Sumário

- Introdução:
- Etapas para o desenvolvimento e análise de Programas Paralelos;
- Computação Paralela sobre Sistemas Distribuídos;
- Ambientes para Troca de Mensagens;
  - PVM;
  - MPI;
- Resultados;

Sumário

- Referências para consulta:Onde encontrar mais informaçõe...

- ALMASI, G., GOTTLIEB, A., *Highly Parallel Computing*, Second Edition, The Benjamin/Cummings Publishing Company, 1994.
- QUINN, M. J., *Parallel Computing: Theory and Practice*, Second Edition, McGRAW-HILL, 1994.
- Sunderam, V. S., Geist, G. A., Dongarra, J. and Manchek, R., *The PVM Concurrent Computing System: Evolution, Experiences, and Trends*, Parallel Computing, Vol. 20, No. 4, pp.531-545, April 1994.
- McBryan, O. A., *An Overview of Message Passing Environments*, Parallel Computing, vol. 20, pp. 417-444, 1994.

Sumário

- Referências para consulta:Onde encontrar mais informações...
- Na Internet:
  - http://www.epm.ornl.gov/pvm/pvm_home.html
  - http://www.mpi-forum.org
  - http://www.lam-mpi.org
  - http://lasdpc.icmc.sc.usp.br

Introdução

- Busca por melhor desempenho e menor custo;
- Máquinas "von Neumann"
  - Programação Seqüencial → Uma tarefa por vez;
  - Gargalo de von Neumann → Baixo desempenho;
  - Serialização de problemas paralelos.
- Solução: Arquiteturas Paralelas
  - Programação paralela;
  - Vários processadores trabalhando em uma mesma tarefa;
  - Diversas instruções são executadas em paralelo.

Introdução

- Definição de Programação Paralela

Vários Processos Executados em diferentes Processadores e Trabalhando em Conjunto em um Único Problema

Coleção de Elementos de Processamento que se comunicam e cooperam entre si e resolvem um Problema mais Rapidamente (Almasi)

Processamento de Informação que enfatiza a manipulação Concorrente de Dados que Pertencem a um ou mais Processos que Resolvem um único Problema

Introdução

- Histórico:
  - 1953: bits → palavras;
  - 1958: Processadores de E/S;
  - Anos 70: Pipelines e Processadores de Ponto Flutuante
    - 1975 - Illiac IV:
      - Paralelismo de alto nível;
      - 64 processadores;
      - 16 *racks*;
      - **Área de uma quadra de basquete!**

Introdução

- Histórico:
  - Anos 80:
    - Computadores Vetoriais $\rightarrow$ *Cray;*
    - Máquinas SIMD (*Single Instruction Multiple Data*);
    - Máquinas MIMD (*Multiple Instruction Multiple Data*);
    - *Dataflow*.
  - Problema: Custo
  - Solução: VLSI (anos 80)
    - Popularização dos computadores pessoais;
    - Máquinas com vários processadores;
    - Arquiteturas paralelas a um custo acessível.

Introdução

- Histórico:
  - Final dos anos 80:
    - Possibilidade de utilizar o hardware dos Sistemas Distribuídos como uma arquitetura paralela de memória distribuída (MIMD);
    - Convergência entre as áreas de Computação Paralela e Sistemas Distribuídos;

Introdução

- É importante diferenciar os termos:
  - Programação Seqüencial;
  - Programação Concorrente;
  - Programação Paralela;

Introdução

- Programação Seqüencial:
  - Várias tarefas sendo executadas uma após a outra.
- Programação Concorrente:
  - Várias tarefas sendo executadas concorrentemente;
  - Em arquiteturas paralelas (vários processadores) têm-se: Programação Paralela;
  - Único processador: Pseudo-Paralelismo;
- Programação Paralela:
  - Vários processos executados em diferentes processadores trabalhando em conjunto em um único problema;

Introdução

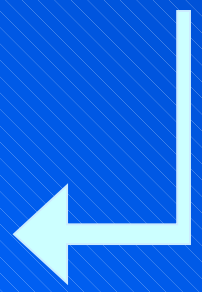
- Exemplos:
  - Como Preparar uma Refeição;
  - Multiplicação paralela de matrizes;

Introdução

- Programação Seqüencial - Multiplicação de matrizes:

```
Inicializar Matrizes A[i][k] e B[k][j];
ler toda a matriz A
ler toda a matriz B
Para t de 1 até i faça
    Para y de 1 até j faça
        Para x de 1 até k faça
            Matriz C[t][y] += A[t][x]*B[x][y]
```

As matrizes devem ser inteiramente lidas e depois é realizada a multiplicação



Introdução

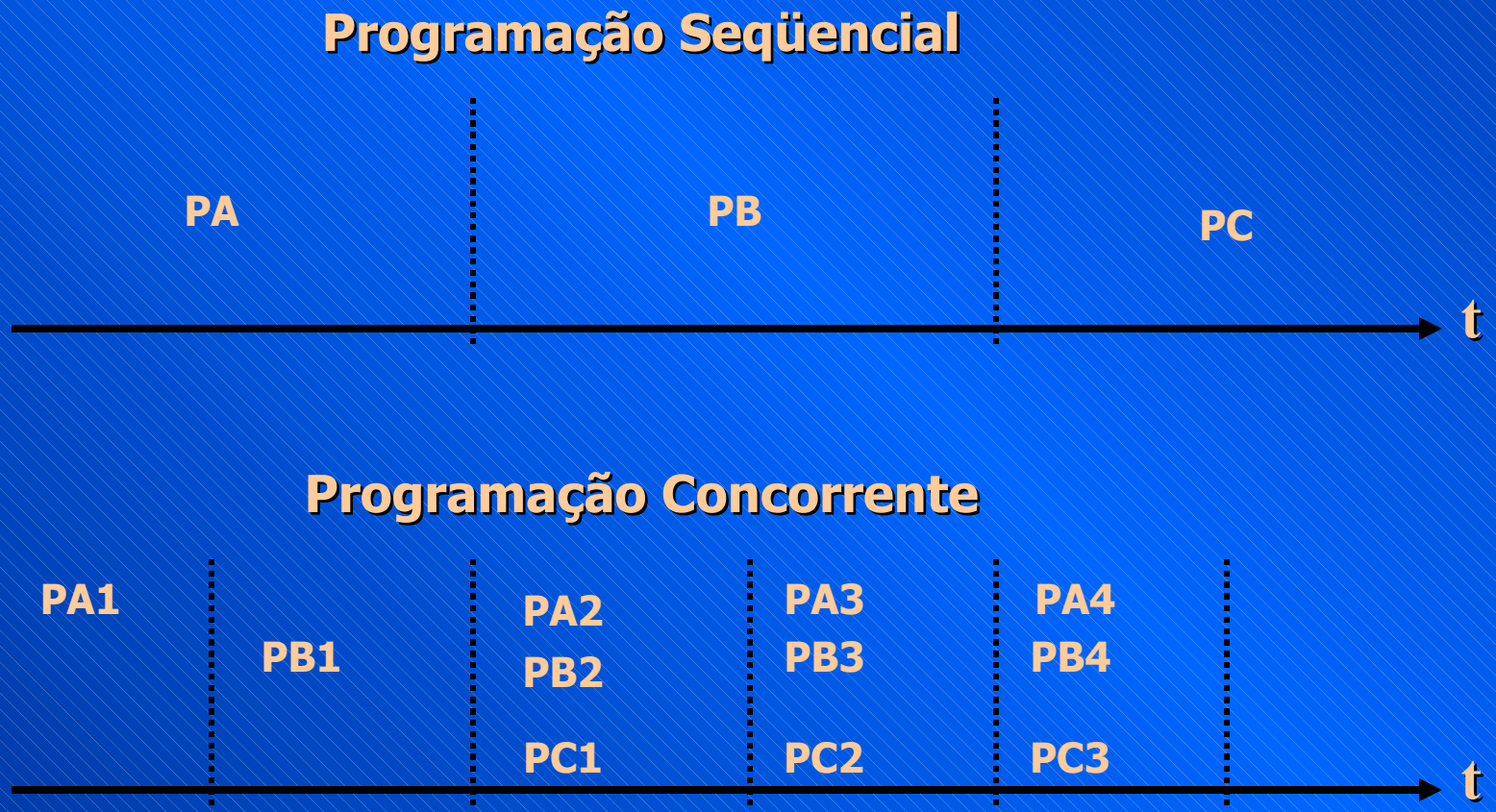
- Programação Concorrente - Multiplicação de matrizes:

```
Inicializar Matrizes A[i][k] e B[k][j];
Para t de 1 até i faça - Proc PAt
    ler linha t da matriz A
Para y de 1 até j faça - Proc PBy
    ler coluna y da matriz B
Para t de 1 até i faça - Proc Pcy
    Para y de 1 até j faça
      Para x de 1 até k faça
            Matriz C[t][y] += A[t][x]*B[x][y]
```

Leitura e multiplicação realizadas de maneira concorrente

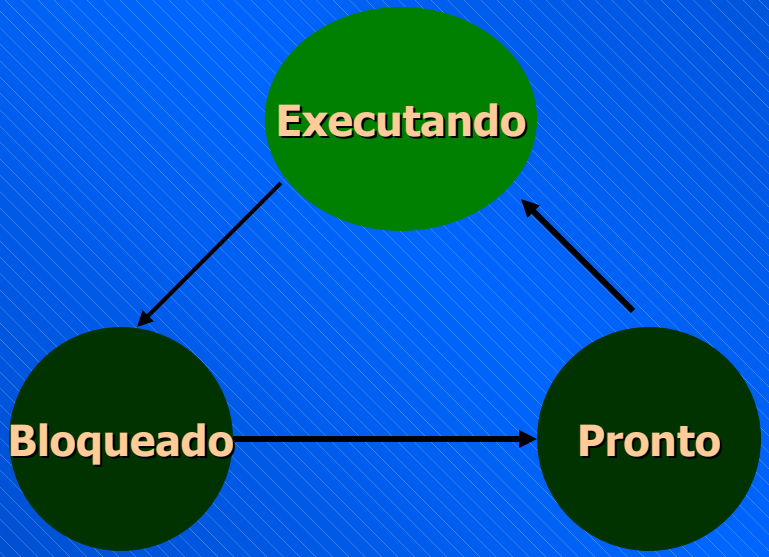
Introdução

- Programação Concorrente:
  - Com o exposto, pode ser verificado que:



Introdução

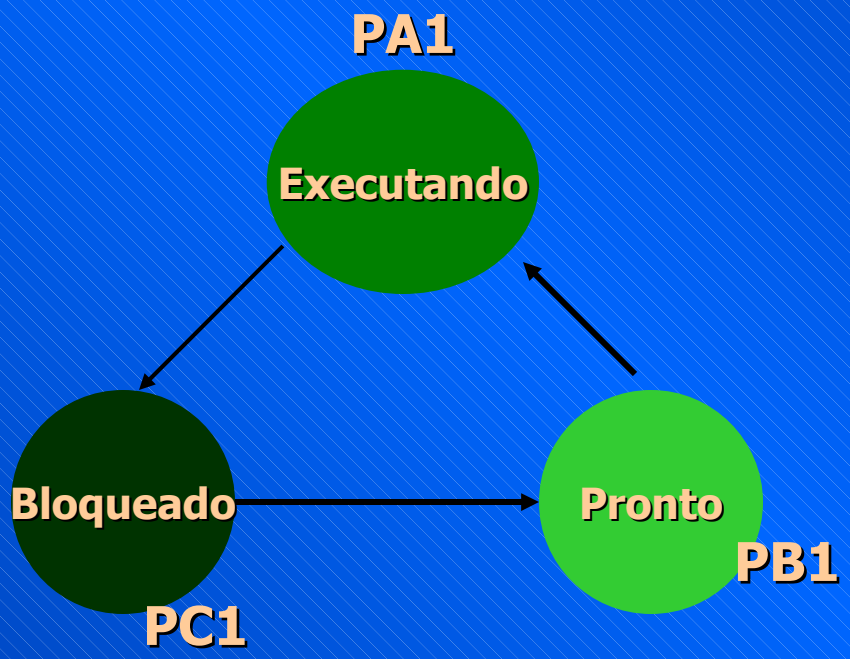
- Programação Concorrente:
  - Considerando 3 estados para um processo em um processador;



  - E que:
    - Processo PAt: Lê linha t da matriz A
    - Processo PBy: Lê coluna y da matriz B
    - Processo PCy: Multiplica a linha t da matriz A pela coluna y da matriz B

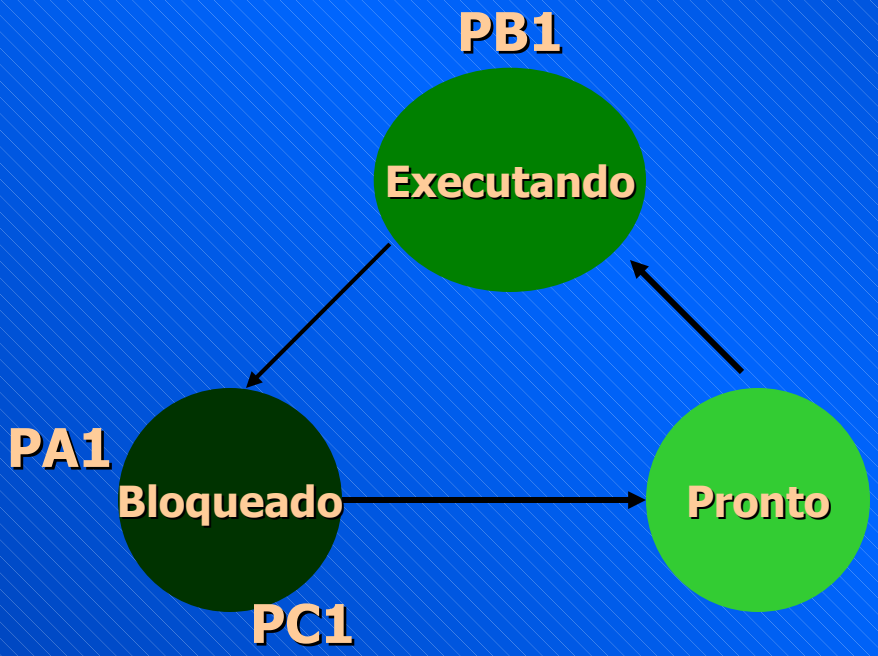
Introdução

- Programação Concorrente:
  - Tem-se que:



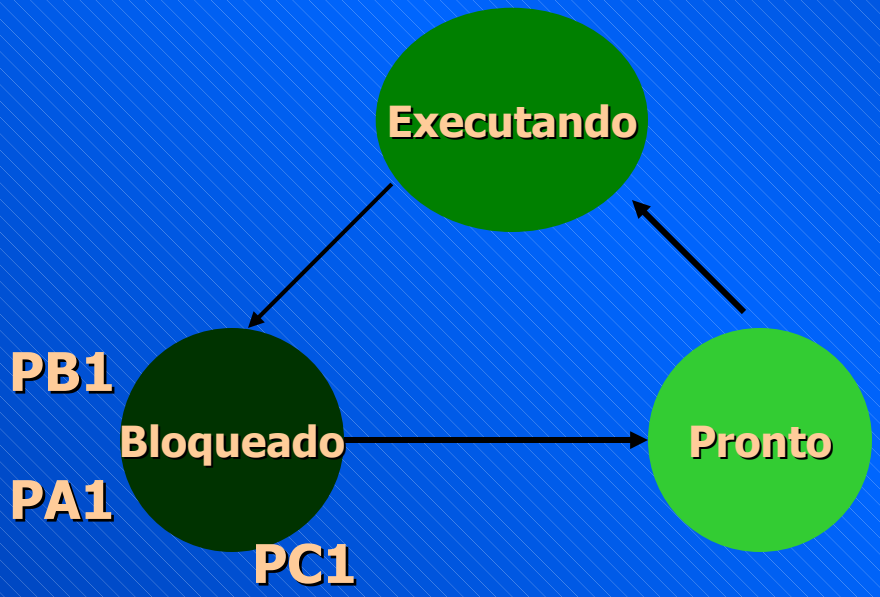
Introdução

- Programação Concorrente:
  - Tem-se que:



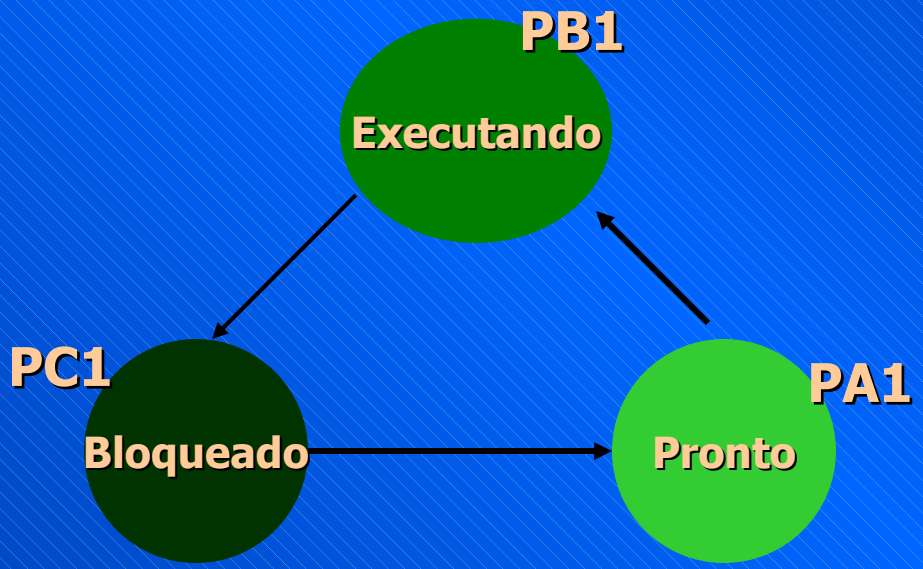
Introdução

- Programação Concorrente:
  - Tem-se que:



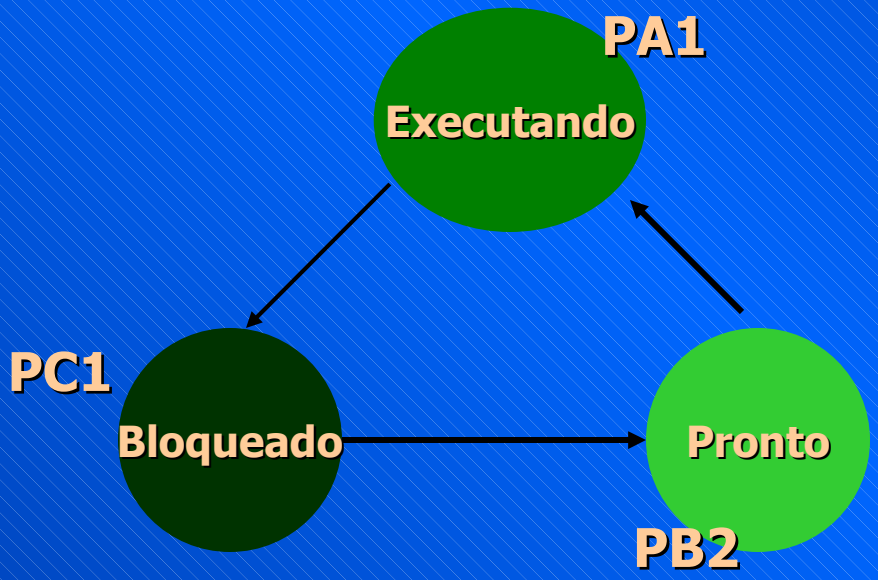
Introdução

- Programação Concorrente:
  - Tem-se que:



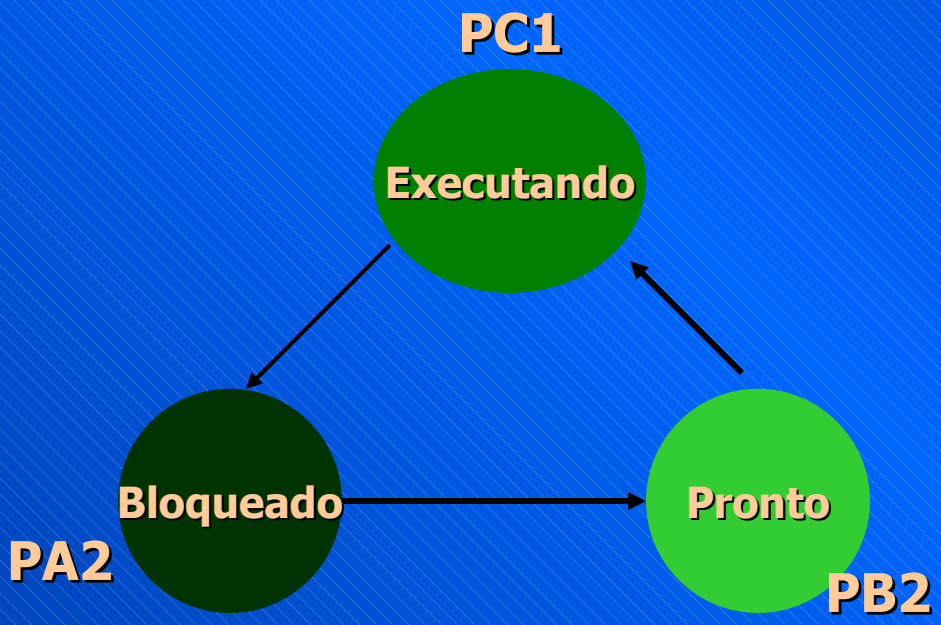
Introdução

- Programação Concorrente:
  - Tem-se que:



Introdução

- Programação Concorrente:
  - Nota-se que nesse estágio existem tarefas sendo executadas **concorrentemente!**



Introdução

- Programação Paralela - Multiplicação de matrizes:

Processador 1

Inicializar Matrizes A[i][k] e B[k][j];
ler matriz A;
enviar matriz A
receber matriz B
para i de 1 até (k/3)
    calcular parte da matriz C

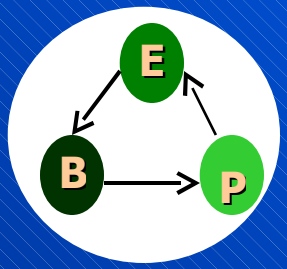
Processador 2

ler matriz B
receber matriz A
enviar matriz B
para i de k/3+1 até 2*(k/3)
    calcular parte da matriz C

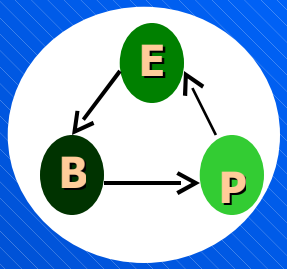
Processador 3

receber matriz A
receber matriz B
para i de 2*(k/3)+1 até k
    calcular parte da matriz C

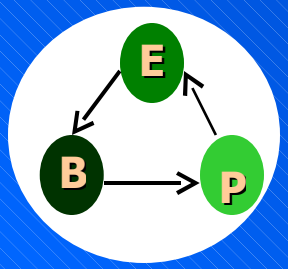
Processador 1



Processador 2



Processador 3



Introdução

- Qual o número ideal de processadores para realizar uma tarefa?
- Qual deve ser a potência e a função de cada um?
- Questões que devem ser verificadas:
  - Sincronismo entre os processos;
  - Granulação das tarefas;
  - Balanceamento de carga;
  - Organização da memória:
    - Centralizada;
    - Distribuída
  - Desempenho.

Introdução

- Comunicação e Sincronismo:
  - Como um processador informa aos outros sobre uma multiplicação que deve ser realizada?
  - Como receber os resultados de todos os processadores que ajudaram no processamento?
  - Como é efetuada a comunicação e o sincronismo entre processos/processadores?

Introdução

- Comunicação e Sincronismo:
  - Utilização de Redes de Interconexão:
    - Relação Custo x Desempenho;
    - Conectividade;
    - Velocidade;
    - Custo;
    - Caminhos Alternativos;
      - Exemplos: Redes Crossbar, Ethernet, Linear, Estrela, etc.
  - Com relação ao sincronismo...
    - Algoritmo Seqüencial: Ordenação Total;
    - Algoritmo Paralelo: Ordenação Parcial $\rightarrow$ Não Determinismo.

Introdução

- Granulação das tarefas: Tamanho das unidades de trabalho submetidas aos processadores;
  - Fatores que influenciam:
    - Número de Processos/Processadores considerados;
    - Tamanho de cada tarefa.
  - Classificada em Fina, Média e Grossa;

Introdução

- Granulação das tarefas:
  - Granulação fina: Paralelismo de baixo nível
    - Grande número de processos pequenos e simples;
    - Unidade de paralelismo: instruções/operações;
  - Granulação Média: Paralelismo de nível médio
    - Vários processos;
    - Unidade de paralelismo: Procedimentos/funções/*loops*.
  - Granulação Grossa: Paralelismo de alto nível
    - Poucos processos grandes e complexos;
    - Unidade de paralelismo: Processos/Programas;

Introdução

- Balanceamento de carga:
  - Como dividir as tarefas entre os processadores?
  - É desejável atribuir uma tarefa “adequada” para cada processador;
  - Tentativa de minimizar comunicação e sincronismo.

Introdução

- Organização da memória:
  - Como a memória é organizada?
  - Qual a quantidade de memória necessária?
  - Fatores que podem influenciar na organização da memória:
    - Potência dos Processadores;
    - Relação Processamento/Comunicação;
    - Freqüência e tipo de E/S.

Introdução

- Organização da memória:
  - Pode ser:
    - Compartilhada: Espaço de endereçamento único entre todos os processadores;
    - Distribuída: Espaço de endereçamento individual (para cada processador).

Introdução

- Desempenho:
  - Há vantagem na utilização de mais que um processador?
  - Quantos processadores seriam necessários?
  - Como medir a vantagem na utilização da computação paralela?

  - Pode-se fazer uso de duas medidas:
    - *Speedup*: Qual o ganho na utilização da computação paralela?
    - Eficiência: Quanto da potência computacional envolvida foi utilizada?

Introdução

- Desempenho:
  - *Speedup* (Sp): Relação entre o tempo para executar um algoritmo em um único processador (T_1) e o tempo para executá-lo em **p** processadores (T_p);

$$Sp = T_1 / T_p$$

  - Exemplo:
    - Tempo para execução da multiplicação de matrizes em um processador: T1=10 ut
    - Tempo para execução da multiplicação de matrizes em 3 processadores: Tp=4 ut (p=3)
    - *Speedup* alcançado → Sp = 10/4 → **Sp = 2,5**

Introdução

- Desempenho:
  - Eficiência (`Ef`): Relaciona *Speedup* e o número de processadores;

$$Ef = Sp/p$$

  - Tomando como base o exemplo anterior (`Sp=2,5` e `p=3`);
    - Eficiência → `Ef=2,5/3` → **`Ef = 0,83`**
  - No caso ideal:
    - *Speedup*=p e Eficiência = 1
  - No caso real:
    - *Speedup* < p e Eficiência < 1

Introdução

- Vantagens
  - **Alto Desempenho** - fim do Gargalo de von Neumann;
  - Solução mais Natural para **Problemas Intrinsecamente Paralelos**;
  - Maior Facilidade de implementação de **Tolerância a Falhas;**
  - Desenvolvimento de **Programas Modulares.**
- Desvantagens
  - **Programação mais Complexa;**
  - **Sobrecargas Introduzidas na Comunicação e Sincronismo.**

Introdução

Tarefa para casa

Ler o Capítulo 1 do Livro:

ALMASI, G., GOTTLIEB, A., *Highly Parallel Computing*, Second Edition, The Benjamin/Cummings Publishing Company, 1994.