Trabajo Práctico Especial 2: Peatones

12 de Octubre de 2022

Objetivo

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

Para este trabajo se busca poder procesar datos de peatones de la ciudad de Melbourne, Australia.

Los datos son extraídos del portal 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 peatones de la ciudad de Melbourne. 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 mediciones serán las que la aplicación obtenga en ejecución a partir del archivo de sensores y no serán aceptadas implementaciones que tengan fijos estos datos.** En otras palabras, la implementación deberá funcionar también para procesar los datos de peatones de cualquier otra ciudad, manteniendo siempre la estructura de los archivos que se presentan a continuación.

Datos de mediciones de Melbourne, a partir de ahora `readings.csv`

- **Origen:** http://www.pedestrian.melbourne.vic.gov.au/
- **Descarga:** `/afs/it.itba.edu.ar/pub/pod/2022/readings.csv`
- **Campos Relevantes:**
  - **`Sensor_ID`:** El identificador del sensor que registró la medición (número entero)
  - **`Year`:** Año de la medición (número entero)
  - **`Month`**: Mes del año de la medición (cadena de caracteres, por ejemplo “October”)
  - **`Mdate`**: Día del mes de la medición (número entero)
  - **`Day`**: Día de la semana de la medición (cadena de caracteres, por ejemplo Wednesday)
  - **`Time`**: Hora de la medición (número entero)
  - **`Hourly_Counts`:** Cantidad total de peatones que registró el sensor en la medición (número entero)

  En el archivo se incluyen además otros campos que pueden ignorarse.

  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 medición conteniendo los datos de cada uno de los campos, separados por “;”.

```
id;Date_Time;Year;Month;Mdate;Day;Time;Sensor_ID;Sensor_Name;Hourly_Counts
2249087;May 29, 2018 03:00:00 PM;2018;May;29;Tuesday;15;51;QVM-Franklin St (North);172
2535774;January 22, 2019 05:00:00 PM;2019;January;22;Tuesday;17;53;Collins St (North);1550
...
```

Datos de sensores de Melbourne, a partir de ahora `sensors.csv`

- **Descarga:** `/afs/it.itba.edu.ar/pub/pod/2022/sensors.csv`
- **Campos:**
  - **`sensor_id`:** El identificador del sensor que registró la medición (número entero)
  - **`sensor_description`:** Nombre del sensor, como el nombre de la calle donde está ubicado (cadena de caracteres)
  - **`status`:** Estado del Sensor. Puede tomar los valores
    - **‘A’: Activo**
    - **‘R’: Removido**
    - **‘I’: Inactivo**

  En el archivo se incluyen además otros campos que pueden ignorarse.

  El archivo se compone de una primera línea de encabezado. De la segunda línea en adelante, cada línea representa un sensor.

```
     sensor_id;sensor_description;sensor_name;installation_date;status;note;direction_1;direction_2;latitude;longitude;location
     45;Little Collins St-Swanston St (East);Swa148_T;2017/06/29;A;;South;North;-37.81414074;144.9660938;(-37.81414074, 144.9660938)
     70;Errol Street (East);Errol20_T;2020/10/12;A;;South;North;-37.80456984;144.94946229;(-37.80456984, 144.94946229)
     ...
```

> ***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 sensores y de mediciones).

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 del archivo de entrada
- Fin de lectura del archivo 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:

```
12/10/2022 14:43:09:0223 INFO  [main] Client (Client.java:76) - Inicio de la lectura del archivo
12/10/2022 14:43:23:0011 INFO  [main] Client (Client.java:173) - Fin de lectura del archivo
12/10/2022 14:43:23:0013 INFO  [main] Client (Client.java:87) - Inicio del trabajo map/reduce
12/10/2022 14:43:23:0490 INFO  [main] Client (Client.java:166) - Fin del trabajo map/reduce
```

Por ejemplo:

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

donde

- `queryX` 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)
- `-DinPath` indica el path donde están los archivos de entrada **`sensors.csv`** y **`readings.csv`**.
- `-DoutPath` indica el path donde estarán ambos archivos de salida **`query1.csv`** y **`time1.txt`**.
- `[params]`: los parámetros extras que corresponden para algunas queries.

De esta forma,

```
$> ./query1 -Daddresses='10.6.0.1:5701;10.6.0.2:5701'
-DinPath=/afs/it.itba.edu.ar/pub/pod/2022/
-DoutPath=/afs/it.itba.edu.ar/pub/pod-write/g7/
```

resuelve la *query* 1 a partir de los datos presentes en `/afs/it.itba.edu.ar/pub/pod/2022/sensors.csv` y `/afs/it.itba.edu.ar/pub/pod/2022/readings.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/g7/query1.csv` y `/afs/it.itba.edu.ar/pub/pod-write/g7/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.

Query 1: Total de Peatones por Sensor

Donde cada línea de la salida contenga, separados por ";" el **nombre del sensor** y la **cantidad total de peatones** registrados por el sensor.

Sólo se deben listar los sensores presentes en el archivo CSV de sensores que tengan el estado Activo.

El orden de impresión es descendente por el total de peatones registrados y luego alfabético por nombre del sensor.

- ❏ Parámetros adicionales: Ninguno
- ❏ Ejemplo de invocación: `./query1 -Daddresses='10.6.0.1:5701' -DinPath=. -DoutPath=.`
- ❏ Salida de ejemplo:

```
Sensor;Total_Count
Town Hall (West);185243492
Flinders Street Station Underpass;163910450
Bourke Street Mall (South);109987361
Melbourne Central;109987361
Princes Bridge;91140173
...
```

Query 2: Total de Peatones por Momento de la Semana para cada año

Donde cada línea de la salida contenga, separados por ";" el **año**, la **cantidad total de peatones registrados** por todos los sensores **durante los *weekdays*** (lunes a viernes) de ese año, **la cantidad total de peatones registrados** por todos los sensores **durante los *weekends*** (sábado y domingo) de ese año y **la suma de ambos valores**.

El orden de impresión es descendente por año.

- Parámetros adicionales: Ninguno
- Ejemplo de invocación: `./query2 -Daddresses='10.6.0.1:5701' -DinPath=. -DoutPath=.`
- Salida de ejemplo:

```
Year;Weekdays_Count;Weekends_Count;Total_Count
2022;88657325;42157989;130815314
2021;97641853;42685412;140327265
...
```

Query 3: La medición más alta de cada sensor (mayores a `min` peatones)

Donde cada línea de la salida contenga, separados por ";" el **nombre del sensor**, la **cantidad de peatones de la medición más alta** de ese sensor y la **fecha y hora de esa medición**.

Sólo se deben listar los sensores presentes en el archivo CSV de sensores que tengan el estado Activo cuya medición más alta tenga una cantidad de peatones mayor al parámetro `min`.

En caso de existir para un mismo sensor dos o más mediciones con el mismo valor más alto, elegir la de la medición más reciente.

El orden de impresión es descendente por cantidad de peatones y luego alfabético por nombre del sensor.

La fecha y hora de la medición más alta debe imprimirse con el formato dd/MM/yyyy HH:00.

- Parámetros adicionales: `min` (Un valor entero mayor a cero)
- Ejemplo de invocación: `./query3 -Daddresses='10.6.0.1:5701' -DinPath=. -DoutPath=. **-Dmin=10000**`
- Salida de ejemplo:

```
Sensor;Max_Reading_Count;Max_Reading_DateTime
New Quay;15782;14/11/2019 11:00
Birrarung Marr;10389;12/11/2019 12:00
Bourke St Bridge;10389;16/11/2019 18:00
...
```

Query 4: Top `n` sensores con mayor promedio mensual de peatones para un año `year`

Donde cada línea de la salida contenga, separados por ";" el **nombre del sensor**, el **mayor promedio mensual de peatones de ese sensor para el año** `year` y **el mes de ese promedio**.

El promedio mensual de peatones de un sensor se calcula como la cantidad total de peatones registrados en las mediciones de un mes dividido la cantidad de días del mes.

El mayor promedio mensual de peatones de un sensor es el promedio más alto de los doce promedios posibles para el sensor.

El orden de impresión es descendente por el promedio mensual de peatones y luego alfabético por nombre del sensor.

Sólo se deben listar los primeros `n` sensores del resultado que estén presentes en el archivo CSV de sensores que tengan el estado Activo.

Los promedios se deben truncar en dos decimales.

- Parámetros adicionales: `n, year` (Ambos valores enteros mayores a cero)
- Ejemplo de invocación: `./query4 -Daddresses='10.6.0.1:5701' -DinPath=. -DoutPath=. **-Dn=3 -Dyear=2021**`
- Salida de ejemplo:

```
Sensor;Month,Max_Monthly_Avg
Flinders La-Swanston St (West);April;1628.49
Southbank;October;1465.41
Bourke St Bridge;March;1440.00
```

Query 5: Pares de sensores que registran la misma cantidad de millones de peatones

Donde cada línea de la salida contenga separados por **;** **el grupo de millones de peatones**, **el nombre del sensor A** que corresponde al grupo de peatones y **el nombre del sensor B** que corresponde al grupo de peatones.

El último grupo posible a mostrar es el de 1.000.000 (incluyendo aquellos pares de sensores que registren entre 1.000.000 de peatones inclusive y 1.999.999 peatones inclusive), es decir, no listar los pares de sensores que registren hasta 999.999 peatones inclusive.

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

Si un grupo está compuesto por un único sensor, el par no puede armarse por lo que no se lista.

El orden de impresión es descendente por grupo y el orden de los pares dentro de cada grupo es alfabético por nombre del sensor.

Sólo se deben listar los sensores presentes en el archivo CSV de sensores que tengan el estado Activo.

- Parámetros adicionales: Ninguno

- Ejemplo de invocación: `./query5 -Daddresses='10.6.0.1:5701' -DinPath=. -DoutPath=.`

- Salida de ejemplo:

```
Group;Sensor A;Sensor B
12.000.000;Flinders Street Station Underpass;Melbourne Central
...
7.000.000;Flinders St-Elizabeth St (East);Spencer St-Collins St (North)
7.000.000;Flinders St-Elizabeth St (East);State Library
...
```

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-g7-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-g7-api`**`</name>`**

➔ Utilizar la versión **3.7.8** 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 del grupo. Por ej `g7` para así evitar conflictos con las colecciones y poder hacer pruebas de distintos grupos en simultáneo.

➔ 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 equipo, 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 grupo 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** (no más de dos carillas) 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.
    - Además el grupo deberá elegir una de las queries donde crean que el uso de *Combiner* puede mejorar los tiempos de la misma, implementarlo y comparar los tiempos de esa query ejecutándose con y sin *Combiner*.
- 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: Java 8, 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. Deberán crear un repositorio donde todos los integrantes del grupo colaboren con la implementación. 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 del trabajo, tanto grupal como individual.

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

Cronograma

- **Presentación del Enunciado: miércoles 12/10**
- **Entrega del trabajo: Estará disponible hasta el jueves 27/10 a las 23:59** la actividad "Entrega TPE 2" localizada en la sección Contenido / Evaluación / TPE 2: Peatones. En la misma deberán cargar el paquete con el **código fuente** y el **documento**
- **El día miércoles 09/11 a las 18:00** cada grupo tendrá un espacio de 10 minutos para un coloquio. Durante el mismo se les hará una devolución del trabajo, indicando la nota 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 integrante del equipo tenga el cluster “levantado”.
- **El día del recuperatorio será el miércoles 16/11.**
- **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`**
    - `java.text.`**`DecimalFormat`**
    - `java.math.`**`RoundingMode`**