11 – Programação Paralela

Porque programação paralela?

"Se um único computador (processador) consegue resolver um problema em N segundos, podem N computadores (processadores) resolver o mesmo problema em 1 segundo?"

11 – Programação Paralela

Porque programação paralela?

Dois dos principais motivos para utilizar programação paralela são:

- Reduzir o tempo necessário para solucionar um problema.
- Resolver problemas mais complexos e de maior dimensão.

Outros motivos são:

- Tirar partido de recursos computacionais não disponíveis localmente ou subaproveitados.
- Ultrapassar limitações de memoria quando a memoria disponível num único computador é insuficiente para a resolução do problema.
- Ultrapassar os limites físicos de velocidade e de miniaturização que atualmente começam a restringir a possibilidade de construção de computadores sequenciais cada vez mais rápidos.

11 – Programação Paralela

Porque programação paralela?

Tradicionalmente, a programação paralela foi motivada pela resolução/simulação de problemas fundamentais da ciência/engenharia de grande relevância científica e econômica, denominados como Grand Challenge Problems (GCPs).

11 – Programação Paralela

Porque programação paralela?

Tipicamente, os GCPs simulam fenômenos que não podem ser medidos por experimentação:

- Fenômeno climáticos (e.g. movimento das placas tectônicas)
- Fenômenos físicos (e.g. órbita dos planetas);
- Fenômenos químicos (e.g. reações nucleares);
- Fenômenos biológicos (e.g. genoma humano)
- Fenômenos geológicos (e.g. atividade sísmica)
- Componentes mecânicos (e.g. aerodinâmica/resistência de materiais em naves espaciais)
- Circuitos eletrônicos (e.g. verificação de placas de computador)

11 – Programação Paralela

Porque programação paralela?

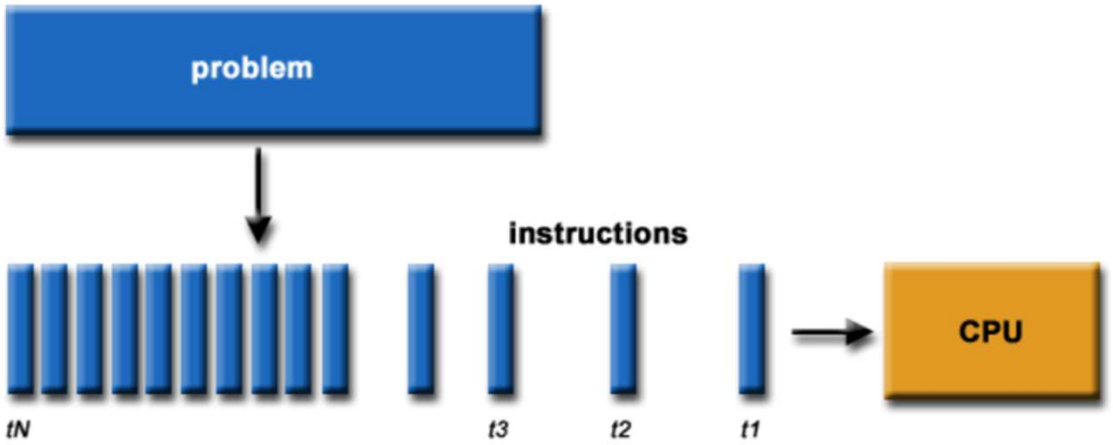
Atualmente, as aplicações que exigem o desenvolvimento de computadores cada vez mais rápidos estão por todo o lado. Estas aplicações ou requerem um grande poder de computação ou requerem o processamento de grandes quantidades de informação. Alguns exemplos são:

- Bases de dados paralelas;
- Mineração de dados (data mining);
- Serviços de procura baseados na web;
- Serviços associados a tecnologias multimídia e telecomunicações;
- Computação gráfica e realidade virtual;
- Diagnostico médico assistido por computador;
- Gestão de grandes industrias/corporações;

11 – Programação Paralela

Programação Sequencial

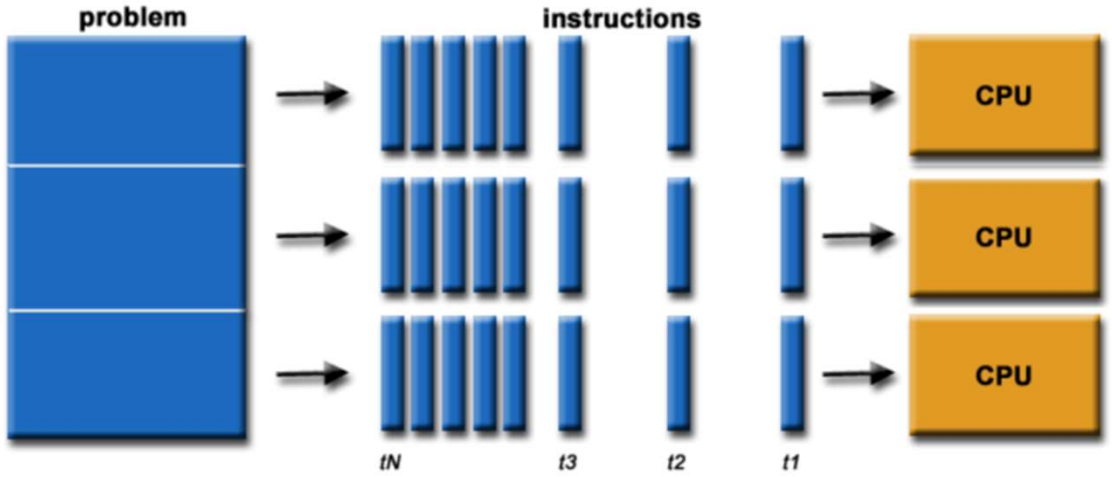
Um programa é considerado programação sequencial quando este é visto como uma série de instruções sequenciais que devem ser executadas num único processador.



11 – Programação Paralela

Programação Paralela

Um programa é considerado programação paralela quando este é visto como um conjunto de partes que podem ser resolvidas concorrentemente. Cada parte é igualmente constituída por uma série de instruções sequenciais, mas que no seu conjunto podem ser executadas simultaneamente em vários processadores.



11 – Programação Paralela

Concorrência ou paralelismo potencial

Concorrência ou paralelismo potencial diz-se quando um programa possui tarefas (partes contiguas do programa) que podem ser executadas em qualquer ordem sem alterar o resultado final.

11 – Programação Paralela

Paralelismo

Paralelismo diz-se quando as tarefas de um programa são executadas em simultâneo em mais do que um processador.

11 – Programação Paralela

Paralelismo Implícito

O paralelismo diz-se implícito quando cabe ao compilador e ao sistema de execução:

- Detectar o paralelismo potencial do programa.
- Atribuir as tarefas para execução em paralelo.
- Controlar e sincronizar toda a execução.

Vantagens e inconvenientes:

- (+) Liberta o programador dos detalhes da execução paralela.
- (+) Solução mais geral e mais flexível.
- (–) Difícil conseguir-se uma solução eficiente para todos os casos.

11 – Programação Paralela

Paralelismo Explícito

O paralelismo diz-se explícito quando cabe ao programador:

- Anotar as tarefas para execução em paralelo.
- Atribuir (possivelmente) as tarefas aos processadores.
- Controlar a execução indicando os pontos de sincronização.
- Conhecer a arquitetura dos computadores de forma a conseguir o máximo desempenho (aumentar localidade, diminuir comunicação, etc.).

Vantagens e inconvenientes:

- (+) Programadores experientes produzem soluções muito eficientes para problemas específicos.
- (–) O programador é o responsável por todos os detalhes da execução(debugging pode ser penoso).
- (–) Pouco portável entre diferentes arquiteturas.

11 – Programação Paralela

Computação Paralela

De uma forma simples, a computação paralela pode ser definida como o uso simultâneo de vários recursos computacionais de forma a reduzir o tempo necessário para resolver um determinado problema. Esses recursos computacionais podem incluir:

- Um único computador com múltiplos processadores.
- Um numero arbitrário de computadores ligados por rede.
- A combinação de ambos.

11 – Programação Paralela

Taxonomia de Flynn

Uma das metodologias mais conhecidas e utilizadas para classificar a arquitetura de um computador ou conjunto de computadores é a taxonomia de Flynn (1966).

- Esta metodologia classifica a arquitetura dos computadores segundo duas dimensões independentes: instruções e dados, em que cada dimensão pode tomar apenas um de dois valores distintos: single ou multiple.

	Single Data	*Multiple Data*
Single Instruction	**SISD** *Single Instruction Single Data*	**SIMD** *Single Instruction Multiple Data*
Multiple Instruction	**MISD** *Multiple Instruction Single Data*	**MIMD** *Multiple Instruction Multiple Data*

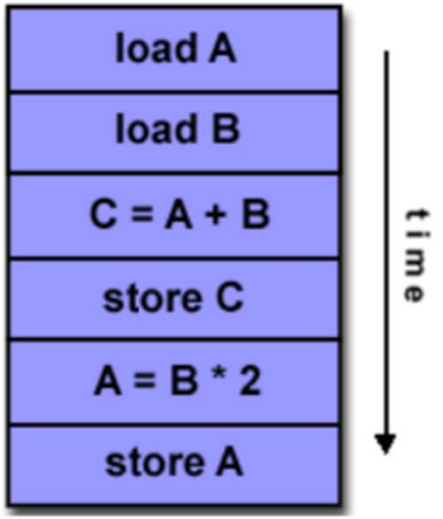
11 – Programação Paralela

SISD – Single Instruction Single Data

Corresponde à arquitetura dos computadores com um único processador.

- Apenas uma instrução é processada a cada momento.
- Apenas um fluxo de dados é processado a cada momento.

Exemplos: PCs, workstations e servidores com um único processador.



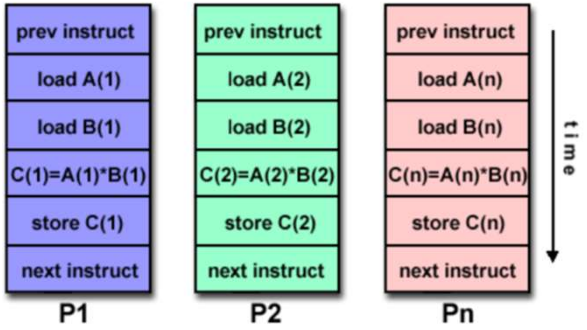
11 – Programação Paralela

SIMD – Single Instruction Multiple Data

Tipo de arquitetura paralela desenhada para problemas específicos caracterizados por um alto padrão de regularidade nos dados (e.g. processamento de imagem).

- Todas as unidades de processamento executam a mesma instrução a cada momento.
- Cada unidade de processamento pode operar sobre um fluxo de dados diferente.

Exemplos: processor arrays e pipelined vector processors.

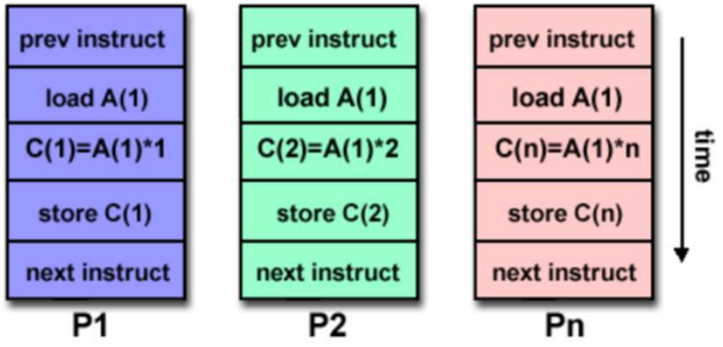


MISD – Multiple Instruction Single Data

Tipo de arquitetura paralela desenhada para problemas específicos caracterizados por um alto padrão de regularidade funcional (e.g. processamento de sinal).

- Constituída por uma pipeline de unidades de processamento independentes que operam sobre um mesmo fluxo de dados enviando os resultados duma unidade para a próxima.
- Cada unidade de processamento executa instruções diferentes a cada momento.

Exemplos: systolic arrays.

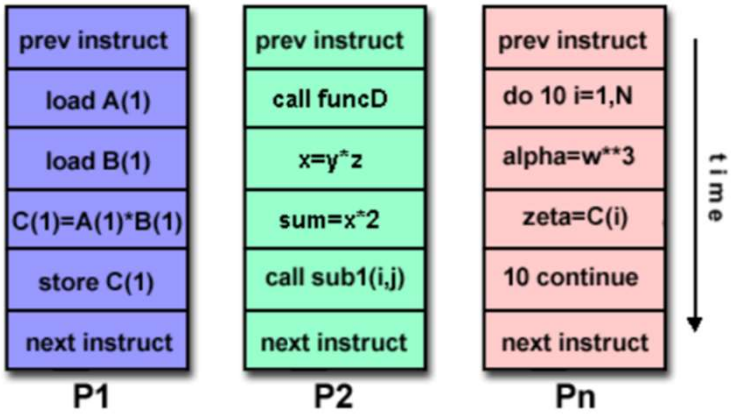


MIMD – Multiple Instruction Multiple Data

Tipo de arquitetura paralela predominante atualmente.

- Cada unidade de processamento executa instruções diferentes a cada momento.
- Cada unidade de processamento pode operar sobre um fluxo de dados diferente.

Exemplos: multiprocessors e multicomputers.

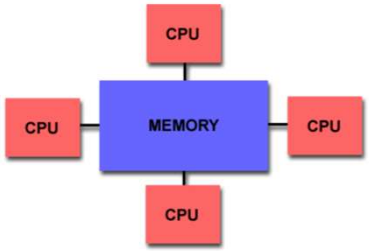


11 – Programação Paralela

Multiprocessors

Um multiprocessor é um computador em que todos os processadores partilham o acesso à memória física.

- Os processadores executam de forma independente mas o espaço de endereçamento global é partilhado.
- Qualquer alteração sobre uma posição de memória realizada por um determinado processador é igualmente visível por todos os restantes processadores.



11 – Programação Paralela

Multiprocessors

Existem duas grandes classes de multiprocessors:

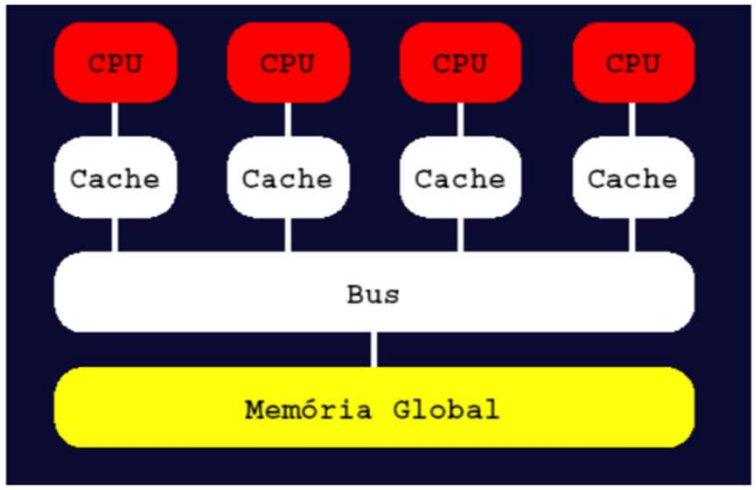
- Uniform Memory Access Multiprocessor (UMA)
- Non-Uniform Memory Access Multiprocessor (NUMA) ou Distributed Multiprocessor

11 – Programação Paralela

Multiprocessors

Uniform Memory Access Multiprocessor (UMA)

- Todos os processadores tem tempos de acesso idênticos a toda a memória.
- A coerência das caches é implementada pelo hardware.

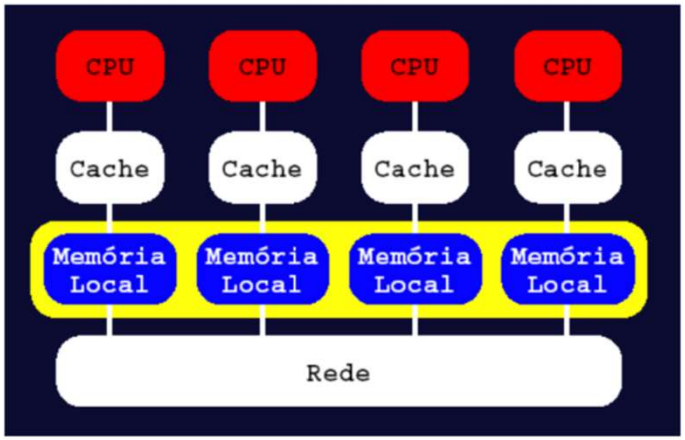


11 – Programação Paralela

Multiprocessors

Non-Uniform Memory Access Multiprocessor (NUMA)

- Os processadores tem tempos de acesso diferentes a diferentes áreas da memória.



Directory-Based Protocol

Associado a cada processador existe um diretório com informação sobre o estado dos seus blocos de memória. Cada bloco pode estar num dos seguintes estados:

- Uncached: não está na cache de nenhum processador.
- Shared: encontra-se na cache de um ou mais processadores e a cópia em memória está correta.
- Exclusive: encontra-se apenas na cache de um processador e a cópia em memória está obsoleta.

11 – Programação Paralela

Multiprocessors

Vantagens e inconvenientes:

- (+) Partilha de dados entre tarefas é conseguida de forma simples, uniforme e rápida.
- (–) Necessita de mecanismos de sincronização para obter um correto manuseamento dos dados.
- (–) Pouco escalável. O aumento do número de processadores aumenta a contenção no acesso à memória e torna inviável qualquer mecanismo de coerência das caches.
- (–) Custo elevado. É difícil e bastante caro desenhar e produzir computadores cada vez com um maior número de processadores.