Práctica 1

1. Ejecutar los siguientes casos y justificar su comportamiento:

a. ./practica1 (multicore)
En este caso, al ejecutar el código empleando más de una cpu, se puede observar que en todas las ejecuciones de los hilos se cumple con el tiempo de coste y el periodo. Esto se debe a que la carga del trabajo se reparte entre los distintas cpus.

b. ./practica1 (monocore)
En este caso, al ejecutar el código empleando solo una cpu, se puede observar que en ninguna de las ejecuciones de los hilos se ha podido cumplir con el tiempo de coste y el periodo especificado, dado que supone una carga demasiado grande para que solo una cpu sea capaz de ejecutarla.

c. ./practica1 (monocore) + stress
En este otro caso, al ejecutar el código con una sola cpu y generando estrés computacional en todas las cpus, ninguno de los hilos de ejecución es capaz de cumplir con el tiempo de coste y el periodo. Comparando este resultado con la ejecución monocore sin estrés computacional, apenas se puede apreciar el cambio que genera en este resultado que las cpus estén estresadas computacionalmente.

d. ./practica1 (multicore) + stress
En este último caso, se ejecuta el código empleando más de una cpu y generando estrés computacional en todas las cpus. El resultado de dicha prueba es muy distinto al obtenido en el resto de escenarios. Se puede observar cómo algunos hilos en algunas iteraciones consiguen cumplir con el tiempo de coste y periodo especificado, mientras que en otros hilos se produce un fallo temporal dado que el tiempo de coste se encuentra fuera del especificado. Comparando este resultado con la ejecución multicore sin estrés computacional, se puede observar la diferencia que genera que todas las cpus se encuentren estresadas. Sin embargo, tras varias pruebas en distintos ordenadores del laboratorio, el resultado varía dependiendo del ordenador en el que se ejecute dicho programa, cumpliendo en algunos los requisitos temporales.

2. ¿En qué casos de ejecución (nombrados anteriormente) el sistema es capaz de cumplir las restricciones temporales (tanto tiempo de cómputo como periodicidad)?
Únicamente en el caso de ejecución multicore sin estrés computacional, sin tener en cuenta el último escenario en el que el resultado varía en función del ordenador.

3. ¿Qué número mínimo de cpus se necesitan para que tu programa ejecute correctamente sin fallos de restricciones temporales? Usa el comando taskset para comprobarlo.
Tras realizar varias pruebas, el número mínimo de cpus que se necesitan para que mi programa ejecute correctamente sin fallos de restricciones temporales son 3.

4. ¿Qué solución se podría proponer para cumplir plazos estrictos temporales de periodicidad en la ejecución de los threads utilizando la configuración actual que tienen los ordenadores del laboratorio?
Se podrían crear los threads modificando su prioridad a un nivel más alto, y así conseguir de esta manera que el planificador otorgue más cpu a dichos threads pudiendo cumplir con los plazos estrictos temporales de periodicidad.

Daniel Alejandro Quinga López