ITBA
Instituto Tecnológico de Buenos Aires

Trabajo Práctico Especial II : Multas de Estacionamiento

Programación de Objetos Distribuidos

Fecha de entrega: 13/06

Alumnos:
Nicolas Suarez Durrels 62468
Bruno Enzo Baumgart 62161
Santiago Jose Hirsch 62169
Camila Sierra Pérez 60242

Docentes:
Marcelo Emiliano Turrin
Franco Román Meola

Grupo 12

Análisis de las queries

Query 1:

Total de Multas por Infracción

El objetivo de esta consulta es obtener la cantidad total de multas para cada tipo de infracción, listadas en orden descendente por cantidad y luego alfabéticamente en caso de empate.

Descripción de los modelos: Para llevar a cabo esta consulta, el proceso se desglosa en varias etapas:

1. **Mapper**: En esta fase, se verifica si cada infracción está presente en la lista de infracciones del archivo CSV. Si es así, el mapper emite un par `(infractionCode, 1)`, donde `infractionCode` es el código de la infracción y `1` indica una ocurrencia de dicha infracción.
2. **Combiner y Reducer**: Estas etapas se encargan de sumar los valores emitidos por el mapper. El combiner realiza una suma preliminar de los valores para cada `infractionCode` localmente antes de enviarlos al reducer, que lleva a cabo la suma total de las ocurrencias para cada tipo de infracción.
3. **Collator**: Finalmente, el collator toma los códigos de infracción y, utilizando una tabla de referencia, los convierte en nombres de infracción legibles. Luego, crea objetos `InfractionCount` que contienen el nombre de la infracción y su cantidad total.

Tiempos de resolución con diferentes nodos/optimizaciones

Para todas las queries realizamos 4 corridas para cada número de nodos, y en los tiempos se especifica el promedio - la desviación estándar. En los otros análisis se realizaron 2 pruebas de cada uno, con NYC y CHI.

-Cantidad de registros utilizados como entrada: 100.000

Nodos	Tiempos (promedio - desvío estándar)
1	1.094s - 0.1736s
2	265.42175s - 14.000s
3	199.55525s - 2.717s

-Cantidad de registros utilizados como entrada: 5.000.000 y un solo nodo
-Hicimos dos corridas

Tiempos (promedio - desvío estándar)	
Con combiner	**Sin combiner**
38.9565s - 1.8757s	45.024s - 1.9706s

Tiempos (promedio - desvío estándar)	
Con optimizador	**Sin optimizador**
38.9565s - 1.8757s	42.852s - 0.613s

Query 2:

Top 3 Infracciones más populares de cada barrio

El objetivo de esta consulta es identificar las tres infracciones más frecuentes en cada barrio.

Descripción de los modelos: Este proceso es similar al de la Query 1, con algunas modificaciones clave:

1. **Mapper**: El mapper emite un par `(countyName, infractionCode)` junto con un valor de `1` por cada registro, donde `countyName` es el nombre del barrio y `infractionCode` es el código de la infracción.
2. **Combiner y Reducer**: Al igual que en la Query 1, estas etapas suman los valores emitidos, pero ahora lo hacen para cada par `(countyName, infractionCode)` para obtener el total de ocurrencias de cada infracción en cada barrio.
3. **Collator**: En esta fase, se crea un mapa `(countyName, InfractionCount)` utilizando los datos agregados. Luego, se ordenan las infracciones por cantidad dentro de cada barrio y se seleccionan las tres con mayor número de ocurrencias. Se asegura que la salida esté ordenada alfabéticamente por el nombre del barrio.

Tiempos de resolución con diferentes nodos/optimizaciones:

-Cantidad de registros utilizados como entrada: 100.000

Nodos	Tiempos (promedio - desvío estándar)
1	1.1835s - 0.1054s
2	262.577s - 15.704s
3	215.97925s - 6.715s

-Cantidad de registros utilizados como entrada: 5.000.000 y un solo nodo

Tiempos (promedio - desvío estándar)	
Con combiner	**Sin combiner**
42.4095s - 3.3784s	44.634s - 3.886s

Query 3:

Top N agencias con mayor porcentaje de recaudación

Esta consulta tiene como objetivo identificar las agencias que emiten infracciones y calcular el porcentaje de recaudación de cada una, listando las N agencias principales.

Descripción de los modelos:

1. **Mapper**: En esta etapa, el mapper emite un par `(agencyName, amount)` para cada infracción, donde `agencyName` es el nombre de la agencia que emitió la infracción y `amount` es el monto de la multa.
2. **Combiner y Reducer**: Estas etapas se encargan de sumar los montos de las multas para cada agencia. El combiner realiza una suma preliminar local y el reducer obtiene la suma total de los montos emitidos por cada agencia.
3. **Collator**: El collator primero calcula el total recaudado por todas las agencias. Luego, determina el porcentaje de recaudación de cada agencia dividiendo el monto total recaudado por la agencia entre el monto total general y truncando el resultado a dos decimales. Se seleccionan las N agencias con los porcentajes más altos, ordenadas en forma descendente por porcentaje y alfabéticamente por nombre en caso de empate.

Tiempos de resolución con diferentes nodos/optimizaciones:

-Cantidad de registros utilizados como entrada: 100.000

Nodos	Tiempos (promedio - desvío estándar)
1	0.13675s - 0.02614s
2	3.82025s - 0.08324s
3	3.86175s - 0.1212s

-Cantidad de registros utilizados como entrada: 5.000.000 y un solo nodo

Tiempos (promedio - desvío estándar)	
Con combiner	**Sin combiner**
3.9665s - 0.2565s	4.7105s - 0.3805s

Tiempos (promedio - desvío estándar)	
Con optimizador	**Sin optimizador**
3.9665s - 0.2565s	5.6255s - 0.0975s

Query 4:

Patente con más infracciones de cada barrio en el rango [from, to]

Esta consulta tiene como objetivo encontrar, para cada barrio, la patente con la mayor cantidad de multas dentro de un rango de fechas específico.

Descripción de los modelos:

1. **Mapper**: El mapper verifica si la fecha de cada registro se encuentra dentro del rango especificado (`from`, `to`). Si es así, emite un par `(countyName, plate)` junto con un valor de `1`.
2. **Combiner y Reducer**: Estas etapas suman los valores emitidos por el mapper para cada par `(countyName, plate)`, obteniendo el total de infracciones por patente en cada barrio.

3. **Collator**: El collator agrupa los datos por `countyName` y selecciona la patente con la mayor cantidad de infracciones en cada barrio. Los resultados se ordenan alfabéticamente por el nombre del barrio.

Tiempos de resolución con diferentes nodos/optimizaciones:

-Cantidad de registros utilizados como entrada: 100.000

Con los datos de CHI:

Nodos	Tiempos (promedio - desvío estándar)
1	0.9075s - 0.0733s
2	28.4605s - 0.3617s
3	17.2055s - 0.2042s

Con los datos de NYC:

Nodos	Tiempos (promedio - desvío estándar)
1	0.137s - 0.0278s
2	3.573s - 0.0178s
3	3.0085s - 0.0651s

Para esta query separamos las corridas según el dataset dado que como hay que especificar la fecha, no van a tener la misma cantidad de infracciones en ambos, por lo que los tiempos varían de forma considerable.

-Cantidad de registros utilizados como entrada: 5.000.000 y un solo nodo

Con los datos de NYC:

Tiempos	
Con combiner	**Sin combiner**
79.496s	96.542s

Tiempos	
Con optimizador	**Sin optimizador**
79.496s	67.455s

Con los datos de CHI:

Tiempos	
Con combiner	**Sin combiner**
27.67s	31.677s

Tiempos	
Con optimizador	**Sin optimizador**
27.67s	25.277

Query 5:

Pares de infracciones que tienen, en grupos de a cientos, el mismo promedio de monto de multa

El objetivo de esta consulta es encontrar pares de infracciones cuyos promedios de monto de multa caen dentro del mismo rango de cientos.

Descripción de los modelos:

1. **Mapper**: El mapper emite el código de infracción y el monto de la multa para cada registro.
2. **Combiner y Reducer**: Estas etapas suman los montos y cuentan las infracciones para cada código de infracción.
3. **Collator**: El collator agrupa las infracciones en rangos de cientos basados en sus promedios de monto de multa. Luego, se generan pares de infracciones dentro de cada grupo, asegurando que no se dupliquen los pares (es decir, si se lista `(IN1, IN2)`, no se lista `(IN2, IN1)`). Los resultados se ordenan de manera descendente por grupo y alfabéticamente por el nombre de la infracción dentro de cada grupo.

Tiempos de resolución con diferentes nodos/optimizaciones:

-Cantidad de registros utilizados como entrada: 100.000

Nodos	Tiempos (promedio - desvío estándar)
1	0.81375s - 0.02086s
2	242.6255s - 2.72s
3	223.438s - 4.42s

-Cantidad de registros utilizados como entrada: 5.000.000 y un solo nodo

Tiempos (promedio - desvío estándar)	
Con combiner	**Sin combiner**
39.505s - 0.435s	40.667s - 0.229s

Tiempos (promedio - desvío estándar)	
Con optimizador	**Sin optimizador**
39.505s - 0.435s	43.657s - 0.209s

Conclusiones generales

A primera vista, algo que nos llamó la atención fue el salto temporal entre un único nodo y múltiples. Analizando un poco más esto, nos dimos cuenta que este aumento se debe a la transferencia de los datos entre los nodos. Esto se observó mediante la actividad de la placa de red de los nodos.

De todas maneras, podemos ver que el tiempo (en la mayoría de los casos) se reduce al pasar de 2 nodos a 3 nodos. Esto podría deberse a una mejor paralelización de los datos, o un balance de cargas más parejo entre los nodos, evitando cuellos de botella. Utilizando más nodos, podríamos esperar que el tiempo se reduzca.

En cuanto al combiner, vimos consistentemente que en el caso de un único nodo y nuestro set de datos, el tiempo fue menor al utilizarlo.

Por último también comparamos los tiempos al utilizar *parallelstream* para ordenar las listas. Al compararlo con la utilización de sort, también observamos que el tiempo es menor en la mayoría de las queries, salvo en el caso de la query 4 que el tiempo fue menor utilizando sort.

Potenciales puntos de mejora

Algo que notamos que nos consumía un gran porcentaje del tiempo, sobre todo en el caso de un nodo sólo, fue la lectura del archivo que podría haber sido paralelizada u optimizada para una mejor performance general. En los casos con múltiples nodos, el análisis podría ser mucho más preciso si hubiéramos contado con una mejor manera para conectar los diferentes nodos entre si, en vez de utilizar Wifi como lo hicimos.

Dificultades

En cuanto al análisis de múltiples nodos, al conectarse por red en vez de por ejemplo por cable directo, la transferencia de los datos era muy costosa y fue dificultoso realizar un análisis de tiempos adecuados. Pudimos correr las queries solo con hasta 3 nodos y estimamos como hubieran sido las tendencias al agregar nodos.