MAC0422 - Sistemas Operacionais

Daniel Macêdo Batista

IME - USP, 24 de Setembro de 2020

Roteiro

Escalonamento de processos

Escalonamento de processos

Tipos de escalonamento

Antes de continuar, uma classificação dos tipos de escalonamento

- Os algoritmos de escalonamento podem ser organizados em classes
- Escalonamento em lote

  Tarefas periódicas sendo realizadas. Faz mais sentido não preempção ou preempção com altos intervalos de tempo
- Escalonamento interativo

  Para programas genéricos é importante para evitar monopólio da CPU por um processo (ou intencional ou por bug). Faz mais sentido ter preempção
- Escalonamento em tempo real

  O tempo é conhecido. A depender do sistema nem precisa de preempção

Objetivos do escalonamento

- Em todos os sistemas: **justiça**, garantia da política e balanceamento
- Lote: **vazão**, **tempo de relógio (fim - submissão)**, utilização da CPU
- Interativo: **tempo de resposta**, proporcionalidade (ao que o usuário espera)
- Tempo real: **atender os prazos**, previsibilidade

SJF (Em lote)

- Shortest Job First
- Tenta ser mais justo do que o FCFS (lembrando que justiça é relativo)
- Ordena os processos prontos em uma fila por ordem do tempo de execução deles. Do mais curto para o mais longo e executa nessa ordem
- Cada processo roda até o final. Não há preempção
- Precisa saber o tempo total de execução do processo e o ideal é que todos os processos cheguem no mesmo momento

SRTN (Em lote)

- Shortest Remaining Time Next
- Versão com preempção do SJF
- Quando um novo processo chega, o tempo de execução dele é comparado com o tempo que falta do processo que está executando. Se o novo processo é mais curto, ele passa a executar e o atual vai pra fila de prontos para continuar sua execução depois
- Apesar de haver preempção, ocorre apenas no evento de chegada de um novo processo
- Precisa saber o tempo total de execução do processo

Round Robin (Interativo)

- A cada processo é dado um intervalo de tempo para ele executar (quantum)
- Se ao fim do quantum o processo ainda está executando (ele pode ter sido bloqueado antes por E/S), ele é suspenso e dá a chance ao próximo para executar
- Se houver bloqueio por E/S, nesse momento o próximo é executado
- O tamanho do quantum precisa ser bem definido. Quantum muito baixo aumenta o overhead do escalonador. Quantum muito alto reduz a interatividade notada pelos usuários

Escalonamento com prioridade (Interativo)

- Similar ao Round Robin mas agora cada processo possui uma prioridade associada
- Processos com mais prioridade ganham mais quantum que processos com menos prioridade ou começam com uma prioridade grande e vão diminuindo. Quando ficar menor que o próximo da fila, dá espaço pro próximo da fila.
- No GNU/Linux, a prioridade é um número inteiro que varia de -20 (mais prioridade) para +19 (menos prioridade) e pode ser atribuída no shell com os comandos `nice` ou `renice`. Em C, há a chamada de sistema `nice`. Colocar -20 em uma compilação demorada tem efeitos perceptíveis!
- Existem várias ideias para usar os números das prioridades. Importante ter cuidado para não causar *starvation* (Exemplo com classes de processos)

Múltiplas filas (Interativo)

- Agora os processos podem receber diferentes classificações
- Cada classe associada a uma fila
- Processos mais importantes $\rightarrow$ uma classe maior
- Classe maior $\rightarrow$ mais quantums
- Os processos começam na menor classe (por exemplo 1 quantum). Se não terminar de executar, da próxima vez roda mais (por exemplo 2 quantum), e assim sucessivamente. Desse jeito tem menos mudanças de contexto do que se fosse um quantum fixo para todos e dá chance de processos rápidos rodarem mesmo quando tem processos muito grandes em execução