CC3301 Programación de Software de Sistemas
Control 3 – Semestre Primavera 2015 – Prof.: Luis Mateu

Pregunta 1

Se necesita estimar el mínimo de una función *f(x)* en un intervalo $[x_0, x_1]$. Para ello se debe evaluar esta función en *m*p+1* valores de *x*. Específicamente se debe evaluar $f(x_0+i*dx)$ con $i= 0, \ldots, m*p$, y $dx= (x1-x0)/(m*p)$ y calcular el mínimo de todas estas evaluaciones. La evaluación de la función *f* es costosa en tiempo de CPU y por ello se necesita paralelizar la estimación utilizando *p* procesos pesados.

Programe la siguiente función:

```
typedef double (*Funcion)(double x);
double min_par(Funcion f, int p, double x0, double x1, int m);
```

La función *min_par* debe usar *fork* para crear *p* procesos pesados. Cada hijo calcula el mínimo de *m* evaluaciones de *f*. El padre debe evaluar *f* en x_1. En total se harán *m*p+1* evaluaciones de *f*. El padre debe retornar el mínimo de todas las evaluaciones de *f*.

Pregunta 2

Esta pregunta consiste en paralelizar la estimación del mínimo de la función *f(x)* en el intervalo [0, 1] evaluando la función en un millón de valores de *x*. Para esta paralelización se utiliza un número desconocido de computadores single-core conectados en red y que actuarán como clientes.

Un proceso servidor corre en anakena (comando *./coordinador*) y se encarga de coordinar la estimación del mínimo y mostrar el resultado final. Para ello descompone el intervalo de búsqueda [0,1] en 1000 subintervalos de la forma $[i*\Delta x, (i+1)*\Delta x]$ con $\Delta x = \frac{1}{1000}$ e *i* tomando valores entre 0 y 999. El servidor acepta conexiones de los clientes a través del puerto 3000 y crea para cada uno de ellos un thread que se encarga de enviar un subintervalo a ese cliente, esperar la recepción del mínimo parcial, enviar de inmedianto un nuevo subintervalo, y así hasta que se acaben todos los subintervalos. Nota: no necesita evaluar *f(1)*.

Cada cliente (comando *./evaluar*) recibe del servidor múltiples subintervalos. Para cada subintervalo el cliente estima el mínimo evaluando la función *f* en 1000 valores de *x*, lo que tomará un buen rato, y lo envía al servidor. Luego recibe de inmediato un nuevo subintervalo (si aún quedan) y continúa evaluando, sin permanecer ocioso en ningún momento. La función *f* es una función dada y es parte del cliente.

El siguiente es un ejemplo de uso que muestra el servidor trabajando con 3 clientes. Los clientes pueden llegar en cualquier momento. El despliegue de los comandos se muestra en orden cronológico.

	cliente 1	*cliente 2*	*cliente 3*
`$ ./coordinador`			
	`$ ./evaluar`		
`env [0.000, 0.001]`	`rec [0.000, 0.001]`	`% ./evaluar`	
`env [0.001, 0.002]`		`rec [0.001, 0.002]`	
`env [0.002, 0.003]`	`rec [0.002, 0.003]`		`% ./evaluar`
`env [0.003, 0.004]`			`rec [0.003, 0.004]`
`env [0.004, 0.005]`		`rec [0.004, 0.005]`	
`env [0.005, 0.006]`			`rec [0.005, 0.006]`
`env [0.006, 0.007]`	`rec [0.006, 0.007]`		
…	…	…	…
`env [0.999, 1.000]`		`rec [0.999, 1.000]`	
`minimo= 5.30450`	`$`	`$`	`$`

Programe el servidor (*coordinador*) y el cliente (*evaluar*). En el servidor no programe la función *main*, solo programe la función *serv*. No se preocupe por el término del servidor. Sí debe preocuparse por el término de los clientes.

Ayuda:

enviar *x* en forma binaria por el socket s	`write(s, &x, sizeof(x));`
recibir *x* en forma binaria por socket el s	`leer(s, &x, sizeof(x));`

Leer entrega 1 si se leyó lo pedido o 0 si se cerró el socket.