Trabajo Práctico de Sistemas Operativos

12 de noviembre de 2014

Universidad de Buenos Aires - Departamento de Computación - FCEN

Integrantes:

- Castro, Damián L.U.: 326/11 ltdicai@gmail.com
- Toffoletti, Luis L.U.: 827/11 luis.toffoletti@gmail.com
- Zanollo, Florencia L.U.: 934/11 florenciazanollo@gmail.com

Índice

1. Introducción

Pthreads Semáforos y variables de condición en pthreads

Desc. breve problema y cómo lo pensamos, quizás un algortithm con el pseudocódigo de la idea sea más legible

2. Resolución

2.1. Variables compartidas y privadas

typedef struct //Cosas del problema explicar que es c/u de sus miembros, //Cosas del pthread int tid;

2.2. Mutexes

2.3. Código

3. Escalamiento

En caso de un aumento drástico en la cantidad de alumnos el problema a tener en cuenta es la memoria.

Un pthread es creado por cada alumno (o cliente) que se conecta al servidor. Esto implica un nuevo stack, el tamaño del stack es variable. Normalmente (y lo vamos a asumir para este caso) es de 2MB.

Entonces para un servidor con 10 clientes (10 pthreads) se necesita[1] una memoria de $10 * 2$ MB $= 20$ MB.
Y para un servidor con 1000 clientes se necesita una memoria de 2GB.

La solución obvia mediante hardware es agregar más memoria principal al servidor de manera que alcance para la cantidad de clientes. Esto conlleva un costo muy alto y puede llegar a volverse inmanejable.

Soluciones mediante software:

Sin pthreads: Se podría implementar un servidor capáz de atender a distintos clientes en 'simultaneo'. Con un mecanísmo de scheduling tal y como se simula paralelismo de procesos contando con solo un CPU. El problema de esta solución es que puede tardar mucho en atender a los clientes (depende estríctamente de la cantidad y el mecanísmo usado). Incluso podría haber *starvation*.

Con pthreads: En vez de tener un pthread por cada cliente se pueden compartir. Es un poco mezclar la idea anterior con pthreads.
Si se crea un pthread que atiende a X cantidad de clientes entonces la cantidad total de memoria requerida para N clientes es de $\frac{N}{X} * 2\ MB$. Se debe elegir un X adecuado ya que si es muy chico se consume más memoria pero los clientes se atienden más rápido y viceversa de ser muy grande.
Por ejemplo para X = 10 la cantidad total de memoria requerida para 1000 clientes es de $\frac{1000}{10} * 2$ MB $= 200$ MB. Se reduce en un 90 %.

[1] En realidad si la memoria no alcanza se podría usar *swapping* pero esto generaría otro problema: *thrashing*