Taller 1 – Procesamiento escalable

Nicolas Jimenez
Documentación de resultados
Universidad de los Andes, Bogotá, Colombia
{en.jimenez}@uniandes.edu.co
Fecha de presentación: octubre 13 de 2020

Tabla de contenido

1 Introducción

Este primer taller tiene como objetivo la construcción de soluciones en ambientes Hadoop o Spark para el procesamiento de la información de un dataset, correspondiente al registro de viajes en taxi en la ciudad de Nueva York, usando técnicas de MapReduce en la implementación de las soluciones y que sean altamente escalables, de esta manera, hacer un análisis básico de la información en base a los resultados obtenidos por la solución propuesta.

2 Objetivos

2.1 Objetivo Principal

- Crear soluciones con técnicas MapReduce para desarrollar los retos propuestos, obtener resultados, analizarlos y lograr la escalabilidad.

2.2 Objetivos Específicos

- Entender los datos del dataset, su formato, su estructura y la forma de procesarlo con técnicas MapReduce.

- Realizar los retos de consultas básicas, buscando la forma de usar técnicas de MapReduce altamente escalables.
- Analizar los datos resultantes de consultas básicas para resolver preguntas de interés y lograr el entendimiento de los datos de forma organizada y comprensible.
- Visualizar los resultados de las consultas básicas en una aplicación web, la cual, permita el ingreso de los parámetros, ejecutar los requerimientos y obtener sus resultados de forma entendible.
- Probar las consultas básicas en diferentes entornos de escalabilidad, documentando los resultados y realizando comparativas entre los mismos.

3 Métodos y tecnología

3.1 Métodos

La metodología utilizada para la solución de los retos propuestos fue la creación de N jobs de acuerdo con la cantidad de retos presentados en el taller, cada job podía solicitar parámetros de entrada o no necesitarlos. Después, una interfaz gráfica encargada de capturar los parámetros de entrada se encargaba de ejecutar el job correspondiente al reto deseado, y por medio de solicitudes asíncronas obtenía los resultados y los mostraba en pantalla.

3.2 Tecnología

Las tecnologías usadas para el desarrollo de las soluciones del taller fueron: en primer lugar, los jobs fueron creados en lenguaje Java mediante la estructura Job, Mapper y Reducer. El frontend fue creado en lenguaje Python, usando el patrón MVC, donde el Controller realiza las solicitudes y ejecuta los scripts Hadoop para la inicialización de los jobs.

4 Análisis de los resultados

4.1 Análisis RA2

Según lo obtenido a través de la consulta MapReduce, cuyo resultado se encuentra en el siguiente enlace:

https://github.com/ABD-MINE4102/202020-Grupo09/blob/master/documentos/taller_1/RA/resultado_ra1.txt

De acuerdo con esto, los análisis que se puede sustentar son:

- En la temporada de San Valentín, es la época del año con mas demanda en los servicios de viajes por taxi.
- Los años con mayor cantidad de viajes para las tres temporadas del año tomadas (Acción de Gracias, Navidad, San Valentín) son del 2010 al 2014.
- La segunda temporada del año con mayor demanda es Acción de Gracias.
- Los años del 2016 al 2018 fueron los años con menor demanda para las tres temporadas.
- Se puede super que en la temporada de Navidad (específicamente el 25 de diciembre), los viajes por taxi son los de menor demanda que las otras tres temporadas, debido a que, las personas buscan estar ese día en familia, disfrutar los regalos (ya comprados en las vísperas) y descansar de todo el ajetreo de las épocas contiguas.

4.2 Escalabilidad de procesos con Hadoop y Map Reduce

a) Utilice un cluster de 1 nodo y el sub-dataset. Instale este cluster en su máquina individual de resultados.
b) Utilice un cluster de 4 nodos y el sub-dataset definido para el paso anterior.
c) Utilice un cluster de 4 nodos y el dataset completo.
d) Utilice un cluster de 20 nodos y el dataset completo.

- Para la comparación de estos dos escenarios primero tenemos que ver los resultados para cada uno de ellos:
  - **Escenario a – RF1:** 163,20 minutos
  - **Escenario b – RF1:** 38,01 minutos
  - **Escenario c – RF1:** 67,88 minutos
  - **Escenario d – RF1:** 18,08 minutos
  - **Escenario a – RF2:** 227,74 minutos
  - **Escenario b – RF2:** 46,88 minutos
  - **Escenario c – RF2:** 83,73 minutos
  - **Escenario d – RF2:** 24,68 minutos
  - **Escenario a – RF3:** 194,50 minutos
  - **Escenario b – RF3:** 44,05 minutos
  - **Escenario c – RF3:** 78,43 minutos
  - **Escenario d – RF3:** 25,16 minutos

- **Escenario a y b:** La diferencia de usar un clúster de 4 nodos a usar un nodo singular, es abismal, eso es debido a que un nodo singular tiene una capacidad de uso de jobs muy limitada y no está distribuida. Por lo tanto, la diferencia en tiempos supera mas del doble que usando el clúster de 4 nodos, así sea, usando un set de data reducida.
- **Escenario b y c:** Es importante resaltar que al disminuir la magnitud del dataset (50%), también el tiempo de procesamiento disminuirá casi en la misma proporción que la cantidad reducida de data, sin embargo, no es lo mismo procesar un set de datos grandes distribuidos que uno mas pequeño, debido a que la carga del sistema es similar y la carga mas grande se presenta al inicio del procesamiento del set de datos.
- **Escenario c y d:** Se puede notar que la comparación del procesamiento de un dataset completo en el clúster de 4 nodos vs el de 20 nodos, la diferencia es muy notoria, el proceso es terminado mucho más rápido en el de 20 nodos, eso quiere decir menos de la mitad del tiempo que el de 4 nodos. Con esto podemos deducir que el proceso de los job creados con MapReduce escalan bastante bien cuando se tiene mas recursos y nodos en el clúster.

4.3 Dificultades

- Una de las dificultades mas grandes de la generación de la solución fue en entendimiento de los datos de entrada, la lógica de como procesarlos y el manejo de la heterogeneidad.
- Probar las soluciones con datos de prueba sesgaba el funcionamiento completo de la solución, y hacer pruebas directamente son el set de datos directamente era muy demorado.
- El filtrado de datos en la etapa Mapper era la que requería mayor lógica y también la entrada de los datos de usuario.

4.4 Logros

- Una vez se entendía la lógica del diseño y ejecución de una solución MapReduce, establecer ciertas reglas era más fácil de ejecutarlas y tener resultados esperados.
- La implementación de un log de eventos tanto en el Mapper como en el Reducer, facilitó encontrar falencias en el desarrollo o inconsistencia de data.
- Una vez definido un dataset de pruebas basado en la data real, se podía reducir el tiempo de procesamiento y obtención de resultados, con baja probabilidad de fallos frente a la ejecución con el dataset real.

4.5 Posibles mejoras

- La información entregada al usuario aun debe ser refinada o mejor detallada para entender el significado de los textos y números mostrados ahí.
- Aunque la información esta entendible hasta cierto punto, hay secciones que deben ser explicadas, sin embargo, el resultado puede ser tomado y usado.

5 Conclusiones

- Las soluciones MapReduce no solo deben tener un clúster de varios nodos para aumentar rendimiento y velocidad de procesamiento, además, las soluciones deben ser altamente escalables y adecuadas para escalarlas horizontalmente.
- Los dataset se pueden procesar no solo como un conjunto de archivos en una carpeta, sino que también como un archivo comprimido en formato .tar, debido a que Hadoop puede leer y procesar estos archivos nativamente, pudiendo, tomar los archivos dentro del .tar para el procesamiento MapReduce.
- Es importante contextualizarse del formato y forma como los datos del set están organizados y presentados, gracias a esto se puede procesarlos de manera optima y abarcar la totalidad de los datos en el set.
- Las técnicas MapReduce permiten sacar filtros de la información en un lenguaje mas entendible tanto para usuarios como para desarrolladores, de esta manera se pueden hacer análisis y toma de decisiones sin necesidad de buscar manualmente en grandes cantidades de datos.
- Es posible el uso de varias tecnologías y estrategias de implementación de MapReduce con Hadoop, lo que permite versatilidad en las soluciones y manejo de varios lenguajes como Java o Python.

6 Enlace aplicación web

- Clúster de 20 nodos
  http://bigdata-cluster1-ambari.virtual.uniandes.edu.co:9009/
- Clúster de 4 nodos
  http://bigdata-cluster2-ambari.virtual.uniandes.edu.co:9009/

7 Bibliografía

1. *MapReduce: Simplified Data Processing.* **Ghemawat, Jeffrey Dean and Sanjay.** 1, s.l. : COMMUNICATIONS OF THE ACM, 2008, Vol. 51.

2. **Chenni Wu, Wei Zhang.** *Business Process Execution Based on Map Reduce Architecture.* Berlin : Springer-Verlag, 2012.

3. **Rob Hall, Josh Attenberg.** *Fast and Accurate Maximum Inner Product.* 2015.

4. **Diana Moise, Denis Shestakov, Gylfi Gudmundsson, Laurent Amsaleg.** *Indexing and Searching 100M Images with Map-Reduce.* 2013.

5. **Installing Anaconda on Linux. [En línea] anaconda. https://docs.anaconda.com/anaconda/install/linux/.**

6. Comandos Hadoop v1. [En línea] https://riptutorial.com/es/hadoop/example/13394/comandos-hadoop-v1.

7. Mucahid Kutlu, Gagan Agrawal. ***A Framework for Easy PArallelization of GEnomic Applications.*** **s.l. : IEEE, 2014.**