Instituto Tecnológico de Buenos Aires

MapReducing Parking Tickets

72.42 Programación Orientada a Objetos - 2024Q1

Alumnos:

Tomás Santiago Marengo, 61587

Abril Occhipinti, 61159

Santino Ranucci, 62092

Agustin Zakalik, 62068

Profesores:

Ing. Marcelo Turrín

Ing. Franco Román Meola

Índice

1. Introducción

- Cómo se diseñaron los componentes de cada trabajo MapReduce, qué decisiones se tomaron y con qué objetivos. Además alguna alternativa de diseño que se evaluó y descartó, comentando el porqué.
- El análisis de los tiempos para la resolución de cada query: En caso de poder, analizar la diferencia de tiempos de correr cada query aumentando la cantidad de nodos (hasta 5 nodos) en una red local. De no poder, intentar predecir cómo sería el comportamiento.
- Potenciales puntos de mejora y/o expansión.
- La comparación de los tiempos de las queries ejecutándose con y sin Combiner.
- Otro análisis de tiempos de ejecución de las queries utilizando algún otro elemento de optimización a elección por el grupo.
- Para todos los puntos anteriores, no olvidar de indicar el tamaño de los archivos utilizados como entrada para las pruebas (cantidad de registros).

2. Anotaciones

3. Cuestiones matemáticas

- Elección de la key del mapa inicial: en un caso ideal tendríamos bastantes nodos y quisieramos que se distribuya de una forma equitativa para no sobrecargar a ningún nodo. Para esto hay que entender COMPLETAMENTE la naturaleza de nuestro dataset.

  Ejemplos:

  Query 2: top 3 infracciones por barrio.

  Si yo decido elegir como key el barrio, cada barrio irá a un nodo, es decir, no existirán dos nodos tales que contengan infracciones del mismo barrio. Entonces me tengo que preguntar varias cosas. Lo mejor es saber de antemano las posibles respuestas a mis preguntas, si fuesen ciudades de buenos aires quizás entendamos más. Pero bueno, las preguntas podrían ser, existe algun barrio que tenga gran porcentaje de las infracciones de transito? Si fuese así, entonces la key no debería ser unicamente barrio, ya que habría un nodo sobrecargado.

  SEGUIR CON MAS EJEMPLOS

- **Hablar del parallel read.**

Reading a single file at multiple positions concurrently wouldn't let you go any faster (but it could slow you down considerably).

Instead of reading the file from multiple threads, read the file from a single thread, and parallelize the processing of these lines. A singe thread should read your CSV line-by-line, and put each line in a queue. Multiple working threads should then take the next line from the queue, parse it, convert to a request, and process the request concurrently as needed. The splitting of the work would then be done by a single thread, ensuring that there are no missing lines or overlaps.

- **Hablar del .json que recibe las columnas del CSV**
- **Hablar del DateFormats que recibe todos los tipos de formatos de fecha con los que se trabaja**
- Query 2: Desempate por descripcion de la infraccion. Si en un barrio existen dos infracciones que tengan la misma cantidad de multas. El TP no dice nada asi que elegimos desempatar asi.
- Hablar del TopNSet y de la generalización de la query2. Igualmente el "3" está wrappeado en el .sh para respetar la consigna, pero se podría elegir cualquier N como en la query 2.
- Query 3: hablar sobre lo de BigDecimal.
- Query 4 y cualquiera: llegamos al Collector y nos preguntamos el orden de magnitud de nuestra data según la naturaleza de nuestro dataset. En el caso de la query 4 tenemos los pares (barrio, patente). La cantidad de barrios de NYC son 62. Posiblemente alguna ciudad importante no tenga más de 100. Luego tenemos las patentes, que podrían ser ordenes de cientos de miles o de millones. Por lo que el orden de magnitud es bastante alto y tiene sentido hacer otro map reduce. No así si fuera por ejemplo la query 2, donde tenemos los pares (barrio, tipo de fraccion), donde los dos están cerca de los 100, por tanto #barrio*tipo_infraccion = 100 * 100 = 10000, que no justifica distribuir la data para otro procesamiento, sino que se puede realizar de manera rápida en el Collator.