Programación de Objetos Distribuidos
Segundo Trabajo Práctico Especial

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 vuelos desde y/o hacia aeropuertos nacionales, basado en datos reales.

Los datos que alimentarán el dataset son datos oficiales elaborados por la EANA (Empresa Argentina de Navegación Aérea) y la ANAC (Administración Nacional de Aviación Civil).

Descripción Funcional

El programa deberá leer de dos archivos csv:

- Los datos de los aeropuertos nacionales
- La información de aterrizajes y despegues registrados por el EANA,

donde el delimitador de campos es un punto y coma. Los archivos tienen las siguientes características:

Detalle de aeródromos y helipuertos, a partir de ahora **aeropuertos.csv**

- **Origen y Consulta:** https://servicios.transporte.gob.ar/gobierno_abierto/detalle.php?t=aeropuertos&d=detalle
- **Descarga:** Pampero**. /afs/it.itba.edu.ar/pub/pod/aeropuertos.csv**
- **Campos Relevantes:**
  - **`oaci`**: El **código de aeropuertos de OACI** es el código de designación de aeropuertos compuesto de cuatro caracteres alfanuméricos que sirve para identificar cada aeropuerto alrededor del mundo. https://es.wikipedia.org/wiki/Código_de_aeropuertos_de_OACI
  - **`denominacion`**: nombre del aeropuerto.
  - **`provincia`**: nombre de la provincia donde se ubica el aeropuerto.

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 un aeropuerto (o similar) conteniendo los datos de cada una de los campos, separados por ;.

Ejemplo de tres líneas del archivo, con encabezado e información de los aeropuertos de "CORONEL BOGADO" y "PUERTO DESEADO".

```
local;oaci;iata;tipo;denominacion;coordenadas;latitud;longitud;elev;uom_elev;ref;distancia_ref;direccion_ref;condicion;control;region;fir;uso;trafico;sna;concesionado;provincia;inhab
...
```

```
CBA;SACO;COR;Aeródromo;CÓRDOBA/ING. AER. A. L. V. TARAVELLA;31°18'36"S
64°12'30"W;-64.20833333;-31.31000000;489.00;Metros;Córdoba;9.0;NNO;PUBLICO
;CONTROL;RANO;SACF;CIVIL;Internacional;SI;SI;CÓRDOBA;NO
...
RVA;;;Aeródromo;AMÉRICA;35°30' 7"S
62°59'24"W;-62.99000000;-35.50194444;106.00;Metros;América;1.0;SO;PUBLICO;
NOCONTROL;RACE;SAEF;CIVIL;Nacional;NO;NO;BUENOS AIRES;NO
```

El aeropuerto de la segunda línea del ejemplo se llama CÓRDOBA/ING. AER. A. L. V. TARAVELLA y está ubicado en la provincia de Córdoba y su identificación es el código local CBA. Cuenta con el código OACI SACO.

El aeropuerto de la tercera línea del ejemplo es el aeródromo de América ubicado en la provincia de Buenos Aires. Únicamente cuenta con el identificador local RVA.

A partir de ahora denominaremos aeropuerto a cualquier establecimiento del archivo `aeropuertos.csv` sin considerar el campo `Tipo` (es decir, sin importar si es Aeródromo o Helipuerto).

Detalle de movimientos de aterrizajes y despegues, a partir de ahora **movimientos.csv**

- **Origen y Consulta:** https://servicios.transporte.gob.ar/gobierno_abierto/detalle.php?t=eana&d=detalle
- **Descarga:** Pampero**. /afs/it.itba.edu.ar/pub/pod/movimientos.csv**
- **Campos Relevantes:**
  - **`Clasificación Vuelo`**: Cabotaje, Internacional y N/A.
  - **`Tipo de Movimiento`**: Despegue y Aterrizaje.
  - **`Clase Vuelo`**: No Regular, Regular, Vuelo Privado con Matrícula Extranjera y Vuelo Privado con Matrícula Nacional
  - **`Origen OACI`**: Código OACI del aeropuerto del cual despegó el vuelo. Dato obligatorio, no está en blanco.
  - **`Destino OACI`**: Código OACI del aeropuerto en el cual aterrizó el vuelo. Tanto este como el anterior pueden ser aeropuertos de otros países y por lo tanto no figurar en el archivo de aeropuertos. Es un dato obligatorio, no está en blanco.

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 un movimiento (**aterrizaje ó despegue**) conteniendo los datos de cada una de los campos, separados por ;.

Ejemplo de tres líneas del archivo, con encabezado e información de dos movimientos:

```
Fecha;Hora;Clase de Vuelo;Clasificación Vuelo;Tipo de Movimiento;Origen
OACI;Destino OACI;Aerolinea Nombre;Aeronave
...
30/06/2019;10;Regular;Internacional;Despegue;SACO;KMIA;American
Airlines;BOEING B-767
...
30/06/2019;15;Regular;Cabotaje;Aterrizaje;SADP;SACO;JetSMART
Airlines;AIRBUS A-320
```

La segunda línea del ejemplo se refiere a un vuelo internacional de American Airlines que despegó a las 10 hs del 30 de junio de 2019 en el aeropuerto de Ezeiza con destino al aeropuerto cuyo código OACI es KMIA. El código KMIA identifica al Aeropuerto Internacional de Miami, Estados Unidos. El aeropuerto KMIA no estará presente en el archivo `aeropuertos.csv` ya que el mismo contiene únicamente aeropuertos nacionales.

La tercera línea del ejemplo se refiere a un vuelo de cabotaje de JetSMART Airlines que aterrizó en el aeropuerto de Córdoba (del ejemplo de aeropuertos.csv) a las 15 hs del 30 de junio de 2019. El mismo provino del aeropuerto de código OACI SADP (El Palomar).

Se considera “Movimiento” al despegue o aterrizaje de un avión en un aeropuerto. Para los propósitos de tráfico del aeropuerto, una llegada y una salida se cuentan como dos movimientos.

Como los datos provienen de un sitio oficial, se asume que el formato y contenido de los archivos es correcto y no es necesario validarlo.

Requerimientos

La aplicación debe poder resolver un conjunto de consultas listadas más abajo. En cada una 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 <u>sólo una</u> de las queries sobre los datos obtenidos a partir de los archivos CSV provistos en esa invocación (archivos `aeropuertos.csv` y `movimientos.csv`).

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:

```
15/10/2019 14:43:09:0223 INFO  [main] Client (Client.java:76) - Inicio de
la lectura del archivo
15/10/2019 14:43:23:0011 INFO  [main] Client (Client.java:173) - Fin de
lectura del archivo
15/10/2019 14:43:23:0013 INFO  [main] Client (Client.java:87) - Inicio del
trabajo map/reduce
15/10/2019 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)
- `-DinPath` indica el path donde están ambos archivos de entrada `aeropuertos.csv` y `movimientos.csv`.
- `-DoutPath` indica el path donde estarán ambos archivos de salida `query1.csv` y `query1.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/
-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/movimientos.csv` y `/afs/it.itba.edu.ar/pub/pod/aeropuertos.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/query1.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.

Muy Importante:

- ➔ **Respetar exactamente el nombre del script, los nombres de los archivos de entrada y salida y el orden y formato de los parámetros del script.**
- ➔ **El nombre del cluster (<group><name>) y los nombres de las colecciones de Hazelcast** a utilizar en la implementación deben ser “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**. Para todas las queries, tener en cuenta que el archivo de salida debe contener la línea de encabezado correspondiente indicada en las salidas de ejemplo.

Query 1: Movimientos por aeropuerto

Donde cada línea de la salida contenga separados por ; el código OACI, la denominación y la cantidad de movimientos del aeropuerto (contabilizando únicamente despegues que tengan al aeropuerto como origen y aterrizajes que tengan al aeropuerto como destino).

El orden de impresión es descendente por movimientos y luego alfabéticamente por código OACI.

Sólo se deberán listar los aeropuertos presentes en `aeropuertos.csv` que registren uno o más movimientos.

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

```
OACI;Denominación;Movimientos
SABE;BUENOS AIRES/AEROPARQUE J. NEWBERY;516161
SAEZ;EZEIZA/MINISTRO PISTARINI;289867
...
SAZD;DOLORES;2
SAWP;PERITO MORENO;1
SAZI;BOLIVAR;1
```

Query 2: Top N aerolíneas según el porcentaje de movimientos de cabotaje, ordenado descendentemente por dicho porcentaje

Donde cada línea de la salida contenga separados por ';' el nombre de la aerolínea y el porcentaje de movimientos de cabotaje de esa aerolínea respecto del total de movimientos de cabotaje (de todas las aerolíneas).

Se deben listar únicamente las `n` aerolíneas con mayor porcentaje de movimientos de cabotaje y una última línea adicional sumarizando el porcentaje de movimientos de cabotaje del resto de las aerolíneas no listadas en la salida (incluyendo aquellos movimientos de cabotaje que no tengan asociado un Aerolínea Nombre o el mismo sea N/A).

El orden de impresión es descendente por porcentaje y luego alfabéticamente por nombre de aerolínea. La línea Otros siempre debe ser la última de la salida. Todos los porcentajes deben imprimirse con dos dígitos decimales (truncados).

- Parámetros adicionales: **n**
- **Ejemplo de invocación:** `./query2 -Daddresses='10.6.0.1:5701' -DinPath=. -DoutPath=. -Dn=5`
- Salida de ejemplo:

```
Aerolínea;Porcentaje
Aerolíneas Argentinas;29.74%
Austral Líneas Aéreas;22.17%
LATAM Argentina;11.77%
Andes Líneas Aéreas;2.14%
SOL Líneas Aéreas;1.69%
Otros;32.46%
```

Query 3: Pares de aeropuertos que registran la misma cantidad de miles de movimientos

Donde cada línea de la salida contenga separados por **;** el grupo de miles de movimientos, el código OACI del aeropuerto A que corresponde al grupo de movimientos y el código OACI del aeropuerto B que corresponde al grupo de movimientos contabilizando únicamente despegues que tengan al aeropuerto como origen y aterrizajes que tengan al aeropuerto como destino.

El último grupo posible a mostrar es el de 1000 (incluyendo aquellos pares de aeropuertos que registren entre 1000 movimientos inclusive y 1999 movimientos inclusive), es decir, no listar los pares de aeropuertos que registren hasta 999 movimientos inclusive.

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

Si un grupo está compuesto por un único aeropuerto, 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 código OACI.

- Parámetros adicionales: **Ninguno**
- **Ejemplo de invocación:** `./query3 -Daddresses='10.6.0.1:5701' -DinPath=. -DoutPath=.`
- Salida de ejemplo:

```
Grupo;Aeropuerto A;Aeropuerto B
24000;SAWC;SAZM
21000;SAVT;SAZB
...
1000;SAMM;SANH
1000;SAMM;SAZF
1000;SANH;SAZF
```

Query 4: Los `n` aeropuertos destino con mayor cantidad de movimientos de despegue que tienen como origen a un aeropuerto `oaci`

Donde cada línea de la salida contenga separados por `;` el código OACI del aeropuerto destino del movimiento y la cantidad de despegues que tienen como Origen OACI al aeropuerto recibido por parámetro y como Destino OACI al aeropuerto de la línea.

El orden de impresión es descendente por movimientos y luego alfabéticamente por código OACI.

Se deben listar únicamente las `n` primeras líneas.

- Parámetros adicionales: `oaci, n`
- **Ejemplo de invocación:** `./query4 -Daddresses='10.6.0.1:5701' -DinPath=. -DoutPath=. -Doaci=SAEZ -Dn=5`
- Salida de ejemplo:

```
OACI;Despegues
SCEL;17837
SBGR;12851
SPJC;10024
SBGL;8398
KMIA;8086
```

Query 5: Los `n` aeropuertos con menor porcentaje de vuelos privados

Donde cada línea de la salida contenga separados por `;` el código OACI del aeropuerto y el porcentaje de movimientos de vuelos privados que tengan como Origen OACI o Destino OACI al aeropuerto de la fila respecto del total de movimientos de todo tipo que tengan como Origen OACI o Destino OACI al aeropuerto de la fila.

Un movimiento es considerado de vuelo privado cuando el campo Clase de Vuelo es “Vuelo Privado con Matrícula Nacional” o “Vuelo Privado con Matrícula Extranjera”.

El orden de impresión es ascendente por porcentaje y luego alfabéticamente por código OACI.

Sólo se deberán listar los aeropuertos presentes en `aeropuertos.csv`.

Se deben listar únicamente las `n` primeras líneas. Todos los porcentajes deben imprimirse con dos dígitos decimales (truncados).

- Parámetros adicionales: **n**
- **Ejemplo de invocación:** `./query5 -Daddresses='10.6.0.1:5701' -DinPath=. -DoutPath=. -Dn=5`
- Salida de ejemplo:

```
OACI;Porcentaje
SAWC;1.60%
SAEZ;2.18%
SARI;2.73%
SABE;4.28%
SAME;4.83%
```

Query 6: Pares de provincias que comparten al menos `min` movimientos

Donde cada línea de la salida contenga separados por `;` el par de provincias distintas y la cantidad de movimientos de todo tipo que comparten ambas provincias.

Dos provincias A y B comparten un movimiento si la provincia del aeropuerto de origen/destino OACI del movimiento es A y la provincia del aeropuerto de destino/origen OACI del movimiento es B.

El orden de impresión es descendente por movimientos y el orden de los pares es alfabético por nombre.

Se deben listar únicamente los pares de provincias distintas que comparten `min` o más movimientos.

- ❏ Parámetros adicionales: **min**
- ❏ **Ejemplo de invocación:** `./query6 -Daddresses='10.6.0.1:5701' -DinPath=. -DoutPath=. -Dmin=50000`
- ❏ Salida de ejemplo:

```
Provincia A;Provincia B;Movimientos
CIUDAD AUTÓNOMA DE BUENOS AIRES;CÓRDOBA;87303
BUENOS AIRES;CIUDAD AUTÓNOMA DE BUENOS AIRES;70633
CIUDAD AUTÓNOMA DE BUENOS AIRES;MISIONES;68406
CIUDAD AUTÓNOMA DE BUENOS AIRES;MENDOZA;67999
CHUBUT;CIUDAD AUTÓNOMA DE BUENOS AIRES;64656
CIUDAD AUTÓNOMA DE BUENOS AIRES;RÍO NEGRO;60868
CIUDAD AUTÓNOMA DE BUENOS AIRES;NEUQUÉN;55630
CIUDAD AUTÓNOMA DE BUENOS AIRES;SALTA;52814
```

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.

Se debe entregar

- 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*.
    - ***Importante:*** 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>`
  - 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 para ejecutar.
- 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, corriendo en clusters, variando la cantidad de nodos e indicando cuál sería la cantidad de nodos mejor para cada una (analizando clústeres de hasta 4 nodos).

Entregas

- **Antes del Domingo 03/11/2019 a las 23:59** deben cargar el ***código fuente*** y el ***documento*** del trabajo práctico especial en la actividad "Entrega TPE" localizada en la sección Contenido / Evaluación / TPE 2 de Campus ITBA.
- **El día 12/11/2019 a las 19:00hs** cada grupo tendrá 15 minutos para configurar su entorno y mostrar la ejecución de la aplicación a la cátedra realizando corridas sobre una misma configuración de nodos a utilizar, con archivos (datasets) a determinar por la cátedra. Se tomará nota de las respuestas y los tiempos de ejecución y esto será parte de la evaluación del trabajo. A criterio de la cátedra también se podrán realizar preguntas sobre la implementación de los mismo.
- **El día del recuperatorio será el 19/11/2019.** Para aquellos equipos que lo requieran, se les podrá pedir que corrijan o agreguen algún elemento o query al proyecto o realizar la implementación de nuevas consultas.
- ***No se aceptarán entregas fuera de los días y horarios establecidos.***

Recomendaciones

- **Clases útiles para consultar**
  - **Hazelcast**
    - `com.hazelcast.mapreduce.Combiner`
    - `com.hazelcast.mapreduce.Collator`
  - **Hazelcast Client**
    - `com.hazelcast.config.GroupConfig`
    - `com.hazelcast.client.config.ClientNetworkConfig`
    - `com.hazelcast.client.config.ClientConfig`
    - `com.hazelcast.client.HazelcastClient`

    - **Java**
        - `java.nio.file.`**`Files`**
        - `java.text.`**`DecimalFormat`**
        - `java.math.`**`RoundingMode`**
- Utilizar la versión **3.7.8** de **`hazelcast-all`**
- **Hazelcast Management Center** para monitorear los nodos del cluster. Nota: Usar una versión coincidente con la de `hazelcast-all` (Link).