

Aula 05 – Atividade prática MapReduce e YARN

Business Intelligence com Ênfase em Big Data, UniRuy | Wyden

Disciplina: Processamento Massivo Paralelo Hadoop e MapReduce

Profa. Gabriela Mota

gbrlamota@gmail.com

Agenda

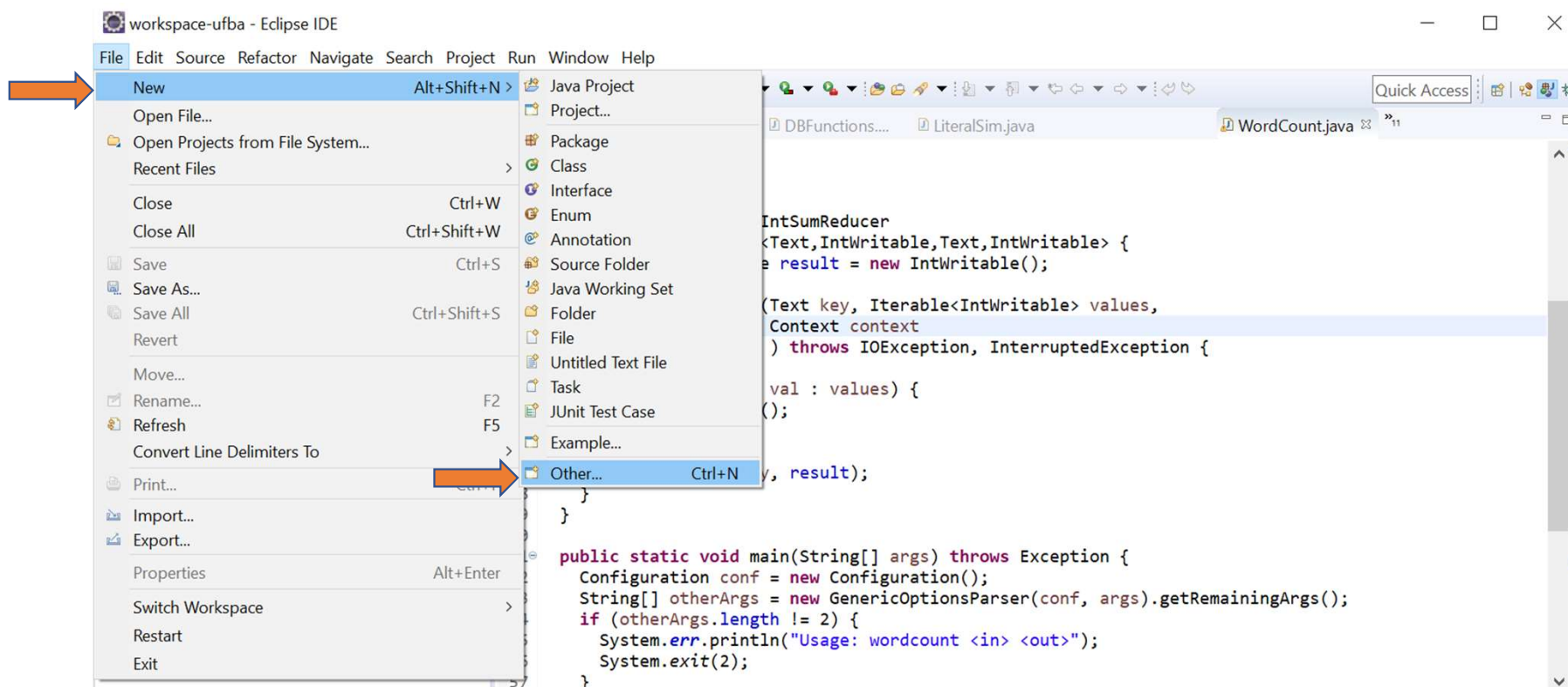
- Desenvolvimento da Atividade 02 – WordCount
- Alterando a Atividade 02 – WordCount personalizado
- Atividade 03 – Lista de Exercícios Hadoop + Pig
- Material Complementar
- Referências

Desenvolvimento da Atividade 02

- WordCount

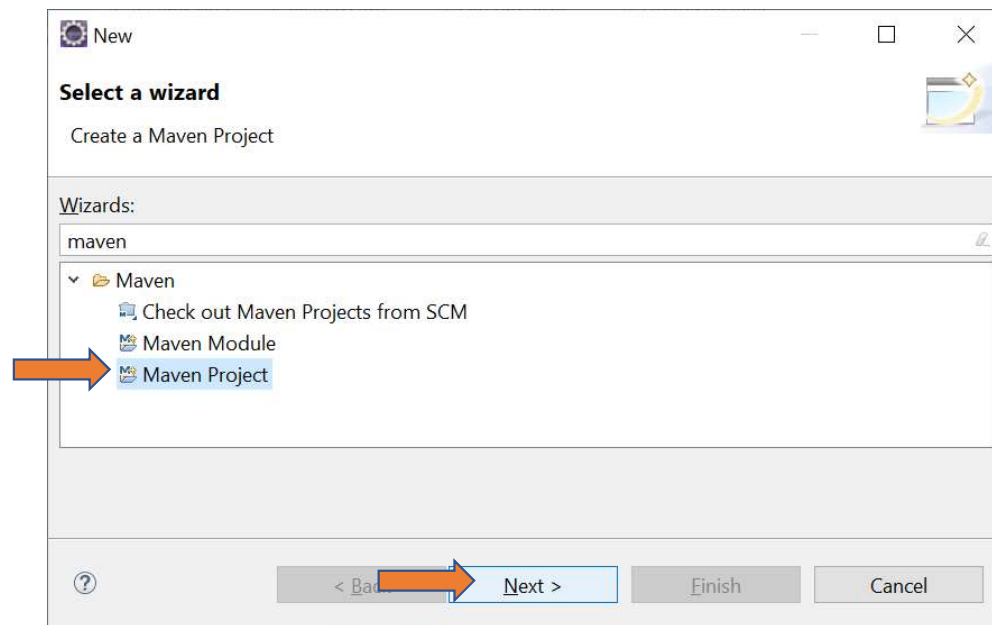
Compilando o projeto

- Crie um projeto Maven no Eclipse



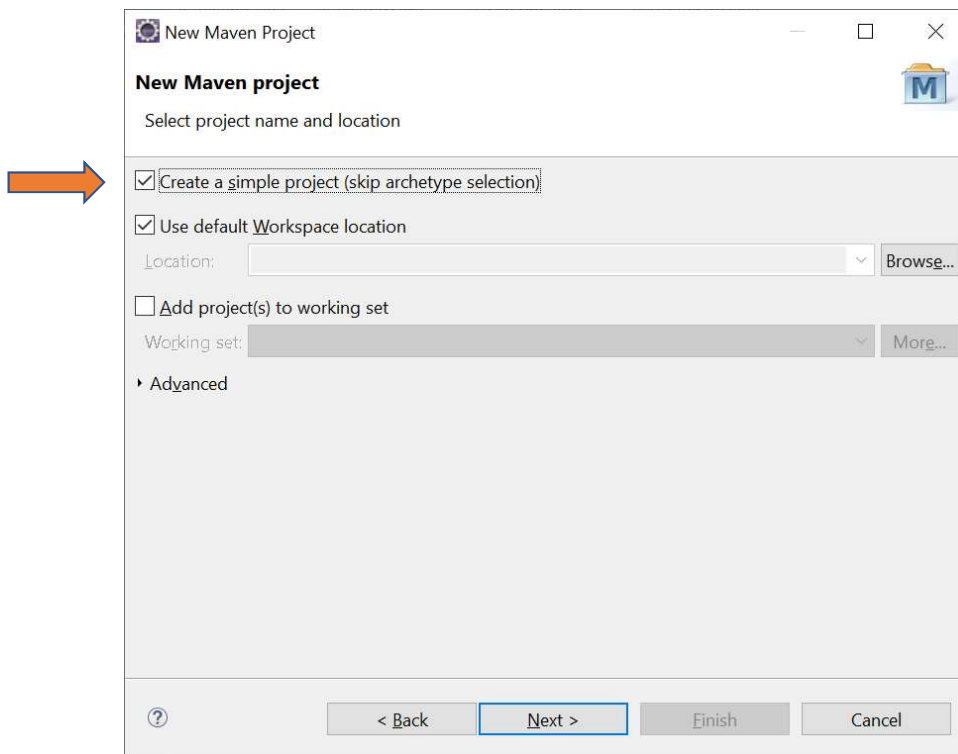
Compilando o projeto

- Crie um projeto Maven no Eclipse



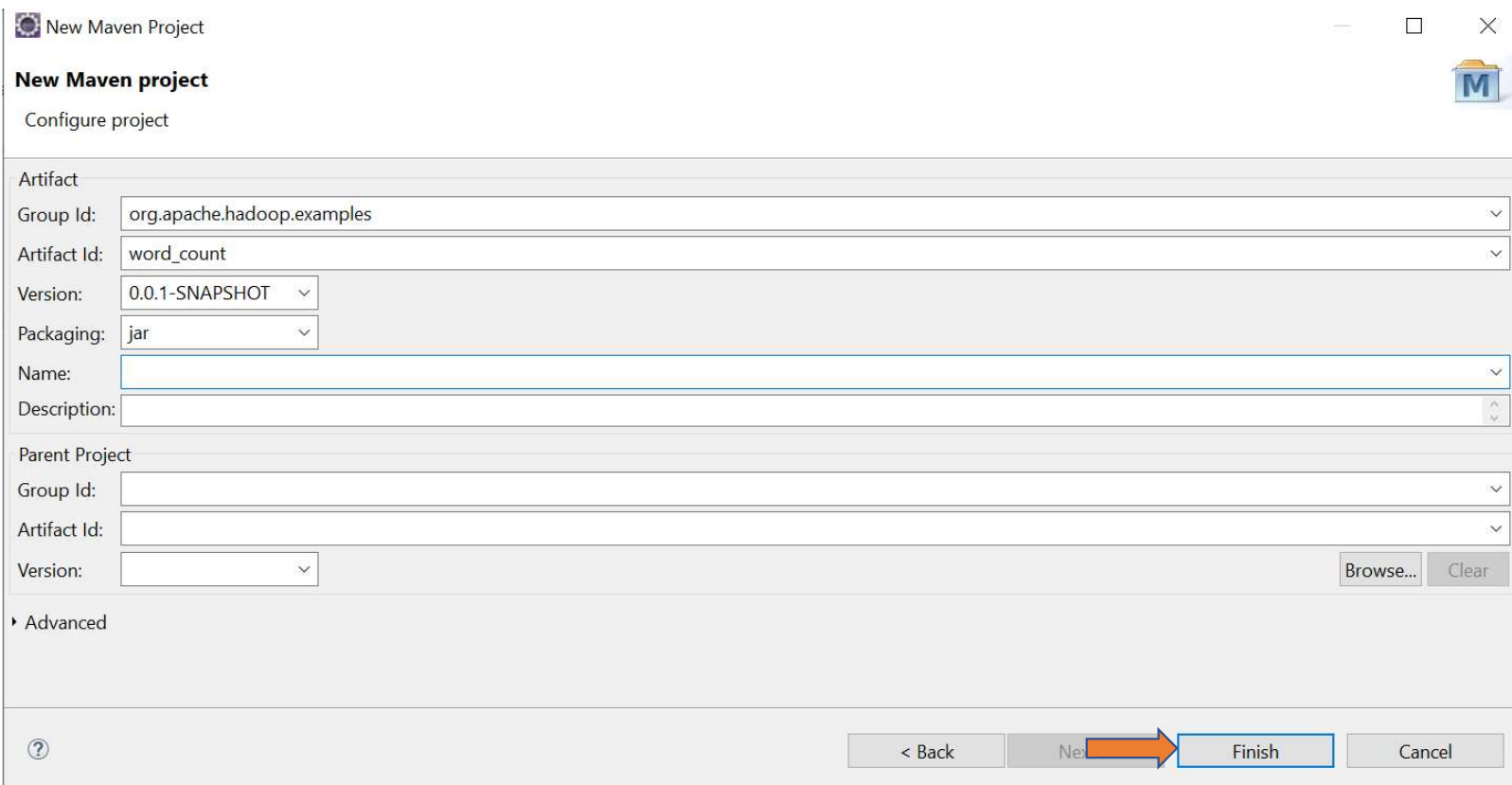
Compilando o projeto

- Crie um projeto Maven no Eclipse



Compilando o projeto

- Crie um projeto Maven no Eclipse



New Maven Project

Configure project

Artifact

Group Id:

Artifact Id:

Version:

Packaging:

Name:

Description:

Parent Project

Group Id:

Artifact Id:

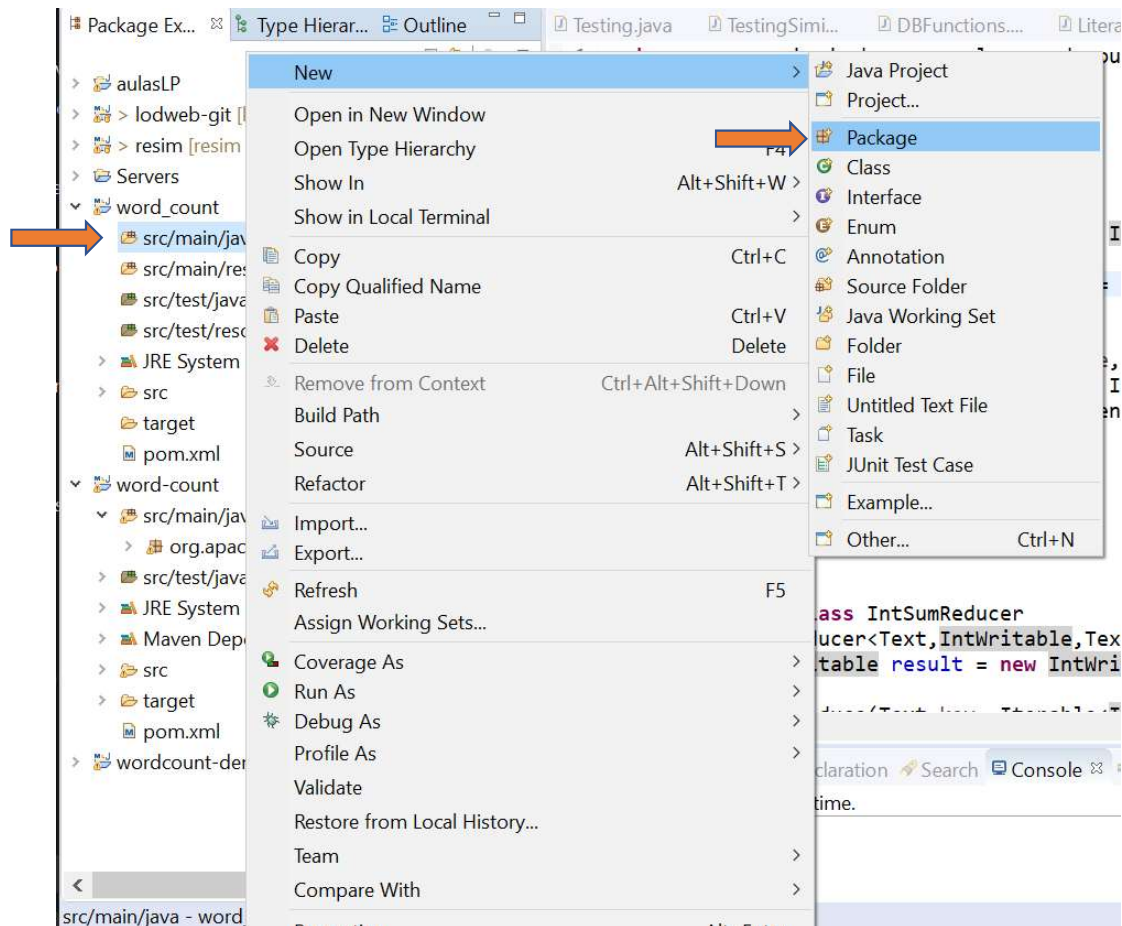
Version:

Advanced

Compilando o projeto

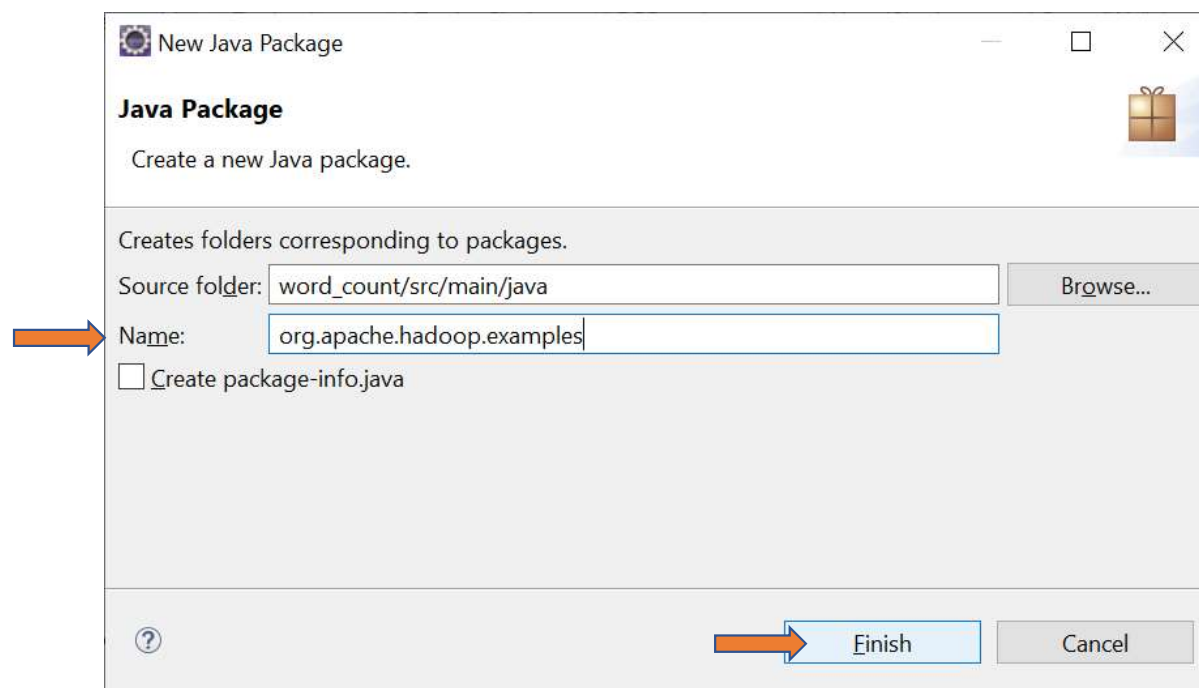
- Crie a hierarquia de pacotes para abrigar o código-fonte

Botão direito em
src/main/java



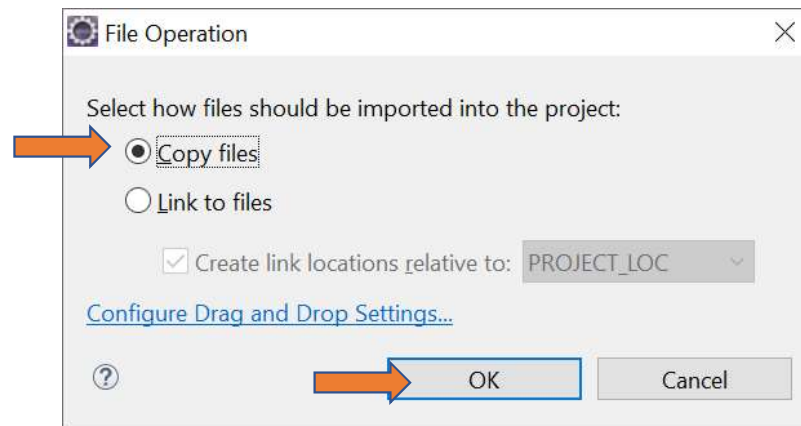
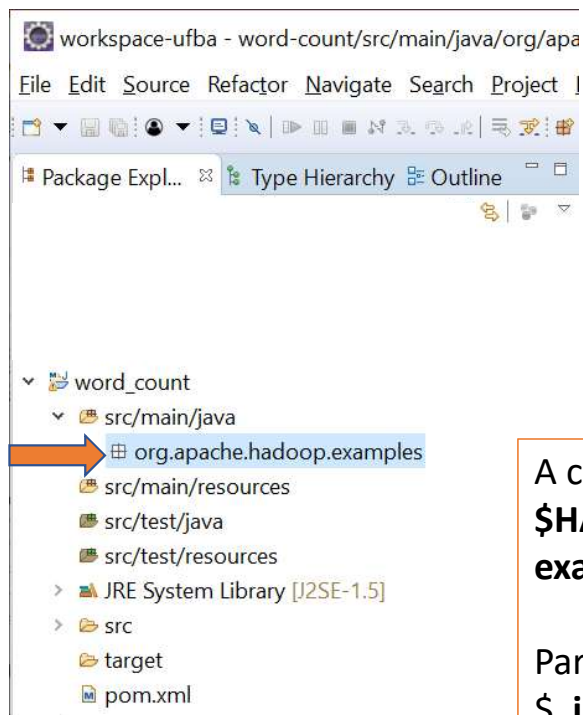
Compilando o projeto

- Crie a hierarquia de pacotes para abrigar o código-fonte



Compilando o projeto

- Arraste a classe WordCount para dentro do pacote criado



A classe WordCount.java vem com a instalação do haddop, dentro do diretório:
\$HADOOP_HOME/share/hadoop/mapreduce/sources/hadoop-mapreduce-examples-2.9.2-sources.jar

Para descompactar:

\$ jar -xf hadoop-mapreduce-examples-2.9.2-sources.jar hadoop-mapreduce-examples

workspace-ufba - word_count/src/main/java/org/apache/hadoop/examples/WordCount.java - Eclipse IDE

File Edit Source Package Expl.

Agora vamos adicionar as dependências do projeto ao arquivo de configuração do Maven para que o Eclipse baixe as bibliotecas necessárias e esses erros no código desapareçam.

Package Expl.

- aulasLP
- > lodweb-git [lodweb-git master]
- > resim [resim master]
- Servers
- word_count
 - src/main/java
 - org.apache.hadoop.examples
 - WordCount.java
 - src/main/resources
 - src/test/java
 - src/test/resources
 - JRE System Library [J2SE-1.5]
 - src
 - target
 - pom.xml
 - word-count
 - src/main/java
 - org.apache.hadoop.examples.word_count
 - WordCount.java
 - src/test/java
 - JRE System Library [J2SE-1.5]
 - Maven Dependencies
 - src
 - target
 - pom.xml
 - wordcount-demo

```
18 package org.apache.hadoop.examples;
19
20 import java.io.IOException;
33
34 public class WordCount {
35
36     public static class TokenizerMapper
37         extends Mapper<Object, Text, Text, IntWritable>{
38
39         private final static IntWritable one = new IntWritable(1);
40         private Text word = new Text();
41
42     public void map(Object key, Text value, Context context
43         ) throws IOException, InterruptedException {
44         StringTokenizer itr = new StringTokenizer(value.toString());
45         while (itr.hasMoreTokens()) {
46             word.set(itr.nextToken());
47             context.write(word, one);
48         }
49     }
50 }
51
52 public static class IntSumReducer
53     extends Reducer<Text, IntWritable, Text, IntWritable> {
54     private IntWritable result = new IntWritable();
55 }
```

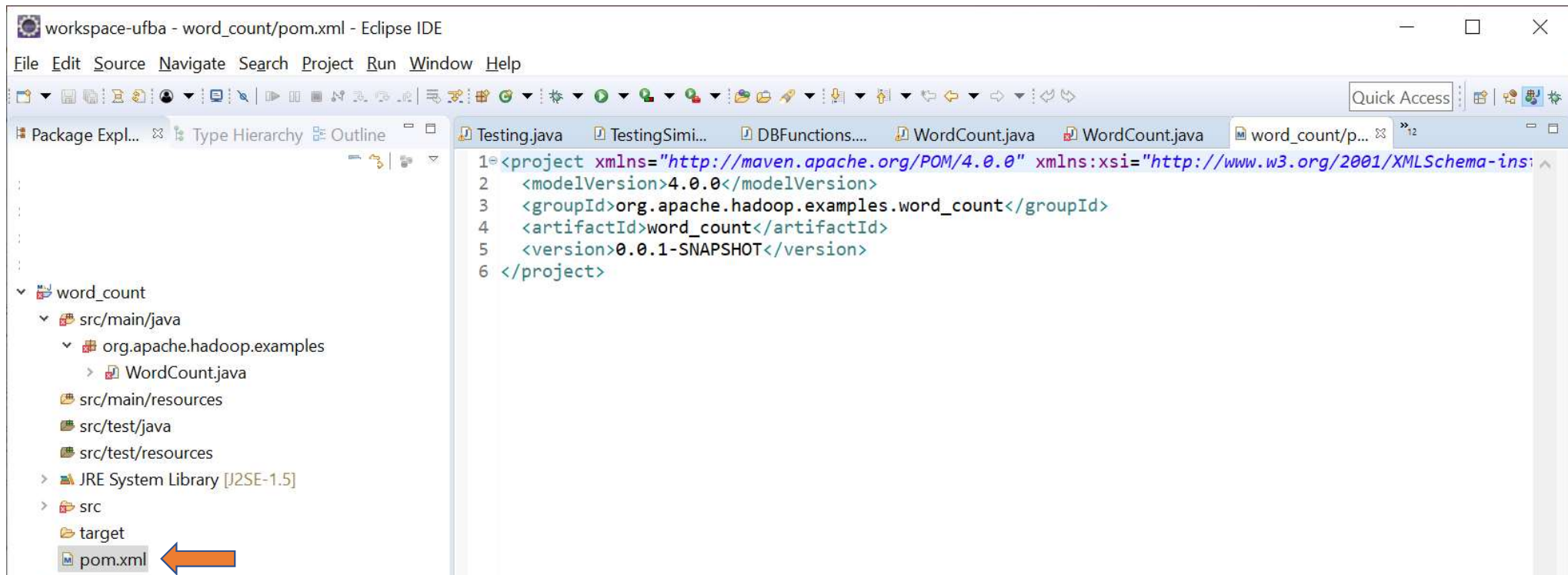
Problems Javadoc Declaration Search Console Progress Debug

No consoles to display at this time.

org.apache.hadoop.examples.WordCount.java - word_count/src/main/java

Compilando o projeto

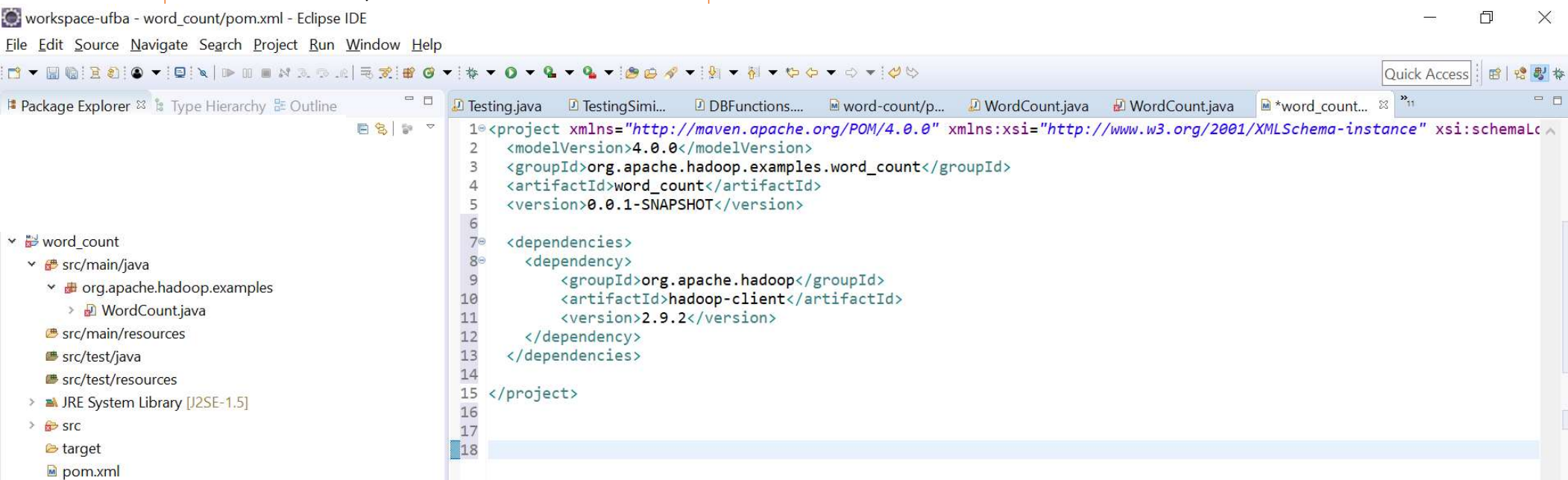
- Dê dois cliques no arquivo pom.xml



Compilando o projeto

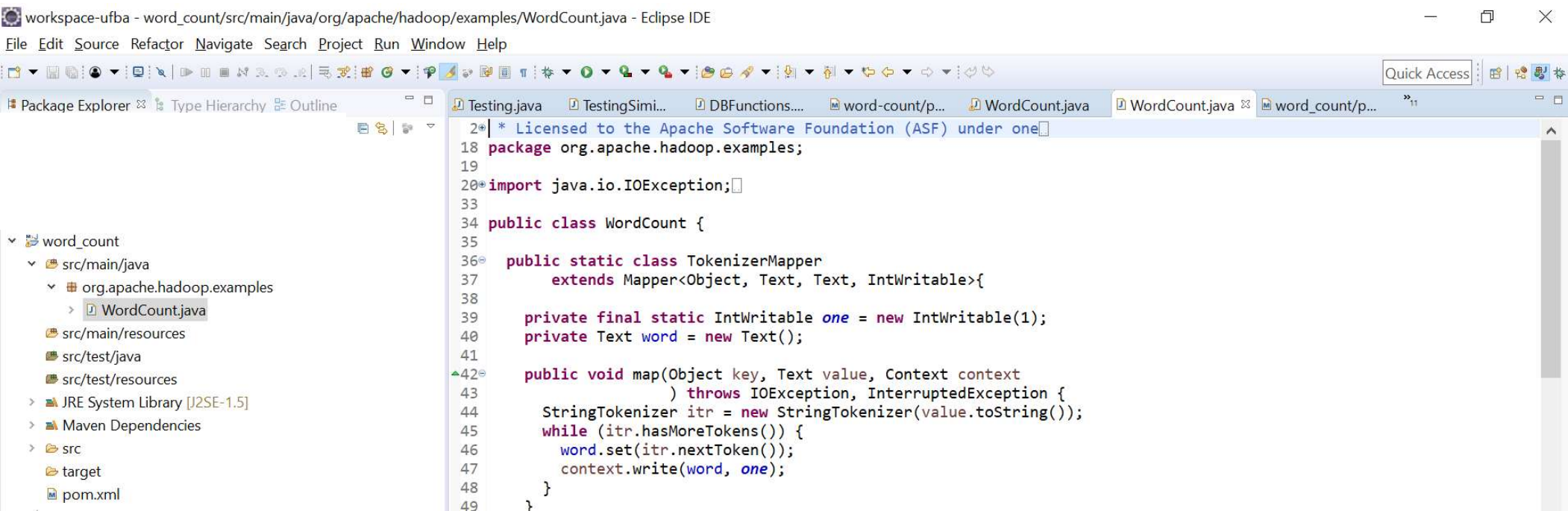
- Acrescente as seguintes configurações dentro da tag <project></Project

```
<dependencies>
  <dependency>
    <groupId>org.apache.hadoop</groupId>
    <artifactId>hadoop-client</artifactId>
    <version>2.9.2</version>
```



Compilando o projeto

- Salve o arquivo pom.xml e aguarde a execução do build do projeto
- Veja que os erros no código da classe WordConunt.java sumiram



```
workspace-ufba - word_count/src/main/java/org/apache/hadoop/examples/WordCount.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

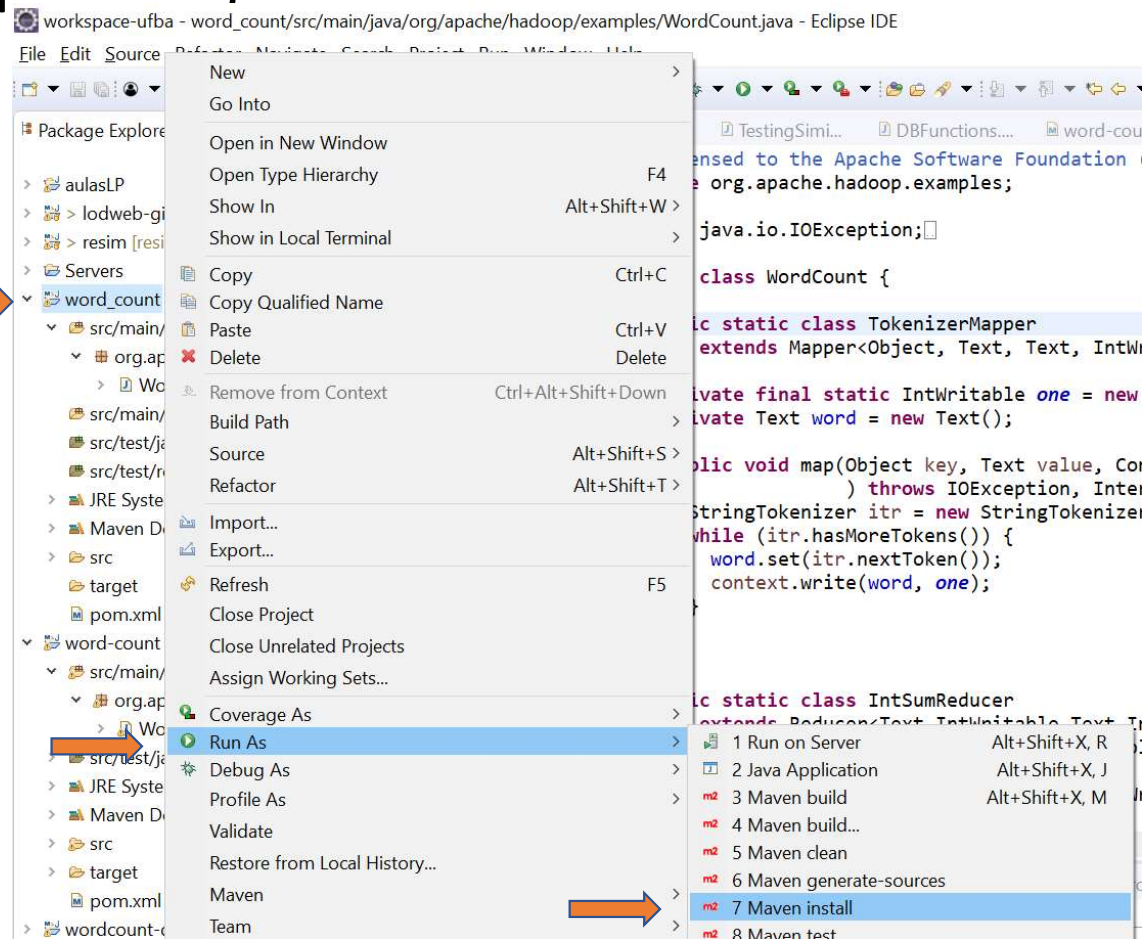
Package Explorer | Type Hierarchy | Outline | Testing.java | TestingSimi... | DBFunctions.... | word-count/p... | WordCount.java | WordCount.java | word_count/p...

word_count
├── src/main/java
│   └── org.apache.hadoop.examples
│       └── WordCount.java
├── src/main/resources
├── src/test/java
├── src/test/resources
├── JRE System Library [J2SE-1.5]
├── Maven Dependencies
├── src
├── target
└── pom.xml

2 * Licensed to the Apache Software Foundation (ASF) under one
18 package org.apache.hadoop.examples;
19
20 import java.io.IOException;
33
34 public class WordCount {
35
36     public static class TokenizerMapper
37         extends Mapper<Object, Text, Text, IntWritable>{
38
39         private final static IntWritable one = new IntWritable(1);
40         private Text word = new Text();
41
42     public void map(Object key, Text value, Context context
43         ) throws IOException, InterruptedException {
44         StringTokenizer itr = new StringTokenizer(value.toString());
45         while (itr.hasMoreTokens()) {
46             word.set(itr.nextToken());
47             context.write(word, one);
48         }
49     }
}
```

Criando o arquivo .jar

Botão direito na
pasta raiz do
projeto



Criando o arquivo .jar

Botão direito na pasta raiz do projeto



workspace-ufba - word_count/src/main/java/org/apache/hadoop/examples/WordCount.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

New
Go Into
Open in New Window
Open Type Hierarchy F4
Show In Alt+Shift+W
Show in Local Terminal
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete Delete
Remove from Context Ctrl+Alt+Shift+Down
Build Path
Source Alt+Shift+S
Refactor Alt+Shift+T
Import...
Export...
Refresh F5
Close Project
Close Unrelated Projects
Assign Working Sets...
Coverage As
Run As
Debug As
Profile As
Validate
Restore from Local History...
Maven
Team

word_count/p... WordCount.java WordCount.java word_count/p...

Licensed to the Apache Software Foundation (ASF) under one
package org.apache.hadoop.examples;

```
import java.io.IOException;
```

```
public class WordCount {
```

```
    public static class TokenizerMapper
```

```
        extends Mapper<Object, Text, Text, IntWritable>{
```

```
        private final static IntWritable one = new IntWritable(1);
```

```
        private Text word = new Text();
```

```
        public void map(Object key, Text value, Context context)
```

Acompanhe o build do .jar pelo log do Maven

```
C:\Program Files\Java\jdk1.8.0_191\bin\javaw.exe (Mar 20, 2020, 12:52:50 AM)
--- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ word_count ---
Nothing to compile - all classes are up to date

--- maven-surefire-plugin:2.12.4:test (default-test) @ word_count ---

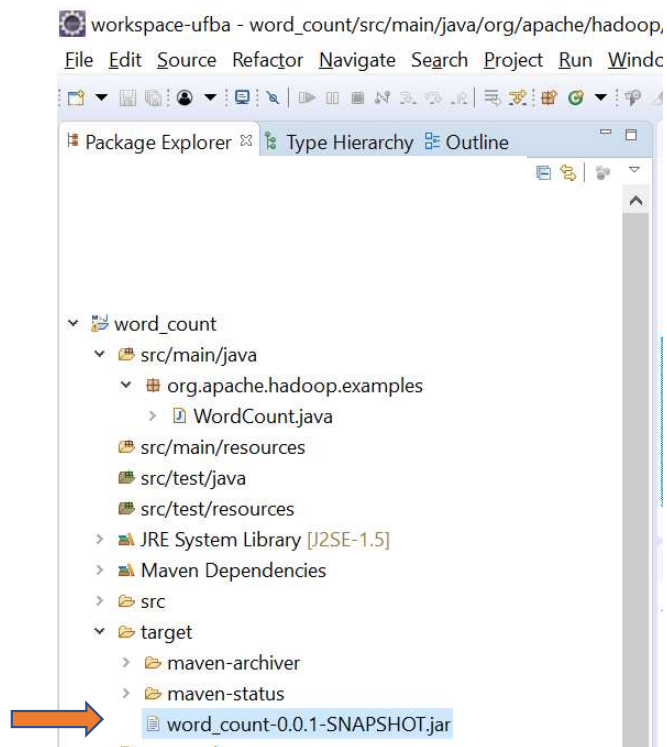
--- maven-jar-plugin:2.4:jar (default-jar) @ word_count ---
Building jar: C:\Users\gbrla\Google Drive\UFBA\Doutorado\workspace-ufba\word_count\target\word_count-0.0.1-SNAPSHOT.jar

--- maven-install-plugin:2.4:install (default-install) @ word_count ---
Installing C:\Users\gbrla\Google Drive\UFBA\Doutorado\workspace-ufba\word_count\target\word_count-0.0.1-SNAPSHOT.jar to C:\Users\gbrla\.m2\repository\org\apache\hadoop\examples\word_count-0.0.1-SNAPSHOT.jar
Installing C:\Users\gbrla\Google Drive\UFBA\Doutorado\workspace-ufba\word_count\pom.xml to C:\Users\gbrla\.m2\repository\org\apache\hadoop\examples\word_count-0.0.1-SNAPSHOT.pom

BUILD SUCCESS

Total time: 9.142 s
Finished at: 2020-03-20T00:53:02-03:00
```


Criando o arquivo .jar



Classe *Combiner*
O arquivo .jar vai
aparecer na pasta
target

Atenção!

- Caso você não queira usar o Eclipse ou outra IDE Java, é possível compilar e gerar o .jar diretamente pela linha de comando do Linux
- Seguir esse tutorial
 - <https://hashnode.com/post/how-i-was-finally-able-to-run-the-infamous-word-count-example-on-hadoop-ciwazrq8u000vgw53qe4saran>

Analizando o código MapReduce

- Classe *Mapper*

- <https://hadoop.apache.org/docs/r2.7.4/api/org/apache/hadoop/mapreduce/Mapper.html>

Sobrescrever esse método, colocando a lógica desejada. Ele é chamado uma vez para cada par de chave / valor no *input split*.

Escrevendo os resultados intermediários.

```
36 public static class TokenizerMapper
37     extends Mapper<Object, Text, Text, IntWritable>{
38
39     private final static IntWritable one = new IntWritable(1);
40     private Text word = new Text();
41
42     public void map(Object key, Text value, Context context
43                     ) throws IOException, InterruptedException {
44         StringTokenizer itr = new StringTokenizer(value.toString());
45         while (itr.hasMoreTokens()) {
46             word.set(itr.nextToken());
47             context.write(word, one);
48         }
49     }
50 }
```

Analizando o código MapReduce

- Classe *Reducer*

- <https://hadoop.apache.org/docs/r2.7.4/api/org/apache/hadoop/mapreduce/Reducer.html>

Sobrescrever esse método, colocando a lógica desejada. Ele é chamado para cada <key, (coleção de valores)> nos *inputs* classificados por chave.

Escrevendo os resultados finais.

```
52 public static class IntSumReducer
53     extends Reducer<Text,IntWritable,Text,IntWritable> {
54     private IntWritable result = new IntWritable();
55
56     public void reduce(Text key, Iterable<IntWritable> values,
57                        Context context
58                        ) throws IOException, InterruptedException {
59         int sum = 0;
60         for (IntWritable val : values) {
61             sum += val.get();
62         }
63         result.set(sum);
64         context.write(key, result);
65     }
66 }
```

Analizando o código MapReduce

- Classe *Reducer*: possui 3 fases
 - *Shuffle*
 - O *Reducer* distribui a saída classificada por chave de cada *Mapper* usando HTTP na rede
 - *Sort*
 - O framework ordena as entradas do *Reducer* por chaves (uma vez que diferentes Mappers podem ter gerado a mesma chave)
 - As fases de *shuffle* e *sort* ocorrem simultaneamente, ou seja, enquanto as saídas estão sendo recuperadas, são também mescladas
 - *Reduce*
 - Reduz um conjunto de valores intermediários que compartilham uma chave para um conjunto menor de valores
 - A saída da tarefa *reduce* geralmente é gravada em um `RecordWriter` via `TaskInputOutputContext.write (Object, Object)`
 - Se você não implementar o *Reducer*, a saída do *Mapper* será escrita diretamente para a classe [OutputFormat](#) sem classificar por chaves
 - A saída do *Reducer* não é classificada novamente

Analizando o código MapReduce

- O programa principal (main)

Criando o Job a partir dos parâmetros básicos de configuração

Setando os parâmetros específicos das tarefas de map e reduce

```
68 public static void main(String[] args) throws Exception {
69     Configuration conf = new Configuration();
70     String[] otherArgs = new GenericOptionsParser(conf, args).getRemainingArgs();
71     if (otherArgs.length < 2) {
72         System.err.println("Usage: wordcount <in> [<in>...] <out>");
73         System.exit(2);
74     }
75     Job job = Job.getInstance(conf, "word count");
76     job.setJarByClass(WordCount.class);
77     job.setMapperClass(TokenizerMapper.class);
78     job.setCombinerClass(IntSumReducer.class);
79     job.setReducerClass(IntSumReducer.class);
80     job.setOutputKeyClass(Text.class);
81     job.setOutputValueClass(IntWritable.class);
82     for (int i = 0; i < otherArgs.length - 1; ++i) {
83         FileInputFormat.addInputPath(job, new Path(otherArgs[i]));
84     }
85     FileOutputFormat.setOutputPath(job,
86         new Path(otherArgs[otherArgs.length - 1]));
87     System.exit(job.waitForCompletion(true) ? 0 : 1);
88 }
89 }
```

Executando o programa MapReduce

- # copiar o arquivo jar para a pasta home do usuário hadoopusr
 - \$ mv ./word-count-0.0.1-SNAPSHOT.jar /home/hadoopusr/
 - # entrar nessa pasta
 - \$ cd /home/hadoopusr/
 - # virar o usuário do hadoop
 - \$ sudo su hadoopusr
 - # criar o arquivo de input
 - \$ touch input.txt
 - # adicionar os animais ao arquivo de texto
 - \$ nano input.txt
- (Tiger Lion Lion Panther Wolf Tiger Tiger Wolf Panther)

Executando o programa MapReduce

- # criar um diretório no hdfs para colocar os arquivos de input e dar as permissões
- \$ hdfs dfs -mkdir -p input
- \$ hdfs dfs -ls
- \$ hdfs dfs -chmod 777 input

- # mover o arquivo criado para o hdfs
- \$ hdfs dfs -put input.txt input

- # verificar se o arquivo foi realmente movido
- \$ hdfs dfs -ls input

- # se a pasta output já tiver sido criada anteriormente, apagar antes de rodar o job
- \$ hdfs dfs -rm -r output

Executando o programa MapReduce

- # executar o jar
- \$ `hadoop jar /home/hadoopusr/word-count-0.0.1-SNAPSHOT.jar wordcount input output`
- # verificar o diretório de saída
- \$ `hdfs dfs -ls output`
- # para ler o arquivo de saída
- \$ `hdfs dfs -cat output/part-r-00000`
- Obs.: fique atentx às mensagens de log que aparecem no console à medida que o job executa.
- Para rastrear os erros, é fornecida uma url de acompanhamento
- Ex.: INFO mapreduce.Job: The url to track the job:
`http://localhost:8088/proxy/application_1584656219581_0002/`

Alterando a Atividade 02 – WordCount personalizado

WordCount personalizado

- Vamos contar somente os animais que são grandes felinos

```
SELECT COUNT(NAME) FROM animals  
WHERE name IN ("Tiger", "Lion", ...)   
GROUP BY name;
```

1. Escreva um código que execute esse filtro no método map da classe WordCount.java

Node 1

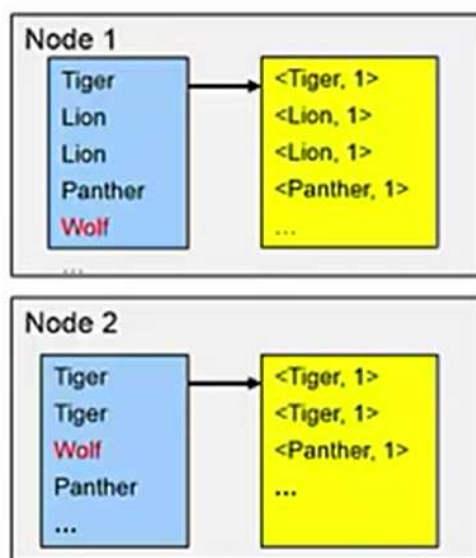
Tiger
Lion
Lion
Panther
Wolf
...

Node 2

Tiger
Tiger
Wolf
Panther
...

WordCount personalizado

- Duas tarefas de Map
 - Filtrar os animais que não são grandes felinos
 - Preparar o count, transformando os dados em **<Text(name), Integer(1)>**

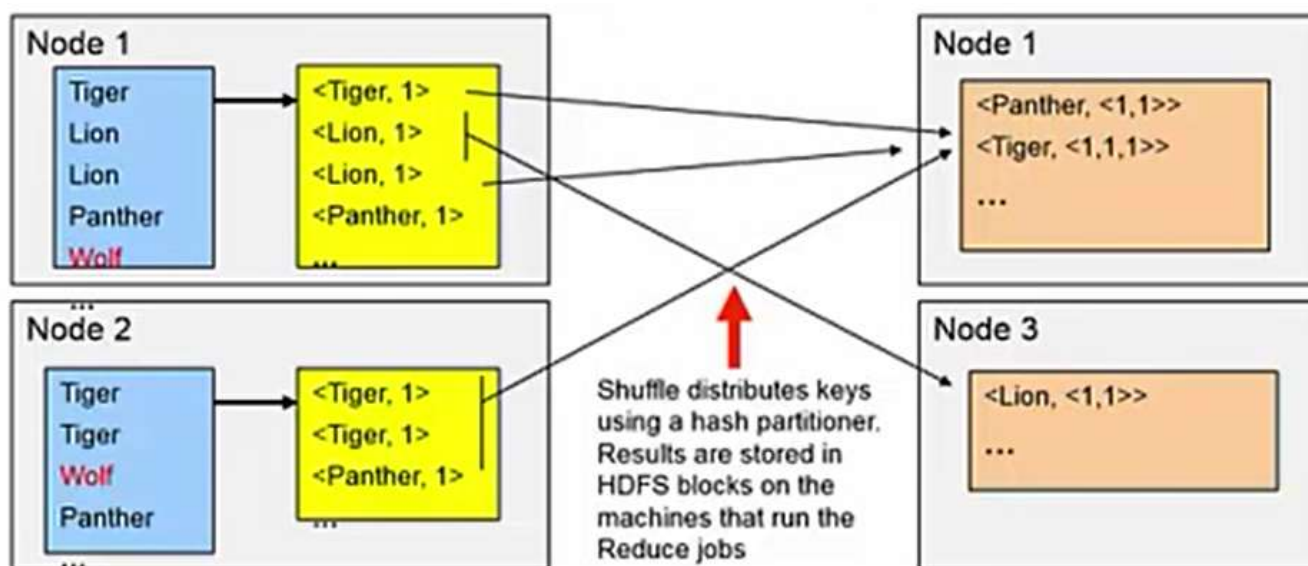


The Map Tasks
are executed
locally on each
split

WordCount personalizado

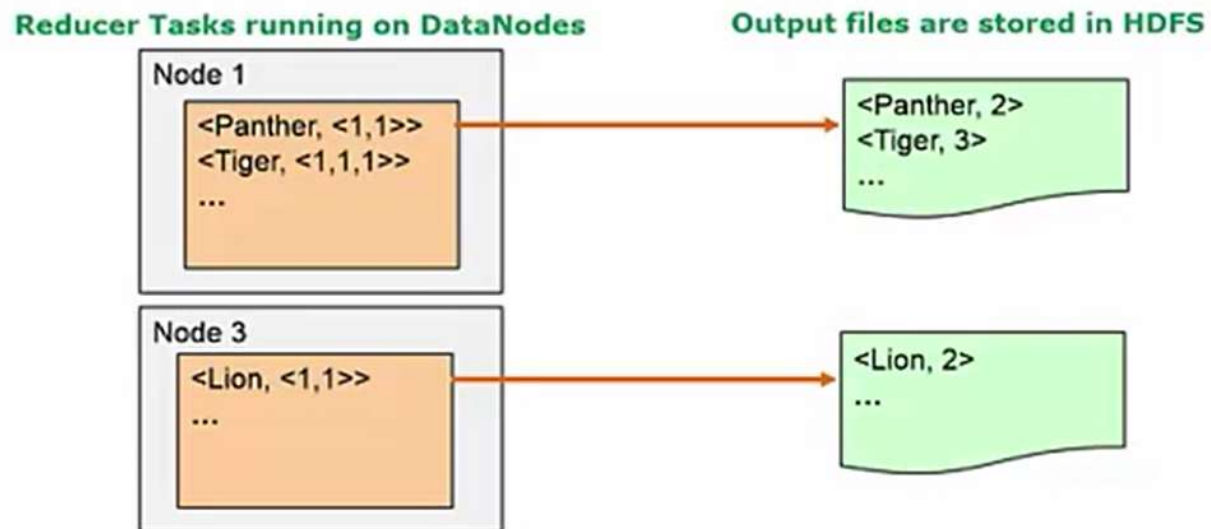
- Shuffle

- Move os dados com a mesma chave para o mesmo nó de reduce
- O número das tarefas map e reduce não precisa ser o mesmo



WordCount personalizado

- Reduce
 - Agrega os valores de cada chave

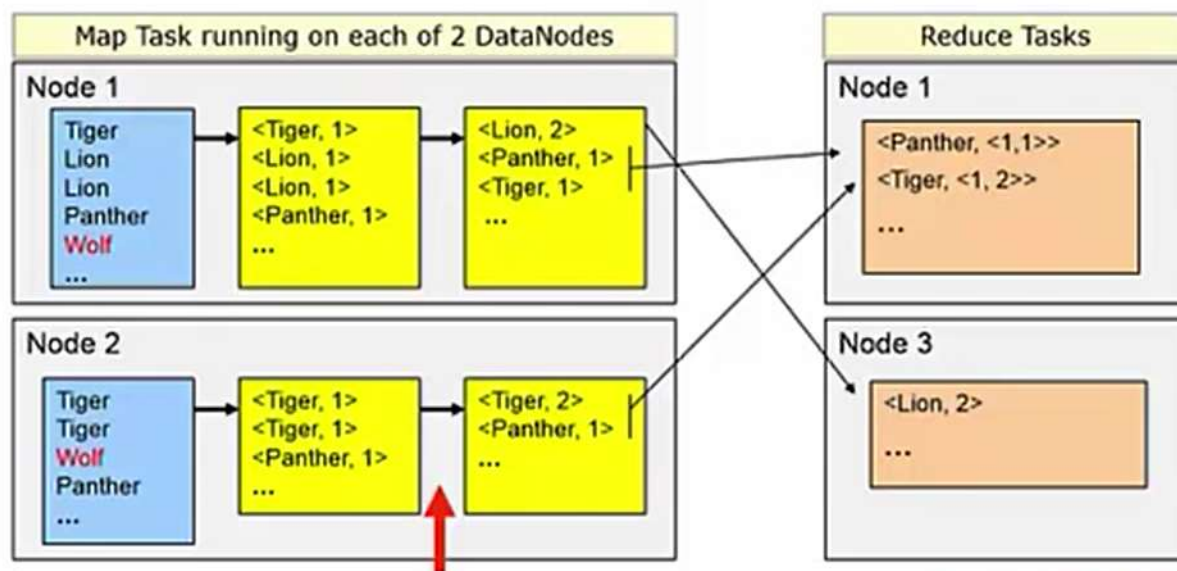


WordCount personalizado

- *Combiner* (opcional)

- Podemos fazer uma pré-agregação na tarefa de *map*, antes do *shuffle*
- Reduz a quantidade de dados enviados pela rede
- Reduz o processamento das tarefas de *reduce*

2. Crie um combiner que agregue os animais pela chave antes de executar o Shuffle



WordCount personalizado

- Classe *Combiner*
 - Os usuários podem opcionalmente especificar um *combiner* no programa principal *main*, via `Job.setCombinerClass(Class)`, para executar a agregação local das saídas intermediárias, o que ajuda a reduzir a quantidade de dados transferidos via rede do Mapper para o Reducer

Perceba que a classe `Combiner` deve estender a classe `Reducer`, porém, quando você configura o `Job` da forma mostrada, o framework Hadoop entende que vai executar essa tarefa antes do Shuffle

`setCombinerClass`

```
public void setCombinerClass(Class<? extends Reducer> cls)
    throws IllegalStateException
```

Set the combiner class for the job.

Parameters:

`cls` - the combiner to use

Throws:

`IllegalStateException` - if the job is submitted

Resumo da Atividade 2

- Rodar o exemplo WordCount tal e qual foi fornecido pela distribuição do Hadoop, observar os dados de saída
- Fazer as alterações solicitadas para gerar o WordCount personalizado
 - 1. Escreva um código que execute esse filtro no método map da classe WordCount.java
 - 2. Crie um combiner que agregue os animais pela chave antes de executar o Shuffle
- Vá registrando em um relatório todos os passos que você executar até chegar ao resultado final
 - Registre também os erros que acontecerem no caminho e como foram solucionados (por você ou por algum colega que tenha eventualmente te ajudado)
 - Ao final do relatório, colocar o log de execução com sucesso
- **Enviar esse relatório até terça-feira 23/03/2020**

Modelo para o relatório da Atividade 2

1. Informações sobre a equipe
2. Informações sobre a plataforma onde a atividade foi desenvolvida, configuração de ambiente etc.
3. Passo-a-passo para desenvolvimento da atividade, incluindo os erros encontrados e soluções
4. log de execução da atividade com sucesso
5. Conclusões finais da equipe (se foi fácil, difícil, como gerenciaram o trabalho de forma remota, dificuldades e pontos positivos)

Atividade 03 – Lista de Exercícios Hadoop + Pig

Atividade 3

- Esta atividade também deve ser desenvolvida em equipe
- Vocês devem se inscrever no curso online gratuito da IBM
 - <https://cognitiveclass.ai/courses/mapreduce-and-yarn>
- Façam o módulo 1 do curso: Module 1: Introduction to MapReduce and YARN
- Façam o exercício hands-on do módulo 1, cujas instruções estão no pdf [Lab11-Hadoop.pdf](#) (você também pode baixar diretamente do ambiente do curso)
- Registrem o passo-a-passo para execução dos exercício em um relatório, assim como na Atividade 2

Atividade 3 - observações

- O guia do exercício vai pedir para que você utilize o ambiente IBM Cloud, mas você pode (e deve) usar o mesmo ambiente que usou para desenvolver a Atividade 2
- Para isso, você terá que instalar a ferramenta Pig, usando esse ou outro tutorial de sua escolha
 - https://www.tutorialspoint.com/apache_pig/apache_pig_installation.htm
- **Enviar o relatório até terça-feira 31/03/2020**

Modelo para o relatório da Atividade 3

1. Informações sobre a equipe
2. Informações sobre a plataforma onde os exercícios foram desenvolvidos, configuração de ambiente etc.
3. Passo-a-passo para desenvolvimento dos exercícios, incluindo os erros encontrados e soluções
 - Os passos podem ser um pouco diferentes do guia, a depender da plataforma utilizada
 - Perceba que existem algumas perguntas que devem ser respondidas, ao longo dos exercícios, as respostas devem ser adicionadas ao passo-a-passo do relatório
4. Adicionar os logs ao final da execução de cada exercício
5. Conclusões finais da equipe (se foi fácil, difícil, como gerenciaram o trabalho de forma remota, dificuldades e pontos positivos)

Material Complementar

- Tutoriais Hadoop
 - <https://www.datasciencecentral.com/profiles/blogs/hadoop-tutorials>
 - <https://hashnode.com/post/how-i-was-finally-able-to-run-the-infamous-word-count-example-on-hadoop-ciwazrq8u000vgw53qe4saran>
- Tutoriais Pig
 - <https://www.codigofluente.com.br/aula-16-hadoop-tutorial-apache-pig/>
 - https://www.tutorialspoint.com/apache_pig/apache_pig_installation.htm

Referências

- Apache Hadoop
 - <http://hadoop.apache.org>
- MapReduce Tutorial
 - <http://hadoop.apache.org/docs/r2.7.4/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html>
- MapReduce and YARN Big Data University **BD0115EN**
 - <https://cognitiveclass.ai/courses/mapreduce-and-yarn>