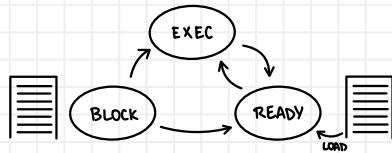


Região crítica: local de compartilhamento de recursos onde é necessário garantir a exclusividade mútua



Motivação para a computação de alto desempenho

- Motivação histórica
 - Execuções mais eficientes no hardware e no software (desempenho, uso de recursos, consumo energético)
 - No hardware: Cache, maior frequência de paralelismo micro-arquitetura, multicore, RAM + rápida...
 - No software: Reduzir complexidade de algoritmos, programação paralela...
- Computação paralela busca alto desempenho
 - Hardware e software específicos
 - Programação paralela herda conceitos de SOs multiprogramados, especificamente da programação concorrente
 - Primeiros conceitos viram da prog. concorrente (esta + geral)
- Programação concorrente visava desenvolver SOs multiprogramados mais confiáveis
 - Primeiros SOs multiprogramados eram uma bagunça:
 - Desenvolvidos em linguagem de montagem
 - Sem fundamentação teórica
 - Tornaram-se enormes com vários erros difíceis de serem rastreados
 - Teste era um tormento devido ao não determinismo de comunicação e sincronização entre processos
 - Geraram uma crise de software: resolvida pela programação concorrente
 - Programação concorrente tornou-se uma das maiores revoluções na programação de computadores
 - ↳ Novas estruturas de comunicação e sincronização organizaram programas
 - Abstrações de processos, memória e E/S permitiram extrapolar a programação concorrente para a programação paralela

Objetivos da Computação Paralela

- Desenvolvimento de soluções computacionais de alto desempenho
 - Indicar explicitamente como as diferentes computações serão executadas paralelamente
 - ↳ Aqui há diferentes níveis de abstração com mais ou menos transparência
 - Usar eficientemente recursos de hardware (processadores, memória e E/S)
- A programação concorrente (e paralela) deve permitir:
 - Ganhos de desempenho com mais transistores
 - Ex: multi e manycores
 - Reduzir o gap entre memória e CPU com acessos concorrentes à memória
 - Ex: sobreposição de requisições à memória

Diferenças entre soluções sequenciais e paralelas

- Detalhes de hardware e software são mais visíveis ao programador
 - Têm forte impacto no desenvolvimento e no desempenho
 - São necessárias diferentes soluções para diferentes Hardwares, SOs e modelos de programação
- Aspectos adicionais a serem considerados:
 - Códigos multidimensionais
 - Identificação do paralelismo na aplicação
 - Maior impacto da arquitetura e software básico
 - Uso de novas ferramentas de software para ao desenvolvimento
 - Uso de novas estruturas de programação:
 - ↳ Iniciar e finalizar processos
 - ↳ Comunicar dados
 - ↳ Sincronizar computações
 - Distribuição da carga de trabalho entre processos e processadores
 - Gerência de dados compartilhados ou locais → Localidade espacial e temporal dos dados são vitais
 - Teste e depuração consideram não determinismo

Código multidimensional: além da visão de dados tem modelos de comunicação e sincronização diferentes.

Conceitos Básicos

- **Computação paralela:** Uso do computador paralelo para aumentar eficiência ou permitir eficácia, usando um programa desenvolvido pela programação paralela
- **Computador paralelo:** Computador com múltiplos processadores, memórias e outros dispositivos replicados
- **Programação concorrente ou paralela ou distribuída:** Atividade de programar em uma linguagem de programação para determinar quando e como tarefas podem ser executadas por diferentes processos ao mesmo tempo (lógico), no mesmo processador ou em processadores diferentes
- **Programa x Processo x Thread**

Processos concorrentes:

Dois ou mais processos que iniciaram e ainda não finalizaram a sua execução

Este definição é bem abrangente.

São processos que concorrem pelos recursos de um sistema computacional

Por definição, podem estar executando em um processador ou em diferentes processadores

Há diferentes variações desta definição na literatura: pseudoparalelismo

Processos paralelos:

São uma especialização de processos concorrentes

Sempre executam em processadores distintos

Paralelismo real

Processos Distribuídos:

Podem ser considerados um sinônimo de processos paralelos

Têm objetivos e modelos de programação distintos

Normalmente visam o compartilhamento de recursos

Encapsulam mais os detalhes de níveis inferiores de hardware e software

Interação (comunicação & sincronização):

Comunicação: troca de dados

Sincronização: garante a ordem de execução dos processos

Granulação (ou granularidade):

Relação entre a computação e a interação

Quanto maior a computação frente à interação, maior a granulação

Granulações mais finas (menores) têm o potencial para oferecer melhores desempenhos em sistemas com recursos mais eficientes

↳ Granulações finas oferecem maior grau de paralelismo

Métricas para avaliar ganho de desempenho:

Speedup (Sp) => determina o ganho de desempenho

Absoluto: considera Tseq o melhor algoritmo sequencial conhecido

Sp = $\frac{Tseq}{Texec}$ (p indica a quantidade de processadores usada)

Relativo: considera Tseq a exec da versão par sobre um processador

Sp = $\frac{Tseq}{Texec}$

Eficiência (Ep) => determina a eficiência no uso de p processadores

E = $\frac{Sp}{p}$

SP linear (Sp = p) ideal, é o que buscamos

SP sublinear (Sp < p) caso comum

SP superlinear (Sp > p) pode acontecer

Escalabilidade: Eficiência permanece constante para a carga de trabalho e o número de processadores aumentam ou diminuem

Modelos de sistemas paralelos:**Modelos de máquina**

Nível de abstração mais baixo

Descrição de hardware e SO: regs, buffers de E/S, assembly

Modelos arquiteturais

Redes de conexão, organização de memória, sincronia de processadores, ...

Classificação de arquiteturas de Flynn

Modelos computacionais

Modelos arquiteturais formais para modelagem de desempenho

RAM, PRAM, PHASE, BSP, LogP, ...

Modelos de programação:

Descrevem semânticas da linguagem de programação

Visão do programador e como ele pode codificar o algoritmo

Sofrem influência dos demais modelos → Diferentes modelos de programação para uma mesma arquitetura

Itens a serem considerados:

Nível do paralelismo: instrução, comandos, procedimentos, loops, processos

Implícito (compiladores) ou explícito (programador)

Especificação de tarefas concorrentes (grau de paralelismo)

Distribuição de tarefas em processos em processadores → Balanceamento da carga

Modo de execução de tarefas → SIMD, SPMD, MPMD, síncrono ou assíncrono

Padrões de comunicação → Passagem de mensagens ou variáveis compartilhadas

Mecanismos de comunicação e sincronização → Organizam a computação e interação entre processos