BIG DATA

LABDATA

Coordenação:

Prof. Dr. Adolpho Walter Pimazzi Canton

Profa. Dra. Alessandra de Ávila Montini

Disciplina: Aplicações de Big Data com Hadoop

Tema da Aula: MapReduce

Profa. Rosangela de Fátima Pereira

Junho de 2016

Currículo

Formação

- Mestrado em Engenharia de Computação pela Escola Politécnica da Universidade de São Paulo (Poli-USP) (em andamento)
- Especialização em Tecnologia Java pela Universidade Tecnológica Federal do Paraná (UTFPR) (2011)
- Tecnologia em Análise e Desenvolvimento de Sistemas pela UTFPR (2011)
- Bacharelado em Administração de Empresas pela Universidade Estadual do Norte do Paraná (UENP) (2007)

Experiência

- Professora de Big Data Analytics em empresas e programas de MBA - FIA (2013 - atual)
- Pesquisadora no Laboratório de Arquitetura e Redes de Computadores (LARC) – USP (2013 - atual)
- Professora de cursos de engenharia na UTFPR (2011 -2012)
- Analista de sistemas na BSI Tecnologia (2009-2010)

LinkedIn: https://br.linkedin.com/pub/rosangela-de-fatima-pereira/68/a10/b56

Apaixonada por **Big Data**!

Objetivo da Aula

Apresentar ao aluno fundamentos da biblioteca MapReduce

Apresentar exemplos práticos da implementação e execução de uma aplicação MapReduce

Conteúdo da Aula

- Revisão da aula anterior
- Biblioteca Hadoop
- MapReduce na Prática

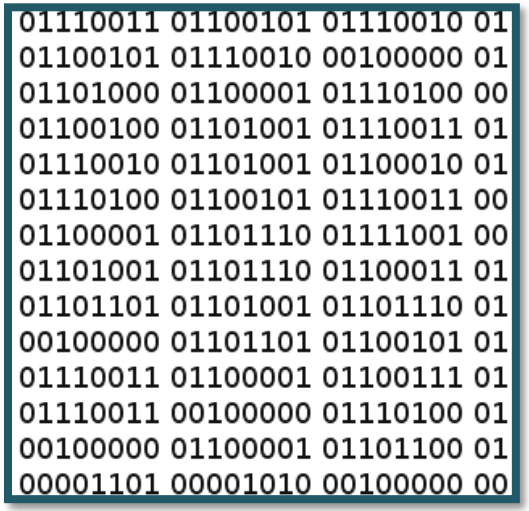
Aula anterior

Linguagem humana

```
a = 2 + 2;
b = a + 5;
c = a + b;
```

INTERPRETADOR

Linguagem de máquina



Aula anterior

Aula anterior

- Java permite que o código fique bem organizado
- A manutenção é mais fácil e menos custosa
- Permite a alteração do código sem alterar o código do cliente
- Permite executar o mesmo código em diversos sistemas operacionais

Aula anterior

Exemplo de um código Java

```
public class MeuPrograma {
    public static void main(String[] args) {
        System.out.println(“meu primeiro programa”);
    }
}
```

Aula anterior

Criação da classe Conta.java

*Conta.java

```java

public class Conta {

    public int numero;
    public String nomeCliente;
    public double saldo;
    public double limite;

    public void sacarDinheiro(double quantidade){
        double novoSaldo = this.saldo - quantidade;
        this.saldo = novoSaldo;
    }

    public void depositarDinheiro(double quantidade){
        this.saldo = this.saldo + quantidade;
    }

}

```

Conteúdo da Aula

- Revisão da aula anterior
- Biblioteca Hadoop
- MapReduce na Prática

Biblioteca Hadoop

Conjunto de classes que podem ser herdadas para utilização de funcionalidades pré-definidas

O conjunto de classe é dividido nos seguintes componentes:

- Hadoop Common
- Hadoop Distributed File System (HDFS)
- Hadoop MapReduce
- Hadoop YARN

Biblioteca Hadoop

Hadoop Common

- Considerado o núcleo do framework Hadoop, uma vez que fornece serviços essenciais e processos básicos, tais como abstração do sistema operacional subjacente e seu sistema de arquivo.
- Hadoop Common também contém os arquivos e scripts Java Archive (JAR) necessários para iniciar o Hadoop.
- O pacote comum Hadoop também fornece código fonte e documentação.

Biblioteca Hadoop

Hadoop Common - Exemplos

- hadoop-config.sh
- hadoop-daemon.sh
- hadoop-daemons.sh
- hdfs-config.sh
- mapred-config.sh
- slaves.sh
- start-all.sh
- start-balancer.sh
- start-dfs.sh
- start-mapred.sh
- stop-all.sh
- stop-balancer.sh
- stop-dfs.sh
- stop-mapred.sh

Biblioteca Hadoop

Hadoop Distributed File System (HDFS)

Conjunto de classes abstratas para um sistema de arquivos genérico.

Pode ser implementado como um sistema de arquivos distribuído, ou como um "local“, para testes.

Exemplos de classes:

org.apache.hadoop.fs.FileSystem

org.apache.hadoop.fs.FileUtil

org.apache.hadoop.fs.FsStatus

Biblioteca Hadoop

Hadoop Distributed File System (HDFS)

Exemplos de métodos:

- copyFromLocalFile(boolean delSrc, boolean overwrite, Path src, Path dst)
- create(Path f, boolean overwrite)
- getHomeDirectory()
- getStatus()
- mkdirs(Path f)
- open(Path f)

Biblioteca Hadoop

Hadoop MapReduce

Conjunto de classes abstratas e interfaces para a implementação e execução de aplicações MapReduce

Exemplos de classes:

org.apache.hadoop.mapred.JobConf

org.apache.hadoop.mapred.FileInputFormat<K,V>

org.apache.hadoop.mapred.FileOutputFormat<K,V>

Biblioteca Hadoop

Hadoop MapReduce

Exemplos de métodos:

- getJar()
- getMapOutputKeyClass()
- getMapOutputValueClass()
- getNumMapTasks()
- getNumReduceTasks()
- setJar(String jar)
- setMapperClass(Class<? extends Mapper> theClass)

Biblioteca Hadoop

Como utilizar essas classes?

1. Adicionar os arquivos jar´s ao projeto
2. Importar as classes que serão utilizadas na implementação

Exemplo

```
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
```

Biblioteca Hadoop

Implementando uma aplicação MapReduce

Biblioteca Hadoop

Os seguintes passos serão executados:

1. **Implementar as classes da aplicação**
2. Gerar um arquivo JAR da aplicação
3. Executar o JAR no ambiente Hadoop
4. Visualizar o resultado da aplicação

Implementação MapReduce

Serão implementadas 3 classes:

ContaPalavrasMap.java

ContaPalavrasReduce.java

ContaPalavrasDriver.java

– Para facilitar o desenvolvimento das classes foi criado um projeto inicial chamado **ContaPalavras** contendo a estrutura-base das classes.

Implementação MapReduce

Software: VMware

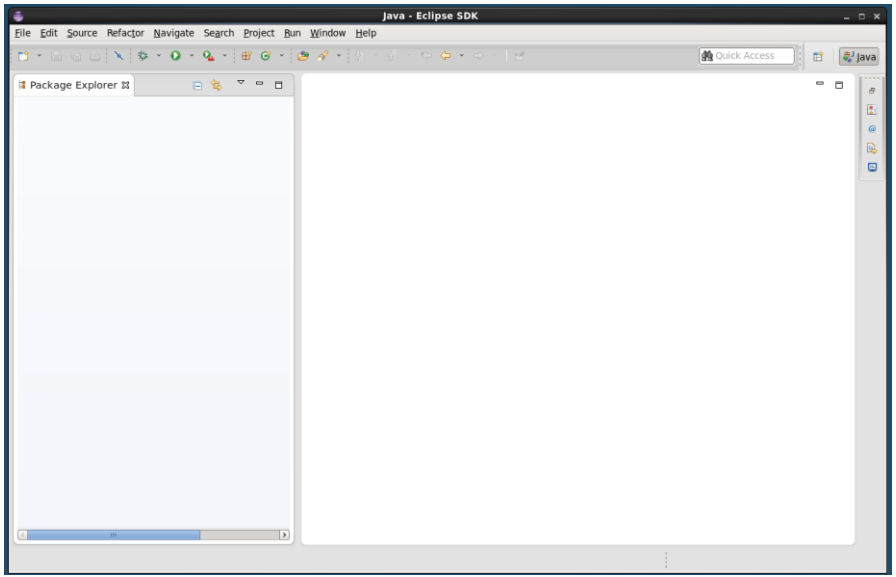
Máquina virtual: Cloudera-training-analyst

cloudera

Implementação MapReduce

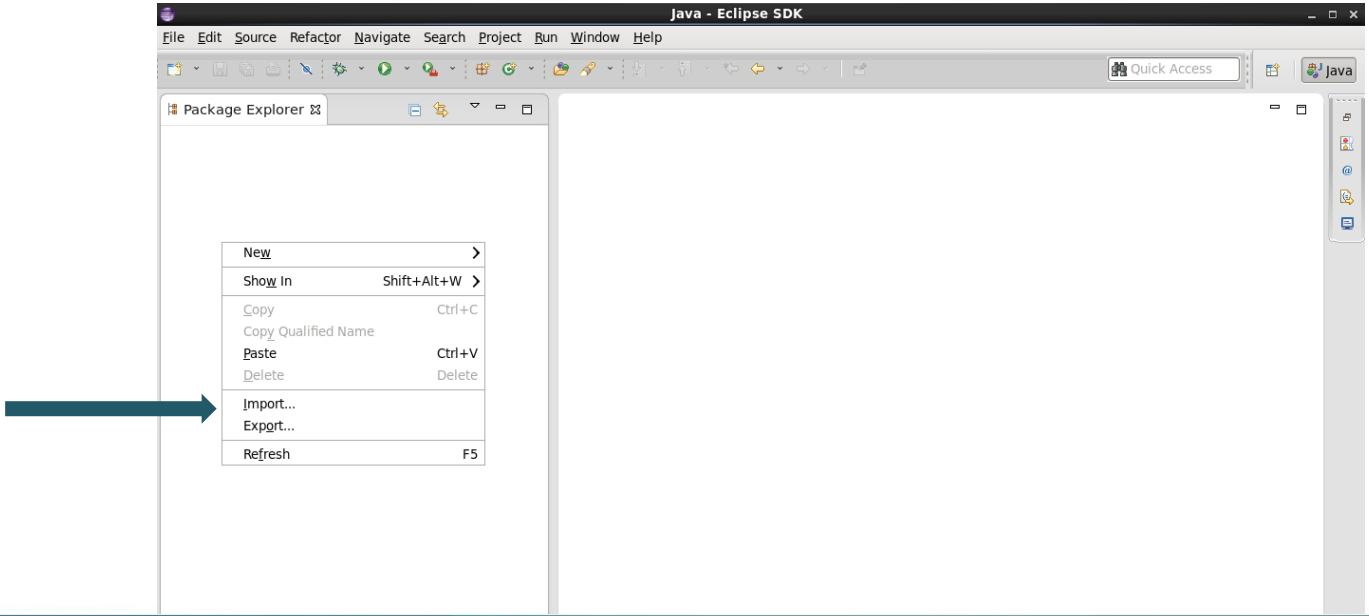
Desenvolvendo a primeira aplicação Java

- Abra a IDE Eclipse



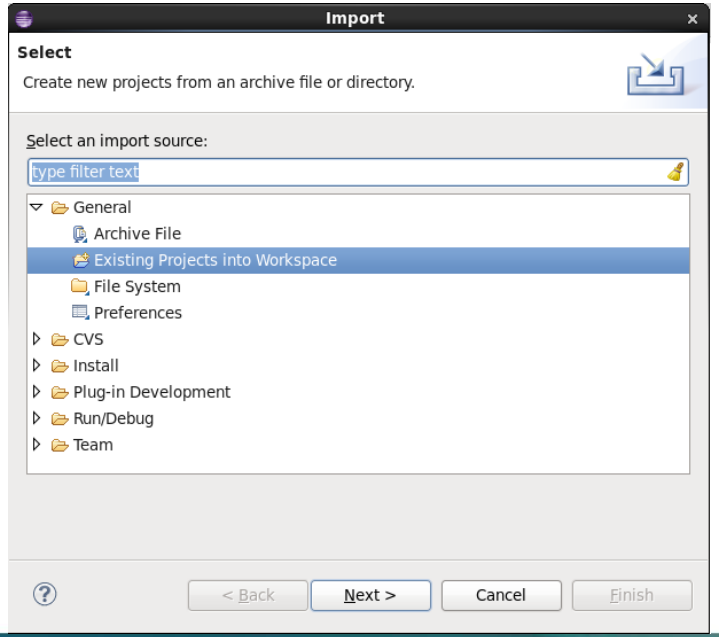
Implementação MapReduce

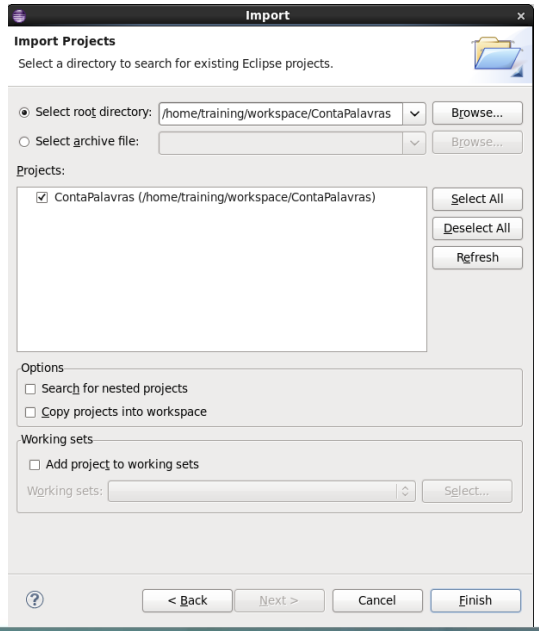
Acessar o projeto no Eclipse pela opção Import



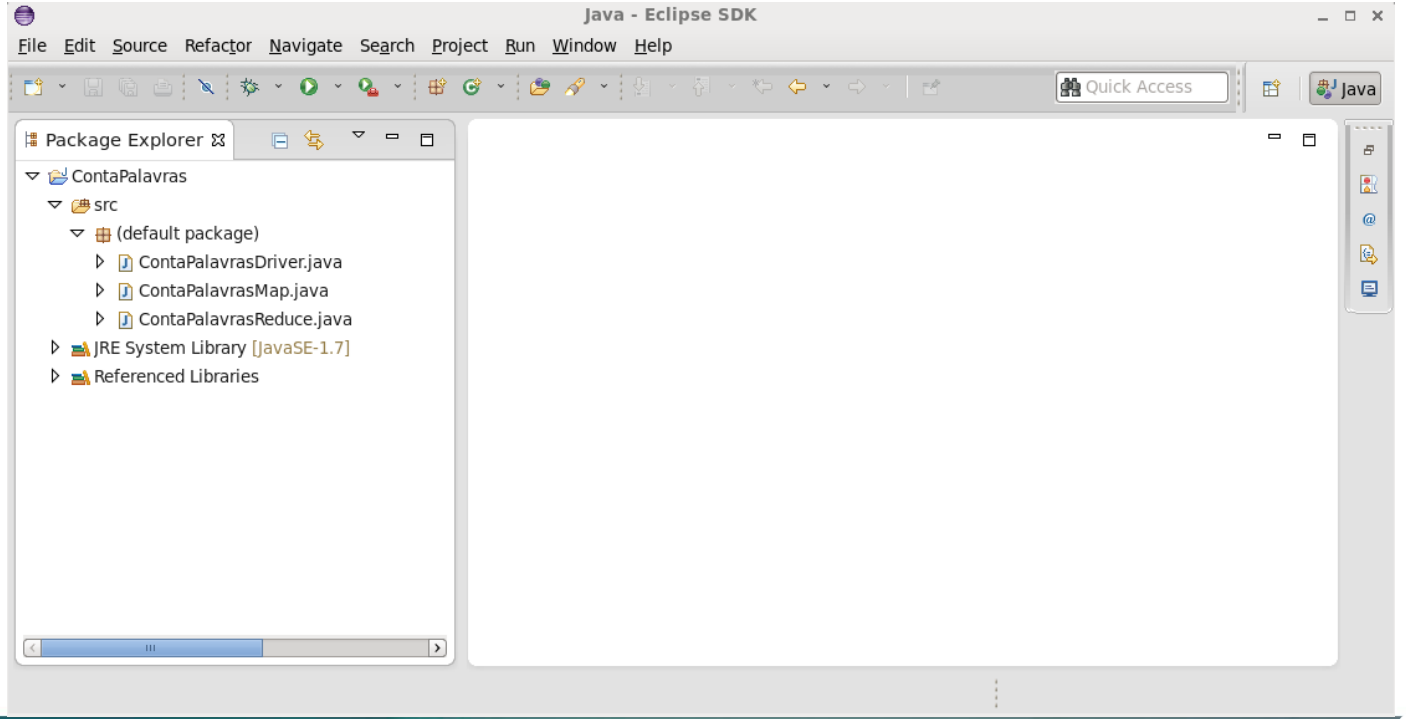
Implementação MapReduce

Selecionar o projeto ContaPalavras





Implementação MapReduce



Implementação MapReduce

Classe ContaPalavrasDriver

Implementação MapReduce

A classe ContaPalavrasDriver possui a seguinte estrutura:

Importação de classes da biblioteca Hadoop

```
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
```

Implementação MapReduce

Declaração da Classe e declaração do método

```
public class ContaPalavrasDriver {

public static void main(String[] args) throws Exception {


        }
}
```

Implementação MapReduce

Criação de um objeto Job

```
public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "contapalavrasdriver");




        }
}
```

Implementação MapReduce

Configuração do Job

```
public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "contapalavrasdriver");
    job.setJarByClass(ContaPalavrasDriver.class);
    job.setMapperClass(ContaPalavrasMap.class);
    job.setReducerClass(ContaPalavrasReduce.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);

            }
}
```

Implementação MapReduce

Definição dos arquivos de entrada e de saída

```
public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "contapalavrasdriver");
    ...
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));


                }
}
```

Implementação MapReduce

Definição dos arquivos de entrada e de saída

```
public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "contapalavrasdriver");
    ...
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);

                }
}
```

Implementação MapReduce

Classe ContaPalavrasMap

Implementação MapReduce

Como os dados são enviados às tarefas Map?

- Especificado por um objeto ***InputFormat***
- O formato de envio deve ser especificado na classe driver
- Deve ser especificado a localização dos dados de entrada
- Esse objeto determina como os dados de entrada serão divididos

Implementação MapReduce

Exemplos de objetos do tipo InputFormat

- TextInputFormat
  - Default, lê cada linha terminada com “\n” como sendo um valor
- FileInputFormat
  - Classe abstrata usada para InputFormats baseados em arquivos
- KeyValueTextInputFormat
  - Determina linhas por meio de um separador (tab por padrão)

Implementação MapReduce

Como definir as chaves e valores na classe?

- Chaves e valores são objetos Java
- As chaves são objetos que implementam a interface **WritableComparable**
- Os valores são objetos que implementam a interface **Writable**
  - **Exemplos:** IntWritable, LongWritable, FloatWritable, Text...

Implementação MapReduce

A classe ContaPalavrasMap possui a seguinte estrutura:

Importação de classes da biblioteca Hadoop

```
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
```

Implementação MapReduce

Declaração da Classe

```
public class ContaPalavrasMap extends Mapper<Object, Text, Text, IntWritable>{




}
```

Implementação MapReduce

Declaração das variáveis utilizadas

```
public class ContaPalavrasMap extends Mapper<Object, Text, Text, IntWritable>{
    private final static IntWritable numeroum = new IntWritable(1);




}
```

Implementação MapReduce

Declaração do método map

```
public class ContaPalavrasMap extends Mapper<Object, Text, Text, IntWritable>{
    private final static IntWritable numeroum = new IntWritable(1);
    public void map(Object key, Text value, Context context) throws IOException,
        InterruptedException {



        }
}
```

Implementação MapReduce

Declaração do código do método map

```
public class ContaPalavrasMap extends Mapper<Object, Text, Text, IntWritable>{
    private final static IntWritable numeroum = new IntWritable(1);
    public void map(Object key, Text value, Context context) throws IOException,
        InterruptedException {
            StringTokenizer itr = new StringTokenizer(value.toString().replaceAll("[^a-zA-Z ]", "").toLowerCase());
            while (itr.hasMoreTokens()) {
                context.write(new Text(itr.nextToken()), numeroum);
            }
            }
}
```

Implementação MapReduce

Classe ContaPalavrasReduce

Implementação MapReduce

A classe ContaPalavrasReduce possui a seguinte estrutura:

Importação de classes da biblioteca Hadoop

```
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
```

Implementação MapReduce

Declaração da classe

```
public class ContaPalavrasReduce extends Reducer<Text,IntWritable,Text,IntWritable> {



}
```

Implementação MapReduce

Declaração do método reduce

```
public class ContaPalavrasReduce extends Reducer<Text,IntWritable,Text,IntWritable> {

    public void reduce(Text key, Iterable<IntWritable> values, Context context
            ) throws IOException, InterruptedException {



        }
}
```

Implementação MapReduce

Declaração do código do método reduce

```
public class ContaPalavrasReduce extends Reducer<Text,IntWritable,Text,IntWritable> {
    public void reduce(Text key, Iterable<IntWritable> values, Context context
            ) throws IOException, InterruptedException {
        int soma = 0;
        for (IntWritable val : values) {
            soma += val.get();
        }
        context.write(key, new IntWritable(soma));
        }
}
```

Implementação MapReduce

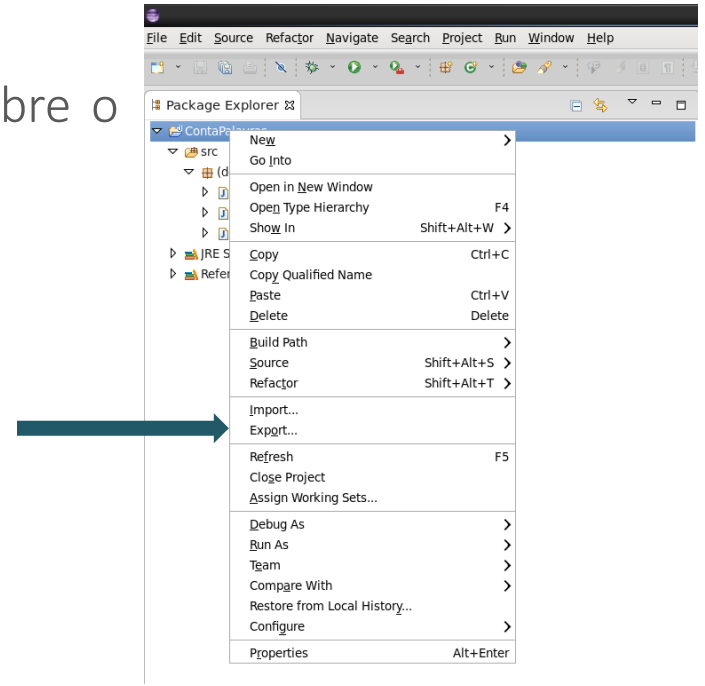
Os seguintes passos serão executados:

1. **Implementar as classes da aplicação**
2. **Gerar um arquivo JAR da aplicação**
3. Executar o JAR no ambiente Hadoop
4. Visualizar o resultado da aplicação

Implementação MapReduce

Selecionar a opção export

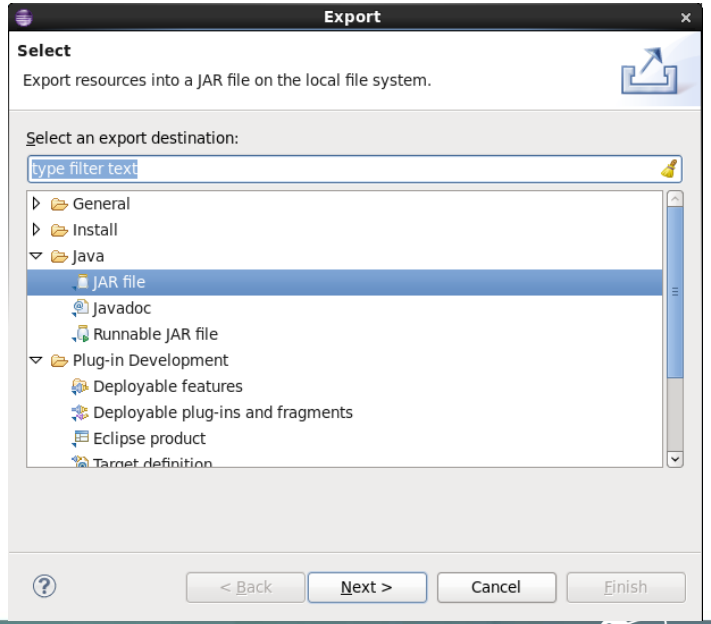
- Clique com o botão direito do mouse sobre o nome do projeto



Implementação MapReduce

Selecionar a opção JAR file

- Após seleção, pressione o botão Next



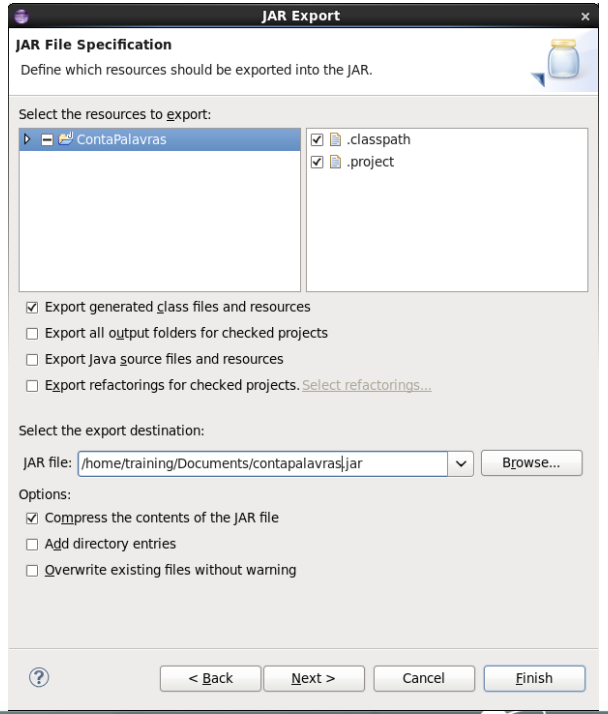
Implementação MapReduce

Definir a localização do JAR

- Definir o seguinte caminho:

/home/training/Documents/contapalavras.jar

- Pressione o botão Finish



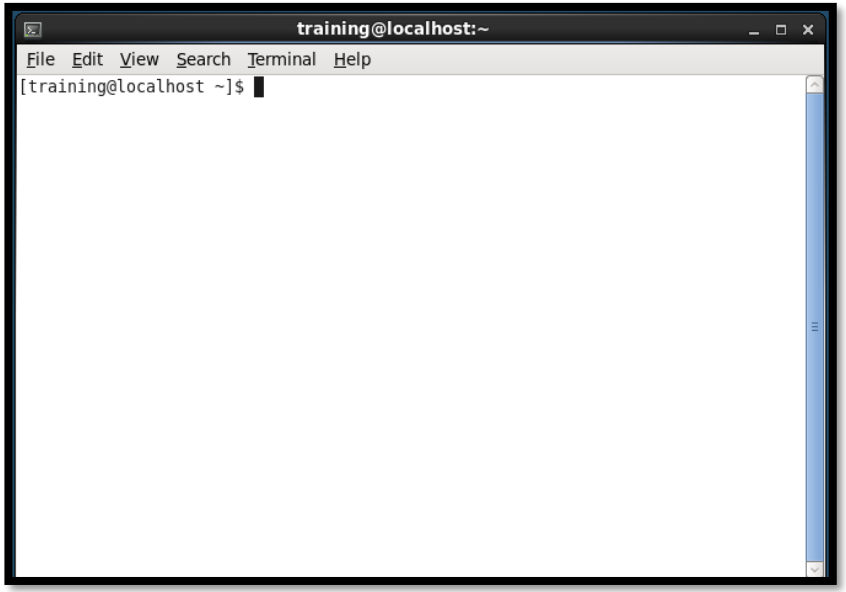
Implementação MapReduce

Os seguintes passos serão executados:

1. Implementar as classes da aplicação
2. Gerar um arquivo JAR da aplicação
3. Executar o JAR no ambiente Hadoop
4. Visualizar o resultado da aplicação

Implementação MapReduce

Abrir um terminal



Implementação MapReduce

Base de dados:

Mensagens da rede social Twitter sobre os temas: Big Data, data visualization, data privacy e Internet of Things

Exemplo

How can data visualization be used in the sales process? Learn how it can yield great insights. http://t.co/Lq2lcpe7d1

RT @Mona_Mourshed: "When Big Data Meets the Blackboard" http://t.co/f3lr4yQW9A via @TheAtlantic

Big data: reduce privacy risks - REPUTATION PROTECT http://t.co/DrAb6LzFJJ

Absolute privacy in handling #mhealth data. Patient security comes first.“

RT @jose_garde: RT @MarshaCollier The Internet of Things and the Currency of Privacy http://t.co/eXwmluX8sL

Implementação MapReduce

Criar um diretório no HDFS como o nome de input_cp.

```
[training@localhost ~]$ hadoop   fs   -mkdir   input_cp
```

Implementação MapReduce

Enviar o arquivo base_tw.txt para o diretório input_cp.

```
[training@localhost ~]$ hadoop   fs   -put  ~/bases/base_tw.txt  input_cp
```

Implementação MapReduce

Executar o jar contapalavras.jar, passando como parâmetro o diretório de entrada input_cp e um diretório de saída output_cp.

```
[training@localhost ~]$ hadoop jar ~/Documents/contapalavras.jar
ContaPalavrasDriver input_cp output_cp
16/04/14 05:30:50 WARN mapred.JobClient: Use GenericOptionsParser for parsing the
arguments. Applications should implement Tool for the same.
16/04/14 05:30:53 INFO input.FileInputFormat: Total input paths to process : 1
16/04/14 05:30:55 INFO mapred.JobClient: Running job: job_201512191358_0059
16/04/14 05:30:56 INFO mapred.JobClient:  map 0% reduce 0%
16/04/14 05:31:52 INFO mapred.JobClient:  map 100% reduce 0%
…
```

Implementação MapReduce

Os seguintes passos serão executados:

1. Implementar as classes da aplicação
2. Gerar um arquivo JAR da aplicação
3. Executar o JAR no ambiente Hadoop
4. Visualizar o resultado da aplicação

Implementação MapReduce

Visualizar o diretório de saída da aplicação

```
#Listar arquivos do diretório resultado
[training@localhost ~]$ hadoop   fs   -ls   output_cp
Found 3 items
-rw-rw-rw-   1 training supergroup          0 2016-04-14 05:32 output_cp/_SUCCESS
drwxrwxrwx   - training supergroup          0 2016-04-14 05:30 output_cp/_logs
-rw-rw-rw-   1 training supergroup     187407 2016-04-14 05:32 output_cp/part-r-00000
```

Implementação MapReduce

Visualizar o resultado da aplicação

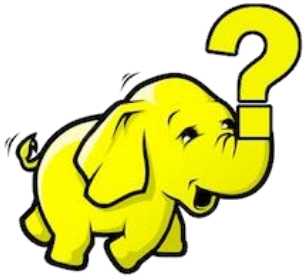
```
#Visualizar arquivo part-r-00000
[training@localhost ~]$ hadoop   fs   -cat   output_cp/part-r-00000
Data            1
DataScience 3
DataScientist"                  10
Facebook     1
NewsForBloggers             1
Python          1
...
```

Implementação MapReduce

Considerações

- O projeto criado pode ser utilizado como base para outras aplicações
- O mesmo código pode ser executado em um cluster com milhares de máquinas
- É possível criar uma aplicação com múltiplas iterações Map e Reduce
- Em caso de erros, verificar os registros de log do Hadoop
- Documentação do Hadoop: http://hadoop.apache.org/

Perguntas



rpereira@larc.usp.br

Referências Bibliográficas

WHITE, Tom. **Hadoop: The definitive guide**. " O'Reilly Media, Inc.", 2012.

DEAN, Jeffrey; GHEMAWAT, Sanjay. **MapReduce: simplified data processing on large clusters.** Communications of the ACM, v. 51, n. 1, p. 107-113, 2008.

GHEMAWAT, Sanjay; GOBIOFF, Howard; LEUNG, Shun-Tak. **The Google file system**. In: ACM SIGOPS Operating Systems Review. ACM, 2003. p. 29-43.

VAVILAPALLI, Vinod Kumar et al. **Apache hadoop yarn: Yet another resource negotiator**. In: Proceedings of the 4th annual Symposium on Cloud Computing. ACM, 2013. p. 5.

SHVACHKO, Konstantin et al. **The hadoop distributed file system**. In: Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on. IEEE, 2010. p. 1-10.

Referências Bibliográficas

GOLDMAN, Alfredo et al. **Apache Hadoop: conceitos teóricos e práticos, evolução e novas possibilidades**. XXXI Jornadas de atualizações em informática, p. 88-136, 2012.

VENNER, Jason. **Pro Hadoop**. Apress, 2009.