Tarea 4 Procesador multicore

1. Objetivo

Diseñar e implementar con SimPy un modelo de computador multicore.

2. Descripción del sistema

Se debe simular el procesamiento de *K* procesos intensivos en el búsqueda de datos en un procesador multicore con C cores. Cada proceso debe acceder a N_k datos antes de finalizar.

Cada proceso ***k*** tiene los siguientes características:

- ID_k : identificador entero mayor que 0.
- $espera_k$: Tiempo que toma el proceso en ser atendido desde que llega al sistema.
- $tfinalizacion_k$: Tiempo que demora el proceso en terminar desde que llega al sistema.
- $[d_1, d_2, ..., d_{Nk}]$: son los datos que deben ser leídos por el proceso antes de completar su ejecución. Estos datos son representados por las $N_k = \{1, ... , 25\}$ primeras letras del abecedario.

El computador tiene los siguientes características:

- *C* : cantidad de cores $C=\{1, 2, 4, 8, ... ,64\}$.
- L1 : memoria caché L1. Puede almacenar hasta M_{L1} datos. La velocidad de acceso a un dato es de $T_{L2}=4$ ciclos de reloj.
- L2 : memoria caché L2. Puede almacenar hasta M_{L2} datos. La velocidad de acceso a un dato es de $T_{L2}=10$ ciclos de reloj. Esta memoria es compartida con la C cores. Se tiene que $M_{L2} > M_{L1}$.
- RAM : memoria principal. Desde el punto de vista de la simulación, es infinita y contiene todos los datos que los procesos necesiten. La velocidad de acceso a un dato es de $T_{RAM}=200$ ciclos de reloj.

Las tareas son asignadas al primer core libre y no son interrumpidas. Si ambos cores se encuentran libres se selecciona uno al azar. Si los datos no se encuentran en la cache L1 se deben buscar en la cache L2, si no se encuentran, entonces buscar en la RAM. Ambas memorias cache son llenadas en forma secuencial. Al llenarse, se comienza de nuevo en la primera posición de memoria.

3. Trabajo a realizar

Realice un simulador (de nombre **`yacs.py`**) de eventos discretos orientado por procesos para el sistema descrito en la sección anterior. Su objetivo es determinar el thoughput, el tiempo de servicio promedio por cada tarea y el tiempo de utilización de cada core. Debe utilizar como base el código **`yacs-base.py`** disponible en el aula virtual.

El script deberá recibir los parámetros del simulador vía argumentos de entrada. Por ejemplo, si se quiere simular un sistema de 500 procesos, con 8 cores, la capacidad de la cache L1 es de 6 datos, la de L2 es de 18, entonces el uso del script debe ser:

```
$ yacs.py --procesos 500 --cores 8 --L1 6 --L2 18
```

Se requiere la elaboración de un informe detallado sobre las tareas realizadas. Éste debe incluir el diseño y la implementación. El diseño consiste en la descripción del sistema, el modelo de filas, variables de estado y salidas que permitan calcular las métricas solicitadas. Para la implementación, se debe incluir un modelo lógico. Éste se debe basar en un diagrama de clases y servirá como herramienta visual para representar las entidades y relaciones del sistema, ofreciendo una visión sistemática y organizada del trabajo realizado. La plantilla se encuentra disponible en el aula virtual de la asignatura.

Deberá entregar un archivo ZIP (**`tarea04-apellido1-apellido2-nombre.zip`**) con el código fuente de su simulador y un archivo pdf con el informe respectivo. El archivo ZIP, al descomprimir, debe tener la estructura que se muestra en la Figura 1

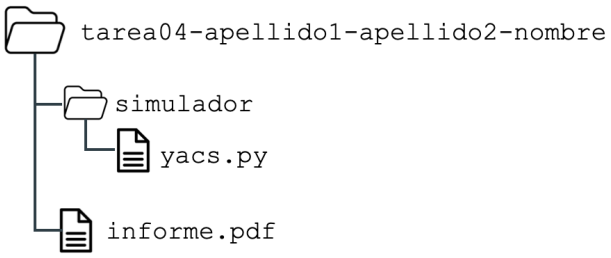

Figura 1