Fundamentos de Sistemas Operacionais I

Prof. Me. Paulo Sérgio Germano

Sistemas Interativos
Escalonamento com Agendamento Garantido

Uma abordagem completamente diferente para o agendamento é fazer promessas reais aos usuários sobre o desempenho, e cumprir as promessas.
Uma promessa realista simples é: Se n usuários estiverem logados enquanto você estiver trabalhando, você receberá cerca de 1 / n recursos da CPU. Da mesma forma, em um sistema de usuário único com n processos em execução, sendo todos iguais, cada um deve receber 1 / n ciclos da CPU.

Para cumprir essa promessa, o sistema deve acompanhar a quantidade de CPU que cada processo teve desde a sua criação. Em seguida, calcula a quantidade de CPU a que cada um tem direito, ou seja, o tempo desde a criação dividido por n. Como a quantidade de tempo de CPU que cada processo realmente teve também é conhecida, é simples calcular a razão entre o tempo real da CPU consumido e o tempo de CPU autorizado.

Sistemas Interativos
Escalonamento por Loteria (Waldspurger e Weihl, 1994)

A idéia básica é: fornecer tickets de loteria de processos para vários recursos do sistema, Ex.: tempo de CPU. Sempre que uma decisão de agendamento tiver que ser feita, um ticket de loteria é escolhido aleatoriamente e o processo que contém o ticket obtém o recurso. Quando aplicado ao agendamento de CPU, o sistema pode armazenar uma loteria 50 vezes por segundo, com cada vencedor recebendo 20 ms de tempo de CPU como prêmio.

Processos mais importantes podem receber tickets extras para aumentar suas chances de ganhar. Em contraste com um escalonador de prioridades aqui a regra é clara: um processo contendo uma fração f dos tickets obterá cerca de uma fração f do recurso em questão.

Sistemas Interativos
Escalonamento por Divisão-Justa

Leva em consideração qual usuário possui um processo antes de agendá-lo. Neste modelo, cada usuário recebe uma fração da CPU e o scheduler escolhe os processos de maneira a aplicá-los. Assim, se dois usuários receberem 50% da CPU, cada um receberá isso, independentemente de quantos processos existirem. Como exemplo, considere um sistema com dois usuários, cada um dos quais foi prometido 50% da CPU. O usuário 1 tem quatro processos, *A*, *B*, *C* e *D*, e o usuário 2 tem apenas um processo, *E*. Se o agendamento de round-robin for usado, uma possível sequência de agendamento que atenda a todas as restrições é a seguinte:

AEBECEDEAEBECEDE ...

Por outro lado, se o usuário 1 tiver o dobro do tempo de CPU que o usuário 2, poderemos obter:

ABECDEABECDE ...

Escalonamento em sistemas de Tempo Real

Nesses casos, ter a resposta certa tarde demais, geralmente é tão ruim quanto não tê-la.

Os sistemas em tempo real geralmente são categorizados como:

- **Tempo real difícil (Hard Real Time)** - significa que há prazos absolutos que devem ser cumpridos

- **Tempo real suave (Soft Real Time) -** significando que perder um prazo ocasional é indesejável, mas mesmo assim tolerável.

Em ambos os casos, o comportamento em tempo real é obtido pela divisão do programa em vários processos, cada um dos quais é previsível e conhecido antecipadamente.

Escalonamento em sistemas de Tempo Real

- Esses processos são geralmente de curta duração e podem ser concluídos em um segundo.
- Quando um evento externo é detectado, é o trabalho do scheduler agendar os processos de forma que todos os prazos sejam cumpridos.

Escalonamento em sistemas de Tempo Real

Os eventos que um sistema em tempo real pode ser classificados como:

- **Periódicos** - significa que ocorrem em intervalos regulares;
- **Aperiódicos** - ocorrem imprevisivelmente.

Um sistema pode ter que responder a vários fluxos de eventos periódicos. Dependendo de quanto tempo cada evento requer para processamento, o manuseio de todos eles pode nem ser possível. Por exemplo, se houver m eventos periódicos e o evento i ocorrer com o período Pi e exigir Ci segundos de tempo de CPU para manipular cada evento, a carga poderá ser tratada apenas se

$$\sum_{i=1}^{m} \frac{Ci}{Pi} \leq 1$$

Um sistema em tempo real que atende a esse critério é considerado escalonável. Isso significa que pode realmente ser implementado.

Escalonamento em sistemas de Tempo Real
Política x Mecanismo

As vezes ocorre que um processo tem muitos filhos correndo sob seu controle. Ex.: um processo de sistema de gerenciamento de banco de dados pode ter muitos subprocessos (filhos).

Cada subprocesso pode estar trabalhando em um pedido diferente, ou cada um pode ter alguma função específica para executar (análise de consulta, acesso ao disco, etc.).

A solução para este problema é separar o mecanismo de scheduler da política de scheduler, onde o algoritmo de scheduler é parametrizado por processos do usuário.

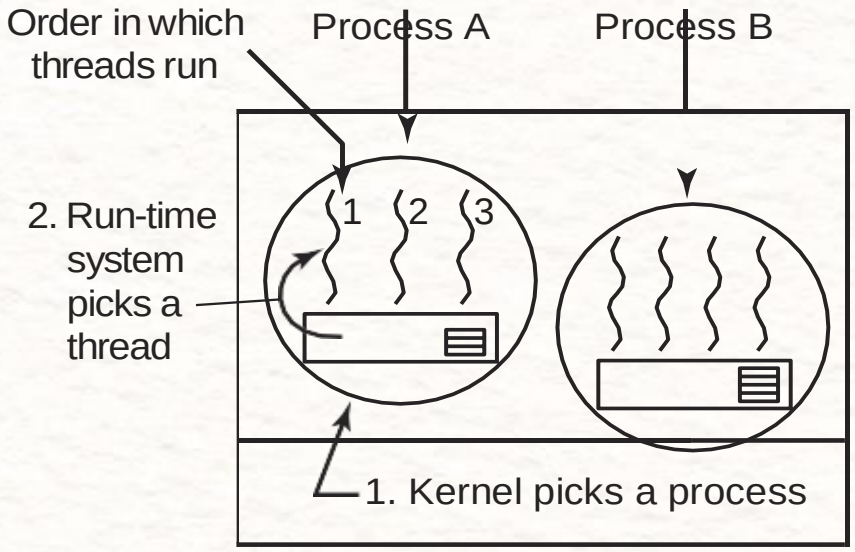
Desta forma, o pai pode controlar como seus filhos são agendados, embora ele próprio não faça o agendamento. Aqui o mecanismo está no kernel mas a política é definida por um processo do usuário.

Escalonamento em sistemas de Tempo Real
Thread Scheduling

Quando vários processos têm múltiplas threads, temos dois níveis de paralelismo presentes: processos e threads.

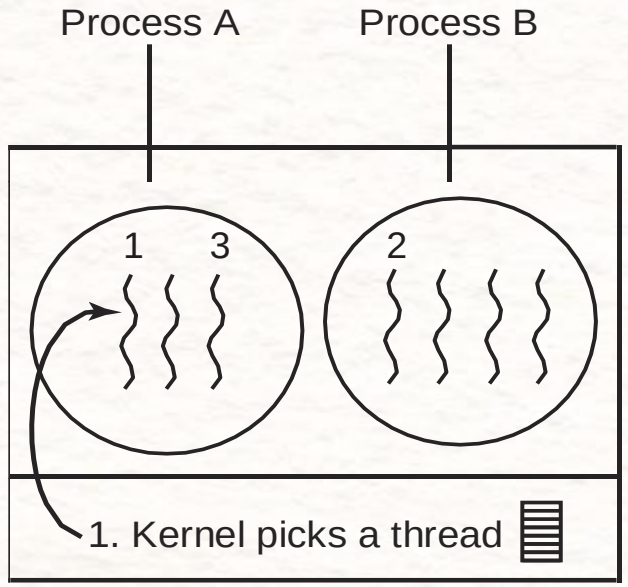
O agendamento nesses sistemas difere substancialmente dependendo dos encadeamentos no nível do usuário ou os encadeamentos no nível do kernel (ou ambos) são suportados.

Escalonamento em sistemas de Tempo Real
Thread Scheduling



Possible: A1, A2, A3, A1, A2, A3
Not possible: A1, B1, A2, B2, A3, B3

(a) Possível escalonamento de threads no nível do usuário com um quantum de processo de 50 mseg e threads que executam 5 ms por estouro da CPU.



Possible: A1, A2, A3, A1, A2, A3
Also possible: A1, B1, A2, B2, A3, B3

(b) Possível escalonamento de threads no nível do kernel com as mesmas características de (a)

Escalonamento em sistemas de Tempo Real
Thread Scheduling

Os algoritimos usados pelo Scheduler em sistemas de Tempo Real são os mesmos já discutidos anterioremente, sendo os algoritimos round-robin e agendamento por prioridade, os mais utilizados.

- Em geral, nestes casos, aplicações específica de thread schedulling a nível de usuário permitem ajustar melhor a programação, do que a nível de kernel.

Pesquisas sobre Processos e Threads

Poucos avanços sobre estes temas, além dos discutidos anteriormente, foram realizados nos últimos anos, tornando grande parte do que discutimos aqui, padrões de implementação, com pequenas alterações em suas características, entre os principais Sistemas Operacionais do mercado.

Referências Bibliográficas

- TANENBAUM, Andrew S., BOSS, Herbert. **Sistemas Operacionais Modernos,** Pearson - 4ª ed., 2016.
- SILBERSCHATZ, A., GALVIN, P.B., GAGNE, G. **Fundamentos de Sistemas Operacionais**, Ed. LTC, 8ª ed., 2011
- DEITEL, H.M.; DEITEL, P.J.; CHOFFNES, D.R. **– Sistemas Operacionais**. Prentice Hall, Tradução da 3ª ed., 2005
- DEITEL, H.M.; DEITEL, P.J. **– C How to Program**. Prentice Hall, Tradução da 3ª ed., 2001
- MIZRAHI, Victorine Viviane. **Treinamento em Linguagem C – Curso Completo módulos 1 e 2**, Ed. Person Education.