2.3 Escalonamento de processos

Escalonador de processos

- Escolhe o processo que será executado
- Usa algoritmos ou políticas de escalonamento
- Tenta otimizar a utilização das CPUs
- Usa critérios e objetivos

Critérios

- **Utilização do processador:** eficiência do uso da CPU mantendo o processador ocupado na maior parte do tempo.
- ***Throughput:*** maximizar a produtividade (*throughput*), executando o maior número de processos em função do tempo.
- **Tempo de processador:** tempo de execução do processo.
- **Tempo de espera:** reduzir o tempo total que um processo aguarda na fila para ser executado.
- **Tempo de *turnaround:*** minimizar o tempo que um processo leva desde sua criação até seu término, considerando a alocação de memória, tempo de espera e tempo do processador e aguardando as operações de entrada/saída.
- **Tempo de resposta:** reduzir o tempo de resposta para as aplicações interativas dos usuários.

Objetivos

Dar privilégios para aplicações críticas.

- Balancear o uso da CPU entre processos.

- Ser justo com todos os processos, pois todos devem poder usar o processador.

- Maximizar a produtividade (*throughput*).

- Proporcionar menores tempos de resposta para usuários interativos.

Escalonadores

- Diferentes SOs usam escalonadores distintos
- SO de tempo real
  - Prioriza as aplicações críticas
- SO de tempo compartilhado
  - Aloca os processos de forma uniforme à CPU (tempos iguais)
    - Processos não esperam muito tempo para executar

Modos

- Modo usuário
  - Os aplicativos operam neste modo
- Modo núcleo
  - Usado nos componentes principais do SO

Alternar processos é oneroso

- O estado do processo e o mapa de memória devem ser salvos
- Armazenando os dados dos registradores na tabela de processos
- A cada troca de processos a memória cache é invalidada

Situações que levam ao escalonamento

- Criação de um novo processo
- Término de um processo
  - Ao ser finalizado, um outro precisa ser executado
- Bloqueio do processo
- Interrupção de entrada / saída
- Interrupções de relógio (“clock”)

Tratamento das interrupções de relógio

- Algoritmos e políticas de escalonamento
  - Não-preemptivo
    - Um processo executa até finalizar
      - Ou até que seja bloqueado
        - Ex.: aguardando I/O de outro processo
  - Preemptivo (que pode ser antecipado)
    - Mais complexo
    - Tempo pré-determinado
      - Qdo chega no tempo, troca para outro processo
    - Aplicações em tempo real priorizadas em função dos tempos dados aos processos
    - Permite a implantação de vários critérios de escalonamento

Ambientes de escalonamento

- Não-preemptivo (NP) e preemptivo (P)
- Lote, interativo e tempo real
- Lote (NP ou P)
  - Algoritmos de escalonamento
    - FIFO (NP)
    - Job mais curta primeiro (SJF – *shortest job first*) (NP)
    - Interativo (P)

Lote

- FIFO (NP)
  - Os processos são inseridos em uma fila à medida que são criados, e o primeiro a chegar é o primeiro a ser executado
- Job mais curto primeiro (SJF – *shortest job first*) (NP)
  - São conhecidos todos os tempos de execução dos jobs
  - Os mais curtos são executados primeiro
  - Recomendado quando todos os jobs já estão na fila de execução
  - Versão P: algoritmo próximo de menor tempo restante
    - Qdo entra um novo job na fila de execução
      - O seu tempo é comparado com o tempo restante do atual
        - Se for menor é executado, e o atual espera

Interativo (P)

- Os algoritmos também podem ser aplicados a sistemas em Lote
- Tipos de escalonamento:
  - Round Robin
  - Por prioridades
  - Garantido
  - Por loteria
  - Fração justa (fair-share)

Interativo - Round Robin

- Também conhecido com algoritmo de escalonamento circular
- Algoritmo simples e muito usado
- Processos organizados em fila circular
  - Cada um recebe um intervalo de tempo máximo (quantum)
- Se ao final de seu quantum o processo ainda estiver executando, a CPU é liberada para outro processo
  - E é colocado no final da fila
- O quantum varia de acordo com o SO, geralmente 10 -100ms
- Não permite que um processo monopolize a CPU

Interativo - por prioridades

- Associa prioridade e tempo máx. a um processo
- Qdo os processos estão à disposição para execução, o de maior prioridade é selecionado
  - A cada interrupção de relógio a prioridade é reduzida
    - Para que não execute indefinidamente
- A prioridade de execução pode ser classificada em
  - Estática
    - Não altera durante a existência do processo
  - Dinâmica
    - Ajusta-se de acordo com os critérios do SO

Interativo - garantido

- Se existirem vários usuários (n) logados em uma máquina, cada um deles receberá 1/n do tempo total da CPU
- O sistema gerencia a quantidade de tempo de CPU de cada processo desde sua criação

Interativo - por loteria

- Baseado em distribuir bilhetes aos processos e os prêmios recebidos por eles são recursos de sistema, incluindo tempo
- Cada bilhete pode representar o direito a um quantum de CPU
- Cada processo pode receber diferentes números de bilhetes, com opções de escolha distintas
- Também existem as ações como compra, venda, empréstimo e troca de bilhete

Interativo - fração justa (fair-share)

- Cada usuário recebe uma fração da CPU
- Por exemplo, se existem dois usuários conectados em uma máquina e um deles tiver nove processos e o outro tiver apenas um, não é justo que o usuário com o maior número de processos ganhe 90% do tempo da CPU
- Logo, o escalonador é o responsável por escolher os processos que garantam a fração justa.

Tempo real

- Tempo é importante e processos executam dentro do tempo e são bloqueados, dando oportunidade para outros
- O escalonamento por prioridades seria o mais adequado em sistemas de tempo real
  - Uma prioridade é vinculada ao processo
  - A importância das tarefas na aplicação são consideradas
- A prioridade deve ser estática
- Não existe fatia de tempo para cada processo executar
  - São priorizados com base em seus prazos de conclusão, critérios de importância, eventos e interrupções

Escalonamento de Threads

- Depende se estão no espaço do usuário ou do núcleo
- Nos threads de usuário o núcleo não sabe de sua existência
- O processo usa o seu escalonador
- Thread de núcleo se assemelha a processo
- Ex.: na implementação de threads em Java, cada thread recebe uma prioridade
  - O escalonador em Java garante que o thread com prioridade maior execute o tempo todo
    - Caso exista mais de um thread com prioridade alta, eles serão executados através de alternância circular