CC3301 Programación de Software de Sistemas

Control 3 – Semestre Primavera 2013 – Prof.: Luis Mateu

Pregunta 1

Un problema que requiere mucho tiempo de CPU es la búsqueda de un factor de un número *x* de gran tamaño, es decir encontrar un valor *f* tal que *x* sea divisible por *f* (que al dividir el resto sea 0). Este problema se resuelve dividiendo x por todos los enteros en el intervalo [2, $\sqrt{x}$]. No es necesario buscar más allá de $\sqrt{x}$ ya que si x es factorizable debe existir al menos un factor en ese intervalo.

En este control Ud. dispone de la siguiente función:

```
uint buscarFactor(ulonglong x, uint i, uint j);
```

En donde *ulonglong* corresponde a los enteros sin signo de 64 bits (unsigned long long) y *uint* son los enteros sin signo de 32 bits. La función *buscarFactor* entrega un entero *f* en el intervalo [*i*, *j*] tal que *x* es divisible por *f*, o cero si no existe un factor en ese intervalo. Observe que aunque *x* sea de 64 bits el factor buscado será representable en 32 bits (tipo *uint*). Con esta función se puede determinar trivialmente si un número x es primo con:

```
uint raiz_x= (uint)sqrt((double)x);
int x_es_primo= buscarFactor(x, 2, raiz_x) == 0;
```

Parte a.-

Suponga que Ud. dispone de un procesador dual core. Programe la función:

```
int es_primo(ulonglong x);
```

Esta función retorna 1 si *x* es primo o 0 si no lo es. Debe usar *fork* para buscar un factor en el intervalo [2, *raiz_x*/2] en el proceso padre al mismo tiempo que busca en el intervalo [*raiz_x*/2+1, *raiz_x*] en el proceso hijo. Use el código de retorno del hijo para transmitir al padre si encontró o no un factor.

Parte b.-

Considerando nuevamente un procesador dual core escriba la función:

```
uint factorizar(ulonglong x);
```

Esta función entrega un factor de *x* o 0 si *x* es primo. Ud. también debe usar *fork* para buscar el factor pedido en paralelo en el padre y en el hijo. Esta vez debe usar un *pipe* para que el hijo envíe al padre el factor encontrado.

Parte c.-

Observe que si ya encontró un factor en uno de los procesos, no necesita seguir buscando en el otro proceso. Modifique la función *es_primo* de la parte *a* de tal forma que si el padre encuentra un factor envíe la señal SIGINT al hijo para concluir la búsqueda. De la misma manera el hijo finalizará cuando encuentre un factor. Esto gatillará el envío de la señal SIGCHLD al padre. Haga que el padre capture esa señal para terminar su propia búsqueda.

Pregunta 2

Esta pregunta también consiste en paralelizar el mismo problema de la pregunta 1. Pero en vez de recurrir a un procesador dual core, considere que se dispone de muchos computadores *single core* conectados a Internet que actuarán como clientes. Un proceso servidor se encargará de coordinar la búsqueda de un factor para el número x.

El espacio de búsqueda de un factor para x se descompone en intervalos del tipo [i, j] tal que $j-i+1=1000000$. Cada cliente recibe un intervalo del socket que lo conecta al servidor; invoca *buscarFactor* para realizar la búsqueda en ese intervalo, lo que toma un buen rato; entrega su respuesta (0 o el factor encontrado) por el socket; lee nuevamente el socket para recibir un nuevo intervalo; etc.

Por otra parte, para cada cliente que se conecta, el servidor crea un thread que se encarga de enviar intervalos a ese cliente, hasta que alguno de los clientes conectados tenga éxito encontrando un factor.

El siguiente es un ejemplo de uso de este sistema. El servidor corre por ejemplo en la máquina *master* y se invoca con el comando *factorizar* especificando como parámetro el número *x*. Use *x= atoll(argv[1])* para convertir el parámetro a *ulonglong*.

```
% ./factorizar 37485903812649747
```

Los clientes se invocan por medio del comando *ayudante* y reciben como parámetro el nombre de la máquina en donde corre *factorizar*, en este caso *master*. Los clientes obtienen un socket para comunicarse con el servidor usando el port 3000 de *master*.

cliente 1	*cliente 2*	*cliente 3*

`% ./ayudante master`
`buscando entre`
`2 y 1000001`
 	
`buscando entre`
`4000002 y`
`5000001`
 	
`%`
	
	`% ./ayudante master`
	`buscando entre`
	`1000002 y`
	`2000001`
	
	`buscando entre`
	`…`
	
	`¡encontrado! factor= …`
	`%`
		
		`% ./ayudante master`
		`buscando entre`
		`2000002 y`
		`3000001`
		
		`%`

Requerimientos: Programe el servidor (*factorizar*) y el cliente (*ayudante*). Cuando un cliente encuentra un factor, debe dejar de enviar nuevos intervalos y cerrar los sockets para que los clientes terminen, pero por simplicidad no necesita terminar el servidor.