TRABAJO PRÁCTICO ESPECIAL 2

Segundo Cuatrimestre de 2023
72.42 - Programación de Objetos Distribuidos

Grupo 6
Mattiussi, Agustin 61361
Ortu, Agustina Sol 61548
Sasso, Julian Martin 61535
Vasquez Currie, Malena 60072

Índice

1. Decisiones de diseño e implementación

Se plantean 3 clases que seran utilizadas a lo largo del proyecto en las distintas querys: **BikeTrip** (identifica los viajes realizados en bicicleta contando con startStationId, startDate, endStationId, endDate e isMember), **Station** (identifican las estaciones como id, name, latitude, longitude) y **Pair** (permite agrupar dos objetos llamados *first* y *second* como un único objeto).

2. Análisis de las queries

2.1. Query 1: Viajes entre estaciones

El objetivo de esta query era obtener la cantidad de viajes entre estaciones. Para lograrlo no solo se reutilizaron las clase s*BikeTrip* y*Pair* mencionadas previamente sino que tambien se creó *BikeTripCount*. Esta última contiene los nombres de las estaciones y el count.

El Mapper recibe un *BikeTrip* y a partir de ello emite un *Pair* conteniendo los ID de las estaciones de salida/destino. En cada caso setea el count inicial en 1. Aparte del Reducer, que se utiliza para ir incrementando el count, se implementó un Combiner y Collator. El primero fue pensado para optimizar el análisis, recibiendo un *Pair* de estaciones y sumando viajes de a 'chunks'. El Collator sirve para obtener los nombres de las estaciones a partir de su ID, además de que ordena el resultado de la forma solicitada.

2.1.1. Tiempos de resolución para diferente cantidad de nodos

Cantidad de registros utilizados como entrada: 1.000.000 (un millón).

Cantidad de nodos	Tiempo de Resolución
1 nodo	2m 56s
2 nodos	35m 36s
3 nodos	42m 18s
4 nodos*	-

(*) El caso de 4 nodos no se pudo realizar para esta query por problemas de conectividad.

2.1.2. Tiempos de resolución para diferente cantidad de nodos

Cantidad de nodos: 1 (uno).
Cantidad de registros utilizados como entrada: 1.000.000 (un millón).

Tiempos de Resolución	
con Combiner	sin Combiner
2m 56s	2m 54s

2.1.3. Tiempos de resolución con optimizaciones

Cantidad de nodos: 1 (uno).
Cantidad de registros utilizados como entrada: 1.000.000 (un millón).

Tiempos de Resolución	
con parallel stream sorting	sin parallel stream sorting
2m 44s	2m 56s

2.2. Query 2: Top N estaciones con mayor promedio de distancia aproximada

En esta query se pide un listado de estaciones junto el promedio de la distancia de los viajes que salen de la misma. Para esto se implemento la clase *DistanceJourney* la cual cuenta con stationId (id de la estación de salida), journeysAmount (cantidad de viajes que salen de la misma) y sumDistances (suma de las distancias recorridas).

Para el map recibimos un *BikeTrip* y chequeamos que la estación de salida y la de llegada pertenezcan a las estaciones listadas en stations.csv y que el viaje no vaya de una estación a la misma. Luego se calcula la distancia con la formula Haversine y se crea un *DistanceJourney* con 1 como cantidad de viajes realizados. Emitiendose el id y esa instancia de *DistanceJourney*.

Luego el combiner, al igual que el reducer, se encarga de sumar las distancias de cada estación y la cantidad de viajes realizados devolviendo un único *DistanceJourney* por estacion. Posteriormente el collator se encarga de devolver un listado de *Pair* con el nombre de la estación y el promedio de sus distancias acumuladas ordenadas descendentemente por promedio de distancias y en segundo lugar descendentemente por nombre de estación. Finalmente selecciono los primeros N elementos de esa lista.

2.2.1. Tiempos de resolución para diferente cantidad de nodos

Cantidad de registros utilizados como entrada: 1.000.000 (un millón).

Cantidad de nodos	Tiempo de Resolución
1 nodo	57s
2 nodos	20m 27s
3 nodos	32m 56s
4 nodos*	-

(*) El caso de 4 nodos no se pudo realizar para esta query por problemas de conectividad.

2.2.2. Tiempos de resolución con y sin combiner

Cantidad de nodos: 1 (uno).
Cantidad de registros utilizados como entrada: 1.000.000 (un millón).

Tiempos de Resolución	
con Combiner	sin Combiner
57s	56s

2.2.3. Tiempos de resolución con optimizaciones

Cantidad de nodos: 1 (uno).
Cantidad de registros utilizados como entrada: 1.000.000 (un millón).

Tiempos de Resolución	
con parallel stream sorting	sin parallel stream sorting
57s	57s

2.3. Query 3: Viaje más largo (en minutos) por estación

Para la tercer Query, se requería obtener el viaje más largo (en minutos) para cada estación de salida, desempatando por fecha de inicio más reciente y ordenando los resultados descendientemente por tiempo y luego alfabéticamente por nombre de estación de origen.

En este caso particular, se implementó la clase *FinishedBikeTrip* para contener los resultados de cada etapa. La misma posee como atributos el ID y el nombre de la estación de destino, la duración del viaje en minutos y la fecha de inicio del mismo.

Para resolver la Query 3, se desarrollaron, además de un Mapper y un Reducer, un Combiner y un Collator. En la etapa del *Mapper*, se toma la información de los viajes realizados (representados como instancias de *BikeTrip*) y se emite como clave el ID de la estación de origen y como valor un FinishedBikeTrip con la información correspondiente para cada viaje.

Posteriormente, el *Combiner* recibe los valores de cada clave y emite el que tenga mayor duración dentro de un *chunk*, para que luego el *Reducer* emita el máximo de estos valores parciales.

Finalmente, en el *Collator* se reemplazan los IDs de las estaciones de origen y destino por el nombre de cada estación y se ordenan los resultados.

2.3.1. Tiempos de resolución para diferente cantidad de nodos

Cantidad de registros utilizados como entrada: 1.000.000 (un millón).

Cantidad de nodos	Tiempo de Resolución
1 nodo	30s
2 nodos	10m 47s
3 nodos	16m 54s
4 nodos*	7m 35s

(*) El caso de 4 nodos fue probado solo con 100.000 datos.

2.3.2. Tiempos de resolución con y sin combiner

Cantidad de nodos: 1 (uno).
Cantidad de registros utilizados como entrada: 7.000.000 (siete millones).

Tiempos de Resolución	
con Combiner	sin Combiner
3m 14s	3m 15s

2.3.3. Tiempos de resolución con optimizaciones

Cantidad de nodos: 1 (uno).
Cantidad de registros utilizados como entrada: 7.000.000 (siete millones).

Tiempos de Resolución	
con parallel stream sorting	sin parallel stream sorting
3m 18s	3m 14s

2.4. Query 4: Días de afluencia neta positiva, neutra y negativa por estación

Por último, la cuarta Query nos pedía obtener la cantidad de días con afluencia neta neutra de esa estación y la cantidad de días con afluencia neta negativa de esa estación, dentro de un rango de fechas que se indica por parámetro.

Para su resolución, además de implementar Mapper, un Reducer, un Combiner y un Collator, se desarrolló la clase *AffluenceInfo* para contener los datos de afluencia de cada estación.

En el *Mapper* se valida que los viajes, representados como instancias de *BikeTrip*, estén dentro del rango de fechas provisto. Si esto se cumple, se realizan 2 emits de instancias de Pair: uno que tiene como clave el ID de la estación de origen y como valor "1", y otro que tiene como clave el ID de la estación de llegada y como valor 1".

Luego, en el *Combiner* crea un mapa que tiene como clave una fecha (representada como *LocalDate*) y como valor la afluencia para ese dia.

El *Reducer* es el encargado de *colapsar* los mapas provistos por el Combiner para obtener el resultado final: un mapa que tiene como clave el ID de cada estación y como valor una instancia de AffluenceInfo con los datos requeridos.

Finalmente, el *Collator* reemplaza los IDs de cada estación por el nombre y ordena los resultados al igual que en queries anteriores.

2.4.1. Tiempos de resolución para diferente cantidad de nodos

Cantidad de registros utilizados como entrada: 1.000.000 (un millón).

Cantidad de nodos	Tiempo de Resolución
1 nodo	30s
2 nodos	11m 4s
3 nodos	18m 33s
4 nodos*	8m 13s

(*) El caso de 4 nodos fue probado solo con 100.000 datos.

2.4.2. Tiempos de resolución con y sin combiner

Cantidad de nodos: 1 (uno).

Cantidad de registros utilizados como entrada: 7.000.000 (siete millones).

Tiempos de Resolución	
con Combiner	sin Combiner
3m 24s	3m 20s

2.4.3. Tiempos de resolución con optimizaciones

Cantidad de nodos: 1 (uno).

Cantidad de registros utilizados como entrada: 7.000.000 (siete millones).

Tiempos de Resolución	
con parallel stream sorting	sin parallel stream sorting
3m 23s	3m 24s

3. Potenciales puntos de mejora y/o expansión

Una mejora que se nos viene a la cabeza al ver la cantidad de líneas que posee el archivo *bikes.csv*, es la paralelización de la lectura del mismo. Teniendo en cuenta, además, que la cantidad total de objetos *BikeRide* creados a partir del archivo no entran en memoria y que por lo tanto deben ser subidos a *Hazelcast* en *batches*, poseer varios hilos que realicen esta tarea en simultáneo puede acelerar el procesamiento de las queries.

Por otra parte, una expansión quizás ambiciosa, podría ser implementar una lógica del lado del servidor que, en función de la cantidad de objetos subidos a *Hazelcast*, agregue nodos al clúster de forma dinámica, considerando cuándo sería necesario (por falta de memoria) y eficiente hacerlo (teniendo en cuenta que más nodos implican más lentitud en el procesamiento).

4. Dificultades

Al momento de comenzar las pruebas del trabajo para más de un nodo, donde entró en juego la serialización de datos a través de la red, nos encontramos con excepciones en cada uno de los *Mapper*, donde se mencionaba que los mismos no eran serializables. Se concluyó que por alguna razón, se estaban intentar serializar los *Maps* obtenidos de la implementación de *HazelcastInstanceAware*. Al añadir la palabra reservada *transient* en la declaración de los mismos (que indica que no deben ser serializados), se solucionó el problema.

Como se mencionó en el análisis de las queries, al momento de hacer pruebas con 4 nodos en clase tuvimos problemas de conexión y no pudimos completar todas las queries con esa cantidad de nodos (se llego a correr query3 y query4 sin problema) por lo que tuvimos que seguir las pruebas con una menor cantidad de nodos.