72.42 Programación de Objetos Distribuidos

Trabajo Práctico Especial 2: Multas de Estacionamiento

23 de Octubre 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 de los archivos de infracciones y de agencias 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)
  - **`Infraction ID`**: Identificador de la infracción (Entero)
  - **`Fine Amount`**: Monto (Número)
  - **`Issuing Agency`**: Agencia (Cadena de caracteres)
  - **`Issue Date`**: Fecha de la multa (Formato YYYY-MM-DD)
  - **`County 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:

```
Plate;Infraction ID;Fine Amount;Issuing Agency;Issue Date;County Name
60876NC;31;115.0;TRAFFIC;2022-03-24;New York City
GND7783;36;50.0;DEPARTMENT OF TRANSPORTATION;2023-01-02;Queens
28620MK;71;65.0;TRAFFIC;2022-09-28;Kings
...
```

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:

```
Infraction ID;Definition
87;FRAUDULENT USE PARKING PERMIT
69;FAIL TO DISP. MUNI METER RECPT
17;NO STANDING-EXC. AUTH. VEHICLE
...
```

Datos de agencias de Nueva York, a partir de ahora `agenciesNYC.csv`

- **Descarga:** `/afs/it.itba.edu.ar/pub/pod/agenciesNYC.csv`
- **Cantidad de registros: 32**
- **Campos relevantes:**
  - **`Issuing Agency`**: Agencia (Cadena de caracteres)

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

```
Issuing Agency
PARKS DEPARTMENT
LONG ISLAND RAILROAD
PARKING CONTROL UNIT
...
```

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

- **Origen:** City of Chicago Parking and Camera Ticket Data (ProPublica)
- **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)
  - **`community_area_name`**: Barrio (Cadena de caracteres)
  - **`unit_description`**: Agencia (Cadena de caracteres)
  - **`license_plate_number`**: Patente (UUID)
  - **`violation_code`**: Código de la infracción (Cadena de caracteres)
  - **`fine_amount`**: Monto (Entero)

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;community_area_name;unit_description;license_plate_number;violation_code;fine_amount
2013-12-02 17:40:00;AUSTIN;CPD;868f87916a821c5597d013570327ce9941c7e49648db9190babccc2a03aaa8a0;0976160F;60
2001-10-01 09:47:00;BELMONT CRAGIN;DOF;ebd792a8f58ccf144834059385ba328151cd23f0e266a734b5d456a9fe949093;0964040B;50
2011-06-14 12:02:00;NEAR WEST SIDE;DOF;a8e45edd567a502c593c153923282bca96e7a5dbb88e5867631eaff3032beca2;0964190;50
...
```

Datos de infracciones 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
0964080B;NO STANDING/PARKING TIME RESTRICTED
```

```
0976210B;WINDOWS MISSING OR CRACKED BEYOND 6
0976220B;SMOKED/TINTED WINDOWS PARKED/STANDING
...
```

Datos de agencias de Chicago, a partir de ahora **`agenciesCHI.csv`**

- **Descarga: `/afs/it.itba.edu.ar/pub/pod/agenciesCHI.csv`**
- **Cantidad de registros: `6`**
- **Campos:**
    - **`agency_name`**: Agencia Emisora (Cadena de caracteres)

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

```
agency_name
DOF
CPD
CPD-Airport
...
```

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 multas, infracciones y agencias).

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:

```
23/10/2024 14:43:09:0223 INFO  [main] Client (Client.java:76) - Inicio de la
lectura del archivo
23/10/2024 14:43:23:0011 INFO  [main] Client (Client.java:173) - Fin de lectura
del archivo
23/10/2024 14:43:23:0013 INFO  [main] Client (Client.java:87) - Inicio del
trabajo map/reduce
23/10/2024 14:43:23:0490 INFO  [main] Client (Client.java:166) - Fin del
trabajo map/reduce
```

Por ejemplo:

72.42 Programación de Objetos Distribuidos

```
$> 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, infracciones y agencias
- `-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/ticketsNYC.csv`, `/afs/it.itba.edu.ar/pub/pod/infractionsNYC.csv` y `/afs/it.itba.edu.ar/pub/pod/agenciesNYC.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 `ticketsCHI.csv`, `infractionsCHI.csv` y `agenciesCHI.csv`

Query 1: Total de multas por infracción y agencia

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

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

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

No se deben listar las infracciones y/o agencias 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;Agency;Tickets
PHTO SCHOOL ZN SPEED VIOLATION;DEPARTMENT OF TRANSPORTATION;60449
FAIL TO DSPLY MUNI METER RECPT;TRAFFIC;40629
NO PARKING-STREET CLEANING;DEPARTMENT OF SANITATION;40629
NO PARKING-STREET CLEANING;TRAFFIC;40629
FAILURE TO STOP AT RED LIGHT;DEPARTMENT OF TRANSPORTATION;30739
...
```

- Salida de ejemplo para CHI:

```
Infraction;Agency;Tickets
EXP. METER NON-CENTRAL BUSINESS DISTRICT;DOF;15977
PARKING/STANDING PROHIBITED ANYTIME;CPD;10481
RUSH HOUR PARKING;CPD;9982
EXPIRED METER CENTRAL BUSINESS DISTRICT;CPD;7384
EXPIRED PLATES OR TEMPORARY REGISTRATION;CPD;5933
...
```

Query 2: Recaudación YTD por agencia

Donde cada línea de la salida contenga, separados por ";" la **agencia**, el **año**, el **mes** de ese año, y el **total de recaudación YTD** (Year To Date) **para esa agencia**.

La **recaudación de un mes de una agencia** consiste en la suma de los montos de todas las multas de ese mes emitidas por esa agencia.

El total de recaudación YTD de un mes de una agencia consiste en la suma de los totales desde el primer mes del año hasta ese mes inclusive. Por ejemplo, el total YTD de Enero 2024 sólo contendrá la recaudación de las multas de Enero 2024. El YTD de Febrero 2024 contendrá la recaudación de las multas de Enero 2024 y de Febrero 2024. El YTD de Diciembre 2024 contendrá la recaudación de las multas de todos los meses de 2024.

El orden de impresión es **alfabético por agencia** y desempata **cronológico por año y mes**.

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

No se deben listar los meses y/o años que no afectan al YTD de una agencia (la recaudación fue 0).

72.42 Programación de Objetos Distribuidos

- Parámetros adicionales: Ninguno

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

- Salida de ejemplo para NYC:

```
Agency;Year;Month;YTD
POLICE DEPARTMENT;2022;11;11730
POLICE DEPARTMENT;2022;12;12400
POLICE DEPARTMENT;2023;1;65
POLICE DEPARTMENT;2023;2;255
...
TRAFFIC;2022;1;18470
TRAFFIC;2022;2;31365
...
```

- Salida de ejemplo para CHI:

```
Agency;Year;Month;YTD
...
CPD;2018;3;32465
CPD;2018;4;83705
CPD;2018;5;91700
DOF;2005;1;77095
DOF;2005;2;128005
DOF;2005;3;241825
...
```

Query 3: Porcentaje de patentes reincidentes por barrio en el rango `[from, to]`

Donde cada línea de la salida contenga, separados por ";" el **barrio** y el **porcentaje de patentes reincidentes para ese barrio**.

Se considera que **una patente es reincidente en un barrio** si cuenta con **al menos `n` multas de una misma infracción en ese barrio registradas en el rango `[from, to]`**, donde `n`, `from` y `to` son tres parámetros adicionales.

El porcentaje de patentes reincidentes de un barrio se obtiene a partir del total de patentes reincidentes (únicas) de ese barrio y el total de patentes (únicas) de ese barrio (patentes que tienen al menos una multa en ese barrio en el rango indicado).

El orden de impresión es **descendente por porcentaje** y **desempata alfabético** por **barrio**.

Los valores de los porcentajes se deben truncar (no redondear) a dos decimales.

- Parámetros adicionales: `n` (Entero mayor o igual a 2), `from` (String DD/MM/YY), `to` (String DD/MM/YY)

- Ejemplo de invocación: `sh query3.sh -Daddresses='10.6.0.1:5701' -Dcity=NYC -DinPath=. -DoutPath=. -Dn=2 -Dfrom=01/01/2021 -Dto=31/12/2021`

- Salida de ejemplo para NYC:

```
County;Percentage
Bronx;25.19%
Kings;3.16%
New York City;68.97%
...
```

- Salida de ejemplo para CHI:

```
County;Percentage
ALBANY PARK;50.46%
ARCHER HEIGHTS;39.35%
ARMOUR SQUARE;55.22%
...
```

Query 4: Top N infracciones con mayor diferencia entre máximos y mínimos montos para una agencia

Donde cada línea de la salida contenga, separados por ";" la **infracción**, el **monto mínimo** de las multas con esa infracción emitidas por `agency`, el **monto máximo** de las multas con esa infracción emitidas por `agency` y **la diferencia entre ambos montos**, donde `agency` es un parámetro adicional (donde debe reemplazar los guión bajo "_" por espacios " ").

El orden de impresión es **descendente por diferencia** entre ambos montos y **desempata alfabético** por **infracción**.

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

Sólo se deben listar hasta **las primeras "n" infracciones**, donde `n` es un parámetro adicional.

- Parámetros adicionales: `n` (Entero mayor o igual a 1), `agency` (String)

- Ejemplo de invocación: `sh query4.sh -Daddresses='10.6.0.1:5701' -Dcity=NYC -DinPath=. -DoutPath=.` **`-Dn=3 -Dagency=DEPARTMENT_OF_TRANSPORTATION`**

- Salida de ejemplo para NYC:

```
Infraction;Max;Min;Diff
PHTO SCHOOL ZN SPEED VIOLATION;35;650;615
BLUE ZONE;9;604;595
BUS LANE VIOLATION;29;624;595
```

- Ejemplo de invocación: `sh query4.sh -Daddresses='10.6.0.1:5701' -Dcity=CHI -DinPath=. -DoutPath=.` **`-Dn=3 -Dagency=CPD-Airport`**

- Salida de ejemplo para CHI:

```
Infraction;Max;Min;Diff
OBSTRUCTED OR IMPROPERLY TINTED WINDOWS;25;250;225
SMOKED/TINTED WINDOWS PARKED/STANDING;25;250;225
PARK OR BLOCK ALLEY;0;200;200
```

Query 5: Infracciones con multas 24/7 en el rango `[from, to]` (Exclusivo para el grupo de 4 integrantes)

Donde cada línea de la salida contenga **la infracción** que cuente con al menos una multa labrada en cada una de las 24 horas del día para cada uno de los días de un rango de días `[from, to]`, donde `from` y `to` son dos parámetros adicionales.

Por ejemplo si no existe al menos una multa labrada con la infracción STORAGE-3HR COMMERCIAL para la hora 22 de un día cualquiera del rango `[from, to]`, entonces la infracción STORAGE-3HR COMMERCIAL no debe listarse.

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

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

- Parámetros adicionales: `from` (String DD/MM/YY), `to` (String DD/MM/YY)
- Ejemplo de invocación: `sh query5.sh -Daddresses='10.6.0.1:5701' -Dcity=NYC -DinPath=. -DoutPath=. -Dfrom=01/01/2020 -Dto=31/12/2020`
- Salida de ejemplo para NYC:

```
Infraction
EXCAVATION-VEHICLE OBSTR TRAFF
NO STOPPING-DAY/TIME LIMITS
VIN OBSCURED
...
```

- Salida de ejemplo para CHI:

```
Infraction
RUSH HOUR PARKING
SAFETY CHAINS REQUIRED
THEATER ENTRANCE/EXIT
...
```

Hechos y Consideraciones

Para simplificar el desarrollo se pueden tomar los siguientes considerandos como ciertos:

- El formato de los archivos es correcto, no es necesario validar que los nombres de las columnas coincidan, ni que el tipo de dato de los valores de una columna es el esperado ni que la cantidad de columnas es la esperada.
- Los valores de los parámetros de los clientes no contienen espacios (para el caso de `agency` se espera guion bajo en lugar de espacios)
- Si el archivo de salida existe, reemplazar el contenido (no *appendear*)

Requisitos

El trabajo práctico debe realizarse en los mismos grupos formados para el primer trabajo práctico especial.

72.42 Programación de Objetos Distribuidos

Se requiere implementar:

- Cada una de las consultas utilizando **uno o más *job*s MapReduce** que pueda correr en un ambiente distribuido utilizando un *grid* de Hazelcast.
- Los clientes indicados cada uno como una aplicación de consola diferente

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.

Muy Importante:

➔ **Respetar EXACTAMENTE**
- **Los nombres de los *scripts* y sus parámetros** (Si se pide `-DinPath` no se aceptará `-DinputPath`, `-DpathEntrada`, etc.)
- **El orden y formato de salida enunciado, tanto para los clientes de consola como para los archivos CSV de salida**

➔ **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>`**`tpe1-g7-api`**`</artifactId>`**
- El `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>`**`tpe1-g7-api`**`</name>`**
- 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 `g7` para así evitar conflictos con las colecciones y poder hacer pruebas de distintos alumnos en simultáneo utilizando la misma red.

Material a entregar

Cada grupo deberá subir al **Campus UNICAMENTE 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

    - El directorio oculto `.git/` donde se encuentra la historia de commits y modificaciones. Recordar que la descarga desde github.com no incluye el directorio .git/.
    - **No se deben entregar los binarios.** Recordar de ejecutar el comando `mvn clean` antes de la entrega.
- 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).

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 cluster 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 23/10**
- **Entrega del trabajo: Estará disponible hasta el jueves 07/11 a las 23:59** la actividad "TPE 2" localizada en la sección Contenido / Evaluaciones. En la misma deberán cargar el archivo compactado.
- **El día del coloquio será el miércoles 20/11**.
- **El día del recuperatorio será el miércoles 27/11**
- **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 y métodos útiles para consultar**
  - `java.lang.System#getProperty(java.lang.String)`
  - `java.nio.file.Files.write`
  - `java.nio.file.Files.lines`
  - `java.text.DecimalFormat` y `java.math.RoundingMode`
  - `java.time.format.DateTimeFormatter`
  - `java.time.YearMonth`