INE 5616

Banco de Dados II

Prof. Mario Dantas

Conteúdo Programático

Parte I

- Sistemas Distribuídos e Paralelos
- SQL Embutida
- Indexação e Hashing
- Processamento de Consultas
- Transações

Conteúdo Programático

Parte II

6. Controle de Concorrência

7. Bancos de Dados Paralelos

8. Bancos de Dados Distribuídos

9. Bancos de Dados Orientados a Objetos

Avaliação

Provas: serão efetuadas 2 provas, a primeira abordando o conteúdo das unidades 1, 2, 3, 4 e 5 e a segunda sobre as unidades 6, 7, 8 e 9.

[Pesos: P1 – 35% P2 – 45%]

Projeto: deve ser apresentado ao final da disciplina aos colegas e ao professor.

[Peso: 20 %]

Prova de recuperação: cobrirá todo o conteúdo ministrado na disciplina, inclusive nas aulas de laboratório e nos seminários.

Obs.: serão atribuídas notas individuais aos alunos mesmo para atividades efetuadas em grupo.

Bibliografia

- Silberschatz, A.; Korth, H.; Sudarshan, S. “Sistema de Banco de Dados”, 3a Edição, Makron Books, 1999.

-Date, C. J. “Introdução a Sistemas de Bancos de Dados”, 7a Edição, Ed. Campus, 2000.

- Özsu, M.; Valduriez, P. “Princípios de Sistemas de Banco de Dados Distribuídos”, 2a Edição, Ed. Campus, 2001.

- Ramakrishnan, R. “Database Management Systems”, 2a Edição, McGraw-Hill, 2000.

Conteúdo Programático

Parte I

- Sistemas Distribuídos e Paralelos
- SQL Embutida
- Indexação e Hashing
- Processamento de Consultas
- Transações

Sistemas Distribuídos

Prof. Mario A. R. Dantas, PhD
mario@computer.org

Bibliografia

• Distributed Systems - Concepts and Design - Second Edition; George Coulouris, Jean Dollimore and Tim Kindberg; Editora Addision-Wesley, 1994.

• Distributed Operating Systems - Concepts and Design ; Pradeep K. Sinha; Editora IEEE Computer Society Press, 1997.

Conteúdo Programático

I - Conceitos Básicos

II - Sistemas Distribuídos

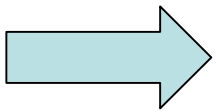
III - Troca de Mensagem
(Message Passing)

Conteúdo Programático

IV - Memória Compartilhada Distribuída
(Distributed Shared Memory -DSM)

V - Gerência de Recursos

I - Conceitos Básicos



O que é um sistema operacional (S. O.) ?

Conjunto de funções, executadas pelo processador, implementadas em *software* como um programa qualquer.
De uma forma macro, podemos dizer que temos duas importantes funções que devem ser atribuição de um sistema operacional :

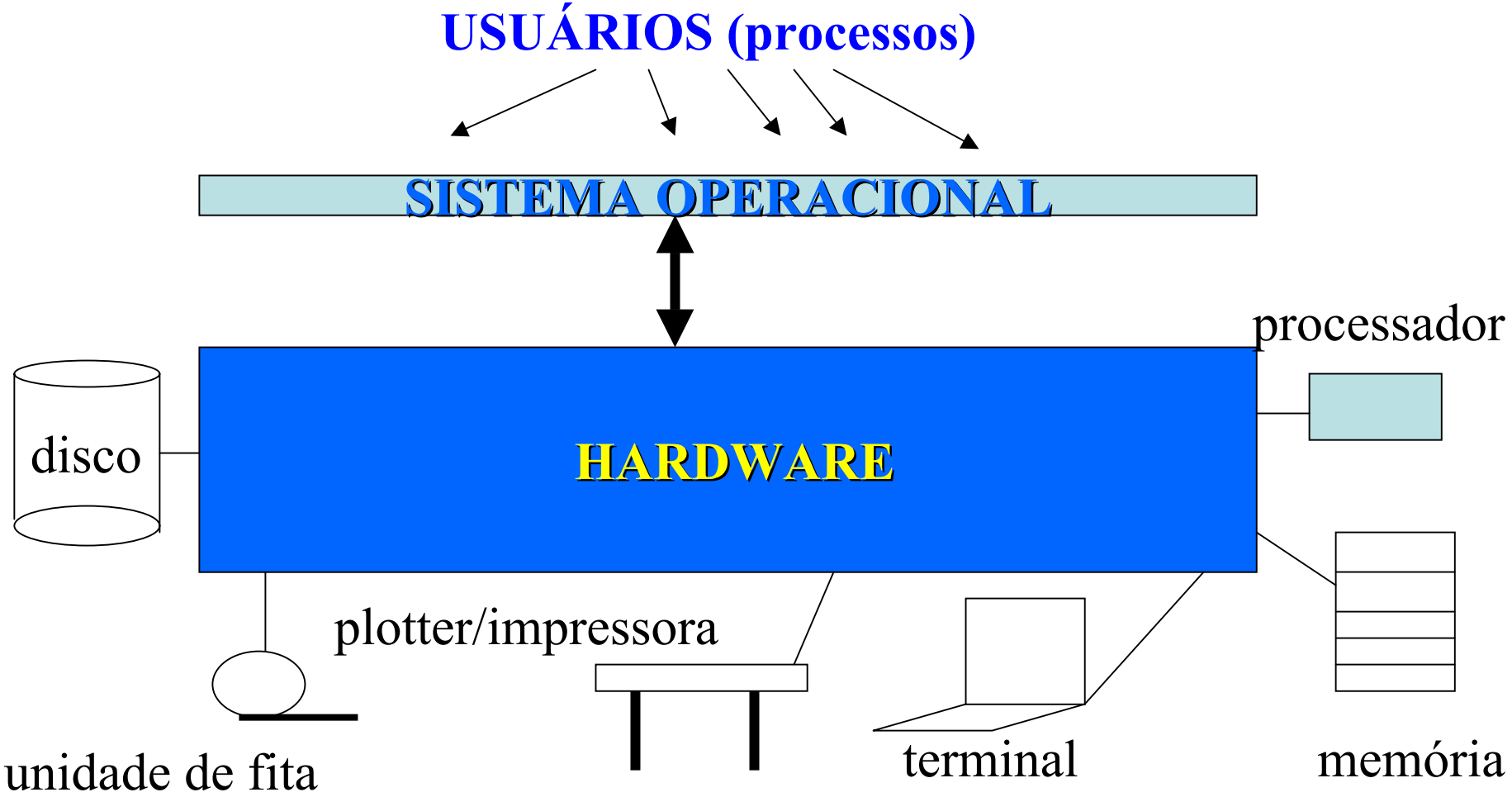
- **acesso a uma *máquina virtual*;**

- **gerenciamento dos recursos.**

Acesso a uma máquina virtual

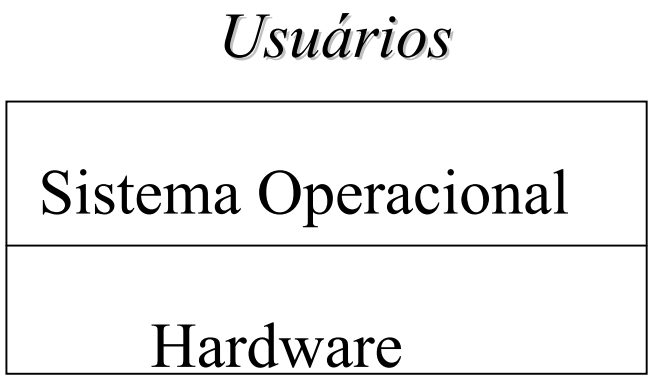
• Imagine com deve ser processada uma operação de leitura em um disco.

• Como deve ser processado o compartilhamento de acesso a um dispositivo, qualquer, de hardware ?


USUÁRIOS (processos)
SISTEMA OPERACIONAL
HARDWARE
processador
disco
plotter/impressora
unidade de fita
terminal
memória

Máquina de Níveis

Visando a simplificação da visão geral de um computador é muito comum o uso de módulos para as partes macro de um sistema.




Aplicativos
Utilitários
Sistema Operacional
Linguagem de Máquina
Microprogramação
Dispositivos Físicos

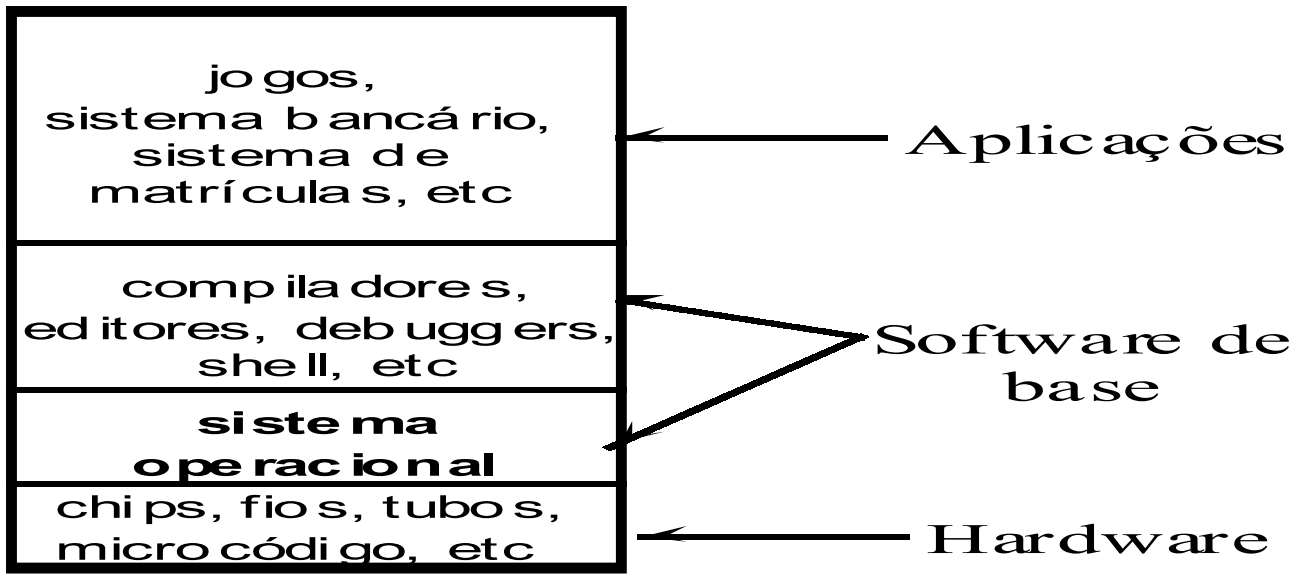
Notas : (1) os nomes *monitor, executivo, supervisor ou* controlador são algumas vezes encontrados para descrever um S.O. .

(2) **Computador = hardware + software**

hardware: componentes físicos

software: conjunto de todos os programas

O sistema operacional é um programa ou conjunto de programas.



Software de Base:

- Sistemas operacionais: modo protegido
- Utilitários: modo usuário

FUNÇÕES DO SISTEMA OPERACIONAL

- Qual a função do programa sistema operacional?
  - Algumas definições:
    - [Brinch73] *Um sistema operacional é um conjunto de procedimentos que permite que um grupo de pessoas compartilhe uma instalação de maneira eficaz*
    - [Madnick74] *O sistema operacional é o gerenciador dos recursos da máquina*
    - [Fortier86] *O sistema operacional fornece ao usuário uma visão de sua interface com a máquina*

FUNÇÕES DO SISTEMA OPERACIONAL

Um sistema operacional possui duas grandes funções: criar para o usuário uma abstração do hardware e gerenciar os recursos da máquina [Krakowiack87] [Tanenbaum92].

FUNÇÃO 1 - MÁQUINA ESTENDIDA

- A primeira função de um sistema operacional é criar para o usuário uma máquina abstrata mais simples que a máquina real.
  - A máquina abstrata ou máquina estendida é equivalente ao hardware, porém muito mais simples de manipular.

máquina física ----SO---> máquina abstrata

FUNÇÃO 1 - MÁQUINA ESTENDIDA

máquina física	máquina abstrata
1) verificar se o motor está ligado 2)posicionar o braço mecânico (bloco, setor, trilha) 3)recuperar o dado e traduzi-lo para o modo desejado 4)colocar o dado na posição indicada	1) abrir o arquivo fd=open(“arquivo”); 2) ler o arquivo read(fd,&dado);

FUNÇÃO 1 - MÁQUINA ESTENDIDA

- O programador não quer tratar de todos estes detalhes
- O programador NÃO PODE tratar de todos estes detalhes

⇨ A máquina estendida “esconde” a complexidade do hardware e protege os usuários

FUNÇÃO 2 - GERENTE DE RECURSOS

- O computador é um conjunto de recursos que serão compartilhados.
  - *recursos físicos*: processadores, memórias, discos, terminais, etc.
  - *recursos abstratos*: processos, arquivos, etc.
- Para todo recurso, o sistema operacional deve:
  - manter informações sobre o recurso (endereço, estado, etc).
  - decidir quem pode acessar o recurso
  - alocar o recurso
  - liberar o recurso
- Quanto à utilização de recursos, o SO deve:
  - ser eficiente (maximizar a utilização dos recursos)
  - possuir um tempo de resposta previsível

Gerência de Processos

- Visão Geral
  - O conceito de processo é um dos conceitos mais importantes para qualquer sistema operacional
  - ***Um processo é um programa em execução***. Além do código executável, nós temos uma pilha de execução, um apontador para esta pilha, um contador de programa, valores dos registradores da máquina, outras informações.
  - Cada processo tem um identificador único, conhecido como pid (process id).

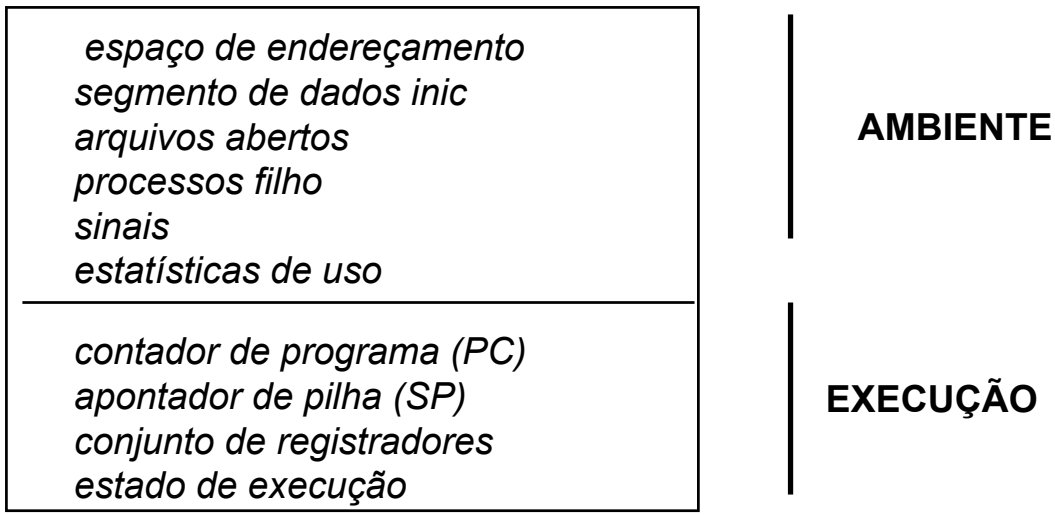
- As informações sobre um processo estão armazenadas na tabela de processos, acessada pelo pid.

- Durante a execução, o processo compartilha o processador com outros processos em execução (escalonamento de processador).

- Um processo interage com outros processos através de mecanismos de comunicação.

Modelo de Processo

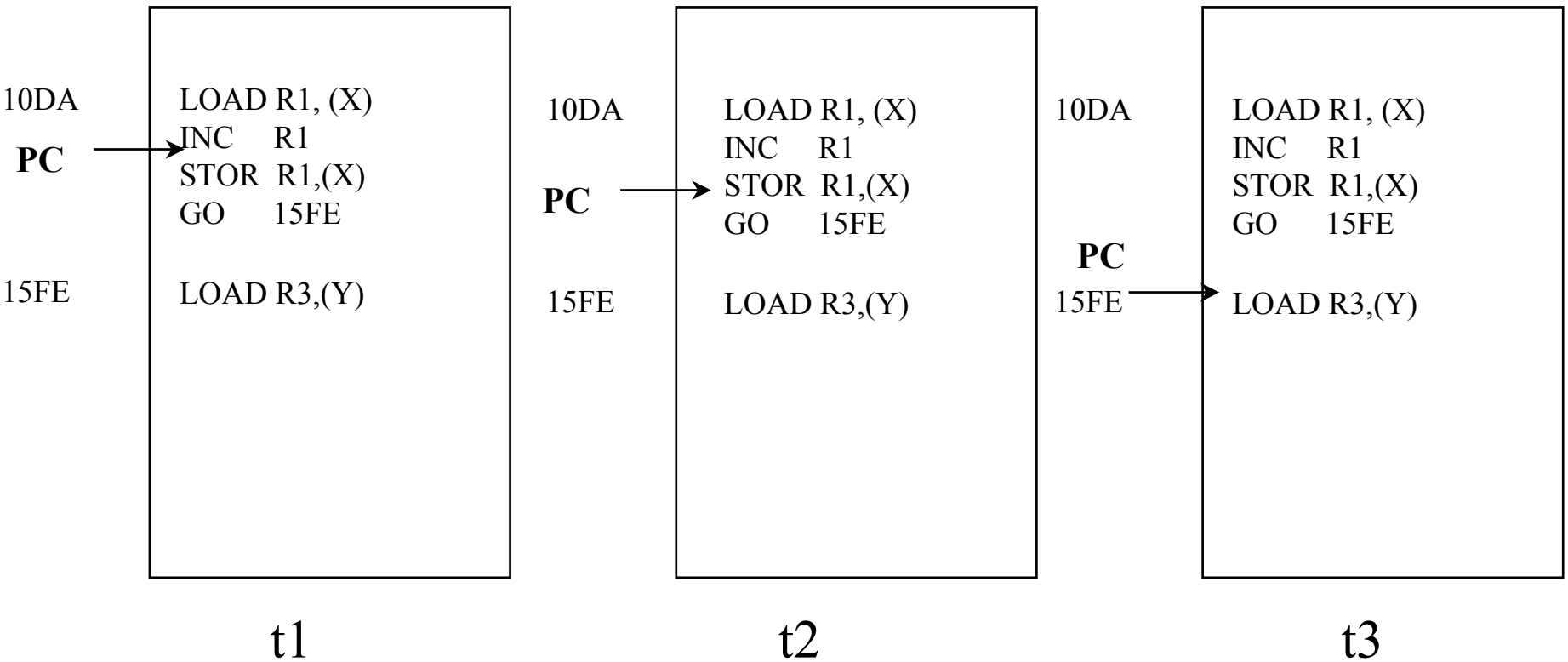
- Informações que fazem parte de um processo:



  - Classificação dos modelos de processos quanto ao custo de troca de contexto e de manutenção
    - “heavyweight” (processo tradicional)
    - “lightweight” (threads)

Modelos de Processo

- Modelo de processos tradicional (heavyweight)
  - Neste caso, o processo é composto tanto pelo ambiente como pela execução. Processo tradicional: ambiente + execução. Cada processo possui um único fluxo de controle (contador de programa) e roda de forma independente dos demais.

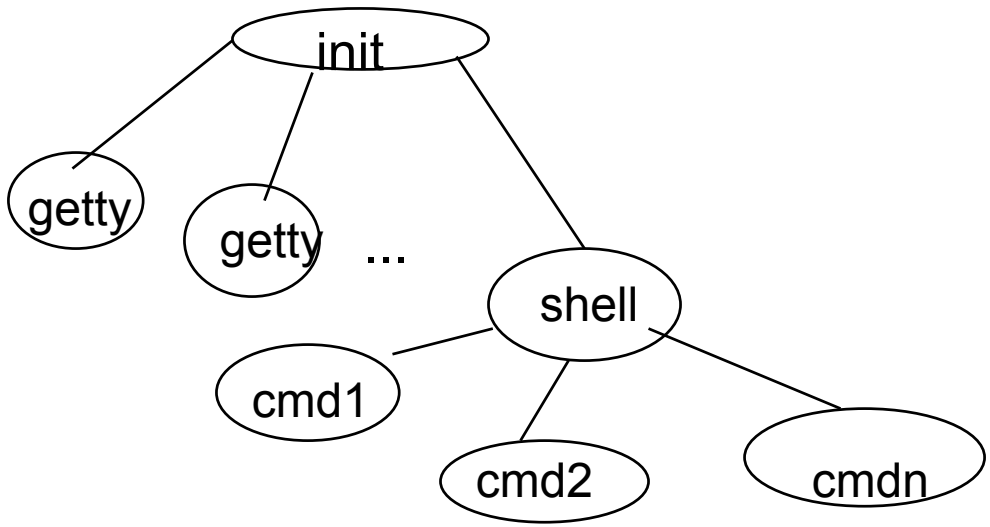


Modelos de Processo

- Como, em um dado instante, pode haver vários processos ativos ao mesmo tempo, o processador é chaveado entre os diversos processos. Por esta razão, fica praticamente impossível prever o tempo de execução de um processo, pois este dependerá da carga do sistema.
- Diferença entre um processo e um programa (por analogia):
  - programa: receita de bolo (passivo)
  - processo: ato de fazer o bolo (ativo)
- A noção de processo envolve sempre uma noção de atividade.

Modelos de Processo

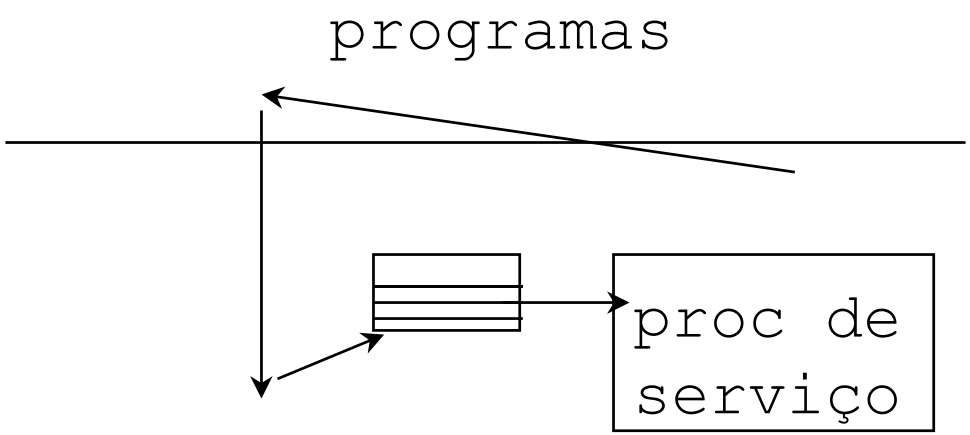
- Hierarquia de Processos
  - Todo sistema operacional deve possuir mecanismos que permitam a criação de processos. Geralmente, um processo somente é criado por outro processo, o que nos leva a uma hierarquia em árvore.
  - Um processo é criado a partir de um programa (MS-DOS) ou a partir de outro processo (clonagem) (Unix).



ESTRUTURA DOS SOs

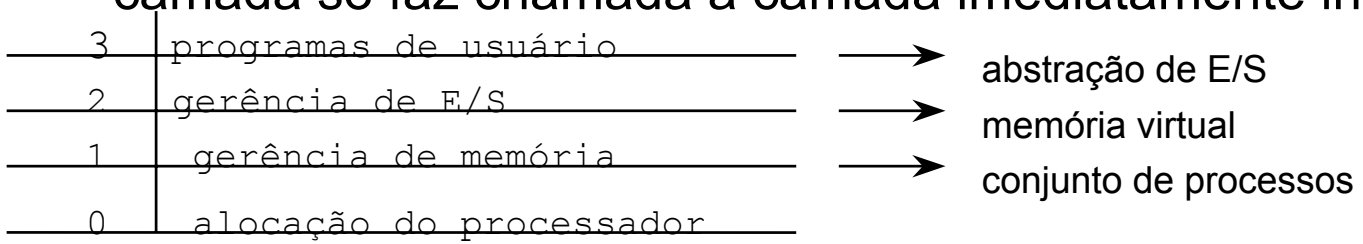
- Sistemas Monolíticos
  - É a organização de SO mais comum. Não há estrutura. O sistema é um conjunto de procedimentos que chamam um ao outro. Quanto ao tempo de resposta, é a organização mais eficaz. Pode-se obter um mínimo de estruturação se os procedimentos são forçados à fazer uma SVC (supervisor call) ou gerar um trap. É a organização do Unix comercial.
  - O sistema operacional roda em modo kernel enquanto os demais programas rodam em modo usuário.
  - O sistema operacional monolítico possui geralmente três “camadas”: um procedimento principal que chama os procedimentos de serviço, um conjunto de procedimentos de serviço que tratam as chamadas ao sistema e um conjunto de procedimentos que ajudam os procedimentos de serviço.

ESTRUTURA DOS SOs



ESTRUTURA DOS SOs

- Sistemas em Camadas
  - O sistema é organizado em camadas funcionais. Cada camada só faz chamada à camada imediatamente inferior.



  - Exemplos: THE (1968 - Dijkstra), MULTICS (BELL, MIT)
  - A noção de camadas é fortemente reforçada pelo hardware.

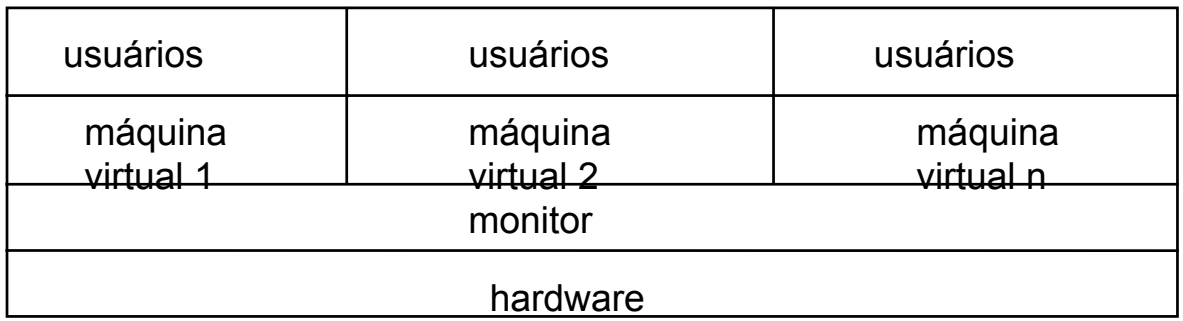
ESTRUTURA DOS SOs

- Máquinas Virtuais
  - Os sistemas operacionais estruturados como máquinas virtuais possuem, no mais baixo nível, um monitor da máquina virtual, que simplesmente implementa a multiprogramação. Em cima do monitor, várias máquinas virtuais podem ser utilizadas. As máquinas virtuais implementam uma cópia fiel do hardware, com modo kernel/usuário, E/S, interrupções, etc.

ESTRUTURA DOS SOs

- Máquinas Virtuais
  - Os sistemas operacionais estruturados como máquinas virtuais possuem, no mais baixo nível, um monitor da máquina virtual, que simplesmente implementa a multiprogramação. Em cima do monitor, várias máquinas virtuais podem ser utilizadas. As máquinas virtuais implementam uma cópia fiel do hardware, com modo kernel/usuário, E/S, interrupções, etc.

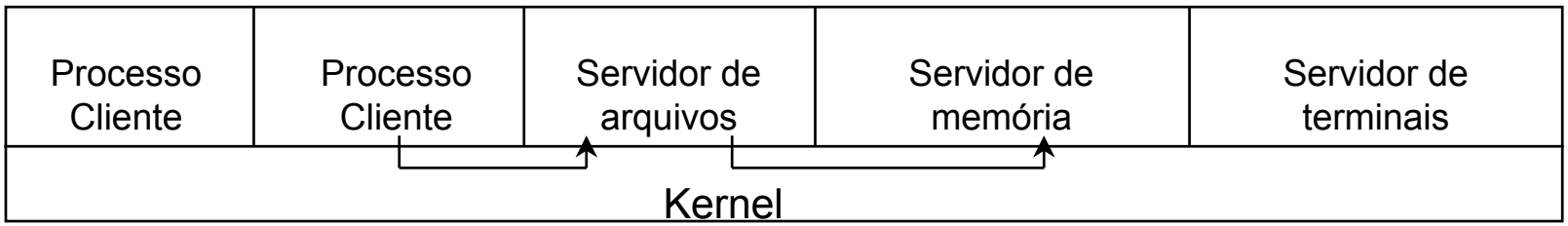
ESTRUTURA DOS SOs



Exemplo: VM/370 da IBM

ESTRUTURA DOS SOs

- Modelo Cliente-Servidor: Nesta organização, a maior parte das funções do SO é implementada à nível de usuário (processos clientes). O kernel funciona como um servidor para os processos clientes e implementa simplesmente a abstração de processos e a comunicação entre eles. Este tipo de estrutura, além de tornar mais simples o projeto do sistema operacional, faz com que ele fique mais confiável. Uma pane em um servidor de arquivos, por exemplo, não derruba o kernel. Infelizmente, existem alguns serviços, como os drivers de dispositivos de E/S, onde a maior parte dos procedimentos deve rodar no modo kernel. Estes processos são chamados servidores críticos.



Exemplo: sistema operacional distribuído Mach

HISTÓRICO

- Os sistemas operacionais estão intimamente ligados às arquiteturas nas quais eles rodam. Por isso, analisamos a evolução dos sistemas operacionais em função da evolução das arquiteturas.
- O matemático inglês Charles Babbage (1792-1871) constrói o primeiro “computador”, que não funciona devido a problemas tecnológicos.

HISTÓRICO

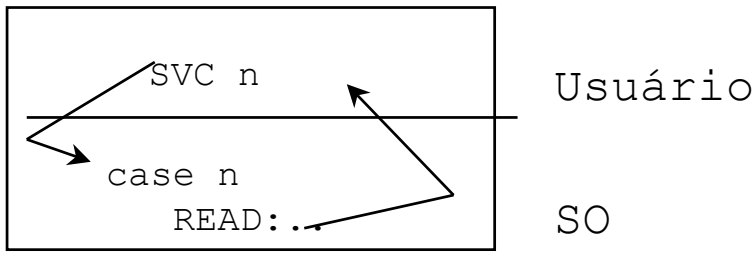
- Primeira Geração (1945-1955)
  - *Hardware*: válvulas e painéis. No início dos anos 50 aparecem os cartões perfurados e as impressoras de linha.
  - *Sistema operacional*: Não existe. O programador utiliza diretamente a console da máquina para executar seu programa (linguagem de máquina ou fios). Sistema mono-usuário. Um único grupo de pessoas concebia, construía, programava, utilisava e fazia a manutenção das máquinas.
  - *Linguagem*: linguagem de máquina

HISTÓRICO

- Segunda Geração (1955-1965)
  - *Hardware*: transistores. Os computadores começam a ser viáveis comercialmente. Aparição da fita magnética.
  - *Sistema operacional:* É feita a separação entre projetistas, construtores, operadores, programadores e técnicos de manutenção. Existe um pseudo-SO que lê os cartões, executa o programa e escreve os resultados na impressora. Sistema mono-usuário.
    - *Sistemas batch*: conceito de job (programa ou conjunto de programas com características similares). Cartões de início e fim de job. Pseudo-linguagem de controle de execução, transferência de controle automática entre jobs. A execução de um job pode ser feita ao mesmo tempo que a E/S de outros.

HISTÓRICO

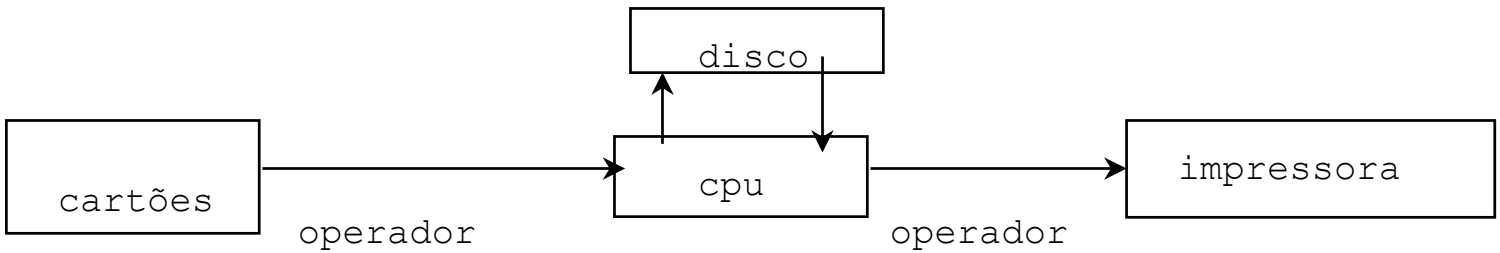
- *Modo Monitor/usuário*: é feita a proteção do SO contra o usuário. O SO é acessado através de chamadas ao sistema.



  - *Temporizador:* O usuário executa durante um tempo pré-definido (objetivo: evitar loops infinitos)
  - Menos interação entre o usuário e a máquina

– *Linguagens*: Fortran, Assembly, COBOL

HISTÓRICO



HISTÓRICO

- Terceira Geração (1965-1980)
  - *Hardware*: circuitos integrados. Aparição do disco. Surgem duas linhas diferentes de máquina: comercial e calculo científico. A IBM tenta conciliar as duas com a linha System/360. Surgem os minicomputadores (DEC PDP-1 com 4k de RAM e palavras de 18 bits - 1961).
  - *Sistema Operacional*:
    - *Portabilidade*: IBM cria a linha 360 para rodar em várias configurações de máquinas. SO enorme e complexo.
    - *Independência de dispositivos*: execução de um mesmo programa com cartões e impressora ou com fita magnética. Dispositivos lógicos de E/S.
    - *SPOOL* (Simultaneous Peripheral Operation On Line): leitura dos cartões e escrita em disco.

HISTÓRICO

- *Multiprogramação*: Surgiu para evitar que a CPU fique ociosa enquanto o processador espera o término de uma operação de E/S. Vários jobs estão em execução ao mesmo tempo. Aparecem os processadores dedicados à E/S
- *Sistemas time-sharing*: visam garantir um tempo de resposta menor aos usuários. Surgem os terminais.
- *Gerência de memória*: para executar um programa que não cabe inteiramente na memória, são propostas duas soluções: overlay e memória virtual.

HISTÓRICO

- *Projeto do Sistema Multics*, cujo objetivo é fornecer bastante poder computacional à centenas de usuários. Projeto conjunto do MIT, GE e Bell. Enorme sucesso acadêmico. Enorme fracasso comercial.
- *Nascimento do Unix*, sistema operacional baseado no MULTICS escrito em C.

– *Linguagens:* PL/I, APL, Algol, B, C, etc.

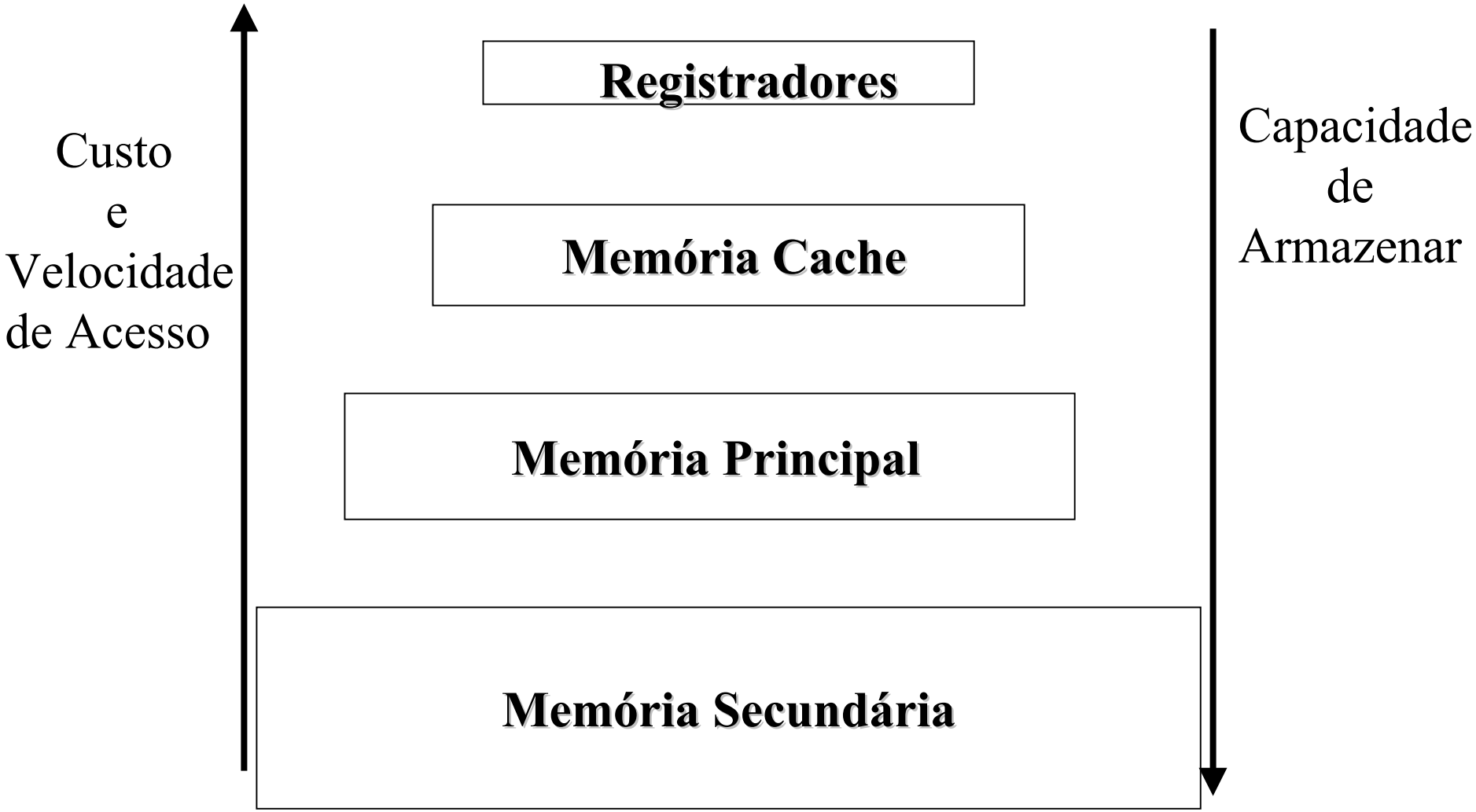
HISTÓRICO

- Quarta Geração (1980-1990)
  - *Hardware*: circuitos LSI (large scale integration) fazem com que o custo do computador caia bastante. Disseminação dos microcomputadores. Aparição do *mouse*.
  - *Sistema operacional:* Leva-se em conta a interface com o usuário na concepção dos pacotes de software e do sistema operacional. Dois sistemas dominam o mercado: MS-DOS e Unix.
    - *Redes de computadores*: os usuários estão conscientes de que várias máquinas compõem o sistema e endereçam solicitações explicitamente a elas: Apollo/Domain, Appletalk, Token Ring, etc.
    - *Sistemas distribuídos*: as diferentes máquinas que compõem o sistema são percebidas pelo usuário como uma única máquina virtual.

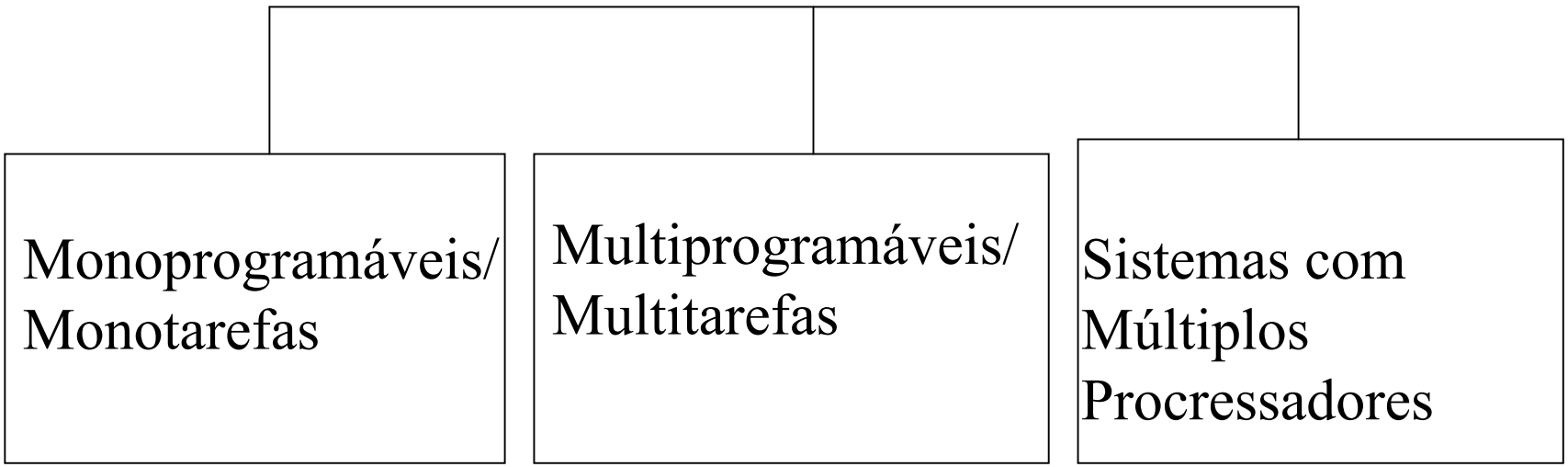
HISTÓRICO

- Quarta Geração (1990-20xx)
  - *Hardware*: circuitos ULSI (ultra large scale integration) fazem com que o custo do computador caia ainda mas. Disseminação de supercomputadores vetoriais (Cray, Fujitsu); máquinas paralelas e massivamente paralelas (Meiko, CM-5); dos sistemas Beowulf.
  - *Sistema operacional:* Leva-se em conta a interface com o usuário na concepção dos pacotes de software e do sistema operacional. Dois sistemas dominam o mercado: Windows e Unix.
    - *Redes de computadores*: grande redes WANs *(Internet*);
    - *Sistemas distribuídos e paralelos*: as diferentes máquinas que compõem o sistema são percebidas pelo usuário como uma única máquina virtual.

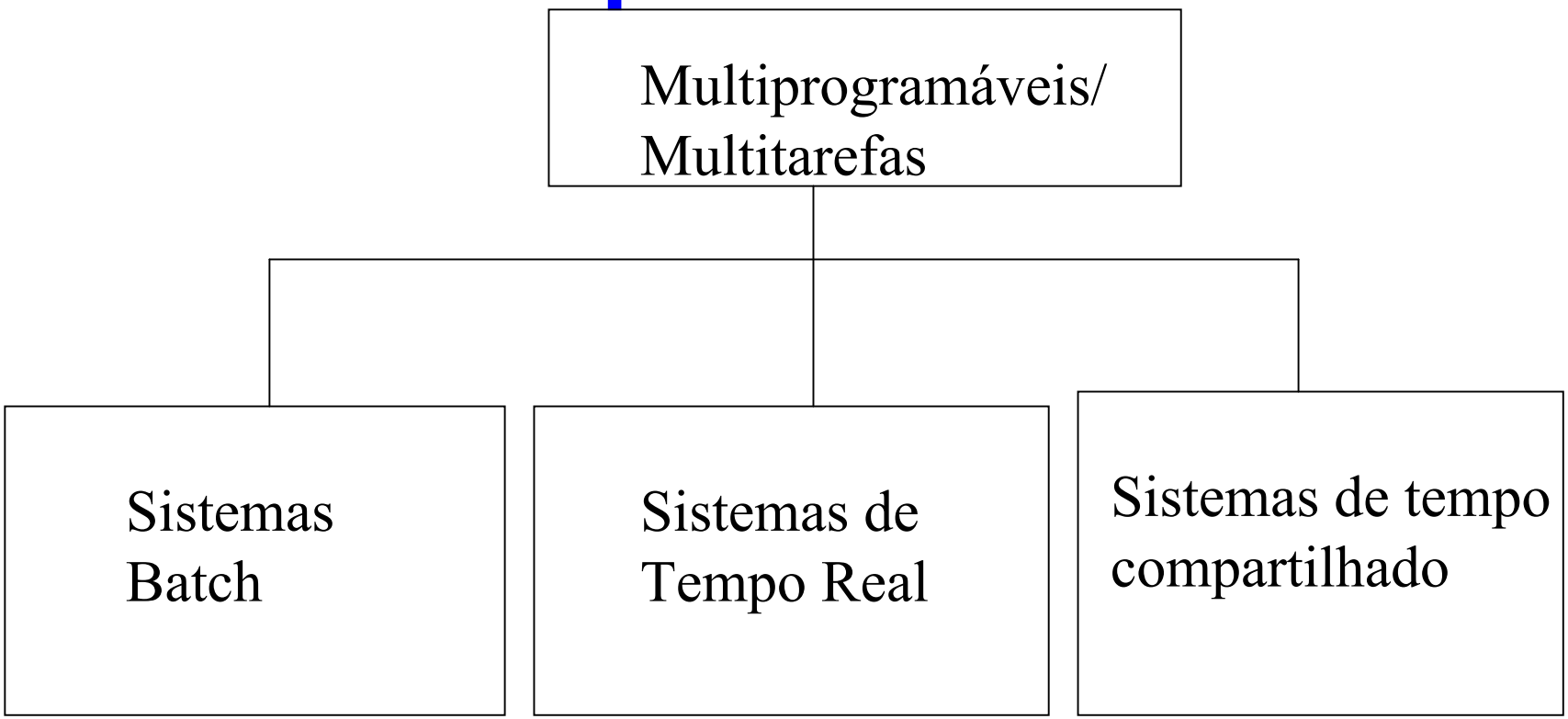
Conceitos de Hardware/Software



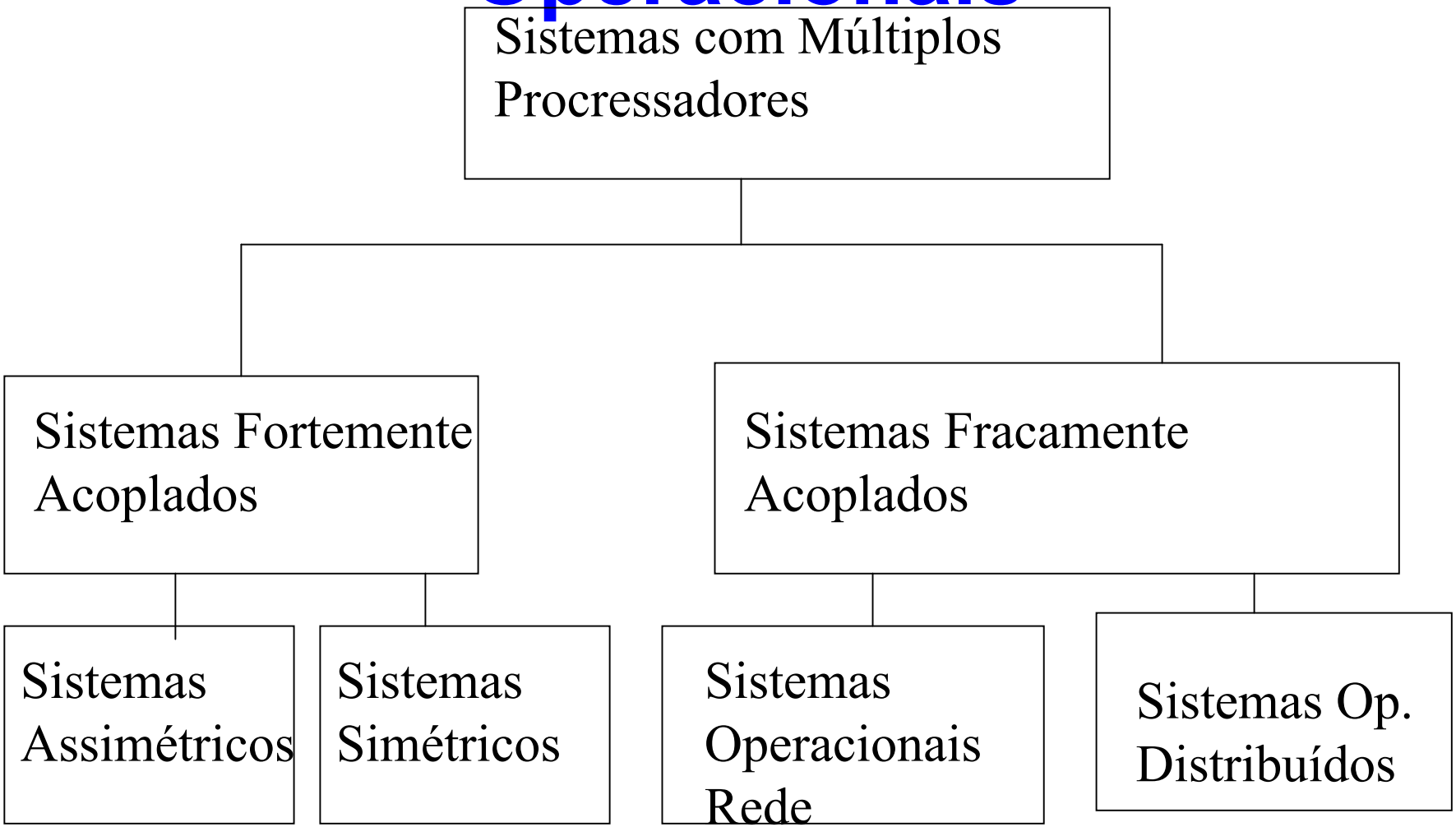
Tipos de Sistemas Operacionais



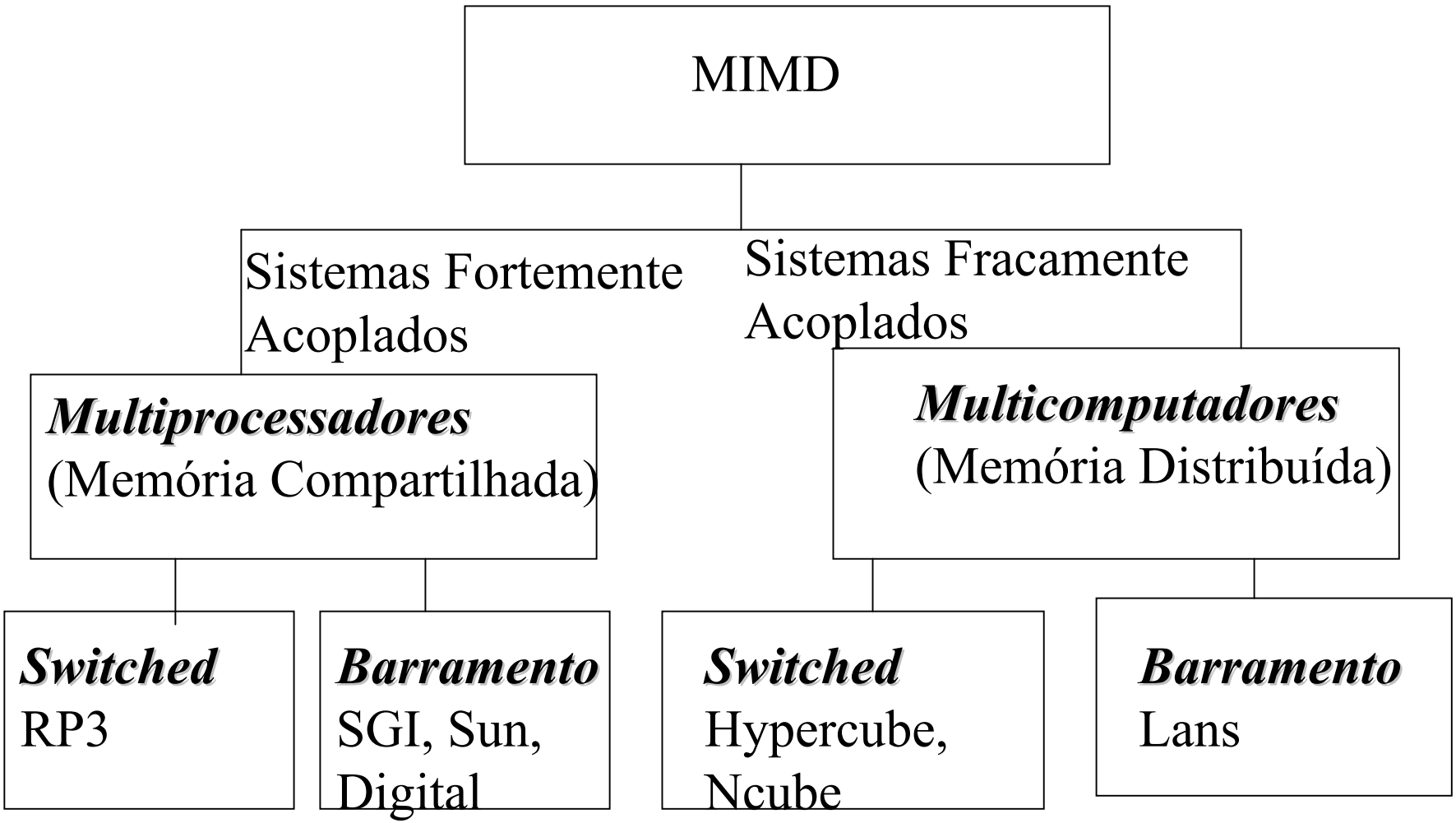
Tipos de Sistemas Operacionais



Tipos de Sistemas Operacionais



Taxonomia de Flynn - Extendida



II - Sistemas Distribuídos

Quando buscamos diminuir o tempo de um processamento temos três caminhos a seguir:

- Processar mais rápido;
- Processar de forma mais inteligente;
- Pedir ajuda, dividindo o processamento.

Sistemas Distribuídos

A ***primeira alternativa*** implica na adoção de soluções de hardware mais eficientes, o que depende, em primeira instância, do desenvolvimento de microprocessadores mais rápidos. Neste tocante, Gordon E. Moore publicou em 1965 sua previsão de que a performance dos processadores dobraria a cada 18 meses, seguindo uma taxa de crescimento de 48% ao ano, o que ficou conhecida como a ***Lei de Moore.***

Sistemas Distribuídos

Hoje, assistimos com assombro o cumprimento de suas previsões ao longo destes anos, mas fatores físicos tais como a dissipação do calor impedirão que estas taxas de crescimento se mantenham elevadas indefinidamente.

Sistemas Distribuídos

Mesmo com a capacidade de processamento atingida pelos microprocessadores atuais, ainda assim existem aplicações cujos requisitos de recursos computacionais são tão elevados que não há como serem executadas em sistemas centralizados, nem hoje nem em um futuro próximo.

Sistemas Distribuídos

O ***segundo aspecto*** diz respeito ao emprego de algoritmos otimizados. Ao se analisar a performance de um algoritmo dois são os critérios de maior relevância :

- **complexidade no tempo e**

- **complexidade no espaço.**

A primeira refere-se ao tempo de processamento e a segunda à memória utilizada.

Sistemas Distribuídos

Exprime-se complexidade através de notação assintótica, da qual destaca-se a função $f(n) = O(g(n))$, que define $g(n)$ como limite superior para $f(n)$ a menos de um valor constante, ou seja $f(n) \leq c * g(n)$, para c constante e $n \geq n_0$.

Alguns problemas não podem ser abordados com a capacidade do hardware atual porque a complexidade dos algoritmos que os solucionam (e.g. $O(2^n)$, $O(n^3)$ e $O(n^2)$) inviabiliza seu uso para n muito grande.

Sistemas Distribuídos

Esperar pelo desenvolvimento de um hardware mais eficiente ou pela descoberta de um algoritmo otimizado (e.g. *O*(log n) e *O*(n)) para resolver um problema não são alternativas aceitáveis, às vezes, pois que o prazo para tais soluções ocorrerem não pode estimado.

Sistemas Distribuídos

A ***terceira alternativa***, pedir ajuda, permite-nos trabalhar com o hardware e os algoritmos existentes hoje, com o emprego de um ambiente de processamento paralelo.

Sistemas Distribuídos

A fim de conceituarmos sistemas paralelos, vamos primeiro conceituar **sistemas distribuídos**.

Na literatura não há consenso, neste curso seguiremos o conceito que define sistema distribuído como :

"um conjunto de elementos [de processamento (EP)] que se comunicam através de uma rede de interconexão e que utilizam software de sistema distribuído. Cada elemento é composto por um ou mais processadores e uma ou mais memórias."

Sistemas Distribuídos

Sistemas paralelos, podem ser conceituados como uma forma restrita de sistemas distribuídos, onde o objetivo de todo o sistema é resolver um único problema no menor tempo possível, buscando, desta forma, otimizar a performance.

Sistemas distribuídos, então, são mais genéricos, admitindo várias outras formas de otimização.

Sistemas Distribuídos

Os *sistemas distribuídos* são mais complexos quando comparados com os *sistemas centralizados*. Alguns pontos que devem ser trabalhados nos sistemas distribuídos são :

- comunicação eficiente entre os componentes do sistema ;
- tratamento de perda de mensagens ;
- tratamento especial para evitar congestionamento da rede ;
- segurança ;

Sistemas Distribuídos

Os ambientes distribuídos devem seu crescimento de popularidade aos seguintes pontos :

- Aplicações Originalmente Distribuídas

(Exemplos : reservas de passagens, caixas eletrônicos)

- Informações Compartilhadas entre Usuários Distribuídos

(Exemplos : projetos desenvolvidos em localidades geograficamente distribuídas - *Computer-Supported Cooperative Working (CSCW))*

Sistemas Distribuídos

- Compartilhamento de Recursos ;
- Melhor relação preço-desempenho ;
- Melhoria no tempo de resposta e alto-desempenho ;
- Melhoria na Confiabilidade do ambiente ;
- Crescimento Modular ;
- Melhor flexibilidade para as solicitações dos usuários;

Sistemas Distribuídos

Tanenbaum e Van Renesse definem um *sistema operacional distribuído* da seguinte forma :

Um sistema operacional distribuído é aquele qua parece aos olhos dos usuários como sendo um sistema operacional centralizado, todavia este executa em múltiplas, independentes unidades centrais de processamento (UCP). O conceito fundamental é a transparência. Em outras palavras, o uso de múltiplos processadores deve ser invisível para o usuário. De outra forma, poderíamos dizer que o usuário deve ver o sistema como sendo uma máquina monoprocessada virtual, não como uma coleção de máquinas.

Sistemas Distribuídos

No projeto de um sistema distribuído, inúmeros pontos de *transparência* devem ser alcançados :

- *Acesso ;*
- *Localidade - nome do recurso e usuário ;*
- *Replicas ;*
- *Falhas ;*

Sistemas Distribuídos

- *Migração de processos ;*
- *Concorrência ;*
- *Desempenho ;*
- *Escalabilidade ;*

Sistemas Distribuídos

Exemplos de Sistemas Operacionais Distribuídos são :

- Amoeba : http://www.cs.vu.nl/vakgroepen/cs/amoeba_papers.html

  ftp://ftp.cs.umanitoba.ca/pub/bibiographies/Distributed/amoeba.html
- V-System: http://www-dsg.stanford.edu/Publications.html

  ftp://ftp.cs.umanitoba.ca/pub/bibliographies/Os/os.html
- Mach :

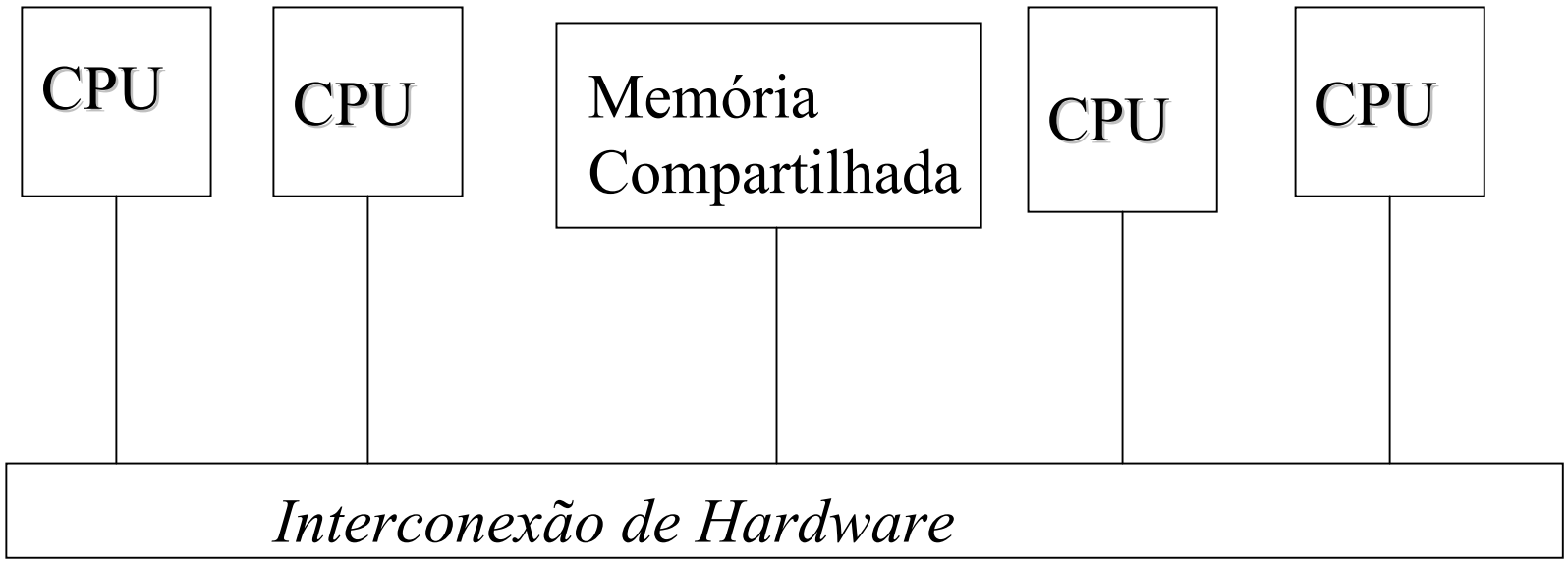
  http://www.cs.cmu.edu/afs/cs/project/mach/public/www/doc/publications.html

  ftp://ftp.cs.umanitoba.ca/pub/bibliographies/Distributed/Mach.html
- Chorus :

  ftp://ftp.cs.umanitoba.ca/pub/bibliographies/Distributed/chorus.html

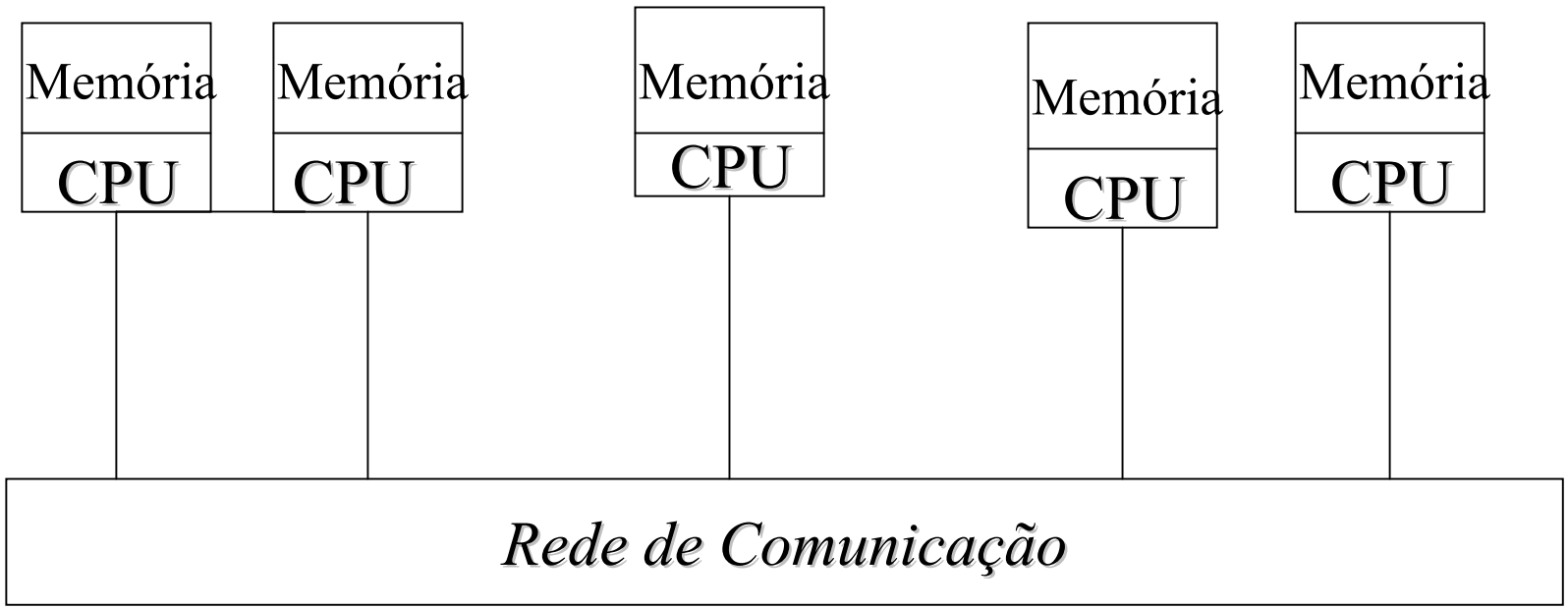
Multiprocessadores *vs* Multicomputadores

Multiprocessador (tightly coupled-system)



Multiprocessadores *vs* Multicomputadores

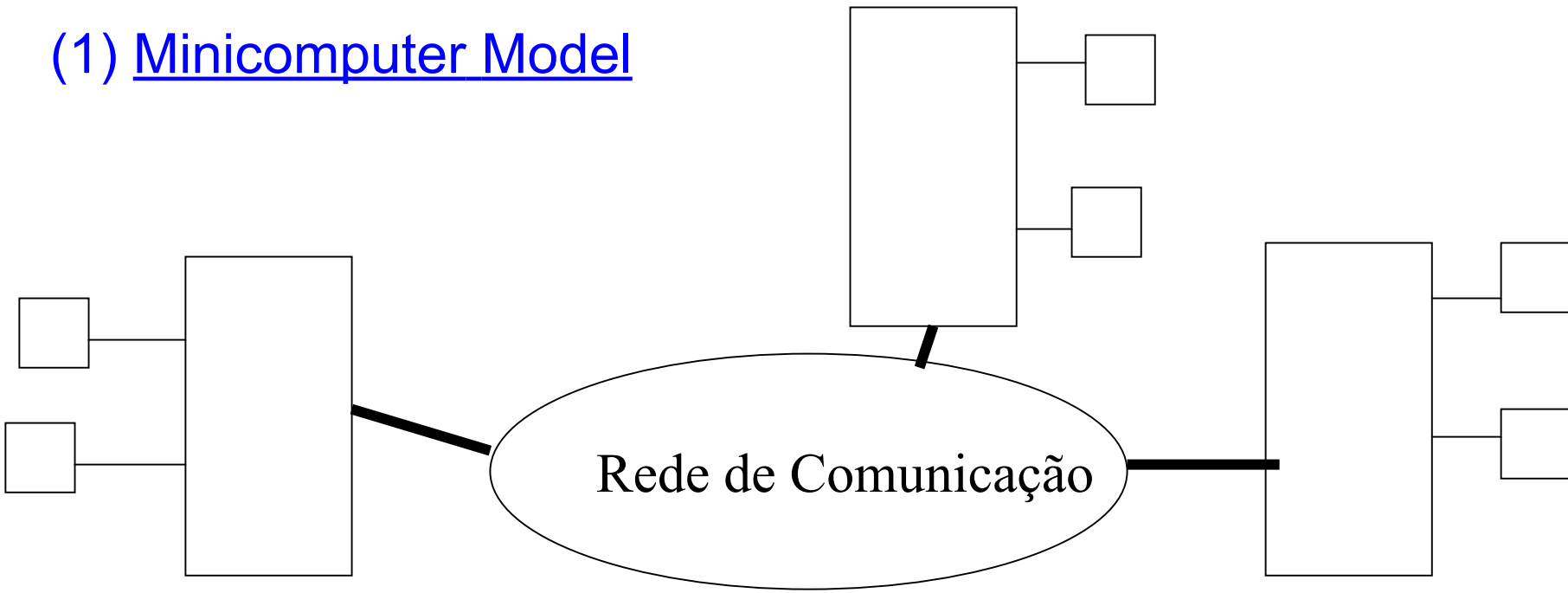
Multicomputador (loosely coupled-system)



Modelos de Sistemas Distribuídos

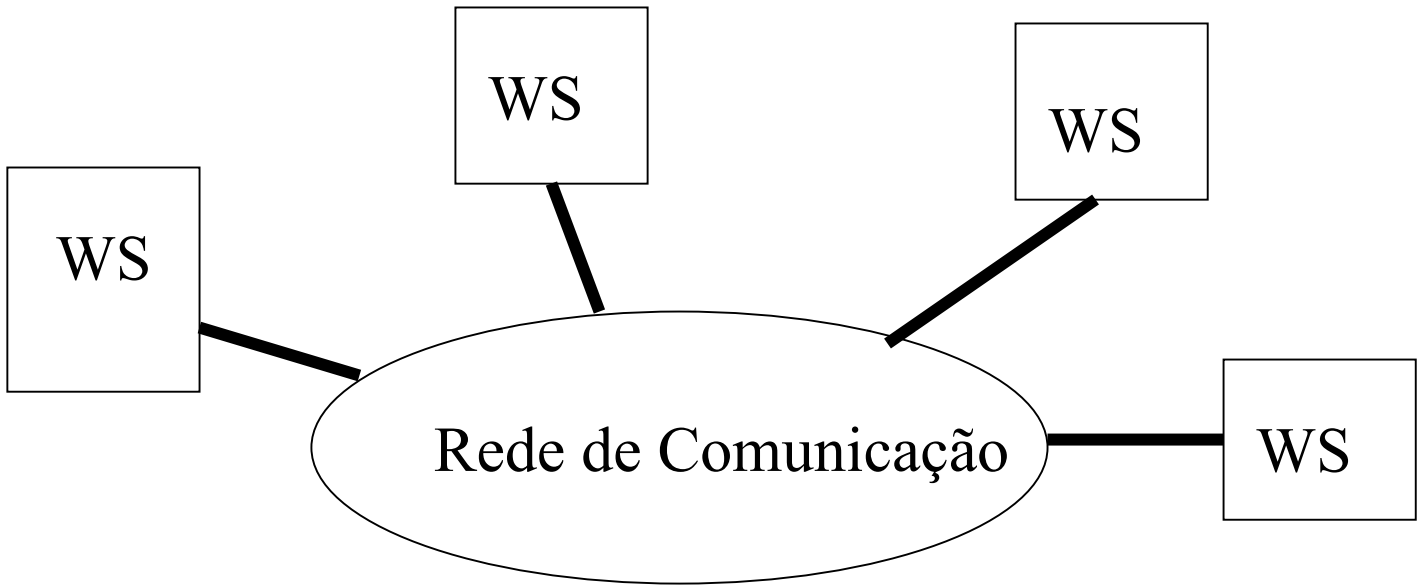
- São cinco os modelos, usualmente, classificados na construção dos sistemas distribuídos :

(1) Minicomputer Model



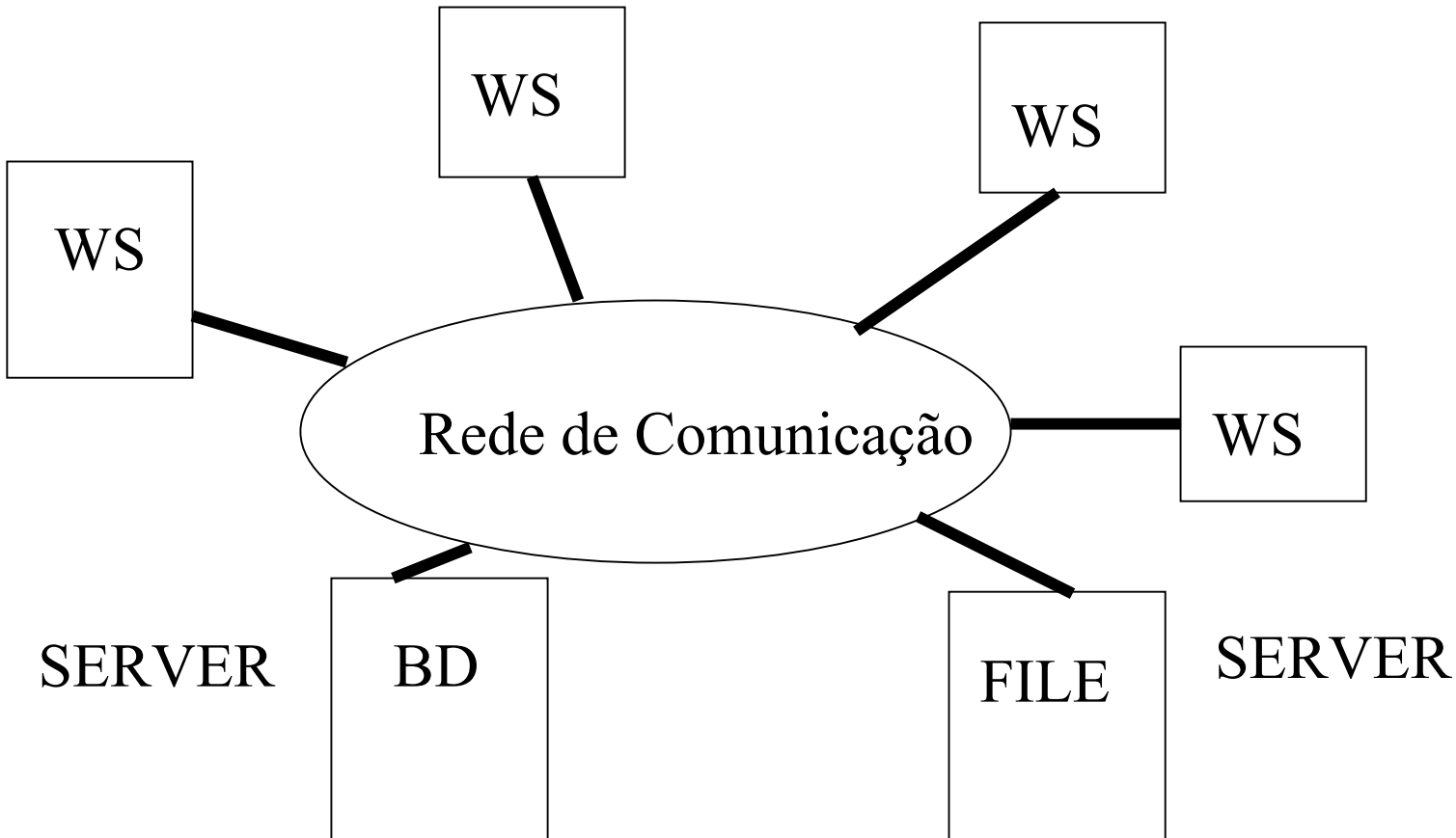
Modelos de Sistemas Distribuídos

(2) Workstation Model



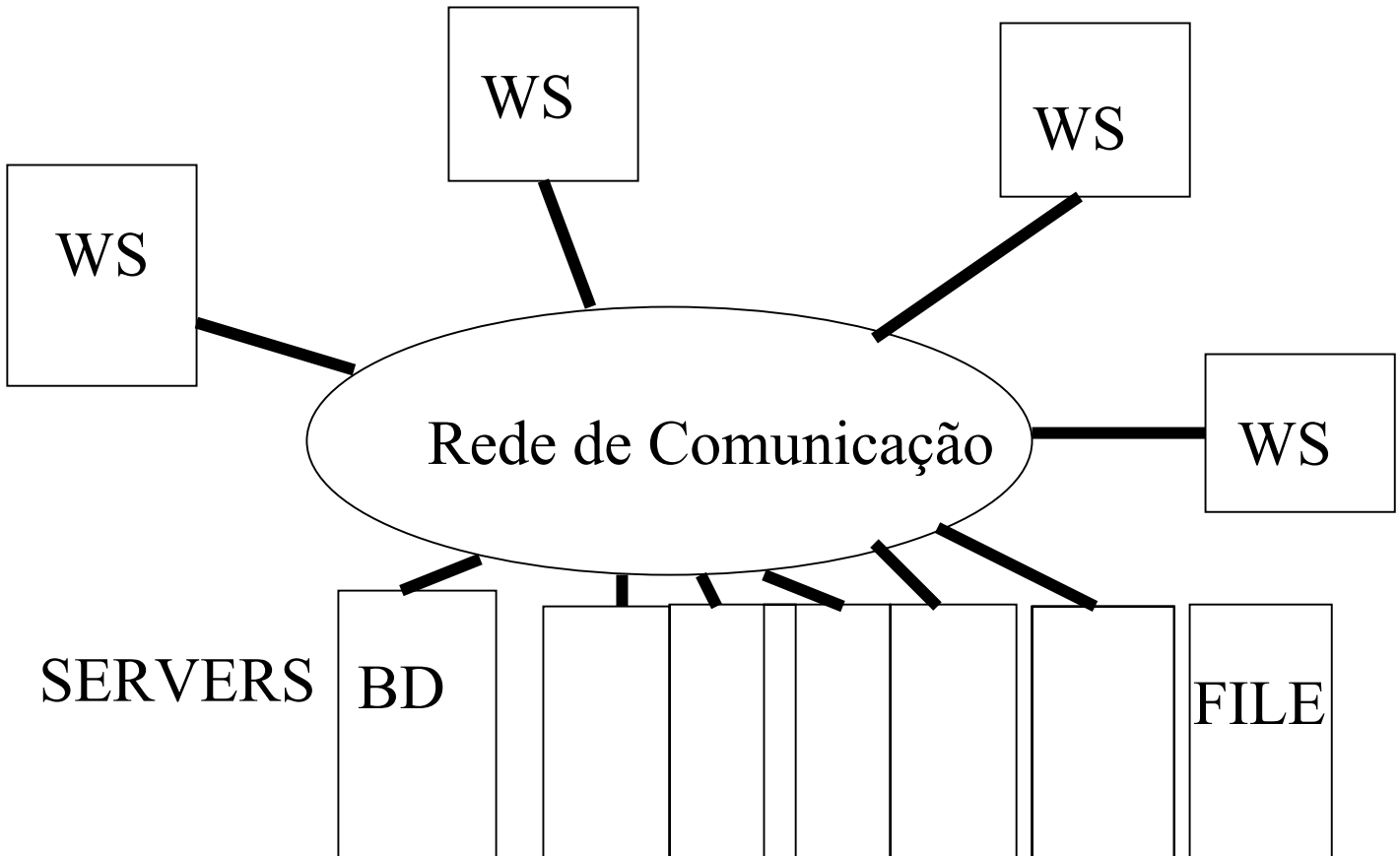
Modelos de Sistemas Distribuídos

(3) Workstation-Server Model



Modelos de Sistemas Distribuídos

(4) Processor-Pool Model



Modelos de Sistemas Distribuídos

(5) Hybrid model : A utilização de um modelo híbrido, entre os modelos workstation-server e processor-model, são muitas vezes construídos como uma reposta a alocação dinâmica de recursos.

Sistemas Distribuídos

Modelos de Programação

Cada modelo de programação especifica a maneira como os diferentes processos da aplicação trocarão informações entre si e quais operações de sincronização estarão disponíveis para coordenar suas atividades.

Os dois modelos mais comumente empregados são :

- *memória compartilhada*
- *troca de mensagem*

Sistemas Distribuídos

Memória Compartilhada: neste modelo os processos compartilham um espaço de endereçamento virtual no qual todos podem ler e escrever.

A troca de informações pode ser feita simplesmente com o uso de variáveis compartilhadas. A vantagem, do ponto de vista do programador, é que não há necessidade de se estabelecer comunicação explícita entre processos, o que permite simplificar o desenvolvimento de algoritmos.

Sistemas Distribuídos

A comunicação entre diferentes linhas de controle de um processo (*threads*) baseia-se neste modelo. É o modelo naturalmente empregado em arquiteturas multiprocessadas. Para arquiteturas com memória distribuída pode-se empregar uma extensão deste modelo, chamado ***Memória Compartilhada Distribuída*** (*Distributed Shared Memory*, DSM).

Sistemas Distribuídos

Este modelo tende a perder eficiência à medida que o número de EP aumente ou à medida que a rede de interconexão apresente taxas de transmissão baixas. Em termo gerais, DSM apresenta baixa escalabilidade , o que impõe sérias restrições para emprego em sistemas globalmente distribuídos;

Sistemas Distribuídos

Troca de Mensagens: neste modelo cada processo possui seu próprio espaço de endereçamento. A comunicação entre eles deve ser feita de forma explícita, o que pode dificultar a tarefa de programação.

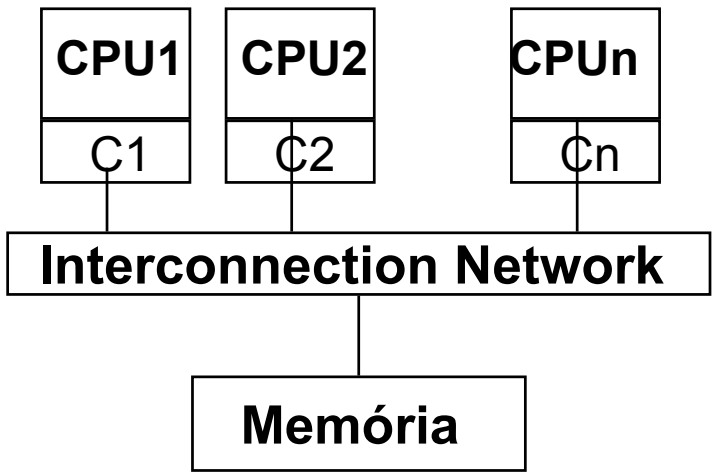
O emprego de bibliotecas especializadas para troca de mensagens (e.g. Message Passing Interface, Parallel Virtual Machine) facilitam o trabalho do programador e permitem o desenvolvimento de programas paralelos muito eficientes. Este modelo adequa-se naturalmente a arquiteturas de memória distribuída.

III - Troca de Mensagem

A comunicação entre processos, estejam estes num mesmo computador ou num ambiente distribuído, pode ser efetuada através da troca de mensagem entre os mesmos. Este paradigma é, geralmente, denominado de *message passing*.

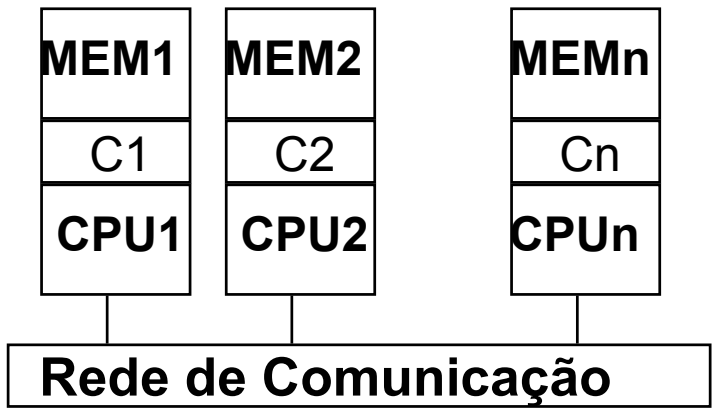
Modelos de Programação Paralela

Arquiteturas
Fortemente-Acopladas



Shared-Memory

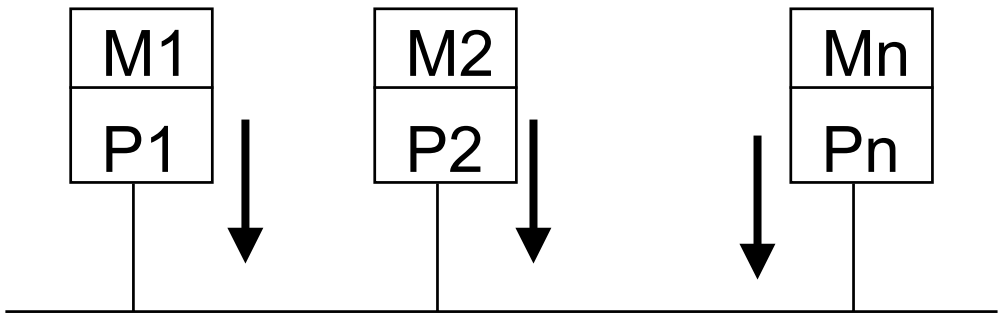
Arquiteturas
Fracamente-Acopladas



Troca de Mensagem

Message Passing

No ambiente distribuído paralelo rodando o paradigma de troca de mensagem, processadores trocam mensagens entre processos durante a execução da aplicação.



Troca de Mensagem

Um sistema de troca de mensagem é um pacote que é, geralmente, empregado numa linguagem de programação (Exemplo: C, C++ e Fortran).

As facilidades desejáveis num ambiente de troca de mensagem são:

- simplicidade - na construção de aplicações usando as primitivas do pacote de troca de mensagem ;
- semântica uniforme - processo local e remoto deve obedecer ao conjunto de primitvas;
- efficiência - uso de piggybacking e minimizar custo de comunicação entre processos;
- confiabilidade - na ocorrência de falhas de nós o sistema deve ser capaz de terminar a ligação entre processos;

Troca de Mensagem

- Correção - observar pontos tais como atomicidade e entrega ordenada;
- flexibilidade - emprego de primitivas tais como *multicast e broadcast* ;
- segurança - autenticação do destinatário pelo remetente e vice-versa, criptografar uma mensagem antes do envio numa rede;
- portabilidade - uso do sistema em sistema operacionais diferentes ;

Troca de Mensagem

Exemplo de pacotes de troca de mensagem são :

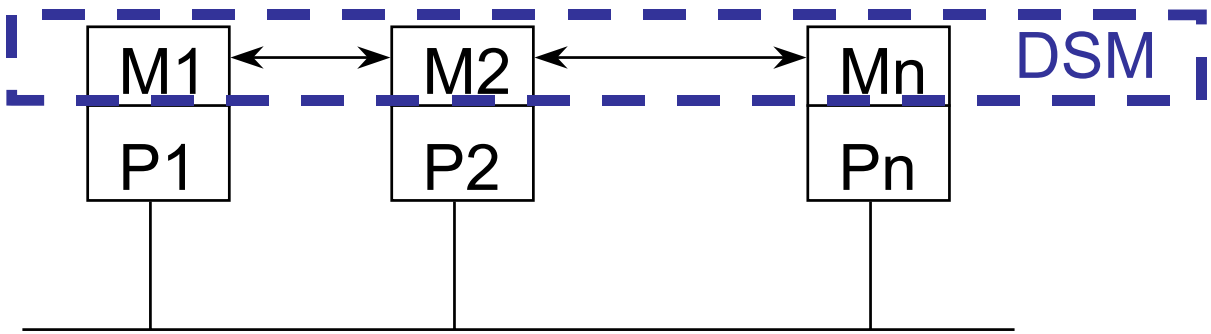
- PARALLEL VIRTUAL MACHINE (PVM)
- P4
- Chimp
- Express
- PARMACS

Troca de Mensagem

- IBM-EUI
- ZIPCODE
- MESSAGE PASSING INTERFACE (MPI)

IV – Distributed Shared Memory

Implementação da abstração de memória compartilhada num ambiente distribuído.



V - Gerência de Recursos

Os sistemas distribuídos são caracterizados pela abundância de recursos e a transparência do sistema. Processos podem migrar entre máquina durante sua execução, dispositivos podem se tornar disponíveis, a rede pode estar *pesada* entre outras várias situações. Assim, a gerência de alocação de recursos é um tópico fundamental para o sucesso (ou fracasso de um sistema distribuído).

Inúmeras técnicas foram propostas para a alocação de processos em sistemas distribuídos. Esta podem ser classificadas como :

- *Alocação de tarefas* : todo processo submetido por um usuário é visto como uma coleção de tarefas e estas são escalonadas para os nós que podem executar tais tarefas, assim melhorando o desempenho.

• *Balanceamento de Carga* : todos os processos submetidos ao sistema pelos usuários são distribuídos entre os computadores do sistema equalizando a carga do sistema como um todo.

•*Compartilhamento de Carga* : é efetuado um esforço para que nenhuma máquina do sistema fique ociosa quando houverem processos para serem executados.