Católica SC

Escalonamento de processos

Aluno: Yuri Jovêncio Pereira
Jaraguá do Sul/SC
2023

Escalonamento de processos:

Em um sistema operacional a multiprogramação funciona de maneira em que a CPU fique alternando entre a execução de vários processos, onde as tarefas são executadas em uma ordem de precedência, sendo que cada um dos processos é executado dentro de uma fatia de tempo também conhecido como “time-sharing”. Com o escalonamento de tarefas a alocação de memória e o tempo do processador são melhorados.

A parte do sistema operacional que faz a escolha é chamada de escalonador e o algoritmo utilizado para isso é chamado de algoritmo de escalonamento. O escalonamento de processos é a distribuição do acesso aos recursos de um sistema dentre os processos que o solicitam, otimizando assim o rendimento dos recursos. No escalonamento é decidido sobre que trabalhos serão admitidos pelo sistema, sobre quais processos serão mantidos na memória principal e que processo utilizará a CPU quando ela estiver livre.

Podemos notar a importância do escalonamento por exemplo nos servidores em rede, pois neles é muito comum que múltiplos processos acabem competindo pela CPU sendo necessário escolher quais processos devem ter uma importância maior. O escalonador além de decidir o processo certo que deve ser executado, também deve se preocupar em fazer uso eficiente da CPU.

O escalonamento é necessário em diversas situações, sendo uma delas quando um novo processo é criado, uma decisão precisa ser tomada sobre que processo deve ser executado primeiro, o pai ou o filho, tendo em vista que os dois processos estão em um estado pronto, é uma decisão de escalonamento normal onde qualquer um dos processos podem ser executados, o escalonador pode escolher executar o pai ou o filho. Outra situação é ao término de um processo, esse processo já não pode mais ser executado, sendo necessário a escolha de outro do conjunto de processos prontos. Se não há nenhum processo pronto, então um processo ocioso gerado pelo sistema é executado.

Quando ocorre uma interrupção de E/S, uma decisão de escalonamento pode ser tomada, se a interrupção foi gerada por um dispositivo de E/S que concluiu sua tarefa, algum processo que foi bloqueado esperando pela E/S pode estar pronto para executar. Cabe ao escalonador determinar se deve dar prioridade para o processo recém liberado, ao processo que estava sendo executado no momento da interrupção, ou algum outro processo.

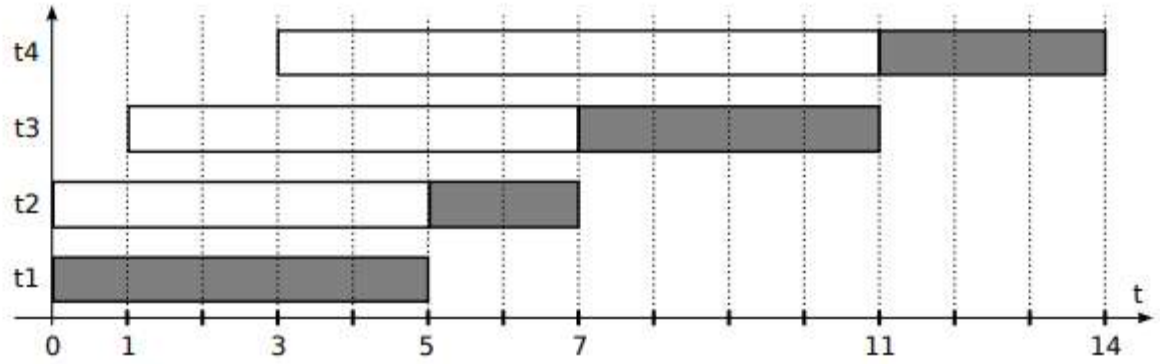
Podemos observar dois cenários de escalonamento, sendo eles o escalonamento não preemptivo e o escalonamento preemptivo. O não preemptivo ocorre apenas em situações onde é praticamente obrigatório que uma decisão seja tomada. Esse cenário possui as condições de criação de um novo processo, término de um processo, bloqueio de processo, após alguma interrupção. Já o escalonamento preemptivo escolhe um processo e concede a ele a CPU durante um certo período de tempo, finalizado esse tempo, a CPU passa a ser de outro processo. Esse cenário tem as seguintes condições: Criação de um novo processo, término de um processo, processo ser bloqueado, após alguma interrupção, periodicamente a cada k intervalos de relógio.

Um exemplo de algoritmo de escalonamento não preemptivo é o FCFS (first-come first-served), ele consiste em atender as tarefas em sequência, à medida em que ficam prontas, ou seja, conforme ordem de chegada na fila de tarefas prontas. Consideremos as

tarefas na fila de tarefas prontas, levando em consideração a duração estimada de processamento e as datas de entrada no sistema, conforme mostrado na tabela abaixo.

tarefa	t1	t2	t3	t4
ingresso	0	0	1	3
duração	5	2	4	3

Na imagem a seguir podemos observar o escalonamento do processador utilizando o algoritmo FCFS cooperativo, o qual não possui quantum ou outras interrupções. Os quadros sombreados indicam o uso do processador, onde apenas uma tarefa ocupa o processador em cada momento. Os quadros em branco representam as tarefas que já entraram no sistema e estão aguardando o processador (tarefas prontas).



https://joaoricardao.files.wordpress.com/2012/07/algoritmos_escalonamento.pdf

Calculando o tempo médio de execução (Tt, a média de tt(ti)) e o tempo médio de espera (Tw, a média de tw(ti)) para o algoritmo FCFS, podemos observar o seguinte:

$$T_t = \frac{t_t(t_1) + t_t(t_2) + t_t(t_3) + t_t(t_4)}{4} = \frac{(5-0) + (7-0) + (11-1) + (14-3)}{4}$$
$$= \frac{5+7+10+11}{4} = \frac{33}{4} = 8.25s$$

$$T_w = \frac{t_w(t_1) + t_w(t_2) + t_w(t_3) + t_w(t_4)}{4} = \frac{(0-0) + (5-0) + (7-1) + (11-3)}{4}$$
$$= \frac{0+5+6+8}{4} = \frac{19}{4} = 4.75s$$

https://joaoricardao.files.wordpress.com/2012/07/algoritmos_escalonamento.pdf

É natural que em diferentes ambientes sejam necessários diferentes tipos de algoritmos de escalonamento, essa situação surge por conta das diferentes áreas de aplicação e de tipos de sistemas operacionais terem metas diversas. Em outras palavras, o que o

escalonador deve otimizar não é o mesmo em todos os sistemas. Podemos destacar três ambientes, sendo eles: Lote, interativo e tempo real.

Os sistemas em lote são amplamente utilizados no mundo dos negócios, para folhas de pagamento, controle de estoque, gerenciamento de contas a receber e contas a pagar, cálculos de juros em bancos, processamento de pedidos de indenização em companhias de seguros, entre outras atividades periódicas. Em sistemas em lote, não há necessidade de os usuários aguardarem ansiosamente em seus terminais por uma resposta imediata a uma solicitação menor. Como resultado, algoritmos não preemptivos ou algoritmos preemptivos com longos intervalos para cada processo são frequentemente aceitáveis. Essa abordagem reduz as trocas de processos e melhora o desempenho. Na verdade, os algoritmos em lote são amplamente utilizados e muitas vezes aplicáveis a outras situações, tornando seu estudo interessante, mesmo para pessoas que não estão envolvidas em computação empresarial de grande escala.

Em ambiente interativo a preempção é essencial para evitar que um processo tome conta da CPU e impeça outros processos de executarem. Mesmo que nenhum processo tenha a intenção de executar indefinidamente, um erro em um programa pode causar um bloqueio e impedir que todos os outros processos sejam executados. A preempção é necessária para evitar esse comportamento indesejado. Servidores também se encaixam nessa categoria, pois eles atendem a múltiplos usuários, que normalmente estão muito apressados, assim como os usuários de computadores.

Em sistemas com restrições de tempo real, a preempção nem sempre é necessária, porque os processos nesses sistemas não podem executar por longos períodos de tempo, normalmente realizam seu trabalho e bloqueiam rapidamente. A diferença em relação aos sistemas interativos é que os sistemas de tempo real executam apenas programas que contribuem diretamente para o progresso da aplicação à mão. Sistemas interativos são de propósito geral e podem executar programas arbitrários que não são cooperativos e até mesmo podem ser maliciosos.

Sobre o algoritmo por prioridades, cada processo possui uma prioridade, onde o processo pronto com maior prioridade ganha a CPU. Um processo de alta prioridade pode manter a CPU pelo tempo que desejar. É possível reduzir a prioridade de um processo em cada interrupção do relógio, ou estabelecer um quantum máximo. A atribuição das prioridades pode ser estática, ou seja, definida previamente e não alterada durante a execução, ou dinâmica, onde as prioridades são ajustadas de acordo com o comportamento e necessidades do sistema operacional.

Sabendo isso podemos imaginar um cenário onde há três programas a serem executados, o Word, o Google Chrome e um compilador C++. Se iniciarmos abrindo o word o sistema operacional irá carregar o programa na memória e criará um processo dedicado a ele, alocando os recursos necessários, como o tempo de CPU e memória. Em sequência ao abrirmos o Google Chrome o processo vai se repetir, carregar o programa na memória e então criar um novo processo para a execução do navegador. Novos recursos serão alocados para esse processo, incluindo espaço de memória adicional para armazenar as páginas da web abertas e outras informações relacionadas ao navegador. Por fim, ao iniciarmos o compilador C++ o sistema operacional carregará o programa na memória e mais uma vez um novo processo será criado, dessa vez para poder executar o compilador, e por último, recursos

adicionais são alocados. Por fim podemos dizer que normalmente o Word e o Chrome, que são aplicativos interativos, terão uma prioridade mais alta em relação ao compilador de C++, por se tratar de um processo que costuma ficar em segundo plano.

Referências:

http://www.univasf.edu.br/~andreza.leite/aulas/SO/ProcessosEscalonamento.pdf

https://afontedeinformacao.com/biblioteca/artigo/read/97706-o-que-significa-multiprogramacao-em-sistemas-operacionais

http://ldemetrio.com.br/Livros/Livros_TI/segunda_unid/Sistemas%20Operacionais%20Modernos%20-%20Tanenbaum%20-%204%20Edi%C3%A7%C3%A3o.pdf

https://joaoricardao.files.wordpress.com/2012/07/algoritmos_escalonamento.pdf