Trabajo Práctico Especial 2: Multas de Estacionamiento

29 de Mayo de 2024

Objetivo

Diseñar e implementar una aplicación de consola que utilice el **modelo de programación MapReduce** junto con el framework **HazelCast** para el procesamiento de **multas de estacionamiento**, basado en datos reales.

Para este trabajo se busca poder procesar datos de multas de estacionamiento de las ciudades de **Nueva York, EEUU** y **Chicago EEUU**.

Los datos son extraídos de los respectivos portales de gobierno en formato CSV.

Descripción Funcional

A continuación se lista el ejemplo de uso que se busca para la aplicación: procesar los datos de multas de estacionamiento de las ciudades de Nueva York y Chicago. Sin embargo, es importante recordar que la mayor parte de la implementación no debe estar atada a la realidad de los ejemplos de uso. **Por ejemplo, las estadísticas serán las que la aplicación obtenga en ejecución a partir del archivo de infracciones y no serán aceptadas implementaciones que tengan fijos estos datos.** En otras palabras, la implementación deberá funcionar también para procesar los datos de multas de cualquier otra ciudad, manteniendo siempre la estructura de los archivos que se presentan a continuación.

Datos de multas de Nueva York, a partir de ahora `ticketsNYC.csv`

- **Origen:** Open Parking and Camera Violations
- **Descarga:** `/afs/it.itba.edu.ar/pub/pod/ticketsNYC.csv`
- **Cantidad de registros: 15.000.000**
- **Campos:**
  - **`Plate`**: Patente (Cadena de caracteres)
  - **`Issue Date`**: Fecha de la multa (Formato YYYY-MM-DD)
  - **`InfractionCode`**: Código de la infracción (Entero)
  - **`Fine Amount`**: Monto (Número)
  - **`County Name`**: Barrio (Cadena de caracteres)
  - **`Issuing Agency`**: Agencia Recaudadora (Cadena de caracteres)

El archivo se compone de una primera línea de encabezado, con los títulos de cada campo. De la segunda línea en adelante, **cada línea representa una multa de estacionamiento** conteniendo los datos de cada uno de los campos, separados por “;”. Por ejemplo:

```
Plate;Issue Date;InfractionCode;Fine Amount;County Name;Issuing Agency
GXH1273;2022-08-25;36;50.0;Kings;DEPARTMENT OF TRANSPORTATION
F35GMJ;2017-07-30;53;115.0;New York City;TRAFFIC
HAN3178;2019-10-15;37;35.0;Queens;TRAFFIC
...
```

Datos de infracciones de Nueva York, a partir de ahora `infractionsNYC.csv`

- **Descarga:** `/afs/it.itba.edu.ar/pub/pod/infractionsNYC.csv`
- **Cantidad de registros:** `97`
- **Campos relevantes:**
  - **`Code`**: Código Identificador (Entero)
  - **`Definition`**: Infracción (Cadena de caracteres)

El archivo se compone de una primera línea de encabezado. De la segunda línea en adelante, **cada línea representa una infracción**. Por ejemplo:

```
CODE;DEFINITION
87;FRAUDULENT USE PARKING PERMIT
67;PEDESTRIAN RAMP
85;STORAGE-3HR COMMERCIAL
...
```

Datos de multas de Chicago, a partir de ahora `ticketsCHI.csv`

- **Origen:** City of Chicago Parking and Camera Ticket Data
- **Descarga:** `/afs/it.itba.edu.ar/pub/pod/ticketsCHI.csv`
- **Cantidad de registros:** `5.000.000`
- **Campos:**
  - **`issue_date`**: Fecha y hora de la multa (Formato YYYY-MM-DD hh:mm:ss)
  - **`license_plate_number`**: Patente (UUID)
  - **`violation_code`**: Código de la infracción (Cadena de caracteres)
  - **`unit_description`**: Agencia Recaudadora (Cadena de caracteres)
  - **`fine_level1_amount`**: Monto (Entero)
  - **`community_area_name`**: Barrio (Cadena de caracteres)

El archivo se compone de una primera línea de encabezado, con los títulos de cada campo. De la segunda línea en adelante, **cada línea representa una multa de estacionamiento** conteniendo los datos de cada uno de los campos, separados por ";". Por ejemplo:

```
issue_date;license_plate_number;violation_code;unit_description;fine_level1_amount;community_area_name
2005-03-10 16:30:00;25515fcf196d56759b5ebdf9242553b7123f5233184cfb38c2a3f45cc33fe5c8;0964080A;CPD;50;LOOP
2004-03-25 09:09:00;3ac2ab0e3e3b5650d244a5144f8504b9a9c9ebefa10f855cdd414733596a96ef;0964190A;DOF;30;LINCOLN PARK
2004-03-25 09:09:00;8df0227fb8b46ab982e659377ec530f13f0e1df780577b9ea920319e0fb201c3;0964190B;DOF;50;NEAR NORTH SIDE
...
```

Datos de infraciones de Chicago, a partir de ahora `infractionsCHI.csv`

- **Descarga: `/afs/it.itba.edu.ar/pub/pod/infractionsCHI.csv`**
- **Cantidad de registros: `128`**
- **Campos:**
  - **`violation_code`**: Código Identificador (Cadena de caracteres)
  - **`violation_description`**: Infracción (Cadena de caracteres)

El archivo se compone de una primera línea de encabezado. De la segunda línea en adelante, **cada línea representa una infracción**. Por ejemplo:

```
violation_code;violation_description
0968040J;MOTOR RUNNING IN WRIGLEY BUS PERMIT ZONE
0964160C;NO DISPLAY OF BACK-IN PERMIT
0964070;SNOW ROUTE: 2'' OF SNOW OR MORE
...
```

Se asume que el formato y contenido de los archivos es correcto

Requerimientos

La aplicación debe poder resolver un conjunto de consultas listadas más abajo. En cada una de ellas se indicará un ejemplo de invocación con un script propio para correr únicamente esa query con sus parámetros necesarios.

Cada corrida de la aplicación resuelve sólo una de las queries sobre los datos obtenidos a partir de los archivos CSV provistos en esa invocación (archivos CSV de infracciones y multas).

La respuesta a la *query* quedará en un archivo de salida CSV.

Para medir performance, se deberán escribir en otro archivo de salida los *timestamp* de los siguientes momentos:

- Inicio de la lectura de los archivos de entrada
- Fin de lectura de los archivos de entrada
- Inicio de un trabajo MapReduce
- Fin de un trabajo MapReduce (incluye la escritura del archivo de respuesta)

Todos estos momentos deben ser escritos en la salida luego de la respuesta con el timestamp en formato: `dd/mm/yyyy hh:mm:ss:xxxx` y deben ser claramente identificables.

Ejemplo del archivo de tiempos:

```
29/05/2024 14:43:09:0223 INFO  [main] Client (Client.java:76) - Inicio de la
lectura del archivo
29/05/2024 14:43:23:0011 INFO  [main] Client (Client.java:173) - Fin de lectura
del archivo
29/05/2024 14:43:23:0013 INFO  [main] Client (Client.java:87) - Inicio del
trabajo map/reduce
29/05/2024 14:43:23:0490 INFO  [main] Client (Client.java:166) - Fin del
trabajo map/reduce
```

Por ejemplo:

```
$> sh queryX.sh -Daddresses='xx.xx.xx.xx:XXXX;yy.yy.yy.yy:YYYY' -Dcity=ABC
-DinPath=XX -DoutPath=YY [params]
```

donde:

- `queryX.sh` es el script que corre la query `X`.
- `-Daddresses` refiere a las direcciones IP de los nodos con sus puertos (una o más, separadas por punto y coma)
- `-Dcity` indica con qué dataset de ciudad se desea trabajar. Los únicos valores posibles son **`NYC`** y **`CHI`**.
- `-DinPath` indica el path donde están los archivos de entrada de multas e infracciones
- `-DoutPath` indica el path donde estarán ambos archivos de salida **`query1.csv`** y **`time1.txt`**.
- `[params]` son los parámetros extras que corresponden para algunas queries.

De esta forma,

```
$> sh query1.sh -Daddresses='10.6.0.1:5701;10.6.0.2:5701' -Dcity=NYC
-DinPath=/afs/it.itba.edu.ar/pub/pod/
-DoutPath=/afs/it.itba.edu.ar/pub/pod-write/
```

resuelve la *query* 1 a partir de los datos presentes en `/afs/it.itba.edu.ar/pub/pod/infractionsNYC.csv` y `/afs/it.itba.edu.ar/pub/pod/ticketsNYC.csv` utilizando los nodos `10.6.0.1` y `10.6.0.2` para su procesamiento. Se crearán los archivos `/afs/it.itba.edu.ar/pub/pod-write/query1.csv` y `/afs/it.itba.edu.ar/pub/pod-write/time1.txt` que contendrán respectivamente el resultado de la *query* y los *timestamp* de inicio y fin de la lectura del archivo y de los trabajos map/reduce.

De invocarse con `-Dcity=CHI` utilizará los datos presentes en `infractionsCHI.csv` y `ticketsCHI.csv`.

Query 1: Total de multas por infracción

Donde cada línea de la salida contenga, separados por ";" **la infracción** y **la cantidad total de multas con esa infracción**.

El orden de impresión es **descendente por cantidad total de multas** y desempata **alfabético** por la infracción.

Sólo se deben listar las infracciones presentes en el archivo CSV de infracciones.

No se deben listar las infracciones con una cantidad total de 0 multas.

- Parámetros adicionales: Ninguno
- Ejemplo de invocación: `sh query1.sh -Daddresses='10.6.0.1:5701' -Dcity=NYC -DinPath=. -DoutPath=.`

- Salida de ejemplo para NYC:

```
Infraction;Tickets
PHTO SCHOOL ZN SPEED VIOLATION;25698186
NO PARKING-STREET CLEANING;11521009
FAIL TO DSPLY MUNI METER RECPT;8269682
NO STANDING-DAY/TIME LIMITS;6839484
...
```

- Salida de ejemplo para CHI:

```
Infraction;Tickets
EXPIRED PLATES OR TEMPORARY REGISTRATION;5817742
EXP. METER NON-CENTRAL BUSINESS DISTRICT;4180475
STREET CLEANING OR SPECIAL EVENT;4180475
PARKING/STANDING PROHIBITED ANYTIME;3751839
...
```

Query 2: Top 3 infracciones más populares de cada barrio

Donde cada línea de la salida contenga, separados por ";" el **nombre del barrio**, **la infracción con más multas** de ese barrio, **la segunda infracción con más multas** en ese barrio y **la tercera infracción con más multas** en ese barrio**.**

El orden de impresión es **alfabético por barrio**.

Sólo se deben listar las infracciones presentes en el archivo CSV de infracciones.

En el caso de que en un barrio no existan hasta tres infracciones distintas, indicar un "-" en la columna `InfractionTopX` correspondiente.

- Parámetros adicionales: Ninguno

- Ejemplo de invocación: `sh query2.sh -Daddresses='10.6.0.1:5701' -Dcity=NYC -DinPath=. -DoutPath=.`

- Salida de ejemplo para NYC:

```
County;InfractionTop1;InfractionTop2;InfractionTop3
Bronx;PHTO SCHOOL ZN SPEED VIOLATION;NO PARKING-STREET CLEANING;FAIL TO DSPLY MUNI METER RECPT
Kings;PHTO SCHOOL ZN SPEED VIOLATION;NO PARKING-STREET CLEANING;FAIL TO DSPLY MUNI METER RECPT
New York City;NO STANDING-DAY/TIME LIMITS;NO PARKING-DAY/TIME LIMITS;FAIL TO DSPLY MUNI METER RECPT
...
```

- Salida de ejemplo para CHI:

```
County;InfractionTop1;InfractionTop2;InfractionTop3
ALBANY PARK;STREET CLEANING OR SPECIAL EVENT;EXP. METER NON-CENTRAL BUSINESS DISTRICT;RESIDENTIAL PERMIT PARKING
ARCHER HEIGHTS;RESIDENTIAL PERMIT PARKING;EXPIRED PLATES OR TEMPORARY REGISTRATION;STREET CLEANING OR SPECIAL EVENT
ARMOUR SQUARE;EXP. METER NON-CENTRAL BUSINESS DISTRICT;EXPIRED PLATES OR TEMPORARY REGISTRATION;STREET CLEANING OR SPECIAL EVENT
...
```

Query 3: Top N agencias con mayor porcentaje de recaudación

Donde cada línea de la salida contenga, separados por ";" el **nombre de la agencia que labró la infracción**, y **el porcentaje de recaudación** de la misma.

El porcentaje de recaudación de una agencia se obtiene a partir de la suma de los montos de todas las infracciones labradas por la agencia dividido la suma de los montos de todas las infracciones labradas por todas las agencias**.**

El orden de impresión es **descendente por porcentaje** y **desempata alfabético** por el nombre de la agencia que labró la infracción.

Sólo se deben listar las primeras `n` agencias, donde `n` es un parámetro adicional.

Los valores de los porcentajes se deben truncar a dos decimales.

- Parámetros adicionales: `n` (Entero)
- Ejemplo de invocación: `sh query3.sh -Daddresses='10.6.0.1:5701' -Dcity=NYC -DinPath=. -DoutPath=. -Dn=4`
- Salida de ejemplo para NYC:

```
Issuing Agency;Percentage
TRAFFIC;69.19%
DEPARTMENT OF TRANSPORTATION;25.16%
POLICE DEPARTMENT;3.97%
...
```

- Salida de ejemplo para CHI:

```
Issuing Agency;Percentage
CPD;50.46%
DOF;39.35%
Miscellaneous;5.22%
...
```

Query 4: Patente con más infracciones de cada barrio en el rango `[from, to]`

Donde cada línea de la salida contenga, separados por ";" el **nombre del barrio**, la **patente con más multas** de ese barrio y la **cantidad de multas** correspondiente**.**

El orden de impresión es **alfabético por barrio**.

Sólo se deben listar las multas labradas en el rango `[from, to]`, donde `from` y `to` son dos parámetros adicionales.

- Parámetros adicionales: `from` (String DD/MM/YY), `to` (String DD/MM/YY)
- Ejemplo de invocación: `sh query4.sh -Daddresses='10.6.0.1:5701' -Dcity=NYC -DinPath=. -DoutPath=. -Dfrom='01/01/2017' -Dto='31/12/2017'`

- Salida de ejemplo para NYC:

```
County;Plate;Tickets
Bronx;94903JA;573
Kings;RHC3492;533
New York City;2121796;1024
…
```

- Salida de ejemplo para CHI:

```
County;Plate;Tickets
ALBANY PARK;c94d83c96b6835504fd7f59088b12a1953730dd882eac0dc089b1d1d250abf1b;490
ARCHER HEIGHTS;c94d83c96b6835504fd7f59088b12a1953730dd882eac0dc089b1d1d250abf1b;
734
ARMOUR SQUARE;c94d83c96b6835504fd7f59088b12a1953730dd882eac0dc089b1d1d250abf1b;96
…
```

Query 5: Pares de infracciones que tienen, en grupos de a cientos, el mismo promedio de monto de multa

Donde cada línea de la salida contenga separados por ";" **el grupo de cientos de promedio de monto de multa**, **el nombre de la infracción A** que corresponde al grupo y **el nombre de la infracción B** que corresponde al al grupo.

El promedio de monto de multa de una infracción se obtiene a partir de la suma de los montos de todas las multas con esa infracción dividido la cantidad de multas con esa infracción.

Si por ejemplo la infracción IN1 tiene un promedio de $230, la infracción IN2 tiene un promedio de $270, la infracción IN3 tiene un promedio de $290 y la infracción IN4 tiene un promedio de $400 entonces las infracciones IN1, IN2 e IN3 formarán pares porque sus promedios, en cientos, coinciden. Entonces se formarán los pares (IN1, IN2), (IN1, IN3) y (IN2, IN3) que pertenecen al grupo de $200 donde estarán los promedios entre $200 y $299 ambos inclusive.

El orden de impresión es **descendente por grupo** y el orden de los pares dentro de cada grupo es **alfabético por infracción**.

Sólo se deben considerar las infracciones presentes en el archivo CSV de infracciones.

El último grupo posible a mostrar es el de $100 (incluyendo aquellos promedios entre $100 inclusive y $199 inclusive), es decir, no listar los pares de infracciones que tengan un promedio hasta $99 inclusive.

No se debe listar el par opuesto (es decir si se lista `Group;Infraction A;Infraction B` no debe aparecer la tupla `Group;Infraction B;Infraction A`).

Si un grupo está compuesto por una única infracción, el par no puede armarse por lo que no se lista.

- Parámetros adicionales: Ninguno

- Ejemplo de invocación: `sh query5.sh -Daddresses='10.6.0.1:5701' -Dcity=NYC -DinPath=. -DoutPath=.`

- Salida de ejemplo para NYC:

```
Group;Infraction A;Infraction B
500;BLUE ZONE;FAILURE TO DISPLAY BUS PERMIT
...
100;ANGLE PARKING-COMM VEHICLE;BIKE LANE
100;ANGLE PARKING-COMM VEHICLE;BUS PARKING IN LOWER MANHATTAN
...
```

- Salida de ejemplo para CHI:

```
Group;Infraction A;Infraction B
200;EXCESSIVE DIESEL POWERED VEHICLE ENGINE RUNNING;DISABLED PARKING ZONE
200;EXCESSIVE DIESEL POWERED VEHICLE ENGINE RUNNING;INVALID PLACARD
200;EXCESSIVE DIESEL POWERED VEHICLE ENGINE RUNNING;NO CITY STICKER VEHICLE
UNDER/EQUAL TO 16,000 LBS.
200;DISABLED PARKING ZONE;INVALID PLACARD
200;DISABLED PARKING ZONE;NO CITY STICKER VEHICLE UNDER/EQUAL TO 16,000 LBS.
200;INVALID PLACARD;NO CITY STICKER VEHICLE UNDER/EQUAL TO 16,000 LBS.
...
```

Muy Importante:

- ➔ **Respetar exactamente los nombres de los *scripts*, los nombres de los archivos de entrada y salida y el orden y formato de los parámetros del *scripts*.**
- ➔ En todos los `pom.xml` que entreguen deberán definir el `artifactId` de acuerdo a la siguiente convención: "`tpe2-gX-Z`" donde X es el número de grupo y Z es `parent`, `api`, `server` o `client`. Por ejemplo: **`<artifactId>`**`tpe2-g5-api`**`</artifactId>`**
- ➔ En todos los `pom.xml` que entreguen deberán incluir el tag `name` con la siguiente convención: "`tpe2-gX-Z`" donde X es el número de grupo y Z es `parent`, `api`, `server` o `client`. Por ejemplo: **`<name>`**`tpe2-g5-api`**`</name>`**
- ➔ Utilizar la versión **3.8.6** de **`hazelcast-all`**
- ➔ **El nombre del cluster (`<group><name>`) y los nombres de las colecciones de Hazelcast** a utilizar en la implementación deben comenzar con "g" seguido del número de grupo. Por ej `g5` para así evitar conflictos con las colecciones y poder hacer pruebas de distintos alumnos en simultáneo utilizando la misma red.
- ➔ La implementación debe **respetar exactamente el formato de salida enunciado**. Tener en cuenta que los archivos de salida deben contener las líneas de encabezado correspondientes indicadas en las salidas de ejemplo para todas las *queries*.

Condiciones del trabajo práctico

- **El trabajo práctico debe realizarse en los mismos grupos formados para el primer trabajo práctico especial.**
- Cada una de las opciones debe ser implementada **con uno o más *job*s MapReduce** que pueda correr en un ambiente distribuido utilizando un *grid* de Hazelcast.
- Los componentes del *job*, clases del modelo, tests y el diseño de cada elemento del proyecto queda a criterio del alumno, pero debe estar enfocado en:
    - Que funcione correctamente en un ambiente concurrente MapReduce en Hazelcast.
    - Que sea eficiente para un gran volumen de datos, particularmente en tráfico de red.
    - Mantener buenas prácticas de código como comentarios, reutilización, legibilidad y mantenibilidad.

Material a entregar

Cada alumno deberá subir al **Campus ITBA** un archivo compactado conteniendo:

- El **código fuente** de la aplicación:
    - Utilizando el arquetipo de Maven utilizado en las clases.
    - Con una correcta separación de las clases en los módulos *api*, *client* y *server*.
    - Un README indicando cómo preparar el entorno a partir del código fuente para ejecutar la aplicación en un ambiente con varios nodos.
    - No se deben entregar los binarios.
- Un **documento breve** explicando:
    - 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).
- La **historia de Git**: El directorio oculto .git/ donde se detallan todas las modificaciones realizadas.

Corrección

El trabajo no se considerará aprobado si:

- No se entregó el trabajo práctico en tiempo y forma.
- Faltan algunos de los materiales solicitados en la sección anterior.
- El código no compila utilizando Maven en consola (de acuerdo a lo especificado en el README a entregar).
- El servicio no inicia cuando se siguen los pasos del README.
- Los clientes no corren al seguir los pasos del README.

Si nada de esto se cumple, se procederá a la corrección donde se tomará en cuenta:

- Que los procesos y queries funcionen correctamente según las especificaciones dadas.
- El resultado de las pruebas y lo discutido en el coloquio.
- La aplicación de los temas vistos en clase: Concurrencia y Hazelcast.
- La modularización, diseño testeo y reutilización de código.
- El contenido y desarrollo del informe.

Uso de Git

Es obligatorio el uso de un repositorio Git para la resolución de este TPE. No se aceptarán entregas que utilicen un repositorio git con un único *commit* que consista en la totalidad del código a entregar.

Los distintos *commits* deben permitir ver la evolución individual del trabajo.

Muy importante: **los repositorios creados deben ser privados, solo visibles para los autores y la cátedra en caso de que se lo solicite específicamente.**

Cronograma

- **Presentación del Enunciado: miércoles 29/05**
- **Entrega del trabajo: Estará disponible hasta el jueves 13/06 a las 23:59** la actividad "Entrega TPE 2" localizada en la sección Contenido / Evaluación / Entrega TPE 2. En la misma deberán cargar el paquete con el **código fuente** y el **documento**
- **El día miércoles 26/06 a las 18:00** cada grupo tendrá un espacio para un coloquio. Durante el mismo se les hará una devolución del trabajo, indicando la corrección y los principales errores cometidos. A criterio de la cátedra también se podrán realizar preguntas sobre la implementación. Opcionalmente se les podrá solicitar la ejecución de la aplicación a la cátedra para revisar el funcionamiento de alguna funcionalidad. Para ello es necesario que un alumno del grupo tenga el cluster “levantado”.
- **El día del recuperatorio será el miércoles 03/07.**
- **No se aceptarán entregas pasado el día y horario establecido como límite.**

Dudas sobre el TPE

Las mismas deben volcarse en los **Debates** del Campus ITBA.

Recomendaciones

- **Clases útiles para consultar**
  - **Hazelcast**
    - com.hazelcast.mapreduce.**Combiner**
    - com.hazelcast.mapreduce.**Collator**
  - **Java**
    - java.nio.file.**Files.lines**
    - java.text.**DecimalFormat** y java.math.**RoundingMode**
    - java.time.format.**DateTimeFormatter**