Sistemas Operacionais

Prof. Robson de Souza

Aulas 19 e 20

Conteúdo: Escalonamento de processos (Continuação...)

Escalonamento por prioridades

Esse escalonamento surge da necessidade de se considerar fatores externos. A cada processo é atribuída uma prioridade e, ao processo executável com a prioridade mais alta é permitido executar. Isso pode acarretar problemas, por exemplo, se um processo tiver uma prioridade muito baixa e sempre chegarem novos processos na fila com prioridades mais altas, o processo com baixa prioridade pode nunca executar.

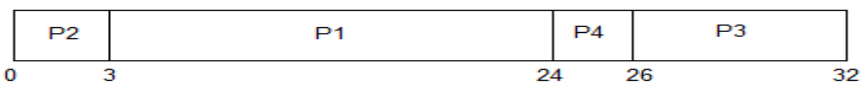
Para evitar esse tipo de problema, uma possível solução seria reduzir a prioridade de um processo em execução a cada interrupção de relógio. Outra possibilidade é atribuir um quantum máximo a um processo, se ele atingir o quantum, terá de dar lugar a outro.

Prioridades podem ser atribuídas aos processos estática ou dinamicamente. O sistema também pode atribuir dinamicamente as prioridades para atingir certos objetivos. Por exemplo, se um processo altamente orientado a E/S quisesse a CPU, deveria recebê-la imediatamente. Fazer um processo orientado a E/S esperar um longo tempo pela CPU significa tê-lo ocupando a memória por tempo demais desnecessariamente. Um algoritmo simples e que funciona bem para processos orientados a E/S é atribuir 1/f à prioridade, sendo f a fração do último quantum que o processo usou. Por exemplo, se o quantum é de 100ms e o processo usou apenas 2ms do seu quantum, ele receberá prioridade 50. Um processo que executou 50ms receberá prioridade 2 e um processo que usou os 100ms receberá prioridade 1.

No escalonamento por prioridades é possível agrupar os processos em classes de prioridades e usar o escalonamento entre as classes e, dentro de cada classe, usar o escalonamento circular. Desse modo o algoritmo funciona de maneira que, se em uma classe de prioridade mais alta existirem processos executáveis, executa-se um por quantum usando escalonamento circular e nunca se perde tempo com as classes mais baixas. Se a classe alta estiver vazia (sem processos para executar), passa para a classe abaixo e assim por diante. Se a classe de prioridade mais alta voltar a ter processos, ela é executada novamente. Obviamente, se as prioridades não forem ajustadas, as classes mais baixas podem nunca executar.

PROCESS	BURST TIME	PRIORITY
P1	21	2
P2	3	1
P3	6	4
P4	2	3

The GANTT chart for following processes based on Priority scheduling will be,



The average waiting time will be, (0 + 3 + 24 + 26)/4 = 13.25 ms

Escalonamento garantido

Esse método faz promessas reais sobre o desempenho aos usuários e os satisfaz. Uma promessa realista e fácil de cumprir seria: “Se houver N usuários conectados, cada usuário receberá cerca de 1/N de CPU”.
Um outro exemplo seria: “Se existem N processos em execução e todos são iguais, cada um deve receber 1/N ciclos de CPU”.

O sistema deve manter o controle sobre a quantidade de CPU que cada processo recebe desde a sua criação. O sistema deve realizar os cálculos necessários e ajustar as taxas dos processos que estão recebendo menos tempo de CPU, colocando eles para executar por mais tempo nas próximas vezes.

Escalonamento por loteria

Fazer promessas aos usuários e satisfazê-los é uma boa ideia, mas difícil de implementar. O escalonamento por loteria é um método mais simples de se implementar. A ideia é dar “bilhetes” de loteria aos processos, cujos prêmios são vários recursos do sistema. Se houver uma decisão de escalonamento, um bilhete de loteria será escolhido aleatoriamente e o processo que tem o bilhete terá o recurso.

Aos processos mais importantes podem ser atribuídos bilhetes extra para aumentar as chances desse processo obter o recurso. Se aparecer um novo processo e alguns bilhetes forem dados a ele, já no próximo sorteio sua chance será proporcional ao número de bilhetes que ele tem.

Processos cooperativos podem trocar bilhetes entre si, se assim desejarem. Esse escalonamento pode ser usado para resolver problemas difíceis de solucionar a partir de outros métodos.

Escalonamento por fração justa (fair-share)

Tendo como pressuposto que cada processo é escalonado por si próprio, sem a preocupação de quem é seu dono. Se um usuário A tiver 9 processos e um usuário B tiver 1, o usuário A recebe 90% da CPU e o B apenas 10%. Para evitar isso, alguns sistemas consideram a propriedade do processo antes de escaloná-lo. Nesse modelo, a cada usuário é alocada uma fração de CPU e o escalonador escolhe os processos de modo que garanta essa fração.

Cada usuário usa apenas a sua fração, sem importar quantos processos tem. Exemplo: 2 usuários, 50% da CPU para cada, o usuário 1 tem 4 processos (A,B,C,D), o usuário 2 tem 1 processo (E). Com escalonamento circular, uma possível sequencia seria:

A E B E C E D E A E B E …

Se for destinado 2 vezes mais tempo de CPU ao usuário 1, podemos ter:

A B E C D E A B E C D E …

Escalonamento em Sistemas de Tempo Real (STR)

Um sistema de tempo real é aquele no qual o tempo tem uma função essencial. O computador deve reagir apropriadamente a estímulos externos dentro de um intervalo de tempo. Se os bits de um CD de música demorarem a ser convertidos em música, a música soará diferente. Outros exemplos: Monitoramento de pacientes, piloto automático, entre outros. Nesses casos, a resposta tardia é tão ruim quanto não ter nada.

Sistema de tempo real crítico → Há prazos absolutos que devem ser cumpridos.

Sistema de tempo real não crítico → Descumprimento ocasional de um prazo é indesejável, mas tolerável.

O comportamento de tempo real é implementado dividindo-se o programa em vários processos cujo comportamento é previamente conhecido. Em geral esses processos tem vida curta e podem executar em bem menos de um segundo.

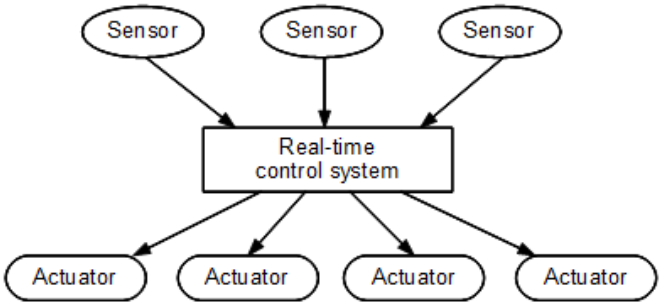
Quando é detectado um evento externo, o trabalho do escalonador é escalonar os processos de tal maneira que todos os prazos sejam cumpridos. Os eventos aos quais um STR pode precisar responder são caracterizados como: Periódicos (Ocorrem em intervalos regulares) e Aperiódicos (Ocorrem de modo imprevisível)

Os algoritmos de escalonamento de tempo real podem ser:

Estáticos → Tomam suas decisões de escalonamento antes de o sistema começar a executar.

Dinâmicos → Tomam suas decisões em tempo de execução.

O escalonamento estático só funciona quando há prévia informação perfeita disponível sobre o trabalho necessário a ser feito e os prazos que devem ser cumpridos. O escalonamento dinâmico não precisa disso.



https://www.tankonyvtar.hu/hu/tartalom/tamop412A/2011-0042_szoftverfejlesztesi_folyamatok_angol/ch10s03.html

Referências bibliográficas:

TANENBAUM, Andrew. 2ª ed. **Sistemas Operacionais Modernos,** Editora Pearson, 2003.

SILBERSCHATZ, Abraham. **Sistemas Operacionais com JAVA,** 6ª ed. Editora Campus

MACHADO, Francis B. **Arquitetura de Sistemas Operacionais,** 4ª ed, LTC, 2007.