Noms: Victor Correal i Sergi Gil

CASO Pràctica Temps Real

22-05-2020

Resultats obtinguts en:

ARM-v7 @ 800MHz

Linux 4.19.97-v7+

- ***Hackbench:***

L'objectiu principal d'aquest test és mesurar el temps que triga el sistema en executar un nombre determinat de tasques. Per això, el test requereix com a paràmetres una quantitat determinada de file descriptors (el 40 i el 80 de l'exemple) i el mode (-P si ho volem fer amb processos i -T si ho volem fer amb threads). Per a cada file descriptor tenim un receptor (receiver) i tota la resta són emissors (senders). El sender s'encarrega d'enviar un nombre determinat de bytes.

Paràmetres:	***Time (s)***
Processes, 40	5.39
Processes, 80	22.64
Threads, 40	5.66
Threads, 80	Error: Too many open files

Comparant els resultats obtinguts en el nostre experiment amb els proporcionats a l'enunciat, veiem que ens triga molt més (pitjors resultats), potser degut a la diferencia de freqüència de CPU (1.70GHz i nosaltres 800Mhz). No obstant això, es pot observar la mateixa proporción entre els Processes-Threads 40 y el Processes 80 (més o menys un x4 en temps).

- ***pip_stress, pi_stress:***

El programa busca examinar la manera d'actuar del kernel en els canvis de prioritat per posix threads amb mutex. Consisteis en establir un thread principal, un de report i alguns grups de 3 threads que tenen com objectiu provocar la condició d'inversió de prioritats, on un thread amb high priority es bloquejat perquè un altre thread amb low-priority necesita el recurs compartit que està fent servir ell. El pip_stress vol fer el mateix que el pi_stress però amb processos enlloc de threads.

Un cop modificat el pip_stress per a permetre l'execució per un usuari normal, hem vist que el resultat no ens ha variat. Igual ha passat en implementar la nova versió del pip_stress amb priority ceiling.

- ***pmq_test:***

L'experiment mesura la latència de la comunicación entre processos amb cues de missatges POSIX mitjançant parelles de threads sincronizats.

Parella de threads:	***Mínim***	***Mig***	***Màxim***
#1 -> #0	12	21	85
#3 -> #2	12	19	94

Els nostres resultats són bastant més elevats en comparació amb els de l'enunciat. No obstant això, els nostres màxims, tot i ser més grans que els de l'enunciat, tenen menys marge de diferencia entre ells (22 a l'enunciat i 9 els nostres). Probablement es deu a que la nostra CPU té 800MHz i la de l'enunciat té 1.70GHz.

- ***signaltest:***

L'experiment té com objectiu calcular el temps de resposta del kernel davant d'un signal mitjançant threads i reprogramació de signals, a més de l'ús de prioritats.

Resultat nostre:

T: 0 (13234) P: 0 C: 10000 Min: 30 Act: 51 Avg: 50 Max: 426

Els nostres resultats són molt més elevats, sobretot en la màxima obtinguda (426 la nostra i 134 la de l'enunciat). Probablement es deu a que la nostra CPU té 800MHz i la de l'enunciat té 1.70GHz.

- ***cyclictest:***

Aquest experiment mesura la quantitat de temps que ha pssat entre que expira un timer i el següent timer s'activa.

Resultat nostre:

T: 0 (16187) P: 0 I:1000 C: 10000 Min: 26 Act: 72 Avg: 76 Max: 5358

La diferencia de resultats es absurda i molt desproporcionada. Això es molt possible que es degui a que fem servir una Raspberry Pi com a máquina.