

Trabajo Práctico Especial 2: Árboles

13 de Octubre de 2021

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>
- ❖ **Descarga:** /afs/it.itba.edu.ar/pub/pod/2021/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

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
```

72.42 Programación de Objetos Distribuidos

```
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/2021/barriosBUE.csv](https://afs/it.itba.edu.ar/pub/pod/2021/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/>
- ❖ **Descarga:** [/afs/it.itba.edu.ar/pub/pod/2021/arbolesVAN.csv](https://afs/it.itba.edu.ar/pub/pod/2021/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

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;CULTIVA  
R_NAME;COMMON_NAME;ASSIGNED;ROOT_BARRIER;PLANT_AREA;ON_STREET_BLOCK;  
ON_STREET;NEIGHBOURHOOD_NAME;STREET_SIDE_NAME;HEIGHT_RANGE_ID;DIAMET  
ER;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]}  
...
```

72.42 Programación de Objetos Distribuidos

Datos de barrios de Vancouver , a partir de ahora **barriosVAN.csv**

- ❖ **Descarga:** [/afs/it.itba.edu.ar/pub/pod/2021/barriosVAN.csv](http://afs/it.itba.edu.ar/pub/pod/2021/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:

```
13/10/2021 14:43:09:0223 INFO [main] Client (Client.java:76) - Inicio de
la lectura del archivo
13/10/2021 14:43:23:0011 INFO [main] Client (Client.java:173) - Fin de
lectura del archivo
```

72.42 Programación de Objetos Distribuidos

```
13/10/2021 14:43:23:0013 INFO [main] Client (Client.java:87) - Inicio del trabajo map/reduce
13/10/2021 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: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 time1.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/2021/
-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/2021/barriosBUE.csv y
/afs/it.itba.edu.ar/pub/pod/2021/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.

72.42 Programación de Objetos Distribuidos

Query 1: Total de árboles por barrio

Donde cada línea de la salida contenga, separados por “;” el nombre del barrio y el total de árboles pertenecientes al barrio.

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 barrio y luego alfabético por nombre de barrio.

- Parámetros adicionales: Ninguno

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

- Salida de ejemplo para :

NEIGHBOURHOOD;TREES

12;38921

11;36104

9;36104

10;32625

4;31221

...

- Salida de ejemplo para :

NEIGHBOURHOOD;TREES

RENFREW-COLLINGWOOD;13621

KENSINGTON-CEDAR COTTAGE;12721

HASTINGS-SUNRISE;11518

DUNBAR-SOUTHLANDS;9342

KITSILANO;7991

...

Query 2: Para cada barrio, la especie con mayor cantidad de árboles por habitante

Donde cada línea de la salida contenga, separados por “;” el nombre del barrio, el nombre científico de la especie con mayor índice de árboles por habitantes de ese barrio y el índice 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 alfabético por nombre de barrio.

El índice debe imprimirse truncado con dos decimales.

- Parámetros adicionales: Ninguno

72.42 Programación de Objetos Distribuidos

- Ejemplo de invocación: ./query2 -Dcity=BUE -Daddresses='10.6.0.1:5701' -DinPath=. -DoutPath=.
- Salida de ejemplo para 🇦🇷:

```
NEIGHBOURHOOD;COMMON_NAME;TREES_PER_PEOPLE
1;Fraxinus pennsylvanica;0.01
10;Fraxinus pennsylvanica;0.07
11;Fraxinus pennsylvanica;0.05
12;Fraxinus pennsylvanica;0.07
13;Fraxinus pennsylvanica;0.03
...
```

- Salida de ejemplo para 🇨🇦:

```
NEIGHBOURHOOD;COMMON_NAME;TREES_PER_PEOPLE
ARBTUS-RIDGE;PISSARD PLUM;0.04
DOWNTOWN;RED MAPLE;0.00
DUNBAR-SOUTHLANDS;KWANZAN FLOWERING CHERRY;0.04
FAIRVIEW;RED MAPLE;0.00
GRANDVIEW-WOODLAND;KWANZAN FLOWERING CHERRY;0.02
...
```

Query 3: Top n barrios con mayor cantidad de especies distintas

Donde cada línea de la salida contenga, separados por ";" el nombre del barrio y el total de especies distintas de ese barrio.

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

El orden de impresión es descendente por el total de especies distintas y luego alfabético por nombre de barrio.

Solo se listan los primeros "n" barrios del resultado.

- 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 🇦🇷:
- ```
NEIGHBOURHOOD;COMMON_NAME_COUNT
9;268
8;259
15;212
12;211
11;203
```
- Salida de ejemplo para 🇨🇦:

## 72.42 Programación de Objetos Distribuidos

```
NEIGHBOURHOOD;COMMON_NAME_COUNT
DUNBAR-SOUTHLANDS;367
RENFREW-COLLINGWOOD;324
HASTINGS-SUNRISE;312
KITSILANO;312
KENSINGTON-CEDAR COTTAGE;302
```

Query 4: Pares de barrios que registran la misma cantidad de cientos de especies distintas

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 la misma cantidad de cientos de especies distintas de árboles en su barrio.

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

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

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

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

No se debe listar el grupo 0.

- Parámetros adicionales: Ninguno
- Ejemplo de invocación: ./query4 -Dcity=BUE -Daddresses='10.6.0.1:5701' -DinPath=. -DoutPath=.
- Salida de ejemplo para :

```
GROUP;NEIGHBOURHOOD A;NEIGHBOURHOOD B
200;10;11
200;10;12
...
100;3;6
100;5;6
```

- Salida de ejemplo para :

```
GROUP;NEIGHBOURHOOD A;NEIGHBOURHOOD B
300;DUNBAR-SOUTHLANDS;GRANDVIEW-WOODLAND
300;DUNBAR-SOUTHLANDS;HASTINGS-SUNRISE
...
200;VICTORIA-FRASERVIEW;WEST POINT GREY
200;WEST END;WEST POINT GREY
```

## 72.42 Programación de Objetos Distribuidos

Query 5: Pares de calles de un barrio X que registran la misma cantidad de decenas de árboles de una especie Y

Donde cada línea de la salida contenga separados por ; el nombre de la calle A y el nombre de la calle B donde las calles A y B están presentes en el barrio X y cuentan con la misma cantidad de decenas de árboles de una especie Y.

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

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

El orden de impresión es descendente por grupo y luego alfabético por el nombre de la primera calle del grupo. El orden de los pares dentro de cada grupo es alfabético por nombre de calle.

No se debe listar el grupo 0.

- Parámetros adicionales: neighbourhood (Un String no vacío con el nombre del barrio) y commonName (Un String no vacío con el nombre de la especie)

- Ejemplo de invocación: ./query5 -Dcity=BUE -Daddresses='10.6.0.1:5701' -DinPath=. -DoutPath=. -Dneighbourhood='KITSLANO' -DcommonName='NORWAY\_MAPLE'

- Salida de ejemplo para :

```
GROUP;STREET A;STREET B
60;Independencia Av;Perú
60;Independencia Av;Santa Fe Av.
60;Perú;Santa Fe Av.
...
10;Madero Eduardo Av.;cerrito
10;Viamonte;cerrito
```

- Salida de ejemplo para :

```
GROUP;STREET A;STREET B
150;W 11TH AV;W 13TH AV
20;COLLINGWOOD ST;W 10TH AV
...
10;MAPLE ST;W 12TH AV
10;MAPLE ST;W 16TH AV
10;W 12TH AV;W 16TH AV
```

## 72.42 Programación de Objetos Distribuidos

### Query 6: Especies que están presentes en todos los barrios (Sólo grupo de 5 integrantes)

Donde cada línea de la salida contenga, separados por ";" el nombre de la especie que tenga al menos un árbol en cada uno de los barrios presentes en el archivo CSV de barrios.

El orden de impresión es alfabético por nombre de especie.

Parámetros adicionales: Ninguno

Ejemplo de invocación: ./query6 -Dcity=BUE -Daddresses='10.6.0.1:5701' -DinPath=. -DoutPath=.

Salida de ejemplo para :

```
COMMON_NAME
Acacia visco
Acer buergerianum
...
Tipuana tipu
Ulmus procera
```

Salida de ejemplo para :

```
COMMON_NAME
AKEBONO FLOWERING CHERRY
AMERICAN SWEETGUM
...
WESTERN RED CEDAR
WILLOW OAK
```

#### 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>

## 72.42 Programación de Objetos Distribuidos

- 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 jobs 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 contenido:

- El **código fuente** de la aplicación:
  - Utilizando el arquitecto 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.

## 72.42 Programación de Objetos Distribuidos

- 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 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 13/10**

## 72.42 Programación de Objetos Distribuidos

- **Entrega del trabajo:** Estará disponible hasta el jueves 28/10 a las 23:59 la actividad "Entrega TPE 2" localizada en la sección Contenido / Evaluación / TPE 2. En la misma deberán cargar el paquete con el **código fuente** y el **documento**
- **El día miércoles 10/11 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" con la posibilidad de compartir pantalla. Los coloquios se llevarán a cabo en un aula virtual ya creada para cada grupo de Campus. 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 17/11.**
- **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
  - **Java**
    - java.nio.file.Files
    - java.text.DecimalFormat
    - java.math.RoundingMode