Marco Antonio ALMEIDA PAZMIÑO[1],
Juan Alfonso LARA TORRALBO[2] y David LIZCANO CASAS[3]

Frameworks para la gestión, el almacenamiento y la preparación de grandes volúmenes de datos Big Data

Sumario

Fecha de entrada: 11-12-2014
Fecha de aceptación: 30-12-2014

Extracto:

Los sistemas meteorológicos, como es el Sistema Mundial de Información Global de la Organización Meteorológica Mundial, necesitan almacenar diferentes tipos de imágenes, datos y archivos. Big Data y su modelo 3V puede proporcionar una solución adecuada para resolver este problema. Este tutorial presenta algunos conceptos en torno al *framework* Hadoop, la implementación y estándar de facto de Big Data, y la forma de almacenar los datos semiestructurados generados por las estaciones meteorológicas automáticas usando este *framework*. Finalmente, se presenta un método formal para generar informes del tiempo utilizando los *frameworks* que conforman el ecosistema de Hadoop.

Palabras claves: Big Data, Hadoop, HDFS, Organización Meteorológica Mundial, Sistema de Información Global, internet de las cosas.

[1] M. A. Almeida Pazmiño, desarrollador en Apache ServiceMix (ESB), Apache Camel, Hadoop, HBase y Pig y estudiante de la Universidad a Distancia de Madrid (udima).
[2] J. A. Lara Torralbo, profesor de la Universidad a Distancia de Madrid (udima).
[3] D. Lizcano Casas, profesor de la Universidad a Distancia de Madrid (udima).

Frameworks for management, storage and preparation of large data volumes Big Data

Abstract:

Weather systems like the World Meteorological Organization´s Global Information System need to store different kinds of images, data and files. Big Data and its 3V paradigm can provide a suitable solution to solve this problem. This tutorial presents some concepts around the Hadoop framework, de facto standard implementation of Big Data, and how to store semi-estructured data generated by automatic weather stations using this framework. Finally, a formal method to generate weather reports using Hadoop´s ecosystem frameworks is presented.

Keywords: Big Data, Hadoop, HDFS, World Meteorological Organization, Global Observing System, Internet of Things.

1. INTRODUCCIÓN

Las empresas de hoy están mejorando sus servicios mediante el análisis exhaustivo sobre los datos que almacenan. Para lograr este objetivo, estas empresas han tenido que tratar con diversos tipos de problemas durante la historia, como son administrar datos que pueden ser estructurados[4], semiestructurados[5] y no estructurados[6]; el crecimiento acelerado de los datos, pudiendo llegar al orden de los *terabytes* por día; el espacio de memoria necesario para procesar esta cantidad de datos puede ser excesivo dependiendo del equipo desginado o, si no lo es, y este tiene espacio suficiente, podría tomarle un tiempo considerable proporcionar una respuesta. Toda esta problemática ha sido abordada por el paradigma Big Data con su famoso modelo 3V, que significa velocidad, volumen y variedad. Es decir, un sistema debe proporcionar la velocidad suficiente para poder procesar con baja latencia un volumen considerable (orden de los *gigabytes*, *terabytes*, *pentabytes*, etc.) de una variedad de datos (texto, audio, vídeo, *posts* en blogs, fotos, *tweets*, *e-mails*, imágenes satelitales, entre otros) con menor costo en memoria que soluciones previas.

Hadoop es ahora el estándar de facto para el almacenamiento en paralelo de datos a gran escala y para el procesamiento distribuido de los mismos. Sus partes constitutivas son Hadoop Distributed File System (HDFS) y MapReduce. Con HDFS se pueden almacenar datos a gran escala a través de un sistema de directorios distribuido en (centenas de) clústeres de computadores. MapReduce, por otra parte, permite especificar de qué manera los datos se almacenarán o se recuperarán (operaciones *map* y *reduce*, respectivamente) del HDFS y también del procesamiento en paralelo cercano a los datos[7] (Holmes, 2014).

[4] http://en.wikipedia.org/wiki/Data_structure

[5] http://en.wikipedia.org/wiki/Semi-structured_data

[6] http://en.wikipedia.org/wiki/Unstructured_data

[7] Procesamiento cercano a los datos se refiere a que la operación requerida se realiza en el computador donde se encuentran almacenados los datos en cuestión.

La red de instrumentos meteorológicos más grande se conoce como el Sistema de Información Global [Global Observing System (GOS)], que pertenece a la Organización Meteorológica Mundial (WMO). A través del GOS, la WMO utiliza y publica de forma gratuita estos datos meteorológicos para producir observaciones y predicciones precisas para el desarrollo sostenible de las sociedades y para ayudar a salvar vidas y bienes, a proteger los recursos y el medioambiente.

El paradigma Big Data, mediante su *framework* Hadoop, solventaría todos los requerimientos necesarios por el GOS: la facilidad del incremento del espacio de almacenamiento y procesamiento para datos meteorológicos gracias a su configuración en clúster, la posibilidad de ejecutar predicciones en tiempo real (para crear sistemas de alerta temprana) y estadísticas haciendo uso de algoritmos de aprendizaje de máquina o cualquier otro algoritmo diseñado a medida de acuerdo a la necesidad.

Este tutorial tiene como principal objetivo mostrar cómo se puede aprovechar la arquitectura del HDFS y MapReduce de Hadoop para el almacenamiento y procesamiento de datos meteorológicos. Habla también de algunos proyectos de Apache que forman parte del ecosistema de Hadoop para ayudar a los usuarios al tratamiento de este tipo de datos.

El paradigma Big Data, mediante su *framework* Hadoop, solventaría todos los requerimientos necesarios por el Global Observing System (GOS)

2. MARCO TEÓRICO

2.1. El paradigma Big Data

Big Data comprende un gran volumen y variedad de datos que requieren ser procesados a una alta velocidad para mejorar la toma de decisiones de las empresas, su optimización y mejora continua.

Esta definición implícitamente nos hace pensar en la existencia de un ciclo de vida para el modelo 3V. Este ciclo de vida se muestra en la figura 1 y se detalla a continuación:

Figura 1. **Ciclo de vida de Big Data**

Fuente: elaboración propia.

- **Fase de adquisición.** Corresponde a los datos provenientes de fuentes internas y externas (de propiedad de la empresa o adquiridos) en una variedad de formatos y fuentes que permiten satisfacer las necesidades empresariales identificadas. Las fuentes pueden ser de dos tipos: huellas digitales generadas por los humanos y los datos de máquina. Las huellas digitales generadas por los humanos pueden ser un clic, una interacción, un *post* en algún foro o red social, un *e-mail*, entre otros. Al contrario, los datos de máquina provienen principalmente de sensores, radares, satélites, cámaras de vídeo, *routers*, servidores, entre otros. En esta fase puede ser necesario el empleo de un ESB[8] (*enterprise service bus*) o un *framework* de integración[9].
- **Fase de almacenamiento.** Esta fase permite, de forma general, la agregación, consolidación, control de calidad, persistencia y mantenimiento de los datos de la empresa. Estos procesos son conocidos como el «gobierno de los datos». Un estándar bien conocido para

[8] http://es.wikipedia.org/wiki/Enterprise_service_bus

[9] http://camel.apache.org/

implementar este gobierno se denomina MDM (*master data management*).

- **Fase de procesamiento.** Para obtener los resultados esperados, esta fase se divide en subfases, las mismas que conforman el ciclo de vida del procesamiento. Estas subfases son las siguientes: identificar el problema, seleccionar datos y preprocesar los datos (Prajapati, 2013).

  La subfase «identificar el problema» permite a la empresa identificar ciertos aspectos importantes para su mejor rendimiento. Por ejemplo, si se trata de una empresa dedicada a la meteorología y lo que se desea es obtener algoritmos para predicción del clima, es importante asignar atributos a los datos que provienen de los sensores según el nivel de confianza (dato correcto, inválido, no utilizable, entre otros).

  En la subfase «seleccionar datos», se deben seleccionar las fuentes de los datos relacionados con el problema en cuestión y también se deben especificar los atributos de los datos necesarios. Para continuar con el ejemplo anterior, para crear el algoritmo de predicción del clima, las fuentes de los datos serán las estaciones meteorológicas automáticas y el único atributo requerido será el dato correcto.

  Una vez ya seleccionados los datos, la subfase «preprocesar datos» aplica operaciones sobre los datos, como limpieza, agregación, clasificación y formato.

- **Fase de análisis.** En esta fase se aplican técnicas de minería de datos[10] y los algoritmos de aprendizaje de máquina[11] sobre los datos. Los algoritmos de aprendizaje de máquina más utilizados son los siguientes: regresión, clasificación, agrupación y recomendación.

[10] http://es.wikipedia.org/wiki/Miner%C3%ADa_de_datos

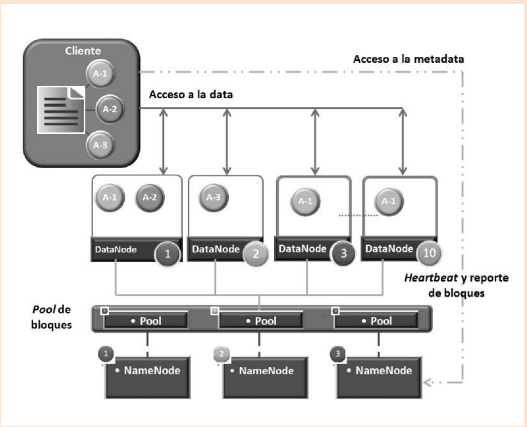
[11] http://es.wikipedia.org/wiki/Aprendizaje_autom%C3%A1tico

2.2. Arquitectura del Sistema de Directorios Distribuido de Hadoop: el HDFS

El HDFS de Hadoop es un sistema de directorios que permite almacenar y gestionar cualquier tipo de datos, ya sean del tipo estructurados, semiestructurados o no estructurados, de manera distribuida en un clúster escalable de servidores.

El HDFS de Hadoop se diferencia de los sistemas de archivos convencionales debido a que antes de proceder a almacenar un archivo, un cliente HDFS lo fragmenta en bloques. Cada fragmento es replicado según el factor de replicación que el usuario especifique (por defecto, tres) a través de múltiples nodos seleccionados aleatoriamente como se observa en la figura 2. La tolerancia a fallos del HDFS permite que el archivo sea recuperado en cualquier momento en caso de que se pierda cualquiera de los nodos. A cada uno de estos nodos se los conoce como DataNode dentro del contexto de Hadoop. Además de la tolerancia a fallos, otra ventaja que brinda la fragmentación es la optimización del uso del ancho de banda de la red al momento de trabajar con grandes flujos de datos.

Figura 2. **La federación HDFS de Hadoop**



Fuente: elaboración propia.

El elemento de la arquitectura denominado NameNode gestiona únicamente un conjunto de bloques a los cuales se les asigna un espacio de nombre de archivos, sin embargo, puede acceder a cualquiera de los DataNodes para almacenar los datos. Al espacio de nombres de archivos y a su respectivo conjunto de bloques se los denomina espacio de nombres del volumen. Para que los NameNodes puedan gestionar la metadata de cada uno de los DataNodes, estos le envían un *heartbeat* y un reporte de bloques cada

tres segundos. Un *heartbeat* permite conocer a los NameNodes que un DataNode se encuentra activo, mientras que el reporte de bloques mantiene actualizada la metadata de los bloques que contiene cada DataNode.

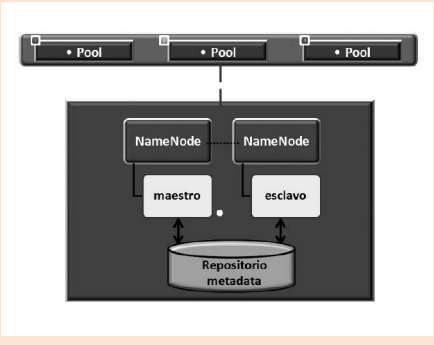
La federación HDFS es una característica de Hadoop que permite añadir NameNodes totalmente independientes al sistema para permitir la escalabilidad del mismo. La federación HDFS ofrece las siguientes ventajas:

- **Rendimiento.** Al tener varios NameNodes, las operaciones de escritura y lectura mejoran.
- **Aislamiento.** Mejoran el rendimiento de los clientes y otras aplicaciones ya que pueden realizar sus operaciones en diferentes NameNodes.

Si un NameNode es eliminado, también se eliminará el espacio de nombres del volumen, es decir, desaparecerán todos los bloques almacenados en los DataNodes que fueron asignados a este NameNode. Es aquí donde la HDFS HA juega un papel importante dentro de la nueva arquitectura.

La HDFS HA permite introducir un NameNode maestro y otro esclavo como se muestra en la figura 3. Durante las operaciones con el cliente solo el maestro puede estar activo. En caso de que el maestro sufra algún desperfecto, el NameNode esclavo entra en funcionamiento de manera inmediata ya que los NameNodes maestro y esclavo mantienen un repositorio común para su metadata y también debido a que el NameNode esclavo, cuando se encuentra en modo pasivo, es capaz de escuchar los *heartbeats* de los DataNodes.

Figura 3. **La HDFS HA de Hadoop**



Fuente: elaboración propia.

El HDFS de Hadoop es un sistema de directorios que permite almacenar y gestionar cualquier tipo de datos (...) de manera distribuida en un clúster escalable de servidores

2.3. Conceptos generales del Sistema de Información Global de la Organización Meteorológica Mundial

La WMO es un organismo especializado de las Naciones Unidas. Se encarga de estudiar el estado y el comportamiento de la atmósfera terrestre, los océanos, el clima, y de los recursos hídricos. Toda esta información proviene de diversas clases de instrumentos instalados en diferentes ubicaciones geográficas.

A todos estos instrumentos que componen el GOS, la WMO los ha clasificado en tres categorías, que son las siguientes:

- **Instrumentos de clase 1.** Estos instrumentos miden diversas magnitudes físicas (por ejemplo, temperatura, presión atmosférica) y están instalados en el área de interés.
- **Instrumentos de clase 2.** A diferencia de los instrumentos de clase 1, estos miden magnitudes físicas de manera remota dentro de un área o volumen.
- **Instrumentos de clase 3.** Son instrumentos que miden la velocidad del viento desde el seguimiento de las metas físicas y su desplazamiento observado durante el tiempo.

De aquí en adelante se pondrá especial énfasis en los instrumentos de clase 1 que implementan el proceso denominado «observación meteorológica automática» (OMA). La OMA se refiere a las actividades involucradas en la conversión de las mediciones de los elementos meteorológicos en señales eléctricas a través de sensores, el procesamiento y la transformación de estas señales en los datos meteorológicos, para luego transmitir la información resultante por cable, por radio o almacenar los mismos automáticamente en un medio de persistencia de datos. A los instrumentos que implementan el proceso OMA se los conoce como «estaciones meteorológicas automáticas» (AWS).

Las AWS se subdividen en dos tipos, que son las AWS en tiempo real y las AWS en tiempo diferido. Estos tipos de AWS se detallan a continuación:

- **AWS en tiempo real.** Estas estaciones tienen la capacidad de transmitir la data generada después de seguir el proceso OMA a través de un enlace de cable o por radio (red celular o satelital dependiendo de su ubicación geográfica) de manera automática o por censo de un usuario.
- **AWS en tiempo diferido.** Al contrario que las AWS en tiempo real, estas estaciones no tienen acceso a la red para la transmisión de data, por tal razón estas deberán almacenar la data generada después de seguir el proceso OMA en algún medio persistente de data.

Las ventajas de las AWS se mencionan a continuación:

- Permiten una observación continua de las diferentes magnitudes físicas sin necesidad de intervención humana alguna.
- Este tipo de estaciones pueden ser instaladas en ubicaciones geográficas inaccesibles.
- Eliminan los errores que pueden cometer los humanos al momento de tomar los valores de las diferentes magnitudes físicas.
- Se pueden eliminar o adicionar sensores cuando se desee sin afectar al funcionamiento de la estación.
- La aplicación de técnicas de estandarización permite la homogeneización en la configuración de estas estaciones reduciendo notablemente el tamaño del equipo técnico requerido.
- Se pueden configurar estas estaciones con niveles óptimos de precisión para la toma de valores de las diferentes magnitudes físicas.

Por la extensión y complejidad que abarca cada uno de estos tipos de instrumentos, este tutorial se enfocará únicamente en construir un prototipo para el manejo de los datos meteorológicos provenientes de las AWS en tiempo diferido que pertenecen a los instrumentos de clase 1 como se mencionó anteriormente. Sin embargo, este prototipo puede ser adaptado para las AWS en tiempo real mediante la automatización y sincronización del mismo con la llegada de los archivos.

Una AWS se encuentra constituida por un conjunto de componentes, los mismos que se detallan a continuación:

- **Sensores.** Un sensor es un instrumento especializado para detectar algún tipo de fenómeno físico o químico presente en el medioambiente. Cuando detecta un fenómeno, este instrumento transmite (por lo general) una señal eléctrica que deberá ser interpretada por otro instrumento, como, por ejemplo, un registrador de datos.
- **Registrador de datos (*datalogger*).** Es un instrumento electrónico que permite transformar las señales eléctricas provenientes de otros instrumentos en datos y registrarlos a intervalos fijos durante un periodo de tiempo. También permite la trasmisión de estos datos registrados hacia un servidor FTP o a través de puertos TCP. El formato de los archivos que transmiten (por lo general) son *.csv (*comma separated values*) y contienen datos semiestructurados.

Figura 4. **Muestra del contenido de un archivo generado por un registrador de datos**

```
1  2013-11-01 00:00:04,TEMPERATURA AIRE,25.9,°C
2  2013-11-01 00:01:04,TEMPERATURA AIRE,25.9,°C
3  2013-11-01 00:02:04,TEMPERATURA AIRE,25.9,°C
4  2013-11-01 00:03:04,TEMPERATURA AIRE,25.9,°C
5  2013-11-01 00:04:04,TEMPERATURA AIRE,25.9,°C
6  2013-11-01 00:05:04,TEMPERATURA AIRE,25.9,°C
7  2013-11-01 00:06:03,TEMPERATURA AIRE,25.9,°C
8  2013-11-01 00:07:03,TEMPERATURA AIRE,25.9,°C
9  2013-11-01 00:08:03,TEMPERATURA AIRE,25.9,°C
10 2013-11-01 00:09:03,TEMPERATURA AIRE,25.9,°C
11 2013-11-01 00:10:03,TEMPERATURA AIRE,25.9,°C
12 2013-11-01 00:11:03,TEMPERATURA AIRE,25.9,°C
13 2013-11-01 00:12:03,TEMPERATURA AIRE,25.9,°C
14 2013-11-01 00:13:03,TEMPERATURA AIRE,25.9,°C
15 2013-11-01 00:14:03,TEMPERATURA AIRE,25.8,°C
```

Fuente: elaboración propia.

En la figura 4 se puede observar una muestra del contenido de un archivo *.csv que fue transmitido por un registrador de datos. Su configuración le permitió registrar datos de la magnitud física temperatura del aire a un intervalo fijo de un minuto durante un periodo de tiempo de cuatro meses. A este conjunto de datos se le denominará *dataset*[12]. La cantidad de datos que contiene este *dataset* es aproximadamente de 170.873

[12] http://es.wikipedia.org/wiki/Conjunto_de_datos

filas; a cada una de estas filas le corresponden cuatro columnas que representan a las variables fecha de la toma del dato, nombre, valor y unidad de medida de la magnitud física, respectivamente.

El ejemplo muestra un *dataset* (conjunto de datos meteorológicos) ideal, es decir, que los datos que contiene están libres de errores (mal formación de caracteres, espacios en blanco entre comas, entre otros).

Suponiendo que este *dataset* no fuera ideal (y aun si lo fuera), ya puede ocasionar problemas a un sistema DBMS tradicional para el tratamiento de su contenido y almacenamiento del mismo (Nathan y Warren, 2014). Si se desea tratar el contenido de este *dataset* con un DBMS tradicional, se requiere de la creación (o adquisición) de una aplicación externa que implemente el proceso de extraer, transformar y cargar (ETL). Siendo esta una solución válida, este ETL podría requerir mayor cantidad de recursos de memoria si en algún momento se desea añadir variables al *dataset* y reducir el intervalo fijo del registrador de datos. A todo esto se le debería multiplicar por el número de registradores que tenga el GOS.

Los siguientes problemas tienen que ver con el control de calidad o filtrado de este volumen considerable de datos y el procesamiento necesario para la obtención de información a partir del mismo.

Para este tutorial se propone el procesamiento del *dataset* que se mostró en la figura 4 utilizando el ecosistema de Hadoop para la obtención de la siguiente información: temperatura del aire mínima, máxima y promedio de cada hora, día y mes.

3. DESARROLLO

La vía recomendable para comenzar a usar Hadoop es utilizando un ambiente local preconfigurado. En el mercado existen muchos de estos ambientes disponibles para descargarse, los cuales son citados en la tabla 1 junto con sus requerimientos:

Tabla 1. **Ambientes preconfigurados virtuales de Hadoop**

Nombre empresa	Nombre producto	Versión	Memoria RAM	Sistema operativo	Producto de virtualización
HortonWorks	HortonWorks Sandbox	2.1	4	Windows Mac	Oracle VirtualBox 4.2
Cloudera	Cloudera QuickStart VM	5	8	SO de 64 bits	VMware, VirtualBox y KVM
MapR	MapR SandBox	3.1.0	8	SO de 64 bits	VMware y VirtualBox

Fuente: elaboración propia.

Para este trabajo, se utilizará el producto HortonWorksSandbox, ya que es una versión *open-source* con soporte empresarial. Además ocupa un espacio de memoria RAM aceptable para un equipo de escritorio o portátil.

3.1. Desarrollo del prototipo

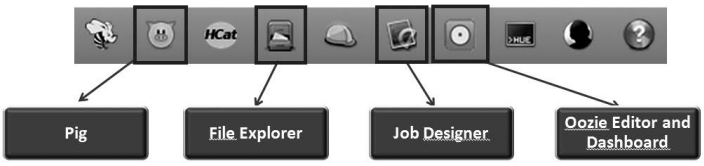
HortonWorks Sandbox incluye el proyecto Apache Hue, que es una interfaz web para trabajar con el ecosistema de Hadoop. Esta página se encuentra localizada en la URL http://localhost:8000/.

Para la construcción del prototipo se necesita realizar las siguientes tareas:

- **Creación del espacio de trabajo.**
- **Cargar del *dataset* al espacio de trabajo.**
- **Creación de flujos de datos con Pig.**
- **Creación de un flujo de trabajo con Oozie.**

Todas estas tareas involucran un *framework* de Hadoop presente en la barra de herramientas de Hue (véase figura 5).

Figura 5. ***Frameworks* de la barra de herramientas de Hue**



Fuente: elaboración propia.

3.1.1. Creación del espacio de trabajo con el File Explorer de Hue

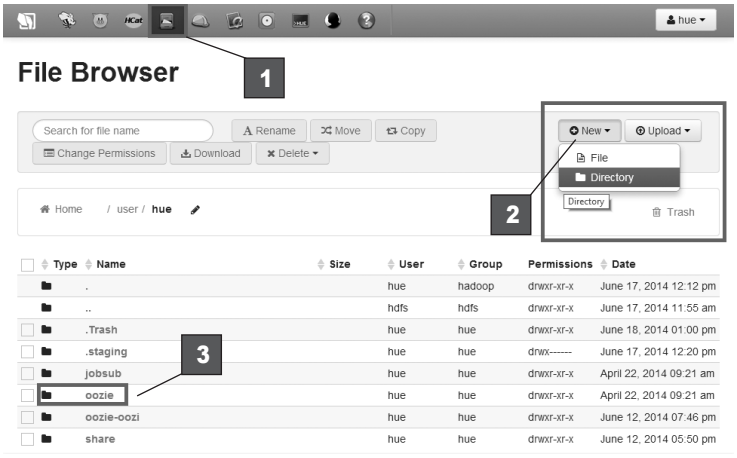
El File Explorer, como su nombre indica, es un explorador de directorios y archivos que permite interactuar con el HDFS de Hadoop.

El espacio de nombres asignado para el usuario hue es '/user/hue'.

El procedimiento para crear espacios de nombres se muestra en la figura 6. Siguiendo este procedimiento se deben crear los espacios de nombres asignados para el almacenamiento del *dataset*, el *script* y el reporte. Estos espacios de nombres son los siguientes:

- '/user/hue/oozie/workspaces/TFM/datasets/aws'.
- 'user/hue/oozie/workspaces/TFM/scripts'.
- '/user/hue/oozie/workspaces/TFM/reportes'.

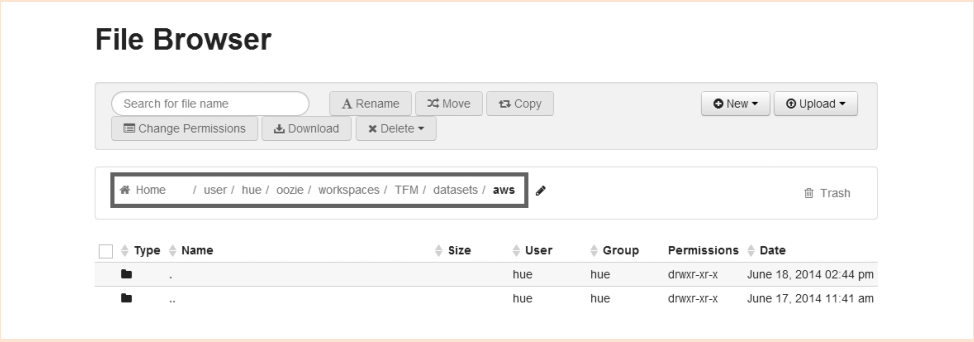
Figura 6. **Procedimiento para crear un directorio en el HDFS de Hadoop utilizando la interfaz web de Hue**



Fuente: elaboración propia.

El resultado de la creación del espacio de nombres para el *dataset* se muestra en la figura 7.

Figura 7. **Espacios de nombres asignados para el almacenamiento de los *datasets* necesarios para la construcción del prototipo**

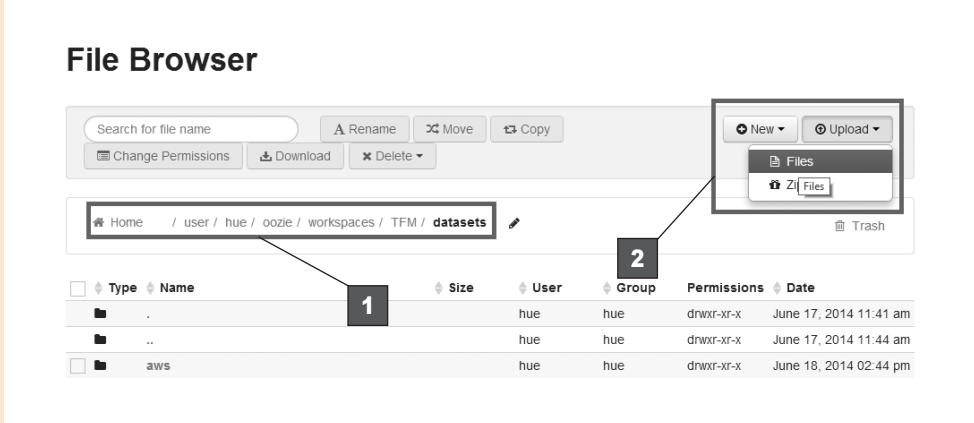


Fuente: elaboración propia.

3.1.2. Carga del dataset al espacio de trabajo utilizando Hue

La carga de archivos desde un sistema de directorios externo hacia el HDFS de Hadoop se realiza también utilizando el File Explorer de Hue, el mismo que se revisó en el apartado anterior. El procedimiento para cargar archivos a un espacio de nombres se muestra en la figura 8.

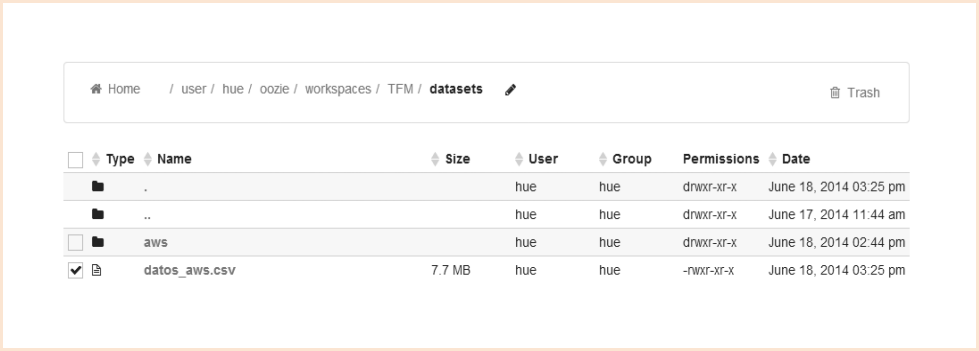
Figura 8. **Procedimiento a realizar para cargar un archivo en el HDFS de Hadoop utilizando la interfaz web de Hue**



Fuente: elaboración propia.

El archivo que se debe cargar es datos_aws.csv. Este contiene valores separados por comas de la magnitud física temperatura. En la figura 9 se muestra el resultado de haber cargado el *dataset*.

Figura 9. **El *dataset* datos_aws.csv, dentro del espacio de nombres asignado en el HDFS**



Fuente: elaboración propia.

3.1.3. Creación del flujo de datos con Pig

Pig es un *framework* que permite crear flujos de datos utilizando el HDFS y el sistema de procesamiento paralelo MapReduce de Hadoop. Este último realiza operaciones denominadas jobs sobre los datos almacenados en el HDFS. Un trabajo (*job*) está compuesto por tres fases, que son:

- **Mapear (*map*).**
- **Reordenar (*shuffle*).**
- **Reducir (*reduce*).**

La figura 10 muestra un flujo de datos, el mismo que tiene un único trabajo (Gates, 2011).

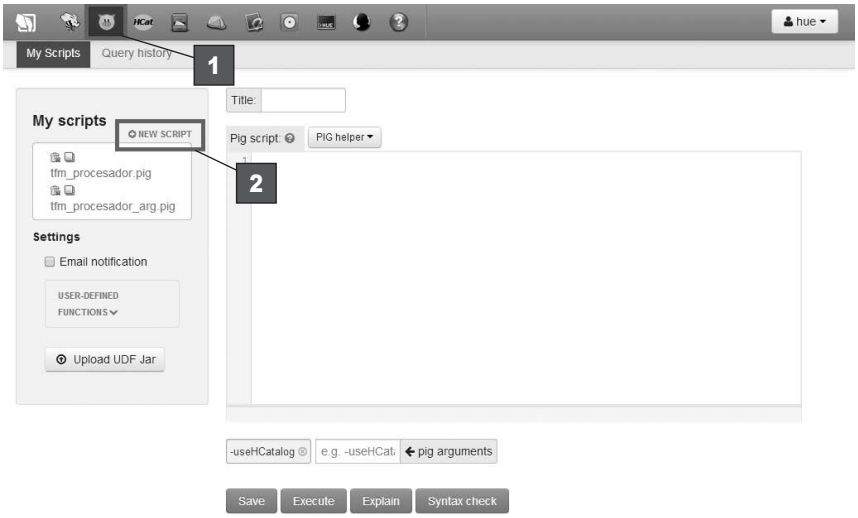
El flujo de datos que se va a realizar en este apartado se denominará tfm_procesador.Pig. El procedimiento para la creación de un flujo de datos se muestra en la figura 11.

El flujo, en la fase de Datos de Entrada, tiene como objetivo cargar el *dataset* que se encuentra en el espacio de nombres creado en el apartado 3.1.1. Para cargar el *dataset*, como se puede observar en la figura 12, se requiere indicar que el separador entre columnas de datos es la coma, usando la función interna PigStorage y, a continuación, el nombre que se le asignará a cada columna del *dataset* junto con su tipo de dato.

Figura 10. **Esquema de flujo de datos con Pig**

Fuente: elaboración propia.

Figura 11. **Procedimiento para la creación de un flujo de datos con Pig**



Fuente: elaboración propia.

Por ejemplo, en la figura 12 se puede apreciar que la columna fechaC se especificó con el tipo de dato chararray. Esto se debe a que Pig, al momento de realizar este trabajo, tenía un limitante al momento de carga de *datasets* que contuvieran fechas. Por tal razón se requería de una conversión de tipos de datos antes de enviarlos al trabajo. Para resolver este problema, se recorrió cada uno de los valores de la columna fechaC utilizando el operador FOREACH para convertirlos al tipo de dato *datetime* utilizando la función ToDate. El resultado de esta conversión se almacenó en una denominada «fecha» mediante el operador AS y se le asignó el tipo de dato *datetime*. Esta columna reemplaza a la columna fechaC. También se especificó que a continuación de la columna fecha se añadieran todas las columnas que siguen a la columna magnitud_fisica incluyendo esta última haciendo uso del operador puntos suspensivos (...).

Figura 12. **Fase de entrada de datos. Carga de datos desde el *dataset* almacenado en el HDFS**

```
datos_aws =
LOAD '/user/hue/oozie/workspaces/TFM/datasets/aws/datos_aws.csv'
USING PigStorage(',')
AS (
fechaC:chararray,
magnitud_fisica:chararray,
valor:float,
unidad_medida:chararray
);
```

Fuente: elaboración propia.

La estrategia de ordenamiento está asociada a la división de los datos en secciones. Recordemos que la información que se desea obtener a partir del *dataset* es la siguiente: temperatura del aire mínima, máxima y promedio de cada hora, día y mes. Si somos observadores, podremos darnos cuenta de que el año, el mes, el día y la hora son variables que se pueden expresar como tipos de datos enteros (*int*) y que podrían añadirse como columnas (por cada fila de datos) a continuación de las columnas que siguen a la columna magnitud_fisica. A continuación se detalla únicamente el procedimiento que se siguió para añadir la columna anio. Para obtener el año a partir de una fecha se necesita de la función interna de Pig, GetYear, la cual requiere como parámetro una variable con el tipo de dato *datetime*. Este parámetro se obtiene de aplicar el mismo procedimiento seguido en el párrafo anterior para la conversión del tipo de dato a la columna fechaC. Finalmente, mediante el operador AS, se procede a añadir como columna el resultado de la función GetYear. Para obtener el mes, el día y la hora se deben utilizar las funciones GetMonth, GetDay y GetHour, respectivamente.

Toda esta compleja operación se muestra en la figura 13. Las pocas líneas de código utilizadas demuestran que Pig es un *framework* especializado para el tratamiento de datos semiestructurados (Lam, 2011).

Figura 13. **Fase de entrada de datos. Preparación de los datos**

```
datos_aws_columna_fecha =
FOREACHdatos_aws
GENERATE
        ToDate(fechaC, 'yyyy-MM-ddHH:mm:ss') AS (fecha:datetime),
        magnitud_fisica..,
GetYear((datetime)ToDate(fechaC, 'yyyy-MM-ddHH:mm:ss'))
AS (anio:int),
GetMonth((datetime)ToDate(fechaC, 'yyyy-MM-ddHH:mm:ss'))
AS (mes:int),
GetDay((datetime)ToDate(fechaC, 'yyyy-MM-ddHH:mm:ss')) AS
(dia:int),
GetHour((datetime)ToDate(fechaC, 'yyyy-MM-ddHH:mm:ss'))
AS (hora:int);
```

Fuente: elaboración propia.

Una muestra del resultado de la subfase mapear se muestra en la tabla 2. Se puede apreciar que la columna fechaC del tipo chararray ha sido reemplazada por la columna fecha del tipo *datetime* y que han sido añadidas las columnas año, mes, día y hora.

Tabla 2. **Ejemplo de una fila en la fase de entrada de datos y de una fila procesada en la subfase mapear**

Fila original	2013-11-01 00:00:04, TEMPERATURA AIRE,25.9,°C
Fila procesada	2013-11-01T00:00:04.000-07:00, TEMPERATURA AIRE,25.9,°C,2013,11,1,0

Fuente: elaboración propia.

La subfase reordenar permite agrupar el *dataset* dependiendo de las necesidades de los usuarios. Para el ejemplo se realizará la agrupación necesaria para poder efectuar el cálculo de la temperatura del aire mínima, máxima y promedio para cada día del año. Esto se logra utilizando la función interna de Pig, GroupBy. Como parámetro de entrada, esta función necesita una o más columnas para agrupar los datos. Hay que tomar en cuenta que en el *dataset* pueden estar presentes datos de distintos años y que esto podría dar lugar a una ambigüedad al momento de agruparlos. Para evitar esto, dentro de la función GroupBy se deberían ingresar las columnas año, mes y día como se muestra en la figura 14.

Figura 14. **Ejemplo de agrupación de datos por día utilizando Pig**

```
datos_aws_agrupados_anio_mes_dia =
GROUPdatos_aws_columna_fecha
BY(anio,mes,dia);
```

Fuente: elaboración propia.

Finalmente, en la subfase reducir de la fase trabajo se efectúan los cálculos necesarios sobre las agrupaciones realizadas en la subfase reordenamiento. Siguiendo el ejemplo anterior, para obtener la temperatura del aire mínima, máxima y promedio para cada día del año se debe recorrer cada una de las filas del *dataset*, resultado de la subfase reordenar, y aplicar las funciones internas de Pig, min, max y avg, a la columna valor de las filas del *dataset* que resultó de la subfase mapear, como se muestra en la figura 15.

Figura 15. **Ejemplo de cálculo de estadísticas por día en Pig**

```
datos_aws_estadisticos_diario =
FOREACHdatos_aws_agrupados_anio_mes_dia
GENERATE
MIN(datos_aws_columna_fecha.fecha),
MAX(datos_aws_columna_fecha.fecha),
MIN(datos_aws_columna_fecha.valor),
MAX(datos_aws_columna_fecha.valor),
AVG(datos_aws_columna_fecha.valor);
```

Fuente: elaboración propia.

También fue necesario añadir dos columnas más al inicio de este *dataset* resultado de la subfase reducir. Estas columnas tienen como objetivo informar al usuario del periodo de tiempo sobre el cual se realizaron los cálculos estadísticos. Una muestra del resultado se presenta en la tabla 3.

Tabla 3. **Muestra del resultado de la subfase reducir**

Hora de inicio	Hora de fin	Min	Max	Avg
2013-11-01T00:00:04.000-07:00	2013-11-01T23:54:03.000-07:00	23,80	33,80	26,97
2014-02-28T00:00:03.000-08:00	2014-02-28T23:54:03.000-08:00	26,10	33,70	28,79

Fuente: elaboración propia.

La fase de salida de datos tiene como objetivo almacenar el *dataset*, resultado de la fase trabajo, al espacio de nombres creado en el apartado 3.1.1. para los elementos de salida del prototipo. Este procedimiento se logra mediante el uso de la función interna de Pig, Store Into, que recibe como parámetro un espacio de nombres. Para este caso es '/user/hue/oozie/workspaces/TFM/reportes, como se muestra en la figura 16.

Figura 16. **Almacenamiento de un *dataset* en el HDFS de Hadoop utilizando Pig**

```
STOREdatos_aws_estadisticos_diario
INTO '/user/hue/oozie/workspaces/TFM/reportes/diario/';
```

Fuente: elaboración propia.

Para poder crear el flujo de trabajo que se detallará en el próximo apartado es necesario almacenar este flujo de datos en un archivo al cual llamaremos tfm_accion_reportes.Pig dentro de un directorio local de su computador. Antes de almacenarlo se debe realizar una leve modificación en el segmento de código que se mostró en la figura 12. En lugar de especificar un directorio de entrada directamente, se especificará como un parámetro. A este parámetro lo hemos denominado $directorio_dataset. El resultado de esta modificación se muestra en la figura 17.

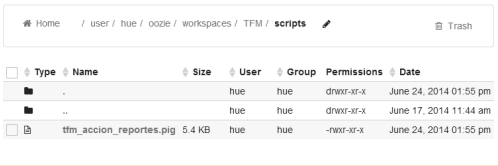
Figura 17. **El parámetro $directorio_dataset que simboliza el directorio de entrada en Pig**

```
datos_aws =
LOAD '$directorio_dataset'
USINGPigStorage(',')
AS (
      fechaC:chararray,
magnitud_fisica:chararray,
valor:float,
unidad_medida:chararray
);
```

Fuente: elaboración propia.

Una vez almacenado el flujo de datos, se debe seguir el procedimiento que se presentó en el apartado 3.1.2 para cargar este archivo en el espacio de nombres asignado para el *script* en el apartado 3.1.1. El resultado de este procedimiento se muestra en la figura 18.

Figura 18. **El *script* tfm_accion_reportes.Pig dentro del espacio de nombres asignado en el HDFS**



Fuente: elaboración propia.

3.1.4. *Creación del flujo de trabajo con Oozie*

En muchas ocasiones resulta importante poder ejecutar varios flujos de datos de manera secuencial automáticamente. Esto se puede lograr a través del proyecto de Apache denominado Oozie. Dentro del contexto de Oozie, a la ejecución secuencial de flujos de datos se los denomina como flujo de trabajo o *workflow*.

Oozie es *framework* que se ejecuta dentro de un servidor web y viene integrado dentro de la interfaz web de Hue.

Para el diseño de un flujo de trabajo, Oozie utiliza un método gráfico denominado DAG (*directed acyclic graph*)[13]. El DAG del flujo del trabajo que se va a implementar se muestra en la figura 19.

Antes de continuar es importante mencionar los conceptos que abarca un flujo de trabajo de Oozie. A continuación se detallan estos conceptos:

- **Acción** (*action*). Es una etapa o paso dentro del flujo de trabajo. En la figura 19 se puede observar un ejemplo de una acción denominada Ejecutar flujo de datos.
- **Transición** (*transition*). Corresponde a la siguiente acción a ejecutar después de la ejecución de otra. Según la figura 19 no existe ninguna transición.
- **Trabajo** (*job*). Es un proceso que corre dentro del servidor web que ejecuta la definición del flujo de trabajo de Oozie.

[13] DAG es un gráfico que no contiene ciclos en el mismo. Es una colección de nodos conectados directamente.

Las acciones en Oozie pueden ser de dos tipos:

- **Sincrónica.** Es aquella que se ejecuta dentro del proceso de un servidor web.
- **Asincrónica.** Esta acción se ejecuta en el clúster de Hadoop.

Las transiciones en Oozie son gobernadas por condiciones. Similar a las acciones, estas transiciones pueden ser de dos tipos:

- **Condiciones estructurales.** Son condiciones estáticas. El DAG de la figura 19 es un ejemplo de este tipo de condiciones.
- **Condiciones en tiempo de ejecución.** Nacen a partir del resultado de algún tipo de acción. Estos resultados pueden ser variables o banderas denominadas *paths*.

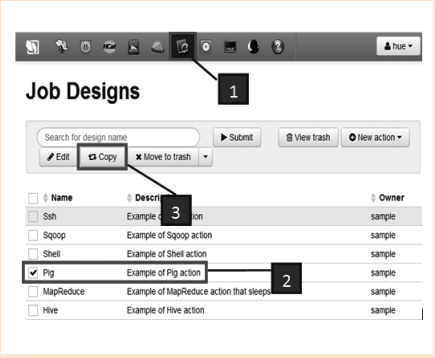
El DAG de la figura 19 contiene una acción asíncrona, la misma que fue creada en el apartado 3.1.3 con *Pig*, y contiene una transición con una condición estructural.

Figura 19. **DAG del flujo de trabajo a implementar**

Fuente: elaboración propia.

La interfaz web de Hue permite la creación de un flujo de trabajo con Oozie de manera gráfica. Para esto debemos hacer un clic sobre el icono correspondiente al Job Designer. En esta pantalla de Hue se muestran algunos diseños que pueden servir como plantillas. Para este caso vamos a modificar la plantilla denominada Pig. Hacemos clic sobre la plantilla y realizamos clic sobre la opción Copy, como se muestra en la figura 20.

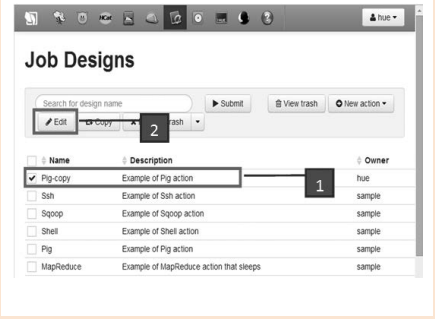
Figura 20. **Procedimiento para crear un flujo de trabajo de Oozie a partir de una plantilla**



Fuente: elaboración propia.

Una vez realizado el clic sobre el botón Copy aparecerá una copia de la plantilla denominada Pig-copy. Para editarla según las necesidades del problema, la seleccionamos y pulsamos el botón Edit como se muestra en la figura 21.

Figura 21. **Procedimiento para editar un flujo de trabajo de Oozie**



Fuente: elaboración propia.

Por facilidad de presentación, las opciones para editar una plantilla se mostrarán en dos partes denominadas figura 22-A y figura 22-B.

La figura 22-A contiene las opciones Name, Description, Script name y Prepare. El nombre asignado a este flujo de trabajo será tfm_reportes_ft. La descripción podría ser cualquier criterio que describa mejor a este flujo de trabajo. El *script* que se debe seleccionar es tfm_accion_reportes.Pig, el mismo que se creó y cargó en el apartado 3.1.3. La opción Prepare permite al usuario configurar ciertas condiciones iniciales necesarias para la ejecución del flujo de trabajo, por ejemplo, el borrar el espacio de nombres asignado para los elementos de salida del prototipo que se creó en el apartado 3.1.1 con el fin de evitar conflictos de integridad en los reportes. Este procedimiento también se muestra en la figura 22-A.

Figura 22-A. **Opciones Name, Description, Script name y Prepare de un flujo de trabajo de Oozie**

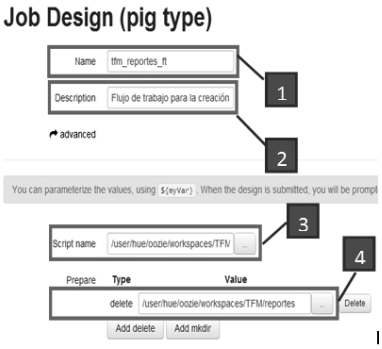
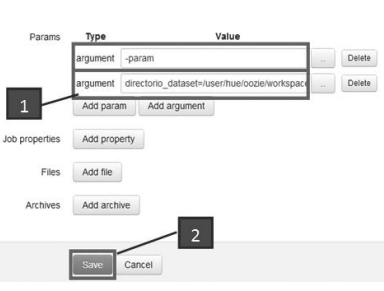


Figura 22-B. **Ejemplo de cómo pasar argumentos de Oozie como parámetros a una acción de Pig**



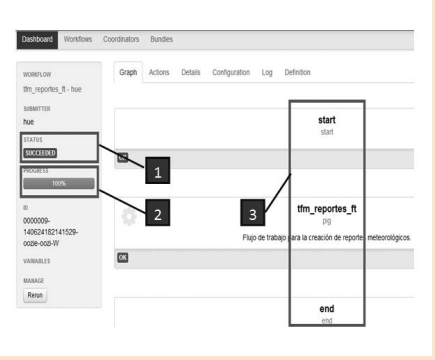
Fuente: elaboración propia.

La figura 22-B muestra la opción Argument de Oozie para pasar parámetros de entrada a una acción (o acciones) de un flujo de trabajo. Esta opción es necesaria para poder ejecutar el flujo de datos almacenado en el *script* denominado tfm_accion_reportes.Pig que se creó al final del apartado 3.1.3. En Pig, el paso de parámetros se debe especificar a través de la opción -param y, a continuación, el nombre del parámetro (en este caso, directorio_dataset) seguido de un igual y el valor asignado al parámetro (este flujo de datos requiere el espacio de nombres donde se encuentra almacenado el *dataset* /user/hue/oozie/workspaces/TFM/datasets/aws/datos_aws.csv). A continuación se deben eliminar los argumentos que venían incluidos en la plantilla y almacenar el flujo de trabajo haciendo clic sobre el botón Save.

Después de almacenar el flujo de trabajo, Hue retornará a la pantalla principal del Job Designer. Para ejecutar este flujo de trabajo se debe seleccionar el mismo y hacer clic sobre el botón Submit.

Inmediatamente Hue desplegará la pantalla de Oozie Editor/Dashboard. En esta pantalla se puede monitorear el estado y el progreso del flujo de trabajo. También se puede observar el DAG correspondiente al flujo de trabajo creado. En este caso, este DAG contiene los nodos *start* y *end*, y además ejecuta una acción del tipo Pig. Esta pantalla se muestra en la figura 23.

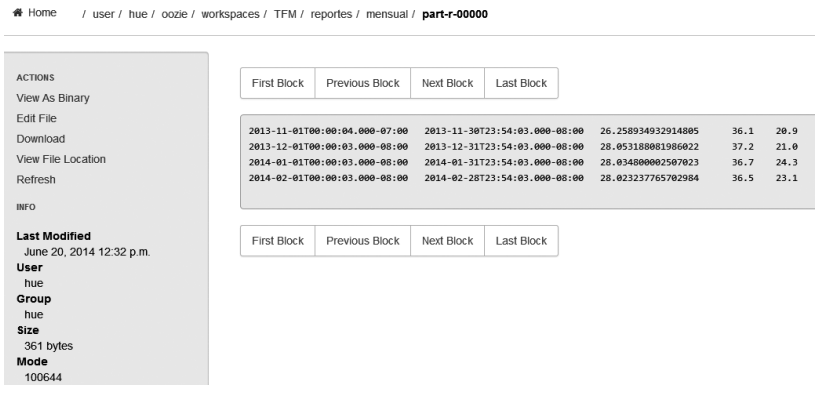
Figura 23. **La pantalla del Editor/Dashboard de Oozie mostrando el estado y el progreso de un flujo de trabajo**



Fuente: elaboración propia.

Una vez ejecutado el flujo de trabajo, se puede explorar el espacio de nombres asignado para los reportes y se observa que Pig generó un archivo denominado part-r-00000 (nomenclatura propia de los bloques) que contiene la información requerida. En la figura 24 se puede observar el resultado del reporte meteorológico mensual.

Figura 24. **Reporte meteorológico mensual generado a partir de la ejecución del flujo de trabajo**



Fuente: elaboración propia.

4. CONCLUSIONES Y LÍNEAS FUTURAS

Hadoop, un framework que implementa el paradigma Big Data, es una buena alternativa para solucionar muchos de los problemas relacionados con el almacenamiento de la gran cantidad y variedad de datos provenientes de los tipos de instrumentos 1, 2, 3 que se describen dentro del GOS de la WMO, como también de las imágenes provenientes de satélites y radares. Esto gracias a su sistema de directorios distribuido, el HDFS, que se instala sobre un clúster de servidores y que proporciona ventajas, como replicación de datos, rendimiento, escalabilidad y confiabilidad.

Para el tratamiento de los datos semiestructurados generados por las AWS, se utilizó el *framework* Pig. A diferencia de los *frameworks* tradicionales, Pig tiene la capacidad de procesamiento en paralelo cercano a los datos utilizando las capacidades computacionales del clúster de Hadoop. La facilidad con la que este *framework* procesa los datos convierte a Pig en una herramienta muy atractiva para los usuarios dedicados a este tipo de tareas. Sin embargo, al momento de escribir esta artículo se observó que Pig tiene inconvenientes al cargar datos que tienen fechas y horas. Para resolver el problema se hizo uso de las mismas funciones internas de este *framework*, pero esto podría significar latencia al momento de construir aplicaciones en tiempo real.

A pesar de que el HDFS de Hadoop es suficiente para el almacenamiento de datos no estructurados y semiestructurados, existen proyectos que permiten proyectar una estructura semejante a las DBMS tradicionales sobre el HDFS de Hadoop. Este es el caso del proyecto de Apache denominado Hive. Con Hive, se puede construir un *data warehouse* para facilitar la consulta y gestión de grandes conjuntos de datos que residen en el HDFS de Hadoop. Las consultas a los datos se pueden realizar mediante un lenguaje denominado HiveQL, el mismo que tiene una sintaxis muy semejante al estándar SQL-92 (Capriolo et ál., 2012).

El nacimiento de las ciudades inteligentes y la evolución de la sociedad del futuro es un campo atractivo para Big Data y se está evidenciando en los últimos años. El concentrar datos de diferentes fuentes para luego procesarlos e interrelacionarlos con diferentes propósitos podría generar nuevas áreas de estudio. Por ejemplo, se pueden combinar datos de educación, de la Administración pública, de la medicina, etc. Como ejemplo concreto, los autores están realizando trabajos de investigación donde se combinan datos de medicina, meteorología y de aprendizaje máquinas para determinar en qué temporadas del año se podrían presentar las condiciones climáticas necesarias para el aparecimiento de cierto virus. Con esto se lograría que los ciudadanos de estas grandes ciudades sean alertados de manera temprana para que ingieran los alimentos adecuados, utilicen la vestimenta sugerida, tengan en su botiquín los medicamentos necesarios, entre otros.

El nacimiento de las ciudades inteligentes y la evolución de la sociedad del futuro es un campo atractivo para Big Data y se está evidenciando en los últimos años. El concentrar datos de diferentes fuentes para luego procesarlos e interrelacionarlos con diferentes propósitos podría generar nuevas áreas de estudio

5. REFERENCIAS BIBLIOGRÁFICAS

Capriolo, E.; Wampler, D. y Rutherglen, J. [2012]: *Programming hive*, Sebastopol, California, EE. UU., O'Reilly Media.

Gates, A. [2011]: Programming Pig, 1.ª ed., Sebastopol, California, EE. UU., O'Reilly Media.

Holmes, A. [2014]: *Hadoop in practice*, 2.ª ed., Shelter Island, Nueva York, EE. UU., Manning Publications Co.

Lam, C. [2011]: *Hadoop in action*, Stamford, Connecticut, EE. UU., Manning Publication Co.

Nathan, M. y Warren, J. [2014]: *Big Data: principles and best practices of scalable realtime data systems*, Manning Publications.

Prajapati, V. [2013]: *Big Data analytics with R and Hadoop*, Birmingham, Reino Unido, Packt Publishing Ltd.

Cursos Monográficos

Para obtener el ideal del pleno empleo de nuestros alumnos, en el CEF.– hemos ido facilitando una serie de medios en los que se apoyan las distintas formaciones durante el periodo de seguimiento de los másteres, cursos y preparación de oposiciones y, al tiempo, procuramos que muchos de esos medios acompañen a estas personas en su posterior desarrollo profesional. A título orientativo mencionamos a continuación la oferta formativa de algunos cursos.

Diseño de Programas Formativos e-Learning & b-Learning

(3 meses) [3 créditos] (inicio: octubre, febrero y mayo)

Clases a distancia: 290 € o 300 € en 3 plazos de 100 €.

Formador de Formadores on Line

(3 meses) [3 créditos] (inicio: octubre, febrero y mayo)

Clases a distancia: 290 € o 300 € en 3 plazos de 100 €.

Gestión de la Formación

(3 meses) [4 créditos] (inicio: octubre, febrero y mayo)

Clases a distancia: 290 € o 300 € en 3 plazos de 100 €.

Contabilidad Avanzada

(102 h presenciales o 9 meses a distancia) [12 créditos]

(inicio: octubre, febrero y mayo)

Clases presenciales: 1.250 € o 1.290 € en 6 plazos de 215 €.
Clases a distancia: 960 € o 990 € en 6 plazos de 165 €.

Contabilidad Práctica

(90 h presenciales o 5 meses a distancia) [6 créditos]

(inicio: octubre, febrero y mayo)

Clases presenciales: 842 € u 870 € en 6 plazos de 145 €.
Clases a distancia: 520 € o 540 € en 5 plazos de 108 €.

Tributación Práctica

(111 h presenciales o 9 meses a distancia) [11 créditos]

(inicio: octubre, febrero y mayo)

Clases presenciales: 1.160 € o 1.197 € en 7 plazos de 171 €.
Clases a distancia: 825 € u 854 € en 7 plazos de 122 €.

Seguridad Social y Derecho Laboral

(125 h presenciales o 6 meses a distancia) [15 créditos]

(inicio: octubre, febrero y mayo)

Clases presenciales: 1.636 € o 1.696 € en 8 plazos de 212 €.
Clases a distancia: 1.145 € o 1.184 € en 8 plazos de 148 €.

Gestión de Nóminas y Seguros Sociales (Práctica de salarios y cotizaciones)

(48 h presenciales o 4 meses a distancia) [4 créditos]

(inicio: octubre, febrero y mayo)

Clases presenciales: 485 € o 500 € en 5 plazos de 100 €.
Clases a distancia: 368 € o 380 € en 5 plazos de 76 €.

Gestión de Redes Sociales en la Empresa (Community manager)

(39 h presenciales o 4 meses a distancia) [4 créditos]

(inicio: octubre y febrero)

Clases presenciales: 550 € o 570 € en 4 plazos de 142,50 €.
Clases a distancia: 432 € o 441 € en 3 plazos de 147 €.

Técnico en Marketing

(100 h presenciales o 5 meses a distancia) [10 créditos]

(inicio: octubre y febrero)

Clases presenciales: 1.615 € o 1.668 € en 6 plazos de 278 €.
Clases a distancia: 765 € o 792 € en 6 plazos de 132 €.

Más información en:
www.cef.es