Unidad 5. Planificación de Colas

Planificación

El Objetivo de la planificación es repartir el tiempo disponible de atención de un Recurso Servidor entre todos los Elementos que están en cola y disponen su Servicio. En un instante dado, en una cola pueden existir diversos Elementos listos para ser Atendidos. Sin embargo solo uno de ellos puede serlo (en cada Servidor). De ahí la necesidad de que una parte del sistema gestione de alguna manera qué Elemento debe ser atendido en cada momento.

Existen dos tipos de algoritmos de planificación, expropiativos y no expropiativos. Los primeros permiten que se atienda el Elemento hasta que acabe su trabajo, los segundos, asignan un tiempo de Atención a cada Elemento después del cual se lo saca y se atiende a otro Elemento y así repetidamente, hasta que cada Elemento acabe su trabajo.

Criterios De Performance

Los distintos algoritmos o políticas de colas tienen diferentes propiedades y pueden favorecer a una determinada clase de procesos, por lo tanto debo ver cual usar en cada situación. Las características que se usan para estas comparaciones pueden ser de sustancial importancia para la determinación del mejor algoritmo.

Algunos criterios que se usan son:

- *Utilización del Servidor* [se mide en %]: Cuando el servidor es muy costoso, se desea que este lo mas ocupada posible.

- *Rendimiento*: [se mide en Procesos/Hora]Si el servidor esta ocupado, significa que se esta realizando trabajo. Una medida de este trabajo es el número de procesos que se completan por hora y recibe el nombre de rendimiento. Para tareas grandes este índice puede ser de 1 por hora; para transacciones cortas, el rendimiento puede ser de 10 trabajos por segundo.

- *Tiempo de espera:* Es el tiempo que el proceso, elemento etc. está en la cola a la espera de ser tratado.

- *Tiempo de Servicio:* Es el tiempo que el servidor tarda en atender el requerimiento del proceso.

- *Tiempo de respuesta:* es el tiempo desde que se realiza un pedido hasta el momento en que se produce la primera respuesta. Esta medida, llamada tiempo de respuesta, es el tiempo que el proceso tarda en comenzar a responder, pero no el tiempo que tarda en emitir esa respuesta.

- *Tiempo de retorno* [Tespera + Tservicio]: Desde el punto de vista de un proceso en particular, el criterio importante es cuanto le lleva completar el trabajo. El

intervalo desde el momento de inicio hasta su final, es el tiempo de retorno. El Tiempo de Retorno es Mayor o Igual al Tiempo de Respuesta. El La respuesta fuese solo 1 Bit, entonces, ambos tiempos serian iguales.

Una vez elegido un criterio, generalmente deseamos optimizarlo, debido a eso existen varios algoritmos o política de colas.

En sentido general: BUSCAMOS:

- *Maximizar:*
  - *Utilización del Servidor*
  - *Rendimiento del Servidor*
- *Minimizar:*
  - *tiempo de retorno*
  - *tiempo de espera*
  - *tiempo de respuesta.*

Métodos De Planificación De Colas

First-Come-First-Served: (FCFS)

Es por lejos el algoritmo más simple. Se lo implementa con una cola FIFO. Cuando entra un proceso a la cola, este se coloca al final de la misma. El código para este método es simple de escribir y entender, sin embargo, la performance del FCFS es a menudo muy pobre.

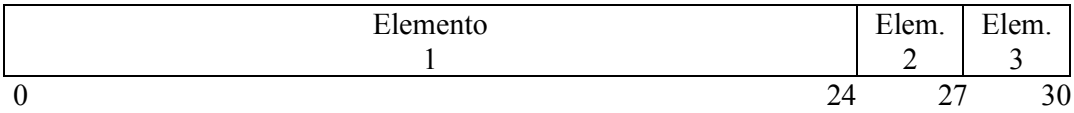
Los Algoritmos estudiados en el capitulo anterior, cuando se desencola, se esta utilizado un sistema de planificación FCFS.

Análisis de Performance FCFS

Veamos los siguientes 3 elementos. Por cada uno suponemos que conocemos el tiempo que llevará tratarlo (24, 3 y 3). Podemos computar el tiempo de retorno promedio para servirlos.

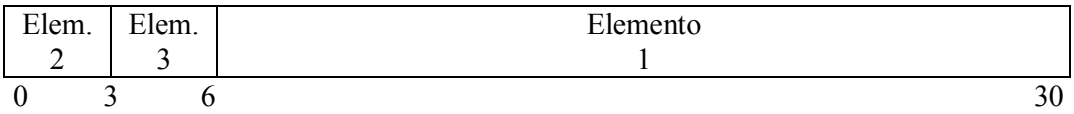
Elemento	Tiempo de Ejecución
1	24
2	3
3	3

Vemos el resultado en el gráfico de Gantt:



El tiempo de retorno para elemento 1 es 24; para el 2 es 27 y para el 3 es 30. El tiempo de retorno promedio es (24+27+30)/3 = 27.

Si el orden de llegada es 2,3,1, tenemos:



Aquí el tiempo de retorno promedio es ahora muy inferior: (3+6+30)/3 = 13.

Así vemos que el tiempo de retorno promedio para FCFS, generalmente, no es mínimo y puede variar mucho.

Shortest Job First: (SJF)

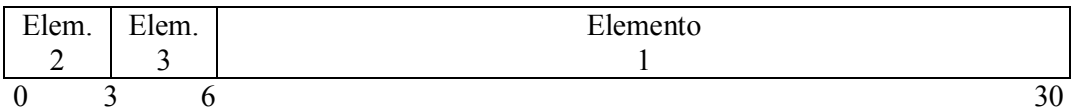
Asocia a cada elemento el tiempo que llevará ejecutarlo, y así lo hace al que sea el más corto. Si existen 2 elementos con el mismo tiempo de tratamiento se usa FCFS. Esto permite introducir el concepto de Prioridad, en este caso, asociado al tiempo de ejecución. La ventaja que presenta este algoritmo sobre el algoritmo FCFS es que minimiza el tiempo de finalización promedio, como podrá verse en el Análisis.

Análisis de Performance SJF

Analicemos el mismo ejemplo anterior:

Elemento	Tiempo de Ejecución
1	24
2	3
3	3

El orden de ejecución es primero 2, luego 3 (recordar que si 2 trabajos tiene la misma duración se los atiende en política FCFS) y por ultimo el trabajo numero 1. Así tenemos un gantt:



Aquí el tiempo de retorno promedio es ahora muy inferior: (3+6+30)/3 = 13. Este análisis demuestra lo superior del SJF contra el FCFS.

Ahora Analicemos otro Ejemplo:

Elemento	Tiempo de Ejecución
1	11
2	8
3	6
4	3

Veamos el Gantt de FCFS

Elemento 1	Elemento 2	Elemento 3	Elemento 4
0.. 11	19	25	28

Tr(FCFS)= (11+19+25+28)/4=20,75

Veamos el Gantt de SJF

Elemento 4	Elemento 3	Elemento 2	Elemento 1
0.. 3	9	17	24

Tr(SJF)= (6+9+17+24)/4=14

SJF es un buen algoritmo pues tiene el mínimo tiempo de espera promedio para un conjunto de elementos. Se demuestra que tratando a un elemento corto antes que uno largo, se disminuye más el tiempo de espera del corto que el incremento de espera del largo. Por lo tanto el promedio de tiempo de espera se achica.

La Gran dificultad del SJF es conocer el tiempo de tratamiento de los elementos.

Prioridad

SJF es un caso especial del algoritmo de planificación por prioridades. Se asocia una prioridad a cada elemento y se otorga el servidor al trabajo con mayor prioridad.

Aquellos trabajos con igual prioridad, son planificados de acuerdo a FCFS.

El SJF simplemente es un algoritmo de prioridad donde la prioridad (p) es la inversa de del tiempo de tratamiento (T), p=1/T, aquí se puede ver que a mayor tiempo de ejecución, menor prioridad y viceversa.

Notemos que siempre mencionamos prioridad alta o baja. Generalmente la prioridad se asigna a un rango fijo de números, tal como 0 a 7 o 0 a 4095. Sin embargo no hay uniformidad acerca de si 0 es la mayor o menor prioridad. Algunos sistemas usan los números más chicos para denotar una menor prioridad; otros los usan para los trabajos de mayor prioridad. Esto puede llevar a confusión.

Un gran problema con el algoritmo de planificación por prioridades es el bloqueo indefinido o inanición. Un elemento listo para ser ejecutado pero que no

consigue el servidor puede considerarse bloqueado. Un esquema de este tipo puede dejar esperando indefinidamente a la espera del servidor a procesos con baja prioridad.

Una solución al bloqueo indefinido es la técnica conocida como añejamiento: en forma gradual se aumenta la prioridad de un elemento luego de pasado cierto tiempo en espera.

Round-Robin

En este algoritmo de planificación RR se define una pequeña unidad de tiempo, llamado generalmente un "quantum" de tiempo. La cola se trata como una cola circular. Así se recorre la cola asignando el servidor a cada proceso por intervalos máximos de un quantum.

Para implementar RR, a la cola de espera los elementos entran en orden FIFO. Nuevos procesos son agregados al final de la cola. Se toma el primer trabajo de la cola, se le concede el servidor. Pueden ocurrir 2 cosas: el elemento puede tener un tiempo de ejecución menor al quantum, en este caso, el mismo, voluntariamente libera al servidor. Se procede a ejecutar entonces el próximo trabajo de la cola. Caso contrario, se interrumpe la ejecución del elemento, se guardan datos sobre el estado de procesamiento del mismo (conmutación de contexto[1]) y se lo coloca al final de la cola, dando paso al elemento siguiente.

Veamos el mismo ejemplo usado en FCFS:

Elemento	Tiempo de Ejecución
1	24
2	3
3	3

Uso un quantum = 4 unidades de tiempo

Elemento 1	Elem. 2	Elem. 3	Elemento 1	Elemento 1	Elemento 1	Elemento 1	Elemento 1

0 4 7 10 14 18 22 26 30

El tiempo de retorno promedio es: (30 + 7 + 10)/3 = 47/3 = 16 aproximadamente.

En un algoritmo de planificación RR ningún proceso obtiene más de un quantum de tiempo en forma consecutiva.

[1] La operación de conmutación de contexto salva el estado del proceso actual y conmuta a un proceso preparado para ejecución seleccionado mediante la restauración del estado de este nuevo proceso. El estado de un proceso de ejecución lo indican los contenidos de sus registros asociados.

El tiempo que dura una conmutación de contexto es un gasto extra; el sistema no hace nada útil durante la conmutación. El tiempo requerido para la conmutación depende del Sistema.

La performance de un sistema con este esquema, depende fuertemente de tiempo asignado al quantum. Si este es muy grande (infinito), el algoritmo se convierte en FCFS. Si es muy pequeño, el método se llama servidores compartidos: si tengo n procesos, resulta (en teoría) como tener un procesador dedicado a cada proceso con una velocidad de 1/n de la velocidad del servidor real. En este caso el proceso de conmutación de contexto entorpece el sistema.

Simulación de la política de Colas

Para conseguir una evaluación más precisa de algoritmos de planificación, frecuentemente se usan las simulaciones.

Estas implican la programación de un modelo de un computador. Estructura de datos de software representan los principales componentes del sistema. El simulador tiene una variable representando un reloj y a medida que su valor se incrementa, el simulador modifica el estado del sistema para reflejar las actividades de los dispositivos, los trabajos y el planificador. A medida que transcurre la simulación, se guardan e imprimen estadísticas que indican la performance del algoritmo.

Los datos para la simulación pueden generarse de manera diversa. El método más común es usar un generador random de números al azar, que es programado para generar trabajos, tiempos de atención, llegadas, partidas, etc. de acuerdo a probables distribuciones. Las distribuciones pueden ser definidas matemáticamente (uniforme, exponencial, etc.) o empíricamente. Para definir una distribución empírica, se realizan mediciones del sistema bajo estudio. Los resultados pueden así usarse para definir la distribución de eventos en el sistema real. Esta distribución se usa luego para la simulación.

Bibliografía

Extracto de:

- Guido J. Pace (UNNE FCENA). Modelos y simulación,1993.
- Introducción a la Simulación de Eventos Discretos. Gabriel A. Wainer. Departamento de Computación – FCEN, UBA.
- Ing Luis Zuloaga Rotta. Investigación de Operaciones, 2005. UNI FIIS, Peru.