2017 19:23:15
yarn-hduser-nodemanager-JOAOLINUX.out.2	1578 bytes	21/03/2017 23:50:25
yarn-hduser-nodemanager-JOAOLINUX.out.3	1571 bytes	19/03/2017 21:38:55
yarn-hduser-nodemanager-JOAOLINUX.out.4	1571 bytes	19/03/2017 21:23:46
yarn-hduser-nodemanager-JOAOLINUX.out.5	1571 bytes	19/03/2017 21:18:31
yarn-hduser-resourcemanager-JOAOLINUX.log	1461861 bytes	26/03/2017 18:04:20



Depurando um job e os logs do Hadoop

IGTI



All Applications

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	Vcores Used
5	0	0	5	0	0 B	8 GB	0 B	0

Scheduler Metrics

Scheduler Type		Scheduling Resource Type		<memory:1024, vCores:1>			
Capacity Scheduler	[MEMORY]						
Show 20 entries							
ID	User	Name	Application Type	Queue	StartTime	FinishTime	Duration
application_1490557613925_0005	hduser	Ola Mundo	MAPREDUCE	default	Sun Mar 26 18:03:35 -0300 2017	Sun Mar 26 18:03:35 -0300 2017	00:00:00
application_1490557613925_0004	hduser	Ola Mundo	MAPREDUCE	default	Sun Mar 26 17:55:42 -0300 2017	Sun Mar 26 17:55:42 -0300 2017	00:00:00
application_1490557613925_0003	hduser	Ola Mundo	MAPREDUCE	default	Sun Mar 26 17:49:11 -0300 2017	Sun Mar 26 17:49:11 -0300 2017	00:00:00
application_1490557613925_0002	hduser	Ola Mundo	MAPREDUCE	default	Sun Mar 26 17:42:29 -0300 2017	Sun Mar 26 17:42:29 -0300 2017	00:00:00
application_1490557613925_0001	hduser	Ola Mundo	MAPREDUCE	default	Sun Mar 26 16:47:41 -0300 2017	Sun Mar 26 16:47:41 -0300 2017	00:00:00

Showing 1 to 5 of 5 entries

Directory: /logs/userlogs/

Parent Directory

[application_1489968623877_0001/](#) 4096 bytes 19/03/2017 21:11:48
[application_1489969102190_0001/](#) 4096 bytes 19/03/2017 21:19:50
[application_1489969416842_0001/](#) 4096 bytes 19/03/2017 21:25:23
[application_1489970317324_0001/](#) 4096 bytes 19/03/2017 21:43:56
[application_1490151019328_0001/](#) 4096 bytes 22/03/2017 21:49:02
[application_1490151019328_0002/](#) 4096 bytes 22/03/2017 21:52:55
[application_1490151019328_0003/](#) 4096 bytes 22/03/2017 21:56:57
[application_1490151019328_0004/](#) 4096 bytes 22/03/2017 22:02:24
[application_1490151019328_0005/](#) 4096 bytes 22/03/2017 22:06:58
[application_1490151019328_0006/](#) 4096 bytes 22/03/2017 22:13:00
[application_1490151019328_0007/](#) 4096 bytes 22/03/2017 22:18:31
[application_1490151019328_0008/](#) 4096 bytes 22/03/2017 22:31:39
[application_1490151019328_0009/](#) 4096 bytes 22/03/2017 22:33:57
[application_1490151019328_0010/](#) 4096 bytes 22/03/2017 22:37:17
[application_1490151019328_0011/](#) 4096 bytes 22/03/2017 22:50:18
[application_1490151019328_0012/](#) 4096 bytes 22/03/2017 22:52:17
[application_1490151019328_0013/](#) 4096 bytes 22/03/2017 22:56:56
[application_1490151019328_0014/](#) 4096 bytes 22/03/2017 22:59:53
[application_1490307791670_0002/](#) 4096 bytes 23/03/2017 22:28:33
[application_1490307791670_0003/](#) 4096 bytes 23/03/2017 22:32:13
[application_1490307791670_0004/](#) 4096 bytes 23/03/2017 22:38:15
[application_1490557613925_0001/](#) 4096 bytes 26/03/2017 16:49:27
[application_1490557613925_0002/](#) 4096 bytes 26/03/2017 17:42:57
[application_1490557613925_0003/](#) 4096 bytes 26/03/2017 17:49:37
[application_1490557613925_0004/](#) 4096 bytes 26/03/2017 17:56:11
[application_1490557613925_0005/](#) 4096 bytes 26/03/2017 18:03:56

Directory: /logs/userlogs/application_1490557613925_0005/

[Parent Directory](#)

[container_1490557613925_0005_01_000001/](#) 4096 bytes 26/03/2017 18:03:36
[container_1490557613925_0005_01_000002/](#) 4096 bytes 26/03/2017 18:03:45
[container_1490557613925_0005_01_000003/](#) 4096 bytes 26/03/2017 18:03:45
[container_1490557613925_0005_01_000004/](#) 4096 bytes 26/03/2017 18:04:02



Depurando um job e os logs do Hadoop

Directory: /logs/userlogs/application_1490557613925_0005/container_1490557613925_0005_0001

[Parent Directory](#)

stderr	0 bytes	26/03/2017 18:03:56
stdout	163 bytes	26/03/2017 18:04:11
syslog	3125 bytes	26/03/2017 18:04:11
syslog.shuffle	3805 bytes	26/03/2017 18:04:10



Depurando um job e os logs do Hadoop

```
Chave 001 possui mais de 15000 itens ( 18664.428571428572 )
Chave 007 possui mais de 15000 itens ( 37482.75 )
Chave 009 possui mais de 15000 itens ( 21051.75 )
```

Usando System.err.println(...);

localhost:50070/logs/userlogs/application_1490557613925_0006/container_1490557613925_0006_01_000004/

Directory: /logs/userlogs/application_1490557613925_0006/container_1490557613925_0006_01_000004

[Parent Directory](#)

stderr	406 bytes	26/03/2017 18:29:22
stdout	0 bytes	26/03/2017 18:29:01
syslog	2746 bytes	26/03/2017 18:29:21
syslog.shuffle	4433 bytes	26/03/2017 18:29:21

localhost:50070/logs/userlogs/application_1490557613925_0006/container_1490557613925_0006_01_000004/stderr

```
Chave 001 possui mais de 15000 itens ( 18664.428571428572 )
Chave 007 possui mais de 15000 itens ( 37482.75 )
Chave 009 possui mais de 15000 itens ( 21051.75 )
log4j:WARN No appenders could be found for logger (org.apache.hadoop.metrics2.impl.MetricsSystemImpl).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
```

Depurando um job e os logs do Hadoop

- Counters:

```
static enum Classificacao { MAIOR_15000, MENOR_15000, TODOS }  
  
public static class ReduceIGTI extends MapReduceBase implements Reducer<Text, Text, Text, Text> {  
  
    public void reduce (Text key, Iterator<Text> values, OutputCollector<Text, Text> output, Reporter reporter) throws IOException {  
        double media = 0.0;  
        int acumuladorItens = 0, contaVendas = 0;  
        Text value = new Text();  
  
        while (values.hasNext()) {  
            value = values.next();  
            contaVendas++;  
            acumuladorItens += Integer.parseInt(value.toString());  
        }  
        media = acumuladorItens / new Double(contaVendas);  
  
        if (media > 15000)  
            reporter.getCounter(Classificacao.MAIOR_15000).increment(1);  
        else  
            reporter.getCounter(Classificacao.MENOR_15000).increment(1);  
  
        reporter.getCounter(Classificacao.TODOS).increment(1);  
  
        value.set(String.valueOf(media));  
        output.collect(key, value);  
    }  
}
```

Depurando um job e os logs do Hadoop

- Counters:

```
Map-Reduce Framework
  Map input records=20
  Map output records=20
  Map output bytes=260
  Map output materialized bytes=312
  Input split bytes=226
  Combine input records=0
  Combine output records=0
  Reduce input groups=5
  Reduce shuffle bytes=312
  Reduce input records=20
  Reduce output records=5
  Spilled Records=40
  Shuffled Maps =2
  Failed Shuffles=0
  Merged Map outputs=2
  GC time elapsed (ms)=295
  CPU time spent (ms)=2940
  Physical memory (bytes) snapshot=651509760
  Virtual memory (bytes) snapshot=5749903360
  Total committed heap usage (bytes)=467140608
IGTI.ExemploIGTI$Classificacao ←
  MAIOR_15000=3
  MENOR_15000=2
  TODOS=5
Shuffle Errors
  BAD_ID=0
  CONNECTION=0
  IO_ERROR=0
  WRONG_LENGTH=0
  WRONG_MAP=0
  WRONG_REDUCE=0
File Input Format Counters
  Bytes Read=3084
File Output Format Counters
  Bytes Written=71
```

Depurando um job e os logs do Hadoop

- Plugins:
 - Eclipse.
 - Netbeans.

- ✓ Apresentadas maneiras de depurar código no Hadoop.

■ Próxima aula

- Algoritmos iterativos com o Hadoop.

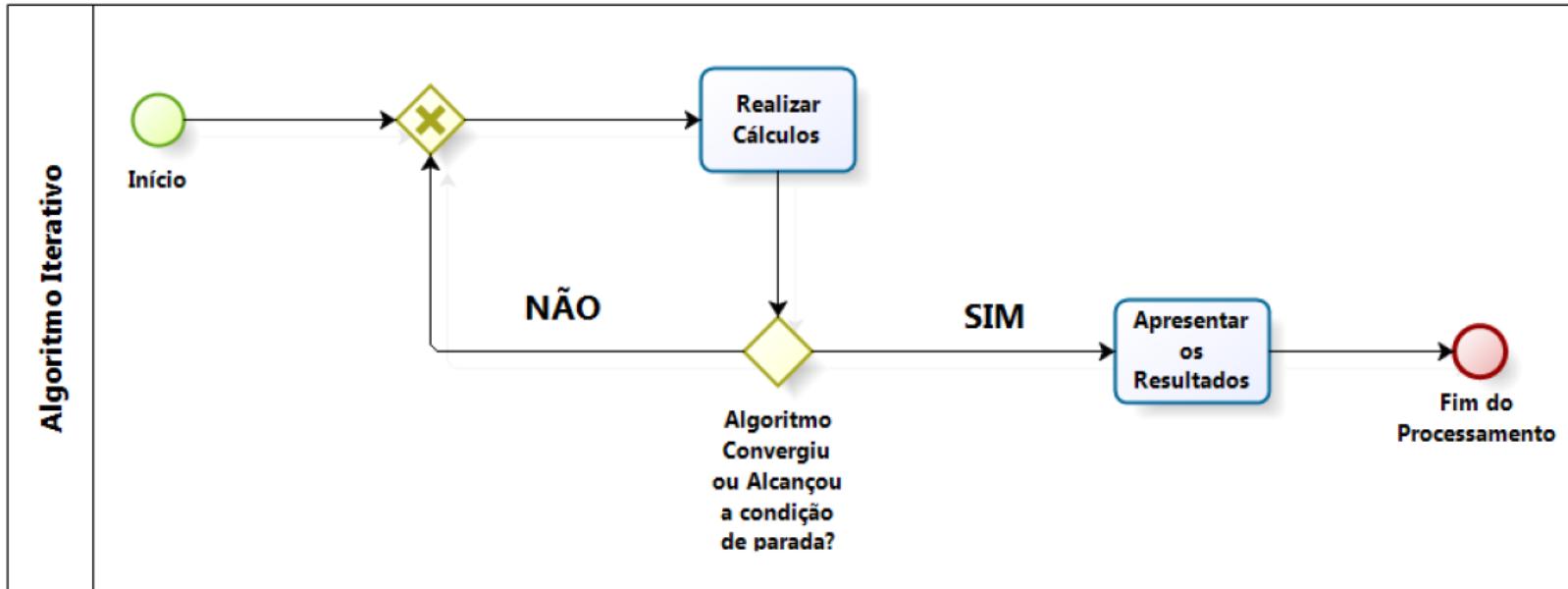


Aula 4.10. Algoritmos iterativos com Hadoop

- ❑ Algoritmos iterativos e o Hadoop.
- ❑ Exemplo de um algoritmo iterativo.

- Hadoop não é bom em lidar com algoritmos iterativos.
- A cada iteração os dados precisam ser gravados e recuperados do disco.
- Dados não são mantidos em memória.
- Isso causa overhead.

Fluxo de um algoritmo iterativo



- Exemplo:
 - 1 job para calcular as médias.
 - 1 job para encontrar qual a média com maior valor.

Algoritmos iterativos com Hadoop

```
public int run (final String[] args) throws Exception {
    try{
        JobConf conf = new JobConf(getConf(), ExemploIGTI.class);
        conf.setJobName("Media");
        conf.setStrings("codigoExecucao", Integer.toString(337));

        final FileSystem fs = FileSystem.get(conf);
        Path diretorioEntrada = new Path("Entrada"), diretorioSaida = new Path("Saida");

        /* Criar um diretório de entrada no HDFS */
        if (!fs.exists(diretorioEntrada))
            fs.mkdirs(diretorioEntrada);

        /* Adicionar um arquivo para ser processado */
        fs.copyFromLocalFile(new Path("/usr/local/hadoop/Dados/arquivoBigData.txt"), diretorioEntrada);

        /* Atribuindo os diretórios de Entrada e Saida para o Job */

        FileInputFormat.setInputPaths(conf, diretorioEntrada);
        FileOutputFormat.setOutputPath(conf, diretorioSaida);

        conf.setOutputKeyClass(Text.class);
        conf.setOutputValueClass(Text.class);
        conf.setMapperClass(MapIGTI.class);
        conf.setReducerClass(ReduceIGTI.class);
        JobClient.runJob(conf);

        JobConf conf_total = new JobConf(getConf(), ExemploIGTI.class);
        conf_total.setJobName("Maior Valor");
        FileInputFormat.setInputPaths(conf_total, diretorioSaida);
        FileOutputFormat.setOutputPath(conf_total, new Path("Total"));
        conf_total.setOutputKeyClass(Text.class);
        conf_total.setOutputValueClass(Text.class);
        conf_total.setMapperClass(MapIGTIMaiorQuantidade.class);
        conf_total.setReducerClass(ReduceIGTIMaiorQuantidade.class);
        JobClient.runJob(conf_total);
    }
    catch ( Exception e ) {
        throw e;
    }
    return 0;
}
```

Algoritmos iterativos com Hadoop

localhost:8088/cluster 90% C Pesquisa

All Applications

 hadoop

Cluster Metrics												
Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	Vcores Used	Vcores Total	Vcores Reserved	Active Nodes	
10	0	0	10	0	0 B	8 GB	0 B	0	8	0	1	

Scheduler Metrics

Scheduler Type			Scheduling Resource Type			Minimum Allocation		
Capacity Scheduler			[MEMORY]			<memory:1024, vCores:1>		
Show	20	entries						
ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus
application_1490557613925_0010	houser	Maior Valor	MAPREDUCE	default	Sun Mar 26 20:56:19 -0300 2017	Sun Mar 26 20:56:51 -0300 2017	FINISHED	SUCCEEDED
application_1490557613925_0009	houser	Media	MAPREDUCE	default	Sun Mar 26 20:55:26 -0300 2017	Sun Mar 26 20:56:17 -0300 2017	FINISHED	SUCCEEDED

Hadoop Overview Datanodes Snapshot Startup Progress Utilities ▾

Browse Directory

/user/hduser

Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	hduser	supergroup	0 B	26/03/2017 20:55:22	0	0 B	Entrada
drwxr-xr-x	hduser	supergroup	0 B	26/03/2017 20:56:16	0	0 B	Saida
drwxr-xr-x	hduser	supergroup	0 B	26/03/2017 20:56:50	0	0 B	Total

Algoritmos iterativos com Hadoop

igti

Diretório entrada – Arquivo original

MapIGTI
ReduceIGTI

Diretório saída – Arquivo com as médias

	Abrir	+
001	18664.428	
004	6644.5	
007	37482.75	
008	2235.0	
009	21051.75	

Diretório total – Maior média

MapIGTIMaiorQuantidade.class ReduceIGTIMaiorQuantidade.class

Abrir ▾

Algoritmos iterativos com Hadoop

	Abrir	+
001	18664.428	
004	6644.5	
007	37482.75	
008	2235.0	
009	21051.75	

Arquivo de entrada

```
public static class MapIGTIMaiorQuantidade extends MapReduceBase implements Mapper<LongWritable, Text, Text, Text> {  
    public void map(LongWritable key, Text value, OutputCollector<Text, Text> output, Reporter reporter) throws IOException {  
        Text txtChave = new Text();  
        Text txtValor = new Text();  
  
        String[] dados = value.toString().split("\t");  
  
        txtChave.set("1");  
        txtValor.set(dados[1]);  
  
        output.collect(txtChave, txtValor);  
    }  
}
```

Algoritmos iterativos com Hadoop

```
public static class ReduceIGTIMaiorQuantidade extends MapReduceBase implements Reducer<Text, Text, Text, Text> {  
    public void reduce (Text key, Iterator<Text> values, OutputCollector<Text, Text> output, Reporter reporter) throws IOException {  
        double maiorValor = 0.0;  
  
        Text value = new Text();  
  
        while (values.hasNext()) {  
            value = values.next();  
            if (Double.parseDouble(value.toString()) > maiorValor)  
                maiorValor = Double.parseDouble(value.toString());  
  
        }  
        value.set(String.valueOf(maiorValor));  
        output.collect(key, value);  
    }  
}
```

Algoritmos iterativos com Hadoop

```
    File System Counters
        FILE: Number of bytes read=77
        FILE: Number of bytes written=355485
        FILE: Number of read operations=0
        FILE: Number of large read operations=0
        FILE: Number of write operations=0
        HDFS: Number of bytes read=313
        HDFS: Number of bytes written=11
        HDFS: Number of read operations=9
        HDFS: Number of large read operations=0
        HDFS: Number of write operations=2
    Job Counters
        Launched map tasks=2
```

17/03/26 20:56:18 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
17/03/26 20:56:18 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
17/03/26 20:56:19 INFO mapred.FileInputFormat: Total input paths to process : 1
17/03/26 20:56:19 INFO mapreduce.JobSubmitter: number of splits:2
17/03/26 20:56:19 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1490557613925_0010
17/03/26 20:56:19 INFO impl.YarnClientImpl: Submitted application application_1490557613925_0010
17/03/26 20:56:19 INFO mapreduce.Job: The url to track the job: http://JOAOLINUX:8088/proxy/application_1490557613925_0010/
17/03/26 20:56:19 INFO mapreduce.Job: Running job: job_1490557613925_0010
17/03/26 20:56:33 INFO mapreduce.Job: Job job_1490557613925_0010 running in uber mode : false
17/03/26 20:56:33 INFO mapreduce.Job: map 0% reduce 0%
17/03/26 20:56:43 INFO mapreduce.Job: map 100% reduce 0%
17/03/26 20:56:51 INFO mapreduce.Job: map 100% reduce 100%
17/03/26 20:56:52 INFO mapreduce.Job: Job job_1490557613925_0010 completed successfully
17/03/26 20:56:52 INFO mapreduce.Job: Counters: 49

Conclusão

- Iteratividade do Hadoop.
- Criação de um algoritmo iterativo.
- Fechamento do capítulo 4.



Desenvolvimento de Soluções com MapReduce utilizando Hadoop

Capítulo 5. Parâmetros de Desempenho

Prof. João Paulo Barbosa Nascimento

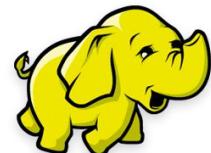


Aula 5.1. Principais parâmetros de desempenho

- ❑ Principais parâmetros de desempenho.
- ❑ Influência dos parâmetros de desempenho.

Principais parâmetros de desempenho

- Hadoop possui mais de 200 parâmetros ajustáveis.
- Entender todos os recursos oferecidos por eles é uma tarefa árdua.
- Muitas combinações podem ser criadas.
- Todos as informações podem ser encontradas na página oficial do Hadoop (<http://hadoop.org>).



Principais parâmetros de desempenho

- Esses parâmetros podem afetar a execução de determinado job.
- Os ajustes exigem planejamento e, principalmente, experimentação.

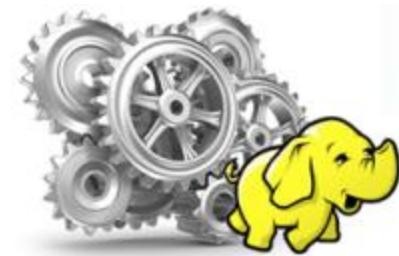


Principais parâmetros de desempenho

- Um mesmo parâmetro pode apresentar comportamento diferente para bases de dados diferentes:
 - Perda ou ganho de desempenho.
- Configurados nos arquivos de configuração XML:
 - Replicação.
 - Tamanho do buffer para ordenação.
 - Tamanho do bloco.



- **mapred.tasktracker.map/reduce.tasks.maximum:**
 - Define a quantidade máxima simultânea de tarefas map/reduce.
 - Que executará em um nó (TaskTracker).
 - Valor padrão: 2.
- **mapred.child.java.opts:**
 - Quantidade de memória atribuída à JVM.
 - Permite alterar várias características da JVM.



Principais parâmetros de desempenho

- **io.sort.mb:**

- Utilizado para definir o tamanho do buffer.
- Ordenação dos dados ao final da fase Map.
- Valor padrão é 100MB.

- **io.sort.spill.percent:**

- Utilizado em conjunto com **io.sort.mb**.
- Define o momento em que os dados do buffer serão enviados ao disco.
- Thread disparada no momento que o buffer atinge o valor definido.
- Padrão 0.80 (80%).

- **mapred.local.dir:**

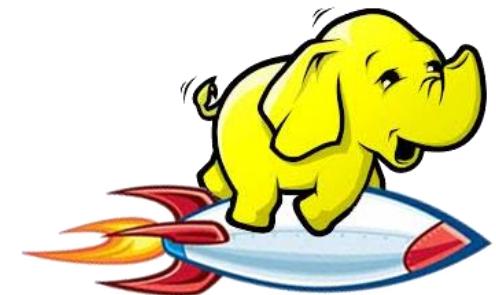
- Caminho onde os dados que estão no buffer serão descarregados em disco.
 - Pode ser apenas um endereço ou vários (separados por vírgula).

- **io.sort.fator:**

- Antes do fim da tarefa Map, os vários arquivos precisam ser mesclados em apenas um.
- Controla o número máximo de streams utilizados para mesclar.
- O valor padrão é 10.

- **mapred.reduce.parallel.copies:**

- Número de threads que irão buscar o resultado da função Map.
- Paralelamente.
- Valor padrão 5.



Principais parâmetros de desempenho

- Papel fundamental:
 - No tempo de processamento.
 - Na escalabilidade.
- Estudos apontam 35% de melhoria do tempo.
- Alguns relatam até 80%.
- Melhoria nos resultados de speedup e eficiência.

Conclusão

- Parâmetros de ajuste de desempenho.
- Importância do planejamento do ajuste.

- Medidas de desempenho em cluster:

- Speedup.
 - Eficiência.

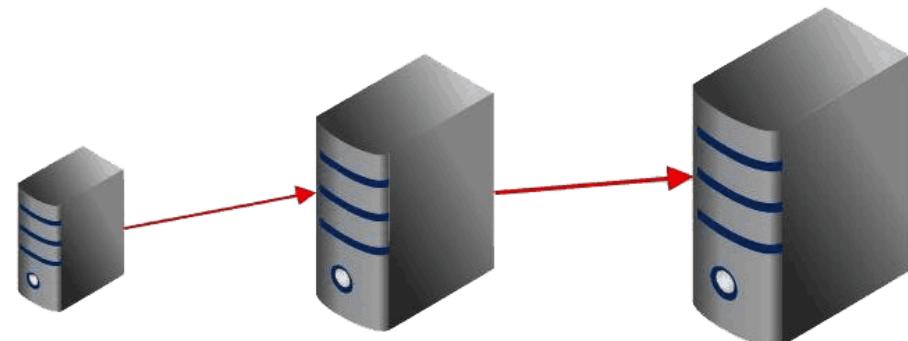


Aula 5.2. Medidas de desempenho em cluster

Nesta aula

- Escalabilidade.
- Speedup.
- Eficiência.

- Escalabilidade de um algoritmo paralelo, em uma arquitetura paralela.
- Característica desejável em qualquer sistema.
- Capacidade de manipular uma quantidade de trabalho de forma uniforme.
- Estar preparado para crescer.



- Análise de escalabilidade.
 - Combinação algoritmo x arquitetura x dados.
 - Buscar a melhor combinação.



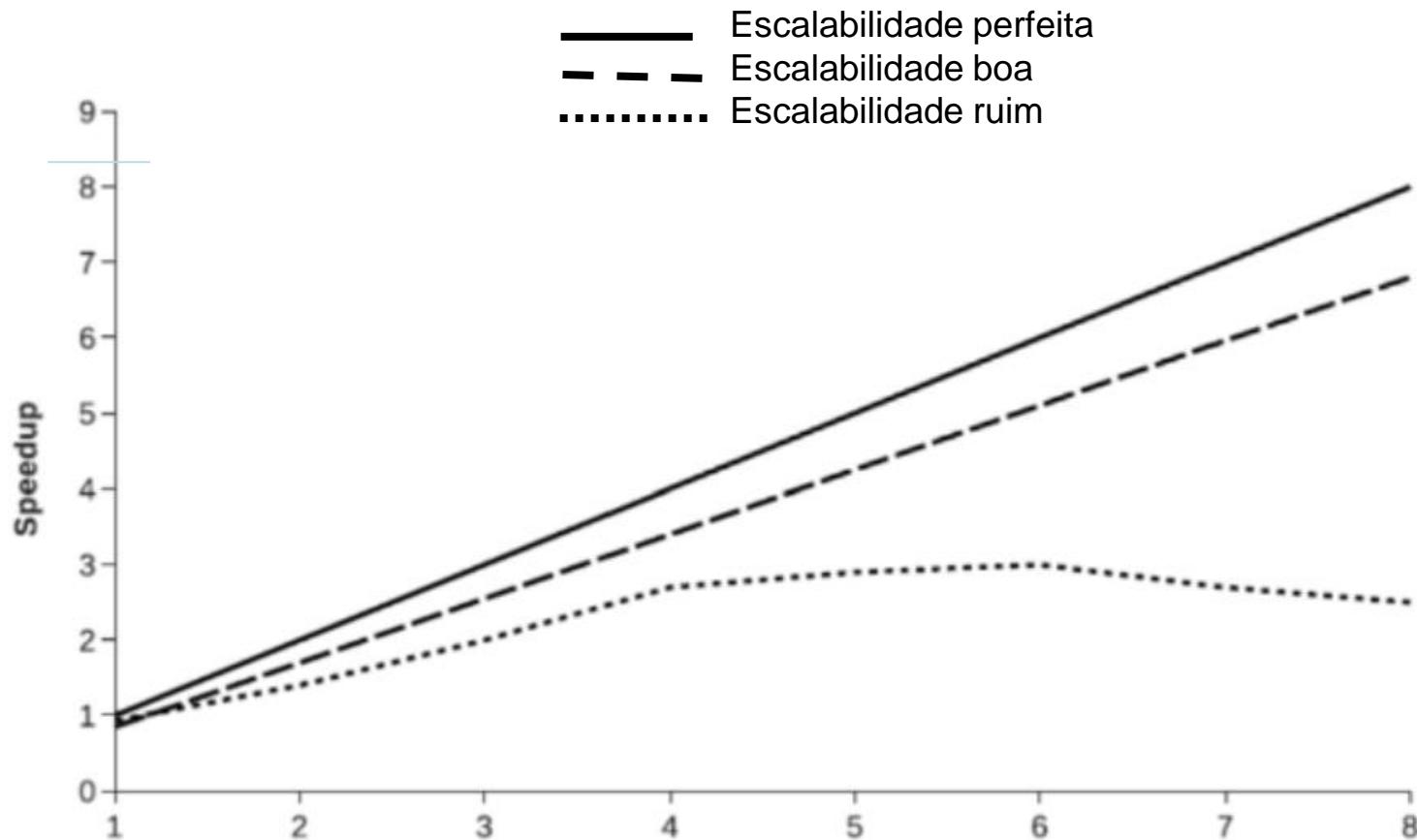
- Speedup:
 - Medida que diz o quanto mais rápido uma tarefa será executada.
 - Usando uma configuração de cluster com alguma melhoria.
 - Formulação da lei de Amdahl.

$$S(p) = \frac{1}{(1 - pctPar) + \frac{pctPar}{p}}$$

- **pctPar** é a porcentagem de tempo de execução que irá executar em paralelo.
- **p** é o número de núcleos aos quais a aplicação irá executar em paralelo.

$$S(p) = \frac{1}{(1 - pctPar) + \frac{pctPar}{p}}$$

Speedup



- Outra métrica relacionada ao Speedup.
- Essa métrica nos diz o quanto bem os recursos estão sendo utilizados.
- Para calcular a eficiência, divide-se o Speedup pelo número de núcleos utilizados.

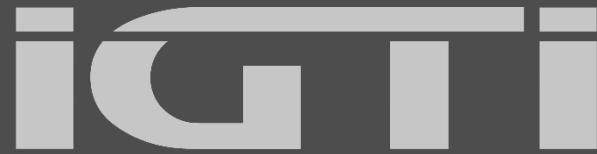
$$E(p) = \frac{S(p)}{p} = \frac{T(1)}{pT(p)}$$

Escalabilidade:

- Speedup.
- Eficiência.

■ Próxima aula

- Apache Hive.



Desenvolvimento de Soluções com MapReduce utilizando Hadoop

Capítulo 6. Um pouco mais do Ecossistema Hadoop

Prof. João Paulo Barbosa Nascimento



Aula 6.1. Apache Hive

Nesta aula

- Apresentação do Apache Hive.
- Aplicação, utilidade e vantagens do uso.

- Começou com a necessidade do Facebook de gerenciar grandes quantidades de dados.
- Os dados eram armazenados em um SGBD todas as noites.
- ETL era executado sobre dados.
- A quantidade crescia rapidamente:
 - 2006: 1TB ao dia.
 - 2010: 10TB.
 - 2013: aproximadamente 500.000.000 de registros de log por dia.
- Era preciso encontrar uma forma de gerenciar esses dados eficientemente.



- O Hive é uma estrutura de DW.
- Não é um banco transacional.
- DW é um banco de dados específico para propósito de análise.
- Construído sobre o Hadoop.
- Pode compilar consultas SQL como jobs MapReduce.
- Executar esses jobs no cluster.

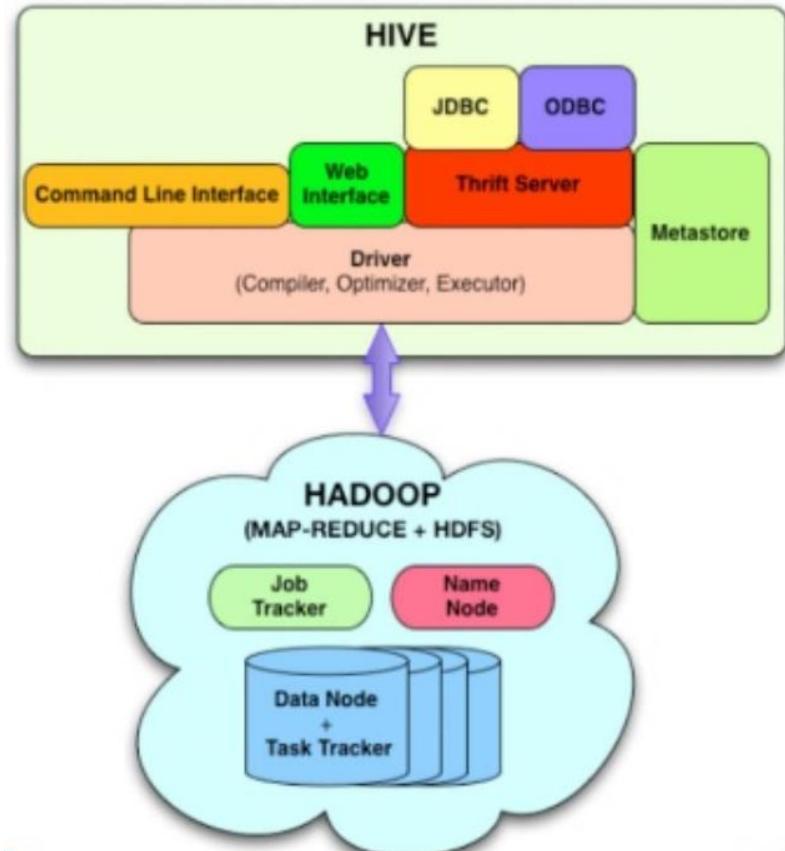


- O Hive armazena os dados no HDFS.
- Principal tarefa: compilar consultas SQL em jobs MR e executar em cluster.



Apache Hive

- Client API para executar sentenças HiveQL (Thrift Server).
- Metastore – catálogo do sistema.
- API – JDBC e ODBC.
- Linha de comando.
- Web Interface.



- Modelo de dados do Hive:
- Tables:
 - Tipos de colunas: int, float, String, date e boolean.
 - Suporta array, struct para JSON.
- Partitions:
 - Fragmentos de uma tabela.



- Metastore:
 - Database: namespace contendo um conjunto de tabelas.
 - Tabela: contém uma lista de colunas e seus tipos.
- Partition:
 - Cada partição possui suas próprias colunas.
 - Informações de armazenamento.
 - São mapeadas para diretórios do HDFS.
- Estatísticas:
 - Informações sobre o banco de dados.



- Comandos Hive:
 - Show Tables.
 - Describe <table_name>.
 - create table estudante (idEstudante INT, nomeEstudante STRING)
 - Alter table <...>
 - Drop table <...>



- O Hive não faz qualquer transformação enquanto está carregando os dados em tabelas.
- Operação de carga é puramente uma operação de cópia, que move arquivos de dados para tabelas Hive.
- Hive utiliza o HiveQL (Hive Query Language).
- Load Data pode ser usado para popular tabelas.
- Funções: Max, Sum, Count, etc.
- Group By.



- O Hive não foi criado para aplicações de tempo real.
- Foi projetado para melhor desempenho analisando grandes quantidades de dados.
- Criação de painéis analíticos de B. I.



Conclusão

- Apresentação do Hive.
- Utilidade.

■ Próxima aula

- HBase.



Aula 6.2. Apache HBase

Nesta aula

- Apresentação do Apache HBase.
- Aplicação, utilidade e vantagens do uso.

- Banco de dados não-relacional.
- Distribuído e open-source.
- Desenvolvido a partir do Google BigTable.
- Escrito em Java.
- Orientado a colunas (família de colunas).
- Executa sobre o HDFS.
- Capacidade de armazenamento de grandes volumes de dados esparsos.

Apache HBase



- Da perspectiva do usuário o HBase é similar a um banco de dados ou uma planilha.
- Existem linhas e colunas armazenando valores.

	A	B	C	D	E
1					
2					
3					
4					
5					
6					
7					

Apache HBase

IGTI

- O HBase pode ter um schema diferente por linha.
- Chamado Schema-less.
- O acesso é pela chave da linha e o nome da coluna.

	col-A	col-B	col-Foo	col-XYZ	foobar
row-1					
	col-A	col-D	col-Foo2	col-XYZ	col-XYZ2
row-10					
	20130423	20130424	20130425	20130426	20130427
row-18					
	MaxVal - ts5	MaxVal - ts4	MaxVal - ts3	MaxVal - ts2	MaxVal - ts1
row-2					

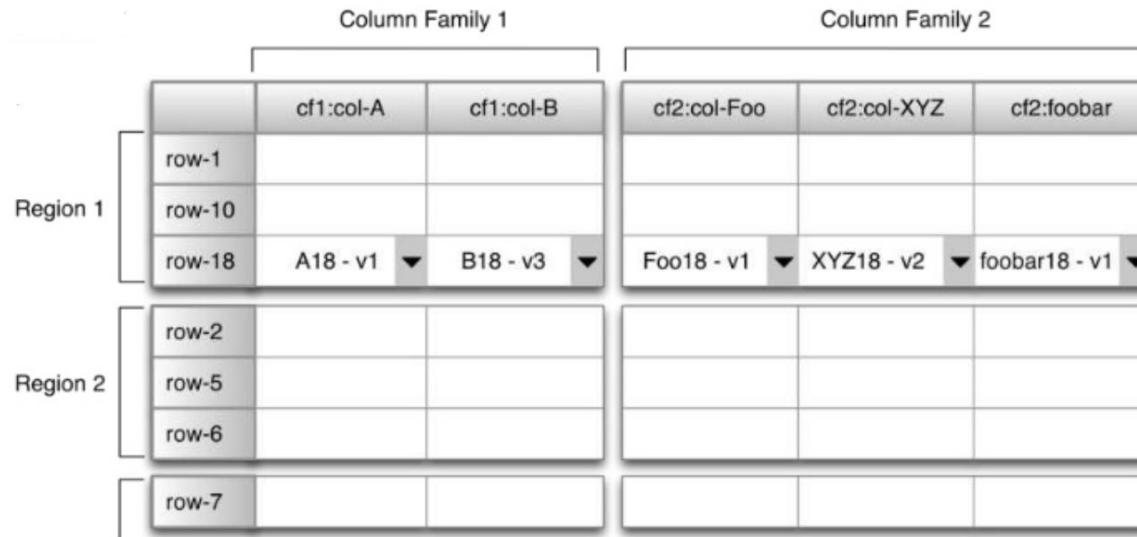


- Cada conjunto de linha/coluna é marcado com um número de versão.
- Permite valores multi-versionados.
- A API permite ao usuário buscar por versões.

	col-A	col-B	col-Foo	col-XYZ	foobar
row-1					
row-10					
row-18	A18 - v1 ▼	B18 - v3 ▼	Foo18 - v1 ▼	XYZ18 - v2 ▼	foobar18 - v1 ▼
row-2		Peter - v2 Bob - v1		Mary - v1	
row-5					
row-6					
row-7					

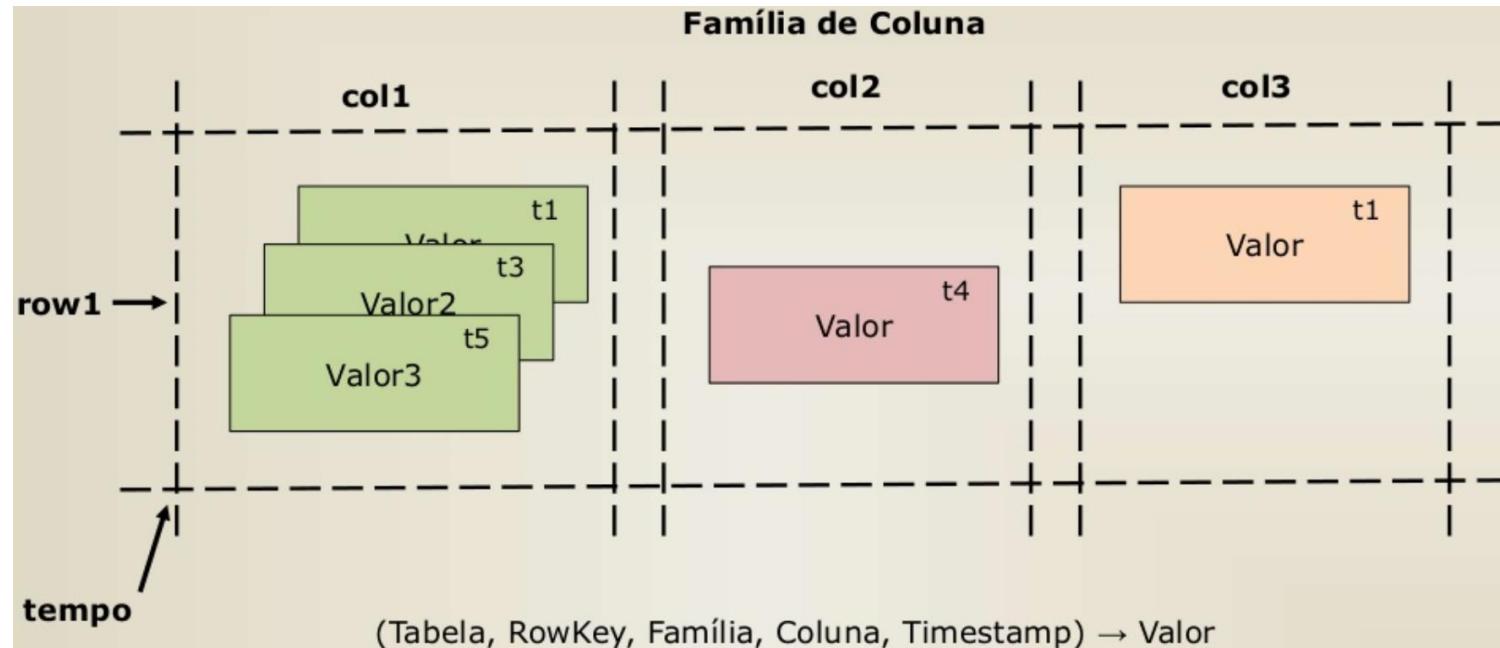
Apache HBase

- Os dados da tabela são partidos em pedaços e distribuídos sobre o cluster.
- Regiões dividem a tabela em fragmentos de tamanho fixo.
- Famílias são divididas dentro de **regiões** para agrupar conjuntos de colunas.



Apache HBase

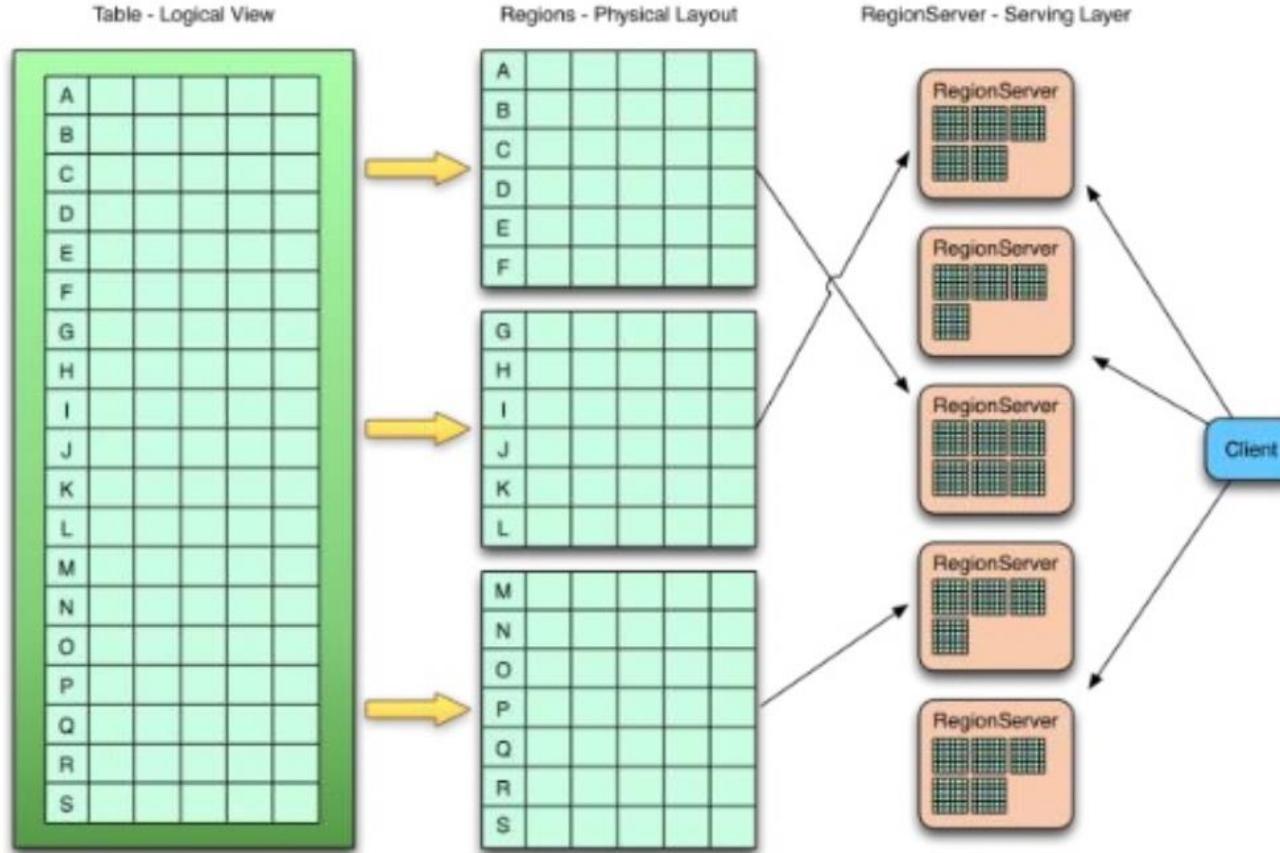
IGTI



- Uma tabela do HBase é formada por um conjunto de regiões (regions).
- Regiões são a unidade básica de trabalho do HBase.
- São compostas por faixas contínuas de RowKeys.
- As regiões possuem objetos de armazenamento que correspondem a famílias de colunas.

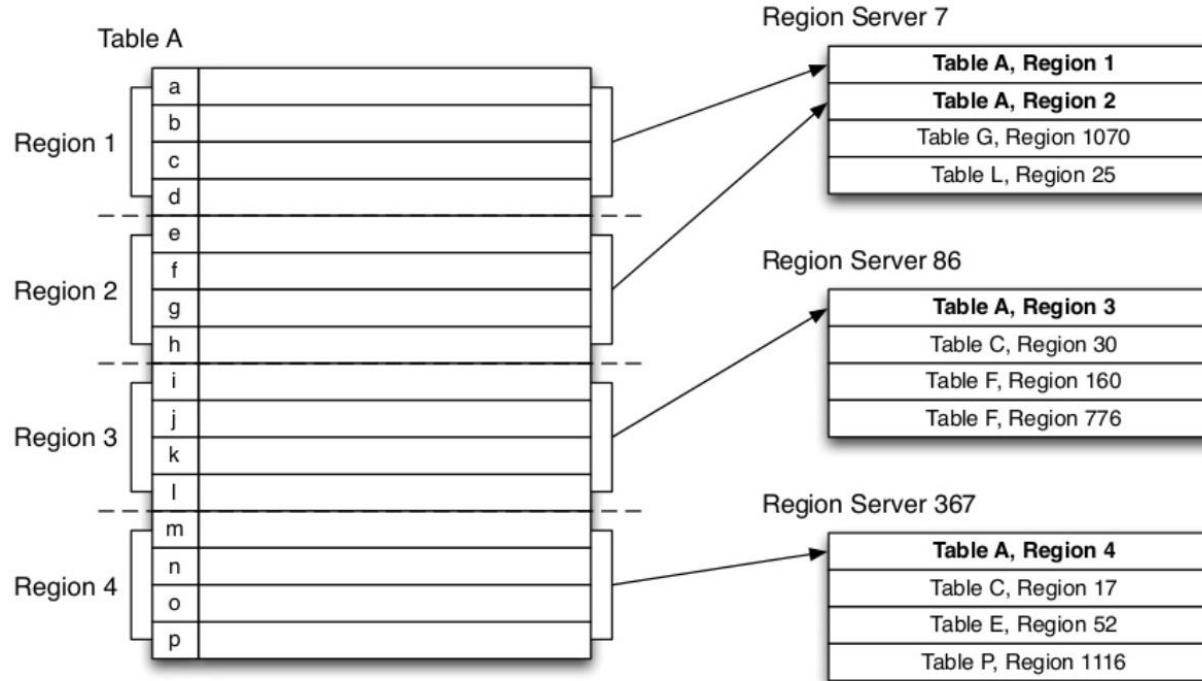
- Uma região é servida por exatamente uma Region Server.
- Cada Region Server atende muitas regiões.
- Os dados da tabela são divididos para os servers.
- Clientes buscam dados diretamente da Region Server.

Apache HBase



Apache HBase

IGTI



Legend:

- A single table is partitioned into Regions of roughly equal size.
- Regions are assigned to Region Servers across the cluster.
- Region Servers host roughly the same number of regions.

- HBase é bom para os seguintes conjuntos de dados:
 - Grandes.
 - Esparsos.
 - Com registros sem normalização.
 - Com muitos clientes (concorrentes).
- HBase deve ser evitado para:
 - Pequenos conjuntos de dados.
 - Registros altamente relacionados.

Apache HBase

- Tratar o HBase como um sistema tradicional de banco de dados, poderá levá-lo ao fracasso.
- HBase segue a filosofia de “escreva uma vez e leia muitas”.

Característica	SGBD	HBase
Transações	Sim	Apenas no nível de linha
Linguagem de consulta	SQL	Get/Put/Scan
Segurança	Autenticação/Autorização	Kerberos
Max Data Size	TB	PB+
Límites de Throughput de escrita / leitura	1000 por segundo	Milhões por segundo
Max Query Size	TB	PB

- Formas de acesso:
 - HBase Shell.
 - API Java e Rest.
 - Integração com outros elementos do ecossistema Hadoop:
 - Spark.
 - Hive.
 - Pig.

Conclusão

Apresentação do HBase.

Características.

■ Próxima aula

- Apache Sqoop.



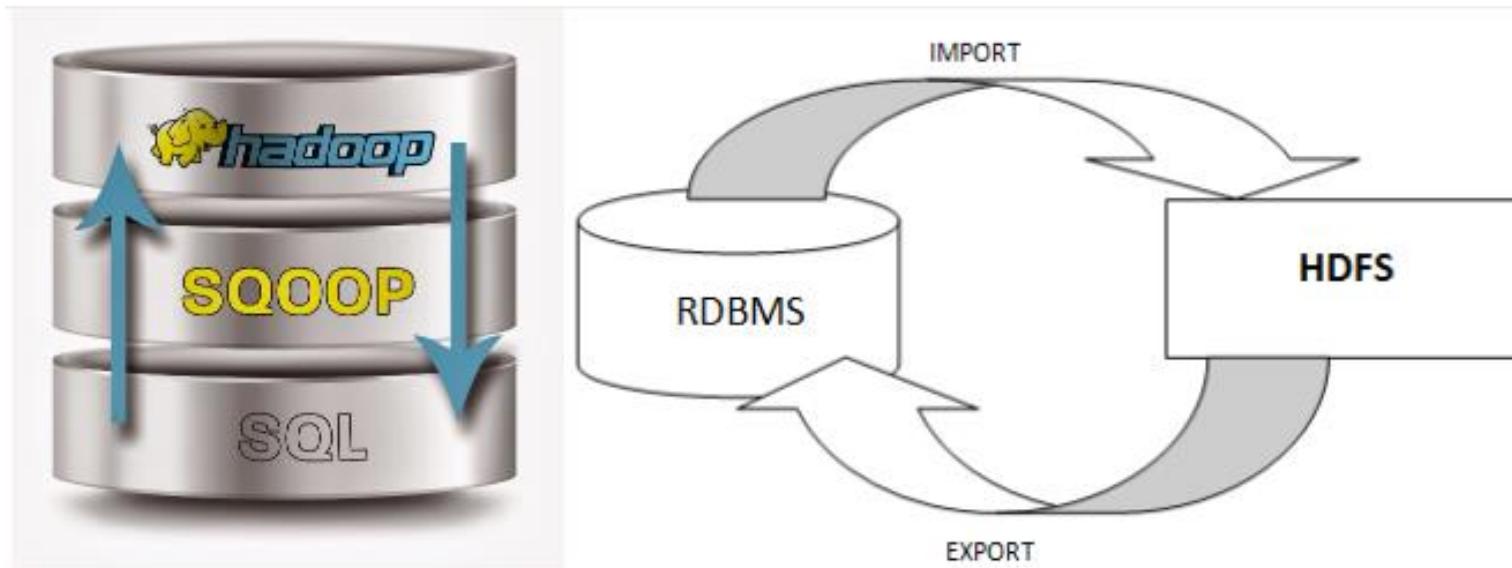
Aula 6.3. Apache Sqoop

- ❑ Apresentação do Apache Sqoop.
- ❑ Características e funcionamento.

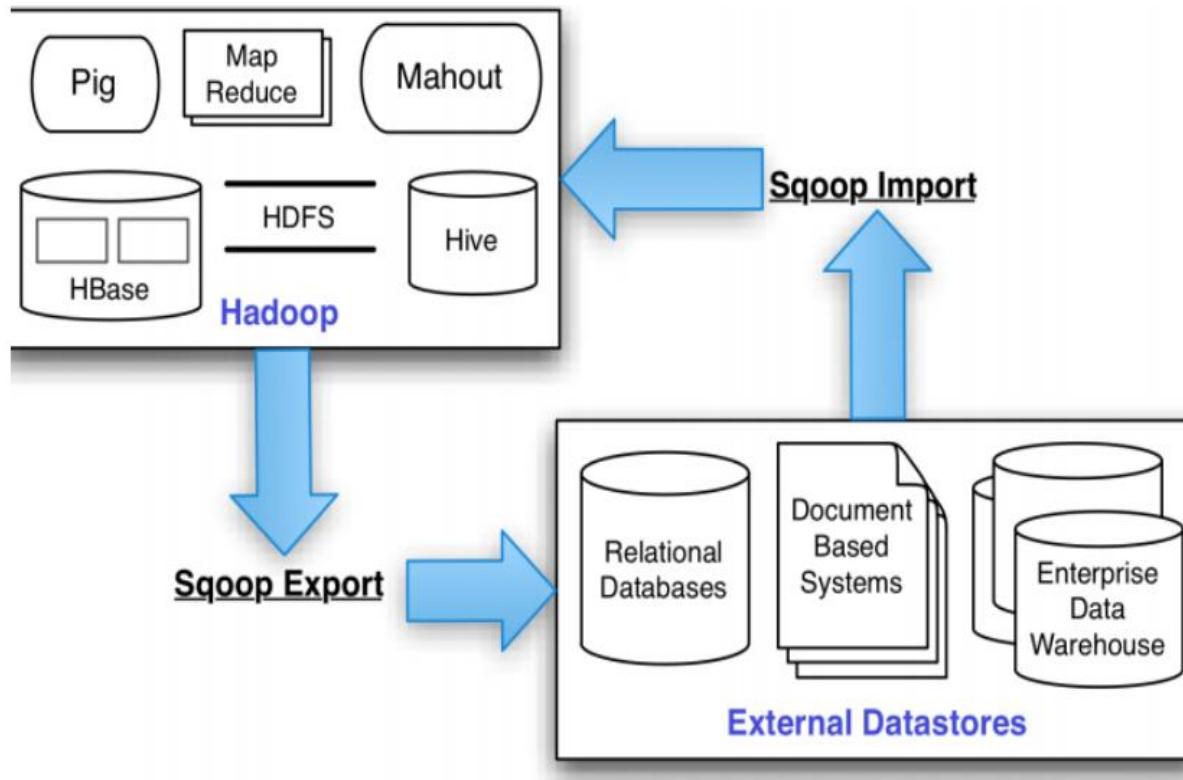
- SQL-to-Hadoop (abreviação):
 - Ferramenta de linha de comando.
 - Desenvolvida para transferir dados entre cluster Hadoop e banco de dados de armazenamento estruturado – relacionais (Oracle, DB2, MySQL, etc.).
 - Projeto iniciado em 2009 pela Cloudera.
 - Virou top-level na ASF (Apache Software Foundation) em 2012.



Apache Sqoop



Apache Sqoop



Apache Swoop



- Ferramenta para ingestão de dados.
- Necessidade de mover/transferir grandes quantidades de dados.
- Transferir dados por script é ineficiente e consome muito tempo.
- Existe a possibilidade de transformar dados.

- O Sqoop permite exportar e importar facilmente dados de armazenamentos estruturados.
- Possibilidade de transferir dados e armazenar no HDFS.
- Povoar tabelas Hive e HBase.
- Executa em cluster Hadoop.

- Realiza a conversão de tipos de campos.
- Aceita a conexão com diversos bancos, via JDBC.
- Utiliza o MapReduce nas atividades de importação e exportação, fornecendo processamento paralelo, distribuído e tolerante a falhas.
- Faz a leitura linha por linha da tabela ao escrever o arquivo no HDFS.
- MySQL, PostGreSQL, Oracle, SQL Server e DB2.
- Possui um conector JDBC genérico.

- Fornece modo de importação incremental.
- Pode ser usado para recuperar apenas novas linhas.
- Sqoop fornece dois tipos de importação incremental:
 - **Append:**
 - Usado para novas linhas.
 - Especifica-se uma coluna contendo um ID (check column).
 - O Sqoop importará linhas onde a check column tem um valor maior que o valor especificado em last value.

- **Last Modified:**

- Pode ser usado quando as linhas da tabela fonte precisam ser atualizados na tabela destino.
- Atualiza um timestamp.
- Colunas no destino que possuem timestamp mais antigo, que na tabela origem serão atualizadas.

Conclusão

- Apresentação do Sqoop.
- Ferramenta ETL.
- Movimentação de grandes quantidades de dados.