Manual de referencia

Una implementación del algoritmo de tickets de Ricart-Agrawala sobre un escenario de compra de entradas, con procesos de varias prioridades en exclusión mutua o concurrentes

Adrián Perdiz Amoedo
Diego Pérez Martínez
Javier Freire Otero
Miguel Vila Rodríguez

Introducción

Simulamos un sistema distribuido real en un único programa ejecutado en un único ordenador. Los nodos en la vida real se corresponden en la simulación a N procesos con diferentes roles, con un único proceso receptor para cada nodo.
La comunicación entre estos nodos se simula con una cola de mensajes System V IPC incluida en el kernel de Linux.
Las sincronizaciones internas se dan con semáforos POSIX.
Es necesario que el sistema en el que se ejecuta tenga los suficientes semáforos y colas de mensajes disponibles.

Parámetros

Se proporcionan varios parámetros configurables al usuario para que ajuste el programa:

- **NUMERO_NODOS**: el número de nodos del sistema distribuido
- **PROCESOS_POR_NODO**: el número de procesos, de cualquier tipo, sin incluir al receptor, que conforman un nodo
- **NUMERO_ADELANTAMIENTOS**: el número de adelantamientos que permite un nodo. Es decir, si un nodo tiene procesos esperando en sí mismo, y procesos esperando en otros nodos, le dará prioridad a **NUMERO_ADELANTAMIENTOS** procesos suyos antes de permitir los de los otros nodos.
- **String de nodos**: una array de string en las que se definen el número de procesos de cada tipo que habrá en cada nodo. Cada string es del formato:

  XcXrXaXpXx

  en la que el X que precede a cada letra es el número de procesos de ese tipo. Las letras se corresponden según la siguiente relación:
  0. c: Consultas
  1. r: Reservas
  2. a: Administración
  3. p: Pagos
  4. x: Anulaciones

  La suma de las X debe ser **PROCESOS_POR_NODO**.

Utilización

Una vez se hayan ajustado los parámetros, el usuario debe compilar el código con el compilador *gcc* en plataforma Linux.
Para ejecutar el programa, basta con ejecutar el binario producido por *gcc*.

Salida

El programa mandará a *stdout* una línea por cada sección crítica en la que se entre. La línea contendrá el número de orden de sección crítica en la que ha entrado (puede ser igual en caso de consultas, que son concurrentes), su rol y el nodo al que pertenece.

Pruebas

Para tomar métricas, guardamos sobre la ejecución del programa valores en variables que, al final de la misma, se guardan en varios archivos para interferir lo mínimo en la ejecución. Las gráficas se generan con la librería *matplotlib* de Python de la siguiente manera: creamos una figura con varios subgráficos y en cada uno de ellos mostramos las distintas métricas. En el primero, mostramos el tiempo que tarda un proceso de un nodo en entrar en la sección crítica; en el segundo, el tiempo que ese proceso está dentro de la sección crítica y, en el tercero, mostramos el número de mensajes enviados por cada nodo en su ciclo de vida.