Universidad Simón Bolívar.
Departamento de Computación y Tecnología de la Información.
Asignatura: Sistemas de Operación.
Profesora: Yudith Cardinale
Estudiantes: Fernando Pérez, carné: 12-11152
Leslie Rodrigues, carné: 10-10613
Sartenejas, mayo de 2015.

Informe detalle de una implementación simplificada del modelo de Map-Reduce.

Introducción.

Se desea implementar un algoritmo simplificado del modelo Map-Reduce para el procesamiento en paralelo de gran cantidad de datos en el lenguaje de programación C. El algoritmo se basa en dos funciones: `Map()` y `Reduce()`, la primera se encarga de aplicar un filtro a los datos, y la segunda se encarga de procesar, de ese filtro, elementos en común; bajo algún criterio y devolver el resultado. Además, se requiere el uso de hilos y procesos en la implementación.

Este documento muestra detalles de la implementación sobre este modelo de datos, así como tablas comparativas de tiempos entre los hilos y procesos, también, análisis y detalles sobre la implementación.

Soluciones

Para la implementación de este modelo, se utilizan dos herramientas fundamentales en la programación concurrente, los hilos y los procesos, específicamente, los hilos a nivel de Kernel. Por lo tanto, la implementación del problema se hace en dos archivos main. Sin embargo, se utiliza una lista enlazada, también implementada, con sus operaciones, además de otras herramientas del sistema de operación tales como semáforos. El hilo principal de la implementación es la lectura de las líneas, luego, se ejecuta map(), se procesa la información obtenida y se inserta en la lista enlazada, y por último, se ejecuta reduce() para determinar amigos en común y escribir en el archivo.

La implementación usando hilos, se basa en el manejo de listas y apuntadores globales para que los mismos puedan manipular estas estructuras y esta ser la forma de comunicación con el proceso padre. El padre ejecuta una serie de llamadas a `pthread_create()` para crear los distintos hilos, estos ejecutan `map()` y se tiene un problema de exclusión mutua sobre la lectura de líneas e inserción en la estructura, por lo que se incluye un semáforo de exclusión mutua. Los hijos, insertan las tuplas en la lista enlazada y terminan su ejecución. Luego, el proceso padre vuelve a ejecutar `pthread_create()` y esta vez, los hilos ejecutan

`reduce()`. Los mismos, escriben en el archivo de salida los resultados, otra vez, se utiliza un semáforo de exclusión mutua.

La implementación usando procesos, y sin usar *pipes* o canales de comunicación entre procesos padres e hijos, se basa en el almacenado de resultados en archivos, es decir, el proceso padre ejecuta la llamada `fork()` y luego, a los hijos, dependiendo de la cantidad de líneas del archivo, ejecutan `map()`. Luego, los hijos almacenan los resultados en un archivo de nombre PID.txt donde PID es el *ID* del proceso, de estos archivos, el proceso padre procede a crear una lista enlazada con los distintos pares de personas con sus amigos. Después, vuelve a ejecutar una llamada `fork()`, esta vez, para que los procesos hijos ejecuten `reduce()`. Se repite el mismo estilo de comunicación y por último, el padre abre cada archivo y en el archivo de salida especificado, escribe los resultados del algoritmo.

Compilación y ejecución.

Para la compilación, sólo debe realizarse desde terminal y en el directorio del proyecto, la instrucción `make`.

Para la ejecución, debe ejecutarse en el mismo terminal, las llamadas:

```
./friendfindP [-n #_de_procesos] <entrada> <salida>
./friendfindH [-n #_de_hilos] <entrada> <salida>
```

Para procesos e hilos, respectivamente, donde:

- -n: Número de procesos o hilos que se ejecutarán sobre las funciones map y reduce. Esta opción es opcional, si no se especifica, el programa se ejecuta con un solo hilo/proceso.
- <entrada>: Nombre del archivo de entrada
- <salida>: Nombre del archivo de salida

Resultados

Se mide el tiempo de ejecución de cada implementación para mostrar la ventajas en procesamiento de datos para hilos y para procesos. El archivo de ejemplo se encuentra en el archivo comprimido para la entrega.