Universidade de Brasília Faculdade UnB Gama

CURSO:	**Engenharia de Software**	
DISCIPLINA:	**Programação para Sistemas Paralelos e Distribuídos**	**TURMA: T01**
SEMESTRE:	**2023-1**	
PROFESSOR(A):	**Fernando William Cruz**	**MATRÍCULA:**
ALUNO:	**Fernando Miranda Calil**	**19/0106565**
	Antonio Aldisio de Sousa Alves Ferreira Filho	**20/2028211**

Lab01 – Framework Hadoop e o paradigma de programação MapReduce

Introdução

O termo Hadoop se refere ao projeto Apache Hadoop, que é uma biblioteca de software (framework) que inclui o MapReduce (estrutura de execução), o YARN (gerente de recursos) e o HDFS (armazenamento distribuído). Essa estrutura permite o processamento distribuído de grandes conjuntos de dados em clusters de computadores usando modelos de programação simples, sendo possível ser feito nas linguagens Java e Python. A biblioteca foi projetada para escalar de servidores únicos para milhares de máquinas, cada uma oferecendo poder computacional e armazenamento locais. Em vez de depender de hardware para fornecer alta disponibilidade, a própria biblioteca foi projetada para detectar e lidar com falhas na camada do aplicativo, fornecendo assim um serviço altamente disponível em um cluster de computadores, cada um dos quais pode estar sujeito a falhas.

O Hadoop MapReduce, um mecanismo de execução no Hadoop, processa cargas de trabalho usando a estrutura do MapReduce, que divide os trabalhos em componentes menores que podem ser distribuídos entre os nós no seu cluster. O mecanismo do Hadoop MapReduce foi criado considerando que qualquer máquina no seu cluster pode falhar, a qualquer momento, e foi projetado com tolerância a falhas. Se um servidor executando uma tarefa falhar, o Hadoop executa essa tarefa em outra máquina até sua conclusão.

O Hadoop também inclui um sistema de armazenamento distribuído, o Hadoop Distributed File System (HDFS), que armazena dados em discos locais do cluster em grandes blocos. O HDFS tem um fator de replicação configurável (com um padrão de 3 vezes), o que proporciona mais disponibilidade e durabilidade. O HDFS monitora a replicação e balanceia dados nos nós, conforme os nós forem falhando e outros forem adicionados.

Contador de palavras

O código foi retirado da documentação do apache hadoop (https://hadoop.apache.org/docs/stable/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html). Ele é separado em duas funções map e reducer, respectivamente, figura 01 e figura 02.

```
  public void map(Object key, Text value, Context context
                  ) throws IOException, InterruptedException {
    StringTokenizer itr = new StringTokenizer(value.toString());
    while (itr.hasMoreTokens()) {
      word.set(itr.nextToken());
      context.write(word, one);
    }
  }
}
```

Figura 01 – Função map do contador de palavras.

```java
public static class IntSumReducer
     extends Reducer<Text,IntWritable,Text,IntWritable> {
  private IntWritable result = new IntWritable();

  public void reduce(Text key, Iterable<IntWritable> values,
                     Context context
                     ) throws IOException, InterruptedException {
    int sum = 0;
    for (IntWritable val : values) {
      sum += val.get();
    }
    result.set(sum);
    context.write(key, result);
  }
}
```

Figura 02 – Função reducer do contador de palavras

É necessário compilar o arquivo e transformar ele em .jar para utilizar o framework do hadoop. Para execução dessa etapa foram encontrados dois problemas. O primeiro problema foi a compatibilidade do java com a versão do hadoop. Para solução disso foi necessário inserir as flags -target 8 -source 8 para deixar o .jar com versão do java necessário para o Hadoop. O segundo foi conseguir identificar o diretório da classe do hadoop utilizado para compilação do código. O próprio hadoop tem um comando que mostra o caminho desse diretório que é hadoop classpath.

Assim, conseguimos realizar o experimento passando como entrada input.txt, visto na figura 03, e obtivemos a saída como esperada, visto na figura 04. Na figura 05 temos o diretório do retorno da execução.

```
a202028211@chococino:~/pspd/hadoop/lab_01/contadorPalavras/input_data$ cat input.txt
Antonio
Fernando
Aldisio
de
Sousa
Fernando
Alves
Ferreira
Filho
Antonio
Aldisio
de
Sousa
Caio
Caio
Maria
Jose
Jose
Fernanda
Fernando
Fernanda
Fernando
```

Figura 03 – Input do contador de palavras

```
a202028211@chococino:~$ hdfs dfs -cat output_data2/part-r-00000
2023-04-15 20:01:48,973 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... usin
g builtin-java classes where applicable
Aldisio 2
Alves   1
Antonio 2
Caio    2
Fernanda        2
Fernando        4
Ferreira        1
Filho   1
Jose    2
Maria   1
Sousa   2
de      2
a202028211@chococino:~$
```

Figura 04 – Saída do contador de palavras

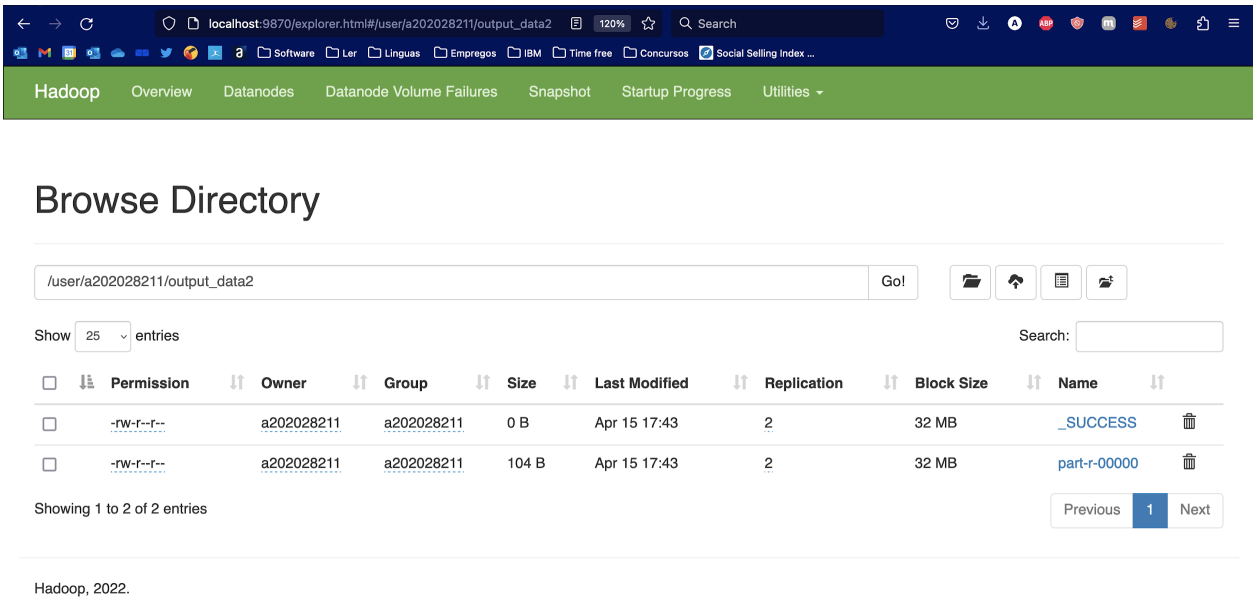

Figura 05 – Diretório da saída do contador de palavras

Rede de amizade

Para realização do segundo experimento, foi necessário apenas alterar a função map, como visto na figura 06. Não encontramos nenhum novo problema na execução de compilar e transformar em arquivo .jar. Na figura 07, temos a entrada e na figura 08 a saída e por fim, na figura 09 o diretório da saída.

```
  public void map(Object key, Text value, Context context
                  ) throws IOException, InterruptedException {

    String aux = value.toString();
    int pos = aux.indexOf(",");
    StringTokenizer itr = new StringTokenizer(aux.substring(0,pos));
    while (itr.hasMoreTokens()) {
      word.set(itr.nextToken());
      context.write(word, one);
    }
  }
}
```

Figura 06 – Função map da rede de amizade

```
a202028211@cm2:~$ hdfs dfs -cat lab02/input*
2023-04-16 14:55:35,208 WARN util.NativeCodeLoader: Unable to load native-hadoop
 library for your platform... using builtin-java classes where applicable
Joao, Maria
Maria, Joao
Lorena, Jose
Jose, Lorena
Joao, Caio
Caio, Joao
Joao, Lorena
Lorena, Joaoa202028211@cm2:~$
```

Figura 07 – Entrada da rede de amizade

```
Lorena, Joaoa202028211@cm2:~$ hdfs dfs -cat lab02_output/*
2023-04-16 14:56:44,340 WARN util.NativeCodeLoader: Unable to load native-hadoop
 library for your platform... using builtin-java classes where applicable
Caio    1
Joao    3
Jose    1
Lorena  2
Maria   1
a202028211@cm2:~$
```

Figura 08 – Saída da rede de amizade

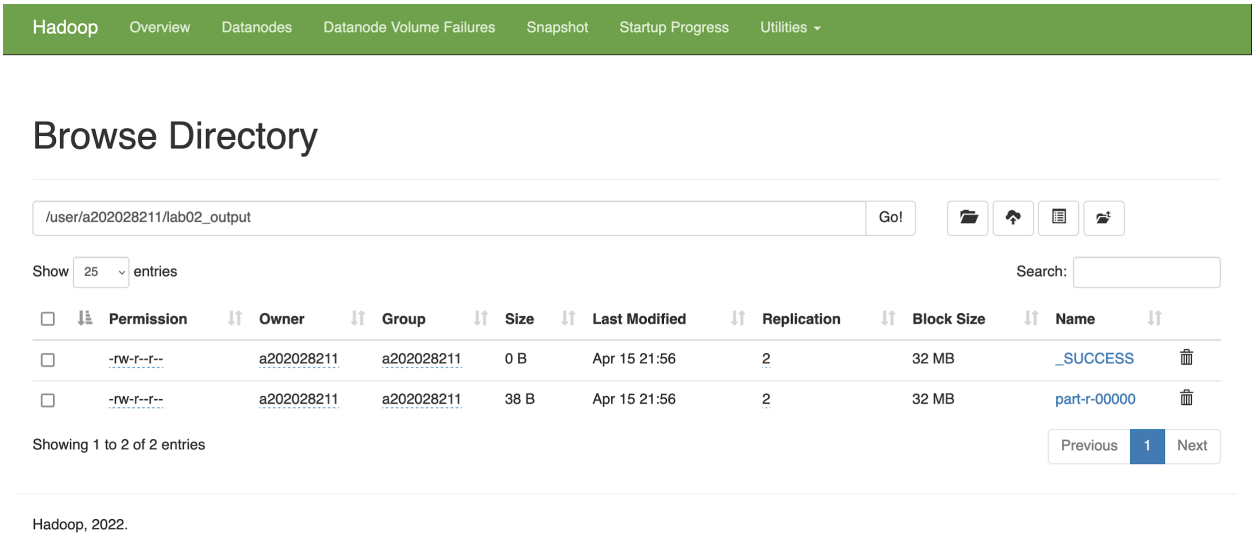

Figura 09 – Diretório da saída do rede de amizades

Referências

[1] - AMAZON**. Apache Hadoop no Amazon EMR** Disponível em : https://aws.amazon.com/pt/elasticmapreduce/details/hadoop/. Acesso em: 15 de abril de 2023.

[2] - APACHE. **MapReduce Tutorial.** Disponível em: https://hadoop.apache.org/docs/stable/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html. Acesso em: 15 de abril de 2023.