Resumen de la clase pasada

Preempt: desalojar un proceso, siempre a la fuerza.
Yield: es cuando un proceso cede voluntariamente el procesador

Todos los UNIX son preemptive, el scheduler saca al proceso

> Linux es un bicho raro, es preemptive pero no sigue la misma logica que UNIX.
> -Mariano

El time sharing puede ser cooperativo o preemptive, en los ultimos anios hay solo dos casos cooperativos, Windows 3.11 y Mac OS 9.
Una politica de time sharing cooperativa implica que los procesos deciden cuando ceder el tiempo de procesador.

> Si bien la cooperatividad es buena, en este caso no. Lo mejor es ser preemptive.
> -Mariano

Time sharing

1. Cuales deberian ser ahora las suposiciones a tener en cuenta?

- Cuales metricas son importantes?

En la vida real se usa algo de teoria de colas.

> Por que no es importante datos y si algoritmos II para esta materia? Porque en sistemas operativos se usan listas, listas doblemente enlazadas, arboles binarios y arboles rojo-negro.
> -Mariano

Multi-Level Feedback Queue

Ataca dos problemas

1. Optimizar el tornaround Time: si optimizas el turnaround tenes que hacer la tarea mas corta primero, pero no sabes cuanto duran…

- Hace que el sistema tienda a ser interactivo, eso es, la interaccion con el usuario es fluida.

MLFQ: Las reglas

MLFQ tiene 8 colas distintas, con una prioridad asignada, la cola 8 tiene mas prioridad y la cola 1 menos prioridad.
En un determinado tiempo una tarea que esta corriendo esta siempre en una determinada cola, no en varias, y se van a usar las distintas prioridades para saber que proceso deberia estar corriendo en un determinado t0.
Si existe mas de una tarea en una cola se hace un Round Robin de los procesos en esa cola.

Las reglas basicas son:

- Si la prioridad de A es mayor que la de B, A se ejecuta y B no
- Si la prioridad de A es igual a la de B, A y B se ejecutan en RR.
- Si una tarea repetidamente no utiliza CPU mientras espera que un dato sea ingresado por el teclado MLFQ va a mantaner su prioridad alta.
- Si por el contrario una tarea usa intesivamente CPU el planificador va a empezar a bajarle la prioridad.

> Va a tender a no darle bola a los procesos de alto computo y darle bola a los procesos interactivos.
> -Mariano

> Por que usa una cola?
> -Uri

> Porque viene de la teoria de colas le dicen asi, la implementacion no tiene nada que ver.
> -Mariano

Primer intento: Como cambiar la prioridad?

- Cuando una tarea entra al sistema se pone con la mas alta prioridad.
- Si una tarea usa un time slice mientras se esta ejecutando su prioridad se reduce una unidad.
- Si una tarea renuncia al uso de la CPU (yield) antes de un time slice completo se queda en el mismo nivel de prioridad.

Problemas con este approach:

- Si hay demasiadas tareas interactivas se van a robar todo el tiempo de computo
- Un programador podria reescribir sus programas para obtenar mas tiempo, usando yield antes de terminar el time slice.

Segundo approach

- Para cambiar el problema de starvation despues de cierto tiempo S se mueven las tareas a la cola con mas prioridad

Haciendo esto se matan dos pajaros de un tiro:

1. Se garantiza que los procesos no caigan en starve, al volver a la cola principal se van a ejecutar en RR.

- Si un proceso se vuelve interactivo lo tratara como tal una vez que haya recibido la prioridad.

Intento 3: llevar mejor la contabilidad

Se debe solucionar otro problema: como prevenir que ventajeen al procesador? (*gaming*)

Nueva regla:

- Una vez que una tarea usa su asignacion de tiempo (*Virtual run*) en un nivel dado (independientemente de cuantas veces haya renunciado al uso de la CPU) su prioridad se reduce. Se guarda cuanto tiempo consumieron antes, si la suma de todos los yields da mas de un time slice es cuando se baja la prioridad.

Proportional Share

Es un algoritmo muy naive, que funciona asignando numeros y con un sorteo al azar, es como una "rifa".

Planificacion avanzada: Planificacion multiprocesador

Narrador: Mariano nos esta explicando la historia de la informatica, el paralelismo y concurrencia, Fortrand y el aeropuerto de Denver. No se que resumir porque no estuve escribiendo, pero es re divertido.

> El que logre que un programa pueda pasar un codigo de secuencial a paralelo se vuelve millonario
> -Mariano

En los ultimos anios los sistemas multi-procesadores han ido creciendo.

> Tenes 8 cores? En uno tenes corriendo el chrome, en otro el sistema operativo, y en los otros el antivirus.
> -Mariano

Multiples nucleos estan empaquetados en un unico chip.

1. Una aplicacion tipica por ejemplo un programa escrito en C usa unicamente una CPU.

- Las aplicaciones multithread pueden diseminar el trabajo a lo largo de multiples CPUs y por ende correr mas rapido.

Hay dos principios, *localidad espacial* y *localidad temporal*.

Que sucede cuando hay multiples procesador en un unico sistema para compartir una unica memoria principal?

La solucion basica que el hardware provee es mediante la monitorizacion de los accesos a memodia.

Un ultimo tema: afinidad de cache

El ultimo tema a tener en cuenta cuando se arma un planificador con multiprocesadores con cache, es conocido como la afinidad del cache.

Cada vez que un proceso hace un context switch el cache debe ser cambiado.

Planificador de cola unicamente

La ventaja es su simplicidad, pero:

1. No es escalable

- El desarrollador tiene que hacer yield si o si, no es preemptive
- La afinidad del cache es un problema, por ejemplo si se tienen 4 procesador y 5 tareas.

Multi-Queue Planification

Consiste en mutiples colas, una para cada procesador con su propia politica (por ejemplo RR).

Scheduler the Linux

Para todos los unixes el scheduler es O(n).

Hasta el kernel 2.4 para Robert Love el Scheduler era facil, "pedestrian" lo describe.
En el kernel 2.6 el objetivo es que el Scheduler fuera O(1), el planificador es CFS, "completely fair

scheduler".
Los procesos de linux tienen un *nice value* de -20 a 19, donde mas bajo es mas prioridad y mas alto menos. Para determinar la prioridad del proceso hace un promedio del nice de los otros procesos, y te da un porcentaje de CPU y no un time slice.