

Laboratorio 1 – Manejo Hadoop.

Nicolas Jimenez, Ivan Parra, Ricardo Moncaleano

Validación conocimientos

Universidad de los Andes, Bogotá, Colombia

{en.jimenez, id.parra, a.moncaleano}@uniandes.edu.co

Fecha de presentación: septiembre 8 de 2020

Tabla de contenido

1 Enlace Commit.....	1
2 Introducción	1
3 Objetivos	1
3.1 Objetivo Principal.....	1
3.2 Objetivos Específicos.....	1
4 Documentación Map Reduce	2
5 Resolución de Preguntas	4
Bibliografía	6

1 Enlace Commit

<https://github.com/ABD-MINE4102/202020-Grupo09/commit/5696ad8d5d0f8f6ca2a899f871bdeffd205a0a39>

2 Introducción

Esta primera entrega busca validar los conocimientos base y necesarios en relación con el tratamiento de data sets y manejo de herramientas que permita extraer información de valor. En esta ocasión se trabajará con data sets que contienen noticias de 1987 de la cadena radial Reuters. Como resultado esperado de esta primera entrega, se buscará comprobar conocimientos en manejo de text mining, diseñando e implementado funciones de retos establecidos como: Conteo de una palabra específica, conteo de palabras totales, las N palabras más frecuentes, así mismo N palabras en un archivo específico. A su vez validar conocimientos en manejo de scripts y tareas programas en UNIX.

3 Objetivos

3.1 Objetivo Principal.

- Validar y completar las habilidades de desarrollo básico de aplicaciones.

3.2 Objetivos Específicos.

- Desarrollar las habilidades básicas de uso de la arquitectura del curso.
- Validar y nivelar las habilidades básicas en procesamiento de información.
- Construir la infraestructura básica de presentación de talleres prácticos

4 Documentación Map Reduce

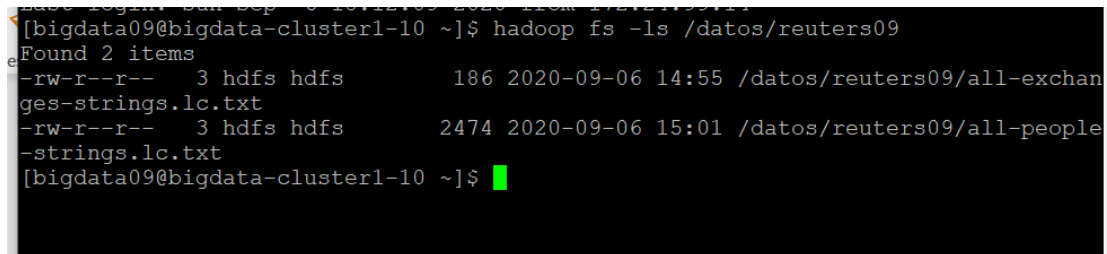
Explique de forma concisa y asertiva cómo opera la solución, cómo es la respuesta y cómo es el comportamiento del proceso en términos del sistema de archivos. Documente con fotos de pantalla el proceso de ejecución y el resultado.

La aplicación JAVA consta de 3 clases principales, una de ellas es el Mapper, el Reducer y la aplicación principal (WordCounter). La aplicación toma como parámetros 2 rutas, una ruta que es la de origen (ruta en hadoop) donde se van a leer los archivos dentro de la misma y se van a procesar, y una ruta de salida (ruta nueva en hadoop) donde se van a publicar los resultados del procesamiento de los archivos.

```
}
public static void ejecutarJob(String entrada, String salida) throws IOException, ClassNotFoundException, InterruptedException
{
    /**
     * Objeto de configuración, dependiendo de la versión de Hadoop
     * uno u otro es requerido.
     */
    Configuration conf = new Configuration();
    Job wcJob=Job.getInstance(conf, "WordCounter Job");
    wcJob.setJarByClass(WordCounter.class);
    //Mapper
    wcJob.setMapperClass(WCMapper.class);

    wcJob.setMapOutputKeyClass(Text.class);
    wcJob.setMapOutputValueClass(IntWritable.class);
    //Reducer
    wcJob.setReducerClass(WCReducer.class);
}
```

Cada vez que se envíe un proceso a ejecutar, la ejecución se distribuye por los servidores del cluster, para que el Mapper procese el funcionamiento y el Reduce obtenga los resultados.



```
[bigdata09@bigdata-cluster1-10 ~]$ hadoop fs -ls /datos/reuters09
Found 2 items
-rw-r--r--  3 hdfs hdfs      186 2020-09-06 14:55 /datos/reuters09/all-exchan
ges-strings.lc.txt
-rw-r--r--  3 hdfs hdfs     2474 2020-09-06 15:01 /datos/reuters09/all-people
-strings.lc.txt
[bigdata09@bigdata-cluster1-10 ~]$
```

En este caso listamos los archivos de la ruta especificada de Hadoop.

```

cluster1-01.virtual.uniandes.edu.co/172.24.99.73:8050
20/09/07 13:37:35 INFO client.AHSPProxy: Connecting to Application History server
at bigdata-cluster1-02.virtual.uniandes.edu.co/172.24.99.74:10200
20/09/07 13:37:35 WARN mapreduce.JobResourceUploader: Hadoop command-line option
parsing not performed. Implement the Tool interface and execute your applicatio
n with ToolRunner to remedy this.
20/09/07 13:37:35 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding f
or path: /user/bigdata09/.staging/job_1597942243319_0034
20/09/07 13:37:35 INFO input.FileInputFormat: Total input files to process : 2
20/09/07 13:37:35 INFO mapreduce.JobSubmitter: number of splits:2
20/09/07 13:37:36 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_15
97942243319_0034
20/09/07 13:37:36 INFO mapreduce.JobSubmitter: Executing with tokens: []
20/09/07 13:37:36 INFO conf.Configuration: found resource resource-types.xml at
file:/etc/hadoop/3.1.4.0-315/0/resource-types.xml
20/09/07 13:37:36 INFO impl.YarnClientImpl: Submitted application application_15
97942243319_0034
20/09/07 13:37:36 INFO mapreduce.Job: The url to track the job: http://bigdata-c
luster1-01.virtual.uniandes.edu.co:8088/proxy/application_1597942243319_0034/
20/09/07 13:37:36 INFO mapreduce.Job: Running job: job_1597942243319_0034
20/09/07 13:37:40 INFO mapreduce.Job: Job job_1597942243319_0034 running in uber
mode : false
20/09/07 13:37:40 INFO mapreduce.Job: map 0% reduce 0%

```

```

BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
  Bytes Read=2660
File Output Format Counters
  Bytes Written=3273
Job: job_1597942243319_0034
Job File: hdfs://bigdata-cluster1-01.virtual.uniandes.edu.co:8020/user/bigdata09
/.staging/job_1597942243319_0034/job.xml
Job Tracking URL : http://bigdata-cluster1-01.virtual.uniandes.edu.co:8088/proxy
/application_1597942243319_0034/
Uber job : false
Number of maps: 2
Number of reduces: 1
map() completion: 1.0
reduce() completion: 1.0
Job state: SUCCEEDED
retired: false
reason for failure:
[bigdata09@bigdata-cluster1-10 ~]$

```

Esta es una ejecución exitosa hacia la ruta anteriormente listada donde hay dos archivos que se le han hecho MAP y un REDUCE, esto quiere decir q se procesaron dos archivos y se tiene un reporte con la respuesta.

```
[bigdata09@bigdata-cluster1-10 ~]$ hadoop fs -cat result_reuters09/part-r-00000
| sort -k 2
abal 1
abdel 1
abdul 1
aguayo 1
alfonsin 1
alhaji 1
ali 1
alptemocin 1
amato 1
amex 1
andersen 1
andriessen 1
aqazadeh 1
aquino 1
arafat 1
ase 1
asx 1
aun 1
babangida 1
balladur 1
baluch 1
```

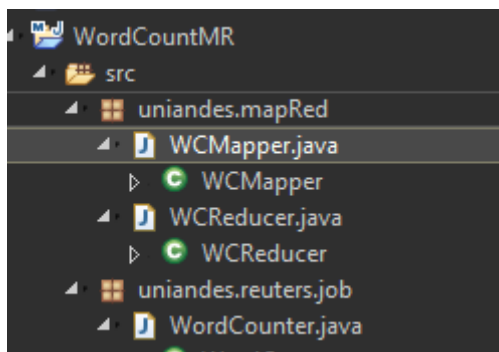
Una vez obtenemos la respuesta en la ruta indicada, al consultarla y ordenarla de forma ascendente, podemos ver el conteo de palabras de los archivos procesados.

5 Resolución de Preguntas

- ¿Cómo se están partiendo los datos, en unidades utilizadas en el proceso map? ¿Dónde en el código se establece esa forma de partirlos? Indíquelo de forma concreta y explique en máximo 3 líneas cómo funciona.

El proceso del Map (clase WCMapper) recibe el texto del archivo y lo convierte en un array (vector) separando cada palabra mediante los delimitadores `([().,!?;:'\"-]|\\s)+`.

```
String[] palabras = value.toString().split("[().,!?;:'\"-]|\\s)+");
```



- ¿Qué debe cambiar en su código si el archivo fuente es una secuencia de datos separados por un símbolo particular ('t', por ejemplo)? Justifique, en máximo 3 líneas.

En la clase WCMapper el arreglo “palabras” se forma mediante el texto recibido y le aplica un Split por determinados caracteres, se debe poner el que se necesite en este caso “\t”

```
String[] palabras = value.toString().split("\t");
```

- ¿Qué debe cambiar en su código si los archivos de entrada constan de una secuencia de datos en formato JSON? Justifique, en máximo 3 líneas.

Para el manejo con formato JSON se debe formar el array mediante cada objeto en JSON, posterior a ello se mapea un objeto JSON para cada objeto en el array y con ello se hace el manejo que corresponda

```
public class WCMapper extends Mapper<LongWritable, Text, Text, IntWritable> {  
  
    @Override  
    protected void map(LongWritable key, Text value,  
        Context context)  
        throws IOException, InterruptedException {  
        HashMap<String, Integer> palabrasLinea=new HashMap<String, Integer>();  
        String[] json = value.toString().split("\\n");  
        String lw;  
        for(Object objJson:Json){  
            JSONObject jsonValue = new JSONObject(objJson);  
            lw = jsonValue.getString("propiedad");  
  
            palabrasLinea.put(lw,  
                palabrasLinea.containsKey(lw)?  
                    (palabrasLinea.get(lw)+1)  
                    :1);  
        }  
        for(String k:palabrasLinea.keySet()){  
            context.write(new Text(k), new IntWritable(palabrasLinea.get(k)));  
        }  
    }  
}
```

- ¿Cuáles son alternativas ya disponibles para trabajar con formatos de entrada diferentes a los que utiliza el ejemplo? Indíquelo en máximo 3 líneas.

Se pueden utilizar archivos todo tipo de archivos planos, como CSV, que las palabras se encuentren delimitadas por los siguientes caracteres ([(),.,!?:;'\"-]|\\s)+

- ¿Qué debe cambiar en el código de ejemplo si quiere contar cuántas noticias de más de 100 palabras hay? Justifique en máximo 3 líneas.

En el proceso del map, no se deben contar y agregar al HashMap las palabras si no contar la noticia si cumple con la condición de tener más de 100 palabras

```
public class WCMapper extends Mapper<LongWritable, Text, Text, IntWritable> {  
  
    @Override  
    protected void map(LongWritable key, Text value,  
        Context context)  
        throws IOException, InterruptedException {  
        String[] palabras = value.toString().split("([().,!?:;'\"-]|\\s)+");  
  
        if (palabras.length > 100)  
            context.write(new Text("Noticia Con más de 100 palabras"), new IntWritable(1));  
    }  
}
```

Bibliografía

4. IEEE. *Manual de estilo de documentos técnicos*. [En línea] [Citado el: 28 de Abril de 2010.] http://standards.ieee.org/guides/style/2009_Style_Manual.pdf.