Normas de estilo para presentación de documentos y reportes técnicos en informática

Andrea Parra Salazar, Felipe González Casabianca
MINE4102 - Análisis Infomático sobre Big Data
Universidad de los Andes, Bogotá, Colombia
{f.gonzalez1899, aparras}@uniandes.edu.co
Fecha de presentación: Septiembre 24 de 2018

1 Resumen Ejecutivo

El siguiente documento resume el producto desarrollado para el Taller 1. Link de la aplicación:

http://172.24.101.19:8000/app1/

Este se desarrolló principalmente en Python 3, usando como framework de soluciones web a **Django** y **Bootstrap** y **D3** para la interacción con el usuario. Las consultas sobre los datos se hicieron utilizando estrategias MapReduce directamente sobre **Hadoop** y la comunicación con el cluster se hizo a traves de secure copy **(scp)**. Los detalles de la arquitectura están en la Figura 1.

1.1 Completitud Componentes

- **Componentes Implementados Totalmente**: Actualmente, el producto cuenta con la implementación completa de:
  - Requerimientos Funcionales: 2 y 3
  - Requerimientos Analíticos: 2 y 3.
- **Componentes Implementados Parcialmente**: 1 y 4
- **Componentes No Implementados**: Ninguno

1.2 Completitud Pruebas

Ver seccion 4.3

2 Datos

Los datos utilizados para el taller corresponden a viajes de taxis en la ciudad de New York City, USA, ofrecidos al público en el link: http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml. Este conjunto de datos está dividido por meses y años en archivos Comma Separated Values (CSV).

2.1 Estrategia de Utilización

Los datos se entregaron en su formato original dentro del Hadoop File System (HDFS) y se accedió a través de este mismo medio a ellos.

2.2 Dificultades

Aunque en la página web se encuentran los diccionarios de datos, no todos los archivos son fieles a esta información. Por lo tanto, se diseñó un trabajo MapReduce que extrajera los distintos encabezados que se encuentran en los datos recibidos el cual arrojó un total de 11 encabezados distintos. Por lo tanto, se infirió los valores de cada posición según la cantidad de valores que tenía

la línea. Esto nos permitió entregarles a los trabajos que se ejecutan en el *clúster* paquetes de líneas, sin tener que preocuparse por el encabezado.

Los retos RF1 y RF4 presentaron problemas de ejecución sobre el data set completo. Al correr en el clúster, se presentaba un error de memoria virtual hacia alrededor del 69% de la ejecución del reducer. Esto tiene que ver con la manera en la que se implementó en mapper ya que la manera en la que le estaba dando información al reducer, estaba poniendo toda la carga computacional en este. El reducer hace una comparación de todos los viajes por destino para hallar el lugar con mas viajes. Lo que se debió haber echo fue encontrar numero máximo de viajes por mapper (encontrando así varios máximos locales), y luego comparar estos valores con el reducer para hallar el máximo global.

3 Arquitectura

A continuación, se encuentra un diagrama de la arquitectura de la solución:

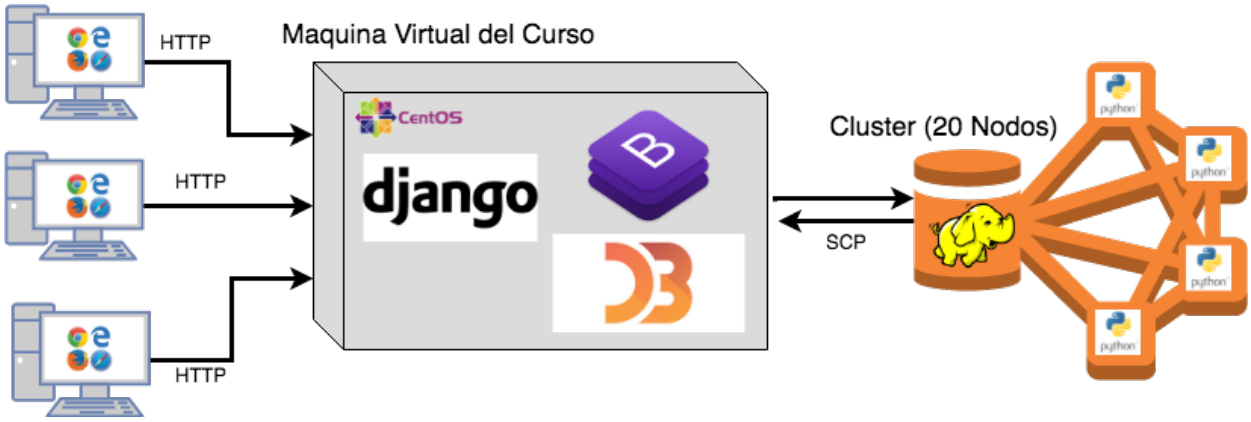

FIGURA 1 MODELO DE ARQUITECTURA

La interacción con el usuario se hace a través de una aplicación web desplegada sobre Django, utilizando Bootstrap y D3 para la visualización. Esta aplicación recolecta los resultados de las ejecuciones que se hacen sobre el *cluster* vía SCP. Debido al tamaño de los datos y la implementación de los trabajos, los tiempos de ejecución son muy altos (alrededor de: 1 hora, ver la sección de escalabilidad) lo que inhabilita que se ejecute un trabajo específico para cada consulta por usuario. La solución entregada sigue entonces el siguiente patrón:

- Cuando un usuario realiza una consulta, la aplicación extrae del *cluster* el resultado de la última ejecución del trabajo asociado, la interpreta y la muestra.
- En horarios de baja demanda se ejecutan los trabajos de nuevo para actualizar el estado de las consultas según los datos guardados. Actualmente, este proceso se hace a las 4:00am que es la hora de disponibilidad exclusiva del *cluster* para el grupo.

Estas decisiones hacen que el producto sea **BASE** (***B****asically* ***A****vailable,* ***S****oft state,* ***E****ventual consistency*) un estándar aceptable en los contextos de Big Data.

4 Detalles Implementación

La totalidad de los requerimientos se realizaron utilizando el patrón MapReduce sobre Python. El esquema de partición de los datos fue el mismo para todos los trabajos: la partición se hizo a nivel de línea, donde cada instancia de las clases *Map* recibe un conjunto de líneas diferenciadas por llave.

Este es el comportamiento estándar de la clase: `org.apache.hadoop.mapreduce.lib.input.TextInputFormat.`

A continuación esta el detalle de implementación para cada uno de los trabajos:

4.1 Requerimientos Funcionales:

	RF1
	Map
Formato Entrada	Línea separada por comas
Formato Salida	Hora<tab>Destino
Descripción	Devuelve la hora de cada viaje, y traduce el destino a un Borough específico de Ney York.
	Reducer
Formato Entrada	Hora<tab>Destino
Formato Salida	hora<tab>destino<tab>coteo
Descripción	Agrupa por hora y realiza un conteo por hora y destino.

	RF2
	Map
Formato Entrada	Línea separada por comas
Formato Salida	mes <tab> costo <tab> 1
Descripción	Revisa si el registro corresponde a un Domingo y de serlo, extrae su mes y costo final asociado al viaje
	Reducer
Formato Entrada	mes <tab> costo <tab> número de viajes
Formato Salida	mes <tab> costo <tab> número de viajes
Descripción	Agrupa por mes (sumando costo y número de viajes)

	RF3
	Map
Formato Entrada	Línea separada por comas
Formato Salida	dia semana <tab> hora del dia <tab> aeropuerto < : > origen < : > número viajes
Descripción	Revisa si el registro corresponde a un viaje desde alguno de los tres aeropuertos. En caso de serlo, extrae la hora y el día de la semana.
	Reducer
Formato Entrada	dia semana <tab> hora del dia <tab> aeropuerto < : > origen < : > #viajes
Formato Salida	d sem <tab> hora del dia <tab> aeropuerto < : > origen < : > #viajes …. origen < : > #viajes

Descripción	Agrupa por dia de la semana, hora del dia y aeropuerto, concatenando o sumando los valores de origen.

	RF4
	Map
Formato Entrada	Línea separada por comas
Formato Salida	Dia<tab>Hora<tab>Mes<tab>Lugar de recogida
Descripción	Devuelve el día, la hora y el mes de cada viaje, y traduce el destino a un Borough específico de Ney York.
	Reducer
Formato Entrada	Dia<tab>Hora<tab>Mes<tab>Lugar de recogida
Formato Salida	Dia de la semana<tab>Hora<tab>Mes<tab>Lugar de recogida<tab>conteo
Descripción	Agrupa por dia de la semana hora del dia, mes y lugar de recogida y el numero de viajes con dichas especificaciones

4.2 Requerimientos Analíticos:

	RA2
	Map
Formato Entrada	Línea separada por comas
Formato Salida	Momento <tab> origen < : > destino < : > #viajes < : > 1
Descripción	Extrae de cada registro que recibe el momento (incluye dia semana y dia mes), el origen y destino.
	Reducer
Formato Entrada	Momento <tab> origen < : > destino < : > #viajes < : > … origen < : > destino < : > #viajes
Formato Salida	Momento <tab> origen < : > destino < : > #viajes < : > … origen < : > destino < : > #viajes
Descripción	Agrupa por momento, concatenando o sumando los valores de numero de viajes de origen a destino

	RA3
	Map
Formato Entrada	Línea separada por comas
Formato Salida	Nombre de Festivo<tab>Hora de recogida<,>costo del viaje<,>1
Descripción	Revisa si el viaje ocurrió durante alguna festividad y devuelve la hora de recogida y el costo del viaje
	Reducer
Formato Entrada	Nombre de Festivo<tab>Hora de recogida<,>costo del viaje<,>1

Formato Salida	Nombre de Festivo<tab>Costo total de viajes<tab>Numero de viajes<tab>Costo promedio
Descripción	Agrupa por festividad y calcula la cantidad de viajes, y el costo promedio para cada festividad

4.3 Experimentos de escalabilidad:

	Data set	Completo	Data set	Parcial
	4-nodos	**20-nodos**	**1-nodo**	**4-nodos**
RF1				1hr37hr32
RF2		1hr13min14	52min16	36min40
RF3	1hr24min28	52min57	1h20min53	40min27
RF4				1hr29hr08
RA2	2hr43min22	1hr40min36	1hr27min07	56min42
RA3	2hr30min27	1hr34min27		1hr47min31

Como se expresó en la sección 2.2 los retos funcionales RF1 y RF4 presentaron problemas de implementación con el data set completo. Debido a un error de lógica en cuanto a la estructuración del MapReduce, no se pudieron ejecutar en 1 nodo con un data set parcial tampoco.

En cuanto a los demás requerimientos, el tiempo promedio de ejecución cambia substancialmente dependiendo del número de nodos disponibles. El tiempo de ejecución entre el clúster de 4 nodos y el de 20 nodos varia aproximadamente en un, 60%. Interesantemente, la escalabilidad de estos trabajos no es lineal. Esto tiene sentido si se tiene en cuenta que la eficiencia de un trabajo depende de varios aspectos. Por un lado depende de cómo se hace la partición de los datos, por otro de cómo se ejecutan los mappers de manera independiente, también de como se hace la fase de shuffle entre el proceso map y el proceso reduce, y finalmente de cómo se ejecuta el paso de reduce.

En cuanto a la ejecución de los requerimientos sobre Hadoop montado en un solo nodo, el cambio más acentuado se dio entre su desempeño y la del clúster de 4 nodos. En este caso la diferencia de ejecución llega hasta un 80%.

5 Evaluación del Producto

El producto cumple con todos los requerimientos de arquitectura y análisis. Algunos requerimientos están claramente mejor implementados que otros, en especial en cuanto a la visualización de los resultados. Esto se debe principalmente a la inexperiencia de algunos de los integrantes del grupo, que se enfrentaron con grandes dificultades en el momento de llevar los datos obtenidos durante la ejecución de los trabajos de MapReduce a un formato web.

Dejando de lado esto, la implementación web cumple con una arquitectura **BASE**. Correctamente programa la ejecución de trabajos MapReduce en el clúster usando *crontab,* y por conexión *scp* extrae estos resultados para hacerlos accesibles al usuario en tiempo real. La estructura del producto es clara y fácil de usar y adecuadamente maneja errores de input del usuario y anomalías en los datos entrando desde los trabajos MapReduce.

6 Discusión

Si volvieramos a realizar el taller solicitado con las lecciones aprendidas, cambiariamos lo siguiente:

- Explorar los procesos MapReduce con Java. Aunque desconocemos las implmentaciones de los otros grupos, por monitores del cluster, sentimos que se ejecutan con mayor eficiencia sobre este lenguaje.
- Indagar mas opciones que ofrece D3. Esta es una librería muy poderosa, pero es necesario tener buenas bases. Tal vez utilizamos ejemplos que no estaban bien construidos.
- Sentimos que el splitter ideal para este proyecto sería dividir cada archivo en varios de menor tamaño, pero copiando el encabezado en cada uno. De esta forma los trabajos solo debe leer la primera llave para saber la estructura del archivo, es muy posible que la ineficiencia de los trabajos se deba a revisar el tipo de linea que entra.