Practico CPU - SistOp

Lautaro Bachmann

Contents

1)

Como resolver

a)

Fijarse a partir de que cantidad de instancias comienza a aumentar considerablemente el walltime

Por cada proceso que supere la cantidad de nucleos el walltime es 25% mayor

b)

Puede que este proceso no sea el unico que se esté ejecutando

Resolucion

a)

El sistema tiene 2 nucleos, ya que al iniciarse mas de dos instancias, cada instancia adicional que se agrega incrementa el walltime

b)

Esto se da por el overhead que puede llegar a tener la computadora, ya sea por cosas como un context switch o que se estén ejecutando otros programas de fondo

c)

1) Una sola instancia que tiene algo de overhead que causa que el cputime sea menor al walltime.

2) Dos instancias, que debido a que el sistema tiene dos nucleos se ejecutan de manera similar al primer caso

3) Una sola instancia, pero con un overhead mas pronunciado

4) Cuatro instancias, pero con un overhead reducido. Acá se puede ver bastante claro que el sistema tiene 2 nucleos.

6) Dos instancias con considerable overhead

2)

Como resolver

Prestar atencion a la parte que dice que “implementa procesos e hilos” ## Resolucion

El walltime es menor al cputime ya que el programa se está ejecutando en varios hilos al mismo tiempo, por ende aumenta el tiempo de procesador, pero no el tiempo de reloj # 3) ## Como resolver Pensar en los casos en los que hay overhead, en los que no hay overhead y en los que se usan varios hilos

Resolucion

user<real

Esto sucede cuando hay overhead en el sistema, ya sea por context switches o que se esté ejecutando algun otro proceso al mismo tiempo

user=real

Esto sucede cuando no hay ningun tipo de overhead en el sistema o en casos puntuales donde se usen hilos que terminen compensando el overhead del sistema operativo.

user>real

Esto sucede cuando el proceso se está ejecutando usando varios hilos

4)

Como resolver

Al hacer un fork tambien se guardan los registros

Resolucion

a) x vale 100 en el proceso hijo

b) La variable cambia segun el valor que le haya asignado el respectivo proceso

c) Sucede exactamente lo mismo, porque fork tambien copia los registros del proceso padre

5)

Como resolver

Pensarlo con n forks

Resolucion

El program imprime $2 \cdot 2^N - 1$, con $N = 3$

6)

Como resolver

Resolucion

En el caso en el que execv se ejecute correctamente, 0

En caso de que execv falle, 1

7)

Como resolver

Resolucion

argv = prog arg1 arg2
argc = 3

Si el programa tiene argumentos convierte el ultimo arg en Null

Ejercicio 7. Indique que hacen estos programas.

```
int main(int argc, char ** argv) {
        if (0<--argc) {
                argv[argc] = NULL;
                execvp(argv[0], argv);
        }

        return 0;
}
```

→ prog, arg1, null

Ambos programas hacen casi lo mismo, solo que el de la derecha usa un fork.

Si no se le pasaron argumentos al programa, este termian su ejecucion. Si se le pasaron argumentos, convierte el ultimo argumento del programa en NULL y ejecuta execvp con la info pasada al programa.

En otras palabras, ejecuta un programa eliminando el ultimo argumento que se le pasó.

9)

Como resolver

Limitar la cantidad de forks que se pueden hacer. Se puede usar ulimit en bash

Resolucion

El programa funciona haciendo un bucle infinito que está constantemente generando forks del programa.

Una forma de limitar sus efectos es reducir la cantidad de forks que puede hacer un programa o poner un limite a la velocidad de crecimiento de los forks de un programa.

una herramienta que se puede usar es ulimit, de bash

12)

Como resolver

Resolucion

a)

Verdadero. Es posible cuando se usan hilos.

b)

Falso. La idea de las virtualizacion de memoria es que los procesos tengan su propio espacio de direcciones, independientemente de donde esten ubicados en memoria fisica.

c)

Verdadero. Los registros del procesador son una parte importante del estado de cada proceso y es necesario guardarlos.

d)

Falso. Hay instrucciones que por cuestiones de seguridad del sistema operativo solo las puede realizar el kernel, como puede ser el acceso a memoria.

e)

Verdadero. Puede suceder cuando solo hay un unico proceso a ejecutar.

f)

Verdadero. El unico proceso que puede llegar a tener PID 0 son los procesos como systemd, por ende, se puede dar por sentado que no habrá otros procesos con PID 0 ### g)

h)

Falso. Dicho proceso pasa a ser un proceso zombie.

i)

Verdadero.

j)

Falso. Se ejecuta cuando execv falla.

k)

Verdadero.

13)

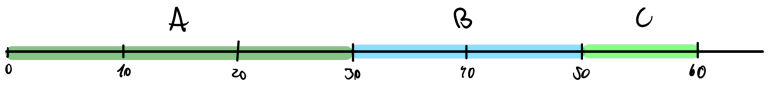
Howto

FCFS es FIFO

Resolucion

Ejercicio 13. Dados tres procesos CPU-bound puros A, B, C con $T_{arrival}$ en 0 para todos y T_{cpu} de 30, 20 y 10 respectivamente. Dibujar la línea de tiempo para las políticas de planificación FCFS y SJF. Calcular el promedio de $T_{turnaround}$ y $T_{response}$ para cada política.

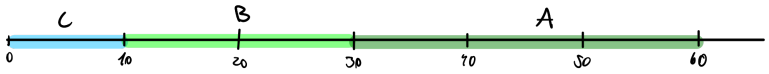
FCFS →



	A	B	C
T_{cpu}	30	20	10
$T_{arrival}$	0	0	0
$T_{completion}$	30	50	60
$T_{turnaround}$	30	20	10
Promedio	20		

	A	B	C
T_{cpu}	30	20	10
$T_{arrival}$	0	0	0
$T_{firstrun}$	0	30	50
$T_{response}$	0	30	50
Promedio	26.66		

SJF →



	A	B	C
T_{cpu}	30	20	10
$T_{arrival}$	0	0	0
$T_{completion}$	60	30	10
$T_{turnaround}$	60	30	10
Promedio	33,33		

	A	B	C
T_{cpu}	30	20	10
$T_{arrival}$	0	0	0
$T_{firstrun}$	30	10	0
$T_{response}$	30	10	0
Promedio	13,33		

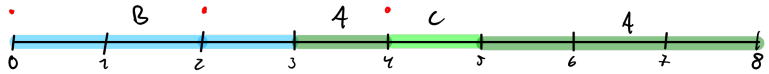
Promedios

	Turnaround	Response
FCFS	20	26,66
SJF	33,33	13,33

14)

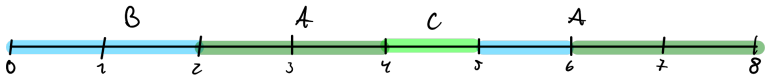
Ejercicio 14. Para esto procesos CPU-bound puros dibujar la línea de tiempo y completar la tabla para las políticas apropiativas (con flecha de running a ready): STCF, RR(Q=2). Calcular el promedio de $T_{turnaround}$ y $T_{response}$ en cada caso.

STCF



Proceso	$T_{arrival}$	T_{CPU}	$T_{firstrun}$	$T_{completion}$	$T_{turnaround}$	$T_{response}$
A	2	4	3	8	6	1
B	0	3	0	3	3	0
C	4	1	4	5	1	0
					3,3	0,33

RR



↳ Asumiendo que prioriza nuevos procesos agregados

Proceso	$T_{arrival}$	T_{CPU}	$T_{firstrun}$	$T_{completion}$	$T_{turnaround}$	$T_{response}$
A	2	4	2	8	6	0
B	0	3	0	6	6	0
C	4	1	4	5	1	0
					4,33	0

15)

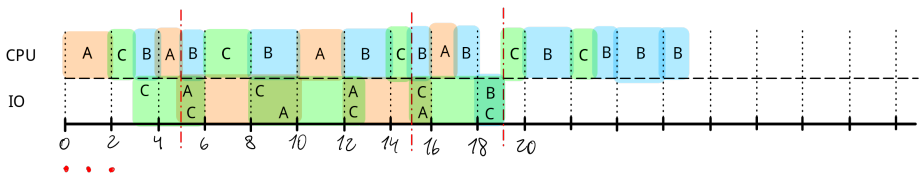
Politica	Batch	Interactiva	Usa Tcpu
FCFS	X		
SJF	X		X
STCF		X	X
RR		X	
MLFQ		X	

16)

Proceso	$T_{arrival}$	T_{CPU}	T_{IO}	T_{CPU}	T_{IO}	T_{CPU}	T_{IO}	T_{CPU}
A	0	3	5	2	4	1		
B	2	8	1	6				
C	1	1	3	2	5	1	4	2

Asumimos:

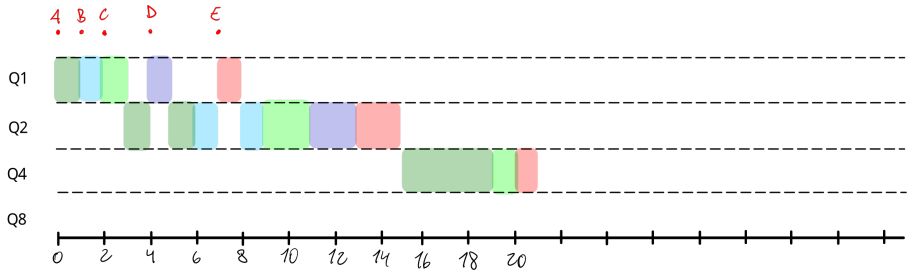
- Se prioritizan nuevos procesos
- El cuanto ocurre de manera fija y cambia siempre de proceso



17)

Ejercicio 17. Realice el diagrama de planificación para un planificador MLFQ con cuatro colas (Q=1, 2, 4 y 8) para los siguientes procesos CPU-bound:

Proceso	$T_{arrival}$	T_{CPU}
A	0	7
B	1	3
C	2	4
D	4	3
E	7	4



18)

a)

Falso. De no ser así el procesador quedaria ocioso y no se aprovecharian bien los recursos

b)

Falso. Pueden darse la casualidad de que sean iguales y por ende tendrian el mismo turnaround

c)

Verdadero. RR con quanto infinito ejecuta hasta finalizar los procesos a medida que van llegando

d)

Verdadero. Sucede si hay varios procesos interactivos en el sistema que tienen una prioridad alta constantemente

e)

Verdadero. Esto permite evitar que se intente burlar al scheduler