Programación de Objetos Distribuidos

Trabajo Práctico Especial

Integrantes:

Bendayan, Alberto (Legajo: 62786)

Boullosa Gutierrez, Juan Cruz (Legajo: 63414)

Deyheralde, Ben (Legajo: 63559)

Profesores:

Meola, Franco Román

Turrin, Marcelo Emiliano

Componentes de cada trabajo MapReduce

Para resolver las cuatro queries utilizamos las mismas etapas en todas. Estas son KeyPredicate, Mapper, Combiner, Reducer y Collator explicadas a continuación para cada una de las queries realizadas.

Query1

Para resolver la Query1 necesitamos de los tres archivos CSV por las condiciones requeridas. Entonces a medida que leemos cada archivo, vamos insertando cada fila en una estructura de Hazelcast.

Para guardar los Tickets, generamos un IMap<Ticket, InfractionAndAgency> que es sobre el cual luego hacemos el trabajo de MapReduce. Con esta decisión tuvimos bastantes "ida y vueltas". En primer lugar empezamos con un IMap<InfractionAndAgency, Ticket> pero al ver los resultados de la query nos dimos cuenta que al hacer un "put" sobre el IMap, estábamos pisando el valor de esa clave entonces luego al hacer el MapReduce, todos los pares InfractionAndAgency nos daban como resultado "1".

Debido a esto nos dimos cuenta que podíamos usar un MultiMap, ya que aparentemente solucionaba este problema al poder guardar más de un Value por cada Key pero después de múltiples intentos, debugging con logs y demás, no pudimos hacer que funcione. Es por esto que volvimos a la solución que explicamos primero. Sin embargo, todo lo modificado se encuentra en la rama MultiMapFailed para poder observar los cambios realizados que no se encuentran funcionando.

Para guardar las infracciones hicimos un IMap<String, String> y para las agencias un Set<String>.

KeyPredicate (CheckInfractionAndAgencyExistence): el predicate en su constructor recibe dos Sets que contienen las agencias e infracciones válidas. Estos Sets los generamos a partir de los archivos de agencias y el de las infracciones. Luego de investigar cómo poder pasarle parámetros a un Predicate, llegamos a la conclusión de que al ser Sets relativamente "chicos" (ya que nunca va a ser un listado interminable de infracciones o agencias) era una opción viable. El predicate funciona como filtro previo al mapper ya que evalúa si la fila que le estamos mandando tiene una agencia existente en el listado de agencias y si la infracción tiene una infracción asociada en el listado de infracciones.

Mapper (InfractionAndAgencyMapper): el mapper de esta query resultó bastante trivial y similar al que vimos en clase para resolver el “hello world”. Por cada par <Ticket, InfractionAndAgency> emitimos un par <InfractionAndAgency, 1>.

Combiner (InfractionAndAgencyCombiner): siguiendo la lógica del caso “hello world”, el combiner junta los valores iguales de InfractionAndAgency y los suma a nivel nodo.

Reducer (InfractionAndAgencyReducer): hace lo mismo que el combiner pero ya con los valores finales y sumando los resultados de todos los nodos.

Collator (InfractionAndAgencyCollator): en el collator lo que hacemos es crear un SortedSet con un Comparator que ordene según lo requerido en la consigna, es decir, orden descendente por total, alfabético por infracción y alfabético por agencia. Este SortedSet es de tipo InfractionAndAgencyTotal que es una composición entre InfractionAndAgency y el total resultante del reducer.

Query2

Para esta query, solamente necesitamos dos de los tres CSV provistos por la cátedra. El primero de estos fue el de tickets el cual decidimos almacenar en un IMap <Ticket, Long> donde Long es el monto de la multa. Al igual que la query1, tuvimos la intención de usar un MultiMap pero luego de una serie de pruebas y fallas hemos decidido usar el IMap, dejando la versión (no funcional) de MultiMap en otra rama MultiMapFailed. La otra colección que usamos fue un Set con las agencias, que luego usamos en el KeyPredicate.

KeyPredicate (CheckAgencyExistence): validamos que la agencia mencionada en el ticket, exista en el set de agencias. En caso que no exista, no tenemos en cuenta ese “ticket” para lo que sigue.

Mapper (AgencyYearMonthMapper): al igual que la query anterior, este paso fue trivial pero con la diferencia de que emitimos un <AgencyYearMonth, Long> donde el Long en lugar de ser un 1 como en la query1 para que funcione como contador, es el monto del ticket para luego poder hacer la cuenta del acumulado.

Combiner (AgencyYearMonthCombiner): muy similar al caso anterior donde agrupamos y sumamos los values de los elementos con una misma clave.

Reducer (AgencyYearMonthReducer): funcionalidad muy similar al combiner pero ya dejando los valores definitivos.

Collator (AgencyYearMonthCollator): acá creamos un SortedSet donde vamos agregando los valores a mostrar con un Comparator que cumple con lo requerido en

la consigna. Se encuentra gran parte de la lógica y es bastante más complejo que en la query1, con una variable acumuladora que reinicia al cambiar de año o de agencia.

Query3

En este caso, utilizamos un único IMap del CSV de tickets donde la clave es CountyPlateInfractionAndDate que contiene el barrio, la patente, la fecha, la infracción cometida y un ticketId que nos sirve de identificador. El valor del mapa es el ticket completo.

KeyPredicate (CheckDatesRange): recibe como parámetros “from” y “to” que son los valores ingresados por la terminal y filtra el mapa para que solo queden valores que estén entre esas fechas.

Mapper (CountyMapper): cuenta con una funcionalidad sencilla donde realizamos dos cosas, cambiar la clave del mapa pasando a un CountyInfractionAndPlate donde no tenemos más la fecha ya que fue filtrada en el Predicate y emitimos un uno por cada clave.

Combiner (CountyCombiner): muy similar a los casos anteriores donde agrupamos y sumamos los values de los elementos con una misma clave.

Reducer (CountyReducer): funcionalidad muy similar al combiner pero ya dejando los valores definitivos.

Collator (CountyCollator): recibe un parámetro “n” que define la cantidad de infracciones mínimas para considerar reincidente. Luego se realizan una serie pasos:

- Se crean dos mapas, uno para almacenar todas las placas por barrio y el otro para almacenar las placas repetidas en un barrio.
- Se iteran los datos y se van agregando los valores correspondientes a los mapas mencionados anteriormente.
- Se crea un sortedSet para retornar los valores
- Por cada barrio se analiza si hay un porcentaje de reincidencia mayor a 0 y si es así se lo agrega al sortedSet de resultados.

Query4

Para esta query utilizamos dos de los tres archivos brindados, los cuales almacenamos en dos archivos diferentes. El primero de ellos, el de ticket, lo hemos decidido almacenar en un IMap donde la clave es el ticket en sí y su valor es

InfractionAndAmount. Luego, tenemos otro IMap que se encarga de guardar las infracciones donde su clave es el ID y su valor, la infracción en sí.

KeyPredicate (CheckAgency): en este caso el KeyPredicate no es para verificar la existencia de la misma (como en la query1) sino para filtrar por la agencia que se pasa como parámetro de la query.

Mapper (InfractionAndAmountMapper): cuenta con una funcionalidad muy sencilla donde recibe un <Ticket, InfractionAndAmount> y devuelve un <String, InfractionAndAmount> donde InfractionAndAmout se mantiene igual y el String vendría a ser el id del ticket.

Combiner (InfractionAndAmountCombiner): pone dentro de un InfractionAndAmountStats el valor mínimo y máximo que se encontró en ese nodo. Por cada <String, InfractionAndAmount> va actualizando en caso de que corresponda el valor mínimo o el valor máximo y esto se hace en una clase nueva que es InfractionAndAmountStats (ValueOut) que guarda el importe mínimo y máximo que se registró hasta este momento del proceso. A diferencia de todos los Combiners anteriores, el ValueIn es de distinto tipo que el ValueOut. Por este motivo, si se quiere probar la query4 sin Combiner, hay que utilizar otro Reducer.

Reducer (InfractionAndAmountReducer y InfractionAndAmountReducer2): funcionalidad similar al combiner, donde se juntan todos los nodos y se termina de conseguir el valor mínimo y máximo pero de todos los datos. En caso de ejecutar la query sin Combiner, hay que utilizar el InfractionAndAmountReducer2 ya que el Combiner cambia de tipo el ValueIn del Reducer.

Collator (InfractionAndAmountCollator): ordena todos los valores en unSortedSet con un Comparator que cumple lo requerido en la consigna y finalmente devuelve los primeros n valores (n se recibe como parámetro).

Pruebas

Los CSV utilizados para las pruebas tienen una longitud de:

- *agenciesCHI.csv:* 7 líneas
- *agenciesNYC.csv:* 33 líneas
- *infractionsCHI.csv:* 129 líneas
- *infractionsNYC.csv:* 98 líneas
- *ticketsCHI.csv: 1754* líneas

- *ticketsNYC.csv: 3893* líneas

Combiner vs Sin combiner (1 solo nodo):

		query1	query2	query3	query4
NYC					
COMBINER	inicio	15:23:09.8684	15:24:19.5134	15:27:05.8322	15:28:26.2271
	fin	15:23:10.4144	15:24:20.2154	15:27:06.7782	15:28:26.4371
	diferencia	**0,546**	**0,702**	**0,946**	**0,21**
NO COMBINER	inicio	15:33:18.1335	15:34:55.4601	15:35:59.1897	15:42:53.6232
	fin	15:33:19.0745	15:34:56.1551	15:36:00.9937	15:42:55.2445
	diferencia	**0,941**	**0,695**	**1,804**	**1,6213**
DIFERENCIA COMBINER VS NO COMBINER		**0,395**	**-0,007**	**0,858**	**1,4113**
		query1	**query2**	**query3**	**query4**
CHI					
COMBINER	inicio	15:22:31.0895	15:24:52.3781	15:26:10.1494	15:28:57.6064
	fin	15:22:31.6711	15:24:52.8701	15:26:11.1254	15:28:57.7174
	diferencia	**0,5816**	**0,492**	**0,976**	**0,111**
NO COMBINER	inicio	15:34:25.3212	15:35:31.7084	15:36:25.3231	15:44:01.4124
	fin	15:34:25.6802	15:35:32.3444	15:36:25.8191	15:44:01.7894
	diferencia	**0,359**	**0,636**	**0,496**	**0,377**
DIFERENCIA COMBINER VS NO COMBINER		**-0,2226**	**0,144**	**-0,48**	**0,266**

La Query3 fue ejecutada con los parámetros: n=2 ; from=01/01/2000 ; to=01/01/2024)

La Query4 fue ejecutada con los parámetros n=4 ; agency = TRAFFIC

Nodos y combiner:

Utilizamos 2 nodos, el 3, 4 y el 5 lo obtuvimos de manera teórica. Esto se debe a que ninguno de los integrantes del grupo tuvo la posibilidad de usar más de 2 computadoras. Todos las tablas están hechas en base al CSV de NYC.

Query1:

Nodos	Tiempo sin combiner (ms)	Tiempo con combiner (ms)
1	941	546
2	78103	63245
3	59054	41491
4	44290,5	31266,288
5	35432,4	24973,41

Query2:

Nodos	Tiempo sin combiner (ms)	Tiempo con combiner (ms)
1	695	702
2	5204	4807
3	4707	4149
4	4304,695	3516,784
5	3997,685	2792,486

Query3:

Nodos	Tiempo sin combiner (ms)	Tiempo con combiner (ms)
1	1804	946
2	70854	48547
3	60145,3	39548,2
4	53241,3	32954,69
5	47297,685	25412,26

Query4:

Nodos	Tiempo sin combiner (ms)	Tiempo con combiner (ms)
1	1612,3	210
2	67716	7948
3	61387,3	6341,4
4	55411,9	5421,8
5	49891,615	4881,5

Potenciales puntos de mejora y expansión

Notamos que las claves de algunos de nuestros IMap son más grandes de lo ideal, esto se debe a que el IMap no acepta colisiones por lo que teníamos errores en las queries. Nuestra primera solución fue poner el ticket completo como key. Luego nos dimos cuenta que era ineficiente por lo que tratamos de migrar a un MultiMap pero nunca nos llegó a funcionar. Todos los cambios realizados para poder implementar el MultiMap están en la branch MultiMapFailed.

Anexo

Query1

Con combiner:

NYC: 07/11/2024 15:23:09.8684 INFO - Inicio del trabajo map/reduce
07/11/2024 15:23:10.4144 INFO - Fin del trabajo map/reduce

CHI: 07/11/2024 15:22:31.0895 INFO - Inicio del trabajo map/reduce
07/11/2024 15:22:31.6711 INFO - Fin del trabajo map/reduce

Sin combiner:

NYC: 07/11/2024 15:33:18.1335 INFO - Inicio del trabajo map/reduce
07/11/2024 15:33:19.0745 INFO - Fin del trabajo map/reduce

CHI: 07/11/2024 15:34:25.3212 INFO - Inicio del trabajo map/reduce
07/11/2024 15:34:25.6802 INFO - Fin del trabajo map/reduce

Query2

Con combiner:

NYC: 07/11/2024 15:24:19.5134 INFO - Inicio del trabajo map/reduce
07/11/2024 15:24:20.2154 INFO - Fin del trabajo map/reduce

CHI: 07/11/2024 15:24:52.3781 INFO - Inicio del trabajo map/reduce
07/11/2024 15:24:52.8701 INFO - Fin del trabajo map/reduce

Sin combiner:

NYC: 07/11/2024 15:34:55.4601 INFO - Inicio del trabajo map/reduce
07/11/2024 15:34:56.1551 INFO - Fin del trabajo map/reduce

CHI: 07/11/2024 15:35:31.7084 INFO - Inicio del trabajo map/reduce
07/11/2024 15:35:32.3444 INFO - Fin del trabajo map/reduce

Query3 (n=2 ; from=01/01/2000 ; to=01/01/2024)

Con combiner:

NYC: 07/11/2024 15:27:05.8322 INFO - Inicio del trabajo map/reduce
07/11/2024 15:27:06.7782 INFO - Fin del trabajo map/reduce

CHI: 07/11/2024 15:26:10.1494 INFO - Inicio del trabajo map/reduce
07/11/2024 15:26:11.1254 INFO - Fin del trabajo map/reduce

Sin combiner:

NYC: 07/11/2024 15:35:59.1897 INFO - Inicio del trabajo map/reduce
07/11/2024 15:36:00.9937 INFO - Fin del trabajo map/reduce

CHI: 07/11/2024 15:36:25.3231 INFO - Inicio del trabajo map/reduce
07/11/2024 15:36:25.8191 INFO - Fin del trabajo map/reduce

Query4 (n=4 ; agency = TRAFFIC)

Con combiner:

NYC: 07/11/2024 15:28:26.2271 INFO - Inicio del trabajo map/reduce
07/11/2024 15:28:26.4371 INFO - Fin del trabajo map/reduce

CHI: 07/11/2024 15:28:57.6064 INFO - Inicio del trabajo map/reduce
07/11/2024 15:28:57.7174 INFO - Fin del trabajo map/reduce

Sin combiner:

NYC: 07/11/2024 15:42:53.6232 INFO - Inicio del trabajo map/reduce
07/11/2024 15:42:55.2445 INFO - Fin del trabajo map/reduce

CHI: 07/11/2024 15:44:01.4124 INFO - Inicio del trabajo map/reduce
07/11/2024 15:44:01.7894 INFO - Fin del trabajo map/reduce