Examen 3 CASO

Victor Correal 47667745s

1) Què és un Sistema Operatiu?

a) Creus que el CNK és un sistema operatiu?

> Personalment, crec que si que és un sistema operatiu, donat que proporciona una interficie entre el software (en aquest cas, un procés MPI amb multithread) i el hardware.

b) En quina de les funcionalitats atribuides a un SO, el CNK té una mancança important?

> Aquest SO te una implementació pensada només en executar un únic procés MPI que efectua càlcus de computació, per aquest motiu la funcionalitat de control i creació de procesos diferents es limita a un proces MPI amb multithreads

2) Mach tasks vs Unix processes

a) Quina era (és) la syscall (o les syscalls) per crear aquest dibuix a Mach? I a Unix?

> A UNIX podem crear un procés amb les crides derivades de clone (per exemple fork) i a Mach s'utilitza el task_create.

b) Quina és la diferència més significativa entre els dos dibuixos? Quina era (és) la syscall per gestionar la zona ratllada a Unix (Linux)? I a Mach?

> A UNIX tenim les crides de malloc o sbrk; d'altra banda a Mach tenim la crida vmap

c) A POSIX (i per tant, s'implementa tant a Linux com a BSD), es defineix una system call per gestionar la memòria de manera semblant a Mach. Quina és? Quina funció de més alt nivel coneixes que la faci servir? En quines circumstàncies?

> La crida es mmap, que a alt nivell es s'utilitza en la crida malloc si l'espai que es demana supera cert llindàr.

d) Explica quins advantages veus al model actual de Linux. Creus que necessita d'un suport hardware?

> En el model de linux, al tenir una regió contigua disponible fa més fàcil i més eficient l'interacció amb el hardware de memória.
>
> Comparat amb el model mach, es possible que les dades dinàmiques estiguin en diferents regions fent que el possible acces sigui mes costos a nivell de hardware, comparat amb una regió contigua
>
> Per tant, una regió contigua proporciona un major rendiment.

> >A

3) Què li va passar al Pathfinder?

a) Quin sistema operatiu es va fer servir a la missió? Encara té suport actualment?

> El sistema operatiu emprat va ser un VxWorks, un sistema operatiu de temps real, actualment encara existeixen productes amb aquest OS.

b) Quin va ser el problema? Com el van poder diagnosticar?

> El problema va ser que el sistema operatiu de VxWorks tenia un bus de control amb exclusió mútua, ja que el bus s'utilitzava en diferents tasques.
>
> El problema ocurreix en el moment que es una tasca perd deadlines per que s'està executant sense el *mutex* i la tasca que té el mutex no es pot executar.
>
> Segurament, diagnosticar aquest problema tenint el hardware tant lluny va requerir l'estudi de traces del sistema, per tal de veure l'estat de les tasques.

c) Quants threads estaven involucrats en el problema? Què feia cadascun?

> Els threads que estaven en conflicte(pel bus) eren tres:
>
> - Dades meteorològiques (baixa prioritat): Aquesta tasca bloqueja el bus i hi publica les dades.
> - Thread de comunicacions (prioritat mitja): Es el thread que comunicava amb la Terra
> - Thread de control del bus(alta prioritat): Bloqueja el bus i posa/treu dades
>
> Aquest threads podien prendre el bus, sortir de CPU amb el mutex agafat (fent que la nova tasca també surti de CPU perque necesita el mutex).
>
> Si justament en aquest punt entra la tasca de comunicacions(de prioritat intermedia) que no allibera el mutex predrem temps amb un mutex agafat i cap tasca fent servir el recurs.
>
> Aquest temps fa enraderir el control sobre el mutex, donant lloc a pèrdues de *deadlines* i a fallades del sistema.

d) Com es va resoldre? Quins dels mecanismes disponibles a la llibreria de threads van fer servir?

> Es va poder resoldre ja que el OS tenia les opcions de debug encara activades , llavorns es va poder reconfigurar el mutex amb un sistema d'herència de prioritats.
>
> Les llibreries poden afegir als mutex atributs.

4) Jitter i les interrupcions

a) Expliqueu el tema del jitter en el sistema operatiu, com afecta als processos i també a les aplicacions paral·leles, i relacioneu-lo amb l'assignació de les interrupcions que veieu en aquesta informació prèvia que hem vist.

> El terme *jitter* es la variació en el temps d'execucció d'alguna tasca/procés degut a interaccions amb altres processos o a les interrupcions. Si tenim un sistema operatiu de temps real, aquest jitter ha d'estar acotat, en cas contrari podem perdre *deadlines*.
>
> En aplicacions paral·leles sobre un sistema operatiu (on sempre acaben comunicant-se d'alguna manera els procesos), es crearan variacions de rendiment (temps d'execucció -> jitter) donat que aquesta comunicació es desordenada, per exemple:
>
> - una tasca A, pot no estar influida per ningu
> - una altra tasca B, pot estar influida per l'execucció de A i tres tasques més.
>
> En un sistema de temps real, no ens podem permetre això i s'han de planificar les interaccions entre tasques.

b) Doneu la vostra opinió sobre si un sistema amb aquestes característiques podria servir per donar algun tipus de servei de temps real en mode hard.

> En un sistema de temps real hard si perdem una *deadline* el sistema falla completament. Observant les dades del sistema podem veure els següents trets importants:

1. La CPU0 és l'única que rep interrupcions de teclat i de la targeta de so.
2. La CPU1 és l'única que rep interrupcions d'altres CPUs i del *real-time clock*.
3. La CPU2 rep interrupcions de la tajeta gràfica i del port USB2
4. La CPU3 en rep del USB1 i del *system managment bus*

Tenint en compte aquests trets sobre el sistema, podem veure que el sistema de l'exercici es un sistema que té un *real time clock*, condició casi indispensable per un hard real time. Seguint evaluant el sistema, veïem que el CPU "central" és el 1, ja que es l'unic que rep interrupcions IPI, per tant, el rendiment que pot donar aquest processador pot variar força, per l'alt nombre de peticions (via IPI) que pot haver d'atendre.

Mirant les dades d'interrupcions per segon, el CPU1 es el que menys en té (308.8 interrupcions/segon), caldria fer un anàlisis per veure si amb 1000 Hz de rellotge tenim el suficient rendiment per tractar les tasques que necesitaria el sistema de temps real i atendre les interrupcions.

Per tant, podem dir que el sistema pot arribar a ser de temps real, sempre que les interrupcions que rep el CPU1 (en el cas pitjor) no fa caure el rendiment (jitter).