PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

IIC2525 - Sistemas Distribuidos - 2/2020

Tarea 1

Entrega: 9 Octubre 2020, 20:00 hrs.

Objetivo

El objetivo de esta tarea es utilizar el lenguaje `Go`, también conocido como `Golang`, en un caso práctico de *multithreading*. En esta entrega deben implementar tres operaciones de álgebra relacional usando el algoritmo `MapReduce` en `Go`.

Golang

Para comenzar con `Go`, es recomendable seguir las guías que ofrece la documentación oficial:

1. Instalación: Download and install
2. Tutoriales básicos: Get started with Go: Hello, World tutorial
3. Aprender Go: A Tour of Go

Es recomendable revisar al menos el capítulo de concurrencia de A Tour of Go: Concurrency para manejar las herramientas básicas de creación y comunicación de *threads* y herramientas de exclusión mutua.

Map Reduce

Es un modelo de programación enfocado en cómputo paralelo. Su objetivo es dado un conjunto de datos, asignarles un identificador y luego, según el indicador generar nuevos subconjuntos ejecutar alguna operación sobre ellos. La arquitectura del `MapReduce` está conformada por dos funciones/etapas: `map()` y `reduce()`, que ocurren de forma ordenada. Antes de ejecutar `reduce()`, deben terminar globalmente todos los hilos de `map()`.

1. `map()`: esta etapa etiqueta los valores de entrada, o sea, según los valores de cada dato, genera una llave o identificador.
2. `reduce()`: esta etapa reúne las tuplas (llave, valores) de acuerdo a la llave y ejecuta alguna operación sobre los valores.

Ejemplo de una implementación de `Mapreduce` en `Go`: A Simple MapReduce in Go.

Operaciones

Las operaciones a implementar son `Select`, `Projection` y `Group-Aggregate`, que son operaciones unarias o sobre una relación/tabla. A continuación, se especifican las operaciones para el contexto de esta tarea:

1. `Select`: dada una relación `r`, se deben seleccionar las tuplas que cumplan en su campo/columna `c` con un filtro que puede ser $<$, $>$, $\leq$, $\geq$, $=$ y $\neq$. No se pedirá manejar más de un filtro a la vez. Debe entregar la nueva relación `r'`. Esta operación es la única que se puede realizar usando sólo `map()` o `reduce()`.

2. `Projection`: dada una relación `r`, se debe mantener las columnas del conjunto `[a, b, ..., n]` de la tupla original. Debe entregar la nueva relación `r'` con los campos pedidos.

3. `Group-Aggregate`: dada una relación `r`, se entrega un campo de agrupación `a` y un campo que será operado, `b`. Las operaciones de agregación pueden ser `MIN`, `MAX`, `AVG` y `SUM`. Debe entregar la nueva relación `r'` con tuplas compuestas por cada `a` único acompañado de su valor agregado.

En la sección de Input se muestra cómo se entregan estos comandos al programa que deben crear.

Un punto de partida para pensar en cómo hacer la implementación: implementación de Álgebra Relacional con `MapReduce`

Datos

Los datos a utilizar son los casos acumulados totales por comuna de Covid-19. Fuente: Casos totales por comuna incremental.

Descripción de los campos

1. `Region` `<string>`: nombre de la región. Incluye tildes y espacios.
2. `Codigo region` `<int>`: código de la región o identificador.
3. `Comuna` `<string>`: nombre de la comuna. Incluye tildes y espacios.
4. `Codigo comuna` `<int>`: código de la comuna o identificador.
5. `Poblacion` `<float>`: cantidad de habitantes la zona.
6. `Fecha` `<time>`: fecha de la medición.
7. `Casos confirmados` `<int>`: número de casos confirmados acumulados hasta la fecha.

Es importante notar que pueden haber campos vacíos. Pueden ignorar las filas en estos casos o rellenar con algún valor predefinido.

Region	Codigo region	Comuna	Codigo comuna	Poblacion	Fecha	Casos confirmados
Arica y Parinacota	15	Arica	15101	247552.0	2020-03-30	6.0
Valparaíso	05	Panquehue	05704	7633.0	2020-03-30	0.0
Magallanes	12	Desconocido Magallanes			2020-09-04	15.0

Input

Argumentos del programa

El código recibirá como único argumento el número de *threads* a usar en cada etapa de `MapReduce`.

```
./program.go N
```

Ingreso de operaciones

El programa deberá recibir por la línea de comandos (`stdin`) la operación que se pide ejecutar. La primera línea siempre corresponderá al nombre de la operación y las siguientes, corresponderán a los detalles de ella.

A continuación se especifican las posibles entradas:

Select

General	Ejemplo
SELECT	SELECT
COL_NAME	Comuna
FILTER	!=
VALUE	Arica

1. `SELECT`: nombre fijo de la operación.
2. `COL_NAME`: nombre de la columna a filtrar.
3. `FILTER`: tipo de filtro. Valores posibles: $<$, $>$, $<=$, $>=$, $==$ y $!=$.
4. `VALUE`: valor para la comparación. Siempre sera un numero o texto.

Projection

General	Ejemplo
PROJECTION	PROJECTION
N	3
COL_NAME_0	Region
...	Comuna
COL__NAME_N-1	Casos confirmados

1. `PROJECTION`: nombre fijo de la operación.
2. `N`: número entero de columnas a mantener.
3. `COL_NAME_N`: nombre de la columnas a mantener.

Group-Aggregate

General	Ejemplo
GROUP	GROUP
COL_NAME_0	Region
AGGREGATE	AGGREGATE
COL_NAME_1	Casos confirmados
FUNCTION	AVG

1. `GROUP`: nombre fijo de la operación.
2. `COL_NAME_0` : columna para agrupar.
3. `AGGREGATE`: nombre fijo de la operación.
4. `COL_NAME_1`: columna para agregar.
5. `FUNCTION`: función de agregación. Valores posibles: `MIN`, `MAX`, `AVG`, `SUM`.

Output

El programa debe devolver un archivo csv con el resultado de la operación.

Informe

Se espera que describan cómo implementaron cada operación de acuerdo al modelo `MapReduce`. Deben detallar qué hace la etapa `map()` en relación a la elección del identificador y qué ejecuta la etapa `reduce()`. Por último, deben explicar de qué forma se distribuyen estas etapas entre los *threads*. Los aspectos formales del informe incluyen incluir el nombre y entregarlo en formato PDF.

Desglose de puntaje

Código (8 puntos)

- Código de `Select`: 1.6 puntos en total.
- Código de `Projection`: 1.6 puntos `map()` y 1.6 puntos `reduce()`
- Código de `Group-Aggregate`: 1.6 puntos `map()` y 1.6 puntos `reduce()`

Nota: Si el código no compila, no se puede evaluar esta parte de la tarea.

Informe (4 puntos)

- Formalidades: 0.1 puntos por el informe en PDF, incluir nombre y cumplir el formato de entrega.
- Descripción de `Select`: 0.78 puntos en total.
- Descripción de `Projection`: 0.78 puntos `map()` y 0.78 puntos `reduce()`
- Descripción de `Group-Aggregate`: 0.78 puntos `map()` y 0.78 puntos `reduce()`

Formato de entrega

El informe debe ser entregado en formato PDF junto al código en un .zip o .rar en canvas. El nombre del archivo comprimido debe ser `NúmeroDeAlumno_IIC2523-2_Tarea1`.

Integridad académica

Los alumnos de la Escuela de Ingeniería de la Pontificia Universidad Católica de Chile deben mantener un comportamiento acorde a la Declaración de Principios de la Universidad. En particular, se espera que mantengan altos estándares de honestidad académica. Cualquier acto deshonesto o fraude académico está prohibido; los alumnos que incurran en este tipo de acciones se exponen a un Procedimiento Sumario. Es responsabilidad de cada alumno conocer y respetar el documento sobre Integridad Académica publicado por la Dirección de Docencia de la Escuela de Ingeniería.

Específicamente, para los cursos del Departamento de Ciencia de la Computación, rige obligatoriamente la siguiente política de integridad académica. Todo trabajo presentado por un alumno para los efectos de la evaluación de un curso debe ser hecho individualmente por el alumno, sin apoyo en material de terceros. Por "trabajo" se entiende en general las interrogaciones escritas, las tareas de programación u otras, los trabajos de laboratorio, los proyectos, el examen, entre otros. Si un alumno copia un trabajo, obtendrá nota final 1,1 en el curso y se solicitará a la Dirección de Docencia de la Escuela de Ingeniería que no le permita retirar el curso de la carga académica semestral. Por "copia" se entiende

incluir en el trabajo presentado como propio partes hechas por otra persona.

Obviamente, está permitido usar material disponible públicamente, por ejemplo, libros o contenidos tomados de Internet, siempre y cuando se incluya la referencia correspondiente.

Lo anterior se entiende como complemento al Reglamento del Alumno de la Pontificia Universidad Católica de Chile. Por ello, es posible pedir a la Universidad la aplicación de sanciones adicionales especificadas en dicho reglamento.