Pontificia Universidad JAVERIANA Bogotá

Sistemas Operativos (SIU4085)

Proyecto 1- Febrero 2015

25%

Análisis Concurrente de un Log

1. Objetivos del Proyecto

a. Resolver un problema concurrente utilizando el concepto de procesos en Linux y el mecanismo de memoria compartida. Los procesos se crearán utilizando la **llamada al sistema *fork().***
b. Resolver el mismo problema utilizando los hilos de la librería POSIX.
c. Comparar el desempeño de una solución secuencial con las dos soluciones concurrentes implementadas.

2. Descripción General

Como se mencionó en la clase, una de las funciones de los SOP es llevar estadísticas sobre el uso de los recursos del sistema por parte de los usuarios y procesos. Estas estadísticas se escriben en archivos denominados *logs*. El objetivo de este proyecto es realizar un programa, que de forma concurrente nos permita analizar un log de una computadora con múltiples procesadores.

Una vez que lea el *log*, el programa ofrecerá al usuario un menú para que éste realice algunas consultas sobre los datos. Se implementarán tres programas: un programa secuencial, un programa concurrente usando procesos y un programa concurrente usando hilos. A continuación se presentan detalles de la solución.

2.1 Entradas

Se realizarán tres programas ejecutables **analogs**, **analogp** y **analogh.** El primero será el programa secuencial y los dos últimos implementarán la concurrencia con procesos e hilos respectivamente.

Cada programa recibirá como parámetros el archivo donde se encuentra el *log* que se va a analizar y su tamaño. Adicionalmente, los programas concurrentes obtendrán como argumento de entrada el número de procesos/hilos concurrentes que se deben crear para analizar el log.

Los comandos se invocarán de la siguiente forma:

Invocación del Proceso desde el Shell	Significado de los argumentos que reciben los procesos
analogs logfile tamlog	**logfile**: archivo que contiene el log que se va a analizar de forma concurrente. En la sección 2.3 se describe el formato del archivo. ***tamlog:*** es el número de líneas que posee el log**.** **nprocesos**: número de procesos que se crearán para revisar el log. Debe ser >1. **nhilos**: número de hilos que se crearán para revisar el log. Debe ser >1.
analogp logfile tamlog nprocesos	
analogh logfile tamlog nhilos	

2.2 Salidas

Una vez invocado cualquiera de los programas, le presentarán un menú al usuario, para que realice alguna de las siguientes consultas:

1. Número de procesos que se ejecutó únicamente en un procesador.
2. Numero de procesos que se ejecutó en 64 o más procesadores.
3. Qué proceso utilizó más CPU (se debe imprimir el identificador del proceso y su uso de CPU y memoria)
4. Cantidad de procesos Interactivos encontrados
5. Número de procesos cuya ejecución fue cancelada por el administrador
6. Salir del sistema

Para cada una de estas alternativas se le dará una respuesta al usuario, así como **el tiempo total** que tardó el programa en obtener la respuesta. El programa termina cuando el usuario introduce la opción 6 del menú.
Para medir el tiempo que se tarda la realización de una consulta concurrente pueden usar gettimeofday .En el enlace http://www.cs.loyola.edu/~jglenn/702/S2008/Projects/P3/time.html encontrarán un ejemplo del uso de esta rutina para medir el tiempo que tarda en ejecutarse un trozo de código.

2.3 Formato del Archivo

A continuación presentamos un ejemplo de un log con 12 líneas y 18 columnas o campos. Cada línea contiene información de un proceso. El log fue pre-procesado y en algunos campos fue colocado el valor -1, ya que los mismos no eran útiles para la persona que realizó el pre-procesamiento. Los estudiantes no deben preocuparse por estos campos. Pueden almacenar sólo aquellos campos relacionados con las consultas. Este log es un extracto de

trazas encontradas en: **http://www.cs.huji.ac.il/labs/parallel/workload/logs.html** **(Logs of Real Parallel Workloads from Production Systems)**

1	0	224904	37349	128	37349	-1	-1	-1	-1	1	7	-1	-1	29	2	-1 -1
2	4751	257510	43349	128	42924	-1	-1	-1	-1	1	7	-1	-1	29	2	-1 -1
3	91769	213864	22	128	-1	-1	-1	-1	-1	1	7	-1	-1	29	2	-1 -1
4	138658	86135	4138	8	4138	-1	-1	-1	-1	1	3	-1	-1	28	2	-1 -1
5	138682	86354	58	4	-1	-1	-1	-1	-1	1	1	-1	-1	2	2	-1 -1
6	140276	84762	681	1	-1	-1	-1	-1	-1	1	2	-1	-1	2	2	-1 -1
7	141888	265755	94	256	60.92	-1	-1	-1	-1	1	18	-1	-1	17	2	-1 -1
8	141902	265840	60	256	52.10	-1	-1	-1	-1	1	18	-1	-1	17	2	-1 -1
9	141918	265888	74	256	67.95	-1	-1	-1	-1	1	18	-1	-1	17	2	-1 -1
10	141934	265952	90	256	75.78	-1	-1	-1	-1	1	18	-1	-1	17	2	-1 -1
11	143227	264753	86	256	79.24	-1	-1	-1	-1	1	18	-1	-1	17	2	-1 -1
12	146537	261534	255	256	246.04	-1	-1	-1	-1	1	18	-1	-1	17	2	-1 -1

A continuación se coloca el significado de los distintos campos en cada línea. Las consultas que deberán implementar tienen que ver sólo con algunos de los campos:

1. **Número de Job o Trabajo** – contador comienza desde 1 y es el identificador del job.
2. **Submit Time** – en segundos (entero). El primer job se introduce en el sistema en el instante 0, a partir de dicho instante, se va colocando el tiempo en el que se van introduciendo los otros trabajos.
3. **Tiempo de Espera**—se mide en segundos. Es la diferencia entre el tiempo en el que se introduce el job al sistema y el tiempo en el que efectivamente comienza su ejecución.
4. **Tiempo de Ejecución** – en segundos. Tiempo total que duró la ejecución del trabajo o job en el sistema.
5. **Número de Procesadores Asignados al Job** (entero). Número de procesadores que usa el trabajo en su ejecución.
6. **Tiempo Promedio de CPU Utilizado** – número real. Tiempo en segundos que utilizó un job o trabajo ejecutándose en modo usuario y en modo sistema. Este es un promedio sobre todos los procesadores utilizados (se divide tiempo total de CPU entre el número de procesadores).
7. **Memoria Usada por el Trabajo** -- en kilobytes. De nuevo, es el promedio por procesador.
8. **Número de Procesadores Solicitados por el Job**.
9. **Tiempo Solicitado**. Esta es una estimación del tiempo total que va a utilizar el trabajo en su ejecución. Cuando un trabajo hace este requerimiento, el tiempo se divide entre el total de procesadores asignados.
10. **Memoria Solicitada** (kilobytes por procesador).
11. **Status** es igual a 1 si el trabajo terminó, 0 si tuvo una falla antes de terminar, y 5 si fue cancelado (terminado) por el administrador antes de su terminación normal. Si se está haciendo checkpointing or swapping, es posible encontrar otros valores; la persona que realizó un pre-procesamiento de los logs, colocó -1 a estos valores especiales.
12. **Identificador del Usuario** – es un número natural que va entre 1 y el número total de usuarios del sistema.

13. **Identificador de Grupo** – es un número natural que va entre 1 y el número total de grupos del sistema.

14. **Número que Identifica a la Aplicación** – es un número natural que va entre 1 y el número total de diferentes aplicaciones ejecutadas durante el intervalo de medición.

15. **Número de Cola**—es un número natural entre 1 y 36. (son colas para procesos batch, en el caso de procesos interactivos, el número debe ser 0).

16. **Numero de Partición** -- es un número natural que va entre 1 y el número de particiones del Sistema. El multiprocesador puede estar dividido en diferentes grupos de clusters, a estos grupos se les llama particiones.

17. **Job Anterior**—es el identificador de un trabajo que se introdujo previemente y de cuya terminación depende el job actual (si tal trabajo no existe se coloca -1).

18. **Tiempo de Reflexion (Think Time) del "Job Anterior"** – segundos que transcurrieron entre la culminación del "Job Anterior" y la introducción de este.

Ejemplos:

Según el log, el job número 5 esperó 86354 segundos antes de ejecutarse. Su ejecución duró 58 segundos y se realizó en 4 procesadores; el trabajo culminó y fue ejecutado por el usuario identificado con el número 1. Se trata de un proceso batch que fue colocado en la cola número 2.

Suponiendo que el archivo donde se encuentra el log se llama *log2-2-2015* y quiero hacer su procesamiento con dos procesos, debería invocar el programa de la siguiente forma:

$ ./analogp *log2-2-2015* 12 2

2.4 División de Trabajo y Comunicación entre Procesos e Hilos

A continuación se coloca un pseudo-algoritmo que describe las principales tareas del programa concurrente con **Procesos**:

0.- Leer los argumentos y validarlos
1. ***Crear la memoria compartida*** *donde se colocará la información del log.*
2. ***Leer*** *el log completo y colocarlo en la memoria compartida o en una estructura de datos (en la carpeta Proyecto/Proyecto I/Memoria Compartida puede ver ejemplos de estos dos enfoques)*
3. ***Mostrar el menú al usuario***
4. Mientras la opción del usuario sea distinta de 6 hacer:
4.1 ***Crear el número de procesos concurrentes*** *según los argumentos de entrada, asignarles una porción del log e informarles sobre la búsqueda que deben realizar. La división del log entre procesos debe ser lo más equitativa posible.*
4.2 ***Cada proceso buscará en su porción*** *(como se indica en la Figura 1)* ***y escribirá un resultado en la memoria compartida****. Una vez realizado este trabajo, el proceso hijo termina su ejecución.*

4.3 Una vez que todos los hijos han culminado, ***el padre lee los datos escritos en la memoria compartida (Figura 2)****, los totaliza y muestra el resultado de la consulta al usuario. También devuelve el tiempo que tomó realizar la operación. El tiempo se tomará únicamente para todas las actividades del paso 4.*

5. Como la opción del usuario fue salir, el programa termina.

Ejemplo:

Suponga que su programa está analizando un log de 12 entradas, como se muestra en la Figura 1, con 3 procesos concurrentes, y que el usuario pidió la siguiente consulta:

Número de procesos que se ejecutó únicamente en un procesador.

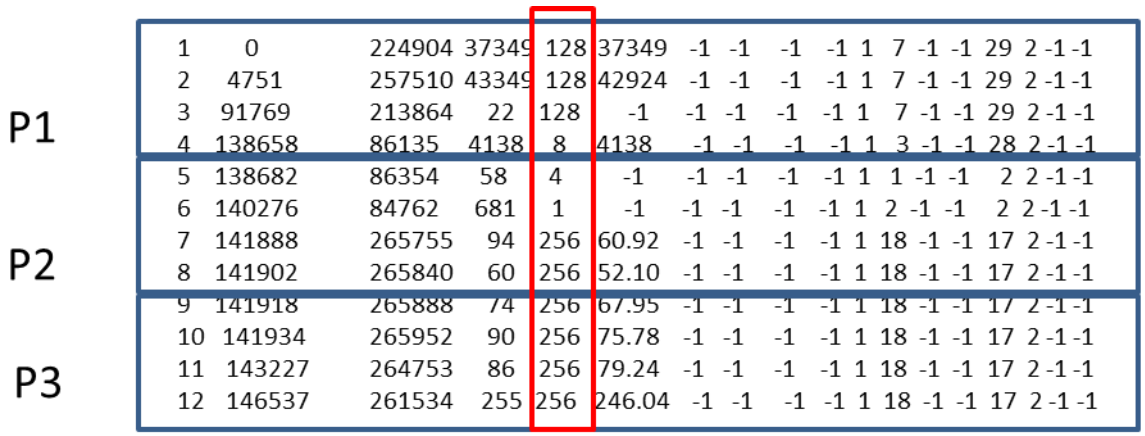

Figura 1: División del log entre tres procesos

En este caso, cada proceso va a revisar 4 entradas de la estructura de datos donde ud, almacenó las diferentes líneas del log, siendo el campo a revisar el número 5 (columna roja). Pueden observar que los procesos 1 y 3 no tienen en sus líneas trabajos ejecutados en 1 procesador. El proceso 2 tiene 1 job. Una vez revisado el log, cada proceso imprime su resultado en memoria compartida como lo indica la Figura 2. Toda la información del log también pudiera estar en la memoria compartida.

Después de que un hijo escribe el resultado termina su ejecución. El padre suma cada posición de este arreglo y devuelve los resultados al usuario.

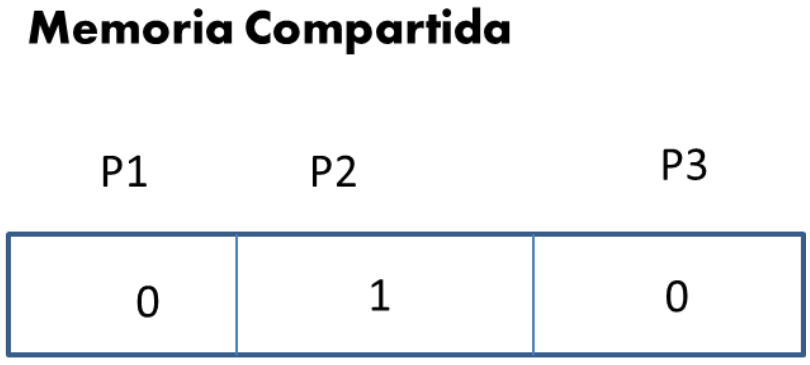

Figura 2: Resultado de cada proceso hijo, según la consulta realizada.

Observaciones importantes:

1. En el caso de los hilos el algoritmo es idéntico pero no es necesario crear en forma explícita una región de memoria compartida. También se miden únicamente las actividades del paso 4.
2. En el caso del algoritmo secuencial, también se mide el tiempo de realizar la consulta y responder al usuario (paso 4); en este caso no se tiene el *overhead* asociado a la creación y destrucción de procesos o hilos.
3. Note que para cada consulta se crean procesos (o hilos nuevos). Los procesos realizan el trabajo y mueren. Cuidado con los zombies!!

Detalles de Implementación

La implementación se realizará en lenguaje C (ansi C). Para la creación de los hilos se utilizará la librería POSIX. **Deben seguir en su código las recomendaciones de la Guía de Programación en C publicada en UVirtual (archivo estiloC.pdf, en la carpeta de información sobre el curso). Se recomienda el uso del compilador gcc, si Ud. usa otro compilador asegúrese de que su código esté en ansi C siguiendo las recomendaciones de la guía de estilo.**

Deben hacer un makefile para crear los tres ejecutables. También se colocará un ejemplo de makefile como parte del material del proyecto.

3.Evaluación del Rendimiento de los Comandos

Ud. deberá comparar el tiempo que se tarda la solución secuencial, la solución con hilos y la solución con procesos para distintos tamaños de matrices.

La medición se realizará de la siguiente forma:

1. Escoja cualquiera de las consultas que realiza el usuario y solicítela en los tres programas con el mismo tamaño de log (100 líneas). Anote los tiempos de la ejecución.
2. Repita el paso anterior aumentando el log a 1000, 5000 y 10000 líneas. Anote los tiempos que tarda la consulta en cada uno de los programas. En la carpeta donde encontró el proyecto, se colocó un archivo tar.gz (Datos para probar), donde hay logs con los 4 tamaños mencionados. Asegúrese de eliminar de cada archivo las primeras 70 líneas que corresponden a comentarios. Cualquier problema de memoria que tenga con el tamaño de los archivos, debe informarlo al profesor.

Una vez realizadas las medidas debe colocar los resultados en un informe que contenga no más de 5 páginas y los siguientes puntos:

- Identificación, título, etc.
- Descripción del problema planteado
- Tablas de datos (cada solución y el tiempo que le tomó al programa realizar la consulta para distintos tamaños de archivos)
- Gráficos donde se pueda observar el comportamiento de las soluciones.
- Comente y analice los resultados obtenidos. Los resultados son consistentes con los conceptos de la clase teórica? Qué puede concluir de las soluciones concurrentes con respecto a la secuencial. Cuál de las dos soluciones concurrentes es más eficiente?

Observaciones Adicionales

- El proyecto lo deben realizar en grupos de como máximo dos estudiantes. Lo deben entregar el jueves de la semana 9 (26 de marzo) antes de las 12 m por UVirtual. La entrega consiste de los códigos fuentes, el makefile y el informe con la evaluación de rendimiento. Estos archivos deben empaquetarse en un único archivo en formato **rar, tar.gz o zip**, el cual se sube a la plataforma . Este único archivo debe tener los apellidos de los integrantes del equipo de la siguiente forma:

  Apellidos(Estudiante1)_Apellidos(Estudiante2).tar.gz (.rar o .zip)

  Ejemplo: EspindolaBuitrago_SanchezCordoba.zip

- La sustentación se realizará en las horas de práctica. Sólo podrán sustentar aquellos que hayan subido la documentación a UVirtual.

- Deben validar llamadas al sistema, parámetros de entrada y utilizar correctamente los recursos que solicita, es decir, no dejar procesos huérfanos o zombies, archivos abiertos, borrar archivos temporales, etc.

Nota: Cualquier duda sobre el enunciado del proyecto debe consultarla con la profesora en forma oportuna. La comprensión del problema y su correcta implementación, según lo indica el enunciado, es parte de lo que se está evaluando.

Suerte

Prof. Mariela Curiel