# Un Análisis a MapReduce, Pig, HCatalog y Oozie en términos de Atributos de Calidad

Luis F. Rivera
Departamento Académico de Tecnologías de Información y Comunicaciones (TICs)
Universidad Icesi
Cali, Colombia
Email: lfrivera@icesi.edu.co

## INTRODUCCIÓN

*Apache Hadoop* es un *framework* de código abierto para el almacenamiento y procesamiento de grandes volúmenes de datos [1]. Generalmente, Hadoop es considerado como un ecosistema, en el que habitan, entre otras, herramientas como *Apache Hive*, *Apache Pig*, y *Apache Oozie*, las cuales fueron concebidas con el objetivo de complementar los cuatro elementos principales del core de Hadoop (HDFS, MapReduce, YARN, y Common)[2].

El objetivo principal de este proyecto consiste en analizar un subconjunto de las herramientas pertenecientes al ecosistema de Hadoop, desde la perspectiva de la infraestructura computacional necesaria ponerlas en marcha y de los principales atributos de calidad involucrados en el uso de las mismas. Para este propósito, un escenario de pruebas controlado fue configurado usando la versión 5.10.1 de *Cloudera Manager* (CDH 5.10.1). Dicha configuración fue establecida como se muestra en la figura 1

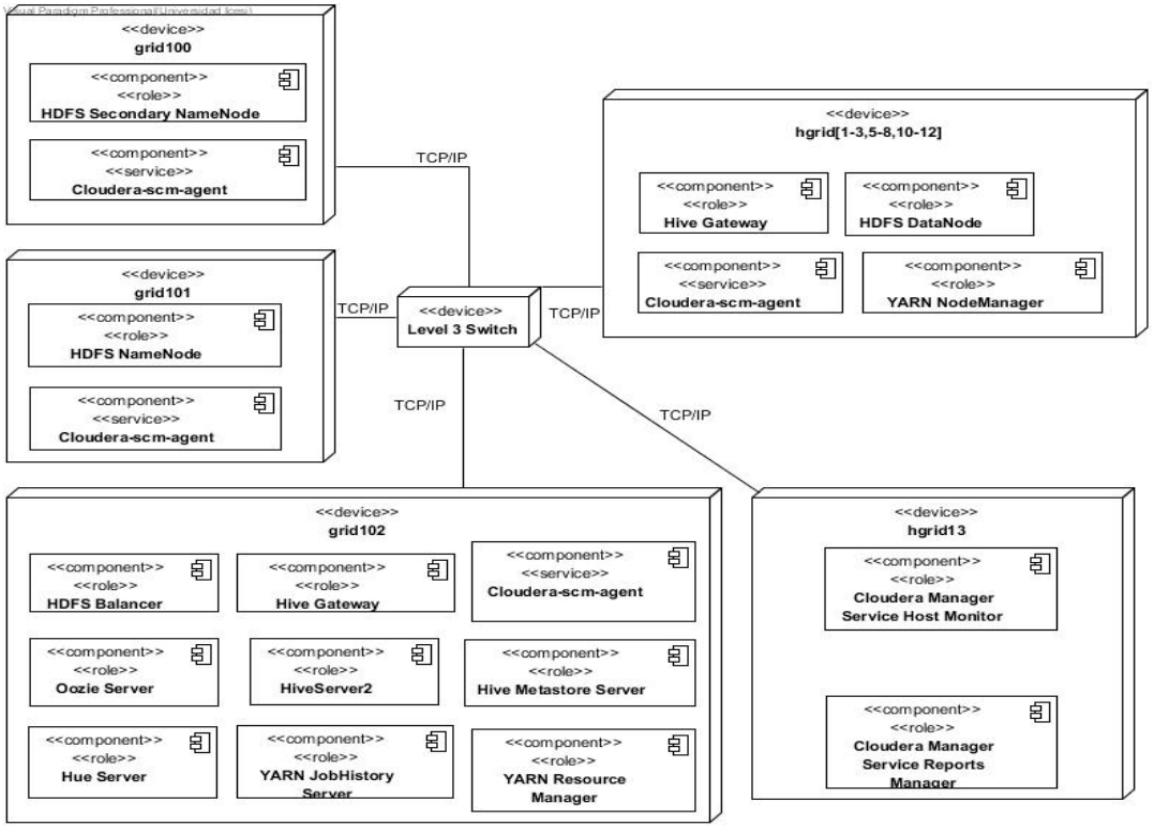



Figura 1. Diagrama de despliegue de la configuración de CDH 5.10.1 en el laboratorio *LIASON* de la Universidad Icesi.

[1]https://hortonworks.com/apache/hadoop/
[2]http://www.bmc.com/guides/hadoop-ecosystem.html

Cada uno de los casos de estudio de minería de datos presente en este proyecto se basa en los datos provistos por el *National Climatic Data Center* (NCDC). Dichos datos son recolectados por medio de sensores climáticos, los cuales recolectan información cada hora, de forma diaria, en distintas estaciones a lo largo del mundo. Dichos datos se encuentran registrados a través de lineas en archivos de texto. La figura 2 muestra la descripción del conjunto de datos mencionado previamente.

```
0057
332130   # USAF weather station identifier
99999    # WBAN weather station identifier
19500101 # observation date
0300     # observation time
4
+51317   # latitude (degrees x 1000)
+028783  # longitude (degrees x 1000)
FM-12
+0171    # elevation (meters)
99999
V020
320      # wind direction (degrees)
1        # quality code
N
0072
1
00450    # sky ceiling height (meters)
1        # quality code
C
N
010000   # visibility distance (meters)
1        # quality code
N
9
-0128    # air temperature (degrees Celsius x 10)
1        # quality code
-0139    # dew point temperature (degrees Celsius x 10)
1        # quality code
10268    # atmospheric pressure (hectopascals x 10)
1        # quality code
```

Figura 2. Descripción del conjunto de datos. Tomado de [1]

Vale la pena aclarar que en el presente documento se muestran los scripts y archivos de configuración más representativos de las distintas ejecuciones realizadas. Se omiten algunos elementos pues su contenido es muy similar a los ya presentados en este documento. Para una revisión más detallada de los distintos artefactos utilizados a lo largo de este proyecto, dirigirse a https://github.com/lfrivera/Oozie-Pig-HCatalog-Demos. Este repositorio contiene no solo el código fuente de las pruebas realizadas, sino también la evidencia (*output* y *screenshots*) de las mismas.

El resto del presente documento se encuentra distribuido como se muestra a continuación. En la sección 1 se describen formalmente el objetivo general y los objetivos objetivos específicos del proyecto. En la sección 2 se presentan los tipos de pruebas ejecutadas sobre las herramientas *MapReduce*, *Apache Pig*, y *Apache Hive* para entender la diferencia entre los tiempos de ejecución de las mismas. En la sección 3 se presentan las pruebas ejecutadas para determinar la escalabilidad de Pig. En la sección 4 se ilustran las distintas pruebas llevadas a cabo sobre las herramientas *Apache Pig* y *HCatalog* para verificar la posibilidad de extender las funcionalidades y capacidades provistas por *Pig*. En la sección 5 se detallan las pruebas realizadas sobre *Apache Ooozie*, las cuales buscan evidenciar la reusabilidad y mantenibilidad de las aplicaciones desarrolladas sobre esta herramienta. La sección 6 muestra los resultados de la ejecución de las pruebas descritas previamente. En la sección 7 se presentan las conclusiones del presente trabajo. Finalmente, la sección 8 muestra las posibilidades de trabajo futuro que se podrían desarrollar a partir de lo que aquí se presenta.

## I. Objetivos del proyecto

El objetivo general del proyecto consiste en analizar, en términos de atributos de calidad, algunas de las herramientas que conforman el ecosistema de Hadoop. Particularmente, en este proyecto se analizará MapReduce, Apache Pig, Apache Hive, HCatalog, y Apache Oozie desde la perspectiva de los atributos de calidad de desempeño (*performance*), reusabilidad (*reusability*), extensibilidad (*extensibility*), y mantenibilidad (*maintainability*). A continación se describen los objetivos específicos del proyecto y los atributos de calidad asociados a cada uno de estos.

1. Evidenciar la diferencia en los tiempos de ejecución de MapReduce, Apache Pig y Apache Hive. Atributo de calidad relacionado: *performance*.
2. Evidenciar la escalabilidad de Apache Pig. Atributo de calidad relacionado: *scalability*.
3. Evidenciar la extensibilidad de Apache Pig mediante HCatalog y su impacto en el desempeño de las ejecuciones. Atributos de calidad relacionados: *performance, extensibility*.
4. Evidenciar la reusabilidad y mantenibilidad de los workflows definidos con Apache Oozie. Atributos de calidad relacionados: *reusability, maintainability*.

## II. MapReduce, Pig y Hive

En esta sección se presentan las ejecuciones realizadas con MapReduce, Apache Pig, y Apache Hive para el cálculo de la temperatura máxima registrada por año para el objetivo específico 1 del presente proyecto.

### *II-A. Problema tratado*

Determinar la temperatura máxima registrada por año para los datos consignados en el dataset del NCDC.

### *II-B. Ejecución con MapReduce*

A continuación se detallan los pasos necesarios para ejecutar el programa *MaxTemperature* en su versión MapReduce-Java.

*1) Preparación:* Inicialmente, es necesario compilar el código fuente del programa *MaxTemperature* en su versión Java. Para hacer esto, es necesario clonar el repositorio del libro *Hadoop: The Definitive Guide*[3]. Una vez hecho lo anterior, se debe proceder a compilar los archivos fuente necesarios mediante *Maven*.Finalmente, la ruta del archivo .jar compilado deberá establecerse en una variable de entorno llamada *HADOOP_CLASSPATH*.

```
1 //Clonación del repositorio.
2 git clone https://github.com/tomwhite/hadoop-book.git
3
4 // Compilación del código fuente.
5 mvn package -DskipTests
6
7 // Definición del classpath de Hadoop.
8 export HADOOP_CLASSPATH=/home/sas6/Oozie-Pig-HCatalog-Demos/assets/hadoop-examples.jar
```

[3]Repositorio provisto por Tom White en https://github.com/tomwhite/hadoop-book

*2) Ejecución del programa MaxTemperature:* Una vez compilado el código fuente y definida la variable de entorno correspondiente, se procede a ejecutar el programa *MaxTemperature*.

```
1 hadoop MaxTemperature /user/hive/warehouse/weather_external/full_data.txt out_mr_300GB
```

*3) Seguimiento a la ejecución del programa:* Una vez iniciada la ejecución del programa, es posible monitorear el progreso del mismo por medio de la consola donde éste se ejecutó o por medio de la interfaz gráfica de YARN.

```
[root@grid102 Oozie-Pig-HCatalog-Demos]# hadoop MaxTemperature /user/hive/warehouse/weather_external/full_data.txt out_mr_300GB
17/07/22 14:23:45 INFO client.RMProxy: Connecting to ResourceManager at grid102.icesi.edu.co/192.168.161.43:8032
17/07/22 14:23:45 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and
 execute your application with ToolRunner to remedy this.
17/07/22 14:23:46 INFO input.FileInputFormat: Total input paths to process : 1
17/07/22 14:23:47 INFO mapreduce.JobSubmitter: number of splits:2314
17/07/22 14:23:47 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1500744570838_0005
17/07/22 14:23:47 INFO impl.YarnClientImpl: Submitted application application_1500744570838_0005
17/07/22 14:23:47 INFO mapreduce.Job: The url to track the job: http://grid102.icesi.edu.co:8088/proxy/application_1500744570838_0005/
17/07/22 14:23:47 INFO mapreduce.Job: Running job: job_1500744570838_0005
17/07/22 14:23:52 INFO mapreduce.Job: Job job_1500744570838_0005 running in uber mode : false
17/07/22 14:23:52 INFO mapreduce.Job:  map 0% reduce 0%
17/07/22 14:24:09 INFO mapreduce.Job:  map 1% reduce 0%
17/07/22 14:24:11 INFO mapreduce.Job:  map 2% reduce 0%
17/07/22 14:24:14 INFO mapreduce.Job:  map 3% reduce 0%
17/07/22 14:24:21 INFO mapreduce.Job:  map 4% reduce 0%
^[[15~17/07/22 14:24:27 INFO mapreduce.Job:  map 5% reduce 0%
```

Figura 3. Monitoreo de la ejecución del programa *MaxTemperature* en MapReduce por medio de la consola en donde éste se ejecutó.

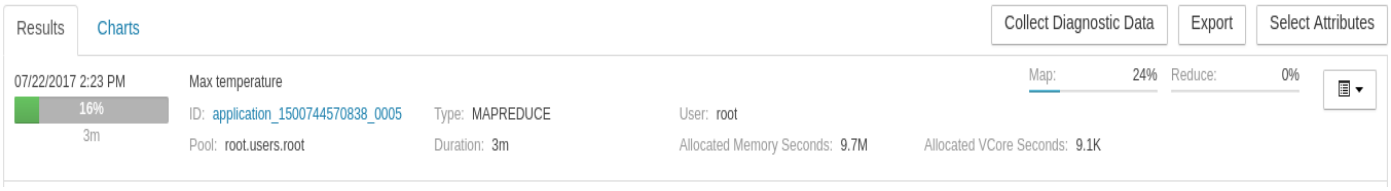



Figura 4. Monitoreo de la ejecución del programa *MaxTemperature* en MapReduce por medio de la interfaz de YARN.

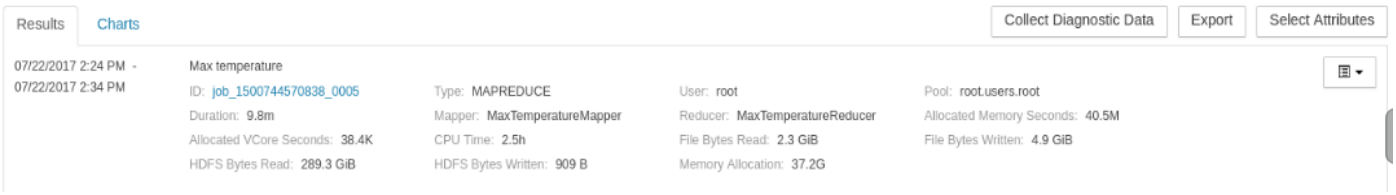



Figura 5. Ejecución finalizada.

### *II-C. Ejecución con Hive*

A continuación se detallan los pasos necesarios para ejecutar el programa *MaxTemperature* en su versión Apache Hive.

*1) Ejecución con la consola de Hive:* El siguiente script detalla la ejecución del programa *MaxTemperature* en su versión Apache Hive.

```
1 ADD jar /usr/lib/hive/lib/hive-contrib-1.1.0-cdh5.10.1.jar;
2 INSERT OVERWRITE DIRECTORY 'out_max_hive_300GB'
3 SELECT observation_date_year, MAX(air_temperature)
4 FROM weather_managed
5 WHERE air_temperature != 9999 AND at_quality_code IN (0,1,4,5,9)
6 GROUP BY observation_date_year;
```

*2) Seguimiento a la ejecución del programa:* El monitoreo de la ejecución del programa podrá realizarse a través de la interfaz gráfica de YARN, o por la información proporcionada por el Job History Server.

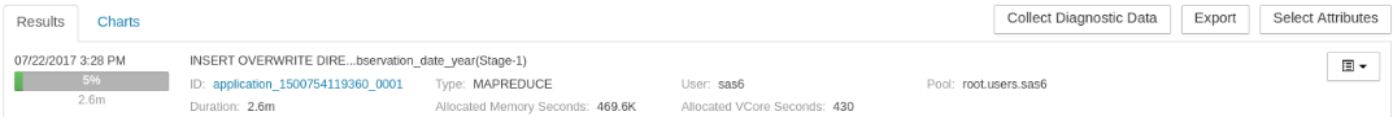



Figura 6. Monitoreo de la ejecución del programa *MaxTemperature* en Hive por medio de la interfaz de YARN.

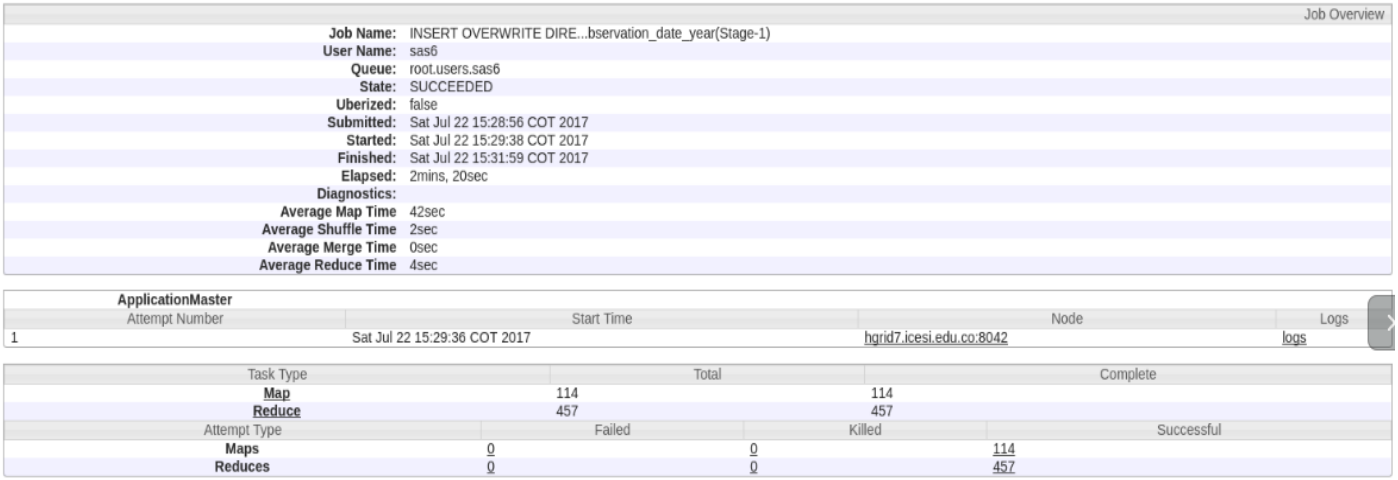



Figura 7. Ejecución finalizada, vista desde el Job History Server.

### *II-D. Ejecución con Pig*

A continuación se detallan los pasos necesarios para ejecutar el programa *MaxTemperature* en su versión Apache Pig.

*1) Definición del script en PigLatin:* El siguiente script contiene el código utilizado utilizado para ejecutar el programa *MaxTemperature* en su versión PigLatin. En las dos primeras lineas del script se detalla el uso de una UDF (*User defined function*), provista por Tom White, para la lectura de registros a partir de la definición de rangos de lectura para sus atributos. El contenido del script fue guardado en un archivo llamado *max-temp.pig*.

```
1 REGISTER $load_loc;
2 records = LOAD '$in_s1' USING com.hadoopbook.pig.CutLoadFunc('16-19,88-92,93-93') AS (year:int,
      temperature:int, quality:int);
3 filtered_records = FILTER records BY temperature != 9999 AND com.hadoopbook.pig.IsGoodQuality(
      quality);
4 grouped_records = GROUP filtered_records BY year;
5 max_temp = FOREACH grouped_records GENERATE group,MAX(filtered_records.temperature);
6 STORE max_temp INTO '$out_max';
```

*2) Definición de los parámetros del script:* Una vez definido el script en PigLatin, se procede a definir en un nuevo archivo los parámetros necesarios para la correcta ejecución del script. Los parámetros mencionados fueron guardados en un archivo llamado *max.param*.

```
1 # Load function location.
2 load_loc=/home/sas6/Oozie-Pig-HCatalog-Demos/assets/pig-examples.jar
3 # Input.
4 in_s1=/user/hive/warehouse/weather_external/full_data.txt
5 # Output.
6 out_max=out_max_pig
```

*3) Ejecución con Grunt en modo batch:* A continuación se detalla el comando utilizado para ejecutar el programa *MaxTemperature* mediante el modo *batch* de Grunt.

```
pig -param_file /home/sas6/Oozie-Pig-HCatalog-Demos/scripts/pig/300GB/max.param /home/sas6/Oozie-Pig-HCatalog-Demos/src/pig/max-temp.pig
```

*4) Seguimiento a la ejecución del programa:* El monitoreo de la ejecución del programa podrá realizarse a través de Grunt, por medio de la interfaz gráfica de YARN, o por la información proporcionada por el Job History Server.

```
2017-07-22 12:31:02,505 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Processing aliases
 filtered_records,grouped_records,max_temp,records
2017-07-22 12:31:02,505 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - detailed locations
: M: records[2,10],records[-1,-1],filtered_records[3,19],max_temp[5,11],grouped_records[4,18] C: max_temp[5,11],grouped_records[4,18] R:
 max_temp[5,11]
2017-07-22 12:31:02,590 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - 0% complete
2017-07-22 12:32:09,727 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - 4% complete
```

Figura 8. Monitoreo de la ejecución del programa *MaxTemperature* en Pig por medio de la consola Grunt desde donde se ejecutó.

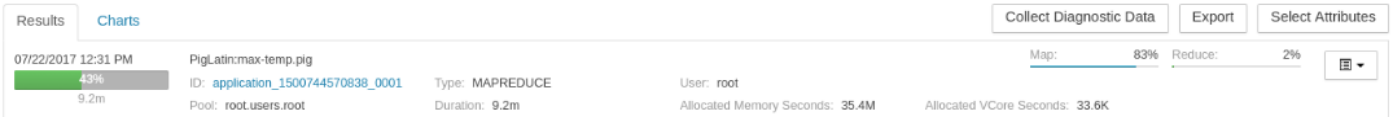



Figura 9. Monitoreo de la ejecución del programa *MaxTemperature* en Pig por medio de la interfaz de YARN.

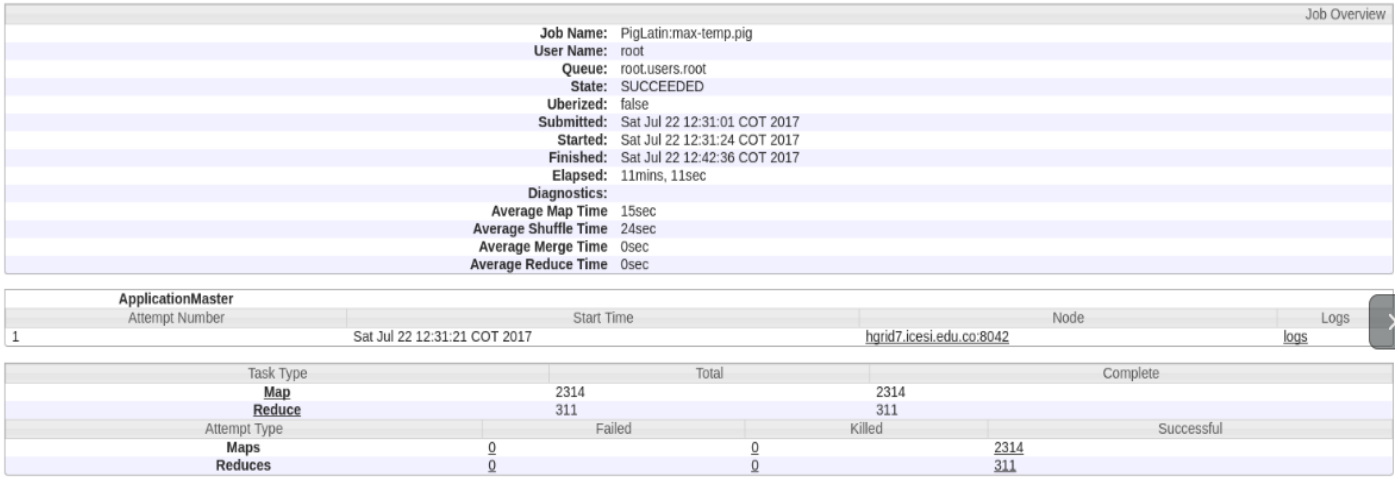



Figura 10. Ejecución finalizada, vista desde el Job History Server.

## III. Escalabilidad de Pig

A continuación se detallan los pasos necesarios para ejecutar el programa *AverageTemperatureCount* con Apache Pig para el objetivo específico 2 del presente proyecto

### *III-A. Problema tratado*

Determinar la temperatura promedio registrada por año y el número de estaciones meteorológicas utilizadas para su cálculo, teniendo como base los datos consignados en el dataset del NCDC.

### *III-B. Definición del script en PigLatin*

El siguiente script contiene el código utilizado utilizado para ejecutar el programa *AverageTemperatureCount* en su versión PigLatin. En las dos primeras lineas del script se detalla el uso de una UDF (*User defined function*), provista por Tom White, para la lectura de registros a partir de la definición de rangos de lectura para sus atributos. El contenido del script fue guardado en un archivo llamado *avg-count-temp-scalability.pig*.

```
1 REGISTER $load_loc;
2 records = LOAD '$in_s1' USING com.hadoopbook.pig.CutLoadFunc('16-19,88-92,93-93') AS (year:int,
      temperature:int, quality:int);
3 filtered_records = FILTER records BY temperature != 9999 AND com.hadoopbook.pig.IsGoodQuality(
      quality);
4 grouped_records = GROUP filtered_records BY year;
5 mean_count_temp = FOREACH grouped_records GENERATE group,AVG(filtered_records.temperature),COUNT(
      filtered_records);
6 STORE mean_count_temp INTO '$out_max';
```

### *III-C. Definición de los parámetros del script*

Una vez definido el script en PigLatin, se procede a definir en un nuevo archivo los parámetros necesarios para la correcta ejecución del script. Los parámetros mencionados fueron guardados en un archivo llamado *avg-count-temp-scalability.param*.

```
1 # Load function location.
2 load_loc=/home/sas6/Oozie-Pig-HCatalog-Demos/assets/pig-examples.jar
3 # Input.
4 in_s1=/user/hive/warehouse/weather_external/full_data.txt
5 # Output.
6 out_max=out_scalabity_pig
```

### *III-D. Ejecución con Grunt en modo batch*

A continuación se detalla el comando utilizado para ejecutar el programa *AverageTemperatureCount* mediante el modo *batch* de Grunt.

```
1 pig -param_file /home/sas6/Oozie-Pig-HCatalog-Demos/scripts/pig/300GB/avg-count-scalabity.param /
      home/sas6/Oozie-Pig-HCatalog-Demos/src/pig/avg-count-temp-scalability.pig
```

### *III-E. Seguimiento a la ejecución del programa*

El monitoreo de la ejecución del programa podrá realizarse a través de Grunt, por medio de la interfaz gráfica de YARN, o por la información proporcionada por el Job History Server.

```
2017-07-29 12:23:51,178 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapred.reduce.tasks is deprecated. Instead, use mapreduce.job.reduces
2017-07-29 12:23:52,357 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - creating jar file Job3535540628769436508.jar
2017-07-29 12:23:56,604 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - jar file Job3535540628769436508.jar created
2017-07-29 12:23:56,604 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapred.jar is deprecated. Instead, use mapreduce.job.jar
2017-07-29 12:23:56,625 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - Setting up single store job
2017-07-29 12:23:56,632 [main] INFO  org.apache.pig.data.SchemaTupleFrontend - Key [pig.schematuple] is false, will not generate code.
2017-07-29 12:23:56,632 [main] INFO  org.apache.pig.data.SchemaTupleFrontend - Starting process to move generated code to distributed cache
2017-07-29 12:23:56,632 [main] INFO  org.apache.pig.data.SchemaTupleFrontend - Setting key [pig.schematuple.classes] with classes to deserialize []
2017-07-29 12:23:56,708 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - 1 map-reduce job(s) waiting for submission.
2017-07-29 12:23:56,710 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker.http.address is deprecated. Instead, use mapreduce.jobtracker.http.address
2017-07-29 12:23:56,723 [JobControl] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at grid102.icesi.edu.co/192.168.161.43:8032
2017-07-29 12:23:56,749 [JobControl] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2017-07-29 12:23:57,676 [JobControl] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2017-07-29 12:23:58,100 [JobControl] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths (combined) to process : 2314
2017-07-29 12:23:58,672 [JobControl] INFO  org.apache.hadoop.mapreduce.JobSubmitter - number of splits:2314
2017-07-29 12:23:58,897 [JobControl] INFO  org.apache.hadoop.mapreduce.JobSubmitter - Submitting tokens for job: job_1501340077565_0010
2017-07-29 12:23:59,157 [JobControl] INFO  org.apache.hadoop.yarn.client.api.impl.YarnClientImpl - Submitted application application_1501340077565_0010
2017-07-29 12:23:59,219 [JobControl] INFO  org.apache.hadoop.mapreduce.Job - The url to track the job: http://grid102.icesi.edu.co:8088/proxy/application_1501340077565_0010/
2017-07-29 12:23:59,220 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - HadoopJobId: job_1501340077565_0010
2017-07-29 12:23:59,220 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Processing aliases filtered_records,grouped_records,mean_count_temp
,records
2017-07-29 12:23:59,220 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - detailed locations: M: records[2,10],records[-1,-1],filtered_record
s[3,19],mean_count_temp[5,18],grouped_records[4,18] C: mean_count_temp[5,18],grouped_records[4,18] R: mean_count_temp[5,18]
2017-07-29 12:23:59,276 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - 0% complete
```

Figura 11. Monitoreo de la ejecución del programa *AverageTemperatureCount* en Pig por medio de la consola Grunt desde donde se ejecutó.

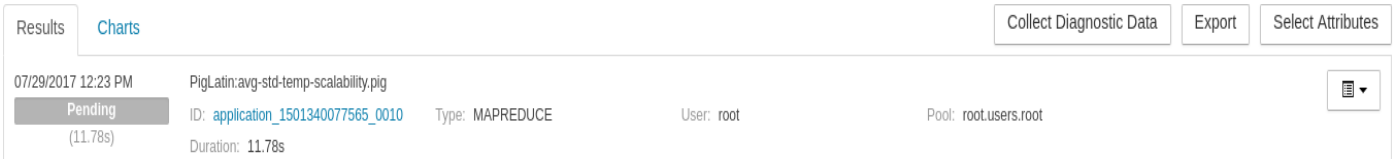



Figura 12. Monitoreo de la ejecución del programa *AverageTemperatureCount* en Pig por medio de la interfaz de YARN.

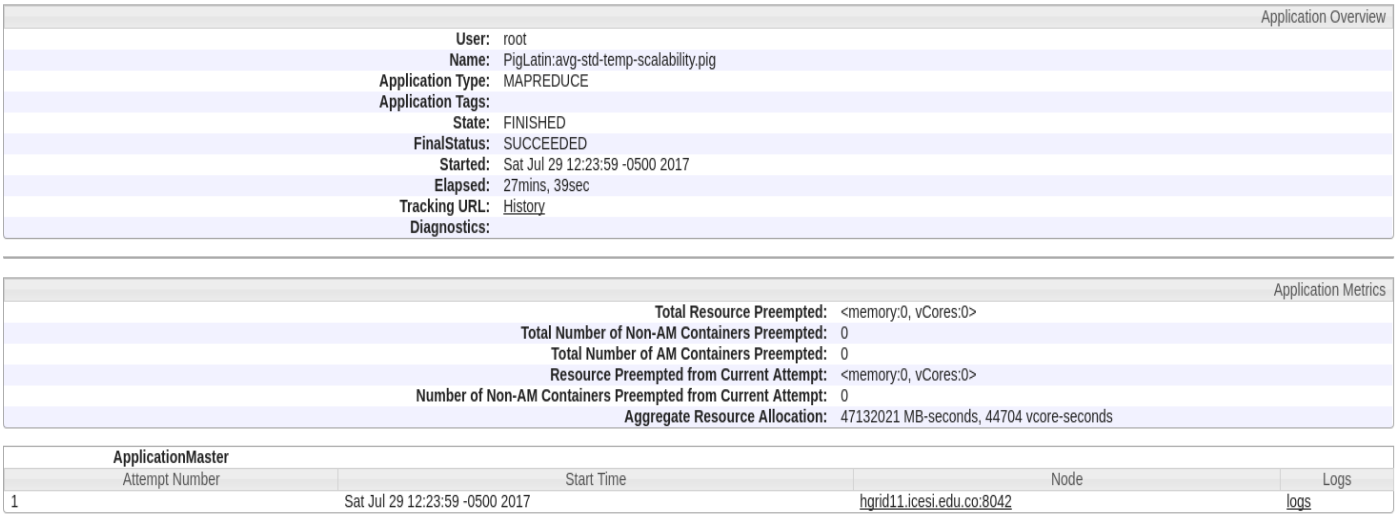



Figura 13. Ejecución finalizada, vista desde el Job History Server.

## IV. Pig y HCatalog

En esta sección se presentan las ejecuciones realizadas con Apache Pig y HCatalog para el objetivo específico 3 del presente proyecto.

### *IV-A. Problemas tratados*

*1) Problema 1::* Determinar la temperatura promedio registrada por año para los años mayores a 1950, teniendo como base los datos consignados en el dataset del NCDC.

*2) Problema 2::* Determinar la temperatura promedio registrada para el año 1910, teniendo como base los datos consignados en el dataset del NCDC.

### *IV-B. Estrategia problema 2: Apache Pig y HCatalog*

A continuación se detallan los pasos necesarios para resolver el problema 2, usando Pig y HCatalog.

*1) Definición del script en PigLatin:* El siguiente script contiene el código utilizado utilizado para ejecutar el programa *AverageTemperature* en PigLatin. En las primera lineas del script se detalla el uso de una UDF (*User defined function*), provista por Tom White, para minimizar el código necesario en el filtrado de registros. En la segunda línea se define el uso de HCatalog para el acceso al *metastore* de Hive. El contenido del script fue guardado en un archivo llamado *avg-1910-temp-hcatalog.pig*.

```
1 REGISTER $load_loc;
2 records = LOAD '$table_name' USING org.apache.hive.hcatalog.pig.HCatLoader();
3 filtered_records = FILTER records BY air_temperature != 9999 AND
    com.hadoopbook.pig.IsGoodQuality((int)at_quality_code) AND observation_date_year == 1910;
4 grouped_records = GROUP filtered_records BY observation_date_year;
5 avg_temp = FOREACH grouped_records GENERATE group,AVG(filtered_records.air_temperature);
6 DUMP avg_temp;
```

*2) Definición de los parámetros del script:* Una vez definido el script en PigLatin, se procede a definir en un nuevo archivo los parámetros necesarios para la correcta ejecución del script. Los parámetros mencionados fueron guardados en un archivo llamado *avg-1910-hcatalog-managed-pb.param*.

```
# Load function location.
load_loc=/home/sas6/Oozie-Pig-HCatalog-Demos/assets/pig-examples.jar
# Input.
table_name=weather_managed_pb
```

*3) Ejecución con Grunt en modo batch:* A continuación se detalla el comando utilizado para ejecutar el programa *Average-Temperature* mediante el modo *batch* de Grunt.

```
pig -param_file /home/sas6/Oozie-Pig-HCatalog-Demos/scripts/pig/300GB/
    avg_1910_hcatalog_managed_pb.param /home/sas6/Oozie-Pig-HCatalog-Demos/src/pig/avg-1910-temp-
    hcatalog.pig
```

*4) Seguimiento a la ejecución del programa:* El monitoreo de la ejecución del programa podrá realizarse a través de Grunt, por medio de la interfaz gráfica de YARN, o por la información proporcionada por el Job History Server.

```
2017-07-29 14:12:39,124 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - creating jar file Job5155842404413470773.jar
2017-07-29 14:12:43,838 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - jar file Job5155842404413470773.jar created
2017-07-29 14:12:43,838 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapred.jar is deprecated. Instead, use mapreduce.job.jar
2017-07-29 14:12:43,855 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - Setting up single store job
2017-07-29 14:12:43,861 [main] INFO  org.apache.pig.data.SchemaTupleFrontend - Key [pig.schematuple] is false, will not generate code.
2017-07-29 14:12:43,861 [main] INFO  org.apache.pig.data.SchemaTupleFrontend - Starting process to move generated code to distributed cache
2017-07-29 14:12:43,861 [main] INFO  org.apache.pig.data.SchemaTupleFrontend - Setting key [pig.schematuple.classes] with classes to deserialize []
2017-07-29 14:12:43,931 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - 1 map-reduce job(s) waiting for submission.
2017-07-29 14:12:43,931 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker.http.address is deprecated. Instead, use mapreduce.jobtracker.http.address
2017-07-29 14:12:43,938 [JobControl] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at grid102.icesi.edu.co/192.168.161.43:8032
2017-07-29 14:12:43,963 [JobControl] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2017-07-29 14:12:44,508 [JobControl] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapred.input.dir is deprecated. Instead, use mapreduce.input.fileinputformat.inputdir
2017-07-29 14:12:44,530 [JobControl] INFO  org.apache.hadoop.mapred.FileInputFormat - Total input paths to process : 1
2017-07-29 14:12:44,540 [JobControl] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths (combined) to process : 1
2017-07-29 14:12:44,763 [JobControl] INFO  org.apache.hadoop.mapreduce.JobSubmitter - number of splits:1
2017-07-29 14:12:44,913 [JobControl] INFO  org.apache.hadoop.mapreduce.JobSubmitter - Submitting tokens for job: job_1501354211633_0002
2017-07-29 14:12:45,202 [JobControl] INFO  org.apache.hadoop.yarn.client.api.impl.YarnClientImpl - Submitted application application_1501354211633_0002
2017-07-29 14:12:45,244 [JobControl] INFO  org.apache.hadoop.mapreduce.Job - The url to track the job: http://grid102.icesi.edu.co:8088/proxy/application_1501354211633_0002/
2017-07-29 14:12:45,244 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - HadoopJobId: job_1501354211633_0002
2017-07-29 14:12:45,244 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Processing aliases avg_temp,filtered_records,grouped_records,record
s
2017-07-29 14:12:45,244 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - detailed locations: M: records[2,10],filtered_records[3,19],avg_tem
p[5,11],grouped_records[4,18] C: avg_temp[5,11],grouped_records[4,18] R: avg_temp[5,11]
2017-07-29 14:12:45,297 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - 0% complete
```

Figura 14. Monitoreo de la ejecución del programa *AverageTemperature* en Pig por medio de la consola Grunt desde donde se ejecutó.

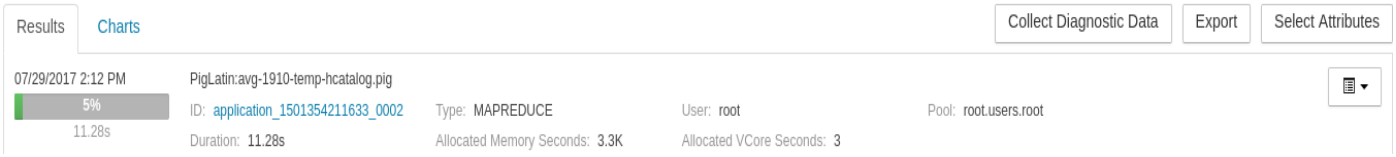
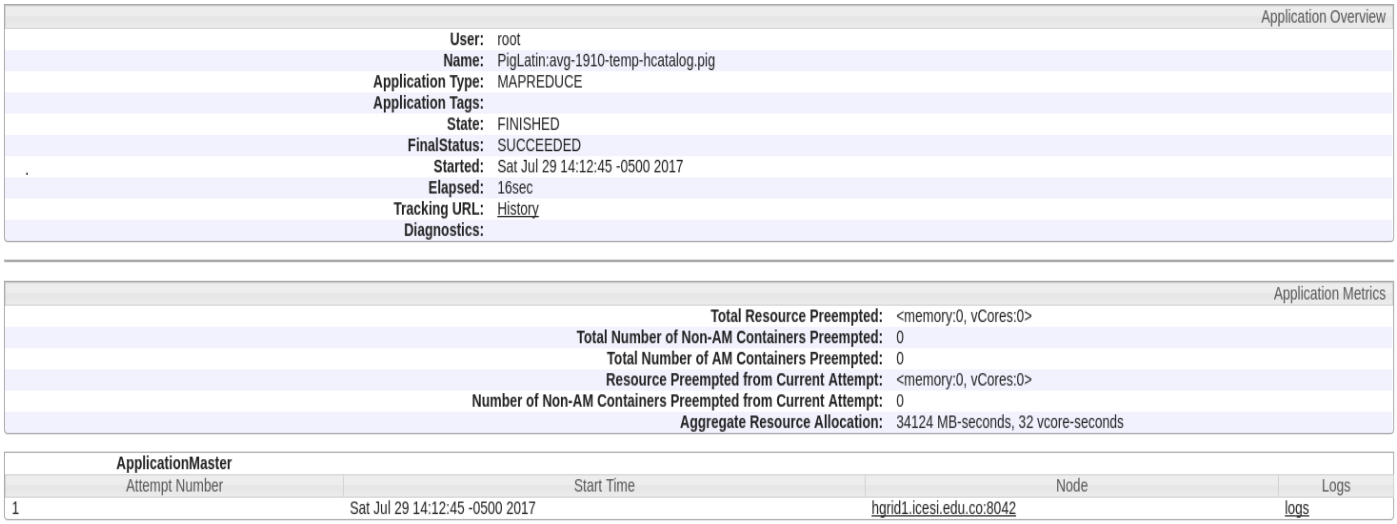



Figura 15. Monitoreo de la ejecución del programa *AverageTemperature* en Pig por medio de la interfaz de YARN.



Application Overview

| | |
|---|---|
| User: | root |
| Name: | PigLatin:avg-1910-temp-hcatalog.pig |
| Application Type: | MAPREDUCE |
| Application Tags: | |
| State: | FINISHED |
| FinalStatus: | SUCCEEDED |
| Started: | Sat Jul 29 14:12:45 -0500 2017 |
| Elapsed: | 16sec |
| Tracking URL: | History |
| Diagnostics: | |

Application Metrics

| | |
|---|---|
| Total Resource Preempted: | <memory:0, vCores:0> |
| Total Number of Non-AM Containers Preempted: | 0 |
| Total Number of AM Containers Preempted: | 0 |
| Resource Preempted from Current Attempt: | <memory:0, vCores:0> |
| Number of Non-AM Containers Preempted from Current Attempt: | 0 |
| Aggregate Resource Allocation: | 34124 MB-seconds, 32 vcore-seconds |

ApplicationMaster

| Attempt Number | Start Time | Node | Logs |
|---|---|---|---|
| 1 | Sat Jul 29 14:12:45 -0500 2017 | hgrid1.icesi.edu.co:8042 | logs |

Figura 16. Ejecución finalizada, vista desde el Job History Server.

## V. Oozie

En esta sección se presentan las ejecuciones realizadas con Apache Oozie para el objetivo específico 4 del presente proyecto.

### *V-A. Problemas tratados*

*1) Problema 1::* Determinar la temperatura máxima promedio registrada en cada día del año para cada estación metereológica, teniendo como base los datos consignados en el dataset del NCDC. Ejemplo: 029070-99999 0101 -68.

*2) Problema 2::* Mejorar los tiempos de ejecución del problema 1.

### *V-B. Estrategia general para el Problema 1*

La estrategia para abordar el primer problema consiste en realizar dos pasos *stages*

1. Calcular la temperatura máxima diaria para cada llave estación-fecha. Ejemplo: 29070-99999 19020101 -94.
2. Calcular el promedio de las temperaturas diarias para cada llave estación-mes. Ejemplo: 029070-99999 0101 -68.

### *V-C. Estrategia problema 2: Apache Oozie, Hive y Pig*

A continuación se detallan los pasos necesarios para resolver el problema 2, usando Oozie, Hive y Pig.

*1) Definición de tabla:* Para facilitar el cambio del primer *stage* por un script de Hive (inicialmente está escrito en Pig), se decidió darle un nuevo formato a los datos del NCDC. A continuación se presenta el script utilizado para crear una nueva tabla de Hive con el nuevo formato de los datos.

```
1  ADD jar /usr/lib/hive/lib/hive-contrib-1.1.0-cdh5.10.1.jar;
2
3  CREATE TABLE weather_managed_lf
4  AS SELECT
5  CAST(CONCAT(usaf,'-',wban) AS STRING),  CAST(CONCAT(observation_date_year,observation_date_month,
       observation_date_day) AS INT),
6          CAST(air_temperature AS INT),
7          CAST(at_quality_code AS INT)
8      FROM weather_external;
```

*2) Definición de los parámetros del workflow:* Así como Pig y Hive, Oozie permite la definición de parámetros para la ejecución de *workflows*. A continuación se presenta el archivo *workflow.properties*, el cual contiene los parámetros necesarios para la correcta ejecución de la aplicación de Oozie.

```
nameNode=hdfs://grid101.icesi.edu.co:8020
resourceManager=grid102.icesi.edu.co:8032
oozie.wf.application.path=${nameNode}/user/${user.name}/mean-max-temp-workflow-hive-pig
# Output for stage 1.
outS1Dir=${nameNode}/user/${user.name}/hive-pig-s1-out
# Input for stage 2.
inS2Dir=${outS1Dir}
# Output for stage 2.
outS2Dir=${nameNode}/user/${user.name}/hive-pig-s2-out
oozie.use.system.libpath=true
oozie.libpath=${nameNode}/user/oozie/share/lib
```

*3) Definición del primer stage:* A continuación se presenta el archivo *s1Hive.sql*, el cual contiene la definición del primer *stage* del *workflow* en cuestión.

```
ADD jar /usr/lib/hive/lib/hive-contrib-1.1.0-cdh5.10.1.jar;
INSERT OVERWRITE DIRECTORY '${outS1}'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '  '
SELECT `_c0`, `_c1`, MAX(air_temperature) FROM weather_managed_lf
WHERE at_quality_code IN (0,1,4,5,9) AND air_temperature <> 9999
GROUP BY `_c0`,`_c1`;
```

*4) Definición del segundo stage:* A continuación se presenta el archivo *s2.pig*, el cual contiene la definición del segundo *stage* del *workflow* en cuestión.

```
records = LOAD '$inS2' AS (station:chararray,date:chararray,temperature:int);
formated_records = FOREACH records GENERATE station, SUBSTRING(date, 4, 8), temperature;
grouped_records = GROUP formated_records BY ($0,$1);
mean_station_day_month = FOREACH grouped_records GENERATE FLATTEN(group),AVG(formated_records.
    temperature);
STORE  mean_station_day_month INTO '$outS2';
```

*5) Definición del workflow:* A continuación se presenta el archivo *workflow.xml*, el cual define el *workflow* que será ejecutado con Oozie.

```
<workflow-app xmlns="uri:oozie:workflow:0.4" name="meanMax">
  <start to="hiveS1"/>
<action name="hiveS1">
    <hive xmlns="uri:oozie:hive-action:0.2">
      <job-tracker>${resourceManager}</job-tracker>
      <name-node>${nameNode}</name-node>
      <prepare>
        <delete path="${outS1Dir}"/>
      </prepare>
      <job-xml>hive-site.xml</job-xml>
    <configuration>
        <property>
            <name>mapred.job.queue.name</name>
            <value>default</value>
        </property>
    </configuration>
      <script>s1Hive.sql</script>
      <param>outS1=${outS1Dir}</param>
    </hive>
    <ok to="pigS2"/>
    <error to="fail"/>
</action>
<action name="pigS2">
    <pig>
      <job-tracker>${resourceManager}</job-tracker>
```

```
      <name-node>${nameNode}</name-node>
      <prepare>
        <delete path="${outS2Dir}"/>
      </prepare>
      <script>s2.pig</script>
      <argument>-param</argument>
      <argument>inS2=${inS2Dir}</argument>
      <argument>-param</argument>
      <argument>outS2=${outS2Dir}</argument>
    </pig>
    <ok to="end"/>
    <error to="fail"/>
  </action>
  <kill name="fail">
    <message>Pig failed, error message[${wf:errorMessage(wf:lastErrorNode())}]
    </message>
  </kill>
  <end name="end"/>
</workflow-app>
```

*6) Ejecución del workflow:* A continuación se presentan los pasos necesarios para ejecutar el *workflow* de Oozie.

```
#Remove oozie app.
hadoop fs -rm -r mean-max-temp-workflow-hive-pig

#Copy oozie app.
hadoop fs -put /home/sas6/Oozie-Pig-HCatalog-Demos/src/oozie/hive-pig/mean-max-temp-workflow-
    hive-pig mean-max-temp-workflow-hive-pig

#Oozie server
export OOZIE_URL="http://grid102.icesi.edu.co:11000/oozie"

#Run job
oozie job -config /home/sas6/Oozie-Pig-HCatalog-Demos/src/oozie/hive-pig/mean-max-temp-workflow-
    hive-pig/workflow.properties -run
```

### *V-D. Seguimiento a la ejecución del workflow*

El monitoreo de la ejecución del workflow podrá realizarse a través del servidor de Oozie.

Documentation

Oozie Web Console

Workflow Jobs | Coordinator Jobs | Bundle Jobs | SLA | System Info

All Jobs | Active Jobs | Done Jobs | Custom Filter

| | Job Id | Name | Status |
|---|---|---|---|
| 1 | 0000000-170729095526026-oozie-oozi-W | meanMax | SUCCEEDED |
| 2 | 0000006-170728082247468-oozie-oozi-W | meanMax | SUCCEEDED |
| 3 | 0000003-170728082247468-oozie-oozi-W | meanMax | KILLED |
| 4 | 0000002-170728082247468-oozie-oozi-W | meanMax | KILLED |
| 5 | 0000001-170728082247468-oozie-oozi-W | meanMax | KILLED |
| 6 | 0000000-170728082247468-oozie-oozi-W | meanMax | KILLED |
| 7 | 0000000-170725211423854-oozie-oozi-W | meanMax | KILLED |
| 8 | 0000002-170725203804533-oozie-oozi-W | meanMax | SUCCEEDED |
| 9 | 0000001-170725203804533-oozie-oozi-W | meanMax | KILLED |
| 10 | 0000001-170725202404256-oozie-oozi-W | meanMax | KILLED |
| 11 | 0000000-170725203804533-oozie-oozi-W | meanMax | KILLED |
| 12 | 0000000-170725202404256-oozie-oozi-W | meanMax | KILLED |
| 13 | 0000000-170718170456121-oozie-oozi-W | pig-app-hue-script | KILLED |
| 14 | 0000001-170718162418785-oozie-oozi-W | pig-app-hue-script | KILLED |
| 15 | 0000000-170718162418785-oozie-oozi-W | pig-app-hue-script | KILLED |
| 16 | 0000000-170718111837653-oozie-oozi-W | pig-app-hue-script | KILLED |

Figura 17. Lista de *jobs* de Oozie ejecutados.

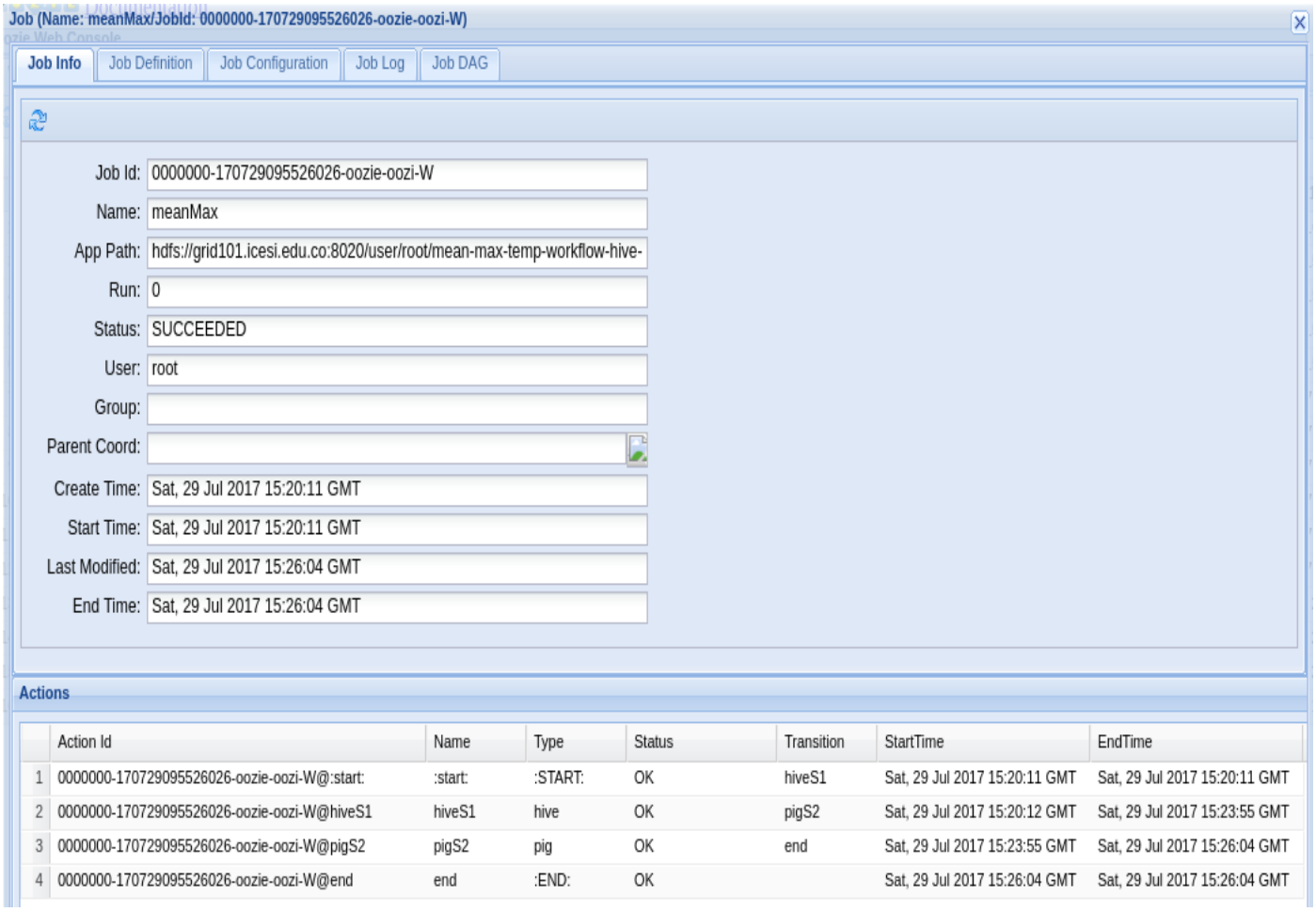



Figura 18. Detalle del *job* de Oozie ejecutado.

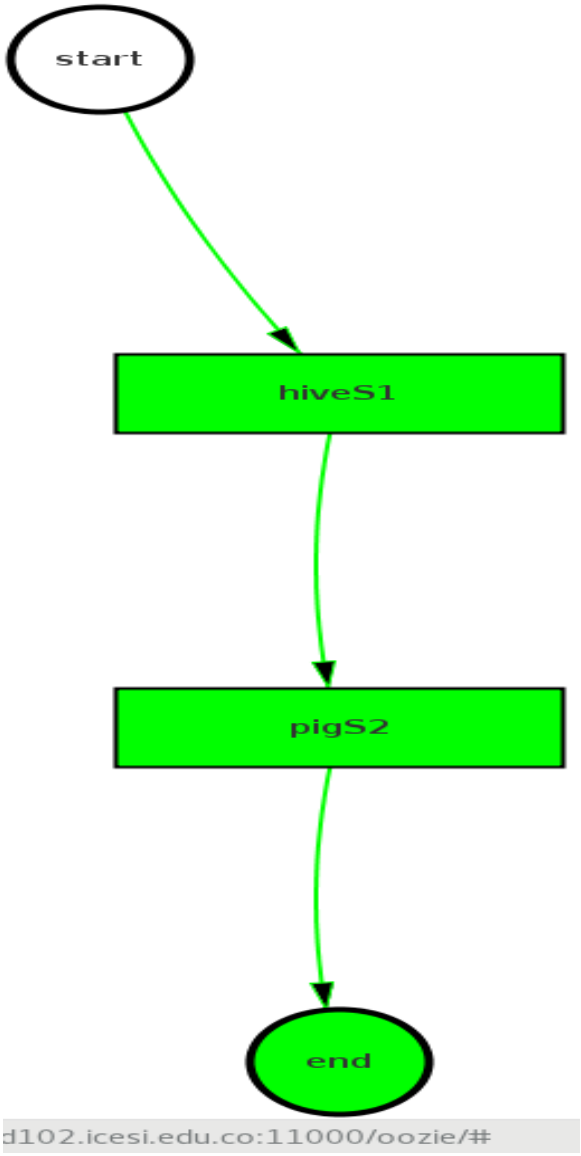



Figura 19. Ejecución finalizada, vista desde el grafo acíclico dirigido generado desde la UI del servidor de Oozie.

## VI. Resultados

En esta sección se presentan los resultados obtenidos a partir de las distintas ejecuciones realizadas.

### *VI-A. Resultados objetivo específico 1: MapReduce vs Hive (external tables) vs Pig*

A continuación se ilustran los distintos tiempos de ejecución obtenidos a partir de las pruebas realizadas con el programa *MaxTemperature* en las versiones MapReduce-Java, Pig, y Hive para el total y el 10 % de los datos del NCDC. La figura 20 compara los tiempos de ejecución del programa *MaxTemperature* en la versión MapReduce-Java con el total y el 10 % de los datos. La figura 21 compara los tiempos de ejecución del programa *MaxTemperature* en la versión Pig con el total y el 10 % de los datos. La figura 22 compara los tiempos de ejecución del programa *MaxTemperature* en versión Hive en su forma nativa (Hive server) y desde Hue (*managed tables*) con el total de los datos. La figura 23 compara los tiempos de ejecución del programa *MaxTemperature* en las versiones MapReduce-Java, Hive (*external tables*), y Pig con el total de los datos.

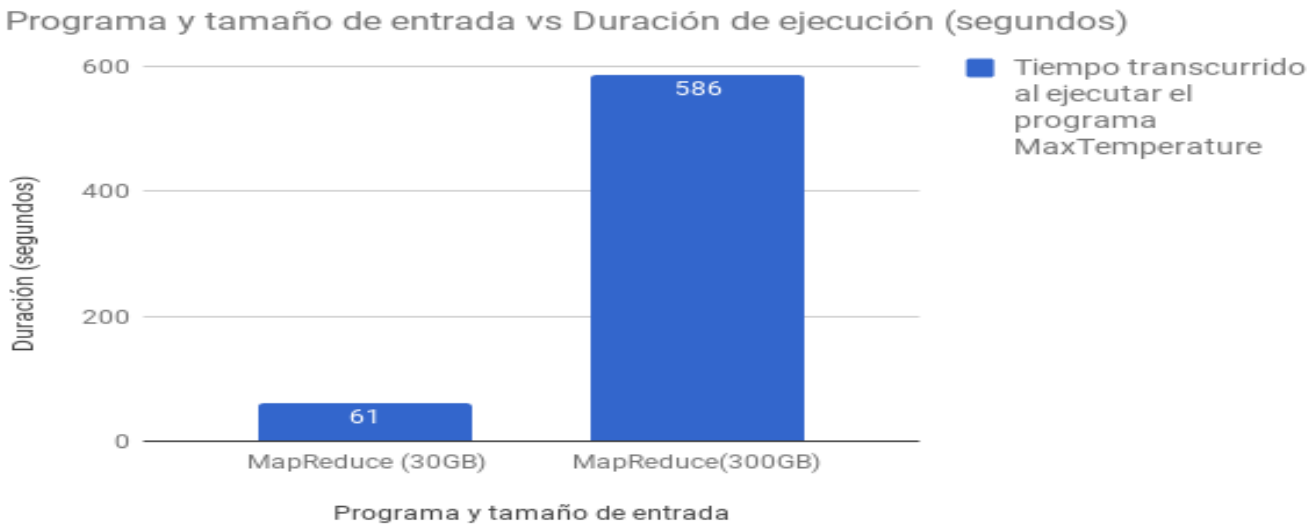



Figura 20. Comparación de MapReduce-Java con el total y el 10 % de los datos.

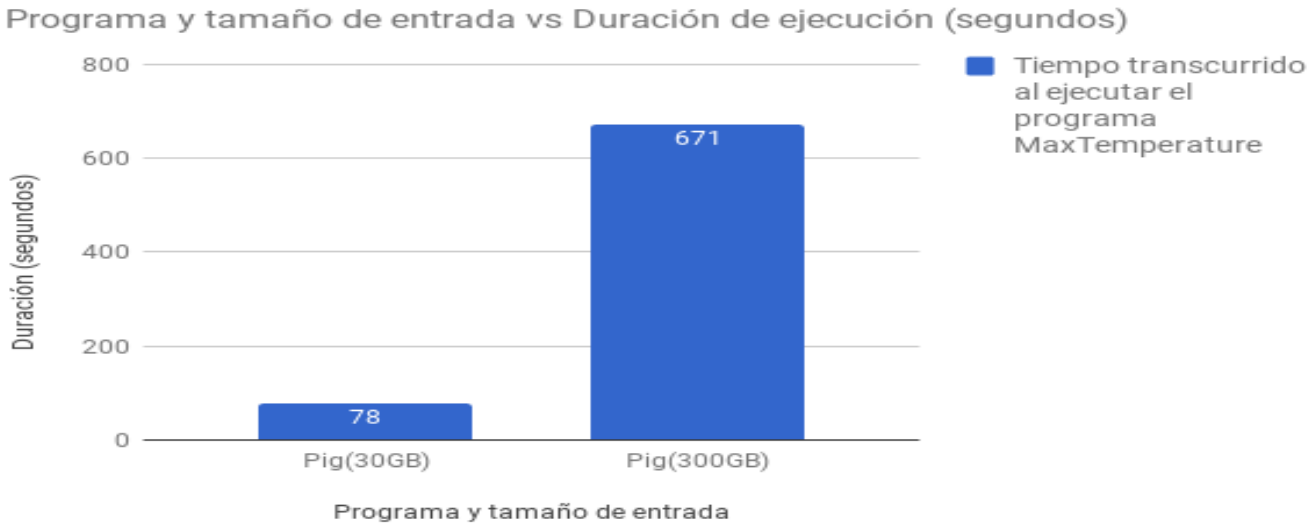



Figura 21. Comparación de Pig con el total y el 10 % de los datos.

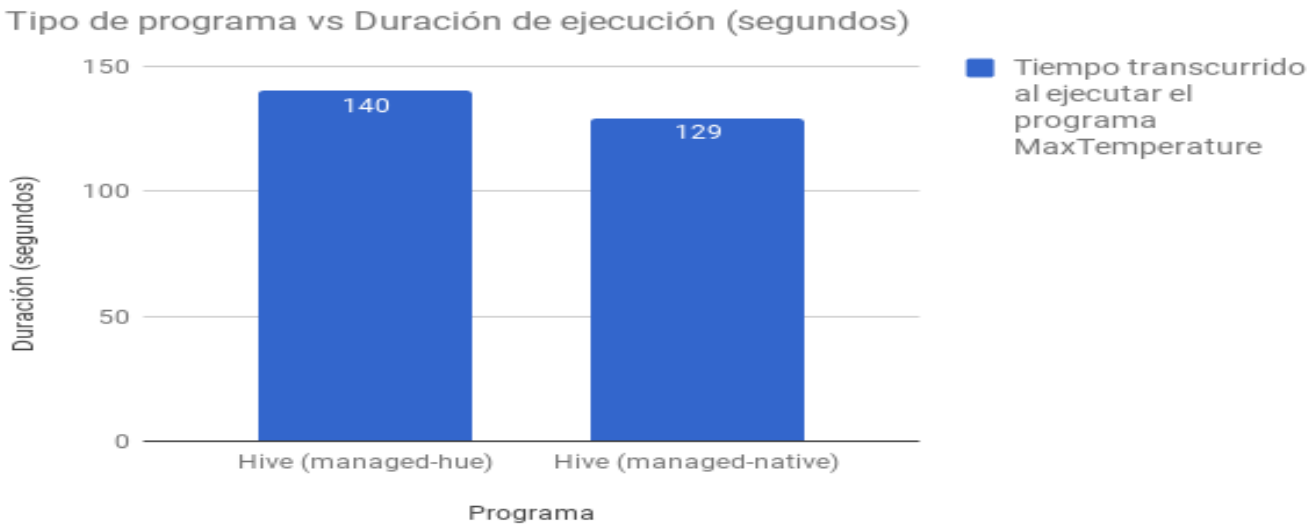



Figura 22. Comparación de ejecución en Hive de forma nativa y desde Hue con el total de los datos.

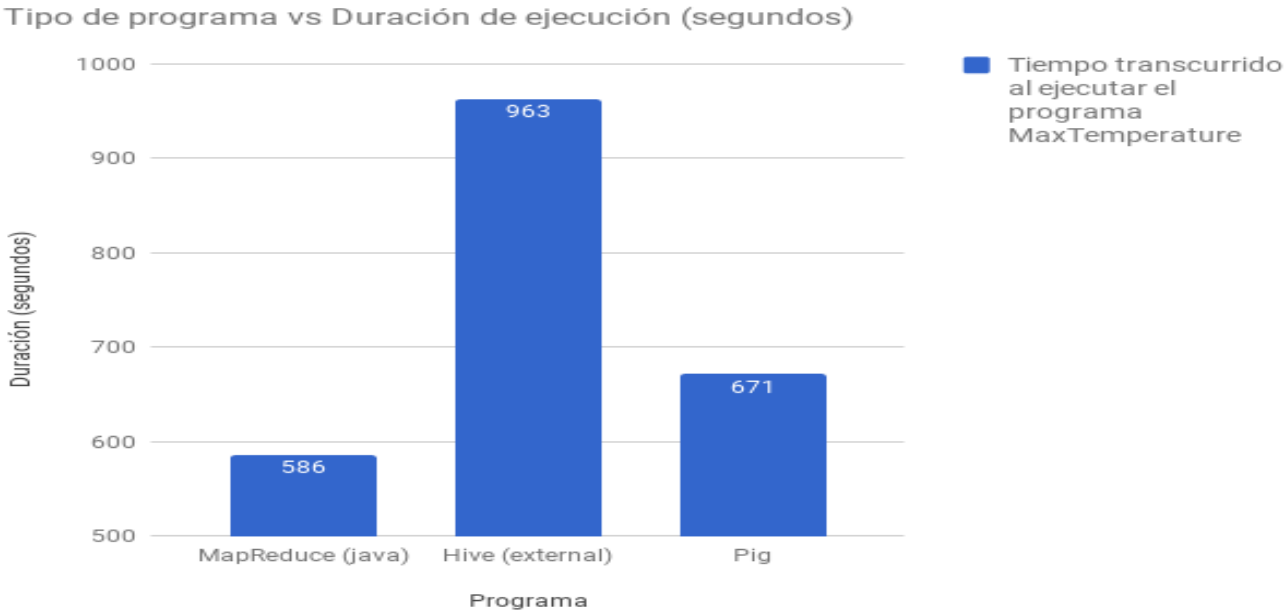



Figura 23. Comparación de ejecución del programa *MaxTemperature* desde MapReduce, Hive (*external table*), y Pig con el total de los datos.

### *VI-B. Resultados objetivo específico 2: Escalabilidad en Pig*

A continuación se ilustran los distintos tiempos de ejecución obtenidos a partir de las pruebas realizadas con el programa *AverageTemperatureCount* para medir la escabilidad de Pig. Para esto, dos distintos enfoques fueron usados: el primero consistía en deshabilitar, mediante la interfaz gráfica de YARN, NodeManagers y trabajar con el total de DataNodes (10 en nuestro caso); el segundo consistía en agregar nuevos nodos al cluster y ejecutar los scripts cada vez que esto sucedía, lo que significaba que las instancias de NodeManagers activas eran igual a los nodos en los que se encontraban replicados los datos. La figura 24 muestra los resultados del primer enfoque, mientras que la figura 25 muestra los del segundo.

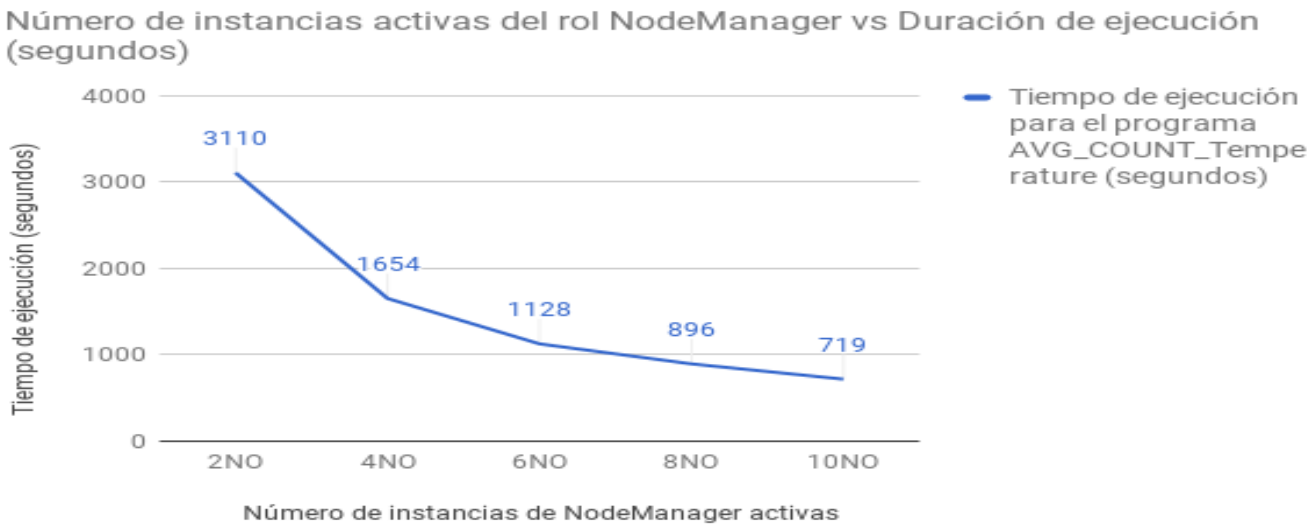



Figura 24. Escalabilidad de Pig en términos del procesamiento de datos (Pig/YARN).

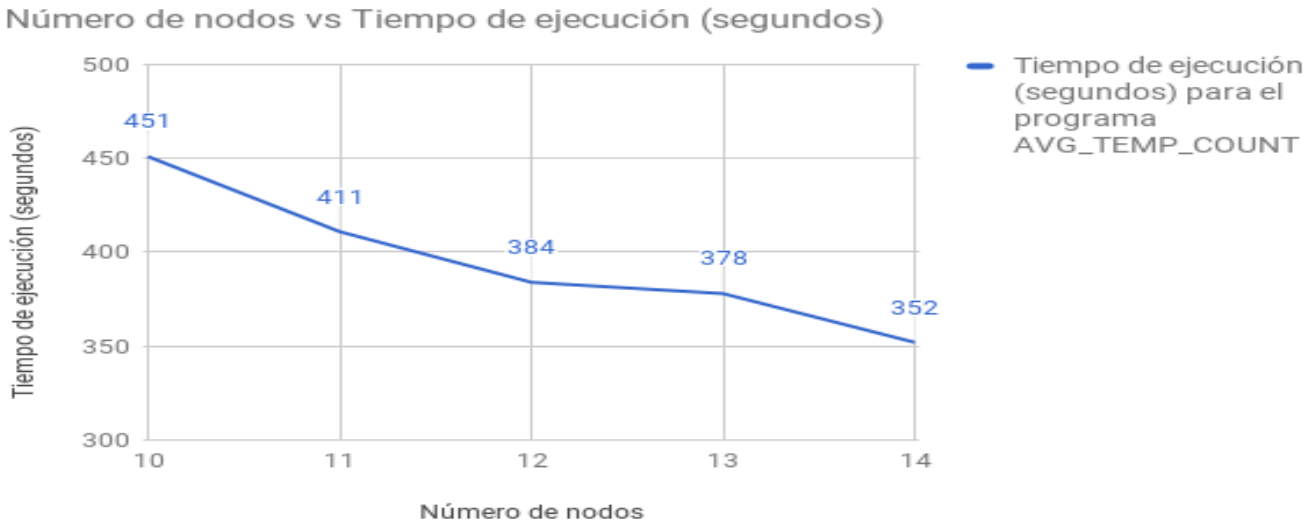



Figura 25. Escalabilidad de Pig (Pig/HDFS/YARN).

Para analizar los resultados evidenciados en las gráficas anteriores, se transformaron los tiempos de ejecución (eje vertical) a una escala logarítmica y se uso la función *lm* del lenguaje de programación R para comprobar si en efecto la escalabilidad era logarítmica. A continuación se presentan los resultados arrojados.

*1) Análisis de la escalabilidad de Pig siguiendo el enfoque 1:*

- **Función de transformación:** $log(y)$
- **p-value:** 0.002416

*2) Análisis de la escalabilidad de Pig siguiendo el enfoque 2:*

- **Función de transformación:** $log(y)$
- **p-value:** 0.004018

### *VI-C. Resultados objetivo específico 3: Extensibilidad de Pig con HCatalog:*

A continuación se ilustran los distintos tiempos de ejecución y tamaños de lectura de datos desde HDFS obtenidos a partir de las pruebas realizadas con el programa *AverageTemperature* para medir la extensibilidad de Pig con HCatalog y su influencia en los tiempos de ejecución. La figura 26 muestra la comparación en los tiempos de ejecución para el programa *AverageTemperature-1950* de Pig con HCatalog (con una tabla no particionada), Pig con HCatalog (con una tabla particionada), y de Pig sin HCatalog. La figura 27 muestra la comparación de la lectura de datos desde HDFS para el programa *AverageTemperature-1950* de Pig con HCatalog (con una tabla no particionada), Pig con HCatalog (con una tabla particionada), y de Pig sin HCatalog. La figura 28 muestra la comparación en los tiempos de ejecución para el programa *AverageTemperature-1910* de Pig con HCatalog (con una tabla no particionada), Pig con HCatalog (con una tabla particionada), y de Pig sin HCatalog. La figura 29 muestra la comparación de la lectura de datos desde HDFS para el programa *AverageTemperature-1910* de Pig con HCatalog (con una tabla no particionada), Pig con HCatalog (con una tabla particionada), y de Pig sin HCatalog.

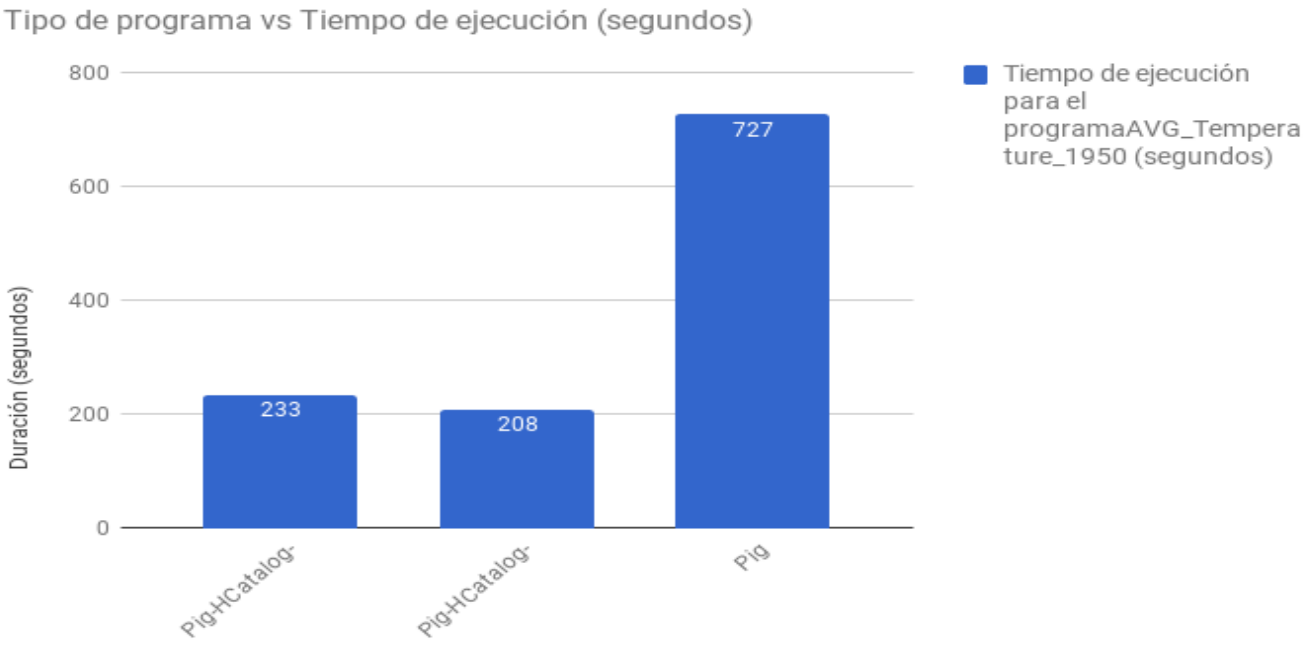



Figura 26. Tiempos de ejecución obtenidos para el programa *AverageTemperature-1950*.

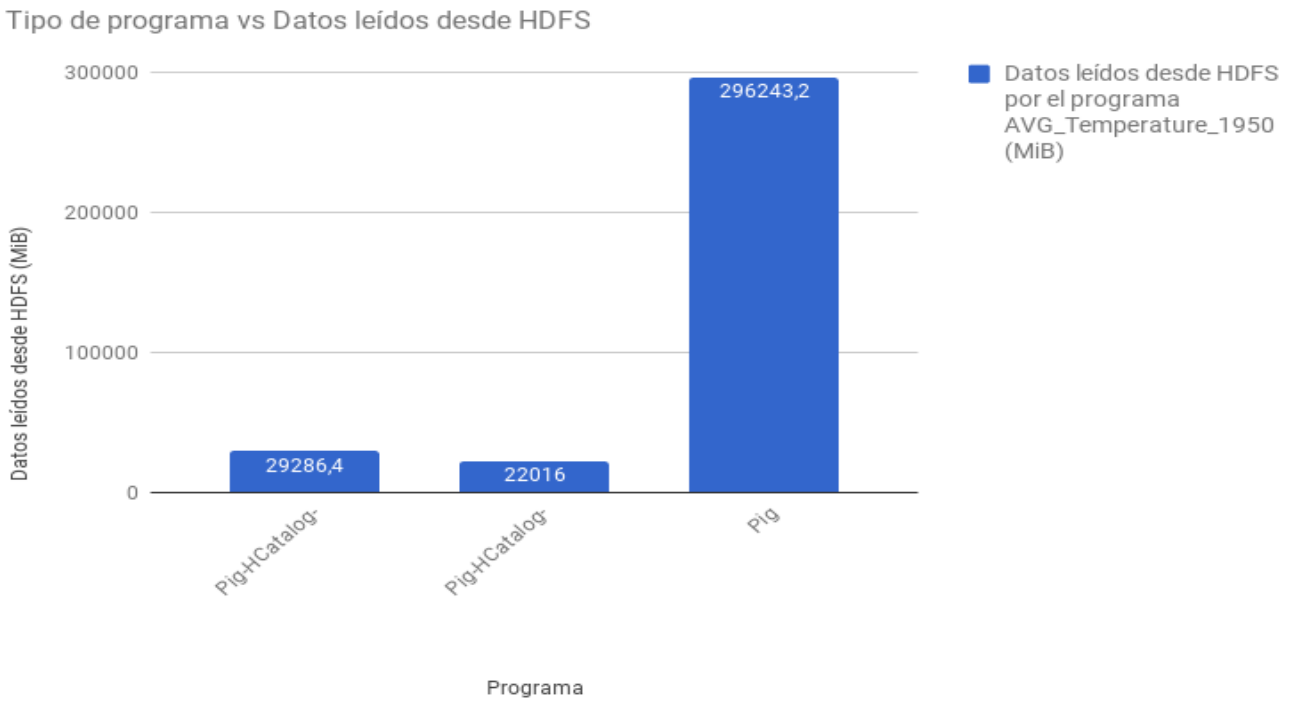



Figura 27. Lectura de datos desde HDFS para el programa *AverageTemperature-1950*.

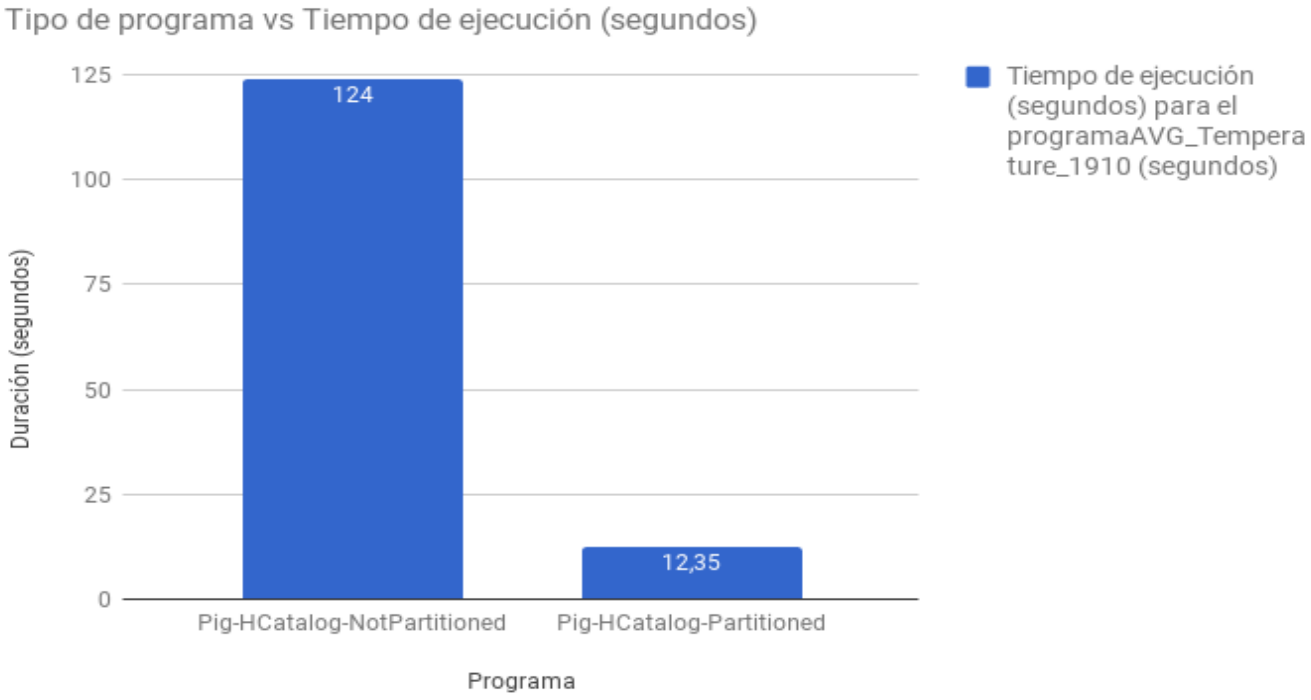



Figura 28. Tiempos de ejecución obtenidos para el programa *AverageTemperature-1910*.

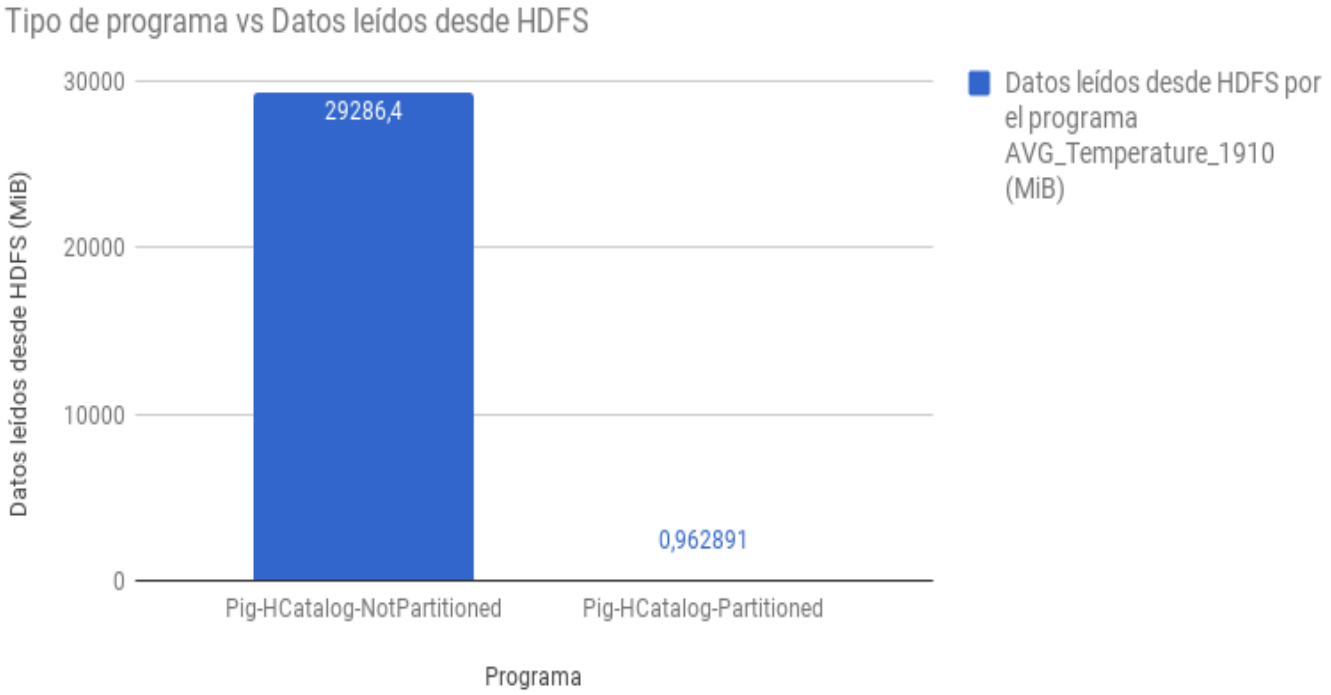



Figura 29. Lectura de datos desde HDFS para el programa *AverageTemperature-1910*.

### *VI-D. Resultados objetivo específico 4: Reusabilidad y mantebilidad en Oozie*

La reusabilidad y mantenibilidad de los *workflows* de Oozie se pudo comprobar al lograr intercambiar fácilmente el primer *stage* del mismo. La figura 30 ilustra los tiempos de ejecución obtenidos a partir de las pruebas realizadas con las definiciones del *workflow* en sus versiones: Pig-Pig y Hive-Pig.

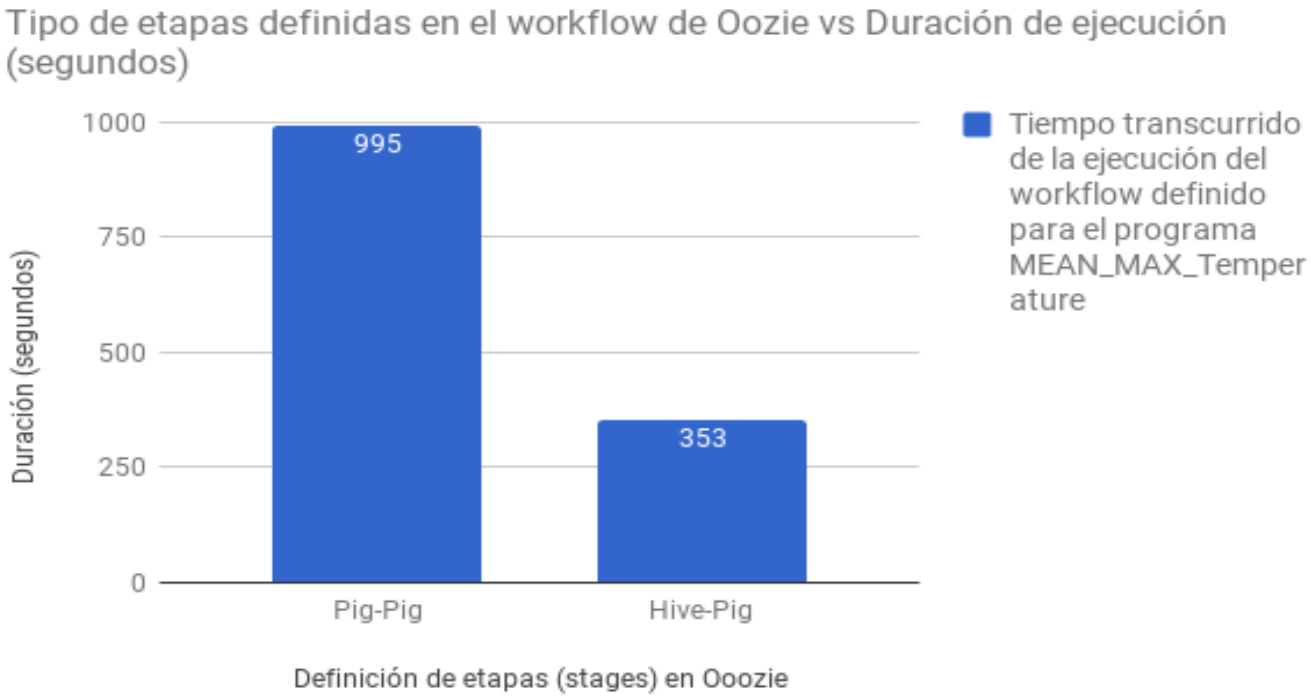



Figura 30. Comparación del *workflow* de Oozie con la combinación Pig-Pig y Hive-Pig.

## VII. Conclusiones

Las pruebas realizadas en este proyecto indican en un principio que:

- Respecto al objetivo específico 1, en términos de tiempos de ejecución, Hive (con *external tables*) ejecuta las actividades en un tiempo significativamente mayor que MapReduce y Pig. Además, como era de esperarse, Pig realiza procesamientos en los datos en un tiempo mayor a MapReduce en su versión Java.
- Respecto al objetivo específico 1, Hue no parece adicionar un *overhead* importante sobre las consultas realizadas en Hive, sin embargo, aunque leve, si hay un costo de realizar la planeación de las ejecuciones con Oozie cuando se ejecutan las consultas desde Hue.
- Respecto al objetivo específico 2, aparentemente, la escalabilidad de Pig no es del todo lineal sino logarítmica.
- Respecto al objetivo específico 3, Pig es un framework extensible que puede aprovechar las ventajas de Hive (*metastore* y tablas particionadas) a través de HCatalog. Sin embargo, no parece haber compatibilidad con tablas externas de Hive.
- Respecto al objetivo específico 4, la reusabilidad y mantenibilidad de los workflows de Oozie se pudo comprobar a través del presente proyecto, pues fue relativamente sencillo actualizar el *workflow* inicial para dar soporte a las nuevas necesidades (mejoras en el rendimiento a partir de las ventajas de Hive).

Es importante aclarar que las conclusiones listadas previamente son el reflejo de las ejecuciones de las distintas tecnologías del ecosistema de Hadoop que fueron tratadas en este proyecto. No deben ser consideradas como conclusiones definitivas, pues se necesitaría un mayor número de pruebas y estudio más completo de las distintas herramientas utilizadas para los fines de este proyecto.

## VIII. Trabajo futuro

A continuación se enuncian aquellas actividades que, por una u otra razón, no fueron tratadas en el contexto del presente proyecto, sin embargo, podrían hacer parte de futuras discusiones.

1. Comparar el rendimiento de Apache Oozie frente a otras alternativas como Apache Falcon o Apache NiFi.
2. Explorar cómo se podrían aprovechar las tablas de Hive con buckets desde Pig, mediante HCatalog.

## Referencias

[1] T. White, *Hadoop: The Definitive Guide*. O'Reilly Media, Inc., 2012.