Pontificia Universidad JAVERIANA Bogotá

Sistemas Operativos (SIU4085)

Proyecto 1- Febrero 2016

20%

Un servidor Concurrente

1. *Objetivos del Proyecto*

a. Resolver un problema concurrente utilizando hilos o procesos.
b. Estudiar el desempeño de la solución propuesta variando algunos parámetros de entrada.
c. Comparar el desempeño de una solución secuencial con la solución concurrente.

2. *Descripción General*

La idea del proyecto es diseñar dos tipos de servidores: un servidor secuencial y un servidor concurrente y comparar el desempeño de ambos realizando varios tipos de consultas a un log.

2.1 Procesos

Los estudiantes deben desarrollar dos programas ejecutables: **serversec**, **serverconc.** El primero será el programa secuencial y el último implementará la concurrencia utilizando procesos o hilos (escoja sólo una de las dos alternativas). La función de ambos programas será idéntica: leer varios tipos de consultas desde un archivo y ejecutarlas contra un ***archivo log***. El formato del archivo de log se describe en 2.4.1. El formato del archivo que contiene las consultas o tareas se describe 2.4.2. A continuación se presentan los detalles de ambos procesos:

El servidor secuencial realizará una tras otra las consultas que se indiquen en el archivo de consultas. No se apoyará de entidades concurrentes. La salida de cada consulta se escribirá en un archivo (***output file***). Cuando termine la ejecución de todas las consultas el servidor imprime el tiempo total que gastó ejecutando consultas y el número total de consultas (válidas y erróneas) ejecutadas.

El servidor concurrente: reparte las diferentes consultas del archivo entre distintos hilos o procesos (tareas concurrentes). Las tareas concurrentes hacen la consulta y escriben el resultado en un archivo temporal o en la memoria. El padre o creador de las tareas concurrentes se encarga de combinar los resultados de los distintos procesos o hilos y producir el ***output file*** con todos los resultados. Esta actividad se puede hacer de distintas formas. A continuación se presentan dos posibles alternativas de diseño para realizar las consultas (no son las únicas):

- A medida que se va leyendo el archivo se va creando una tarea para cada consulta (o grupo de consultas), tal y como se muestra en las figuras I y II para las consultas de la

sección 2.4.2. Los hijos al ser creados, heredan la consulta que van a realizar porque se encuentra en una variable del padre. Cada tarea escribe en un archivo por separado el resultado de su consulta y el padre se encarga de colocar todas las salidas en un archivo, ordenarlo, borrar archivos temporales, etc. **Si se usan procesos, todos los procesos pudieran estar escribiendo en un solo archivo**. No obstante, aunque todos usen el mismo archivo, no estará necesariamente ordenado. Como aquí se crea un proceso por consulta, el padre tiene que encargarse de matar procesos zombies de forma periódica.

- Se crean N entidades concurrentes, el padre divide el archivo de consultas entre estas N entidades. A cada hijo se le pasa el pedazo de archivo que va a procesar. El valor de N lo escogen los programadores, debe ser un numero mayor a 3.

2.2 Entradas de los Servidores

Invocación de los Servidores desde el Shell	Significado de los argumentos que reciben los procesos
Servidor Secuencial: ***$ serversec consulta N log outputfile*** *Servidor Concurrente:* ***$ serverconc consulta N log outputfile***	***consulta***: archivo que contiene las consultas que va a ejecutar el servidor. En la sección 2.4.2 se describe el formato de este archivo. ***N***: número de líneas o consultas del archivo anterior. ***log***: es un archivo que contiene datos de ejecución de procesos. Sobre este archivo se realizarán las consultas. El formato e información que contiene el archivo se explica en 2.4.1 ***outputfile***: es el nombre del archivo donde el servidor (secuencial o concurrente) va a colocar las salidas de cada consulta y la hora (o tiempo exacto) en la que terminó de ejecutarse la consulta. El archivo debe estar ordenado por tiempos. El formato de salida se explica en 2.4.3.

2.3 Salidas

Las salidas de cada uno de los servidores son las siguientes:

- **Tiempo y totalización de consultas realizadas**: cada uno de los tipos de procesos debe imprimir el tiempo total que se tomó en realizar todos las consultas, el

número total de consultas procesadas y cuántas estuvieron formuladas correctamente. Estos datos se imprimen por el terminal. El programa termina cuando todas las tareas (procesos o hilos) han finalizado

- **Archivo de Resultados**: es el *outputfile* de los argumentos de entrada su formato se explica en 2.4.3.

2.4 Formato de los Archivos

2.4.1 Log

El archivo de log contiene detalles de la ejecución de procesos en una máquina paralela. 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).** A continuación se muestra un ejemplo del archivo de log con 12 registros.

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

El significado de cada columna se explica a continuación. Las consultas que deberán implementar tienen que ver sólo con algunos de los campos, pero se explican todos por completitud. Son 18 columnas, cada una identifica un proceso.

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.

    Los archivos *datos100*, datos1000 y datos5000 ubicados en **Proyectos/Proyecto 1/Archivos de Prueba** poseen este formato y los deben utilizar para hacer pruebas durante su desarrollo.

2.4.2 Archivo de Consultas

Se realizarán consultas sobre los procesos, procesadores y algunos tiempos que se encuentran en le log. Las consultas tienen un formato específico, con tres ítems. Los tres ítems se muestran a continuación:

T|P|S|NP tipo cuantos|min|max

Donde **T|P|S|NP** representa el objeto sobre el que se hará la consulta. **T**: Tiempos, **P**: Procesadores, **S**: Status y **NP**: Número de Procesadores. El primer parámetro o ítem de la consulta determina los posibles valores de los otros 2, como lo pueden observar en la tabla I:

Consulta	**Valores posibles de *tipo (segundo ítem)***	**Valores posibles en la tercera columna**	**Observaciones**
T	ejecución, espera, cpu	min o max	Se consulta sobre tiempos de ejecución, espera o

			tiempo promedio de CPU utilizado y **se devolverá el valor mínimo o máximo** según el tercer ítem.
P	asignados, solicitados	cuantos	Se pregunta por los procesadores asignados o solicitados a un proceso y **lo que se devuelve es la cantidad.**
S	bien, cancelado, falla	cuantos	Se pregunta por el **número** de procesos que terminaron Bien, con Falla o fueron cancelados
NP	X **(Es un número entero**)	cuantos	Se pregunta por el número de procesos que se ejecutaron en X procesadores, donde X es un número entero.

Tabla I: Formato de las Consultas.

A continuación se muestran algunos ejemplos de consultas. En comentarios, lo que significa cada línea.
.

T espera min // Devuelve el proceso registrado en el log con el tiempo mínimo de espera.
T ejecución max // Devuelve el proceso con el tiempo máximo de ejecución
S bien cuantos // Devuelve el número de procesos que culminaron bien
NP 128 cuantos //Devuelve el número de procesos que se ejecutaron en 128 procesadores.
S falla cuantos // Devuelve la cantidad de procesos que terminaron con una falla.
T espera cuantos // consulta errónea, ya que el primer ítem sólo admite, min o max de tercero.

Los archivos *consulta20* y *consulta100* ubicados en UVirtual, carpeta **Proyectos/Proyecto 1/Archivos de Prueba** poseen este formato y los deben utilizar para hacer pruebas durante el desarrollo del proyecto.

2.4.3 Archivo de Salidas

El archivo de salida tendrá líneas con el siguiente formato:

Tiempo query result

Donde:
Tiempo: hora a la que termino la consulta hh/min/sec
Query: se debe escribir la consulta que se realizó, delimitada con [ ] o “ ”

Result: resultado de la consulta

Ejemplos de salida para las consultas anteriores, según el archivo de log.

16:20: 00 [T espera min] **84762 6**
16:20:03 [T ejecución max] **43349 2**
16:20:10 [S bien cuantos] **12**
16:21:00 [NP 128 cuantos] **3**
16:21:05 [S falla cuantos] **0**
16:21:10 [T espera cuantos] **error en la consulta**

Notas:

- El archivo de salida final debe estar ordenado por tiempos.
- En el archivo de consultas:
  - Puede haber consultas repetidas.
  - Puede haber consultas erróneas, es decir que no satisfacen el formato de la tabla I.
  - Puede suponer que cada consulta (buena o errónea) contiene sus tres ítems. Esto facilita la lectura del archivo.

Se debe respetar el formato de los archivos de entrada y salida, ya que el día de la sustentación se les suministrará tanto el archivo de consulta como el archivo logs.

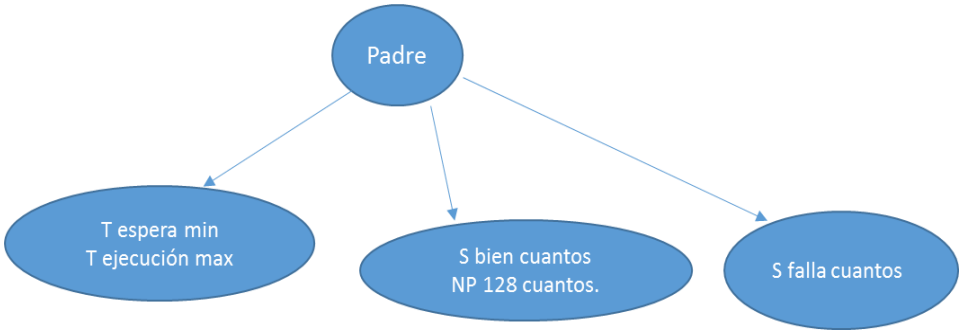


Figura I: Varias consultas por proceso hijo

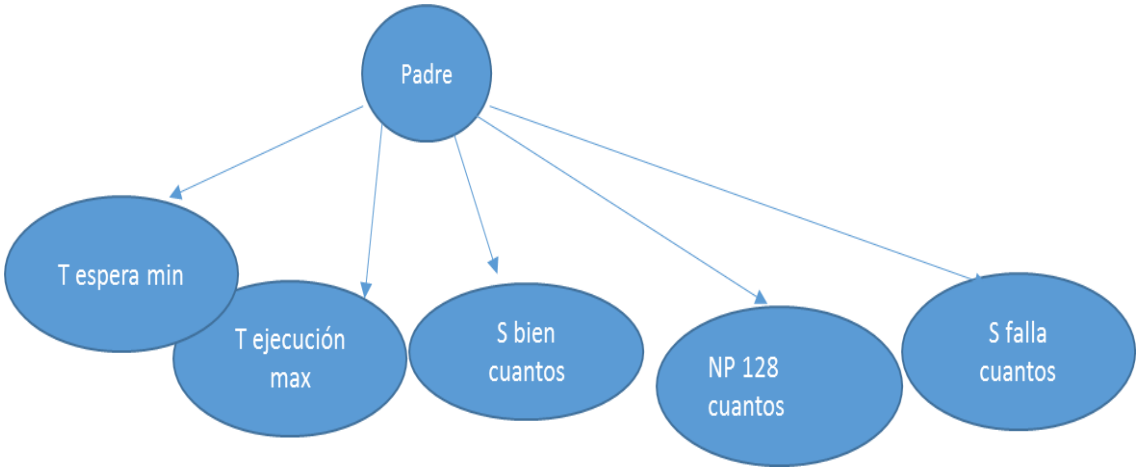


Figura II: Una consulta por proceso hijo

3. Pruebas de Rendimiento

Se realizarán las siguientes pruebas de rendimiento:

1. Comparación de la solución secuencial vs concurrente, utilizando al menos tres tamaños del archivo de consultas. Para ello realizará las siguientes pruebas:

   Mida el tiempo del programa secuencial y del programa paralelo con archivos de consultas de 3 tamaños: 20 consultas, 100 consultas, 500 consultas. Compare y analice los tiempos de los dos servidores. Realice gráficos y tablas. Explique los resultados. La profesora suministrará los archivos con estos tamaños.

2. Solución concurrente variando el tamaño del archivo de consulta y el tamaño del archivo de logs.

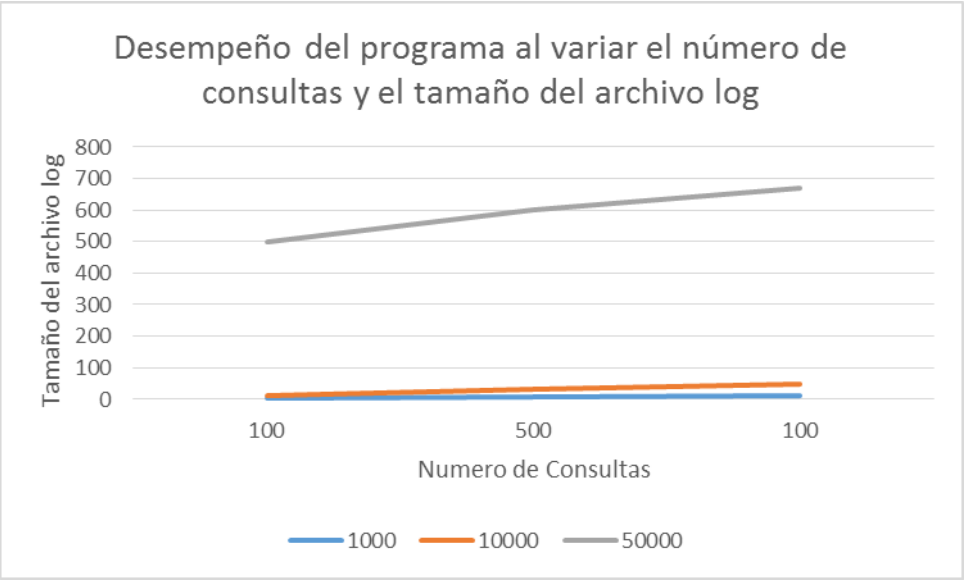
   Realice las medidas indicadas en la tabla II, donde se varían los tamaños del archivo de consultas y del log. En cada caso obtenga el tiempo total de ejecución del servidor concurrente. Coloque los valores en tablas y realice gráficos. En la figura III se muestra cómo pudiera quedar uno de estos gráficos usando valores ficticios. Analice el gráfico. Trate de explicar los resultados con conceptos vistos en la clase. Cree Ud. que podría mejorar el desempeño de su solución. Si hay problemas, cuál creen uds. que es la fuente de los problemas?

   Nota: Para tomar el tiempo de ejecución de un comando puede usar el comando time de la siguiente forma:

   $ ***time serverconc consulta log outputfile***

	Tamaño del log		
Tamaño del archivo de consultas	1000	10000	50000
100	Tiempo total de ejecución	Tiempo total de ejecución	Tiempo total de ejecución
500	Tiempo total de ejecución	Tiempo total de ejecución	Tiempo total de ejecución
1000	Tiempo total de ejecución	Tiempo total de ejecución	Tiempo total de ejecución

Tabla II: Medidas a realizar



4. *Informe*

Una vez realizadas las medidas debe colocar los resultados en un informe que contenga no más de 6 páginas y los siguientes puntos:

- Identificación.
- Responda las siguientes preguntas sobre el diseño realizado:
  - Qué entidades concurrentes usó y por qué?
  - Explique donde escribían los hijos sus resultados y cómo se los comunicaron al padre. Qué tipo de mecanismo de comunicación entre procesos usó: memoria compartida, archivos, etc. En caso de haber usado archivos, cuántos archivos tuvo que crear?
  - Cómo logra que el archivo final esté ordenado. Que algoritmo usa?
- Coloque gráficos y tablas de resultados.
- Mencione los comandos y/o librerías utilizados para las mediciones.
- 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. Qué pasa cuando aumenta la cantidad de las consultas? Qué pasa cuando aumenta el tamaño del archivo a ser consultado.
- Identificó alguna propuesta para mejorar el desempeño de su solución.

Observaciones Adicionales

El proyecto lo deben realizar en grupos de como máximo dos estudiantes. Lo deben entregar el jueves de la semana 9 (31 de marzo) antes de las 12 m por UVirtual. La sustentación se realizará en las horas de práctica. La entrega consiste de los códigos fuentes, el makefile y el informe con la evaluación de rendimiento. Para poder sustentar, el proyecto debe estar en UVirtual.

Debe validar llamadas al sistema y parámetros de entrada. Aunque puede suponer que los archivos vendrán en formato correcto, debe validar que las consultas estén correctamente formuladas. Deben borrar archivos temporales, si los ha usado.

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.

También puede discutir opciones de diseño ya que tienen varias posibilidades para resolver el problema.

Suerte

Prof. Mariela Curiel