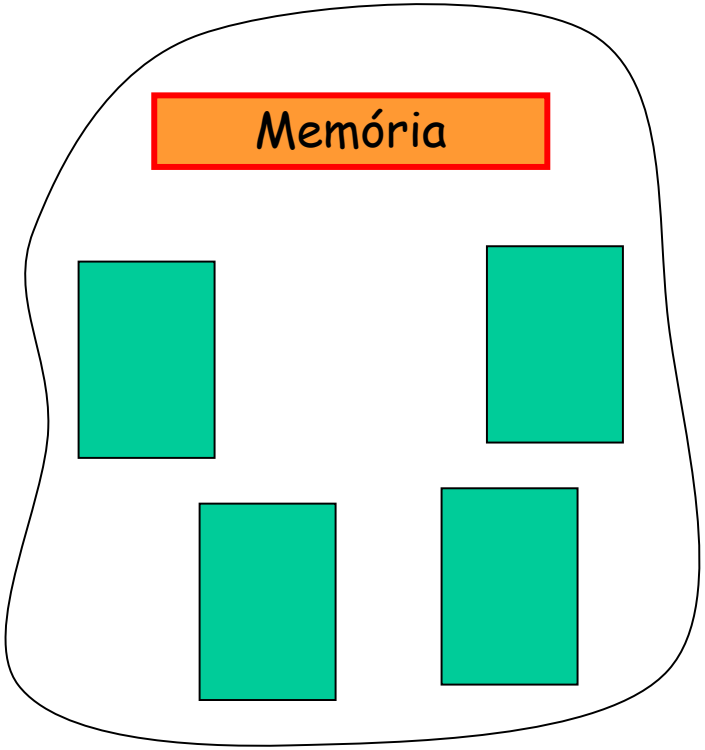
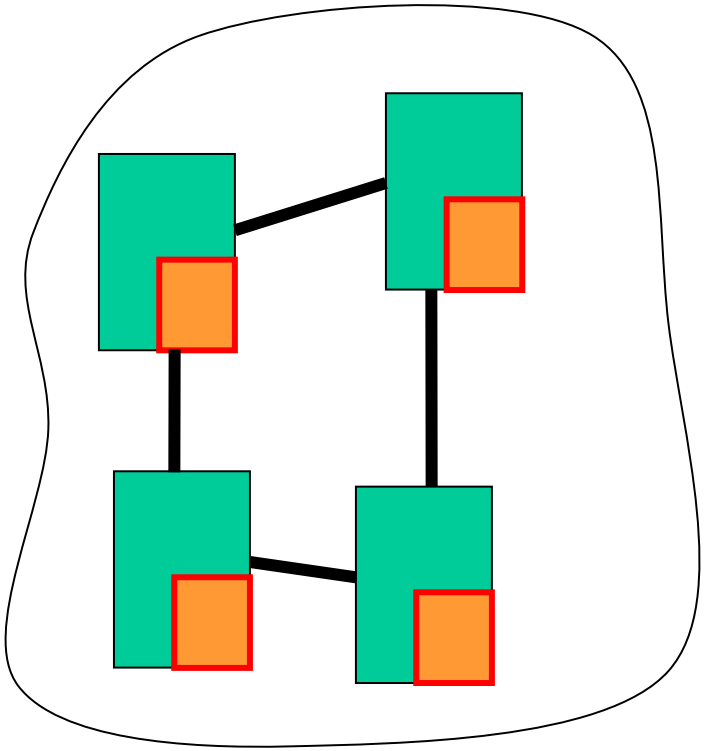
Escalonamento de Processos em Multiprocessadores

Capítulo 10

Processamento Paralelo e Distribuído

Multiprocessadores X Multicomputadores





Processamento Paralelo e Distribuído

- fracamente acoplados
  - geralmente um conjunto de computadores conectados de alguma forma
  - processadores especializados – servidores
  - cluster de processadores
- fortemente acoplados
  - memória compartilhada por vários processadores

Tipos de Paralelismo

- várias aplicações (*jobs*) podem ser executados em paralelo
  - processos independentes
- uma aplicação paralela constituída de vários processos para solucionar um problema
  - processos se comunicam para trocar resultados parciais do problema

Tipos de Paralelismo

- granularidade/granulosidade
  - relação entre a carga computacional e de comunicação
- paralelismo de granularidade grossa
  - processos se comunicam raramente
- paralelismo de granularidade média
  - exemplo: aplicação = conjunto de *threads*
- paralelismo de granularidade fina
  - muita comunicação entre processos

Tipos de Paralelismo

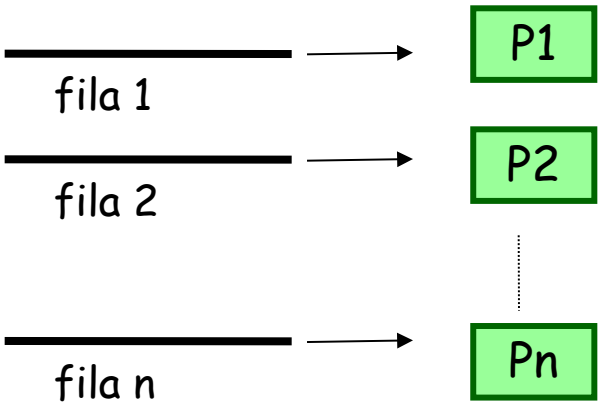
- granularidade/granulosidade
  - relação entre a carga computacional e de comunicação
- paralelismo de granularidade grossa
  - processos se comunicam raramente
- paralelismo de granularidade média
  - exemplo: aplicação = conjunto de *threads*
- paralelismo de granularidade fina
  - muita comunicação entre processos

Escalonamento de Processos

Estático

- conhecimento de características associadas às aplicações antes da execução desta (estimativas)
- relação de precedência entre os componentes da aplicação ou aplicações independentes

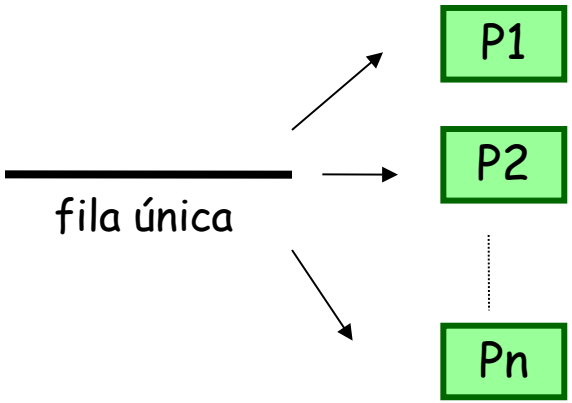
- uma fila por processador
- processador pode ficar ocioso (se estimativas não forem corretas)



Escalonamento de Processos

Estático

- fila global para evitar ociosidade de processador
- contexto do processo disponível para todos os processadores (depende muito do projeto)



Escalonamento de Processos

Dinâmico

- estimativas são conhecidas antes da execução e não as características reais.
- a especificação do escalonamento é feita ao longo da execução da aplicação
  - balanceamento de carga, por exemplo

Onde está o escalonador

Abordagem mestre-trabalhador (cliente-servidor)

- mestre escalona os processos (trabalhadores)
  - localiza-se em um dos processadores
- trabalhadores requisitam necessidades ao mestre
- Problemas
  - se o mestre falhar
  - mestre é um gargalo
  - bom limitar serviços oferecidos pelo mestre

Onde está o escalonador

Abordagem distribuída (arquitetura peer-to-peer)

- um escalonamento distribuído entre processadores
- cada processador escalona processos de um *pool* de processos
- sincronização complicada
  - dois escalonadores não podem escolher um mesmo processo para ser escalonado em algum processador
  - escalonadores tem que trocar conhecimento

Escalonamento de Processos em vários processadores

Diferentes questões devem ser analisadas:

- uma fila com todos os processos prontos
- um conjunto de filas de diferentes prioridades

Exemplo

S1 - sistema com um processador

S2 - sistema duo-processador

- Supor: taxa de processamento de cada processador em S2 = $\frac{1}{2}$ taxa processamento de S1

Escalonamento de Processos em vários processadores

- comparação entre FCFS e round-robin, visto que
  - quantum bem maior que tempo de troca de contexto
  - quantum com valor pequeno quando comparado ao tempo médio de serviço dos processos
- resultado da análise: depende do **coeficiente de variação**

$$C_s = \frac{\sigma_s}{\overline{T}_s}$$

- σ_s - desvio padrão da taxa de serviço
- $\overline{T}_s$ - tempo médio de serviço
- C_s pode ser alto – se muita variação entre os tempos de serviço

Escalonamento de Processos em vários processadores

- FCFS em S2 é amenizado:
  - enquanto um processo longo que chegou primeiro está sendo executado por um processador, outros processos são executados no outro
  - pode ser tão bom quanto *round-robin*, principalmente se o número de processadores aumentar

Escalonamento de *Threads*

- processo = conjunto de *threads*
- em um processador
  - *threads* são vantajosas devido a E/S
- em vários processadores
  - dividir a funcionalidade do processamento entre processadores leva a um maior ganho
- *threads* + multiprocessamento = exploração do grau de paralelismo da aplicação
  - granularidade fina → paralelismo não tão vantajoso
  - alto grau de interação entre *threads*

Escalonamento de *Threads*

Compartilhamento de carga

- uma fila global de *threads* e vários processadores
- processador ocioso executa um *thread* pronto
- Algumas políticas
  - FCFS
  - jobs de menor número de *threads*: prioridade dada aos *threads* de um *job* de menor número de *threads*
  - de menor número de *threads* preemptivo

Escalonamento de *Threads*

Compartilhamento de carga

- FCFS
  - cada *thread* de um *job* é inserida ao final da fila global
  - quando ocioso, o processador seleciona o primeiro *thread* pronto da fila
  - não preemtivo

Escalonamento de *Threads*

Compartilhamento de carga

- jobs de menor número de *threads*
  - fila global é uma lista ordenada por prioridade: um *job* de menor número de *threads* é prioritário
  - não preemptivo
- jobs de menor número de *threads* preemptivo
  - mesmo esquema anterior, mas quando um job com menor número de *threads* chega, os *threads* de um job menos prioritário que estão sendo executados são interrompidos
  - custo de gerenciamento pode ser alto

Escalonamento de *Threads*

Compartilhamento de carga

- estudos mostraram que FCFS é o melhor caso
- é a política mais comumente utilizada

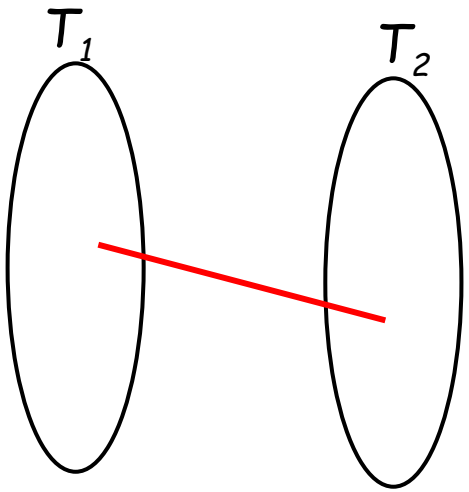
Desvantagens de compartilhamento de carga

- fila centralizada acessada por exclusão mútua
- quanto mais processadores, pior o gargalo
- *threads* preemptivas podem ser executadas em diferentes processadores: dados temporários tem que ser copiados entre as respectivas *caches*

Escalonamento de *Threads*

Gang Scheduling

- escalonamento em grupos de processos nos processadores
- *threads* no grupo geralmente se relacionam (comunicação)



Escalonamento de *Threads*

Gang Scheduling

- se executados em paralelo
  - custos relacionados a sincronização, troca de processos podem diminuir
  - uma decisão de escalonamento para um grupo, logo, menos decisões são tomadas pelo escalonador
  - os *threads* correlatos tem que ser identificados
  - memória compartilhada
- similar a co-escalonamento
  - aplicados a *threads* pequenos
  - em cluster de processadores

Escalonamento de *Threads*

Gang Scheduling

- menor custo de troca de processos
- T1 e T2 são duas *threads* de um mesmo processo
- T1 está sendo executada: precisa sincronizar com T2
- T1 fica em espera até que T2 seja executado em algum processador
  - seria melhor T2 já estar executando em outro processador

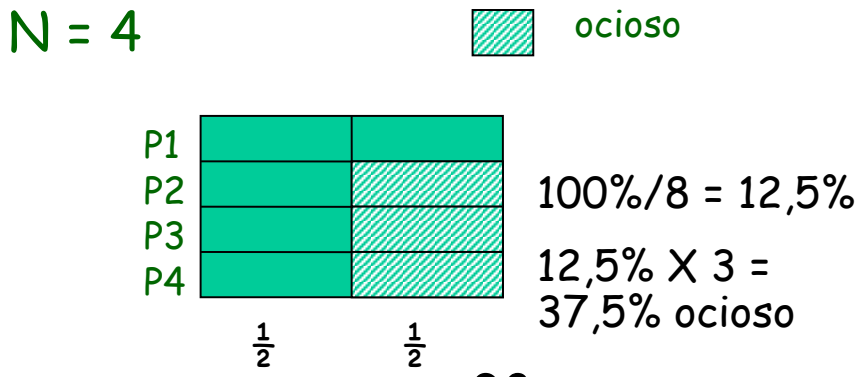
Escalonamento de *Threads*

Gang Scheduling

- N processadores, M aplicações com N ou menos *threads* cada aplicação
  - cada aplicação poderia receber 1/M de fatia de tempo, utilizando os N processadores
  - nem sempre isso é eficiente

A1	4 threads
A2	1 thread

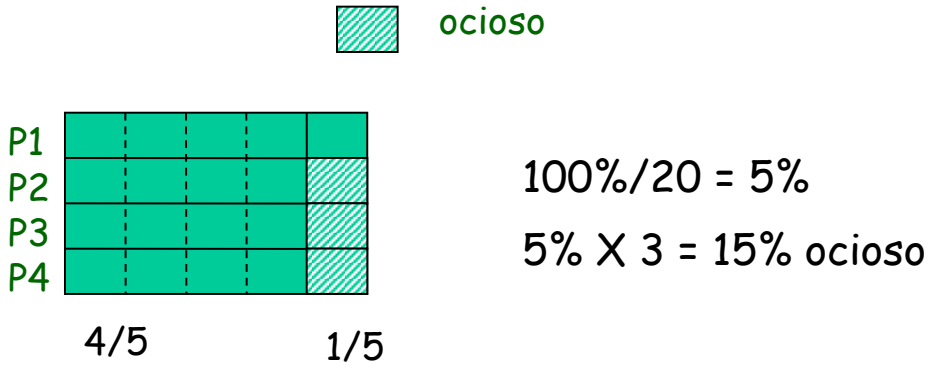
  - alocação de tempo uniforme:



Escalonamento de *Threads*

Gang Scheduling

- uma solução: dar pesos aos processos de acordo com o número de *threads*
  - considerando todos os processos, A1 tem 4/5 das *threads*



100%/20 = 5%

5% X 3 = 15% ocioso

Escalonamento dinâmico

- especificação da alocação durante a execução
- preempção pode ser permitida para migração de processos/threads
- pode acontecer
  - migração de código (custos associados, guardar contexto para execução em outro processador)
  - modificação da alocação entes do início da execução
  - mapeamento de tempos em tempos
- escalonamento nos processadores realizado de acordo com a carga disponível nos processadores

Escalonamento em Tempo Real

- aspectos que podem ser considerados:
  - existe uma análise do comportamento do sistema
  - a análise pode ser estática ou dinâmica
  - de acordo com a análise, o escalonamento é produzido

- esses aspectos podem ser considerados em escalonamento de aplicações de tempo real

Escalonamento em Tempo Real

análise estática

- determina a alocação das tarefas em tempo de execução
- não necessariamente determina o escalonamento, mas as prioridades entre as tarefas
- o sistema pode ser preemptivo, realocar, de acordo com as essas prioridades

análise dinâmica

- considerando as restrições do sistema para atingir o melhor desempenho
- *deadlines* das tarefas são considerados
  - se um processo não atinge seu *deadline* é abortado

Escalonamento em Tempo Real

análise estática

- bom para aplicações que são executadas repetidamente
- problema: estimativas precisas

análise estática, que determina prioridades

- uma análise a priori pode auxiliar na especificação da importância das tarefas
- *deadlines* podem ser considerados

dinâmica

- quando o sistema tem um caráter dinâmico, com tarefas chegando aleatoriamente
- deadlines devem ser obedecidos

Exemplos de Sistema de Tempo Real

- laboratórios de controle
- robótica
- trafego aéreo
- telecomunicações
- sistemas de controle

Sistema de Tempo Real

- as tarefas ou processos podem ser urgentes ou não
- associado a cada tarefa
  - tempo de fim
  - deadline
- hard real time task: deadline tem que ser respeitado

X

- soft real time task: deseja-se atingir o deadline

Características de SO para Sistema de Tempo Real

- tempo de resposta de seus serviços
  - tempo contabilizado a partir de sua requisição
  - depende:
    - tempo da rotina de tratamento de interrupção
    - iniciar a rotina do serviço (troca de contexto)
    - tempo de execução do serviço
    - mais interrupções, se ocorrerem

Características de SO para Sistema de Tempo Real

- Controle do usuário
  - nestes sistemas, o usuário tem maior interação com o SO para alimentar dados da aplicação a ser executada
- Confiabilidade
  - muito importante em Sistemas de Tempo Real
  - falhas devem ser tratadas de forma transparente ao usuário
  - tolerâncias a falhas via software
  - salvamento de dados para recuperação de processo
    - manter arquivos de entrada, etc

Características de SO para Sistema de Tempo Real

Confiabilidade

- com mecanismos de tolerância a falhas, problemas podem ser contornados e a execução continua
- esta operação pode ter *estabilidade*
  - quando ainda com falha, os deadlines são atingidos

Características de SO para Sistema de Tempo Real

Escalonador de curto prazo – papel crucial

- importante que as tarefas críticas (*hard real time tasks*) sejam executadas não ultrapassando deadlines
- o máximo de tarefas não críticas devem ser executadas

Maioria de Sistemas de Tempo real

- dificuldade de atingir deadlines
- quando um deadline está para ser atingido, a tarefa é rapidamente escalonada

Escalonamento baseado em Deadlines

Nos sistemas de tempo real, as seguintes informações são utilizadas:

- tempo que o processo/tarefa fica pronta
  - caso de tarefas periódicas – esta seqüência de tempos pode ser pré-conhecida
- deadline de início e de fim
  - tempo que uma tarefa tem que (a partir do qual deve) começar e tempo que a tarefa deve estar terminada
- tempo de processamento
  - em caso de desconhecimento, o SO pode usar algum modelo de previsão

Escalonamento baseado em Deadlines

- conjunto de recursos requisitados pelas tarefas (sem ser processadores)
- prioridade
  - tarefas críticas podem ter prioridade absoluta (tem que ser executadas o mais rápido)
    - caso de falha: sistema aborta
    - se executado de qualquer modo, prioridades são reavaliadas
- estrutura da tarefa
  - uma tarefa pode ser um conjunto de subtarefas, algumas sendo críticas e outras não

Escalonamento baseado em Deadlines

Seleção e decisões

- qual tarefa deve ser a próxima a ser escalonada
- não preempção
  - quando deadline de início devem ser utilizados
- preempção
  - mais apropriado quando tarefas tem deadlines de fim

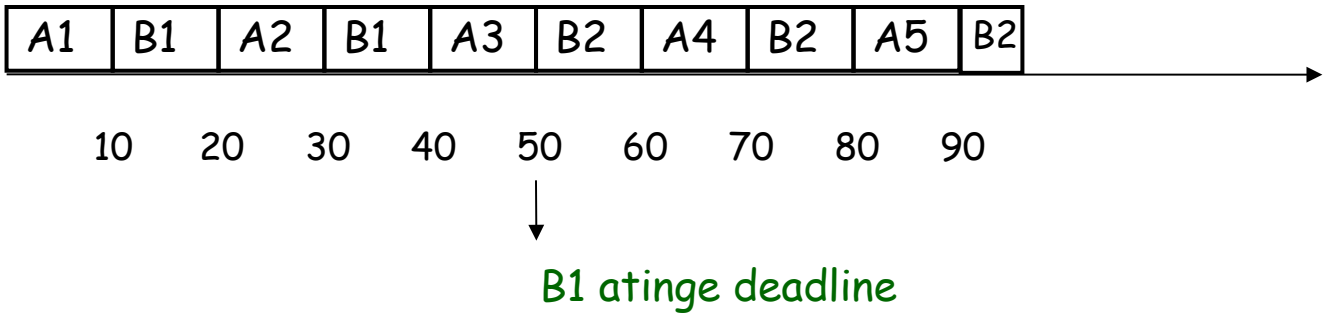
Escalonamento baseado em Deadlines

Processo	Chegada	Tempo de Execução	Deadline de fim
A(1)	0	10	20
A(2)	20	10	40
A(3)	40	10	60
A(4)	60	10	80
A(5)	80	10	100
.....	.....	.....	.....
B(1)	0	25	50
B(2)	50	25	100
.....	.....	.....	.....

Escalonamento baseado em Deadlines

- Decisões são efetuadas a cada 10ms, e escalonamento por prioridade

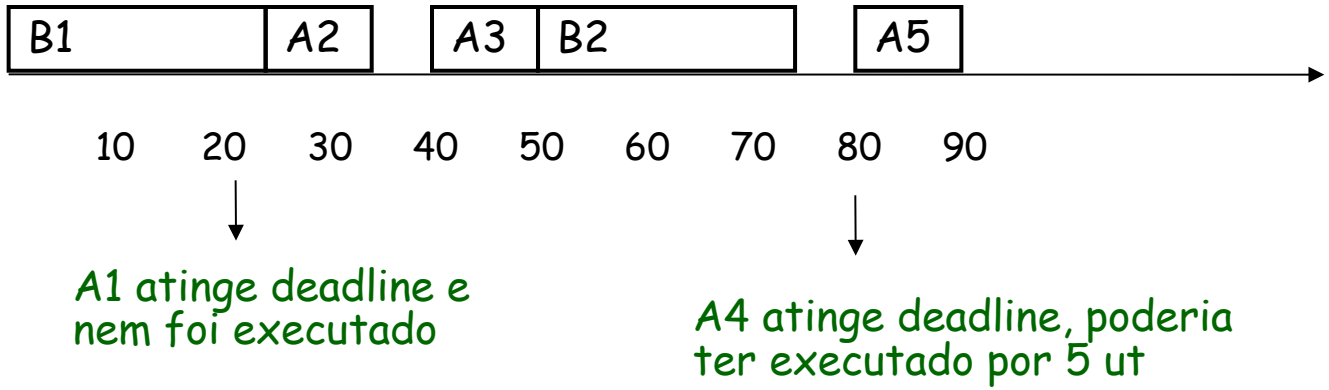
A tem prioridade



Escalonamento baseado em Deadlines

- Decisões são efetuadas a cada 10ms, e escalonamento por prioridade

B tem prioridade



Escalonamento baseado em Deadlines

- Decisões são efetuadas a cada 10ms, e escalonamento por prioridade: menor deadline de fim

Menor deadline de fim