Sistemas Operativos 1/2023
Laboratorio 1

Profesores:
Marcela Rivera (marcela.rivera.c@usach.cl)

I. Objetivos Generales

Este laboratorio tiene como objetivo resolver un problema de manera secuencial, el cual posteriormente debe ser paralelizado utilizando procesos. La aplicación debe ser escrita en el lenguaje de programación C sobre sistema operativo Linux.

II. Objetivos Específicos

1. Conocer y usar las funcionalidades de `getopt()` como método de recepción de parámetros de entradas.
2. Conocer MapReduce y realizar una implementación preliminar.
3. Construir funciones de lectura y escritura de archivos .csv usando `fopen()`, `fread()`, y `fwrite()`.
4. Practicar técnicas de documentación de programas.
5. Conocer y practicar uso de makefile para compilación de programas.
6. Modular el código a través del uso de funciones.

III. Antecedentes

III.A. Procesamiento de datos

El procesamiento de grandes volúmenes de datos cada vez se hace más presente en la vida cotidiana y negocio de las empresas. Cuando se habla de BigData, se hace alusión justamente al trabajo de grandes conjuntos de datos o combinaciones de conjuntos de datos cuyo tamaño (volumen), complejidad (variabilidad) y velocidad de crecimiento (velocidad) dificultan su captura, gestión, procesamiento o análisis mediante tecnologías y herramientas convencionales, tales como bases de datos relacionales y estadísticas convencionales o paquetes de visualización, dentro del tiempo necesario para que sean útiles.

Dependiendo del tamaño de un conjunto de datos, podemos enfrentarnos a mayor dificultad de procesamiento, por lo cual se hace necesario encontrar diferentes maneras de gestionar los datos y procesarlos.

III.B. MapReduce

MapReduce nace en Google, aun cuando esta compañía, tiene otros métodos para realizar cálculos que procesan gran cantidad de datos en bruto. La razón es que apesar de contar con diferentes metodologías y tener que realizar cálculos sencillos los datos de entrada son muy grandes y los cálculos se han distribuido a través de cientos o miles de máquinas con el fin de terminar en un tiempo razonable, lo cual requiere distribución de datos tarea muy importante que no puede presentar errores porque de lo contrario se puede obtener una gran cantidad de errores lo cual afectará los resultados y demandará mayor tiempo para corregirlos.

Como reacción a este problema, ingenieros de Google, diseñaron una nueva abstracción que permite expresar los simples cálculos a realizar, pero oculta los detalles de paralelización, tolerancia a fallos, la distribución de datos y balanceo de carga en una API.

El modelo funcional MapReduce permite paralelizar grandes cálculos fácilmente y usar nueva ejecución como el mecanismo principal para la tolerancia a fallos. Para conseguir esto, MapReduce se descompone en tres principales partes: coordinador encargado de crear las unidades de trabajo que harán los cálculos, una etapa de mapeo donde se busca generar un conjunto de par llave,valor y finalmente una etapa de reduce encargada de operar los valores asociados a una misma llave. De forma gráfica podemos ver la figura 1 donde está la organización general de este método.

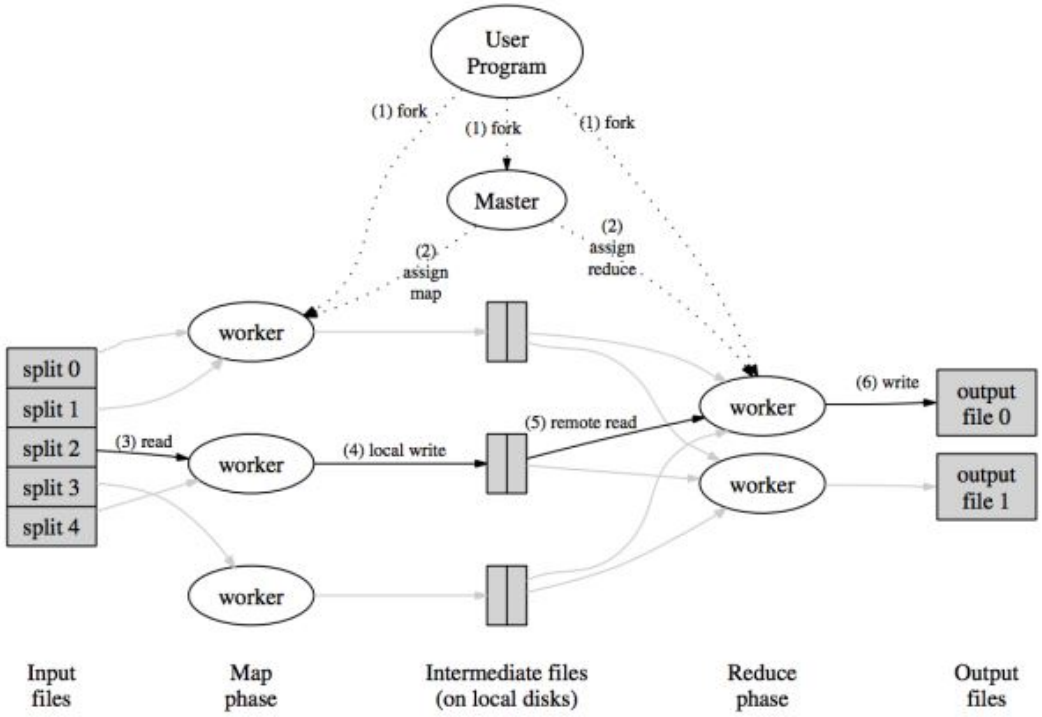

Figure 1. MapReduce

Para comprender mejor las etapas de mapeo y reduce se puede observar la figura 2. Donde básicamente se toma como llave la variable cust_id y por lo tanto la etapa de mapeo agrupará para cada llave todos los valores de amount asociados a la respectiva llave. La etapa de reduce recibe como entrada este par llave,valores y se encargará únicamente de sumar los valores asociados a cada llave.

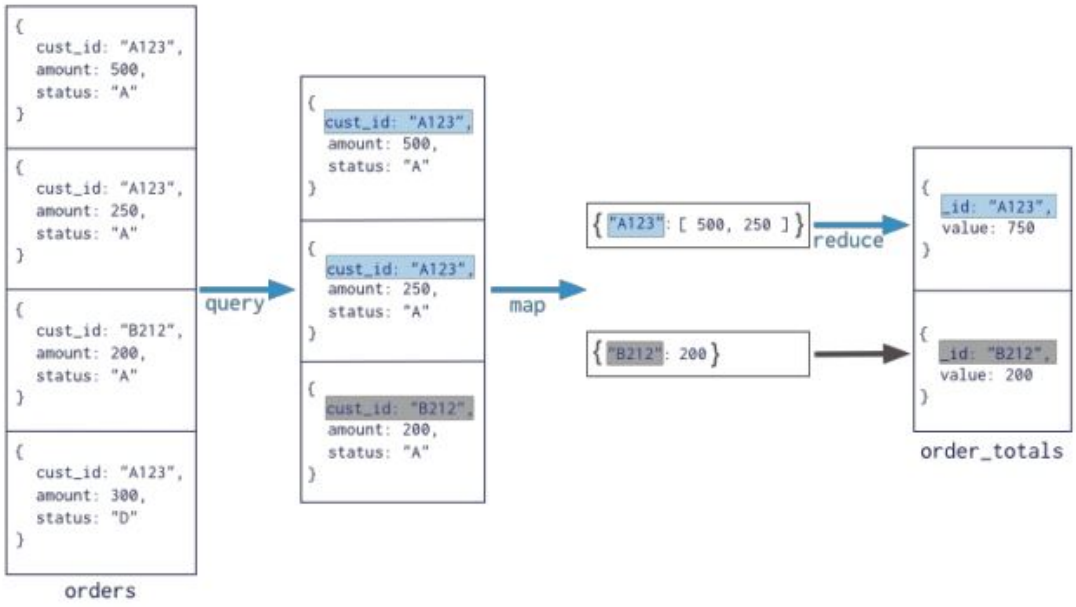

Figure 2. ejemplo práctico de MapReduce

IV. Enunciado

En este laboratorio, se pide que implemente una solución capaz de leer un gran conjunto de datos, específicamente una base de datos chilena perteneciente a la municipalidad de Calbuco donde se almacena un registro de los permisos de circulación correspondientes al año 2022. El objetivo es simular MapReduce creado por Google. En esta primera parte, solo se requiere que sea capaz de procesar los datos, realizando operaciones sencillas para obtener métricas que puedan aportar información útil.

IV.A. Cálculo de propiedades

Para este lab, las propiedades que se calcularán para cada columna son:

1. Indicar tasación total para cada grupo de vehículo
2. Indicar el total del valor pagado para cada grupo de vehículo.
3. Indicar para cada grupo de vehículo la cantidad de vehículos con dos y cuatro puertas.

IV.B. Lógica de solución

V. Enunciado

Se solicitan tres programas. Uno para coordinar y distribuir los datos de entrada a la etapa de mapeo, además de recibirle a la etapa de reduce las respectivas salidas. El segundo código, se encargará de representar la etapa de mapeo, la cual calculará las propiedades descritas previamente. Finalmente el tercer código representa la tercera etapa, correspondiente a Reduce.

V.A. Coordinador

El primer programa debe realizar la tarea de distribuir los datos a la etapa de mapeo, es decir, leerá línea a línea el archivo de entrada para luego entregarle a la función de mapeo la respectiva entrada.

Por último, se encargará de recibirle a la etapa de reduce su respectiva salida para luego escribir un archivo de texto el cual tendrá los resultados finales.

La salida debe tener el formato:

```
Total de tasaciones para Vehiculo Liviano: 23005623
Total de tasaciones para carga: 35889448
Total de tasaciones para Transporte Publico: 26654456
Total de valor pagado para Vehiculo Liviano: 10223665
Total de valor pagado para carga: 15632658
Total de valor pagado para Transporte Publico: 9654456
Total de vehiculos con 2 puertas para Vehiculos Livianos: 1000
Total de vehiculos con 4 puertas para Vehiculos Livianos: 3000
Total de vehiculos con 2 puertas para carga: 100
Total de vehiculos con 4 puertas para carga: 0
Total de vehiculos con 2 puertas para carga: 100
Total de vehiculos con 4 puertas para carga: 0
Total de vehiculos con 2 puertas para Transporte Publico: 0
Total de vehiculos con 4 puertas para Transporte Publico: 50
Total de vehiculos con 5 puertas para Transporte Publico: 10
```

V.B. Programa Mapeo

El segundo programa es el encargado de realizar las operaciones de mapeo. En este caso tenemos tres objetivos por lo que el mapeo abordará estos tres casos a través de tres funciones (una para cada propiedad).

Para el caso de la primera propiedad, el mapeo se encargará de considerar tres llaves principales: Vehículo Liviano, carga y Transporte Publico. Para cada llave, almacenará un listado de las tasaciones que le corresponden a cada grupo, ejemplo:

- Vehículo liviano: 4550702, 2292467, 1322987, ...
- carga: 1, 2,3, ...
- Transporte Publico: 1,1,.....

Esto mismo se repetirá para obtener el valor pagado por grupo y la cantidad de puertas. La idea es que la salida sea un arreglo o estructura que contenga los valores asociadas a las llaves.

V.C. Programa Reduce

Este código se encargará de tener las funciones de reducción, las cuáles en este caso son bastante simples.

1. Para el caso de las tasaciones y valores pagados, la función se encargará únicamente de sumar todos los valores asociados a la respectiva llave.
2. Para el caso de la cantidad de puertas, esta función debe contar la cantidad de repeticiones que hay para cada cantidad de puertas asociadas a un grupo de vehículos, tal como se indica en el ejemplo de archivo de salida mencionado previamente.

El programa se ejecutará usando los siguientes argumentos (ejemplo):

```
$ ./lab1 -i permiso-de-circulacion-2022.csv -c 10000 -d
```

- **-i**: nombre de archivo de entrada
- **-c**: cantidad de líneas de archivo de entrada.
- **-d**: bandera o flag que permite indicar si se quiere ver por consola el resultado final de las métricas.

Ejemplo de compilación y ejecución:

```
>> make
>> ./lab1 -i permiso-de-circulacion-2022.csv.csv -c 10000 -d
```

Como requerimientos no funcionales, se exige lo siguiente:

- Debe funcionar en sistemas operativos con kernel Linux.
- Debe ser implementado en lenguaje de programación C.
- Se debe utilizar un archivo Makefile para su compilación.
- Realizar el programa utilizando buenas prácticas, dado que este laboratorio no contiene manual de usuario ni informe, es necesario que todo esté debidamente comentado.
- Que el programa principal esté desacoplado, es decir, que se desarrollen las funciones correspondientes en otro archivo .c para mayor entendimiento de la ejecución.

VI. Entregables

El laboratorio es en parejas y se descontará 1 punto por día de atraso. Debe subir en un archivo comprimido a usachvirtual los siguientes entregables:

- `Makefile`: Archivo para make que compila el programa. De no incluirlo, el trabajo será evaluado con la calificación mínima.
- `coordinador.c`: archivo con el código del proceso padre. Puede incluír otros archivos fuente. Recuerde que todas las funciones deben estar comentadas, explicadas de forma entendible especificando sus entradas, funcionamiento y salida. Si una función no está explicada se bajará puntaje.
- `mapeo.c`: archivo con el código para el cálculo de propiedades. Puede incluír otros archivos fuente. Recuerde que todas las funciones deben estar comentadas, explicadas de forma entendible especificando sus entradas, funcionamiento y salida. Si una función no está explicada se bajará puntaje.
- `reduce.c`: archivo con el código para el cálculo de propiedades. Puede incluír otros archivos fuente. Recuerde que todas las funciones deben estar comentadas, explicadas de forma entendible especificando sus entradas, funcionamiento y salida. Si una función no está explicada se bajará puntaje.
- Trabajos con códigos que hayan sido copiados de un trabajo de otro grupo serán calificados con la nota mínima.

El archivo comprimido debe llamarse: RUTESTUDIANTE1_RUTESTUDIANTE2.zip
Ejemplo: 19689333k_186593220.zip
NOTA: los laboratorios son en parejas, las cuales deben ser de la misma sección. De lo contrario no se revisarán laboratorios.

VII. Fecha de entrega

28 de Abril antes de las 23:59 horas.