6 ESCALONAMENTO DE CPU

O escalonamento de CPU é ponto chave da multiprogramação. Ela permite que haja mais de um processo em execução ao mesmo tempo. Em ambientes com um único processador, o escalonador realiza o revezamento de uso do processador pelos processos, tornando-o mais eficiente. Muitas vezes, eles têm diferentes fluxos de execução, com uso de diferentes recursos e em diferentes épocas.

6.1 INTRODUÇÃO

O objetivo da multiprogramação é aumentar o índice de aproveitamento da CPU. Ela tenta sempre deixar a CPU ocupada com algum processo. Ele aproveita as operações de I/O, onde a CPU não é utilizada e poderia ficar ociosa para colocar um outro processo em execução. O escalonamento é muito importante para o Sistema Computacional, praticamente qualquer recurso pode ser escalonado.

6.1.1 CICLO DE SURTO DE CPU E I/O

Estas duas características se referem ao fato de que um processo tem intervalos de uso de CPU e I/O. Toda vez que um processo nesta usando a CPU é dito que é um surto de CPU e é um surto de I/O quando ele está em uma operação dessa natureza.

Tais características fazem haver duas categorias:

- Um processo que tem muitas operações de I/O terá muitos surtos curtos de CPU;
- Um programa que usa muito usa muito a CPU terá poucos surtos de I/O, mas serão longos.

6.1.2 ESCALONADOR DE CPU

O escalonador é um processo do Sistema Operacional que seleciona um dos processos que está no estado "PRONTO". Um dos pontos importantes é que nem sempre o primeiro da fila é o primeiro a ser atendido.

6.1.3 ESCALONAMENTO PREEMPTIVO

As decisões de escalonamento de CPU podem ocorrer de quatro maneiras:

1. Quando o processo passa do estado de execução para o estado de espera (bloqueado);
2. Quando um processo passa do estado de execução para o estado pronto;

3. Quando um processo passa do estado bloqueado para o estado pronto;
4. Quando um processo termina.

Nos casos 1 e 4 há obrigatoriamente a troca de processos na CPU. Um novo processo que está no estado pronto deve ser selecionado para usar a CPU. Este tipo de escalonamento é conhecido como cooperativo ou não-preemptivo. Caso contrário, ele é conhecido como preemptivo.

No escalonamento não-preemptivo o processo é mantido na CPU até ele liberar o processador indo para o estado bloqueado ou terminando. Este tipo de escalonamento foi utilizado nas primeiras versões de Windows, até o 98 mais precisamente. O MacOS8 também utilizou este tipo de escalonamento. Neste caso foi utilizado por uma limitação de hardware, pois o escalonamento preemptivo precisa de um timer.

O escalonamento preemptivo tem como uma de suas conseqüências o aumento do custo de implementação do sistema, principalmente no compartilhamento de recursos. Existem outras conseqüências da implementação como, por exemplo:

- Coordenação de filas de requisições de acesso à I/O;
- Coordenação de alterações realizadas pelo kernel do Sistema Operacional;
- Coordenação de interrupções.

6.1.4 DISPATCHER

O dispatcher é o processo que fornece o controle da CPU para o processo selecionado pelo escalonador. Ele deve fazer a mudança de contexto, ou seja, verificar e salvar quais os valores de variáveis, registradores, posição no programa e recursos do processo que está deixando a CPU e acionar o contexto exigido pelo novo processo. O tempo de mudança de contexto é chamado de latência de dispatcher.

6.2 CRITÉRIOS DE ESCALONAMENTO

Existem diversos algoritmos de escalonamento de processadores, com diferentes propriedades. Cada um deles é recomendado para situações diferentes, mas baseados em alguns critérios, os quais estão descritos a seguir:

- Utilização de CPU: a CPU deve ficar o máximo de tempo ocupado. A medida é feita através de percentagem, variando de 0% a 100%. Nos sistemas reais variam entre 40% e 90%;

- *Throughput*: ela também é conhecida como vazão. Ela se refere ao número de processos executados por unidade de tempo. Para longos processos haverá uma baixa vazão, mas para curtos processos deve ser alta.
- Tempo de retorno: refere-se ao tempo total de execução de um processo. Este tempo deve incluir os tempos de espera, acesso à memória e dispositivos de I/O;
- Tempo de espera: é a soma dos tempos que o processo precisou ficar esperando na fila de processos prontos para ser selecionado para ser usado pelo processador;
- Tempo de resposta: existem alguns processos que podem gerar respostas ainda em execução. O tempo de resposta está associado com o intervalo entre o início do processo e a geração da primeira resposta.

O ideal é ter o máximo de uso de CPU e *throughput*, e minimizar o tempo de retorno, resposta e espera. Na maioria dos casos faz-se a média de todos os critérios para otimizar o sistema. Porém, em alguns casos é preciso trabalhar com valores máximos e mínimos, fugindo de valores médios.

6.3 ALGORITMOS DE ESCALONAMENTO

Existem basicamente quatro algoritmos de escalonamento.

6.3.1 FIRST-COME,FIRST-SERVED (PRIMEIRO A CHEGAR, PRIMEIRO SERVIDO)

Este algoritmo FCFS também é conhecido como FIFO (First-In, First-Out – primeiro a entrar, primeiro a sair). Neste caso, o primeiro processo a pedir a CPU, tem direito de usá-la. Os processos que requisitam posteriormente a CPU são alocados em uma fila e são atendidos conforme a ordem de requisição.

Este algoritmo é o mais simples de ser implementado. Porém o tempo de espera ou de resposta pode se tornar demasiadamente grande. Basta imaginarmos uma grande quantidade de processos.

Vamos agora colocar o algoritmo FCFS em uma situação dinâmica. Vamos imaginar um longo processo A com poucas operações de I/O, e muitos processos que fazem muitas operações de I/O. Imagine que A comece a executar, todos ou a maioria dos dispositivos ficarão ociosos, pois os demais processos estão na fila esperando ser selecionados. Se ele é não-preemptivo, todos os processos que usam I/O irão ter que esperar um longo período de tempo para iniciar a sua execução que irá utilizar muito pouco a CPU. Ele não é recomendado para sistemas de tempo compartilhado.

6.3.2 JOB MAIS CURTO PRIMEIRO

Um outro algoritmo para escalonamento de CPU é o do Job mais Curto Primeiro. Ele tenta prever a duração do próximo surto de CPU do processo. Quando a CPU está livre, o escalonador seleciona o processo que, possivelmente, terá o surto de menor duração. Se dois processos tiverem o mesmo tamanho de surto é usado o algoritmo FCFS.

Sua filosofia está próxima de um algoritmo ótimo, porém existem alguns fatores que o tornam uma solução não muito interessante:

- Como prever com exatidão o período do próximo surto de CPU?
- Mesmo se um processo tem um surto maior, ele pode ter uma prioridade maior do que a de outros processos;
- Ele aumenta o tempo de retorno ou de resposta dos processos longos.

Para tentar prever o próximo surto de CPU de um determinado processo ele faz uma análise das últimas chamadas do processo. Espera-se que a duração do surto do processo seja semelhante aos anteriores.

A duração do próximo surto é calculada como a média exponencial das durações medidas anteriores. A base de cálculo é feita da seguinte maneira:

$$A_{tn+1} = A_{tn} + P*A_{tn-1}$$

onde P é o peso das medidas mais antigas.

A sua implementação pode ser preemptiva ou não-preemptiva. Caso ela seja preemptiva, se o novo surto de CPU previsto de um processo for menor que o atual em execução, então ele será executado imediatamente, colocando o processo que estava em execução no estado PRONTO.

6.3.3 ESCALONAMENTO POR PRIORIDADE

Um outro algoritmo para escalonamento de processos é por prioridade. Cada processo é associado a uma prioridade. Os valores maiores indicam os processos de maior prioridade e os menores, os processos de menor prioridade. Existem sistemas operacionais que adotam o padrão inverso. Estes valores, geralmente, são fixos.

De certa forma, o algoritmo do Job Mais Curto Primeiro é um algoritmo de prioridade. Entretanto, o seu parâmetro de análise é o tamanho do surto de CPU do processo.

A prioridade pode ser definida de forma interna ou externamente. Para definir internamente são necessários parâmetros para análise como, por exemplo:

limite de tempo, requisitos de memória, quantidade de arquivos abertos pelo processo e o surto médio de CPU do processo.

Além disso, ele pode ser preemptivo ou não-preemptivo. O peso das prioridades pode ser crescente ou decrescente. Esta característica varia com a implementação do sistema operacional.

6.3.4 ROUD ROBIN

Este algoritmo foi o primeiro a propor uma implementação que simulasse a multitarefa em tempo real. Ele propõe que os processos revezem o uso da CPU através de uma unidade de tempo denominada quantum. Um quantum pode ter duração de 10 a 100 milisegundos, novamente depende da implementação do sistema operacional.

Os processos prontos ficam organizados em uma fila circular. O escalonador fica percorrendo esta fila e revezando a execução dos processos até todos eles acabarem. Esta fila é do tipo FIFO. Neste algoritmo nenhum processo recebe mais de um quantum consecutivo. Dessa maneira, o tempo para o próximo do quantum de um processo será executado é n*q, onde n é o número de processos que há na fila e q é o tamanho do quantum.

O tempo de retorno e de resposta aumenta de acordo com o número de processos na fila. Ele tem uma complexidade maior de implementação, além do fato de que o custo computacional também aumenta, pois há a mudança constante de contexto de processos. Ou seja, no critério de uso da CPU é recomendado que seja considerado o tempo gasto para trocar de processo.

6.3.5 ESCALONAMENTO POR MÚLTIPLAS FILAS

Muitas vezes é interessante ter mais de uma fila, onde cada uma delas aloca processos de características ou prioridades semelhantes. Os processo são permanentemente atribuídos a uma delas. Cada fila tem seu próprio algoritmo de escalonamento, independente dos demais.

Além desses escalonadores, há um escalonador das filas. Cada fila tem prioridade absoluta sobre as filas de menor prioridade, ou pode-se usar o Round-Robin.

Pode ser interessante que processos possam transitar entre as diversas filas. Segundo, suas características ele é realocado em uma fila de maior ou menor prioridade. A construção de um escalonador para múltiplas filas considera:

- O número de filas;
- O algoritmo para escalonamento de cada fila;

- Método para mudar um processo de fila.

6.4 AVALIAÇÃO DE ALGORITMOS

Foram apresentados vários algoritmos para escalonamento de processos. Primeiramente, é possível fazer combinações desses algoritmos e obter modelos híbridos que melhor atendem às necessidades de um sistema operacional. Segundo, quanto mais refinado o algoritmo escolhido, maior será o seu custo computacional e financeiro.

Porém, como selecionar ou especificar um algoritmo de escalonamento? O primeiro passo é especificar os critérios do escalonador e as metas a serem atingidas. Em seguida, escolher um escalonador para ser validado.

6.4.1 MODELAGEM DETERMINÍSTICA

A primeira forma de avaliar um algoritmo é a modelagem determinística, onde toma-se um volume de dados que alimentará o algoritmo. É analisado o comportamento do algoritmo e dos processos. Ela é simples e rápida, porém ela depende de uma série de dados exatos. Conseqüentemente, ele só é exato para aquela entrada de dados. Em situações diferentes, o algoritmo não terá o mesmo desempenho.

A aplicação desse método está em fornecer algoritmos candidatos a adoção. Com conjunto de dados ele pode informar tendências que podem ser analisadas independentemente. Se ela for a única ferramenta para a modelagem do algoritmo de escalonamento, então deve-se ter total conhecimento das situações em que o sistema operacional será empregado.

6.4.2 MODELOS DE FILAS

Como os processos são muitos e variam bastante todos os dias ou períodos de horas, a análise determinística é apenas uma ferramenta para fornecer algoritmos candidatos a escalonadores.

Os principais fatores deterministicos que influenciam no desempenho são os surtos de CPU e de I/O, os quais podem ser modelados através de filas. São utilizadas distribuições de probabilidade que resultam em fórmulas matemáticas para determinar a probabilidade de ocorrência de eventos. Geralmente, a distribuição utilizada é a exponencial, a qual depende somente de sua média.

Os modelos de filas possuem servidores, filas e taxa de chegada de entidades. A CPU é o servidor e a taxa de chegada é a taxa de criação de processos. Esta taxa também obedece a uma distribuição de probabilidade. É importante saber que as distribuições são imprecisas e apenas se aproximam da realidade.

6.4.3 SIMULAÇÕES

Uma vez feita a modelagem determinística e teoria de filas, inicia-se o processo de simulações. São construídos em linguagens de programação, onde estruturas de dados representam os componentes do sistema computacional.

São implementadas as filas com as distribuições de probabilidade ou tenta-se simular um nível de aleatoriedade. Fazer simulações é um processo caro, que requer tempo e especialistas em modelagem de filas.

6.4.4 IMPLEMENTAÇÃO

A única maneira de obter 100% de confiabilidade é a implementação real do escalonador. Ele permite detectar as limitações e ajustar os parâmetros para conseguir uma melhor adequação.

Porém ainda há uma série de considerações:

- Possui um alto custo;
- O ambiente é mutável, então os ajustes dificilmente se adequam sempre.