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 arbolado público, basado en datos reales.
Para este trabajo se busca poder procesar datos del arbolado de dos ciudades:

- Ciudad Autónoma de Buenos Aires, Argentina 🇦🇷
- Vancouver, British Columbia, Canadá 🇨🇦.

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

Descripción Funcional

A continuación se listan los dos ejemplos de uso que se busca para la aplicación: procesar los datos de árboles de la Ciudad Autónoma de Buenos Aires 🇦🇷 y de Vancouver 🇨🇦 . 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, los barrios serán los que la aplicación obtenga en ejecución a partir del archivo de barrios y no serán aceptadas implementaciones que tengan fijos estos datos.** En otras palabras, la implementación deberá funcionar también para procesar los datos de árboles de cualquier otra ciudad, manteniendo siempre la estructura de los archivos que se presenta a continuación.

Datos de árboles de Buenos Aires 🇦🇷, a partir de ahora `arbolesBUE.csv`

- **Origen:** https://data.buenosaires.gob.ar/dataset/arbolado-publico-lineal/archivo/ecf38a47-563f-42c1-9bd4-7cedf35d536b
- **Descarga:** `/afs/it.itba.edu.ar/pub/pod/arbolesBUE.csv`
- **Campos Relevantes:**
  - **comuna:** Nombre del barrio donde se encuentra el árbol.
  - **calle_nombre**: Nombre de la calle donde se encuentra el árbol
  - **nombre_cientifico**: Nombre científico del árbol
  - **diametro_altura_pecho**: Diámetro del árbol a la altura del pecho

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

```
nro_registro;tipo_activ;comuna;manzana;calle_nombre;calle_altura;direccion_normalizada;nombre_cientifico;estado_plantera;ubicacion_plantera;nivel_plantera;diametro_altura_pecho;altura_arbol
79838;Lineal;1; ;Eyle Petrona;0;EYLE, PETRONA 47;Platanus x acerifolia;Ocupada;Fuera de línea;A nivel;1;17
...
```

Datos de barrios de Buenos Aires 🇦🇷, a partir de ahora **barriosBUE.csv**

- **Descarga: /afs/it.itba.edu.ar/pub/pod/barriosBUE.csv**
- **Campos:**
  - **nombre**: Nombre del barrio, para relacionarlo con el campo **comuna** de arbolesBUE.csv.
  - **habitantes**:Cantidad de habitantes del barrio.

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

```
nombre;habitantes
2;149607
14;227003
15;182427
...
```

Datos de árboles de Vancouver 🇨🇦, a partir de ahora **arbolesVAN.csv**

- **Origen:** https://opendata.vancouver.ca/explore/dataset/street-trees/export/?disjunctive.species_name&disjunctive.common_name&disjunctive.height_range_id
- **Descarga: /afs/it.itba.edu.ar/pub/pod/arbolesVAN.csv**
- **Campos Relevantes:**
  - **NEIGHBOURHOOD_NAME:** Nombre del barrio donde se encuentra el árbol.
  - **STD_STREET**: Nombre de la calle donde se encuentra el árbol
  - **COMMON_NAME**: Nombre científico del árbol
  - **DIAMETER**: Diámetro del árbol a la altura del pecho

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 árbol conteniendo los datos de cada uno de los campos, separados por ";".

```
TREE_ID;CIVIC_NUMBER;STD_STREET;GENUS_NAME;SPECIES_NAME;CULTIVAR_NAME;COMMON_NAME;ASSIGNED;ROOT_BARRIER;PLANT_AREA;ON_STREET_BLOCK;ON_STREET;NEIGHBOURHOOD_NAME;STREET_SIDE_NAME;HEIGHT_RANGE_ID;DIAMETER;CURB;DATE_PLANTED;Geom
```

```
20666;2838;W 19TH AV;PRUNUS;CERASIFERA;ATROPURPUREUM;PISSARD PLUM;N;N;10;2800;W 19TH AV;ARBUTUS-RIDGE;EVEN;3;20.0;Y;;"{""type"": ""Point"", ""coordinates"": [-123.16858, 49.25546]}"
...
```

Datos de barrios de Vancouver 🇨🇦, a partir de ahora `barriosVAN.csv`

- **Descarga:** `/afs/it.itba.edu.ar/pub/pod/barriosVAN.csv`
- **Campos:**
  - **nombre**: Nombre del barrio, para relacionarlo con el campo **NEIGHBOURHOOD_NAME** de arbolesVAN.csv.
  - **habitantes**:Cantidad de habitantes del barrio.

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

```
nombre;habitantes
WEST POINT GREY;13065
HASTINGS-SUNRISE;34575
KERRISDALE;13975
...
```

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 barrios y de árboles).

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:

```
15/10/2020 14:43:09:0223 INFO  [main] Client (Client.java:76) - Inicio de
la lectura del archivo
15/10/2020 14:43:23:0011 INFO  [main] Client (Client.java:173) - Fin de
lectura del archivo
15/10/2020 14:43:23:0013 INFO  [main] Client (Client.java:87) - Inicio del
trabajo map/reduce
15/10/2020 14:43:23:0490 INFO  [main] Client (Client.java:166) - Fin del
trabajo map/reduce
```

Por ejemplo:

```
$> ./queryX -Dcity=C -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`.
- `-Dcity` indica con qué dataset de ciudad se desea trabajar. Los únicos valores posibles son **`BUE`** y **`VAN`**.
- `-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 de barrios y de árboles.
- `-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 -Dcity=BUE -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 de 🇦🇷 presentes en `/afs/it.itba.edu.ar/pub/pod/barriosBUE.csv` y `/afs/it.itba.edu.ar/pub/pod/arbolesBUE.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 árboles por habitante

Donde cada línea de la salida contenga, separados por ";" el nombre del barrio y el total de árboles por habitante (que consiste en el cociente entre el total de árboles de ese barrio y el número de habitantes del mismo).

Sólo se deben listar los barrios presentes en el archivo CSV de barrios.

El orden de impresión es descendente por el total de árboles por habitante y luego alfabético por nombre de barrio.

El total de árboles por habitante debe imprimirse truncado con dos decimales.

- Parámetros adicionales: Ninguno

- Ejemplo de invocación: `./query1 -Dcity=BUE -Daddresses='10.6.0.1:5701' -DinPath=. -DoutPath=.`

- Salida de ejemplo para 🇦🇷:

```
BARRIO;ARBOLES_POR_HABITANTE
9;0.25
10;0.19
11;0.19
15;0.16
12;0.11
...
```

- Salida de ejemplo para 🇨🇦:

```
BARRIO;ARBOLES_POR_HABITANTE
SHAUGHNESSY;0.93
KERRISDALE;0.52
SOUTH CAMBIE;0.51
DUNBAR-SOUTHLANDS;0.50
WEST POINT GREY;0.43
...
```

Query 2: La calle por cada barrio con más árboles (mayores a `min`)

Donde cada línea de la salida contenga, separados por ";" el nombre del barrio, el nombre de la calle con más árboles de ese barrio y la cantidad de árboles que hay en esa calle de ese barrio.

Sólo se deben listar los barrios presentes en el archivo CSV de barrios.

Sólo se deben listar los barrios cuya calle con más árboles de ese barrio tenga una cantidad mayor al parámetro `min`.

El orden de impresión es alfabético por el nombre del barrio.

- Parámetros adicionales: `min` (Un valor entero mayor a cero)

- Ejemplo de invocación: `./query2 -Dcity=VAN -Daddresses='10.6.0.1:5701' -DinPath=. -DoutPath=. -Dmin=100`

- Salida de ejemplo para 🇦🇷:

```
BARRIO;CALLE_CON_MAS_ARBOLES;ARBOLES
1;Saenz Peña Roque,Pres. Diagonal Norte Av.;120
10;Lascano;436
11;Lamarca Emilio;721
12;Quesada;532
13;3 de Febrero;439
...
```

- Salida de ejemplo para 🇨🇦:

```
BARRIO;CALLE_CON_MAS_ARBOLES;ARBOLES
ARBUTUS-RIDGE;W KING EDWARD AV;271
DOWNTOWN;PACIFIC BOULEVARD;163
DUNBAR-SOUTHLANDS;W KING EDWARD AV;392
FAIRVIEW;W BROADWAY;386
GRANDVIEW-WOODLAND;VICTORIA DRIVE;276
...
```

Query 3: Top n especies de árbol con mayor promedio del diámetro

Donde cada línea de la salida contenga, separados por ";" el nombre de la especie del árbol y el promedio del diámetro del árbol a la altura del pecho de esa especie.

El orden de impresión es descendente por diámetro y luego alfabético por nombre de la especie.

Solo se listan las primeras **n** especies del resultado

El promedio del diámetro del árbol a la altura del pecho debe imprimirse truncado con dos decimales.

- Parámetros adicionales: `n` (Un valor entero mayor a cero)

- Ejemplo de invocación: `./query3 -Dcity=BUE -Daddresses='10.6.0.1:5701' -DinPath=. -DoutPath=. -Dn=5`

- Salida de ejemplo para 🇦🇷:

```
NOMBRE_CIENTIFICO;PROMEDIO_DIAMETRO
Sterculia coccinea;75.20
Eucalyptus tereticornis;71.11
Tristania conferta;69.37
Salix alba;66.56
Caryota urens;65.27
```

- Salida de ejemplo para 🇨🇦:

```
NOMBRE_CIENTIFICO;PROMEDIO_DIAMETRO
MANCHURIAN BIRCH;37.76
LEYLAND CYPRESS;36.53
AMERICAN CHESTNUT;32.11
GRAY POPLAR;32.11
JAPANESE WALNUT;28.16
```

Query 4: Pares de barrios que contengan al menos `min` árboles de una especie x

Donde cada línea de la salida contenga separados por `;` el nombre del barrio A y el nombre del barrio B donde los barrios A y B cuentan con al menos `min` árboles de la especie `X` en su barrio.

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

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

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

- Parámetros adicionales: `min` (Un valor entero no negativo) y `x` (Un string)
- Ejemplo de invocación: `./query5 -Dcity=BUE -Daddresses='10.6.0.1:5701' -DinPath=. -DoutPath=. -Dmin=11000 -Dname='Fraxinus pennsylvanica'`
- Salida de ejemplo para 🇦🇷:

```
Barrio A;Barrio B
10;12
10;13
10;4
10;9
12;13
12;4
12;9
13;4
13;9
4;9
```

- Salida de ejemplo para 🇨🇦:

```
Barrio A;Barrio B
GRANDVIEW-WOODLAND;KERRISDALE
GRANDVIEW-WOODLAND;KILLARNEY
```

```
GRANDVIEW-WOODLAND;MARPOLE
KERRISDALE;KILLARNEY
KERRISDALE;MARPOLE
KILLARNEY;MARPOLE
```

Query 5: Pares de barrios que registran la misma cantidad de miles de árboles

Donde cada línea de la salida contenga separados por `;` el grupo de miles de árboles, el nombre del barrio A que corresponde al grupo de árboles y el nombre del barrio B que corresponde al grupo de de árboles.

No se deben listar los pares de barrios que registren hasta 999 árboles inclusive.

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

Si un grupo está compuesto por un único barrio, 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 de barrio.

- ❏ Parámetros adicionales: Ninguno

- ❏ Ejemplo de invocación: `./query4 -Dcity=VAN -Daddresses='10.6.0.1:5701' -DinPath=. -DoutPath=.`

- ❏ Salida de ejemplo para 🇦🇷:

```
Grupo;Barrio A;Barrio B
15000;5;6
```

- ❏ Salida de ejemplo para 🇨🇦:

```
Grupo;Barrio A;Barrio B
10000;HASTINGS-SUNRISE;KENSINGTON-CEDAR COTTAGE
8000;KITSILANO;SUNSET
7000;SHAUGHNESSY;VICTORIA-FRASERVIEW
6000;GRANDVIEW-WOODLAND;KERRISDALE
6000;GRANDVIEW-WOODLAND;KILLARNEY
6000;GRANDVIEW-WOODLAND;MARPOLE
6000;GRANDVIEW-WOODLAND;MOUNT PLEASANT
6000;GRANDVIEW-WOODLAND;RILEY PARK
6000;KERRISDALE;KILLARNEY
6000;KERRISDALE;MARPOLE
6000;KERRISDALE;MOUNT PLEASANT
6000;KERRISDALE;RILEY PARK
6000;KILLARNEY;MARPOLE
6000;KILLARNEY;MOUNT PLEASANT
```

```
6000;KILLARNEY;RILEY PARK
6000;MARPOLE;MOUNT PLEASANT
6000;MARPOLE;RILEY PARK
6000;MOUNT PLEASANT;RILEY PARK
...
```

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: "`tpe1-gX-Z`" donde X es el número de grupo y Z es `parent`, `api`, `server` o `client`. Por ejemplo: **`<name>`**`tpe1-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.
- 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 alguno 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 si se lo solicitase.**

Cronograma

- **Presentación del Enunciado: miércoles 14/10**
- **Antes del jueves 29/10 a las 23:59** deben cargar el **código fuente** y el **documento** en la actividad "Entrega TPE 2" localizada en la sección Contenido / Evaluación / TPE 2 de Campus ITBA.
- **El día miércoles 11/11 a las 18:00** cada grupo tendrá un espacio de 15 minutos para mostrar la ejecución de la aplicación a la cátedra. Para ello un integrante del equipo deberá compartir pantalla. Se tomará nota de las respuestas obtenidas con los clientes y esto será parte de la evaluación del trabajo. A criterio de la cátedra también se podrán realizar preguntas sobre la implementación.
  Los coloquios se llevarán a cabo en un aula virtual ya creada para cada grupo de Campus. Durante el mismo se les hará una devolución del trabajo, indicando la nota y los principales errores cometidos. Todos los integrantes del grupo deben acceder primero a la herramienta “Grupos”, elegir su grupo, y luego en “Herramientas del Grupo” elegir la opción “Collaborate”. Por último, deben presionar el botón “Unirse a la sala” para acceder a la sala del grupo. No será necesario que activen la cámara, sí el micrófono. Todos los integrantes (salvo el primer grupo) deberán estar presentes en la sala del grupo diez minutos antes del horario establecido, ya que el horario es aproximado. El horario de cada grupo será comunicado como anuncio de Campus.
- **El día del recuperatorio será el miércoles 18/11.**
- **No se aceptarán entregas pasado el día y horario establecido como límite.**

Dudas sobre el TPE

Las mismas deben volcarse al Foro de Discusión “TPE” del Campus ITBA.

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`**
- **Hazelcast Management Center** para monitorear los nodos del cluster. Nota: Usar una versión coincidente con la de `hazelcast-all` (Link).