Plataformas de Programação Paralela

Organização Lógica de Plataformas Paralelas

Aula 4
Alessandro L. Koerich

Programação Paralela
Pontifícia Universidade Católica do Paraná (PUCPR)
Ciência da Computação – 6º Período

Aula Anterior

- Paralelismo implícito
  - *Pipelining*
  - Execução Superescalar
- Limitações na performance da memória dos sistemas
  - Memória *Cache*

Plano de Aula

- Taxonomia de Arquiteturas
- Organização Lógica
- Resumo
- Próxima Aula

Introdução

- Organização Lógica
  Visão que o programador tem da plataforma
- Organização Física
  Organização do hardware da plataforma

Introdução

- Dois componentes críticos do ponto de vista do programador
  - Maneira de especificar tarefas paralelas → estrutura de controle
  - Mecanismos para especificar a interação entre estas tarefas → modelo de comunicação

Estrutura de Controle

- Modelos para especificar a estrutura de controle de programas e o correspondente suporte arquitetural
- Taxonomia de Arquiteturas
  - SISD
  - MISD
  - SIMD
  - MIMD

Taxonomia de Arquiteturas

- Fundamentação para a análise de processamento paralelo
- Entender quais os tipos de alternativas para processamento que foram identificados
- A classificação mais duradoura e ainda razoável, foi proposta em 1966 por *Flynn*

Taxonomia de Arquiteturas

- A classificação de *Flynn* distingue arquiteturas de computadores de acordo com a maneira como elas podem ser classificadas ao longo de 2 dimensões independentes e de valor binário
  - Independentes: nenhuma das 2 dimensões tem algum efeito sobre a outra
  - Valor Binário: cada dimensão tem somente 2 estados possíveis

Taxonomia de Arquiteturas

- Para a arquitetura de computadores, *Flynn* propôs que as 2 dimensões sejam chamadas:
  - Instrução
  - Dados
- Ambas, podem assumir dois valores:
  - Único
  - Múltiplo.
- As 2 dimensões podem ser representadas por uma matriz 2 x 2 e cada uma das 4 células caracteriza um tipo único de arquitetura

Taxonomia de Arquiteturas

Single Instruction Single Data (SISD)

- Estilo mais antigo de arquitetura de computadores
- É ainda uma das mais importantes: todos os computadores pessoais se encaixam dentro desta categoria bem como a maioria dos computadores já projetados.

Single Instruction Single Data (SISD)

- *Single Instruction*: existe somente um fluxo de instruções sendo produzido pela CPU durante qualquer ciclo de *clock*
- *Single Data*: um e somente um fluxo de dados está sendo empregado como entrada durante qualquer ciclo de *clock*.

Single Instruction Single Data (SISD)

- Estes fatores (*Single Instruction, Single Data*) levam a duas características muito importantes de computadores SISD:
  - (1) Instruções seqüências são executadas uma após as outras em passos em passo constante.
  - Este tipo de execução é chamado de serial, em contraste com paralelo, no qual múltiplas instruções podem ser processadas simultaneamente.

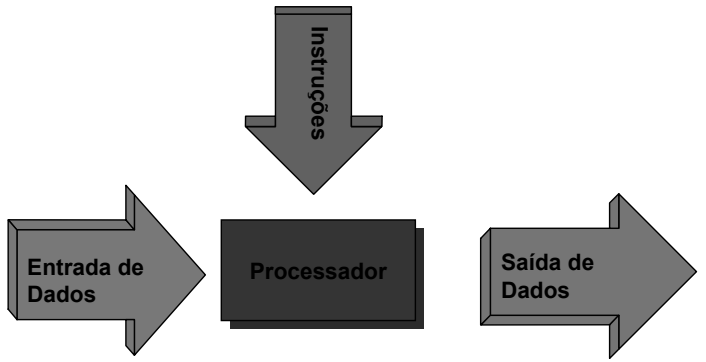
Single Instruction Single Data (SISD)

  - (2) Cada instrução tem um lugar único no fluxo de execução, e portanto um tempo de duração único em que ele "e somente ele" está sendo processado.
  - A execução completa é dita ser deterministica, significando que podemos saber exatamente o que está acontecendo, o tempo todo, e idealmente, podemos recriar o processo exatamente, passo a passo, a qualquer tempo.
- Exemplo: A maioria dos computadores com CPUs de instruções únicas, *workstations*, mini-computadores e mainframes.

Single Instruction Single Data (SISD)

Single Instruction Single Data (SISD)

- A velocidade é limitada pela taxa na qual o computador pode transferir informações internamente. Ex: PC, Macintosh



Multiple Instruction Single Data (MISD)

- Existem poucos tipos de computadores deste tipo.
- Esta categoria foi incluída, mais pelo propósito de manter a apresentação completa do que identificar computadores em funcionamento atualmente.
- Máquinas de propósito específico seriam concebíveis.

Multiple Instruction Single Data (MISD)

- Se encaixariam neste nicho:
  - Filtros de freqüência múltiplos operando sobre um único sinal
  - Algoritmos múltiplos de criptografia tentando quebrar uma única mensagens codificada.
  - Ambos são exemplos onde múltiplos fluxos de instruções independentes são aplicados simultaneamente em um único fluxo de dados.

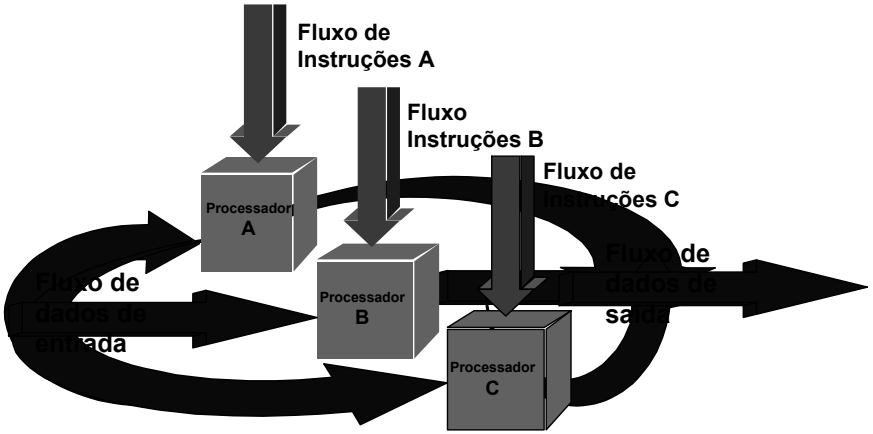
Multiple Instruction Single Data (MISD)

- Um exemplo menos tecnológico de MISD seria o plenário das Nações Unidas:
  - Quando um delegado fala em uma língua de sua escolha, sua fala é traduzida simultaneamente em inúmeras outras línguas para o benefício dos outros delegados presentes.
  - Então, a fala do delegado (dado único) está sendo processado por inúmeros tradutores (processadores) produzindo diferentes resultados.

Multiple Instruction Single Data (MISD)

Multiple Instruction Single Data (MISD)

- É sobretudo mais um modelo teórico do que uma configuração prática. Poucos construídos. Nenhum disponível comercialmente



Single Instruction Multiple Data (SIMD)

- Uma classe muito importante na história da computação.
- Máquinas SIMD são capazes de aplicar simultaneamente, exatamente o mesmo fluxo de instruções à múltiplos fluxos de dados.
- Para certos tipos de problemas, e.g. problemas de dados paralelos, este tipo de arquitetura é perfeitamente adaptado para alcançar taxas de processamento muito rápidas

Single Instruction Multiple Data (SIMD)

- Dados podem ser divididos em muitos pedaços independentes e as múltiplas unidades de instrução podem todas operar sobre eles ao mesmo tempo.
- **Síncrono**
  - Sistemas deste tipo são geralmente considerados síncronos
  - Eles são construídos de maneira a garantir que todas as instruções receberão a mesma instrução ao mesmo tempo, e então todos serão potencialmente capazes de executar a mesma operação simultaneamente.

Single Instruction Multiple Data (SIMD)

- **Deterministico**
  - A qualquer tempo, existe somente uma única instrução sendo executada, embora múltiplas unidades podem estar executando–a.
  - Então, sempre que o mesmo programa é executado nos mesmos dados, usando o mesmo número de unidades de execução, exatamente os mesmos resultados são garantidos em cada passo do processo.

Single Instruction Multiple Data (SIMD)

- Bem adaptada ao paralelismo a nível de instrução / operação.
- A palavra "*single*" em "*single–instruction*" não significa que há somente uma única unidade de instruções como na arquitetura SISD.
- Mas, particularmente que há somente um fluxo de instruções e este fluxo de instruções é executado por unidades múltiplas de processamento em diferentes partes dos dados, todas ao mesmo tempo, executando assim paralelismo.

Single Instruction Multiple Data (SIMD)

- Os mais avançados arranjos de processadores paralelo que estão em produção hoje em dia:
- *Cambridge Parallel Processing Gamma II Plus*: O modelo *Gamma* 1 000 tem 1 024 processadores ordenados em um arranjo 32×32
- O modelo Gamma 4 000 tem 4 096 processadores em um arranjo 64×64.

Single Instruction Multiple Data (SIMD)

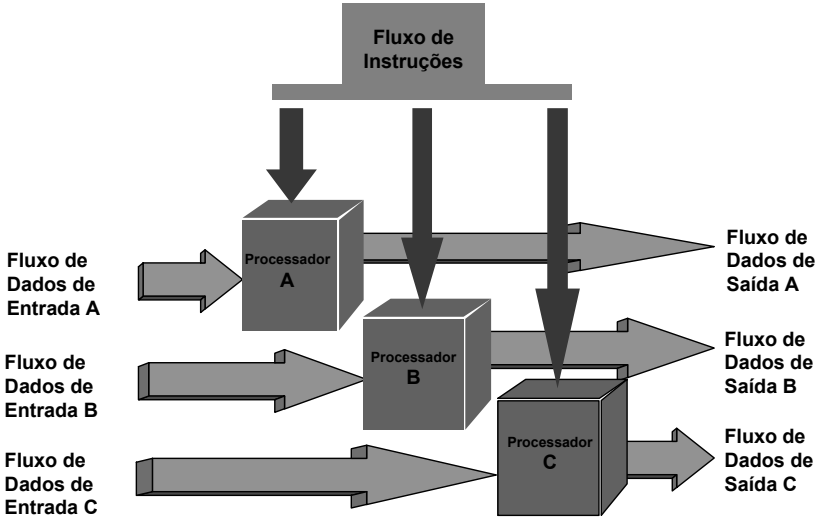
- A exemplo do que acontece em todas as máquinas com arranjo de processadores, o processador de controle tem uma memória separada para guardar as instruções do programa.
- Os dados são guardados em memória de dados associada a cada elemento processador (PE) do arranjo de processadores.

Single Instruction Multiple Data (SIMD)

- *Quadrics Apemille*
  - Os sistemas estão disponíveis em nós de processadores múltiplos de 8, até nós de 128 processadores múltiplos.
  - A topologia de interconexão do *Quadrics* é um grid 3D com interconexões aos lados opostos ('de fato um torus 3-D).

Single Instruction Multiple Data (SIMD)

Single Instruction Multiple Data (SIMD)



Multiple Instruction Multiple Data (MIMD)

- Muitos acreditam que o próximo grande avanço na capacidade computacional estará ligada a esta solução para paralelismo.
- Fornece múltiplos fluxos de instruções simultaneamente aplicados a múltiplos fluxos de dados.
- A mais geral de todas as categorias
- Uma máquina MIMD é capaz de ser programada para operar como se fosse de fato qualquer uma das quatro.

Multiple Instruction Multiple Data (MIMD)

- Fluxos de instruções MIMD síncronos ou assíncronos podem potencialmente ser executados tanto sincronamente quanto asincronamente
- Alguns tipos de algoritmos necessitam um ou outro, e diferentes tipos de sistemas MIMD são melhor adaptados a um ou outro;
- Eficiência ótima depende em fazer com que o sistema que executa o código reflita o estilo de sincronização requerida pelo código.

Multiple Instruction Multiple Data (MIMD)

- Sistemas MIMD deterministicos ou não deterministicos são potencialmente capazes de apresentar um comportamento deterministico, isto é, de reproduzir o exato conjunto de passos de processamento toda a vez que um programa é executado com os mesmos dados.
- Bem adaptado para o paralelismo a nível de blocos, *loops* e sub-rotinas.
- Quanto mais código seja dado a cada processador em uma montagem MIMD, mais eficientemente o sistema total irá operar.

Multiple Instruction Multiple Data (MIMD)

- Isto é devido em grande parte as necessidades de comunicação, particularmente sincronização, que são menos limitadas em níveis acima do paralelismo orientado a instruções.
- Sistemas com instruções múltiplas ou programas únicos estilo MIMD são capazes de funcionar em modo "*multiple-instruction*" verdadeiro, com cada processador fazendo alguma coisa diferente, ou a cada processador pode ser dado o mesmo código.

Multiple Instruction Multiple Data (MIMD)

- Este último caso é chamado SPMD (*Single Program Multiple Data*) e é uma generalização do paralelismo estilo SIMD, com muito menos necessidade de sincronização.
- Exemplos: A seguir temos alguns exemplos das diferentes maneiras que o paralelismo MIMD pode ser realizado:

Multiple Instruction Multiple Data (MIMD)

- Asíncronos: implementam paralelismo MIMD em termos de fluxos de instrução mais ou menos emparelhados:
  - IBM SPx, ou clusters de *workstations*, usando PVM, MPI etc. A natureza das bibliotecas de passagem de mensagens faze com que o sistema paralelo resultante seja muito mais adaptado a tarefas coarser-grained, loosely-coupled.
  - Unidades vetoriais múltiplas trabalhado em cima de um mesmo problema (e.g., Fujitsu VPP5000) Sistemas vetoriais paralelos são muito eficientes em tarefa de dados paralelos, onde a cada unidade do vetor é atribuída responsabilidade pelas computações envolvendo um único segmento dos dados totais.

Multiple Instruction Multiple Data (MIMD)

- Hypercubes (e.g., nCube2S) O hypercube é um da família completa de arquiteturas de rede que fornece múltiplos pontos de conexão entre processadores, tipicamente permitindo que cada processador seja conectado diretamente a 8 ou mais processadores.
- Meshes (e.g., Intel Paragon). Meshes fazem quase a mesma coisa, e vem em configurações 2-D ou 3-D, o primeiro parecendo como um simples grid plano e o segundo como uma caixa

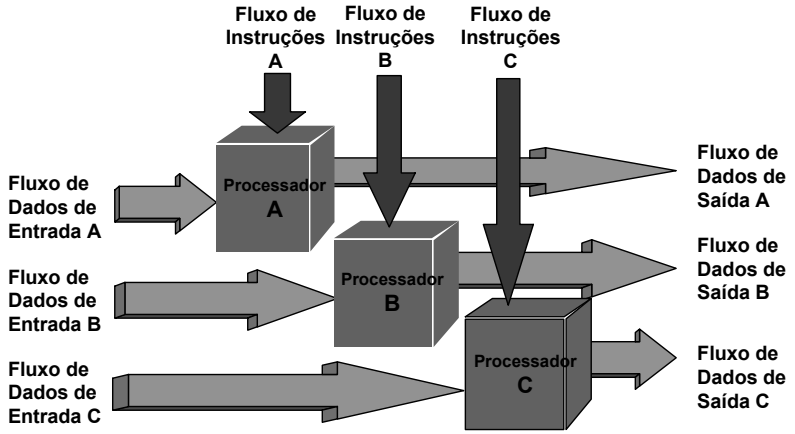
Multiple Instruction Multiple Data (MIMD)

- Síncronos: IBM RS/6000 (até 4 instruções por ciclo), NEC SX-5. Os processadores são agora capazes de executar múltiplas instruções a cada ciclo, contudo nem todos os ciclos estão preenchidos.
- Exemplo: a multiplicação de dois valores de ponto flutuante já carregados nos registradores, a adição inteira correspondente a localização do arranjo na memória do próximo valor de ponto flutuante a ser adicionado, e a captura daquele valor.
- Este tipo de processamento paralelo necessita um compilador sofisticado com a habilidade de ordenar instruções de forma a preservar tanto as instruções precedentes necessárias como reconhecer instruções independentes.

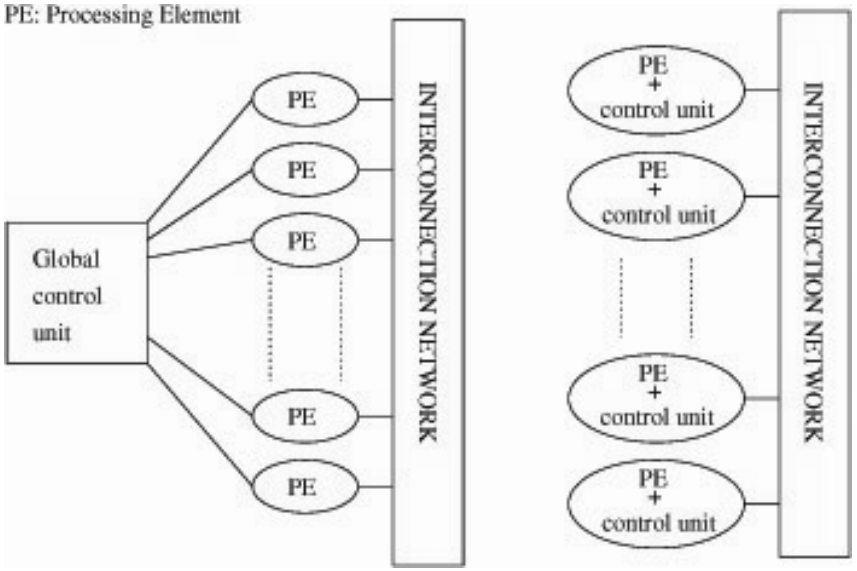
Multiple Instruction Multiple Data (MIMD)

Multiple Instruction Multiple Data (MIMD)

- Diferente das arquiteturas SISD e MISD, computadores MIMD não trabalham sincronamente.



SIMD x MIMD

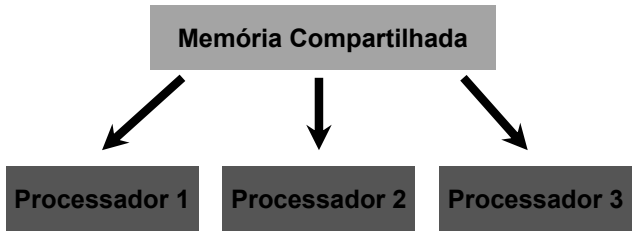


Modelos de Comunicação para Plataformas Paralelas

- Existem duas formas primárias de troca de dados entre tarefas paralelas:
  - Acessando um espaço de dados compartilhado
  - Trocando mensagens

Plataformas de Espaço de Endereçamento Compartilhado

- A visão de espaço de endereçamento compartilhado para uma plataforma paralela suporta um espaço de dados comum que é acessível a todos os processadores.



Plataformas de Espaço de Endereçamento Compartilhado

- A memória em plataformas de espaço de endereçamento compartilhado pode ser:
  - Local: exclusiva para um processador
  - Global: comum a todos os processadores

Plataformas de Espaço de Endereçamento Compartilhado

- UMA: se o tempo necessário para um processador acessar qualquer palavra na memória do sistema (local ou global) é idêntico, a plataforma é classificada como um multicomputador:

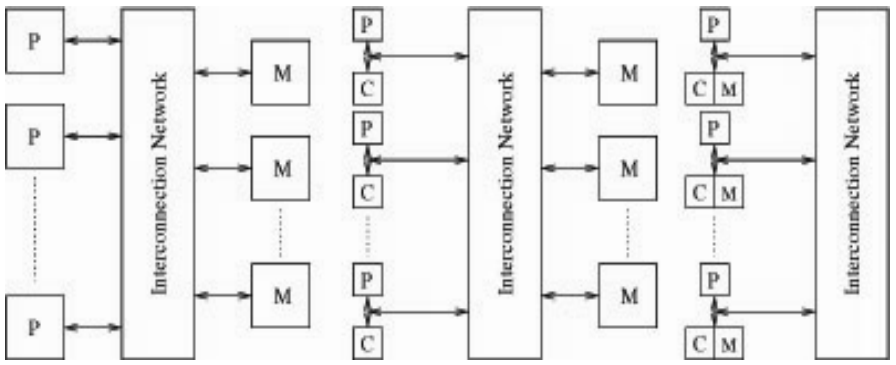
UMA (*uniform memory access*)

Plataformas de Espaço de Endereçamento Compartilhado

- NUMA: se o tempo necessário para um processador acessar certas palavra na memória do sistema é maior que para outras, a plataforma é chamada de uma multicomputador

NUMA (*non–uniform memory access*)

Plataformas de Espaço de Endereçamento Compartilhado



Plataformas de Passagem de Mensagens

- A visão lógica de uma plataforma de passagem de mensagens consiste em p nós de processamento cada um com seu espaço de endereçamento exclusivo
- Cada um destes nós de processamento pode ser tanto:
  - Processadores únicos
  - Multiprocessador com espaço de endereçamento compartilhado

Plataformas de Passagem de Mensagens

- Interações entre processos rodando em diferentes nós deve ser realizada através de mensagens ⇨ passagem de mensagens
- Esta troca de mensagens é utilizada para transferir:
  - Dados
  - Trabalho
  - Sincronizar ações entre processadores

Plataformas de Passagem de Mensagens

- Interações são realizadas pelo envio e recebimento de mensagens, assim, as operações básicas neste paradigma de programação são
- *Send* e *Receive*
- Além disso necessitamos especificar o alvo
- ID
- Mecanismo para designar identificadores únicos para os múltiplos processos sendo executados em paralelo

Plataformas de Passagem de Mensagens

- Outra função necessária:
- *Numprocs*
- Esta função especifica o número de processos participando do grupo.

Plataformas de Passagem de Mensagens

Com estas 4 operações básicas
- *Send, Receive, ID, Numprocs*
- É possível escrever qualquer programa de passagem de mensagens
- Diferentes APIs (interface para programação de aplicativos) de passagem de mensagens, tais como:
  - *Message Passing Interface* (MPI)
  - *Parallel Virtual Machine* (PVM)
- suportam estas operações, além de outras funções de mais alto nível.

Resumo

- Organização Lógica
  Visão que o programador tem da plataforma
- Dois componentes críticos do ponto de vista do programador
  - Maneira de especificar tarefas paralelas → estrutura de controle
  - Mecanismos para especificar a interação entre estas tarefas → modelo de comunicação

Resumo

- A estrutura de controle está relacionado com a arquitetura
- Quatro tipos de arquiteturas
  - SISD
  - MISD
  - SIMD
  - MIMD

Resumo

- O modelo de comunicação está relacionado com a maneira em que os dados são trocados entre tarefas paralelas.
- Duas formas primárias
  - Plataformas de espaço de endereçamento compartilhado
  - Plataformas de passagem de mensagens