Computação Paralela

Regina Helena Carlucci Santana
rcs@icmc.sc.usp.br

Laboratório de Sistemas Distribuídos e Programação Concorrente
ICMC/USP - São Carlos

Sumário

- Introdução;
- Etapas para o desenvolvimento e análise de Programas Paralelos;
- Computação Paralela sobre Sistemas Distribuídos;
- Resultados;
- Considerações Finais;
- Referências para consulta.

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

- Histórico:
  - Anos 70:
    - 1975 - Illiac IV:
      - Paralelismo de alto nível;
      - 64 processadores;
      - 16 *racks*;
      - **Área de uma quadra de basquete!**
  - Anos 80:
    - Computadores Vetoriais → *Cray;*
    - Máquinas SIMD (*Single Instruction Multiple Data*);
    - Máquinas MIMD (*Multiple Instruction Multiple Data*);
    - *Dataflow*.
  - Problema: Custo

Introdução

- Histórico:
  - Solução: VLSI (anos 80)
    - Popularização dos computadores pessoais;
    - Máquinas com vários processadores;
    - Arquiteturas paralelas a um custo acessível.
  - 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;

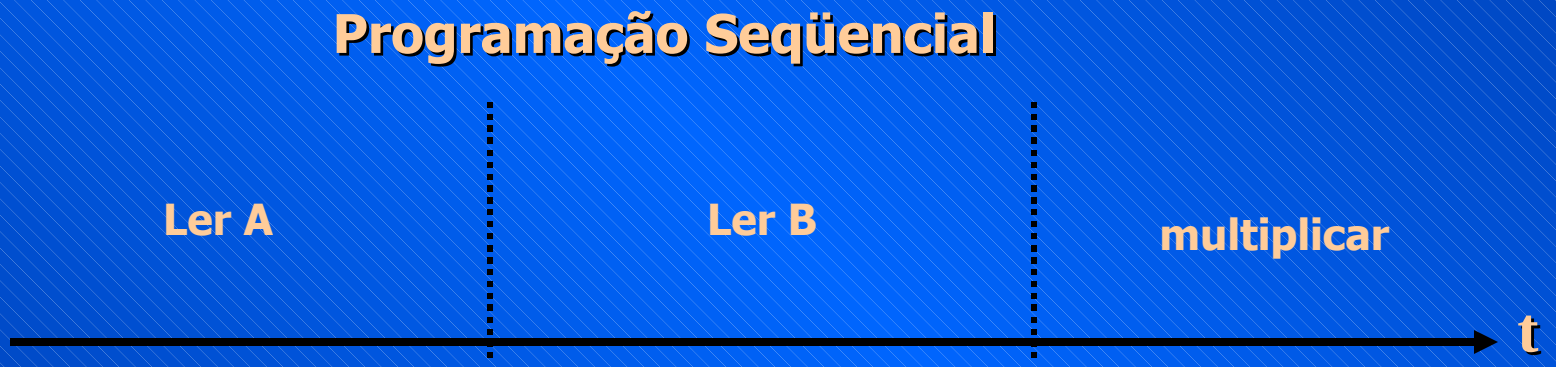
- Programação Seqüencial:
  - Várias tarefas sendo executadas uma após a outra.

Introdução

- Programação Concorrente:
  - Várias tarefas sendo executadas concorrentemente;
  - Em arquiteturas paralelas (vários processadores) têm-se: Programação Paralela;
  - Único processador: Pseudo-Paralelismo;
  - Exemplo: Multiplicação de matrizes

Introdução

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



Introdução

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

- 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 ou Distribuída
  - Desempenho.

Introdução

- Comunicação e Sincronismo:
  - Como um processador informa aos outros sobre uma operação 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

- 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

- 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?
    - Compartilhada: Espaço de endereçamento único entre todos os processadores;
    - Distribuída: Espaço de endereçamento individual (para cada processador).
  - 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

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

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.**

Sumário

- Introdução;
- **Etapas para o desenvolvimento e análise de Programas Paralelos;**
- Computação Paralela sobre Sistemas Distribuídos;
- Resultados;
- Considerações Finais;
- Referências para consulta.

Etapas para o Desenvolvimento e Análise de um Programa Paralelo

- I - Desenvolvimento de um Algoritmo Paralelo
- II - Desenvolvimento do Programa Paralelo
- III - Mapeamento de Processos:
- IV - Teste e Depuração
- V - Avaliação de Desempenho

Desenvolvimento do Algoritmo Paralelo

Abordagem do Algoritmo
Identificação do Algoritmo e Divisão dos Processos
Organização do Trabalho

Desenvolvimento do Algoritmo Paralelo

- Programa Seqüencial:
  - Existência de programa seqüencial com documentação e especificação pobres;
  - Utilização de ferramentas;
  - Baixa flexibilidade - *Speedup* limitado.

- Algoritmo seqüencial
  - Adaptar algoritmo seqüencial e refazer programa;
  - Nem sempre melhor algoritmo seqüencial é o melhor algoritmo paralelo;
  - Maior flexibilidade, mas ainda limitada;
  - Melhores *speedups*, mas ainda pode não ser ideal.

Desenvolvimento do Algoritmo Paralelo

- Problema a ser resolvido:
  - Análise do problema e proposta de algoritmo;
  - Programador deve conhecer bem o problema;
  - Alta flexibilidade, pode-se obter bom *speedup*;
  - Base em algoritmos paralelos que resolvem outros problemas.

Identificação do paralelismo e divisão dos processos

- Aspectos importantes que devem ser considerados:
  - Arquitetura das máquinas disponíveis;
  - Tipo de comunicação;
  - Granulação;
  - Sincronismo.

Organização do Trabalho

- Especifica o modelo de concorrência a ser utilizado;
- Depende da arquitetura considerada:
  - SIMD (*Single Instruction Multiple Data*)
  - MIMD (*Multiple Instruction Multiple Data*)
    - **Memória compartilhada**
    - **Memória distribuída**
- Duas Abordagens:
  - Paralelismo por Dado - Executa as mesmas instruções simultaneamente em um conjunto de dados distintos.
  - Paralelismo por Controle - Executa instruções diferentes sobre dados diferentes.

Desenvolvimento do Algoritmo Paralelo 

Desenvolvimento do Programa Paralelo 

Desenvolvimento de Programas Paralelos

- Programa Seqüencial: conjunto de comandos executados seqüencialmente (PROCESSO);
  - Atribuições, Decisão, *Loops*, Subprogramas, etc.
- Programa Paralelo: Necessidade das construções para programação seqüencial e ainda:
  - Ativação de processos em paralelo;
  - Sincronização e Comunicação.
- Necessidade de linguagens e ferramentas para o desenvolvimento de programas paralelos!

Formas de Expressar Paralelismo

- Mecanismos para ativação de processos paralelos:
  - FORK/JOIN: Ativa dois processos em paralelo;
  - COBEGIN/COEND: Ativa vários processos em paralelo;
  - DOALL: Ativação das iterações de um *loop* em paralelo.

Comunicação e Sincronismo

- Sincronização: Imposição de uma ordem na execução de processos;
- Comunicação: Permite que a execução de um processo influencie na execução do outro;
- Exemplo: Execução da operação a = a+1 paralelamente em dois processadores.

Comunicação e Sincronismo

- Exemplo:

```
proc1                          proc2
lê a
soma 1
armazena a

                               lê a
                               soma 1
                               armazena a
```

se inicialmente a = 0; Então, após a execução a =2

Comunicação e Sincronismo

- Exemplo:

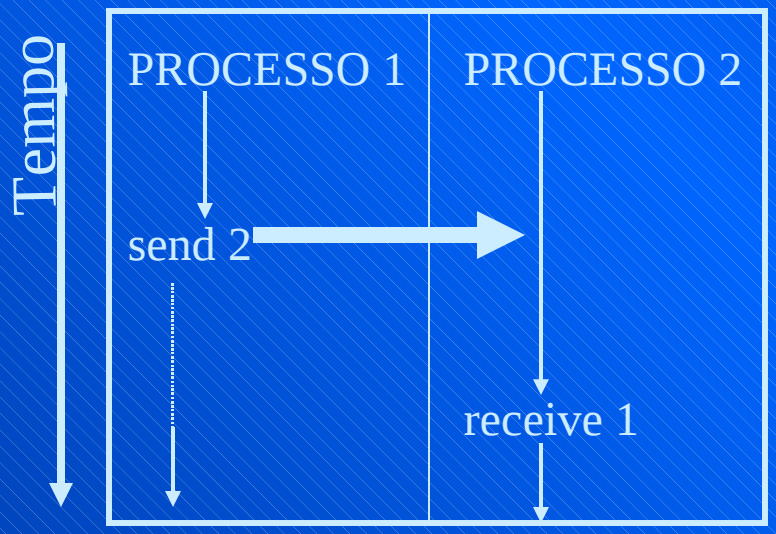
```
proc1                    proc2
lê a

                         lê a

soma 1
armazena a

                         soma 1
                         armazena a
```

No final tem-se a =1 $\rightarrow$ Faltou Sincronização

Comunicação e Sincronismo

- Sincronização em Memória Distribuída → Troca de Mensagens

- Dois processos paralelos executam as primitivas *SEND* e *RECEIVE:*
  - `SEND <lista de expressões> TO <destino>`
  - `RECEIVE <lista de variáveis> FROM <fonte>`



Comunicação e Sincronismo

- Sincronização em Memória Compartilhada $\rightarrow$ Cuidar do controle de acesso aos dados

- Problemas com:
  - Escrita em *buffer* cheio;
  - Leitura em *buffer* vazio;
  - Utilização de variáveis não calculadas.

- Estratégias para controle de acesso à memória compartilhada:
  - **Semáforos:** Variável não negativa (S) sobre a qual são definidas duas operações que devem ser executadas de forma indivisível;

  - **Monitores:** Mecanismo de mais alto nível que encapsula em um único módulo a definição do recurso e das operações que o manipulam;

Linguagens para Programação Paralela

- Fatores que devem ser considerados para a escolha de uma linguagem de programação paralela:
  - Tipo de aplicação: Similar a programação seqüencial;
  - Arquitetura a ser utilizada: Granulação e Comunicação;
  - Usuários: Profissionais da área de computação X outros usuários;
  - Tempo de aprendizagem e desenvolvimento;
  - Desempenho.

Linguagens para Programação Paralela

- Tipos de linguagens disponíveis:

Declarativas

- **Lógicas e Funcionais**
- **Granulação fina**
- **Compilador explora paralelismo**
- **Exemplos: Ling. de fluxo de dados -**
  - **Sisal, Val, Haskell**

Imperativas

- **Paralelismo explorado pelo programador**
- **Mais utilizadas**
- **Diversas opções disponíveis**

Linguagens para Programação Paralela

- Abordagens para Linguagens Imperativas
  - Compiladores que oferecem paralelização automática;
  - Extensões das linguagens seqüenciais, através de compiladores especiais;
  - Linguagens específicas para programação paralela;
  - Biblioteca de troca de mensagens para linguagem de uso geral.

Linguagens para Programação Paralela

- Paralelização automática
  - Requer pouco ou nenhum conhecimento do usuário;
  - Normalmente explora granulação fina;
  - Baixa flexibilidade;
  - *Speedup* depende da aplicação;
  - Exemplo: Computadores vetoriais.

Linguagens para Programação Paralela

- Extensões das linguagens seqüenciais através de compiladores especiais
  - Comandos adicionais para implementação de programas paralelos;
  - Opção adequada para paralelização de programas existentes;
- Vantagem:
  - Não é necessário o aprendizado de uma linguagem nova;
  - Disponibilidade de compiladores.
- Desvantagem:
  - Dificuldade em integrar as construções paralelas de forma clara;
  - Extensões podem interferir na otimização do compilador e diminuir portabilidade e desempenho.

Linguagens para Programação Paralela

- Linguagens específicas para Programação Paralela:
  - Possuem recursos para ativação e coordenação de processos mais naturais e com implementação mais eficientes;
  - Apresentam maior flexibilidade: permitem diferentes tipos de paralelismo;
  - Ferramentas para depuração e detecção de erros;
  - Normalmente apresentam melhor desempenho;
  - Baixa portabilidade;
  - Exemplos: Ada e Occam.

Linguagens para Programação Paralela

- Bibliotecas para troca de mensagens:
  - Extensão de linguagens de programação de propósito geral;
  - Originalmente foram desenvolvidas para máquinas paralelas;
  - Aplicações recuperam a portabilidade perdida ao longo do desenvolvimento da Computação Paralela;
  - Exemplos de plataformas ou ambientes portáteis: P4, Parmacs, Express, PVM, etc;
  - Devido aos diversos ambientes disponíveis, há necessidade de padronização $\rightarrow$ MPI;

Desenvolvimento do Algoritmo Paralelo 

Desenvolvimento do Programa Paralelo 

Mapeamento de Processos 

Mapeamento de Processos

- Questões a serem tratadas:
  - Onde cada processo deve ser executado?
  - Como a comunicação entre processos será viabilizada no sistema de interconexão existente?
  - De forma geral, tem-se:

  - Em Arquiteturas Paralelas com memória distribuída

Necessidade de Mapeamento

Mapeamento de Processos

- O mapeamento de processos paralelos visa minimizar o tempo de processamento;

- Duas estratégias são conflitantes:
  - Processos que podem executar concorrentemente

Mapeados em processadores diferentes

  - Processos que se comunicam freqüentemente

Mapeados no mesmo processador

Desenvolvimento do Algoritmo Paralelo 

Desenvolvimento do Programa Paralelo 

Mapeamento de Processos 

Teste e Depuração de Prog. Paralelos 

Teste e Depuração

- Características dificultam teste de programas paralelos:
  - Vários Processadores;
  - Tempo de Comunicação;
  - Baixa Visibilidade;
- Alguns erros podem ocorrer em programas paralelos:
  - Condição de Disputa;
  - *Deadlock/Livelock;*
  - Espera Infinita;
- Não-determinismo;
- Efeito de Intrusão.

Desenvolvimento do Algoritmo Paralelo 

Desenvolvimento do Programa Paralelo 

Mapeamento de Processos 

Teste e Depuração de Prog. Paralelos 

Avaliação de Desempenho de Prog. Paralelos 

Avaliação de Desempenho em Programas Paralelos

- Fatores que devem ser considerados:
  - Como coletar, analisar e entender os dados sobre o desempenho de um sistema paralelo?
  - Que funções devem ser consideradas para representar o desempenho?
  - Como comparar desempenho de programas em diferentes arquiteturas?

- Pode-se utilizar medidas como o *Speedup* e a Eficiência;

Avaliação de Desempenho em Programas Paralelos

- 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);

$$Sp = T_1 / T_p$$

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

$$Ef = Sp / p$$

Algumas Conclusões ....

- Pesquisas em computação paralela ainda continuam!
- Diversas áreas necessitam de potência computacional;
- Problema – custo de arquiteturas paralelas

Sumário

- Introdução;
- Etapas para o desenvolvimento e análise de Programas Paralelos;
- **Computação Paralela sobre Sistemas Distribuídos;**
- Resultados;
- Considerações Finais;
- Referências para consulta.

Computação Paralela sobre Sistemas Distribuídos

- Motivação:
  - Computação Paralela: Busca por melhor desempenho;
  - Sistemas Distribuídos: Compartilhar recursos;
  - Sistemas Distribuídos: Uma arquitetura MIMD com memória distribuída;
  - Convergência e
- Uma plataforma favorável para o desenvolvimento de aplicações paralelas distribuídas;
  - Granulação Grossa;
  - Problemas comuns: Balanceamento de carga, desempenho, comunicação;

Computação Paralela Distribuída

Computação Paralela sobre Sistemas Distribuídos

- Vantagens:
  - Relação Custo x Desempenho;
  - Melhorar a utilização dos recursos (hardware) já existentes;
  - Utilização de "ferramentas" conhecidas;
  - Aprendizagem relativamente fácil;

Ambientes para Troca de Mensagens

- Oferecem o suporte necessário para o desenvolvimento de aplicações paralelas em SD;
  - Bibliotecas de comunicação: extensão das linguagens seqüenciais;
- Inicialmente desenvolvidos para MPP (*Massively Parallel Processing*);
- Necessidade de tratar a heterogeneidade → Portabilidade;

Ambientes para Troca de Mensagens

- Desenvolvimento de "Plataformas Portáteis";
  - Conjunto de funções independentes de máquina, que executam em diversas plataformas de hardware;
- Diversas opções disponíveis:
  - Exemplos: PVM, Express, P4, PCODE, Parmacs, entre outros;
- Necessidade de padronização $\rightarrow$ MPI.

Exemplos de Ambientes para Troca de Mensagens

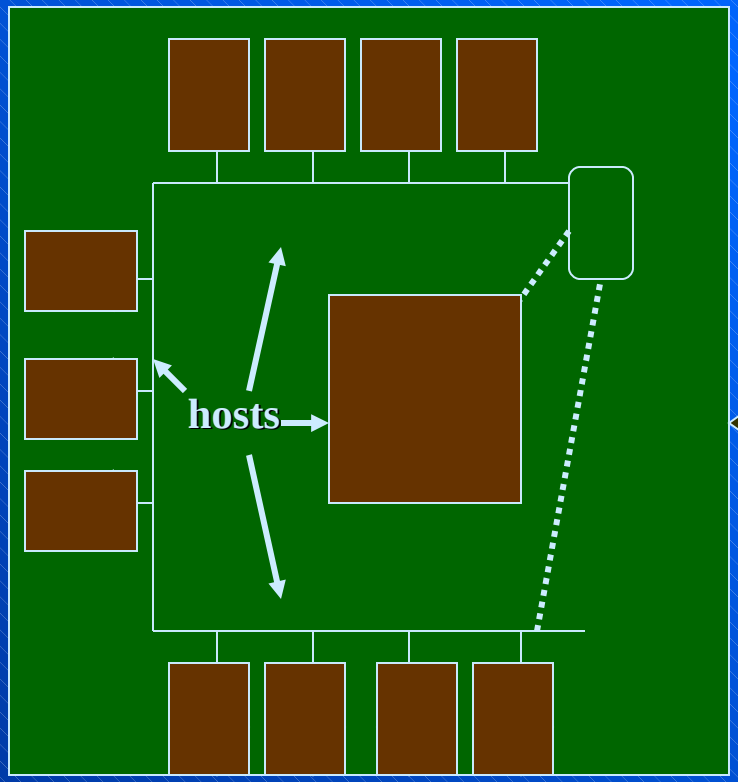
- PVM - *Parallel Virtual Machine*
  - 1ª versão - 1989;
  - ORNL, Univ. of Tenessee e Emory University;
  - Conjunto integrado de bibliotecas e ferramentas de software;
  - Emula um sistema de computação concorrente, flexível e de propósito geral;
  - Padrão “de fato”;

Exemplos de Ambientes para Troca de Mensagens

- PVM - *Parallel Virtual Machine*
  - Possibilita o desenvolvimento de aplicações em C/C++ e Fortran;
  - Todos os programas PVM devem ser "linkados" com a biblioteca PVM;
  - É um ambiente *freeware* e *open-source*;
  - Existem diversas ferramentas para "depurar" programas PVM e também verificar o estado da máquina paralela virtual.

Exemplos de Ambientes para Troca de Mensagens

- PVM - *Parallel Virtual Machine*



Visão Uniforme de uma máquina virtual paralela

Visão arquitetural

Exemplos de Ambientes para Troca de Mensagens

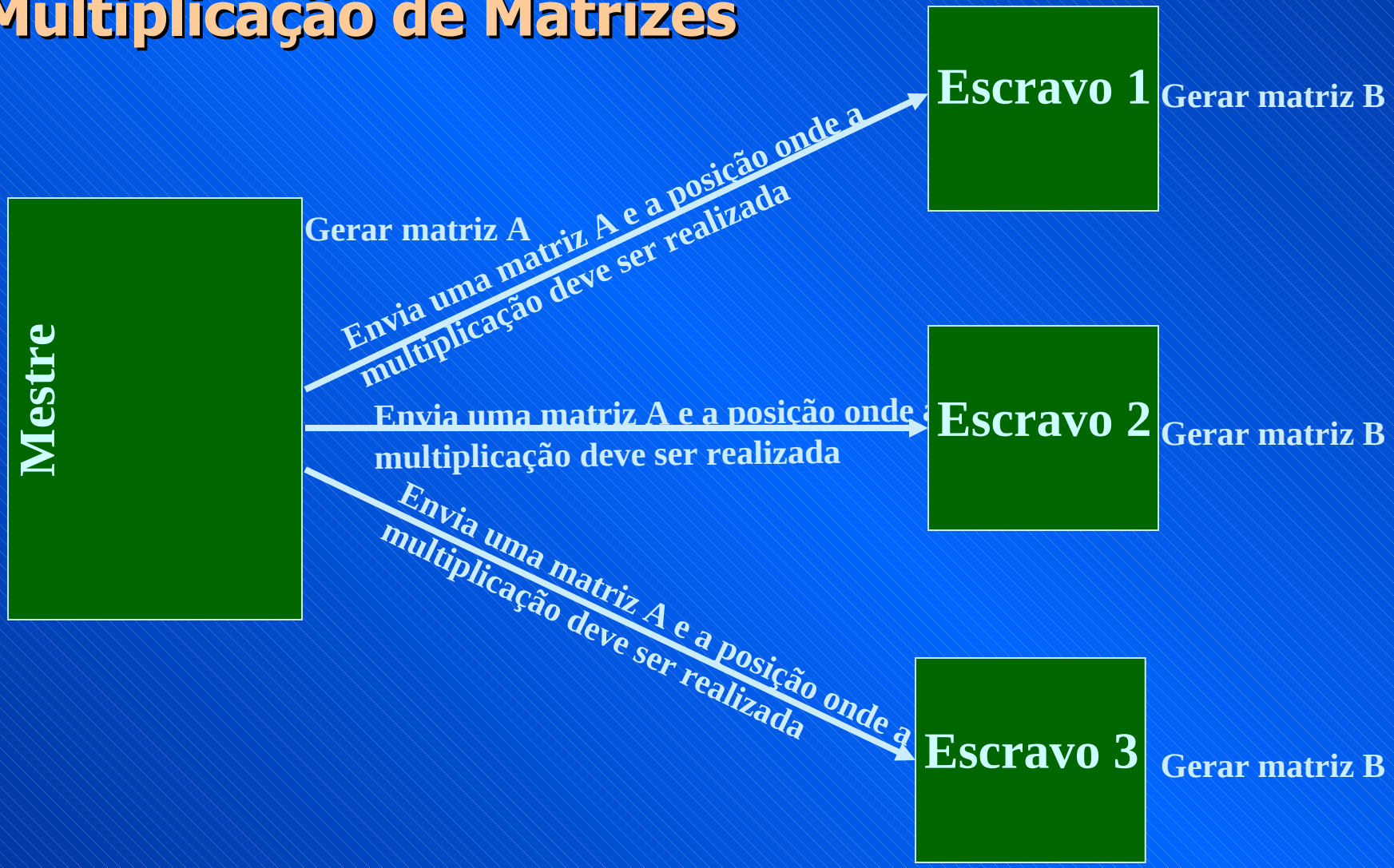
- MPI - *Message Passing Interface*
  - Início: 1992;
  - Proposta de um **Padrão** de interface de passagem de mensagens para computadores com memória distribuída e NOWs (*Networks Of Workstations*);
  - Objetivos:
    - Unir portabilidade e facilidade de uso;
    - Fornecer uma especificação precisa para o desenvolvimento de ambientes de passagem de mensagem;
    - Possível crescimento da indústria de software paralelo;
    - Difusão do uso de computadores paralelos.

Sumário

- Introdução;
- Etapas para o desenvolvimento e análise de Programas Paralelos;
- Computação Paralela sobre Sistemas Distribuídos;
- **Exemplo e Resultados;**
- Considerações Finais;
- Referências para consulta.

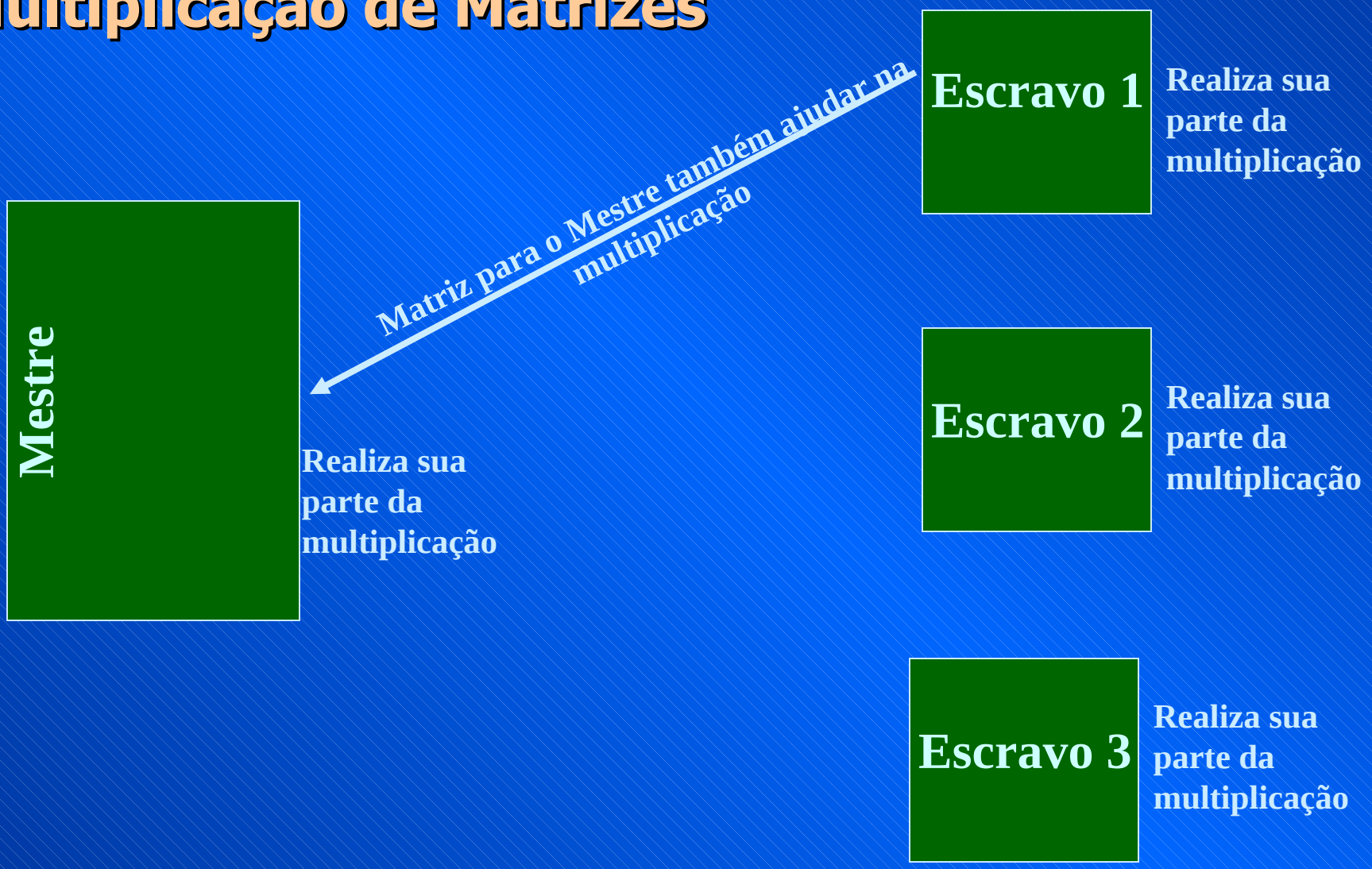
Exemplo de Aplicação em PVM e MPI

- Multiplicação de Matrizes



Exemplos de Aplicações em PVM e MPI (LAM)

- Multiplicação de Matrizes



Exemplos de Aplicações em PVM e MPI (LAM)

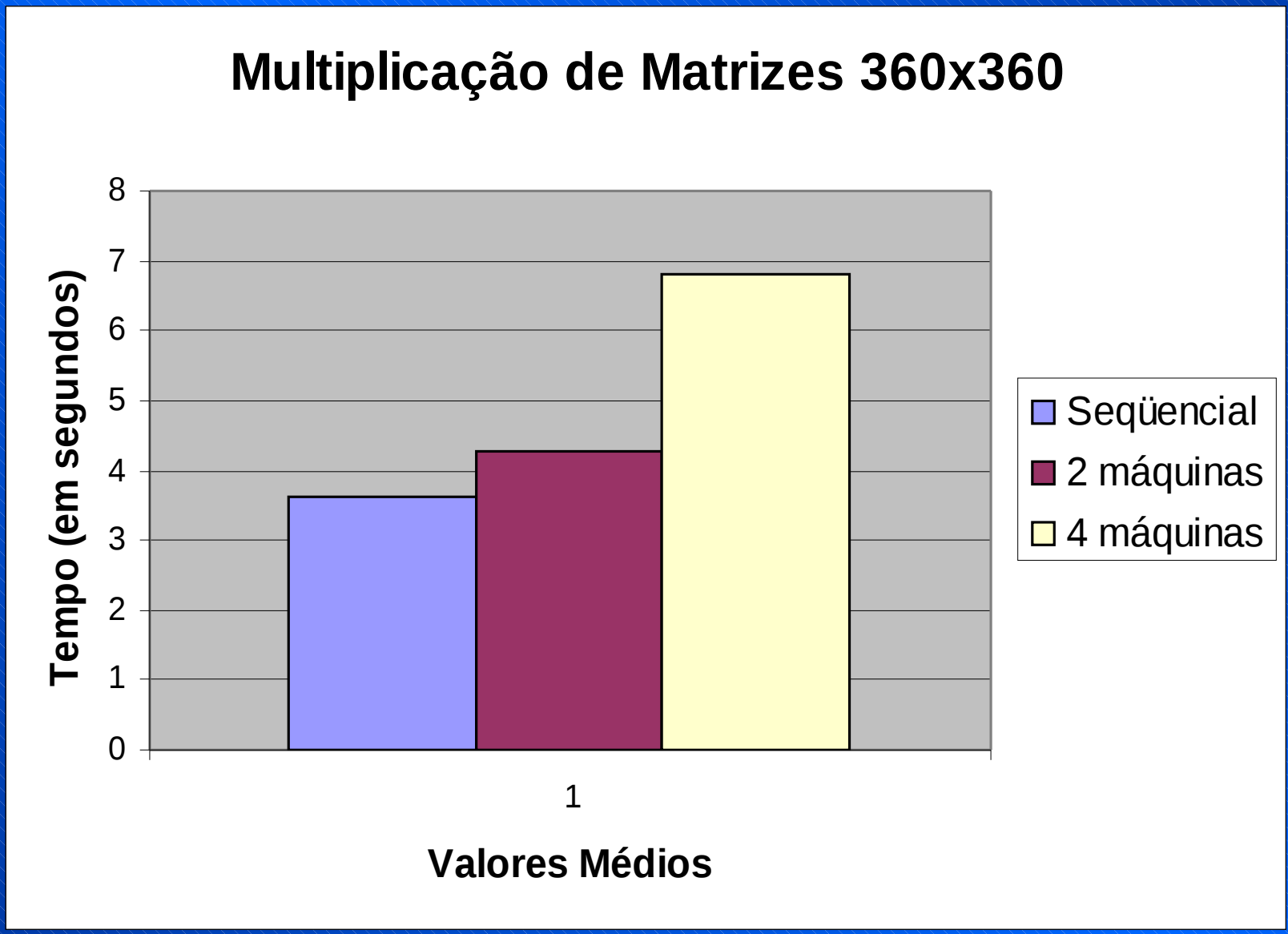
- Multiplicação de Matrizes

Resultados

- Configuração do ambiente de testes:
  - Rede *FastEthernet* 100 Mbits/sec;
  - 4 máquinas, variando de um PIII até um PentiumMMX;
  - Sistema operacional Linux com kernel 2.2.16.
- Comparações:
  - Tempo de execução seqüencial e paralelo;
  - Aplicações PVM x LAM-MPI.

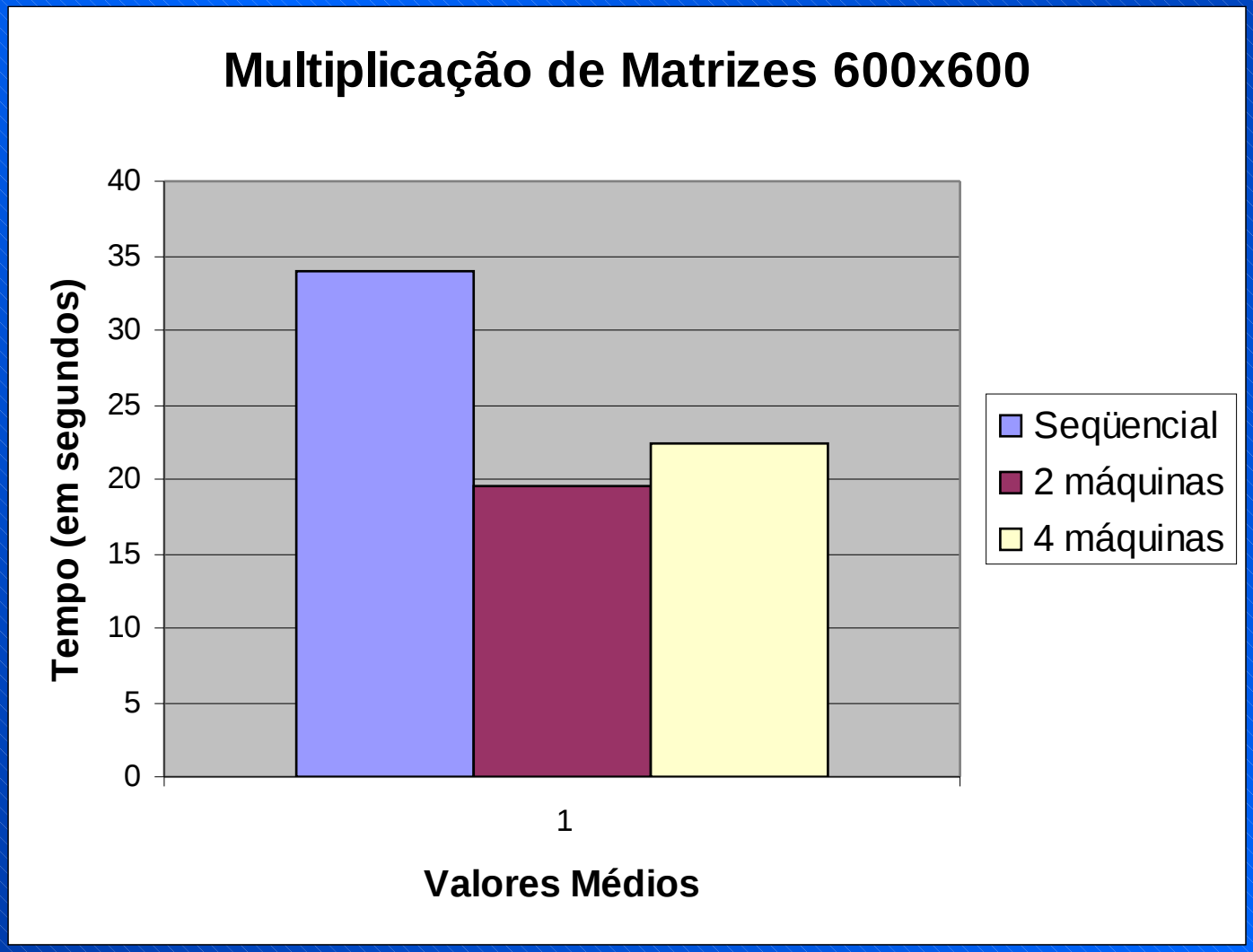
Resultados

- Multiplicação de Matrizes: PVM



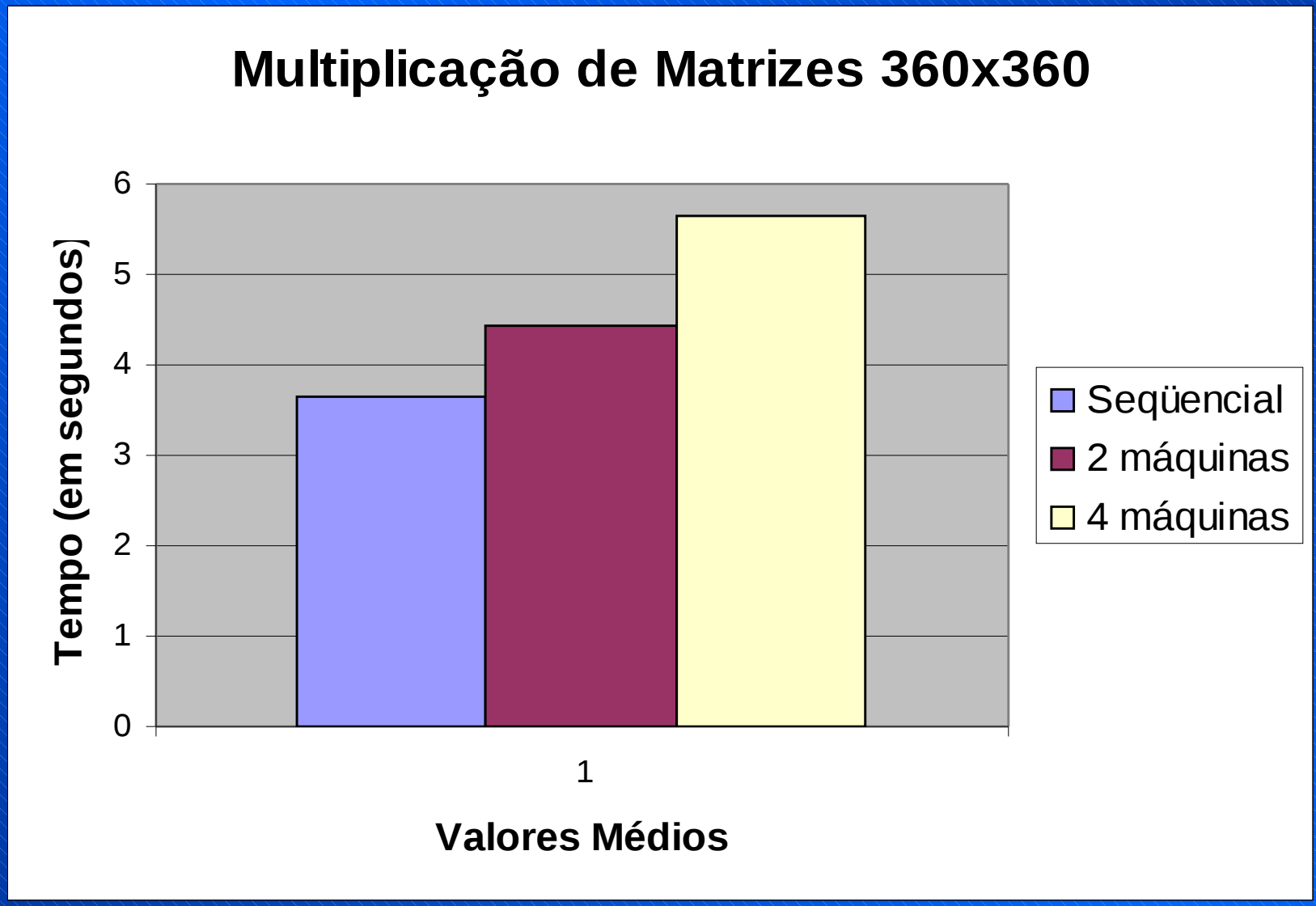
Resultados

- Multiplicação de Matrizes: PVM



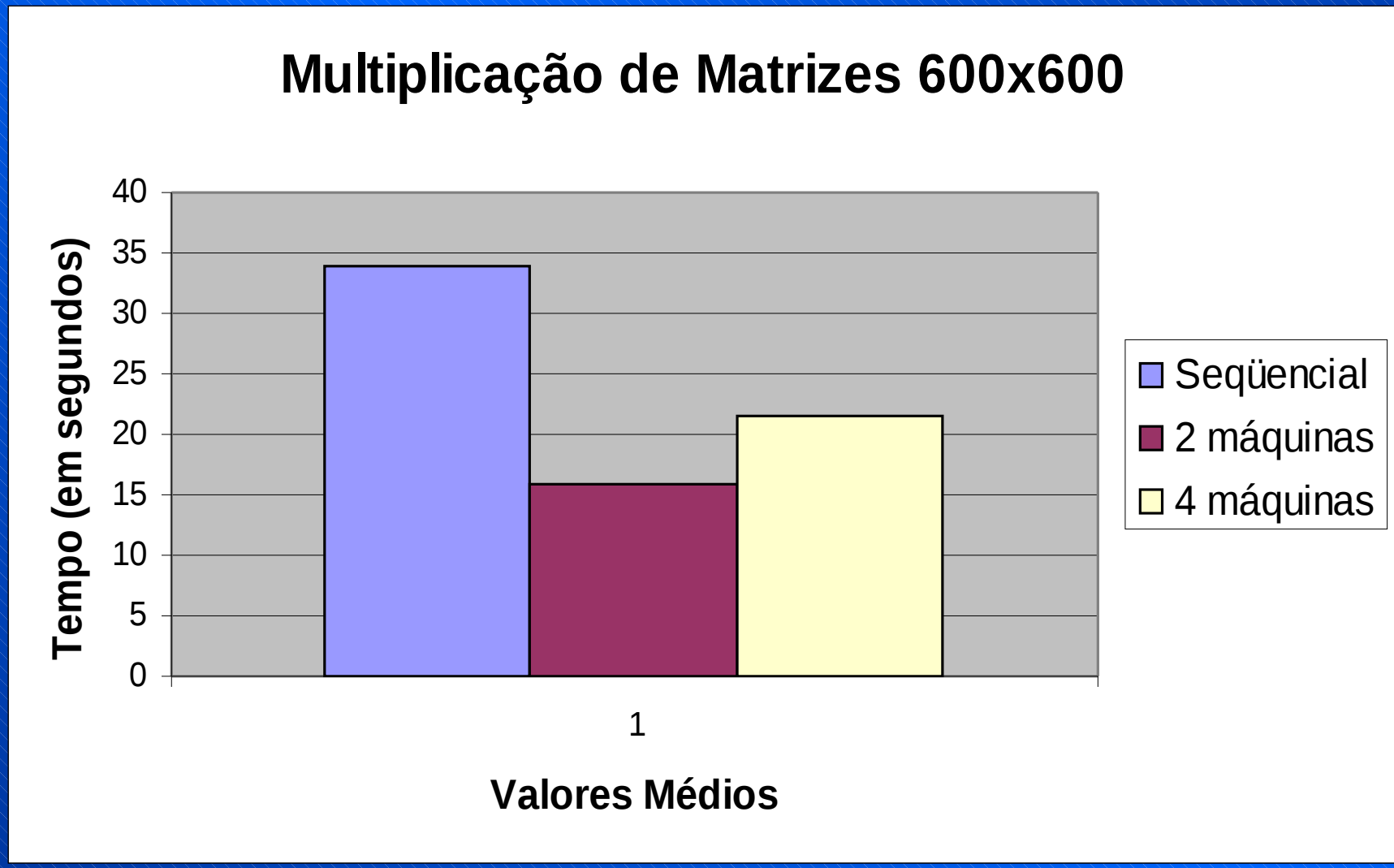
Resultados

- **Multiplicação de Matrizes: LAM-MPI**



Resultados

- Multiplicação de Matrizes: LAM-MPI



Sumário

- Introdução;
- Etapas para o desenvolvimento e análise de Programas Paralelos;
- Computação Paralela sobre Sistemas Distribuídos;
- Exemplo e Resultados;
- **Considerações Finais;**
- Referências para consulta.

Considerações finais - Resultados

Pontos que devem ser verificados:

- Aplicação é adequada para programação paralela?
  - Comunicação
  - Parte seqüencial
  - Tamanho
- Arquitetura utilizada é adequada?
  - Tipo de comunicação
  - Granularidade
- Balanceamento de carga; ☺

Considerações finais - Computação Paralela Distribuída

- União de caraterísticas e ampla disseminação;
  - Linha de pesquisa;
  - Expansão para um número maior de usuários;
- Software Paralelo: Lacunas a serem preenchidas;
  - Portabilidade, Escalabilidade e Facilidade de uso;
- Desenvolvimento de tecnologias em Redes de Computadores + Computação Paralela + Sistemas Distribuídos:
  - Motivação para uso da Computação Paralela Distribuída.

Sumário

- Introdução;
- Etapas para o desenvolvimento e análise de Programas Paralelos;
- Computação Paralela sobre Sistemas Distribuídos;
- Exemplo e Resultados;
- Considerações Finais;
- **Referências para consulta.**

Onde encontrar mais informações...

- 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.

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


Harness the Power
PVM
Condor
High Throughput Computing
Make today a LAM/MPI day!
MPI2
Perguntas??
Dúvidas??
Questões??
Powered by: LAM/MPI
PVM
Parallel Virtual Machine
LINUX Cluster
LaSDPC
Powered by LAM/MPI