Trabajo práctico - Algoritmos en sistemas distribuidos

Sistemas Operativos - 2do Cuat. 2010

Fecha de entrega: Domingo 28 de noviembre de 2010

1. Introducción

En los sistemas operativos distribuidos o de red muchas veces existen varios procesos, distribuidos en distintos puntos de la red, sin una memoria común. En estos casos, el problema de exclusión mutua se complica debido a la falta de estructuras compartidas en una única memoria.

En el año 1981 se publicó el artículo Ricart, G. and Agrawala, A.K., **An optimal algorithm for mutual exclusion in computer networks**, *Communications of the ACM, 24(1)*, págs. 9–17, que presenta una solución óptima (en la cantidad de mensajes) para el problema de exclusión mutua en sistemas distribuidos.

Este TP consiste en implementar la versión del algoritmo presentada en la mencionada publicación, en las secciones 1 y 2.

2. Implementación

Tenemos una cantidad par de procesos, separados en dos grupos: clientes y servidores. Cada cliente está conectado lógicamente con un único servidor. Los clientes leen de un archivo de texto `datos###.txt` los comandos a ejecutar (donde `###` representa el número de cliente). Los comandos posibles son:

- `nada`: el cliente no hace nada.
- `pedir`: el cliente le solicita al servidor que se encargue de darle permiso exclusivo para la sección crítica siguiendo el método de la publicación.
- `escribir` $<$ *string_a_escribir* $>$: comando para imprimir en pantalla la línea. Esta operación debe ejecutarse **ÚNICAMENTE** con la sección crítica asignada.
- `liberar`: libera la sección crítica.

Al iniciar, los clientes abren su archivo de instrucciones y ejecutan, en orden, cada una de las operaciones solicitadas. No hay orden predefinido en la ejecución de los diferentes procesos, sino que es dependiente del *scheduling* que se dé en cada corrida. Al finalizar la secuencia, el cliente y el servidor quedan a la espera de la finalización del resto de los clientes y servidores para el cierre final.

Para implementar el TP utilizaremos MPI[1].

Los procesos pares serán los servidores, cuyos clientes serán los del rango inmediatamente superior, i.e. $1 \rightarrow 0$, $3 \rightarrow 2$, etc.

El cliente ya se encuentra implementado por la cátedra, solicitando al servidor el acceso a una sección crítica cuando el archivo de datos así lo indique. La consigna consiste en implementar

[1] `http://www.mpi-forum.org/docs/docs.html`

la función del servidor que resuelve la sincronización entre los servidores para la asignación de sección crítica en un entorno distribuido.

Dichas funciones deben programarse de manera prolija y documentarse apropiadamente.

Para hacerlo, utilizaremos las siguientes funciones de MPI:

- `MPI_send()` Envío de mensajes.
- `MPI_recv()` Lectura un mensaje.

Una buena descripción de ellas se encuentra en `https://computing.llnl.gov/tutorials/mpi` .

3. Ejecución

Para ejecutar el programa compilado debe usarse:

```
mpirun <--no-daemonize> <-all-local> -np N <./prgcompilado>
```

donde `N` es la cantidad de procesos a lanzar y las opciones entre llaves podrían ser necesarias o no dependiendo de la implementación de MPI que se use.