

# Introdução à Computação de Alto Desempenho

COPPE/UFRJ

Alvaro Coutinho & Albino Aveleda  
{alvaro,bino}@nacad.ufrj.br



# Horários

- 31/07 : 13:00 às 16:30
- 02/08: 13:00 às 16:30
- 04/08: 13:00 às 16:00



# Ementa

---

- Introdução
- Tópicos Básicos em arquitetura de Computadores
- Arquitetura de processadores e memória
- Medidas de desempenho e análise
- Clusters
- Comentários Finais



# Ementa

---

- **Introdução**
- Tópicos Básicos em arquitetura de Computadores
- Arquitetura de processadores e memória
- Medidas de desempenho e análise
- Clusters
- Comentários Finais

# Web-related Material

- Computational Science Education Project



<http://www.phy.ornl.gov/csep/>

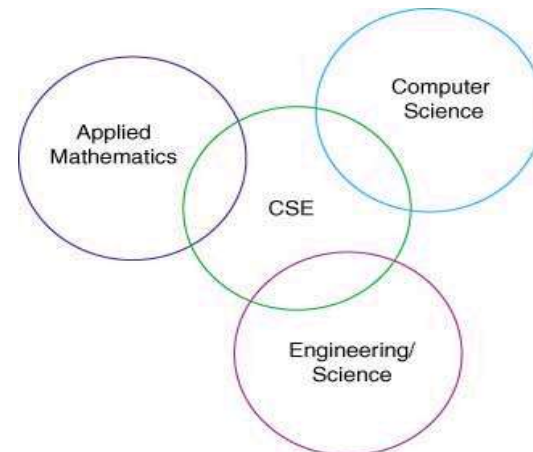
- IEEE Computing in Science and Engineering



IEEE: <http://www.computer.org/cise>

- SIAM

[www.siam.org](http://www.siam.org)



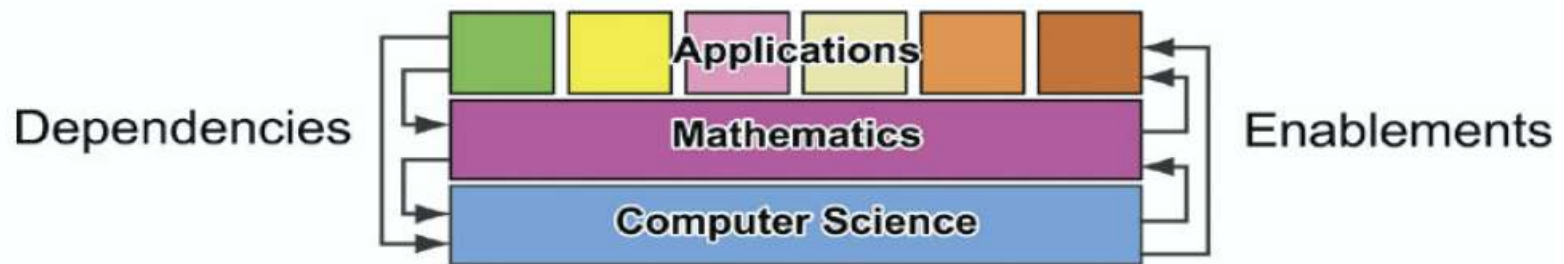
# Referências

- Designing and Building Parallel Programs
  - Ian Foster, 1995 , <http://www-unix.mcs.anl.gov/dbpp>
- High Performance Cluster Computing (Volume 1): Systems and Architecture
  - Rajkumar Buyya, Prentice Hall, 1999
- High Performance Cluster Computing (Volume 2): Programming and Applications
  - Rajkumar Buyya, Prentice Hall, 1999
- How to Build a Beowulf – A Guide to the Implementation and Application of PC Clusters
  - Thomas Sterling et al, The MIT Press, 1999

# Referências

- Beowulf Cluster Computing with Linux
  - William Gropp, Ewing Lusk and Thomas Sterling, MIT Press, 2003
- IA-32 Intel Architecture Software Developer's Manual  
Volume 1: Basic Architecture
  - Intel Corporation, 2004 – <http://www.intel.com>
- Parallel Programing with MPI
  - Peter Pacheco, Morgan Kaufmnn Publishers, 1997
- Using MPI: Portable Parallel Programing with The  
Message Passing Interface
  - William Gropp, Ewing Lusk and Anthony Skjellum, MIT Press, 1999

# Layered Structure of CE&S



From: A SCIENCE-BASED CASE FOR LARGE-SCALE SIMULATION, DOE, 2003



# Demanda por poder computacional

- Aplicações Comerciais:

- ☐ bancos, seguradoras, sistemas industriais
- ☐ acesso intensivo à dados
- ☐ processamento de transações
- ☐ alta disponibilidade

- Aplicações Científicas:

- ☐ universidades, centros de pesquisa, indústria de ponta
- ☐ computação intensiva
- ☐ alto desempenho

# Demanda por poder computacional

- Operações necessárias a execução de aplicações
  - $n$  equações lineares  $\Rightarrow 2n^3/3$  operações de ponto flutuante
- Desempenho do processador:
  - 2GHz  $\Rightarrow 2 \times 10^9$  operações de ponto flutuante por segundo
  - Taxa de acesso à memória
- Desempenho de todos os componentes do sistema



# Aplicações Científicas


- Genoma Humano
- Turbulência dos Fluidos
- Dinâmicas de Veículos
- Circulação de Oceanos
- Dinâmica de Fluidos Viscosos
- Modelagem de Supercondutores
- Cromodinâmica Quântica
- Visão por Computador
- Previsão do Tempo
- Exploração de Petróleo
- Engenharia Ambiental



# Ementa

---

- Introdução
- Tópicos Básicos em arquitetura de Computadores
- Arquitetura de processadores e memória
- Medidas de desempenho e análise
- Clusters
- Comentários Finais

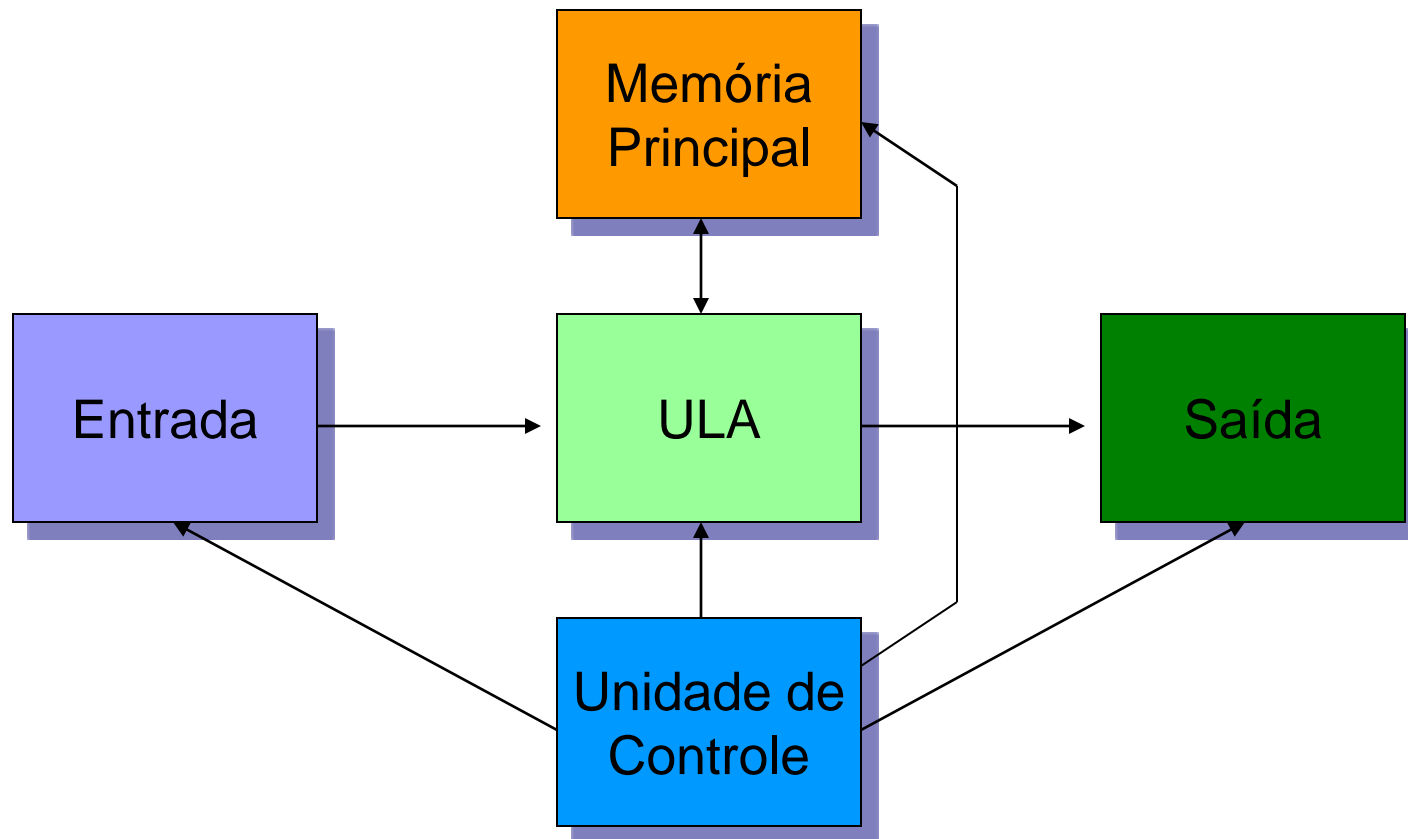


# Processadores

## Introdução

- As arquiteturas dos processadores têm evoluído ao longo dos anos, e junto com ela o conceito de arquitetura avançada tem se modificado
- Nos anos 60 os computadores ditos de grande porte (IBM, CDC) eram considerados exemplos de alto desempenho
- Depois surgiram os supercomputadores, arquiteturas específicas para a realização de cálculos vetoriais (exemplo, Cray)

# Modelo de Von Neumann

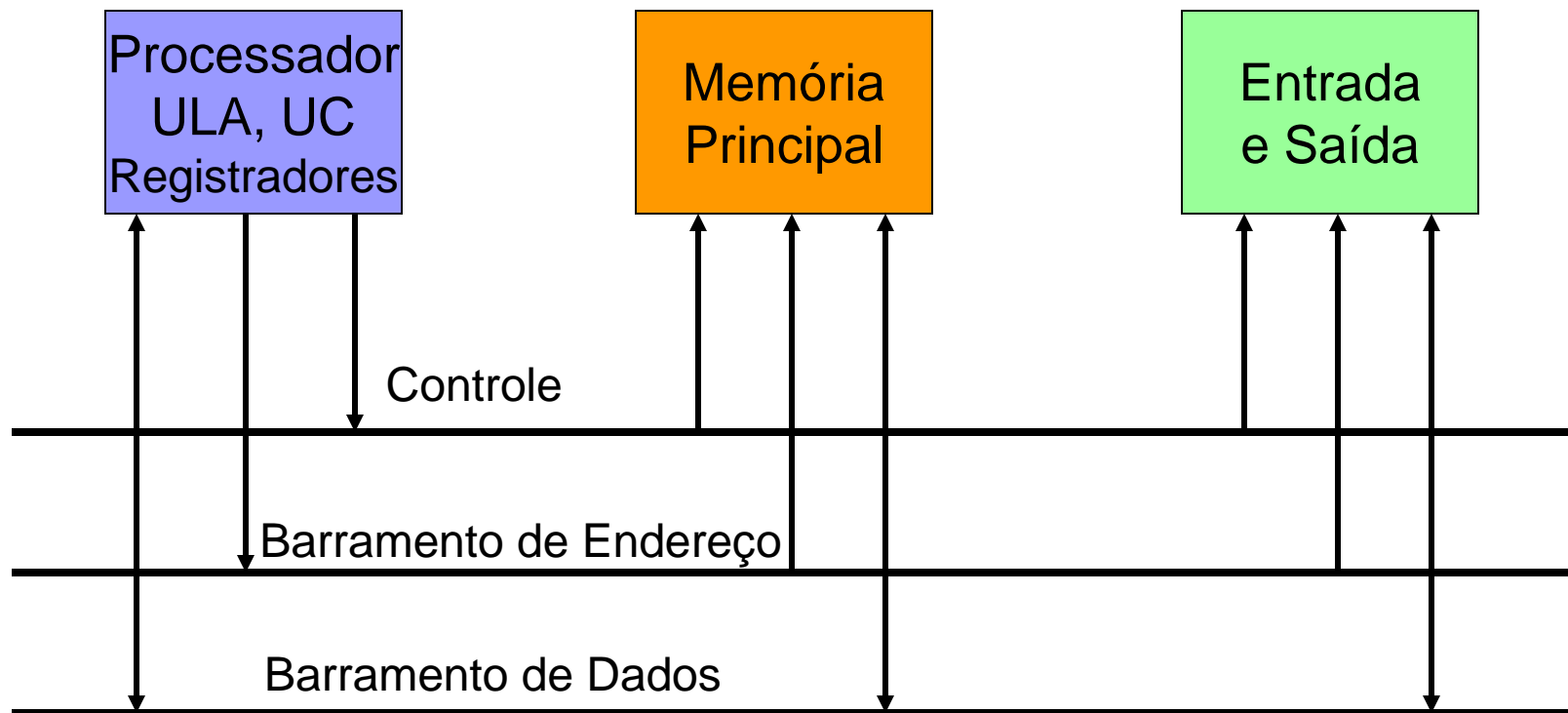




# Modelo de Von Neumann

- Todo computador é formado por 5 partes básicas:
  - Memória
  - Unidade lógica e aritmética (ULA)
  - Unidade de controle (UC)
  - Periféricos de entrada e saída
- ULA: realiza operações aritméticas e lógicas com os dados
- UC: controla o tipo de operação a ser executado
- Memória: armazena os dados e instruções que vão ser utilizados pela ULA e UC
- Os dados a serem operandos são lidos dos dispositivos de entrada para a memória
- Os resultados são enviados para a memória e daí para os dispositivos de saída

# Modelo de barramento de sistema





# The main components of a computer system are:

- *Processors*
- *Memory*
- *Communications Channels*

These components of a computer architecture are often summarized in terms of a PMS diagram.

(P = "processors", M = "memory", S = "switches".)

# Processors

## Fetch-Decode-Execute Cycle

The essential task of computer processors is to perform a *Fetch-Decode-Execute* cycle:

1. In the *fetch* cycle the processor gets an instruction from memory; the address of the instruction is contained in an internal register called the **Program Counter** or PC.
2. While the instruction is being fetched from memory, the PC is incremented by one. Thus, in the next Fetch-Decode-Execute cycle the instruction will be fetched from the next sequential location in memory (unless the PC is changed by some other instruction in the interim. )
3. In the *decode phase* of the cycle, the processor stores the information fetched from memory in an internal register called the **Instruction Register** or IR.
4. In the *execution phase* of the cycle, the processor carries out the instruction stored in the IR.

# Classification of Processor Instructions

Instructions for the processor may be classified into three major types:

1. **Arithmetic/Logic** instructions apply primitive functions to one or two arguments; an example is the addition of two numbers.
2. **Data Transfer** instructions move data from one location to another, for example, from an internal processor register to a location in the main memory.
3. **Control** instructions modify the order in which instruction are executed, for example, in loops or logical decisions.



# Clock Cycles

---

Operations within a processor are controlled by an external clock (a circuit generating a square wave of fixed period).

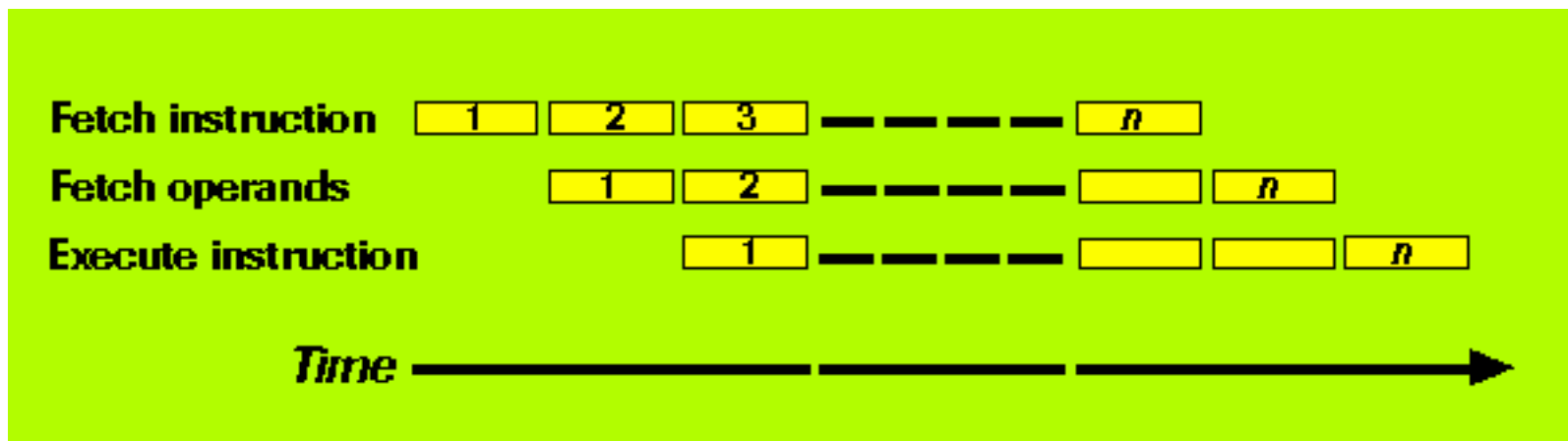
The quantum unit of time is a *clock cycle*. The clock frequency (e.g., 3GHz would be a common clock frequency for a modern workstation) is one measure of how fast a computer is, but the length of time to carry out an operation depends not only on how fast the processor cycles, but how many cycles are required to perform a given operation.

This is a rather involved function of topics to be discussed below.

# Pipeline

- É uma técnica de implementação de processadores que permite a sobreposição temporal das diversas fases de execução
- Aumenta o número de instruções executadas simultaneamente e taxa de instruções iniciadas e terminadas por unidade de tempo
- O pipeline não reduz o tempo gasto para completar cada instrução individualmente
- Melhora o *throughput* de todo trabalho
- Várias tarefas executam simultaneamente usando recursos diferentes

# Pipeline de Instruções



# Características dos pipelines de instrução

- O tempo do ciclo do relógio do processador deve ser igual ou maior que o tempo de execução do estágio mais lento do “pipeline”
- O pipeline deve ser mantido sempre “cheio” para atingir o desempenho máximo
- Algumas instruções contudo podem ter o seu tempo de execução aumentado, pois atravessam estágios em que não realizam nenhuma operação útil

# Características dos pipelines de instrução

- O tempo gasto no processamento de  $M$  instruções em um pipeline com  $K$  estágios e ciclos de máquina igual a  $t$  é dado por:

$$T = [K + (M - 1)] * t$$

- Se um programa tem 10.001 instruções. Quanto tempo leva para ser executado em um processador com pipeline de 5 estágios e relógio de 100 ns?

$$T = (5 + (10.000)) * 100 * 10^{-9} \approx 1 \text{ ms} \quad (\text{com pipeline})$$

$$T = 500 \text{ ns} * 10.001 \approx 5 \text{ ms} \quad (\text{sem pipeline})$$



# Problemas no Uso de Pipelines

- Estágios podem ter tempos de execução diferentes
- O sistema de memória é incapaz de manter o fluxo de instruções no pipeline
  - Solução: O uso de memória cache com alta taxa de acerto e tempo de acesso compatível com o tempo de ciclo do pipeline
- Dependências ou conflitos
  - Conflitos Estruturais
    - Pode haver acessos simultâneos à memória feitos por 2 ou mais estágios
  - Dependências de Dados
    - As instruções dependem de resultados de instruções anteriores, ainda não completadas
  - Dependências de Controle
    - A próxima instrução não está no endereço subsequente ao da instrução anterior
- Tratamento de Exceções

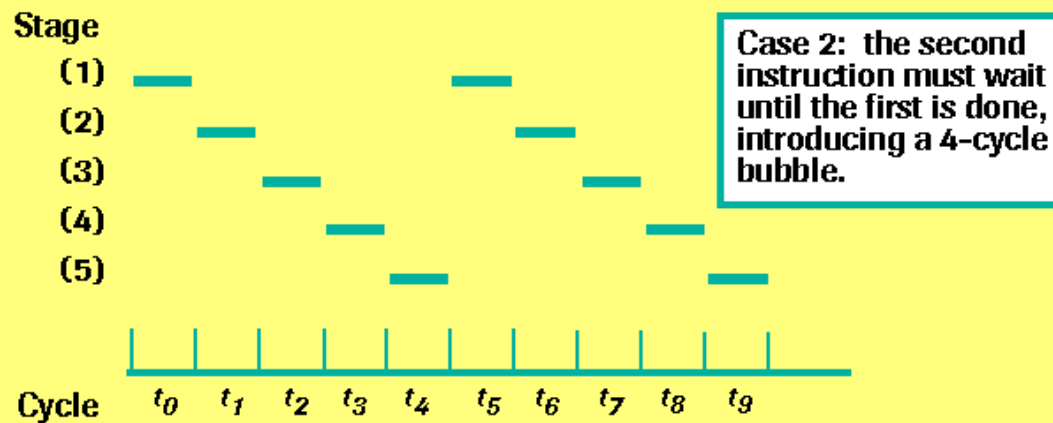
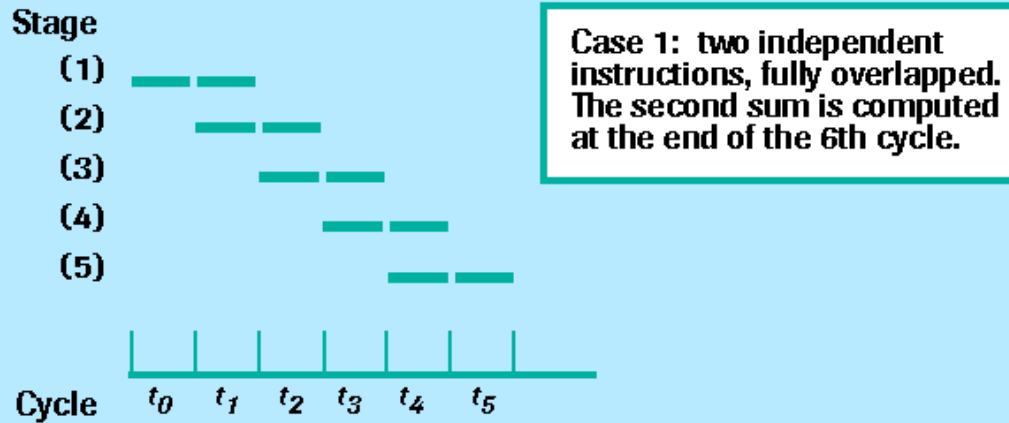


Figure 5 Gantt Charts for Pipelined Floating Point Add

# Arquiteturas CISC

## ■ Características:

- Instruções complexas demandando um número grande e variável de ciclos de máquina para execução
- Uso de diversos modos de endereçamento de operandos
- Instruções com formato muito variável
- Cada fase do processamento da instrução pode ter duração variável em função da complexidade

# Arquiteturas CISC

## ■ Consequências:

- Implementação com uso de pipeline é difícil
- Taxa média de execução das instruções por ciclo tende a ser bastante inferior a 1
- Códigos compactos podem ser gerados pelos compiladores



# Arquiteturas RISC

## ■ Características:

- Instruções mais simples demandando um número fixo de ciclos de máquina para sua execução
- Uso de poucos modos de endereçamento de operandos
- Cada fase de processamento da instrução tem a duração fixa igual a um ciclo de máquina

# Arquiteturas RISC

## ■ Consequências:

- ☐ Implementadas com o uso de pipeline
- ☐ A taxa média de execução de instruções por ciclo de máquina é próxima de 1
- ☐ Processo de compilação é complexo e requer cuidados especiais para otimização do desempenho do código gerado



# Arquiteturas Superescalares

- Possuem mais de um pipeline em paralelo em sua arquitetura
- O barramento tem largura suficiente para buscar mais de uma instrução por ciclo
- Lógica na decodificação das instruções para determinar quais instruções são independentes e podem ser executadas ao mesmo tempo
- O uso de caches separadas para dados e instruções é uma solução para permitir leitura e escrita simultânea



# Ementa

---

- Introdução
- Tópicos Básicos em arquitetura de Computadores
- **Arquitetura de processadores e memória**
- Medidas de desempenho e análise
- Clusters
- Comentários Finais



# Processadores vetoriais

- São arquiteturas “pipelined”, ou seja, uma única instrução opera sobre vários dados, no caso, um vetor;
- Um processador vetorial possui instruções para operar em vetores
- Uma operação vetorial típica:
  - $C = A * B$
  - Onde A, B e C são vetores

# Processadores vetoriais

$$C=A*B$$

Lê os elementos dos vetores	1	2	3	4	5	6	7	8
Soma Expoentes		1	2	3	4	5	6	7
Multiplica mantissas			1	2	3	4	5	6
Normaliza resultado				1	2	3	4	5
Armazena result. em C					1	2	3	4



# Problemas para vetorização

- Tamanho dos vetores
- Stride:
  - Salto entre os endereços de dois elementos consecutivos é chamado de stride
  - Operações de multiplicação de matrizes, o acesso não será feito de forma sequencial, já que elas são armazenadas na memória ou por linha ou por coluna
- Existência de desvios dentro dos “loops” (if)

# Máquinas Vetoriais



Cray X1E Supercomputer

Cray J90/SV1





# Tecnologia HyperThreading

- Desenvolvida pela Intel
- Aumento de desempenho de até 30% (de acordo com a Intel) e aumento menos de 5% no tamanho total do chip em relação ao consumo
- Simula em um processador físico dois processadores lógicos
- Em termos de software, significa que o sistema operacional pode enviar tarefas para os processadores lógicos como se estivesse enviando para processadores físicos em um sistema de multiprocessamento



# Tecnologia HyperThreading

- Cada processador lógico recebe seu próprio controlador de interrupção programável (APIC) e conjunto de registradores
- Os outros recursos do processador físico, tais como, cache de memória, unidade de execução, unidade lógica e aritmética, unidade de ponto flutuante e barramentos, são compartilhados entre os processadores lógicos

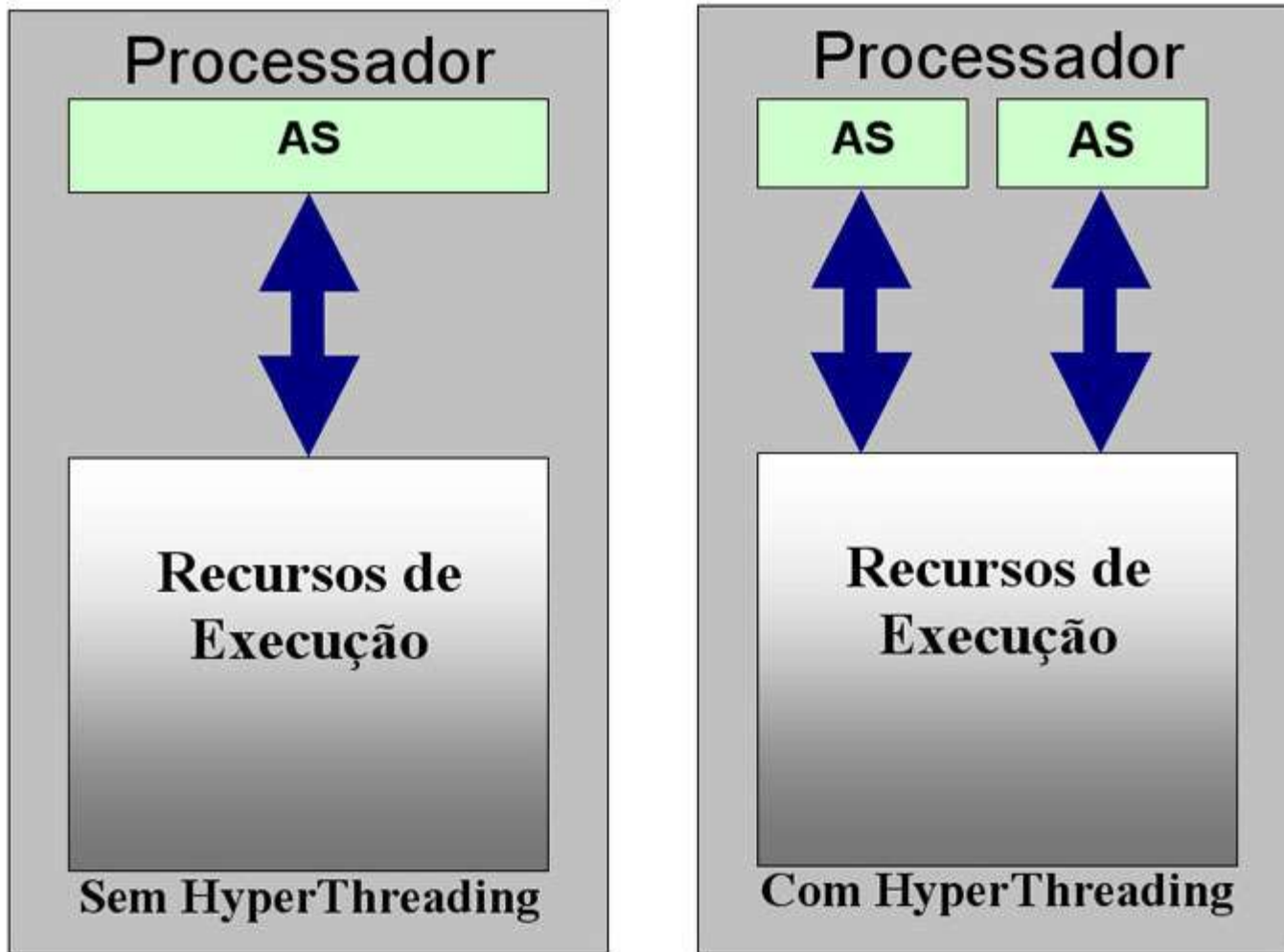


# HyperThreading

## ■ Resumo:

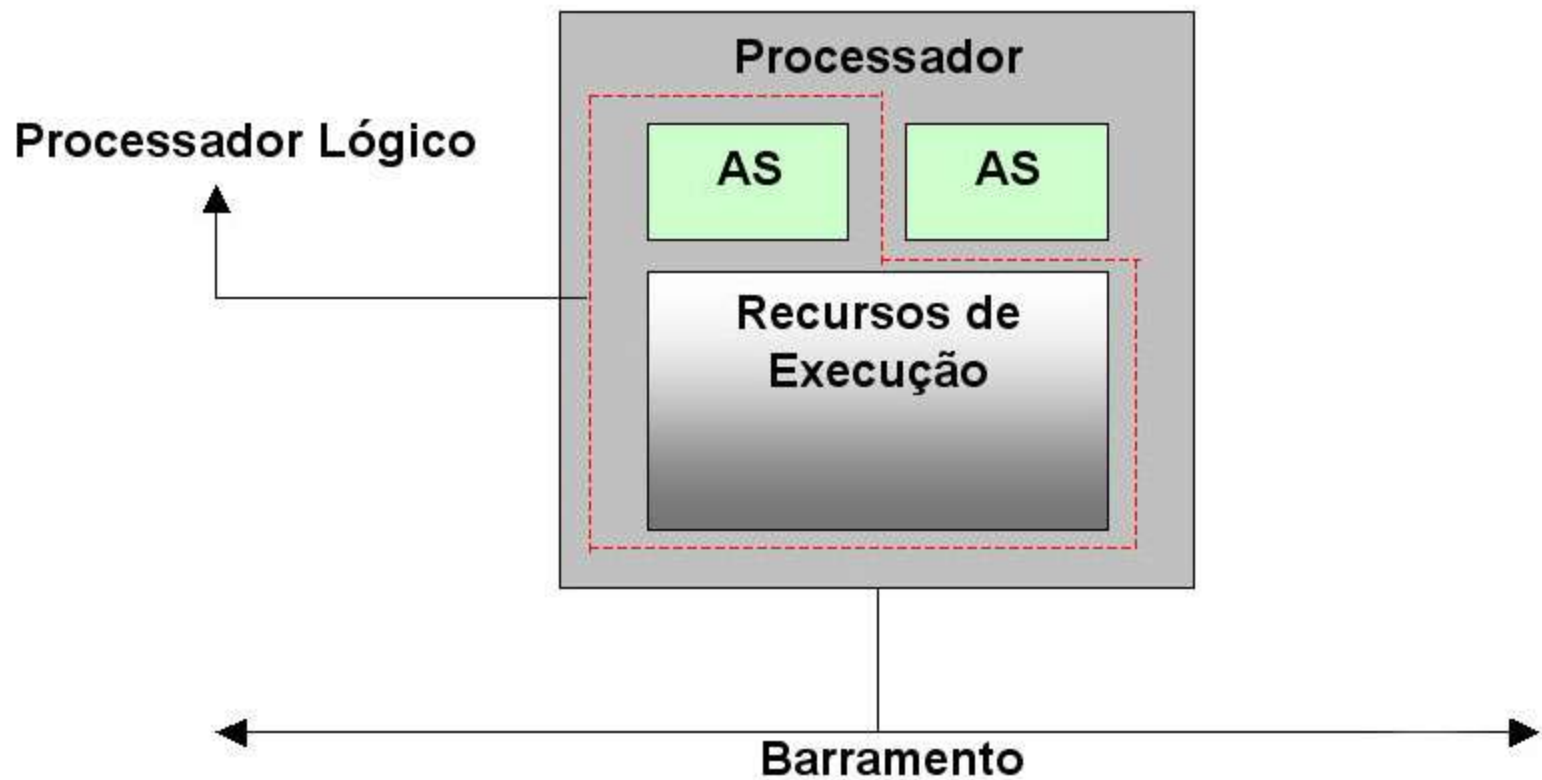
- Execução de mais de um fluxo de código concorrentemente
- Processadores lógicos com estado de execução independentes (registradores e controladores de interrupção)
- Sistema Operacional ou aplicação multi-thread
- Aplicações single-thread em ambientes multi-task

# Tecnologia HyperThreading

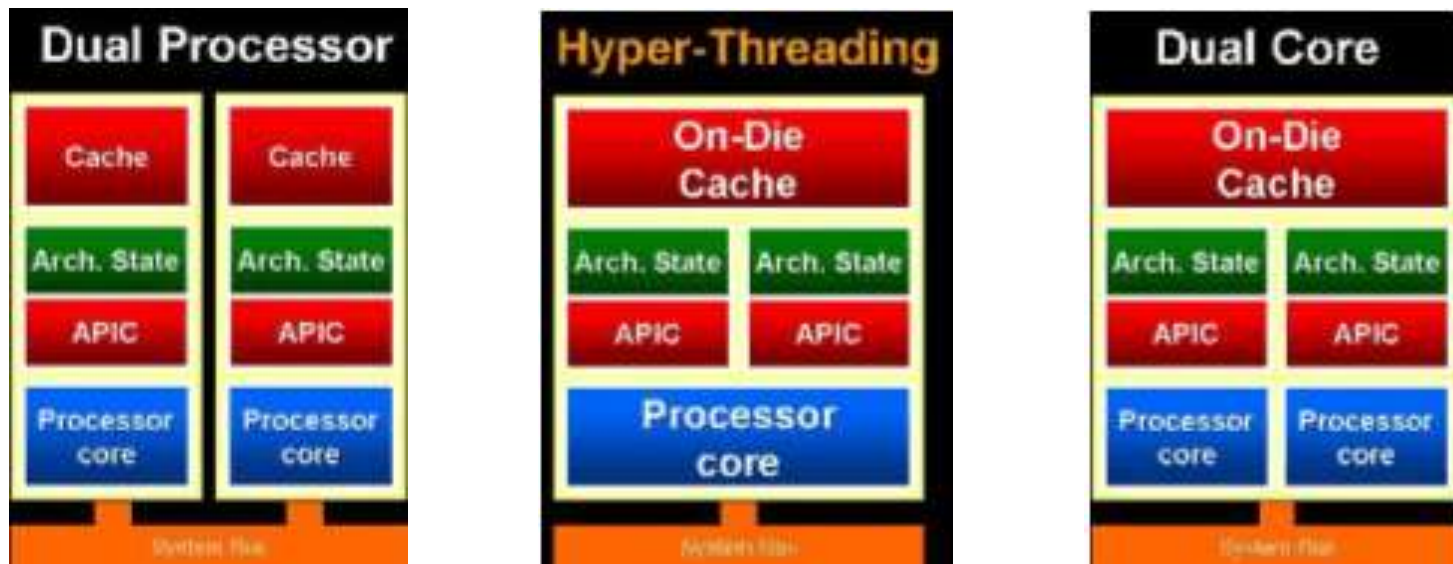




# Tecnologia HyperThreading



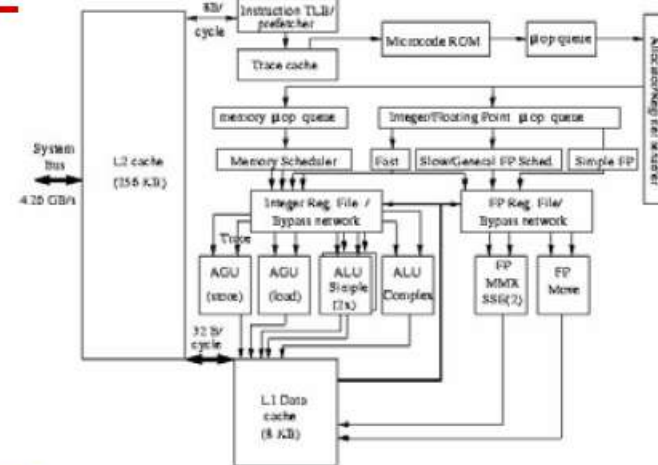
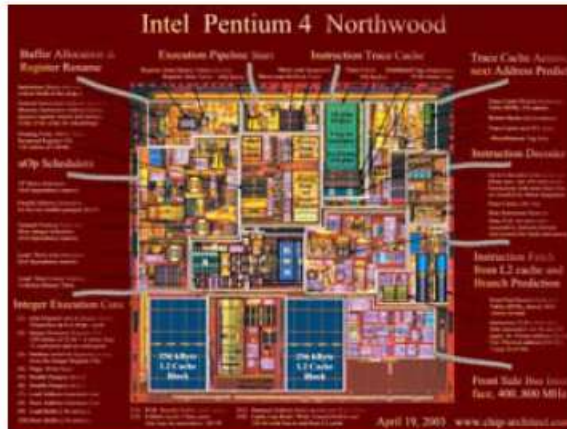
# Tecnologias Intel



- Acesso à memória é baseado no chipset
  - DIMM, DDR, DDR2, etc



# Pentium 4 IA32



- ◆ Processor of choice for clusters
- ◆ 1 flop/cycle
- ◆ Streaming SIMD Extensions 2 (SSE2): 2 Flops/cycle
- ◆ Intel Xeon 3.2 GHz 400/533 MHz bus, 64 bit wide(3.2/4.2 GB/s)
  - Linpack Numbers: (theoretical peak 6.4 Gflop/s)
    - 100: 1.7 Gflop/s
    - 1000: 3.1 Gflop/s
- ◆ Coming Soon: "Pentium 4 EM64T"
  - 800 MHz bus 64 bit wide
  - 3.6 GHz, 2MB L2 Cache
  - Peak 7.2 Gflop/s using SSE2

8



# Tecnologia AMD64 (Diferencial)

## ■ Arquitetura de Conexão Direta

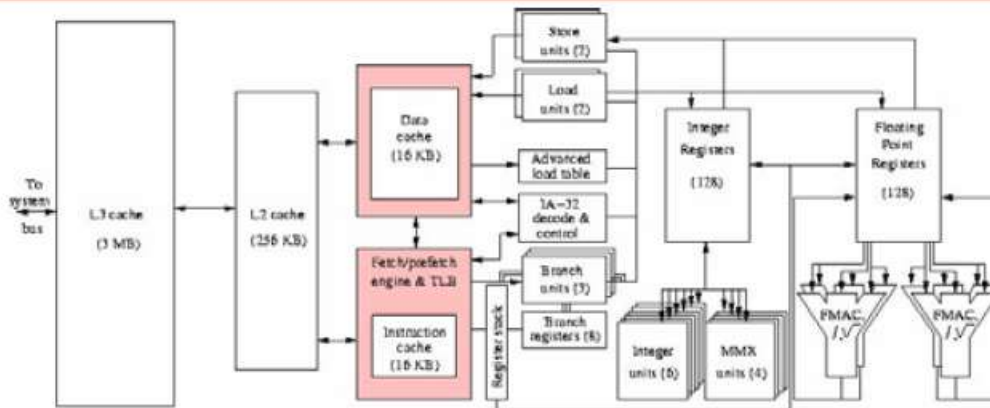
- Ajuda a reduzir os desafios e os gargalos das arquiteturas de sistema
- Conecta diretamente a memória à CPU, otimizando seu desempenho
- Conecta diretamente o subsistema de I/O à CPU, proporcionando uma performance mais equilibrada
- Conecta diretamente os processadores uns aos outros, facilitando um processamento simétrico mais linear

# Itanium2 (IA-64)

- Arquitetura de 64 bits, capaz de executar 6 operações/ciclo de máquina
- Itanium 2 x Arquitetura Risc (de acordo com a Intel)
  - Menor custo
  - Desempenho e confiabilidade para aplicações de missão crítica
- Não é compatível com as arquiteturas x86 e x64
- Tecnologia com alto grau de paralelismo (EPIC: Explicitly Parallel Instruction Computing)



# Itanium 2



- ◆ Floating point bypass for level 1 cache
- ◆ Bus is 128 bits wide and operates at 400 MHz, for 6.4 GB/s
- ◆ 4 flops/cycle
- ◆ 1.5 GHz Itanium 2
  - Linpack Numbers: (theoretical peak 6 Gflop/s)
    - 100: 1.7 Gflop/s
    - 1000: 5.4 Gflop/s

# Sun Fire com Tecnologia CoolThreads

- Métrica SWaP (**S**pace, **W**atts and **P**erformance)
- $SWaP = \text{Desempenho} / (\text{Espaço} \times \text{Consumo de Energia})$ 
  - Desempenho: usando os referenciais que são usados na indústria
  - Espaço: medindo a altura dos servidores em unidades de rack (RUs)
  - Energia: watts consumido pelo sistema



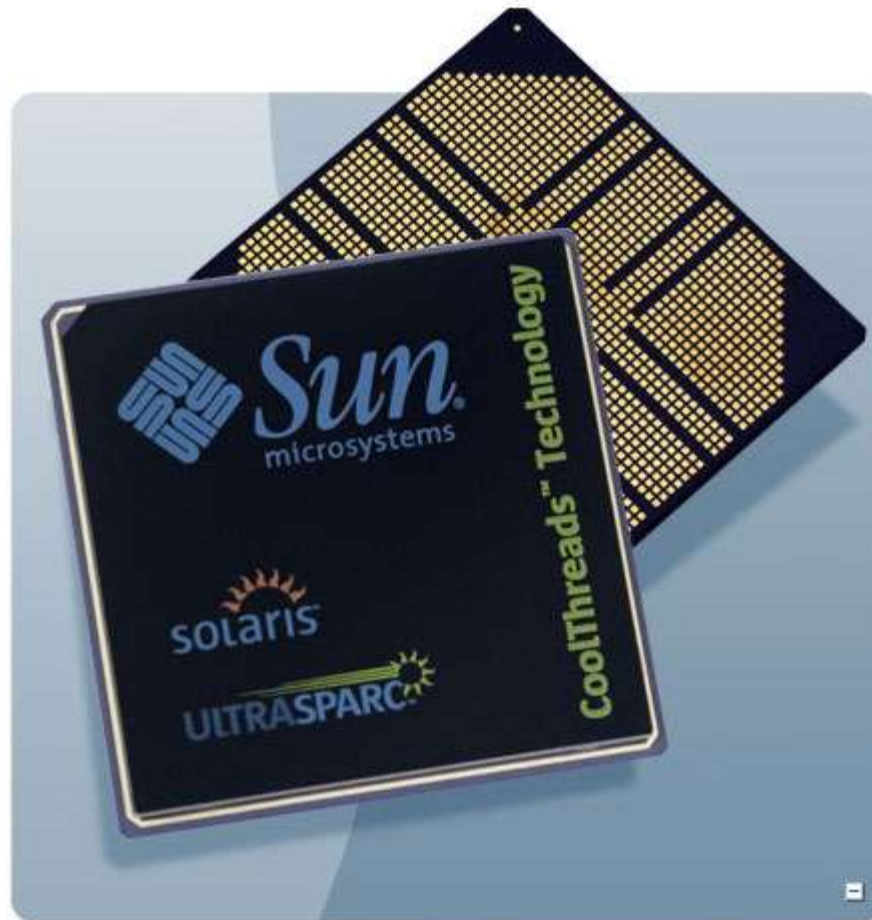
# Sun Fire com Tecnologia CoolThreads

## ■ De acordo com a Sun:

- Desempenho até 3x mais rápido do que a tecnologia atual de servidor
- Eficiência em termos de energia e aquecimento que corta os custos operacionais em 75%
- Projeto compacto de unidades de apenas 1 ou 2 gabinetes
- Compatibilidade com os aplicativos existentes
- Os mais altos níveis de escalonamento e disponibilidade



# Processador UltraSparc



# Computer Memory

## Memory Classifications

Computers have hierarchies of memories that may be classified according to

- Function
- Capacity
- Response Times

## Memory Function

"Reads" transfer information from the memory; "Writes" transfer information to the memory:

- ***Random Access Memory*** (RAM) performs both reads and writes.
- ***Read-Only Memory*** (ROM) contains information stored at the time of manufacture that can only be read.

# Memory Capacity

bit = Menor unidade de memória (valor de 0 ou 1); normalmente abreviado como "b"

byte = 8 bits; normalmente abreviado como "B"

Prefixos comuns:

k=kilo=1000

M=mega= $10^6$

G=giga= $10^9$

T= tera= $10^{12}$

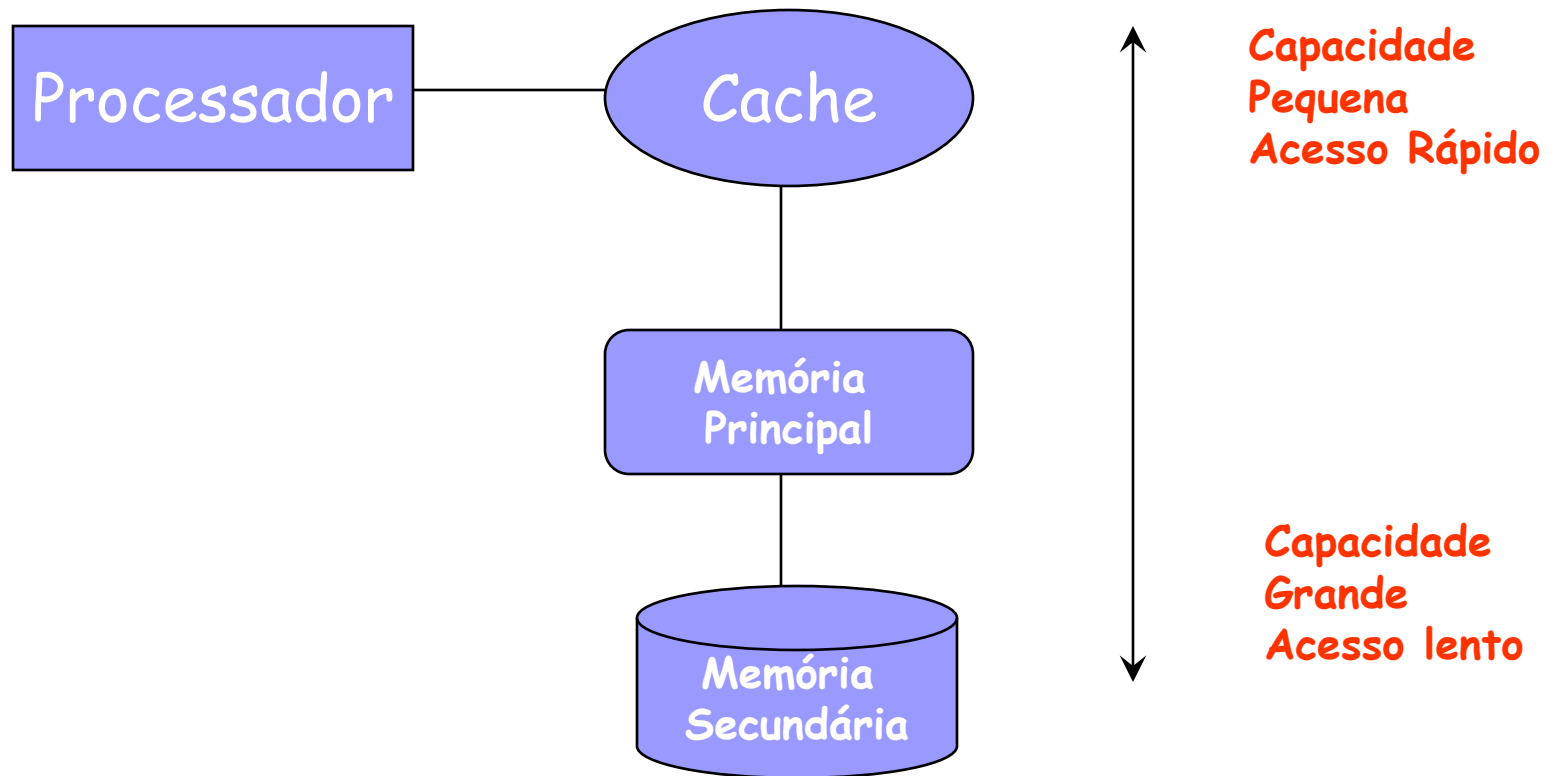
## Memory Response

Memory response is characterized by two different measures:

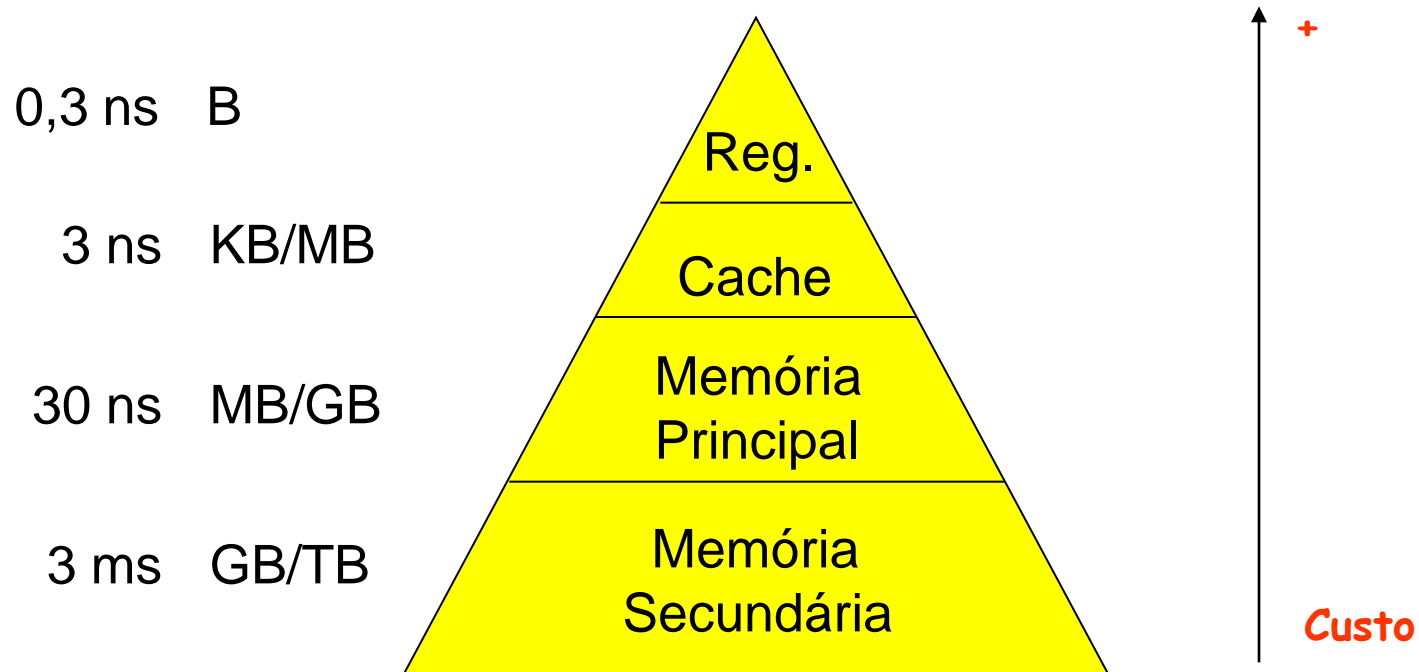
- **Access Time** (also termed *response time* or *latency*) defines how quickly the memory can respond to a read or write request.
- **Memory Cycle Time** refers to the minimum period between two successive requests of the memory.

Access times vary from about 80 ns [ns = nanosecond =  $10^{-9}$  seconds] for chips in small personal computers to about 10 ns or less for the fastest chips in caches and buffers (see below). For various reasons, the memory cycle time is more than the speed of the memory chips (i.e., the length of time between successive requests is more than the 80 ns speed of the chips in a small personal computer).

# Hierarquia de Memória



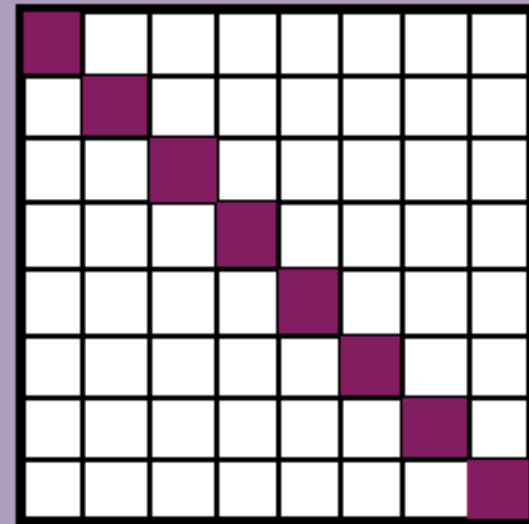
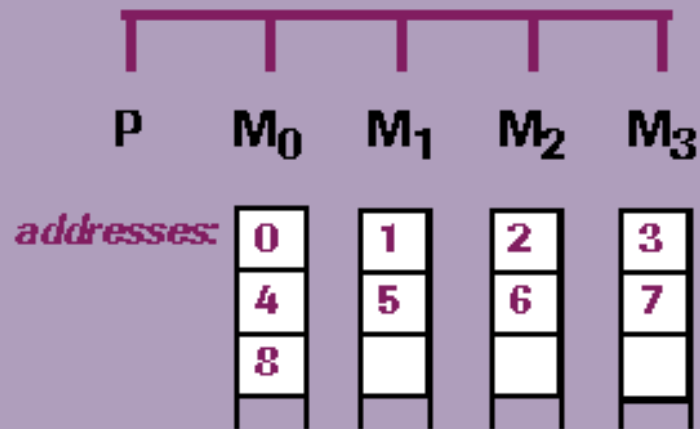
# Hierarquia de Memória



# Conceitos de Localidades

- O funcionamento da hierarquia de memória está fundamentado em duas características encontradas nos programas
  - Localidade Temporal
    - Posições de memória, uma vez referenciadas (lidas ou escritas), tendem a ser referenciadas novamente dentro de um curto espaço de tempo
  - Localidade Espacial
    - Se uma posição é referenciada, posições de memória cujos endereços sejam próximos da primeira tendem a ser logo referenciados

## Interleaved Memory



(a) PMS diagram and memory layout  
(cell  $i$  is in bank  $i \bmod 4$ ).

(b) Elements fetched from an  
8 x 8 array when the stride is 9.

Figure 6 Interleaved Memory



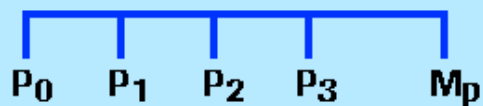
# Funcionamento da Cache

- Durante a busca da palavra que está faltando na cache, é trazido um bloco (ou linha) inteiro da memória principal, ao invés de apenas uma palavra
- O objetivo é minimizar a taxa de falhas nos próximos acessos, seguindo o princípio da localidade espacial
- O tamanho de um bloco é um parâmetro de projeto na memórias caches, tamanhos usuais são 32, 64 e 128 bytes

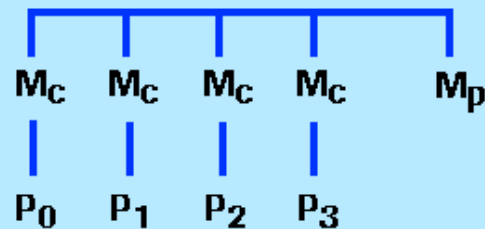


## Bus Contention

(a)



(b)



(a) Four processors sharing a bus.

(b) Local caches reduce bus contention.

(c) Performance curve. Caches move the "knee" to the right, but performance still levels off at around 30 processors.

(c)

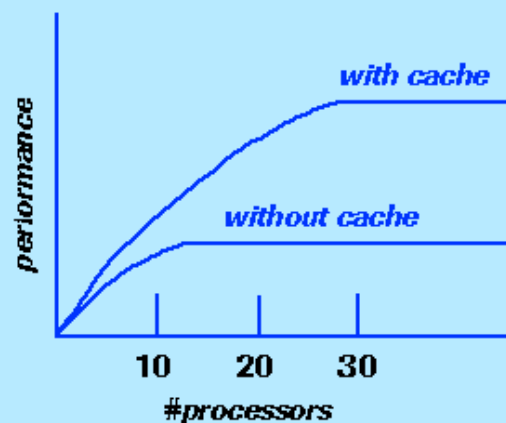



Figure 7 Single Bus Multiprocessor

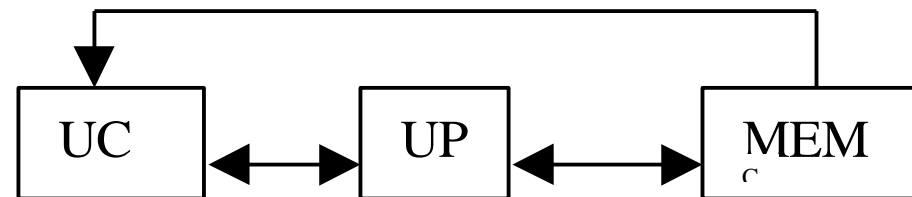


# Taxonomia das Arquiteturas Paralelas

- Classificação de Flynn (1966)
  - Single Instruction Stream (SI)
  - Multiple Instruction Streams (MI)
  - Single Data Stream (SD)
  - Multiple Data Streams (MD)
- Categorias de Arquiteturas
  - SISD (processadores convencionais)
  - SIMD (processadores vetoriais)
  - MIMD (multiprocessadores)

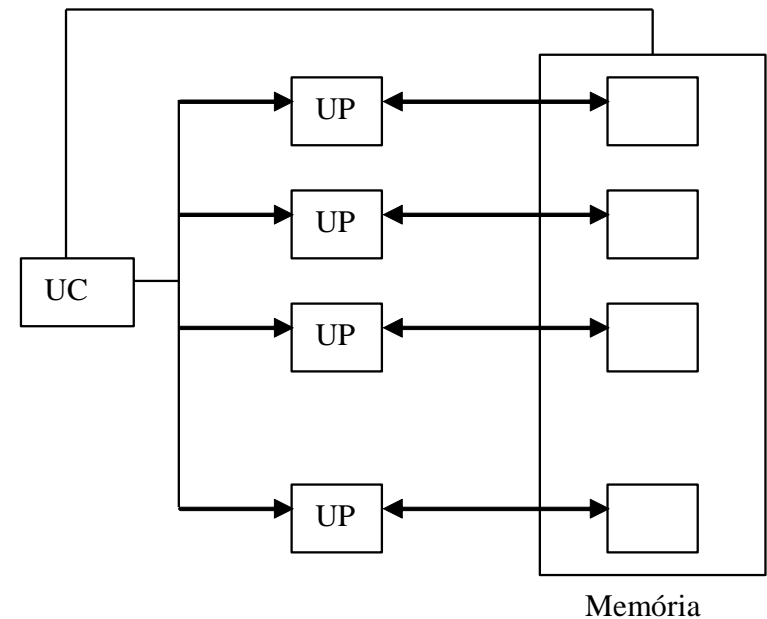
# SISD

- máquinas convencionais com uma CPU (Uma unidade de processamento com uma unidade de controle).



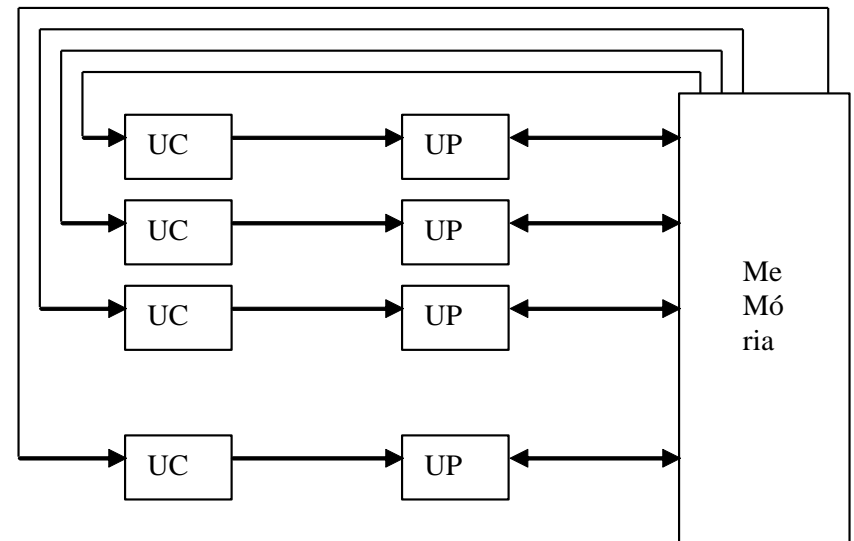
# SIMD

- corresponde ao caso das arquiteturas vetoriais onde a mesma operação é executada sobre múltiplos operandos



# MIMD

- é o caso dos multiprocessadores, onde várias instruções podem ser executadas ao mesmo tempo em unidades de processamento diferentes controladas por unidades de controle independentes (uma para cada unidade de processamento)



# Taxonomia das Arquiteturas Paralelas

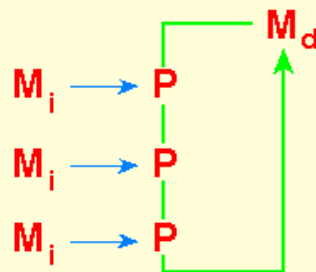
*Single  
Instruction  
Stream*

*Single Data  
Stream*



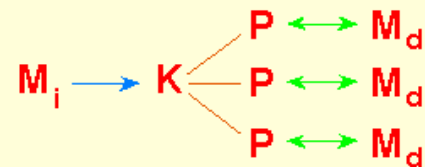
*SISD: A single processor fetches instructions and performs all data processing operations*

*Multiple  
Instruction  
Streams*

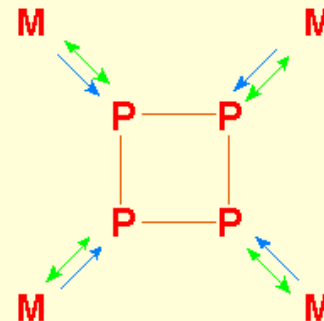


*MISD: A single data stream is operated on by several processors, each with an instruction stream from its own instruction memory.*

*Multiple  
Data Streams*

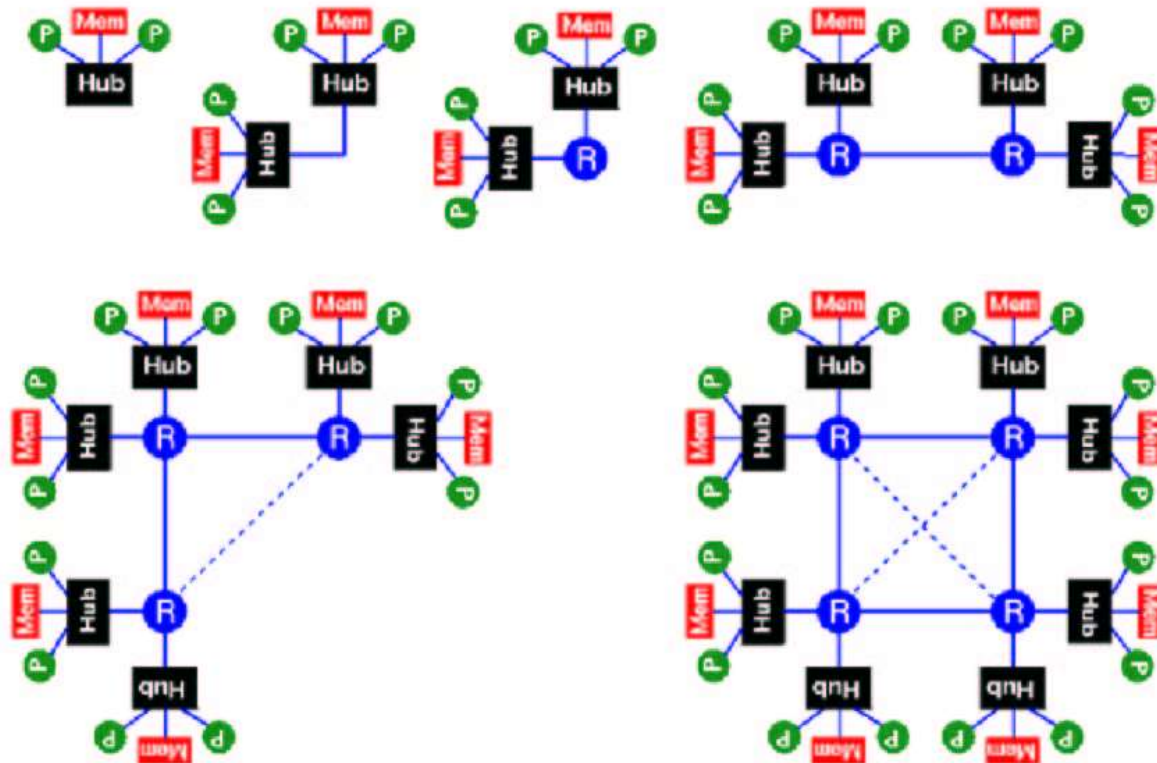


*SIMD: One instruction processor K fetches instructions, broadcasts orders to processing elements P. Typically each P has its own data memory.*



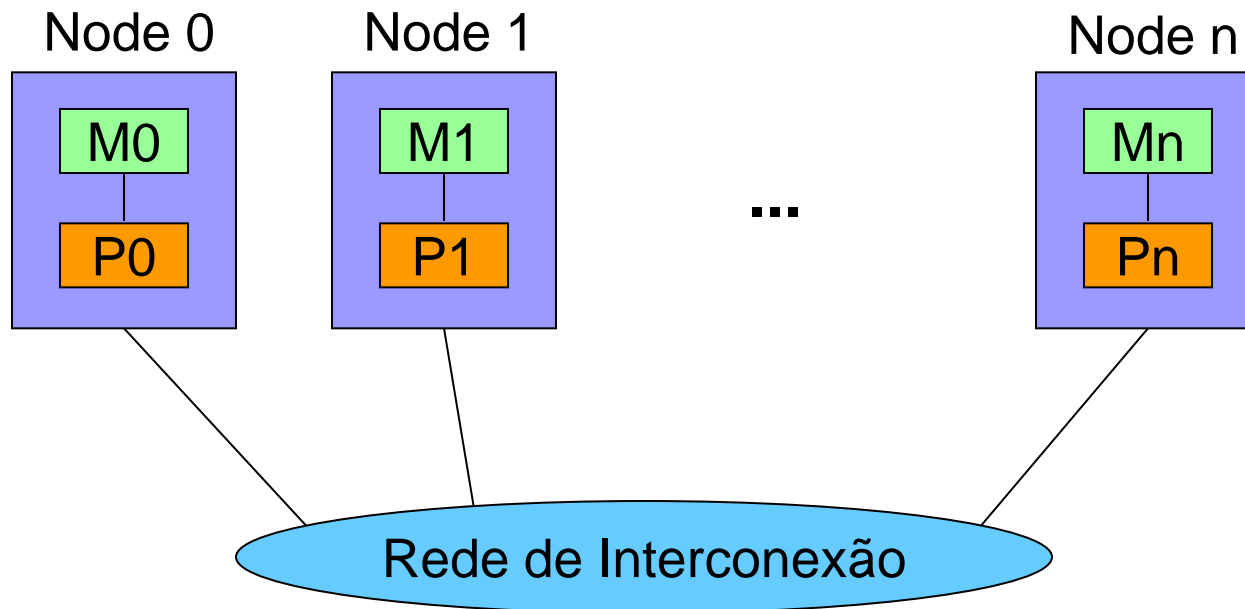
*MIMD: Processors independently fetch instructions and operate on data. Processors communicate directly (as shown), or through shared memory.*

# Scalable Shared-Memory Machines



No-bus, memory physically distributed, memory treated as global address space – thus shared

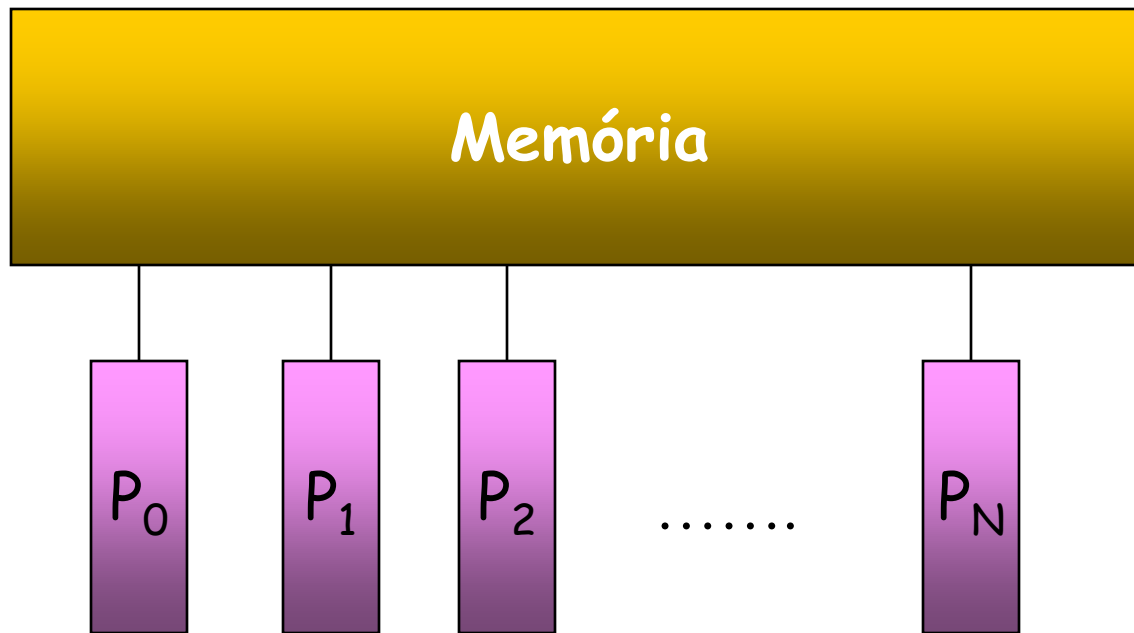
# Tipos Básicos de Arquiteturas Paralelas



Computador com Memória Distribuída



# Tipos Básicos de Arquiteturas Paralelas



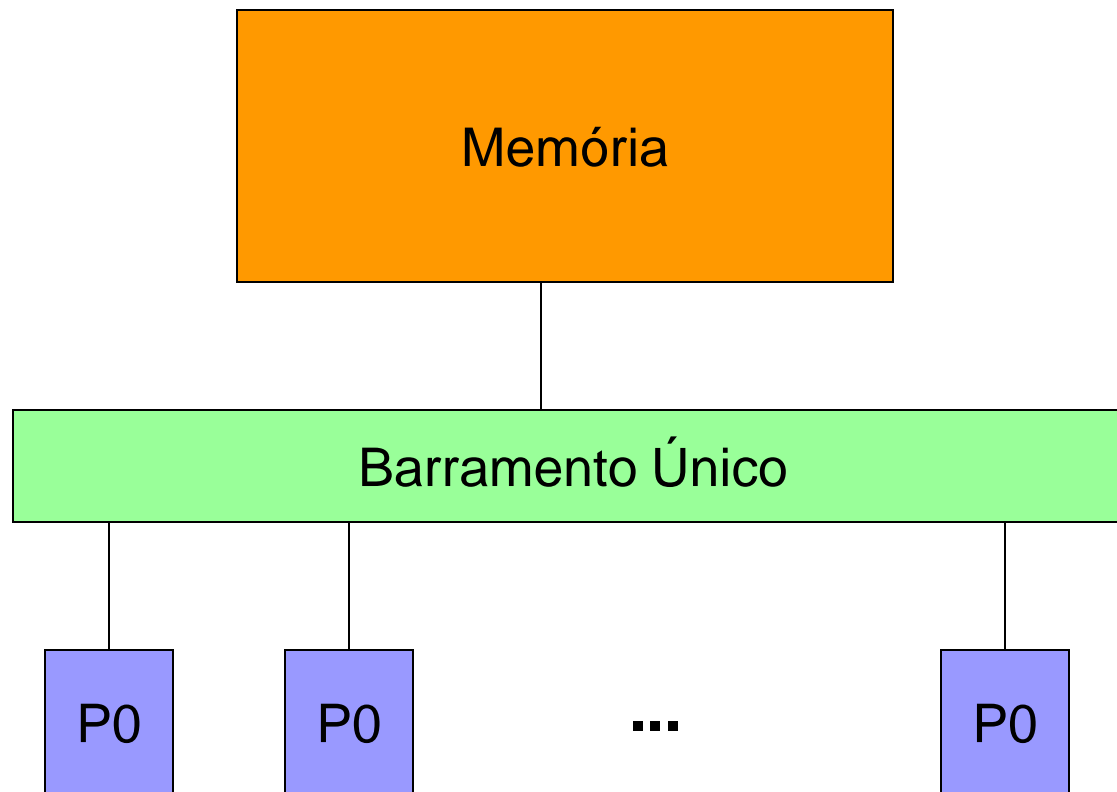
Computador com Memória Compartilhada



# Arquitetura UMA

- Arquiteturas com memória única global
- Tempo de acesso à memória é uniforme para todos os nós de processamento
- Nós de processamento e memória interconectados através de barramento único
- Número reduzido de nós de processamento, em outras palavras, problema de escalabilidade

# Arquitetura UMA

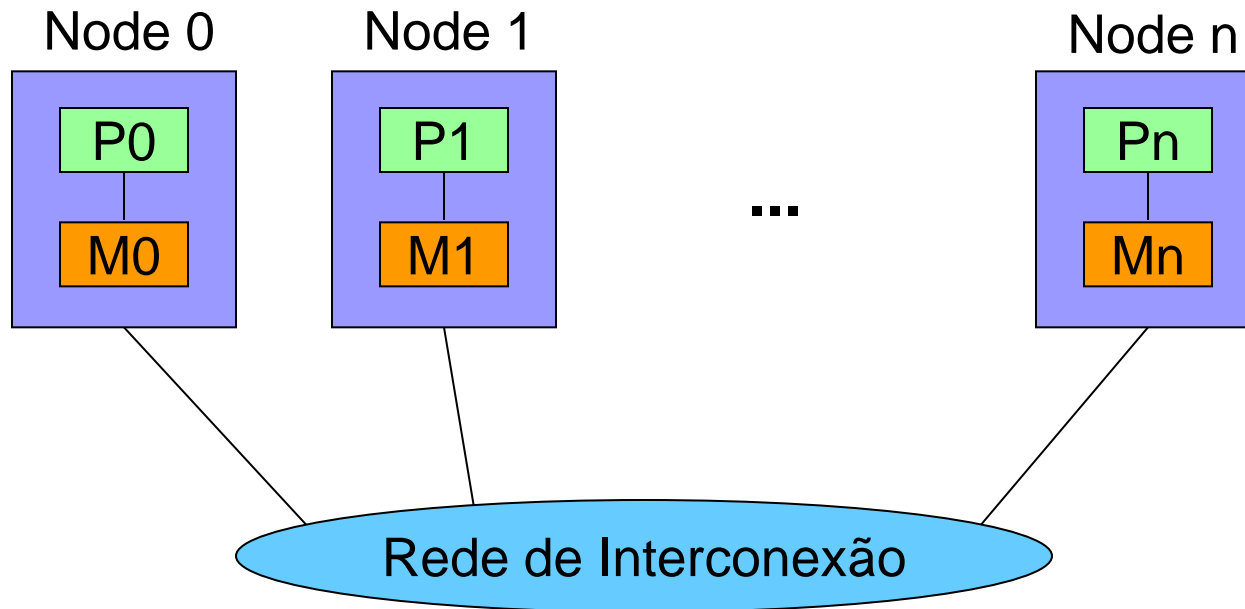




# Arquitetura NUMA

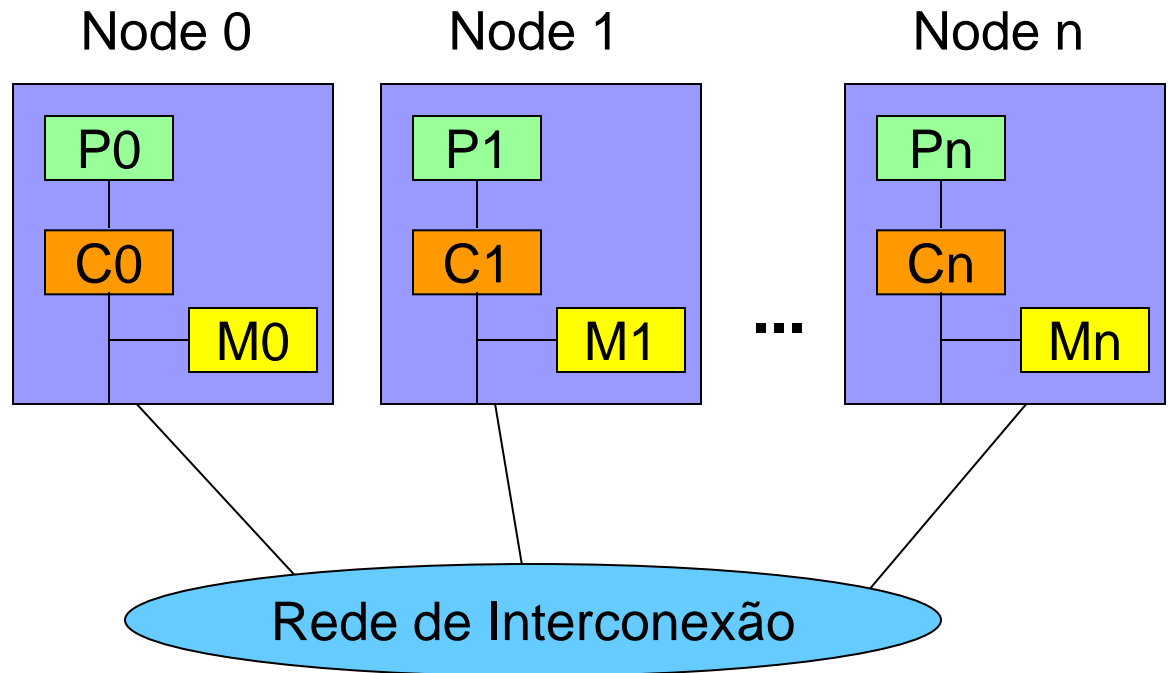
- Nessas arquiteturas a memória é dividida em tantos blocos quanto forem os processadores do sistema, e cada bloco de memória é conectado via barramento a um processador com memória local
- O acesso aos dados que estão na memória local é muito mais rápido do que aos dados em uma memória remota

# Arquitetura NUMA



# Arquitetura CC-NUMA

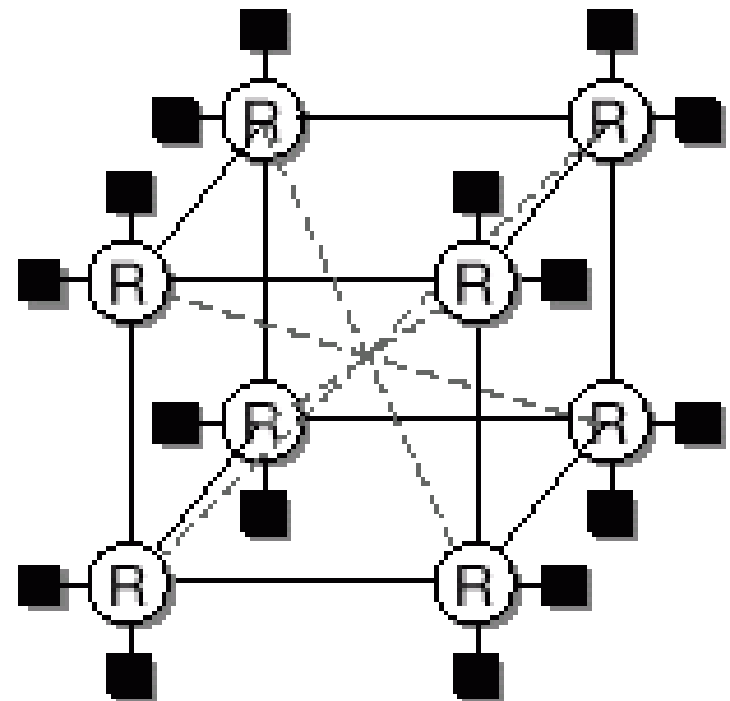
- Cada nó processador possui uma cache local para reduzir o tráfego na rede de interconexão



# Arquitetura de Memória

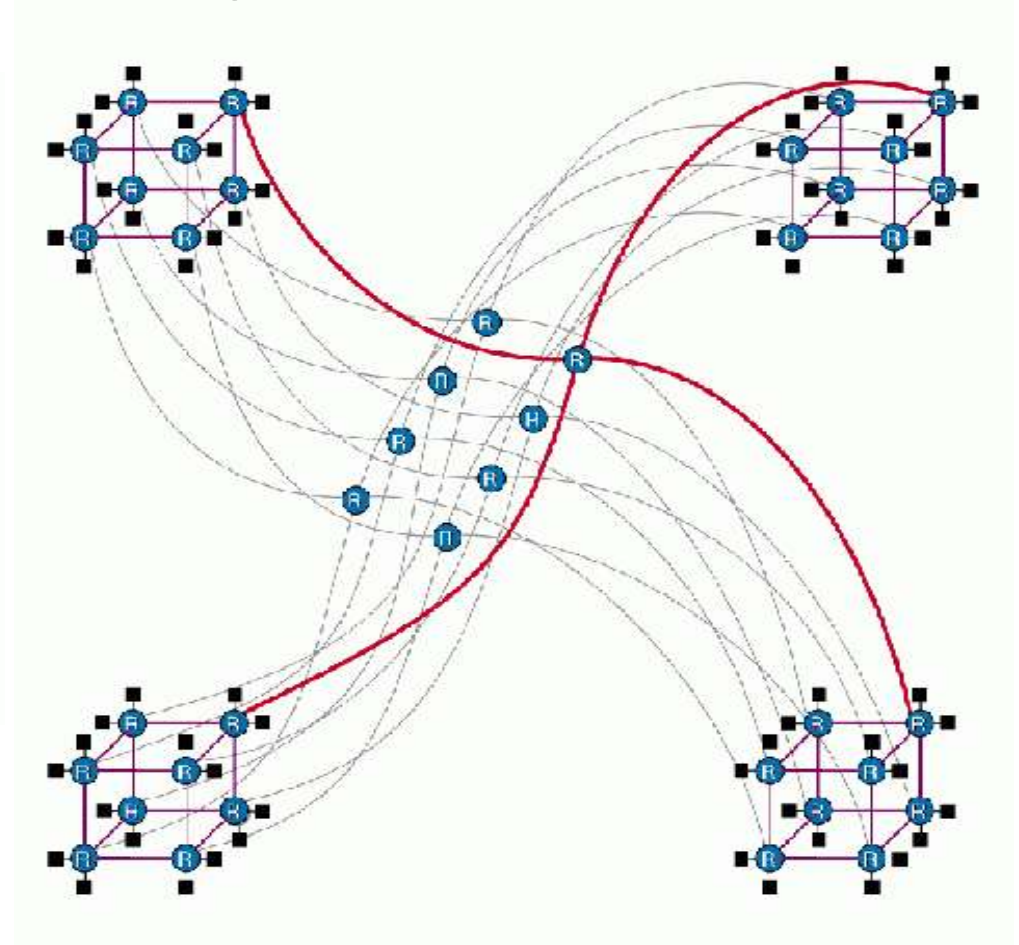
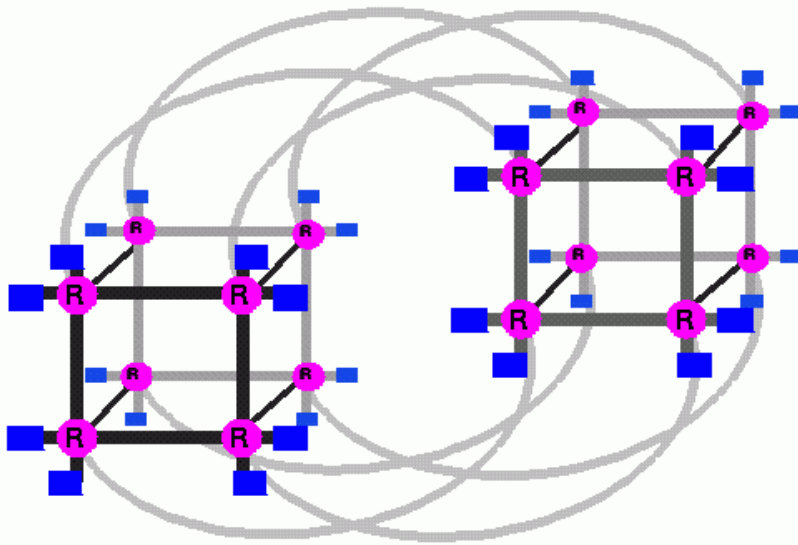
32 Processor System

- Sistemas Híbridos – SGI
  - Origin3000 server - Altix
  - Arquitetura DSM - Distributed Shared Memory
  - ccNUMA - cache-coherent Non-Uniform Memory Access
  - Interligação por Craylink
  - Vértice do cubo:
    - Roteador
    - 2 Hubs
    - Até 4 processadores
    - Memória local



# Arquitetura de Memória

- Sistemas Híbridos – SGI - Origin3000 server -Altix





# Interconnect Topologies for Parallel Systems

A major consideration for parallel systems is the manner in which the processors, memories, and switches communicate with each other. The connections among these define the *topology* for the machine.

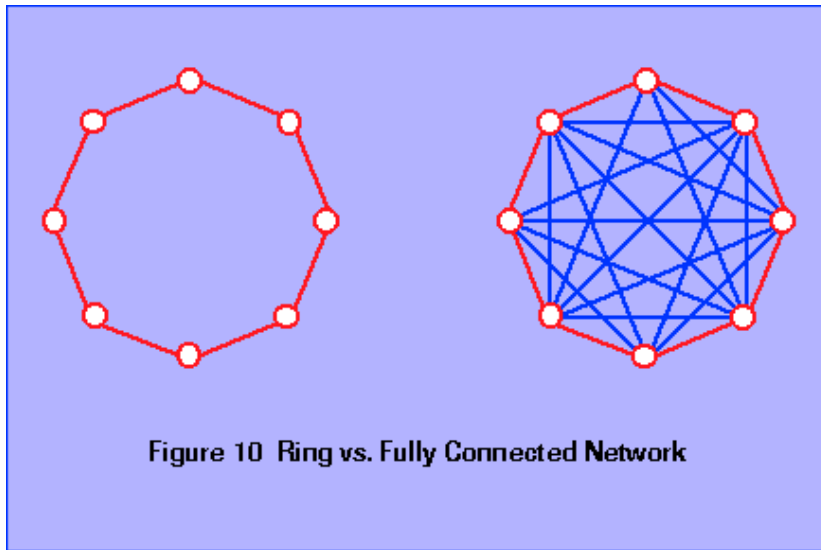


Figure 10 Ring vs. Fully Connected Network

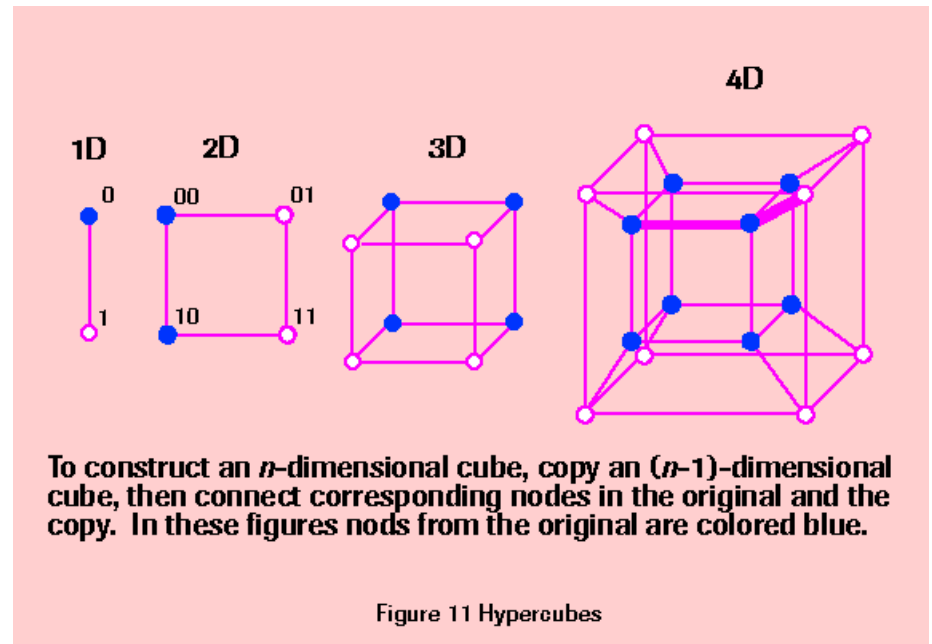


Figure 11 Hypercubes

Ring vs. Fully Connected Network

Hypercubes

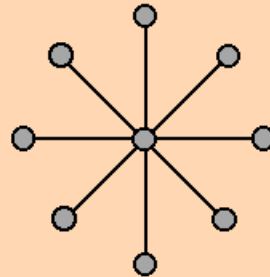
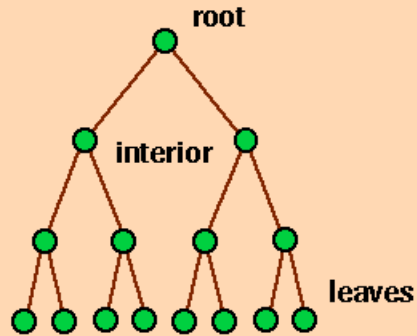
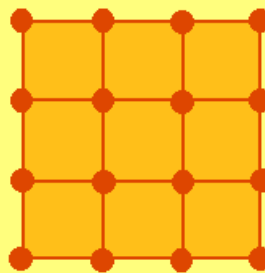


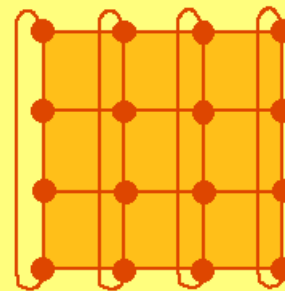
Figure 12 Tree and Star

## Tree and Star Topologies

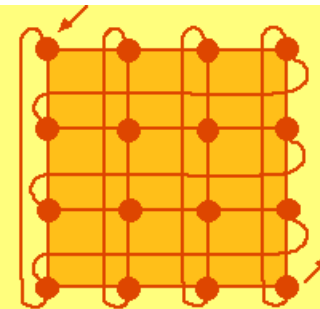
## Mesh Topologies



Planar Mesh



Wraparound  
Connections



Twisted Torus

Figure 13 Mesh Topologies



# Processo / Threads

## ■ Processo

- Criado para a execução de um programa pelo sistema operacional
- Um processo pode gerar cópias, chamadas de processos “filhos”

## ■ Threads

- São partes autônomas de código, criadas dentro de um mesmo processo
- Todas as threads de um processo compartilham os mesmos recursos, em particular o mesmo espaço de endereçamento
- A custo de criação, comunicação e sincronização entre threads é bem menor que entre processos

# Multithreading

- Os sistemas operacionais modernos suportam o conceito de *threads*, ou seja, tarefas independentes dentro de um mesmo processo que podem ser executadas em paralelo ou de forma intercalada no tempo, dentro do esquema tradicional de *time-sharing*
- O contexto específico de uma *thread* é semelhante ao contexto de uma função e, conseqüentemente, a troca de contexto entre *threads* implica no salvamento e recuperação de contextos relativamente leves. Este fato é o principal atrativo do uso de *threads* em contraposição ao uso de processos

# Multithreading

- A comunicação entre tarefas é muito simplificada numa implementação baseada em *threads*, uma vez que neste caso as tarefas compartilham o mesmo espaço de endereçamento de memória do processo como um todo que engloba as *threads*, eliminando a necessidade do uso de esquemas especiais, mais lentos de comunicação entre processos providos pelos sistemas operacionais
- As arquiteturas *multithreading* procuram reduzir o overhead na troca de contexto entre *threads*



# Paralelismo

- As arquiteturas paralelas permitem a execução das tarefas em menor tempo, através da execução em paralelo de diversas tarefas
- O paralelismo pode ser obtido em diversos níveis, com ou sem o uso de linguagens de programação paralela
- Níveis de paralelismo:
  - Nível de instrução (granulosidade fina)
  - Nível de threads (granulosidade média)
  - Nível de processo (granulosidade grossa)

# Escalonamento

- Há dois tipos básicos de escalonamento
  - Baseado em processos
  - Baseado em threads
- O controle é feito por um diagrama com pelo menos três estados básicos:
  - Pronto
  - Executando
  - Aguardando
- As linguagens de programação possuem primitivas específicas para criação de threads e processos



# Conceitos Básicos

- Execução concorrente está associada a idéia de um servidor atendendo a vários clientes através de uma política de escalonamento
- Execução paralela está associada ao modelo de vários servidores atendendo a vários clientes simultaneamente no tempo





# Ementa

---

- Introdução
- Tópicos Básicos em arquitetura de Computadores
- Arquitetura de processadores e memória
- Medidas de desempenho e análise
- Clusters
- Comentários Finais

# Medidas de Desempenho

- Vazão (*throughput*): taxa na qual os pedidos são atendidos pelo sistema
- Utilização: fração do tempo em que o recurso permanece ocupado
- Tempo de resposta: tempo decorrido entre o pedido e o início/conclusão da realização do serviço (latência)
- Escalabilidade: um sistema é dito escalável quando a eficiência se mantém constante à medida que o número de processadores ( $n$ ) aplicado à solução do problema cresce

# Algumas Metricas de Desempenho

**MIPS** = "Millions of instructions per second"

**MFLOP/S** = "Millions of floating-point operations per second"

**Theoretical Peak MFLOPS** = MFLOPS if the machine did nothing but numerical operations

**Benchmarks** = Programs designed to determine performance metrics for machines. *Examples:* LINPACK, NASA-NPB's

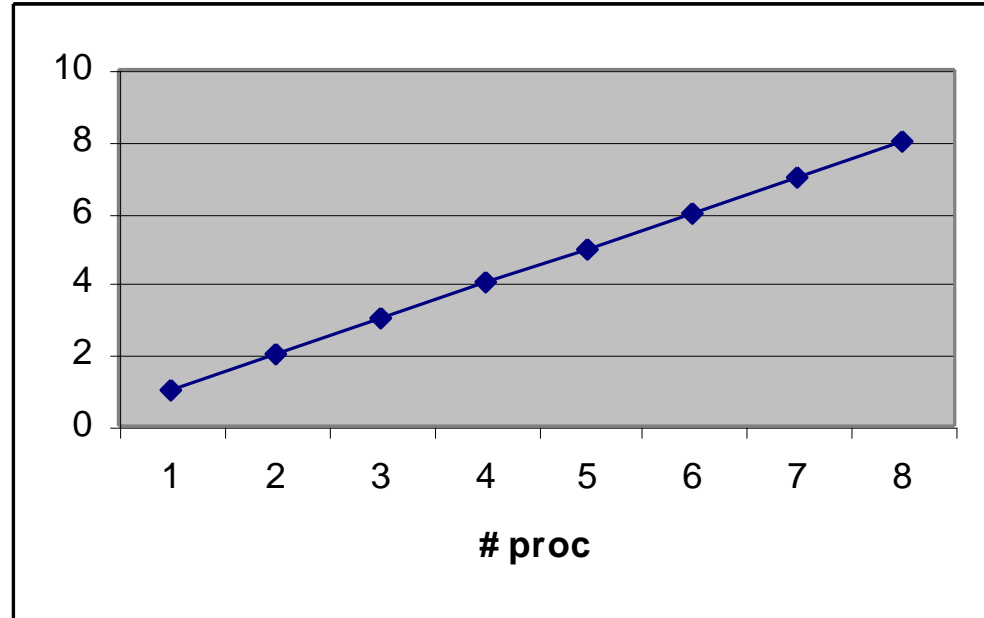
See: R.W. Hockney, The Science of Computer Benchmarking, SIAM, 1995

# Medidas de Desempenho

- $T_1$  = tempo gasto no melhor algoritmo seqüencial
- $T_p$  = tempo gasto para execução do algoritmo paralelo em  $p$  processadores

- Speed-up ideal

$$S_p = \frac{T_1}{T_p}$$

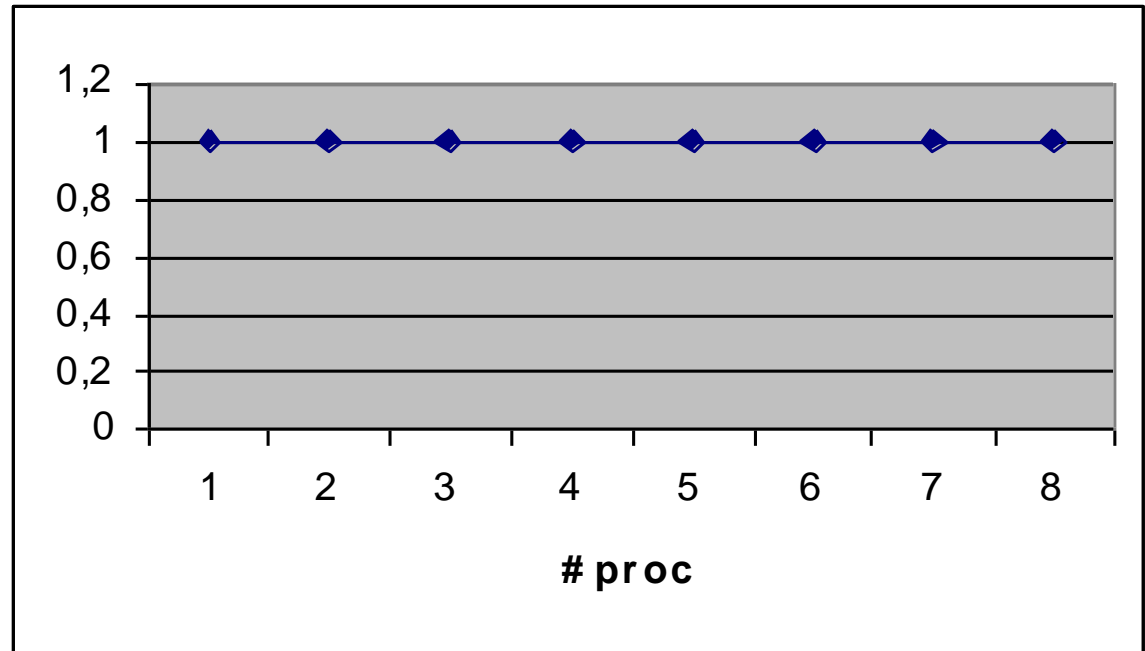


# Medidas de Desempenho

- $S_p$  = Speed-up obtido anteriormente
- $p$  = número de processadores

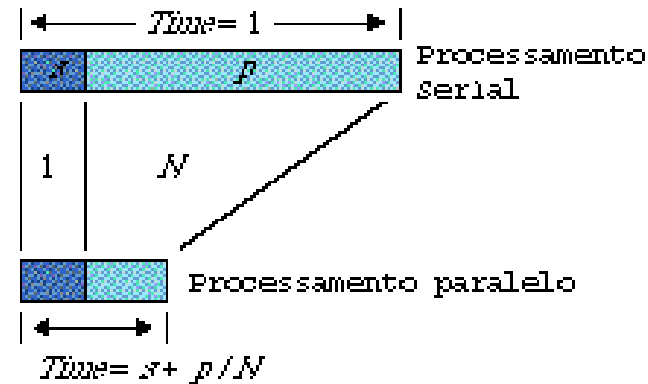
- Eficiência ideal

$$E_p = \frac{S_p}{p}$$

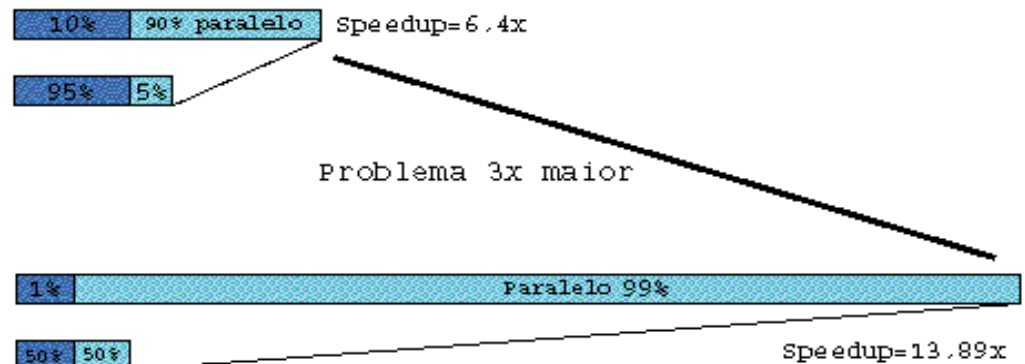


# Lei de Amdahl

$$T = s + p$$



$$S_p = \frac{T_1}{T_p}$$

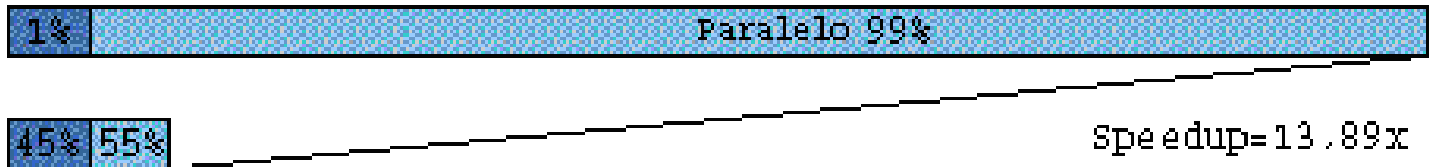


# Lei de Amdahl

- Existe um limite fundamental no *speedup* mencionado na lei de Amdahl que diz:  
*“Em qualquer sistema que tem dois ou mais modos de processamento de diferentes velocidades, o desempenho do sistema será dominado pelo modo mais lento.”*

$$F_{par} = \frac{\frac{1}{S_p} - 1}{\frac{1}{P} - 1}$$

$$S_{pred} = \frac{1}{\frac{F_{par}}{P} + (1 - F_{par})}$$



$$F_{par} = \frac{(1/1.98) - 1}{(1/2) - 1} \cong 0.9898$$

$$S_{pred} = \frac{1}{\frac{0.9898}{16} + (1 - 0.9898)} \cong 13.89$$



# Escalonamento do Problema

- A lei de Amdahl só é relevante para um problema de tamanho fixo, ou quando a fração serial é independente do tamanho do problema, o que é difícil de se encontrar na prática;
- Geralmente a fração paralela aumenta com o aumento do tamanho do problema;
- A taxa de crescimento da fração paralela pode ser caracterizada mantendo uma quantidade constante enquanto  $p$  varia:
  - Tamanho (Amdahl), trabalho computacional, tempo de execução, memória por processador, eficiência, etc.

# Escalabilidade

- Escalabilidade se refere a como um algoritmo paralelo pode utilizar de forma eficiente processadores adicionais;
- Um algoritmo paralelo é **ESCALAVÉL** quando o número de processadores cresce se sua eficiência for constante quando o problema cresce;

# Lei de Gustafson

- A lei de Gustafson apresenta uma visão mais otimista do que a lei de Amdahl, levando em conta o conceito de escalabilidade
- Na prática, grandes sistemas multiprocessados normalmente permitem um aumento do tamanho do problema sem acarretar num acréscimo significativo do tempo computacional. Conseqüentemente, o tamanho do problema não é independente do número de processadores.
- Em outras palavras, para usarem mais processadores deve-se aumentar o tamanho do problema a ser executado em paralelo.

# Comparação das duas leis

- Suponha que  $s$  seja a fração executada seqüencialmente e  $p$  a fração executada em paralelo. Fixando o tempo total de execução ( $s+p$ ) em um único processador como sendo uma constante e fazendo, por conveniência algébrica,  $s+p=1$ , da lei de Amdahl tem-se:

$$S_n = \frac{s + p}{s + \frac{p}{n}} = \frac{1}{s + \frac{1-s}{n}}$$

# Comparação das duas leis

- O fator de *speedup* calculado através da lei de Gustafson considera que a execução em um único processador deve ser  $s+np$ , onde  $n$  é o número de parcelas paralelas que devem ser executadas seqüencialmente. Então, a lei de Gustafson é definida como sendo:

$$S_n = \frac{s + np}{s + p} = s + np = s + (1 - s)n$$

# Comparação das duas leis

- Para exemplificar, suponha que a parcela serial de um código corresponda a 5% e o mesmo será executado em 20 processadores: de acordo com a lei de Amdahl obtém-se um *speedup* de 10,26 ao invés de 19,05 de acordo com a lei de Gustafson.
- Na prática, um programa paralelo encontra-se situado em algum lugar entre essas duas leis.



# Medição de Desempenho: os Testes NPB

NAS Parallel Benchmarks

<http://www.nas.nasa.gov>

5 kernels (EP, MG, CG, 3D-FFT, IS)

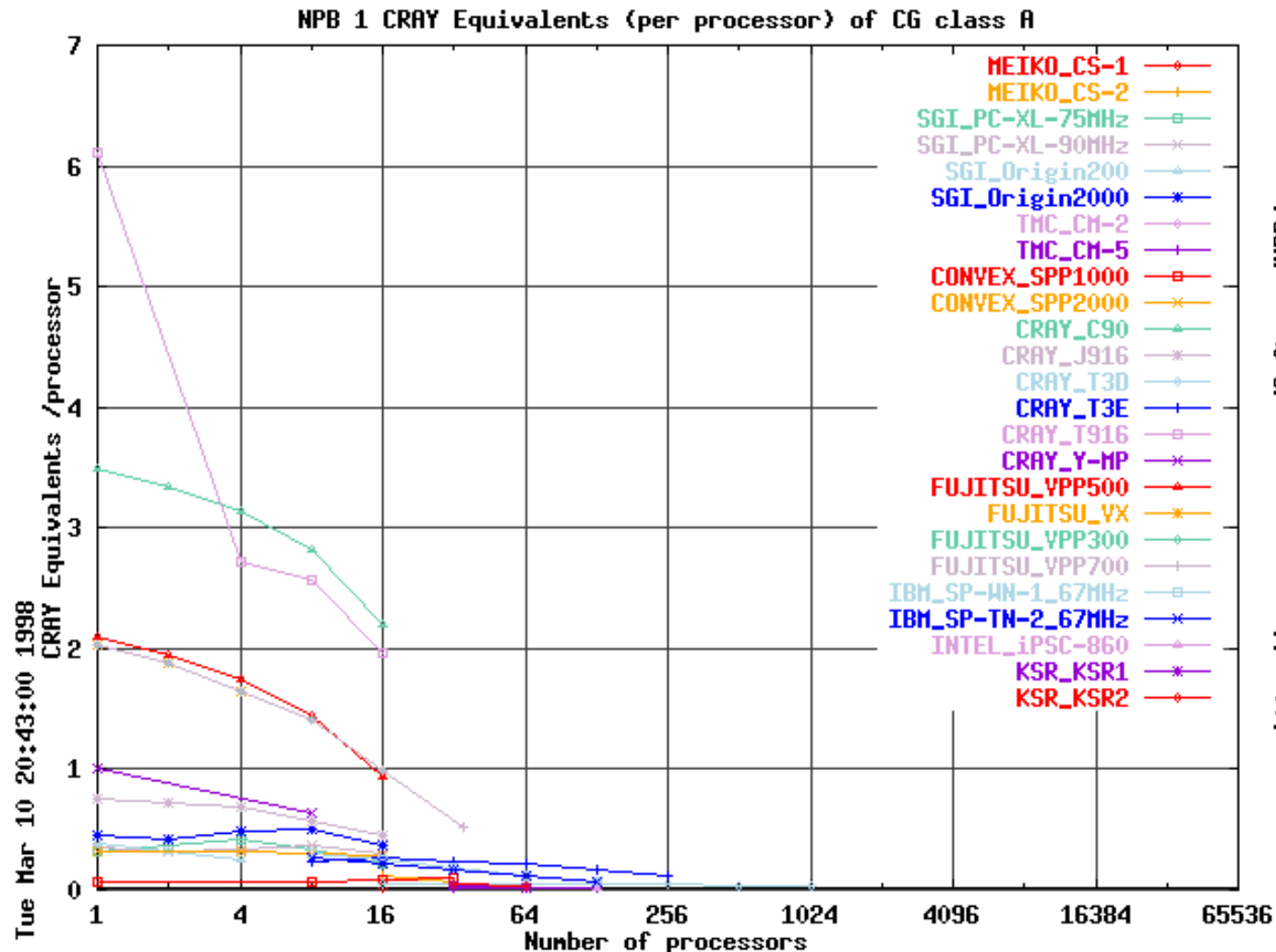
3 simulações de aplicações CFD (LU, SP, BT)

# Especificação do Benchmark CG

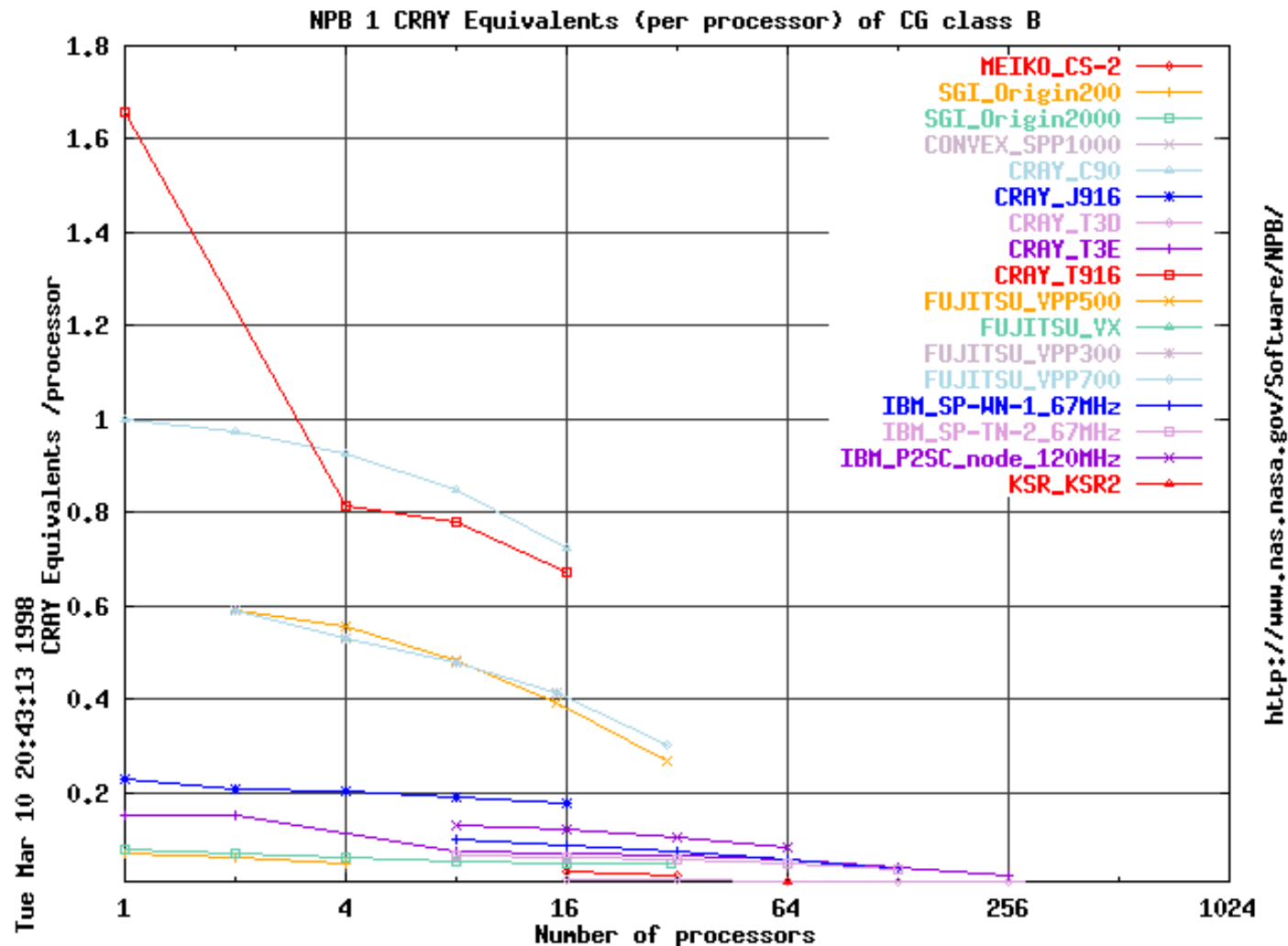
Benchmark CG	Tamanho	Flop ( $\times 10^9$ )	Mflop/s
Classe A	14x103	1.508	127
Classe B	75x103	54.89	447
Classe C	1.5E05	N/A	N/A



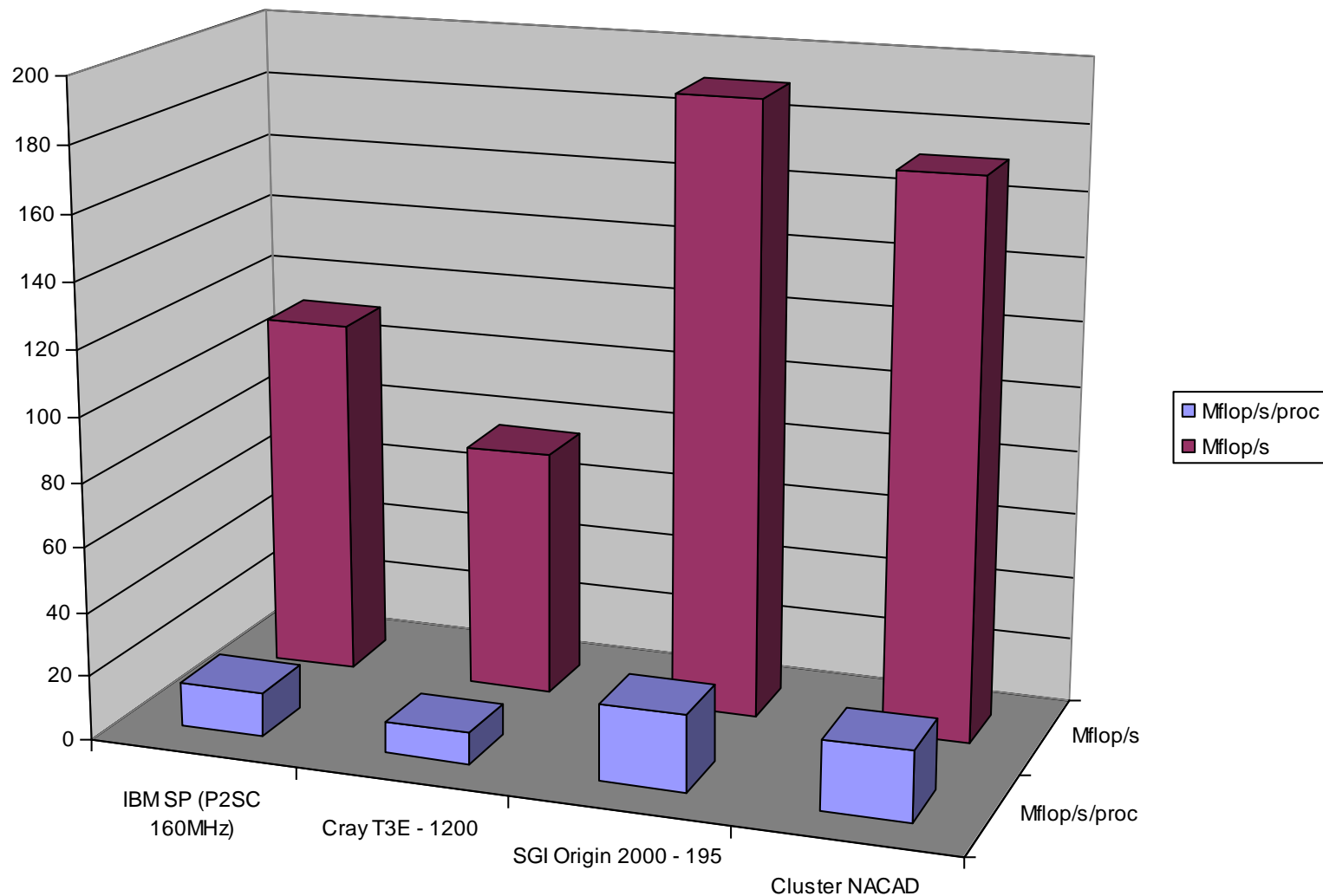
# Resultados NPB CG Classe A



# Resultados NPB CG Classe B



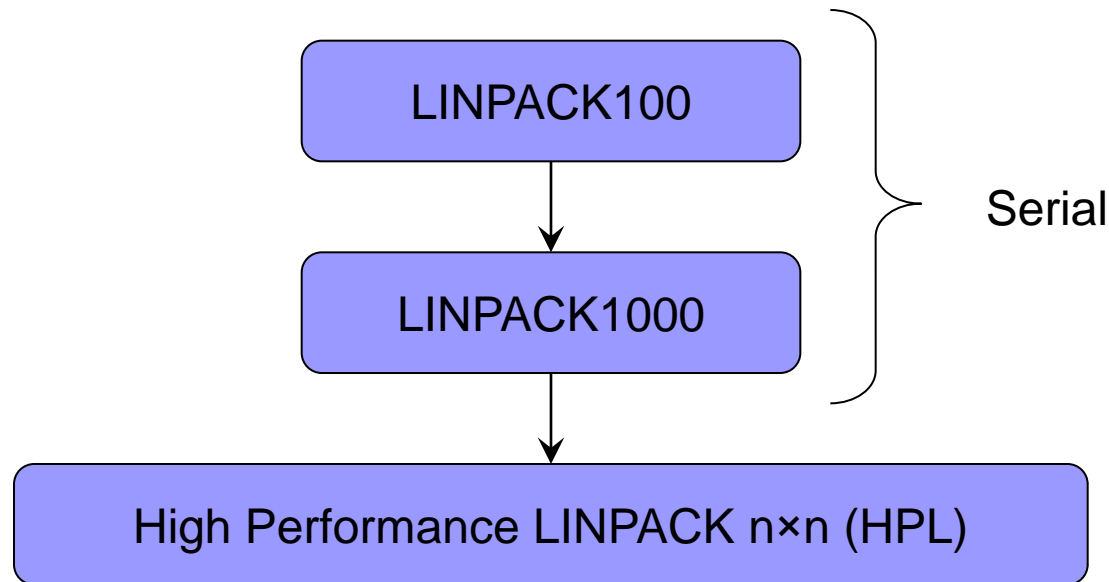
# Resultados NPB CG Classe C



# O Benchmark LINPACK

- Surgiu como pacote de solução de sistemas lineares ( $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$ )
- Foi utilizado como benchmark inicialmente para predição de tempo de execução;
- Relacionou-se o tempo de execução de 23 computadores para um problema de ordem 100 (LINPACK100)
- “O surgimento do TOP500”;
- Atualmente conta com dados de mais de 1300 sistemas de computação.
- <http://www.netlib.org/benchmark/hpl>

# A Evolução do LINPACK



**Outros pacotes para solução de sistemas lineares:**

EISPACK, LAPACK e ScaLAPACK

# Núcleo de Cálculo do LINPACK

- BLAS (Basic Linear Algebra Subprograms):  
<http://www.netlib.org/blas>
  - Nível 1: operações vetor-vetor;
  - Nível 2: operações matriz-vetor;
  - Nível 3: operações matriz-matriz.
- Procurar utilizar BLAS otimizadas:
  - ATLAS (gerador gratuito de BLAS otimizada para qualquer sistema);  
ATLAS: <http://www.netlib.org/atlas>
  - MKL (pacote próprio da Intel)
  - **Goto BLAS**

# Filosofia do Benchmark

- Extrair o desempenho MÁXIMO da máquina rodando o LINPACK;
- Em que consiste a lista TOP500:
  - Resultados para:
    - $R_{\max}$  = Performance em Gflop/s para o maior problema executado;
    - $N_{\max}$  = A dimensão do maior problema executado;
    - $N_{1/2}$  = O tamanho do problema onde foi alcançado metade de  $R_{\max}$ ;
    - $R_{\text{pico}}$  = O pico teórico de desempenho da máquina

# Fatores que Afetam o Desempenho

- Tipo de aplicação;
- Algoritmo;
- Tamanho do problema;
- Nível da linguagem de programação;
- Implementação;
- Esforço humano em otimizar o código;
- Sistema Operacional;
- Hardware;
- Idade do compilador;
- Habilidade do compilador em gerar código otimizado;
- Qualidade do núcleo de cálculo (BLAS otimizada p. ex.);
- Etc...



# Resumo do Benchmark LINPACK

- O que eu preciso para fazer o Benchmark?
  - Recursos humanos: 1 operador + 1 suporte (tempo integral);
  - Fontes e bibliotecas:
    - HPL,
    - BLAS otimizada (ATLAS, MKL, etc...).
  - Compiladores Fortran e C++;
  - Máquina DEDICADA;
  - Tempo: aproximadamente 30 dias;

# The TOP500 List



<http://www.top500.org>  
Lists the top 500 supercompute  
Updated in 06/XX and 11/XX  
Presented by:  
University of Mannheim  
University of Tennessee  
NERSC/LBL

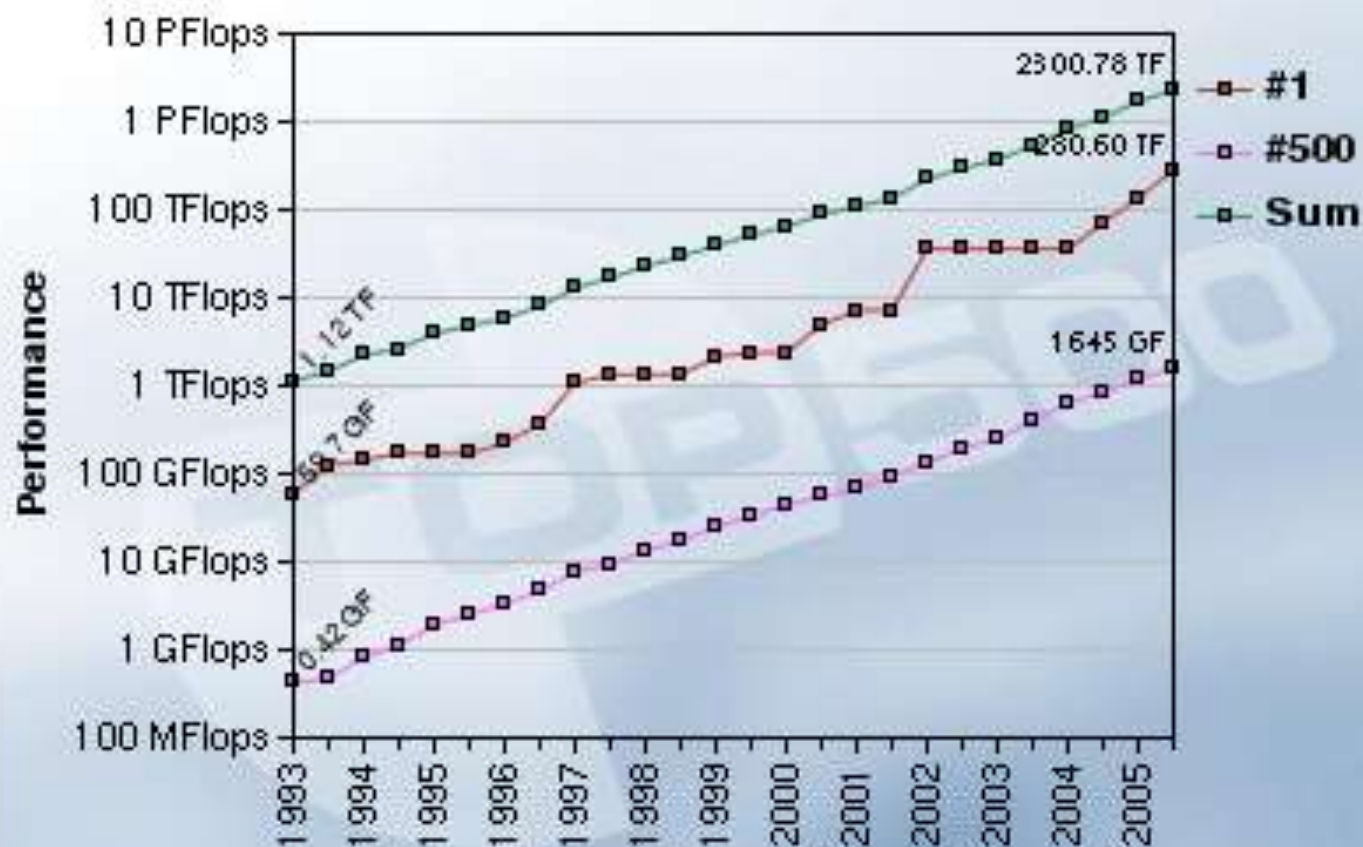
**TOP 5**  
**SUPERCOMPUTER SITES (November 2004)**

 <b>1</b> <b>BlueGene/L</b> DOE/IBM Rochester, USA BlueGene/L DDz Rmax: 70.72 TFlops	 <b>2</b> <b>Columbia</b> NASA/Ames Mountain View, USA SGI Altix/Voltaire Rmax: 51.87 TFlops	 <b>3</b> <b>Earth Simulator</b> Earth Simulator Center Yokohama NEC Rmax: 35.86 TFlops
 <b>4</b> <b>Mare Nostrum</b> Barcelona Supercomputer Center Barcelona, Spain eServer BladeCenter JS20/Myrinet Rmax: 20.53 TFlops	 <b>5</b> <b>Thunder</b> Lawrence Livermore National Lab Livermore, USA Intel Itanium2 Tiger4/Quadrics Rmax: 19.94 TFlops	

# Top500 - Lista

Rank	Site	System Vendor	Processors	Rmax	Rpeak
1	DOE/NNSA/LLNL United States	BlueGene/L eServer Blue Gene Solution IBM	131072	280600	367000
2	IBM Thomas J. Watson Research Center United States	BGW eServer Blue Gene Solution IBM	40960	91290	114688
3	DOE/NNSA/LLNL United States	ASC Purple eServer pSeries p5 575 1.9 GHz IBM	10240	63390	77824
4	NASA/Ames Research Center/NAS United States	Columbia SGI Altix 1.5 GHz, Voltaire Infiniband SGI	10160	51870	60960
5	Sandia National Laboratories United States	Thunderbird PowerEdge 1850, 3.6 GHz, Infiniband Dell	8000	38270	64512

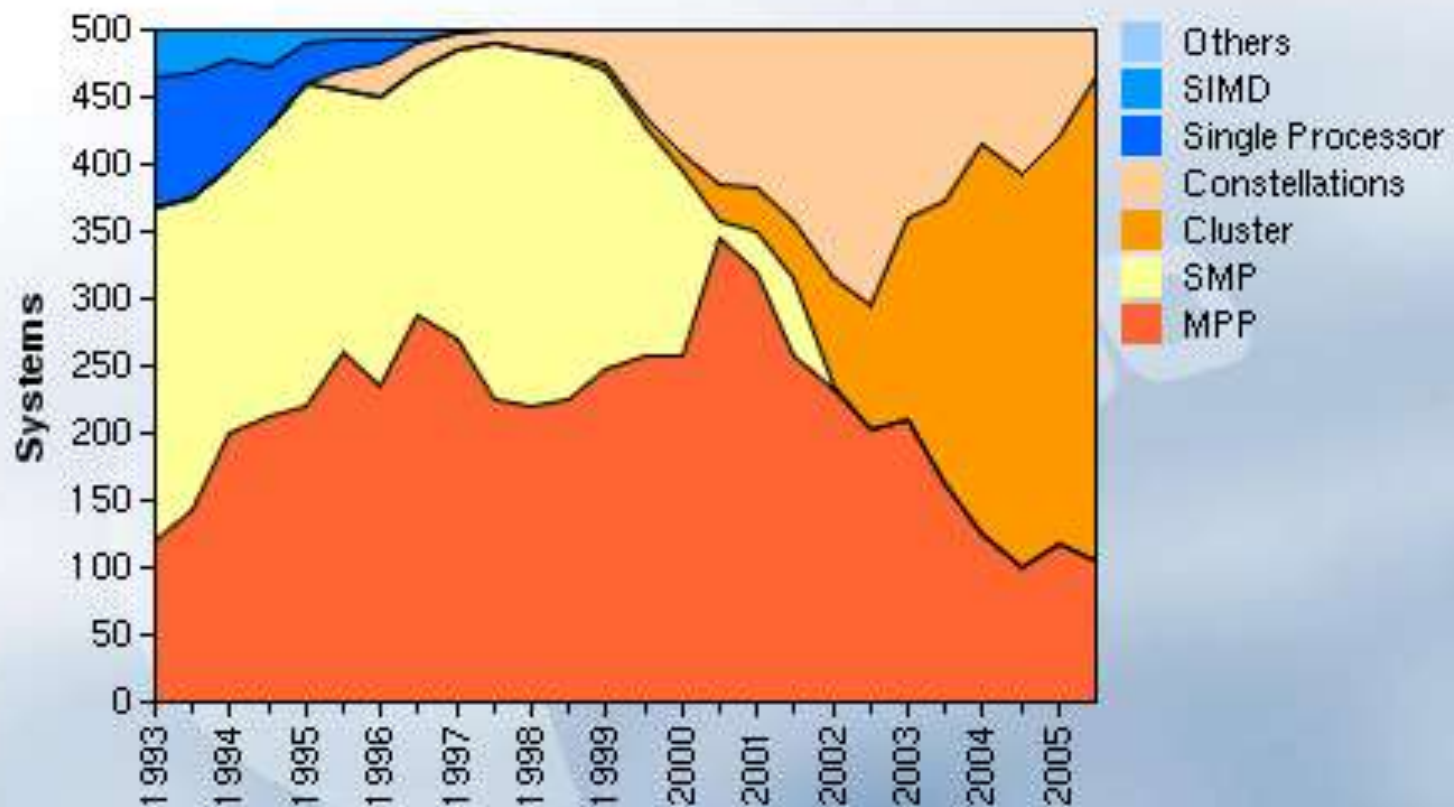
## Performance Development



## Projected Performance Development

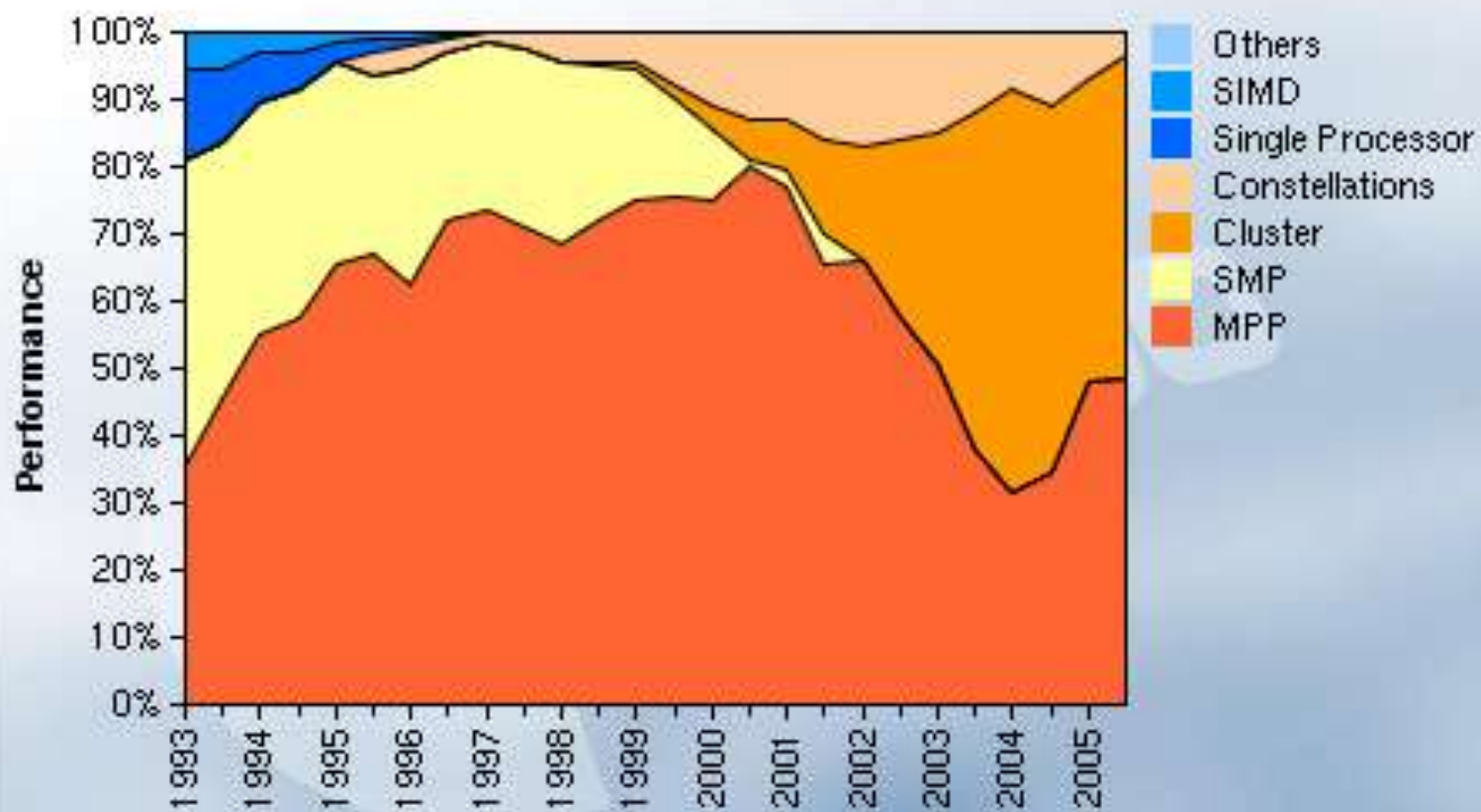


## Architectures / Systems





## Architectures / Performance



## Architectures / Systems

November 2005







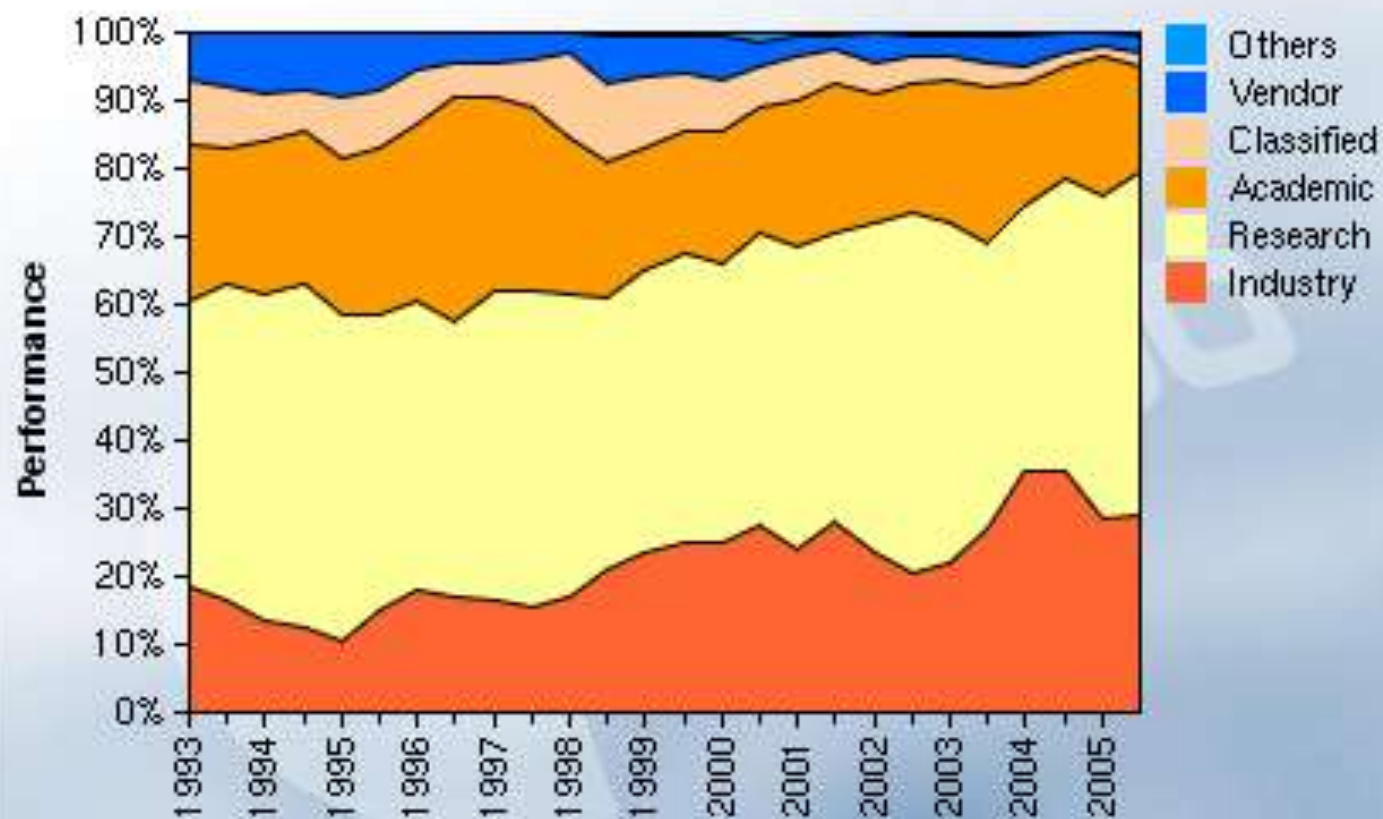
## Architectures / Performance

November 2005

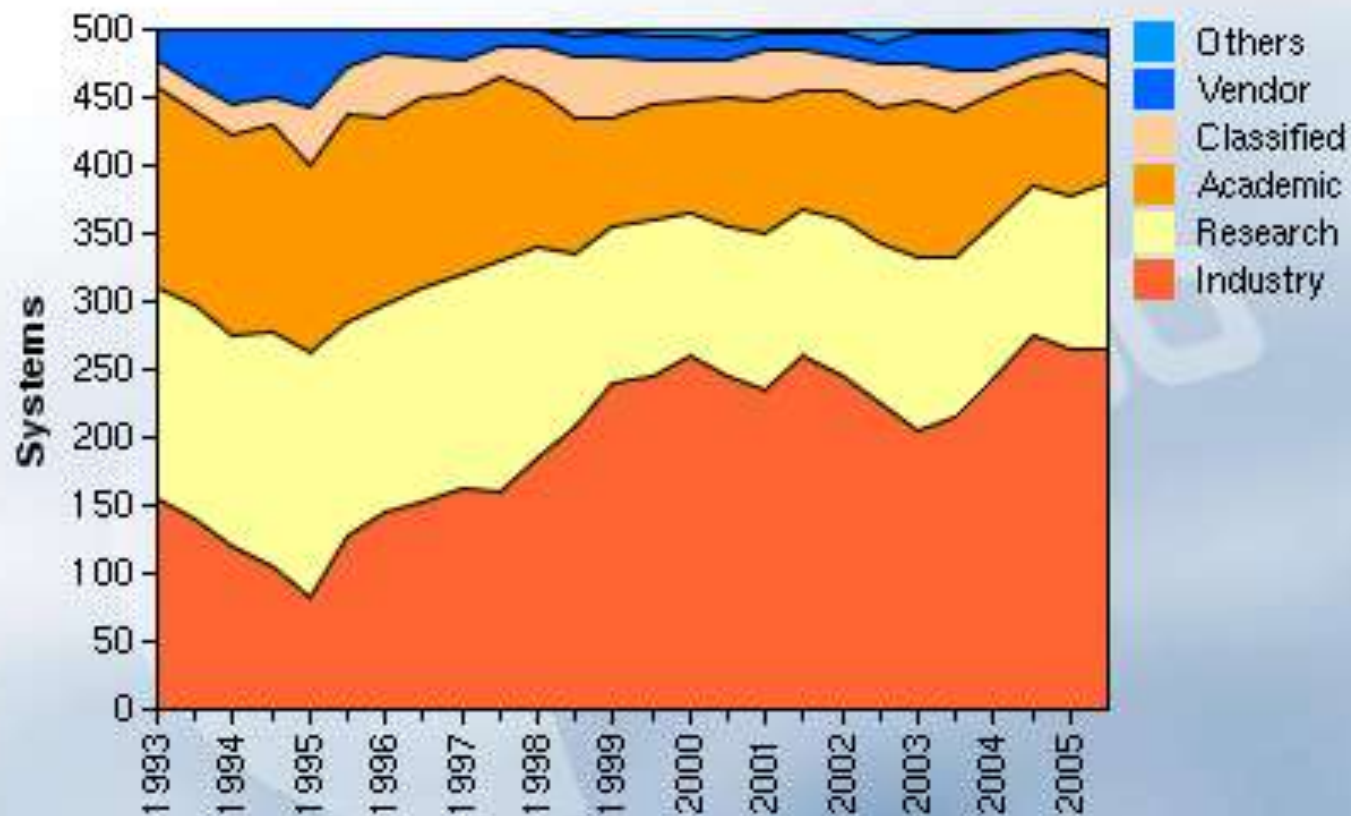




## Customer Segment / Performance



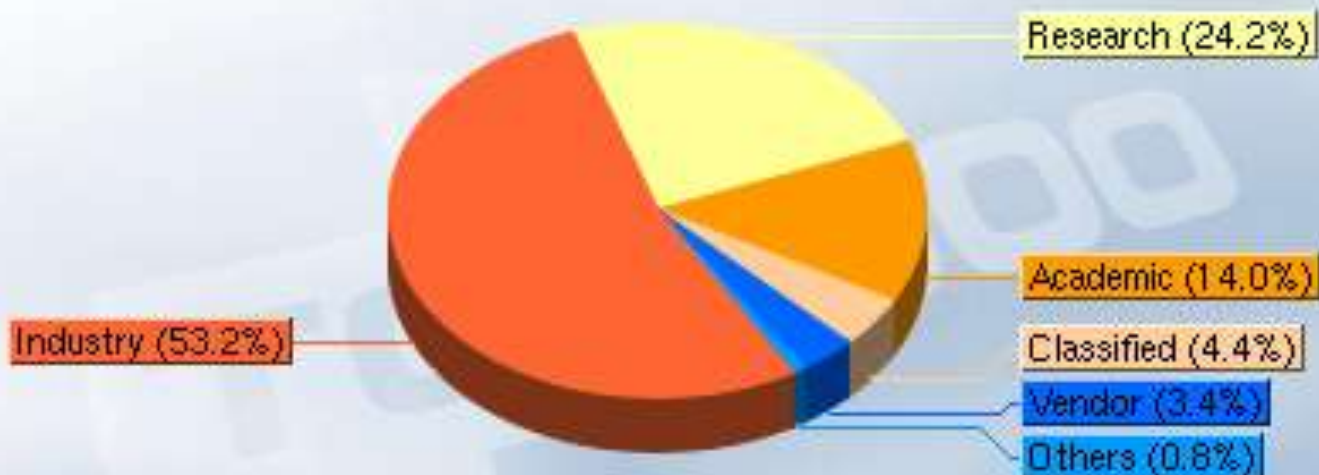
## Customer Segment / Systems





## Customer Segment / Systems

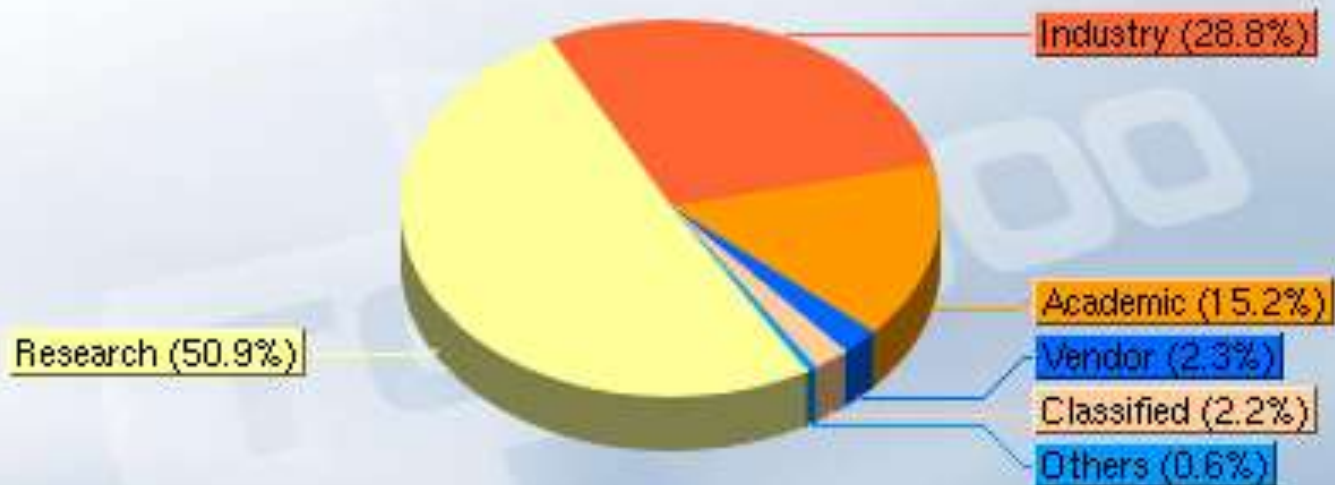
November 2005





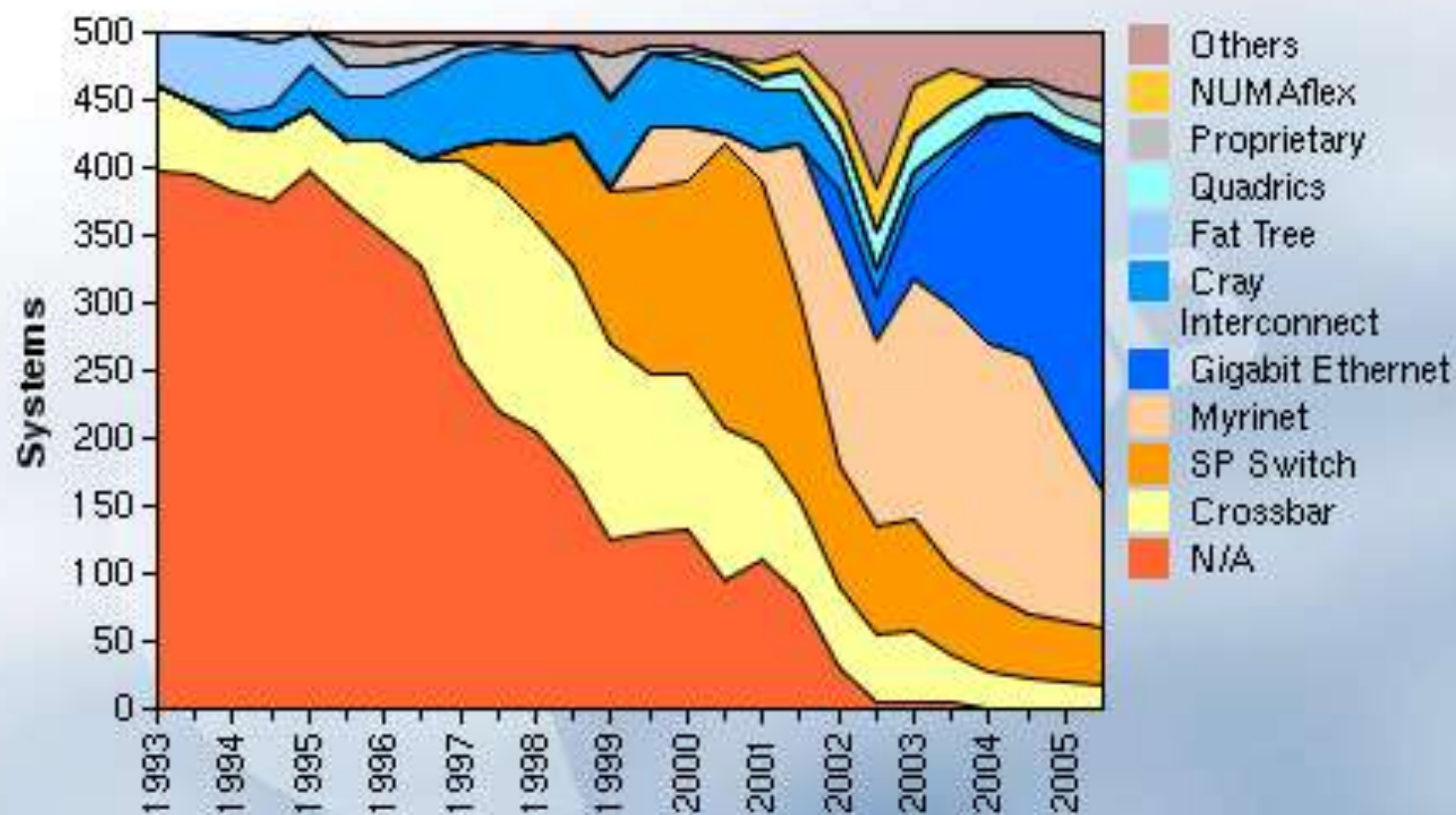
## Customer Segment / Performance

November 2005

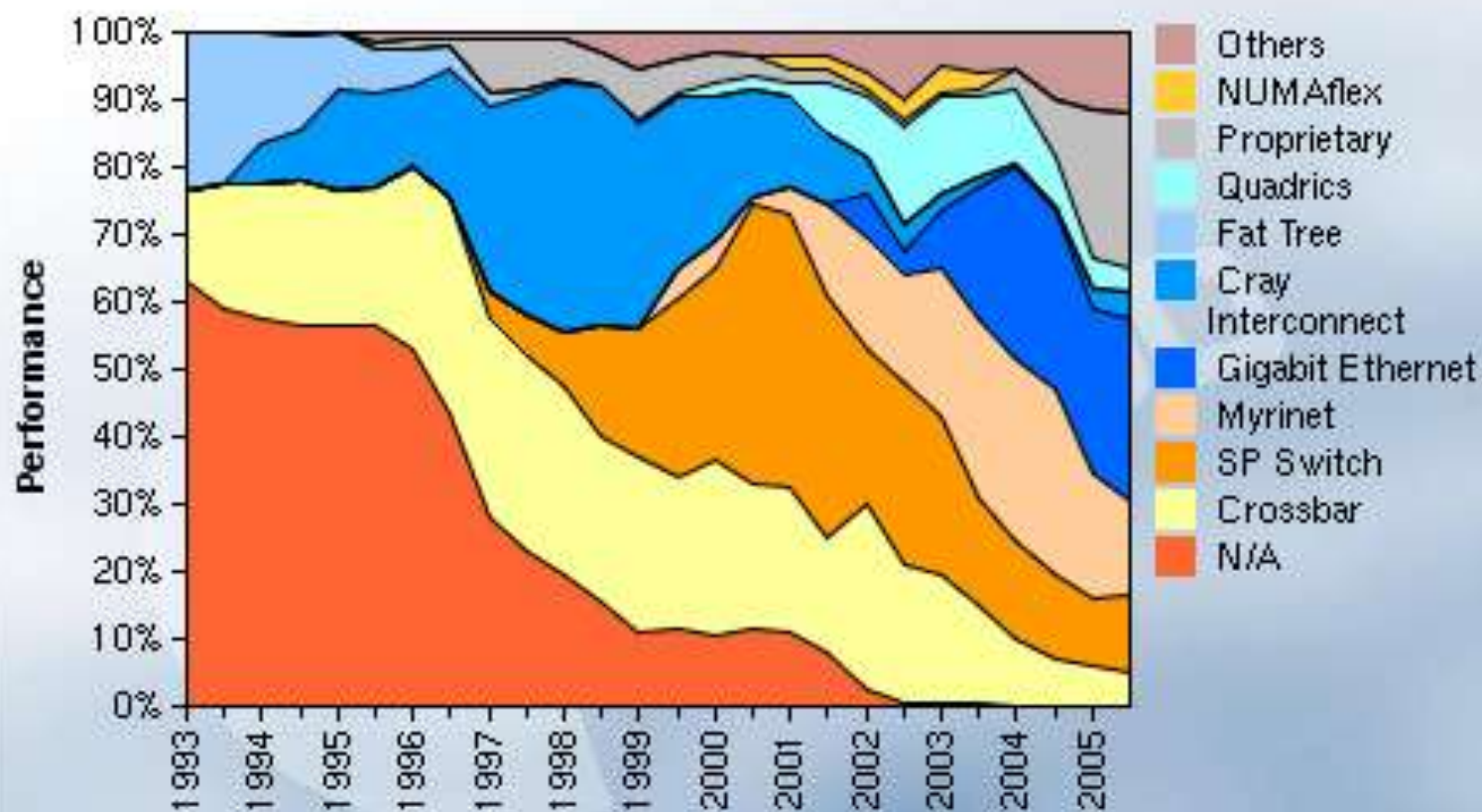




## Interconnect Family / Systems

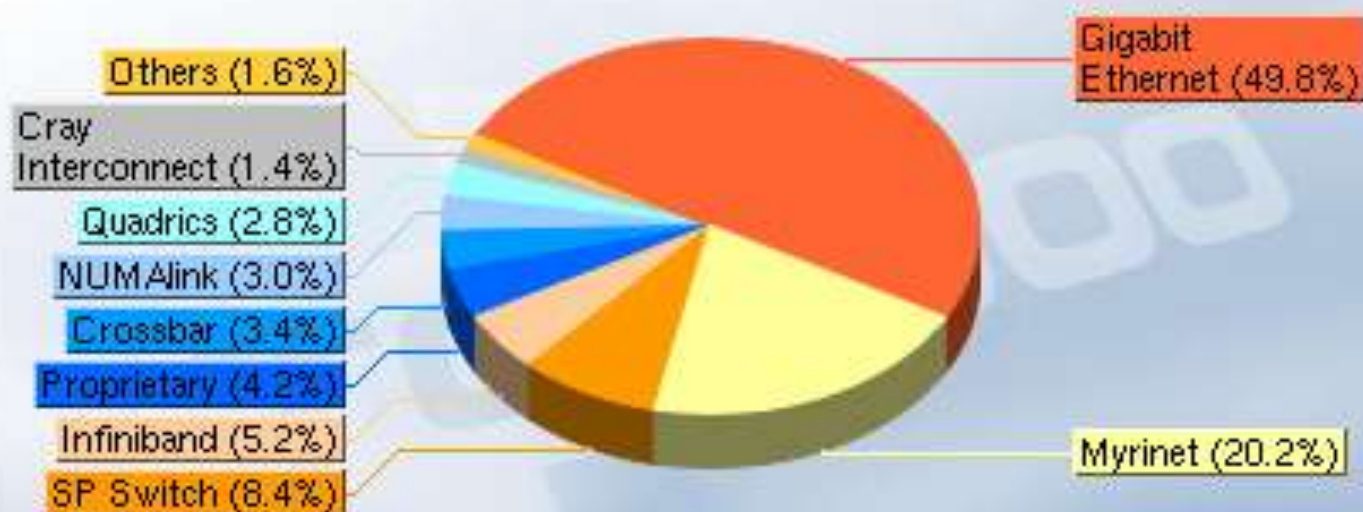


## Interconnect Family / Performance



## Interconnect Family / Systems

November 2005



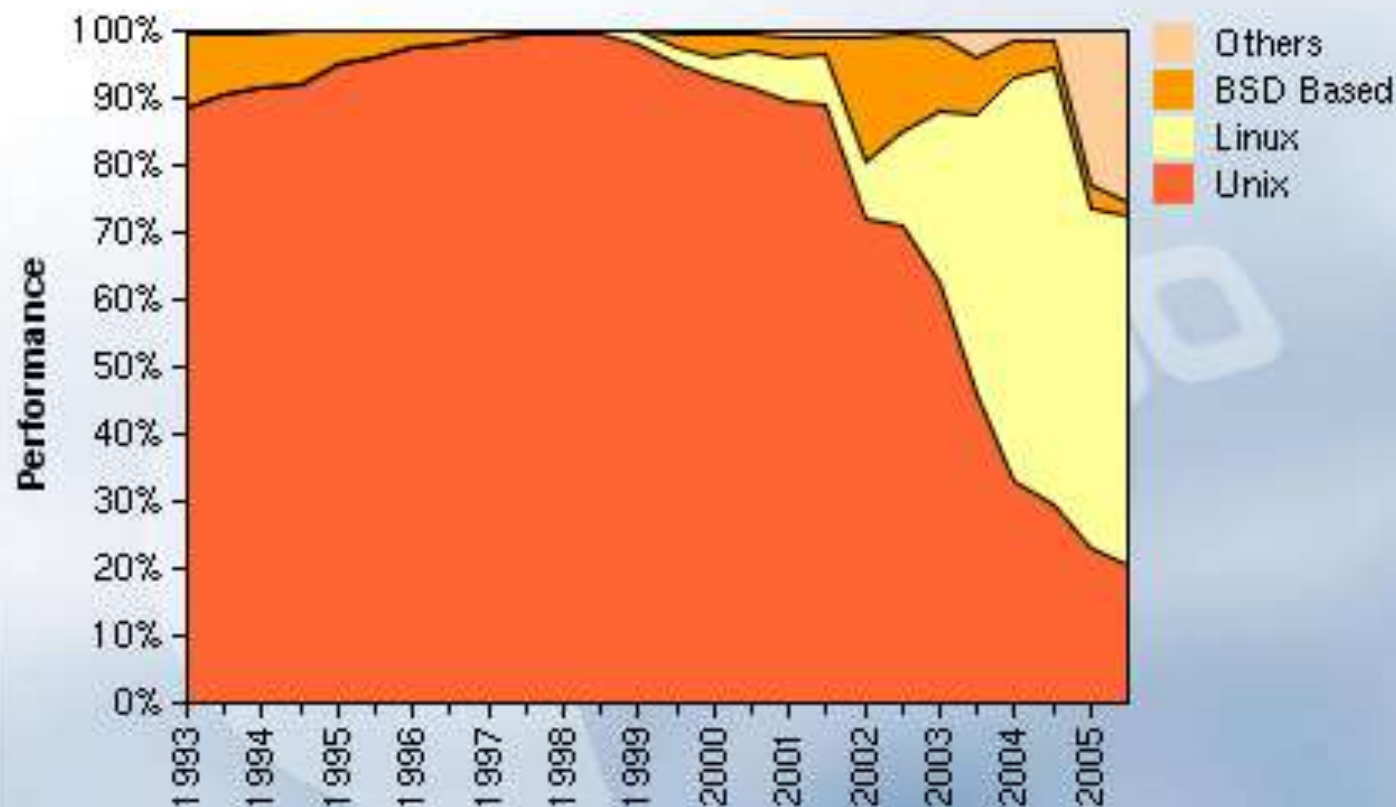


## Interconnect Family / Performance

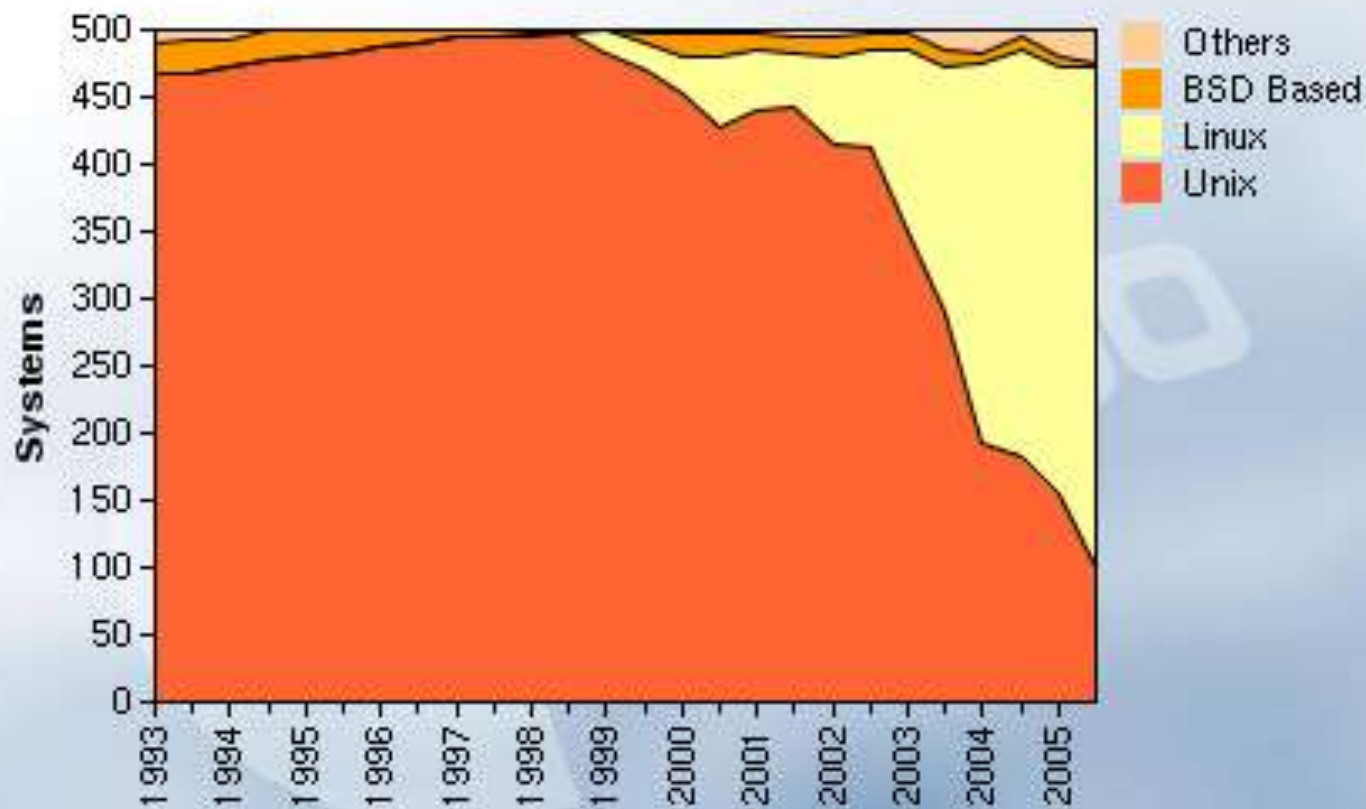
November 2005



## Operating System / Performance



## Operating System / Systems



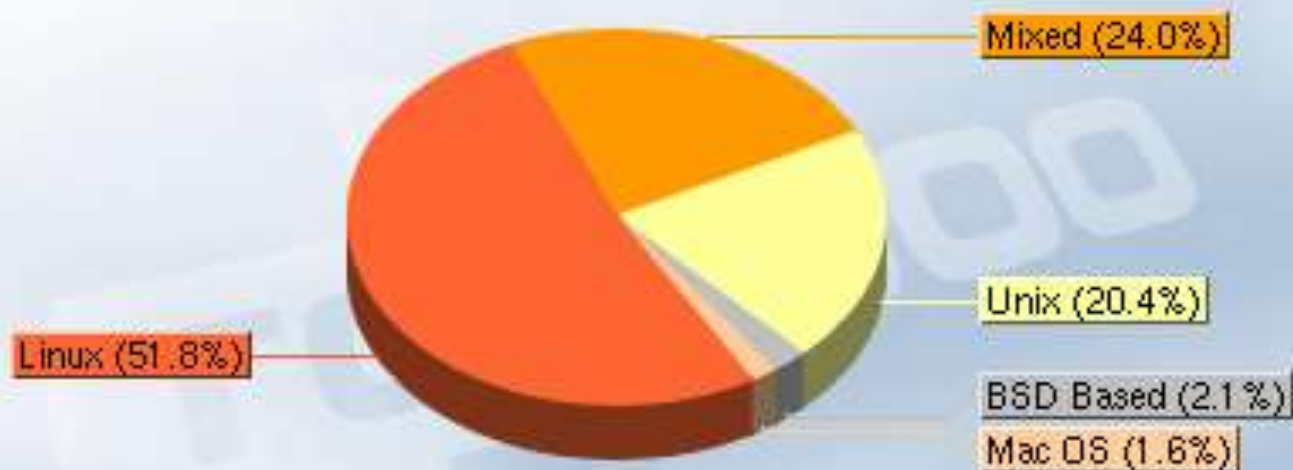
## Operating System Family / Systems

November 2005



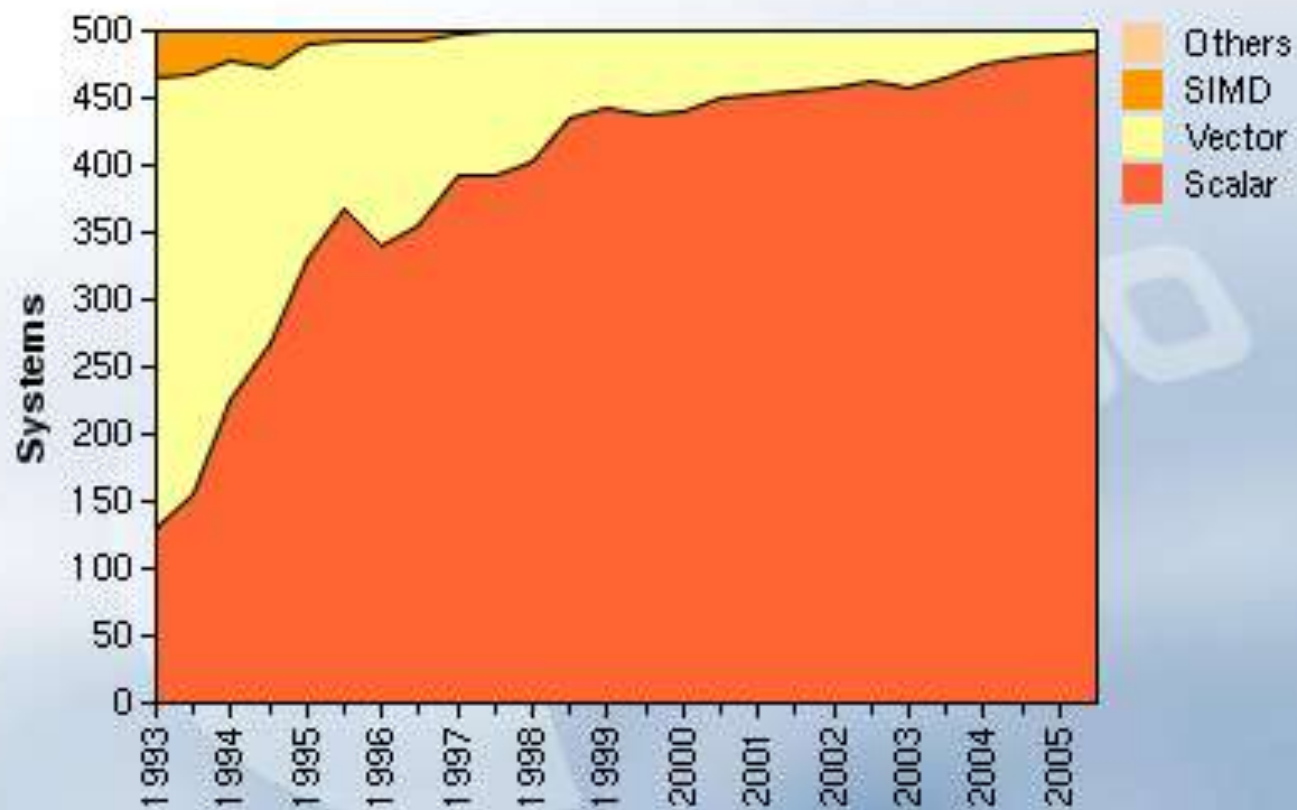
## Operating System Family / Performance

November 2005

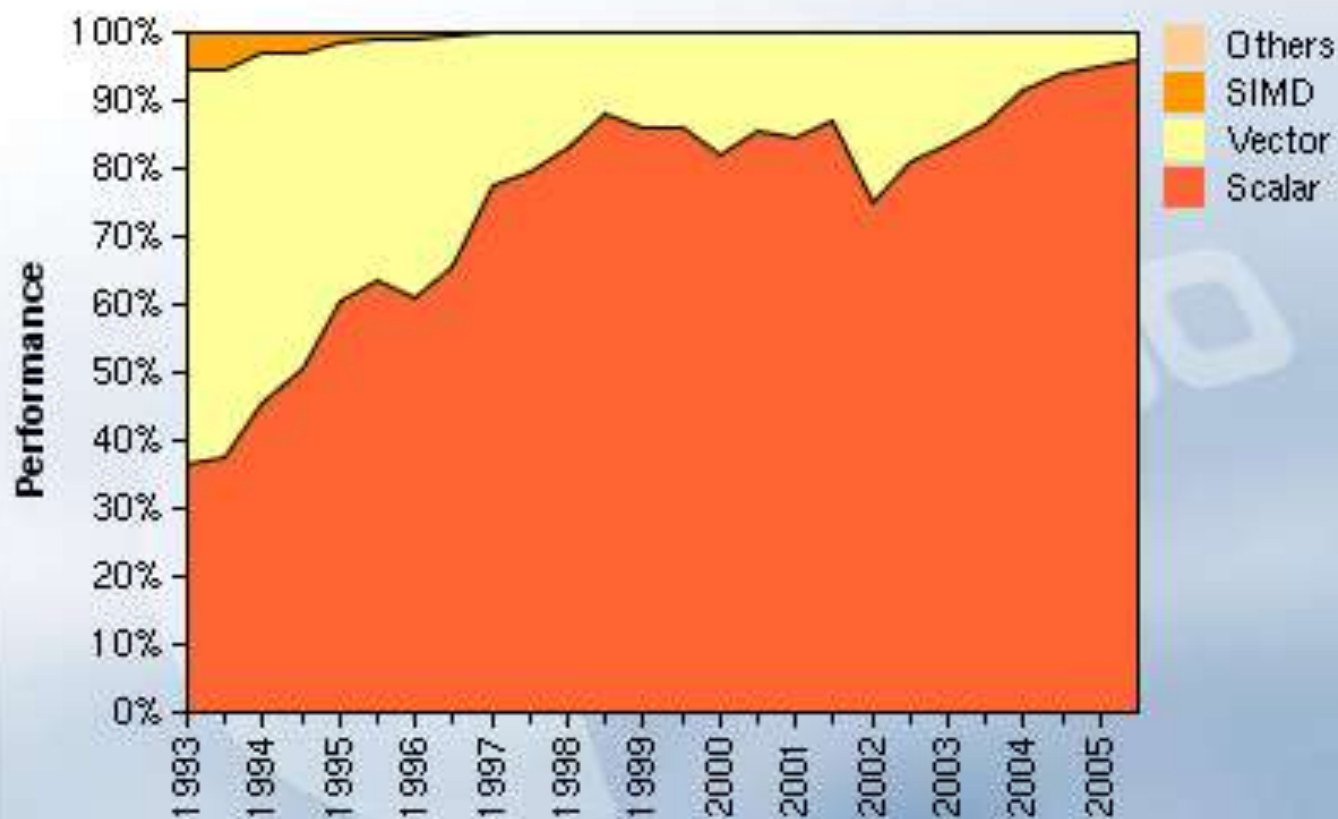




## Processor Architecture / Systems



## Processor Architecture / Performance



## Processor Architecture / Systems

November 2005



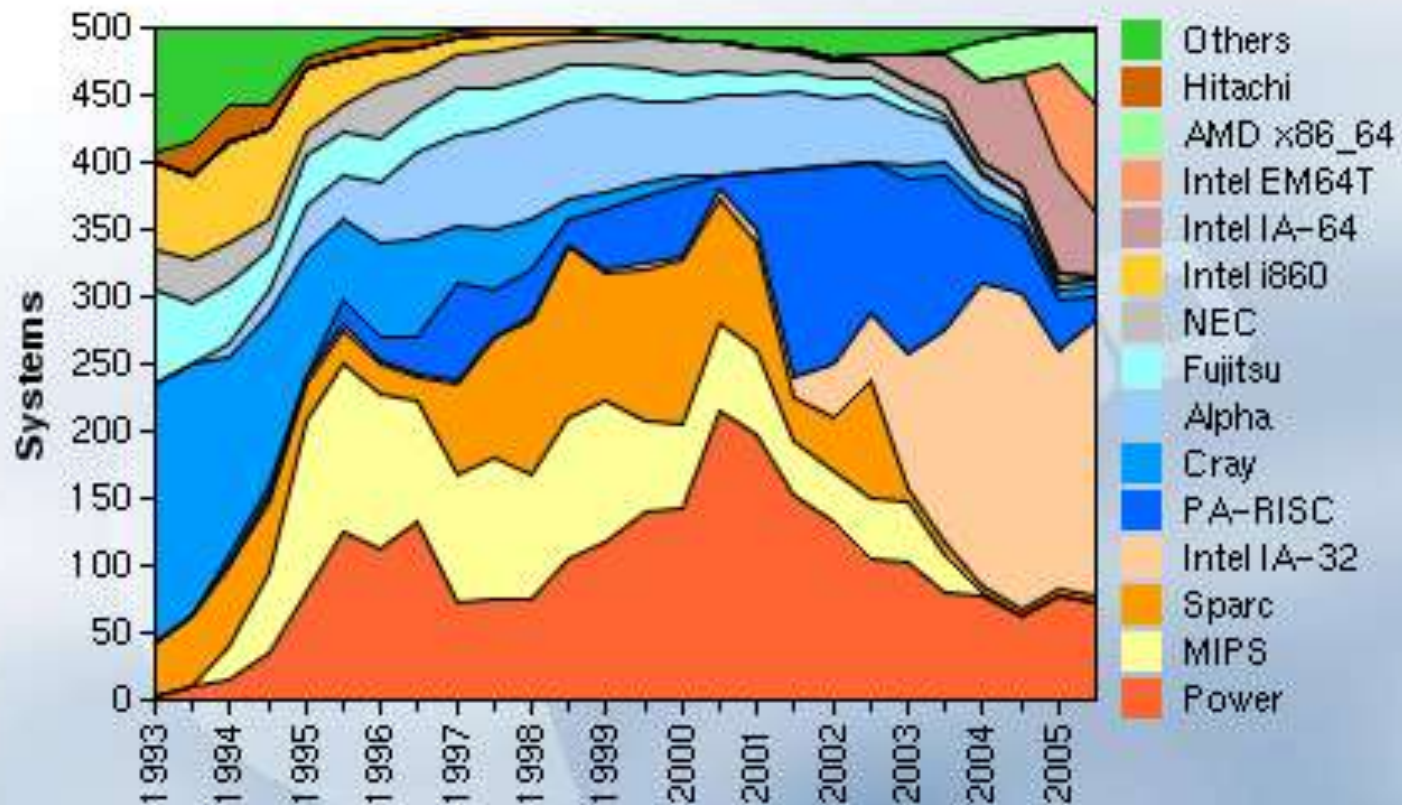
## Processor Architecture / Performance

November 2005

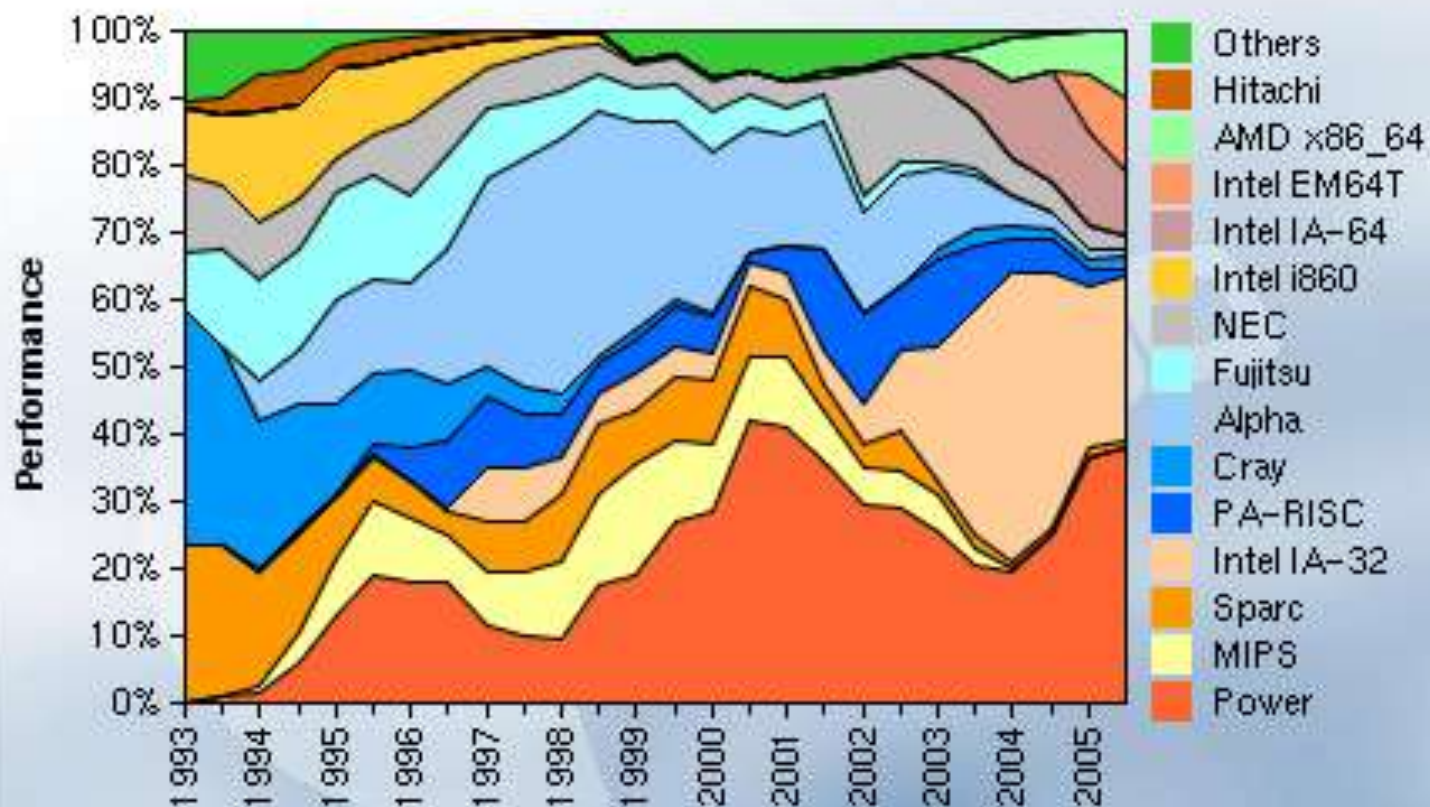




## Processor Family / Systems

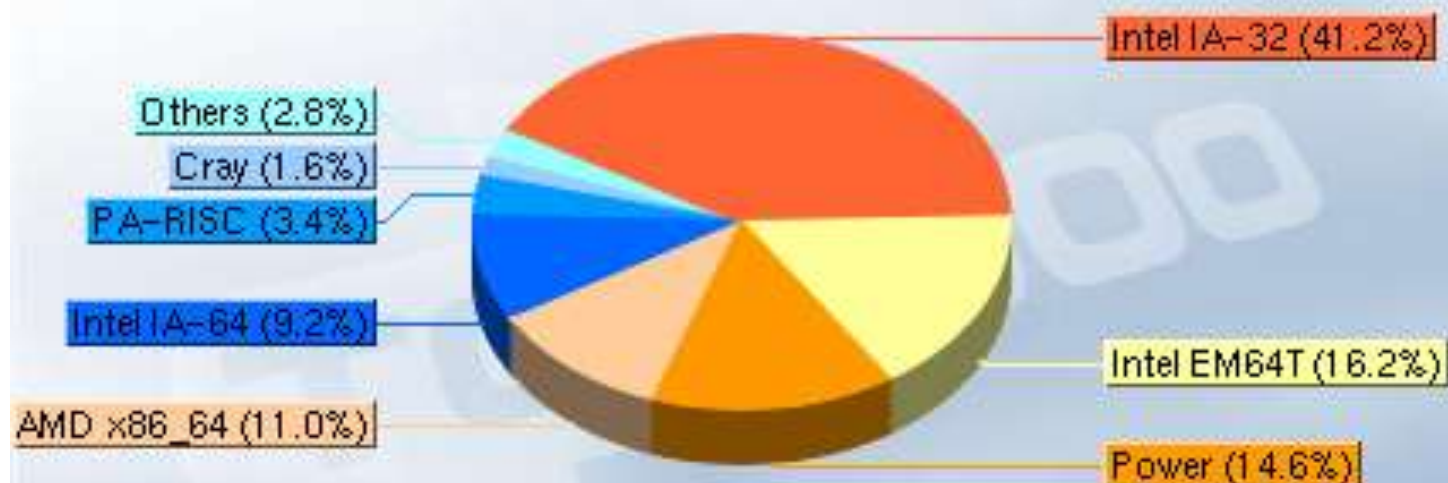


## Processor Family / Performance



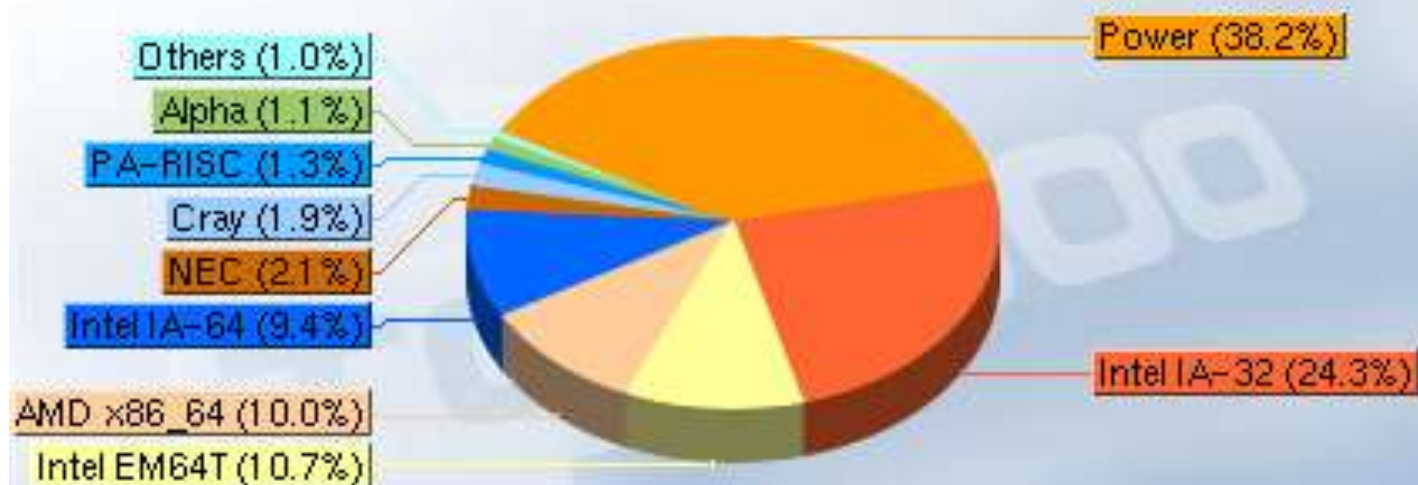
## Processor Family / Systems

November 2005



## Processor Family / Performance

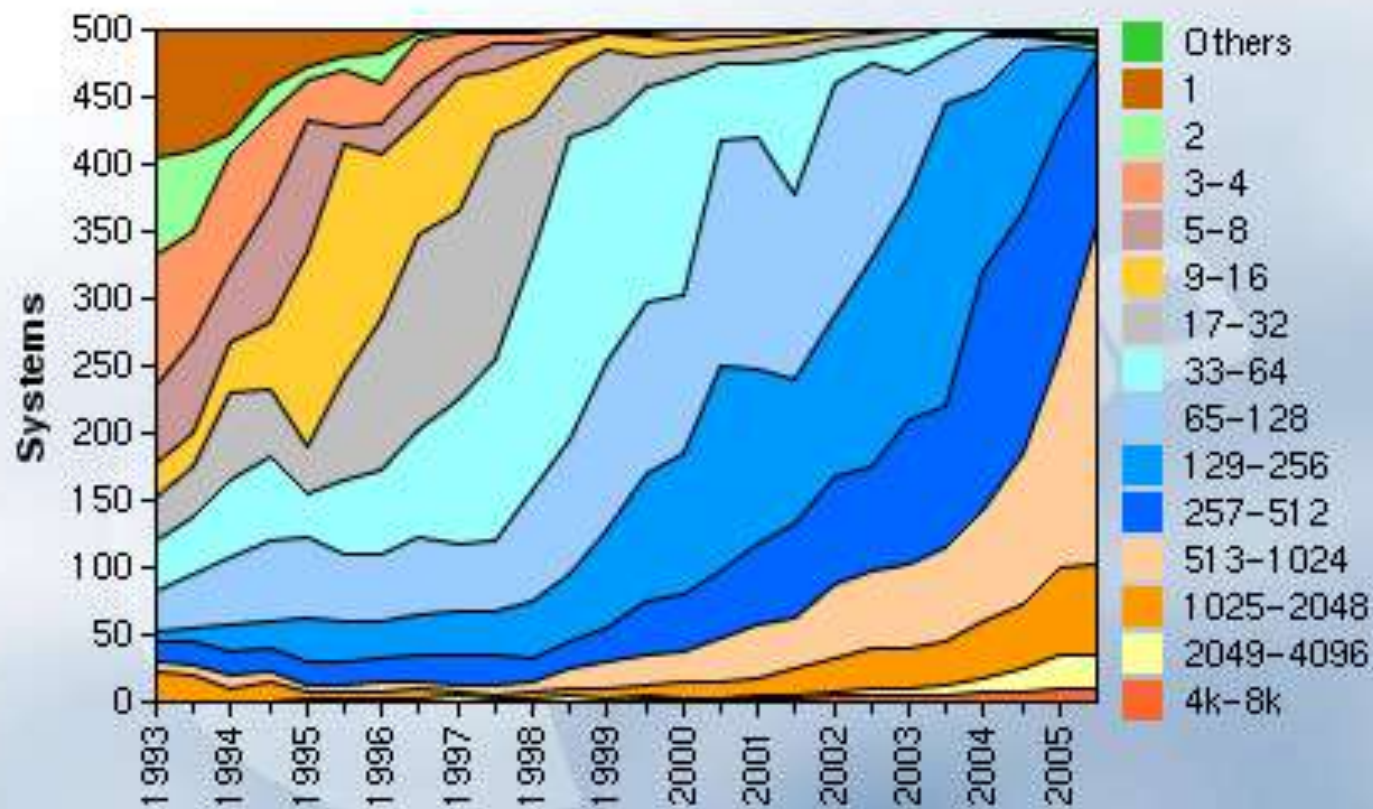
November 2005







## System Processor Counts / Systems



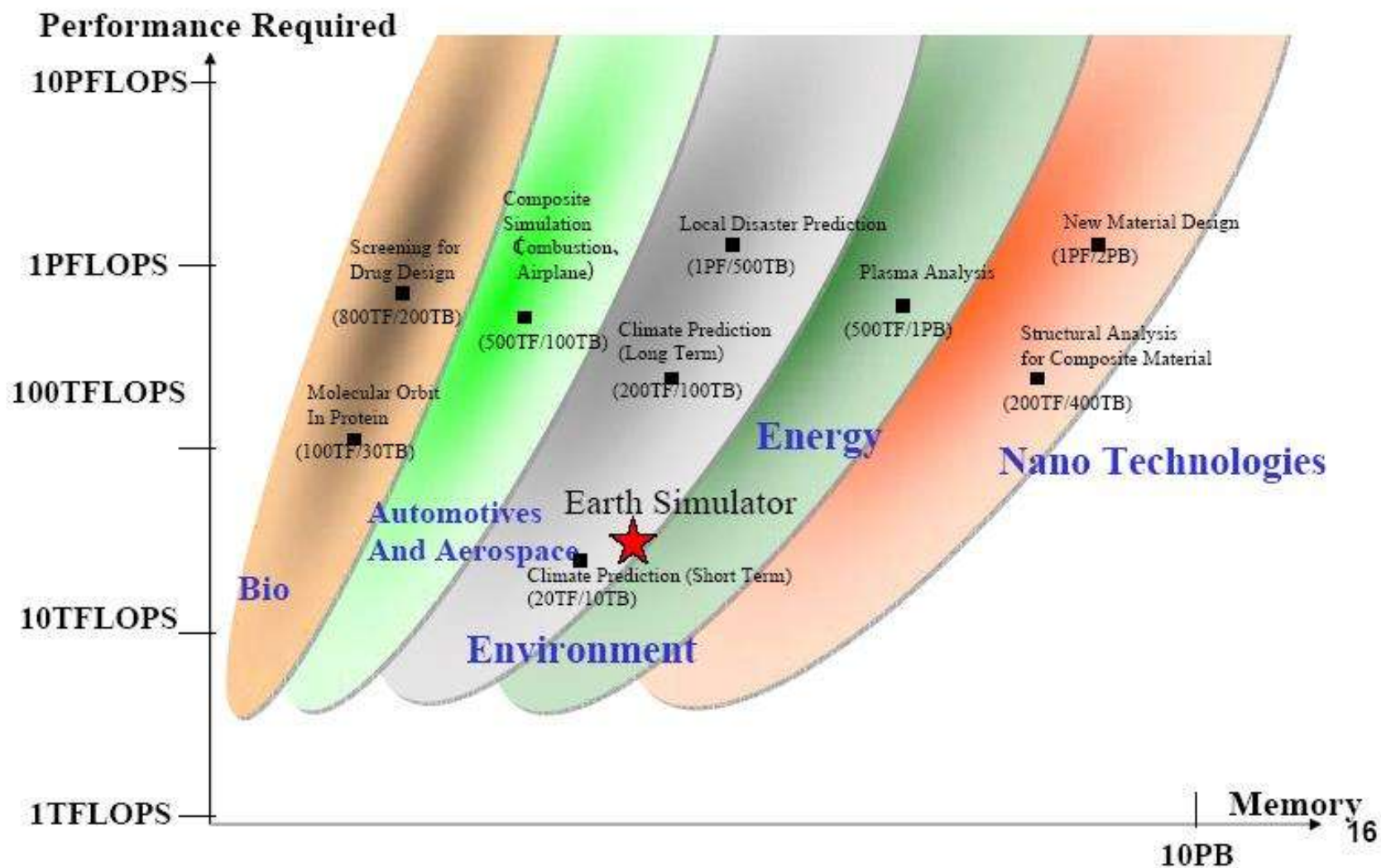
## System Processor Counts / Systems

November 2005



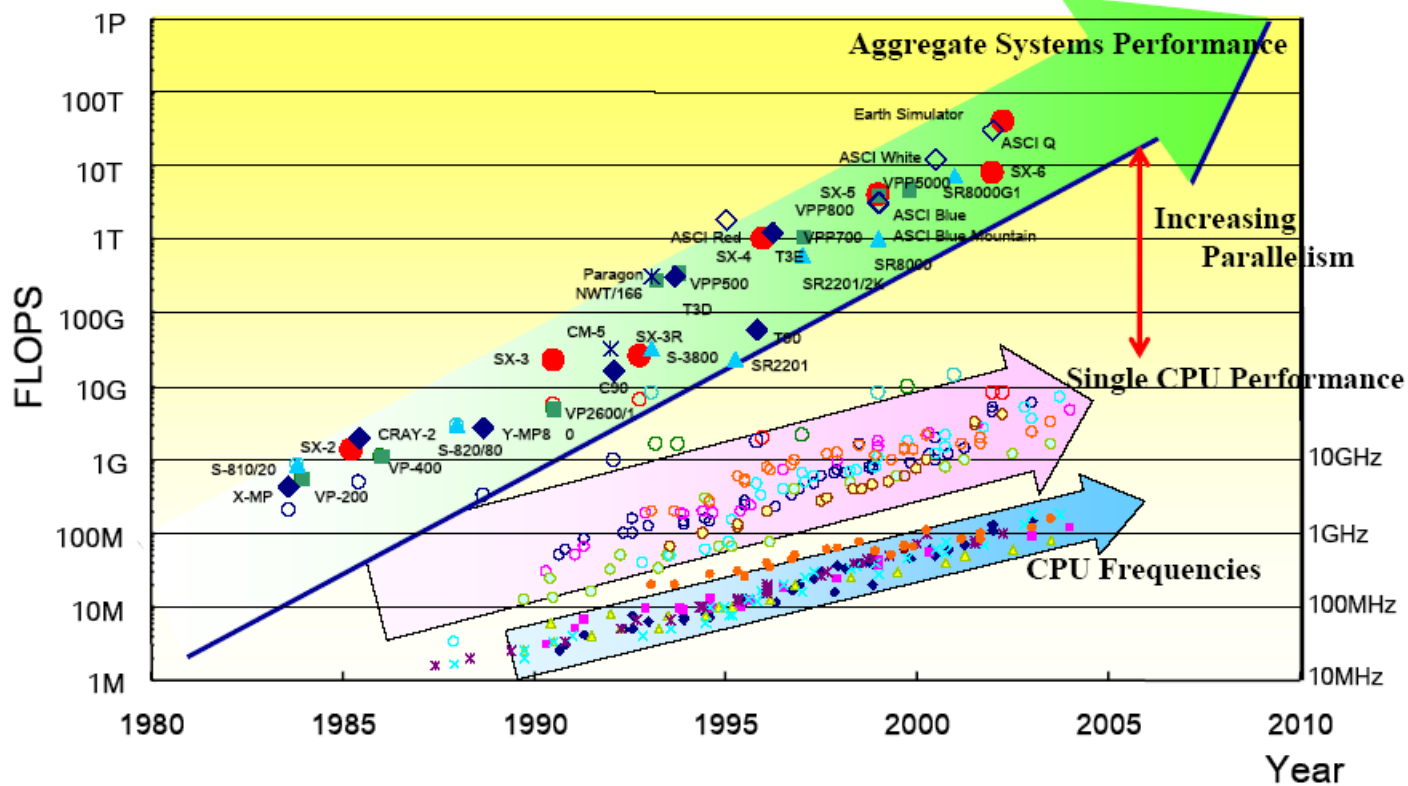
# Brazil in TOP500 (11/2005)

Rank	Site	System Vendor	Processors	Rmax	Rpeak
120	Petroleum Company (C) Brazil	xSeries Cluster Xeon 3.06 GHz - Gig-E IBM	1024	3755	6266.88
122	PETROBRAS Brazil	<i>bwr1</i> Cluster Platform 3000 DL140G3 Xeon 3.06 GHz GigEthernet Hewlett-Packard	1300	3739	7956
164	PETROBRAS Brazil	<i>bw7</i> Cluster Platform 3000 DL140G3 Xeon 3.06 GHz GigEthernet Hewlett-Packard	1008	2992	6169
381	PETROBRAS Brazil	<i>trindade</i> BladeCenter HS20 Cluster, Xeon 3.2 GHz, GigEthernet IBM	512	1922.56	3276.8





# History of High Performance Computers





# Architecture/Systems Continuum

Loosely  
Coupled



Tightly  
Coupled

- ♦ **Commodity processor with commodity interconnect**

- **Clusters**
  - Pentium, Itanium, Opteron, Alpha, PowerPC
  - GigE, Infiniband, Myrinet, Quadrics, SCI
- **NEC TX7**
- **HP Alpha**
- **Bull NovaScale 5160**

- ♦ **Commodity processor with custom interconnect**

- **SGI Altix**
  - Intel Itanium 2
- **Cray Red Storm**
  - AMD Opteron
- **IBM Blue Gene/L (?)**
  - IBM Power PC

- ♦ **Custom processor with custom interconnect**

- **Cray X1**
- **NEC SX-7**
- **IBM Regatta**



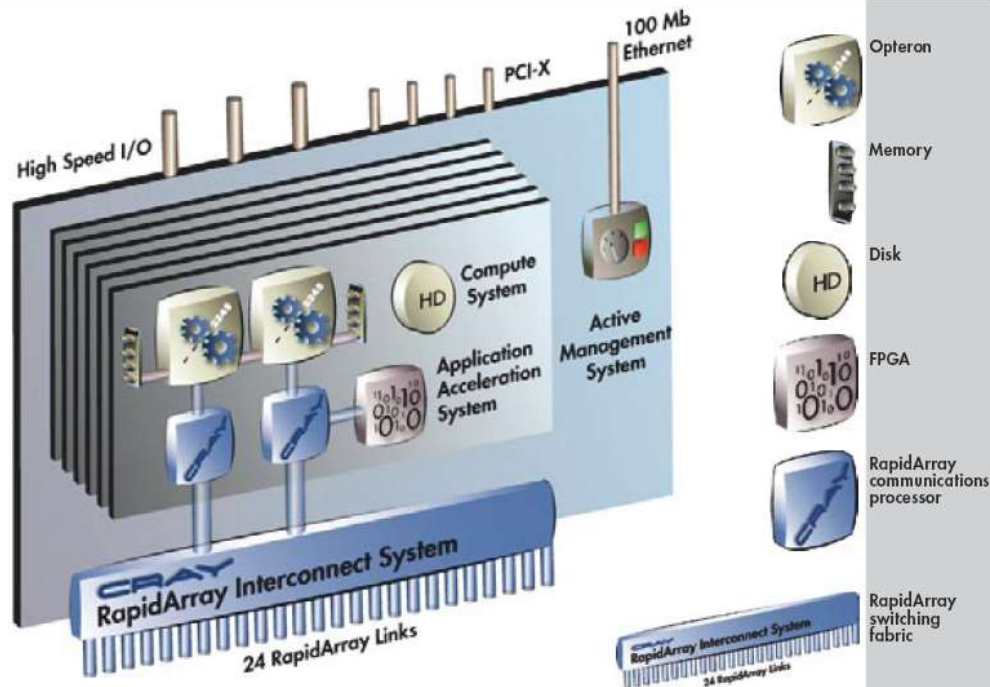
# Commodity Processors

---

- ♦ **AMD Opteron**
  - 2 GHz, 4 Gflop/s peak
- ♦ **HP Alpha EV68**
  - 1.25 GHz, 2.5 Gflop/s peak
- ♦ **HP PA RISC**
- ♦ **IBM PowerPC**
  - 2 GHz, 8 Gflop/s peak
- ♦ **Intel Itanium 2**
  - 1.5 GHz, 6 Gflop/s peak
- ♦ **Intel Pentium Xeon, Pentium EM64T**
  - 3.2 GHz, 6.4 Gflop/s peak
- ♦ **MIPS R16000**
- ♦ **Sun UltraSPARC IV**

# O Cray XD1

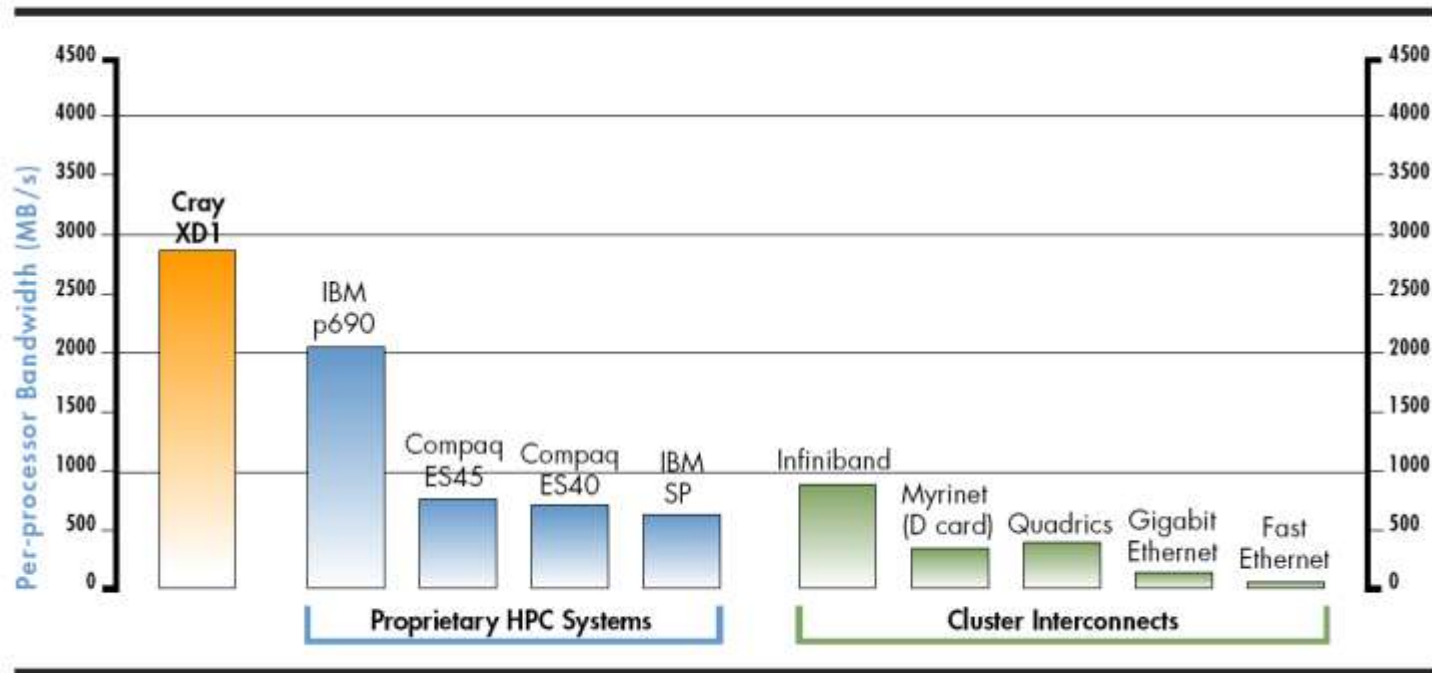
Figure 1: Cray XD1 system architecture

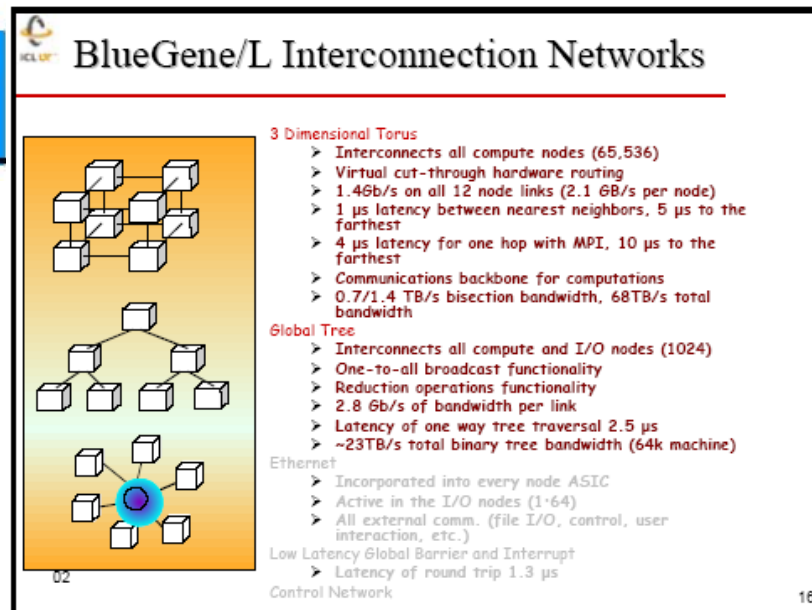
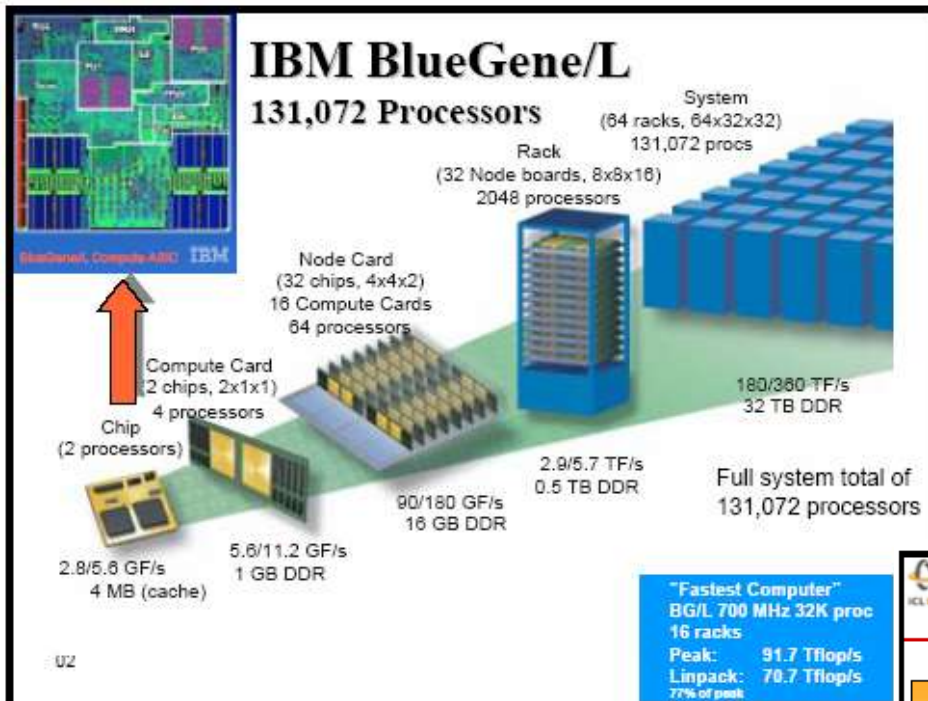


<sup>2</sup> Shared memory systems may be subdivided into SMPs and cache-coherent non-uniform memory access (ccNUMA) systems, depending upon whether or not latency to memory is uniform across all processors. The distinction is not relevant to this paper, and comments made about SMPs are equally applicable to ccNUMA systems.

# O Cray XD1

Figure 2: MPI bandwidth comparison (larger is better)





Fonte: J. Dongarra, <http://www.netlib.org/utk/people/JackDongarra/talks.html>





## NASA Ames: SGI Altix Columbia 10,240 Processor System

- ♦ Architecture: Hybrid Technical Server Cluster
- ♦ Vendor: SGI based on Altix systems
- ♦ Deployment: Today
- ♦ Node:
  - 1.5 GHz Itanium-2 Processor
  - 512 procs/node (20 cabinets)
  - Dual FPU's / processor
- ♦ System:
  - 20 Altix NUMA systems @ 512 procs/node = 10240 procs
  - 320 cabinets (estimate 16 per node)
  - Peak: 61.4 Tflop/s ; LINPACK: 52 Tflop/s
- ♦ Interconnect:
  - FastNumaFlex (custom hypercube) within node
  - Infiniband between nodes
- ♦ Pluses:
  - Large and powerful DSM nodes
- ♦ Potential problems (Gotchas):
  - Power consumption - 100 kw per node (2 Mw total)





# Projeto de computadores

## Integração Crescente X Limite Físico

- performance → velocidade do processador
- acesso à memória mais lento que velocidade do processador
- níveis hierárquicos de memória
- pipeline de instruções e de unidades funcionais
- conjunto de instruções reduzidos – RISC X CISC
- paralelismo on-chip – realiza várias operações por ciclo



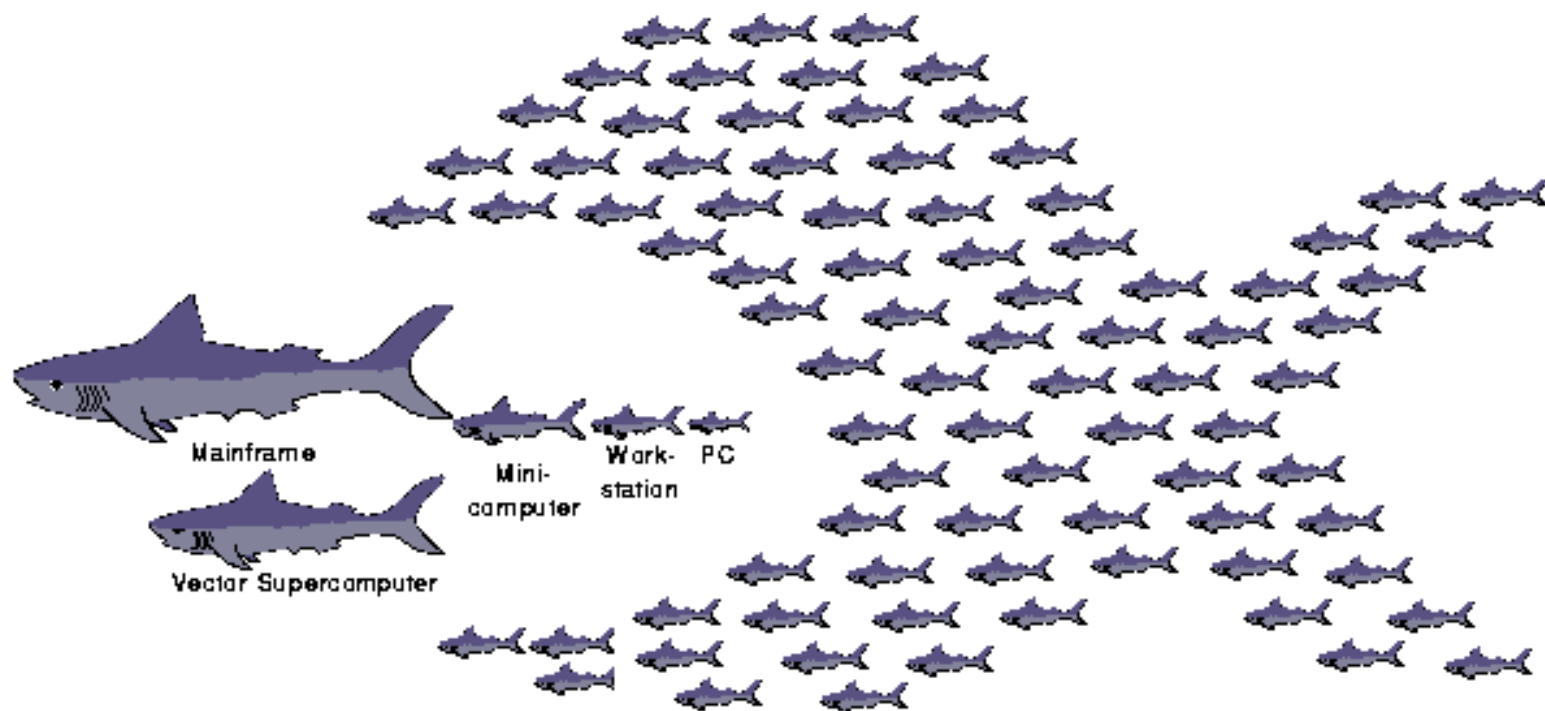


# Ementa

---

- Introdução
- Tópicos Básicos em arquitetura de Computadores
- Arquitetura de processadores e memória
- Medidas de desempenho e análise
- **Clusters**
- Comentários Finais

# Cluster



# Tipos de Clusters

- Basicamente existem três tipos de clusters:

- Tolerante à falhas

- Consiste de pelo menos dois computadores conectados em rede com um software de monitoração entre os dois, quando um falhar o outro assume o trabalho

- Balanceamento de carga

- Utilizado por servidores *web*, o cluster verifica qual nó está menos carregada e envia o pedido para ela

- Computação de alto desempenho

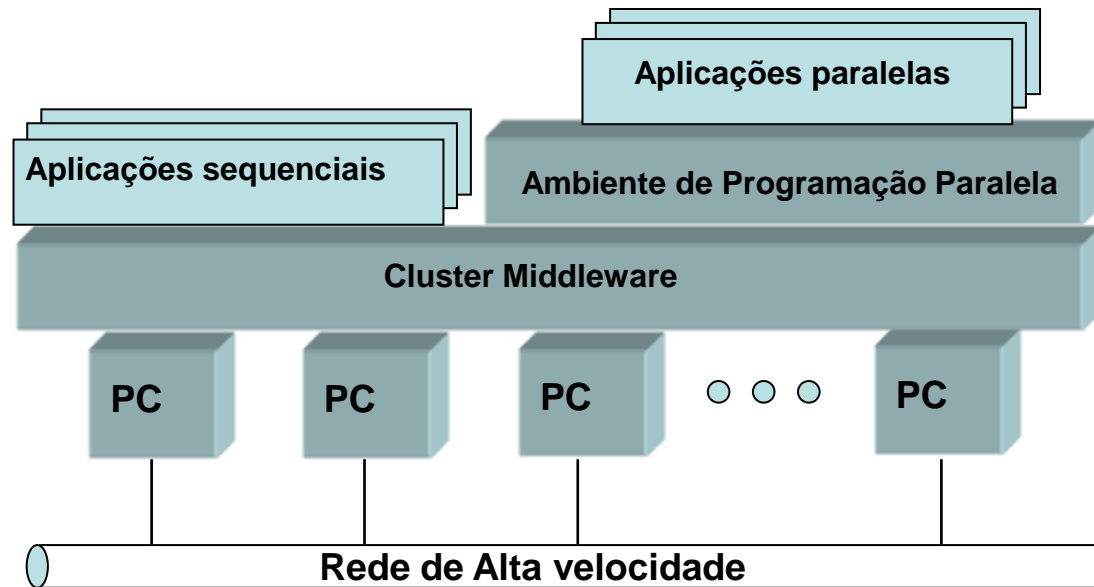
- Os nós são configurados para oferecer o maior desempenho possível



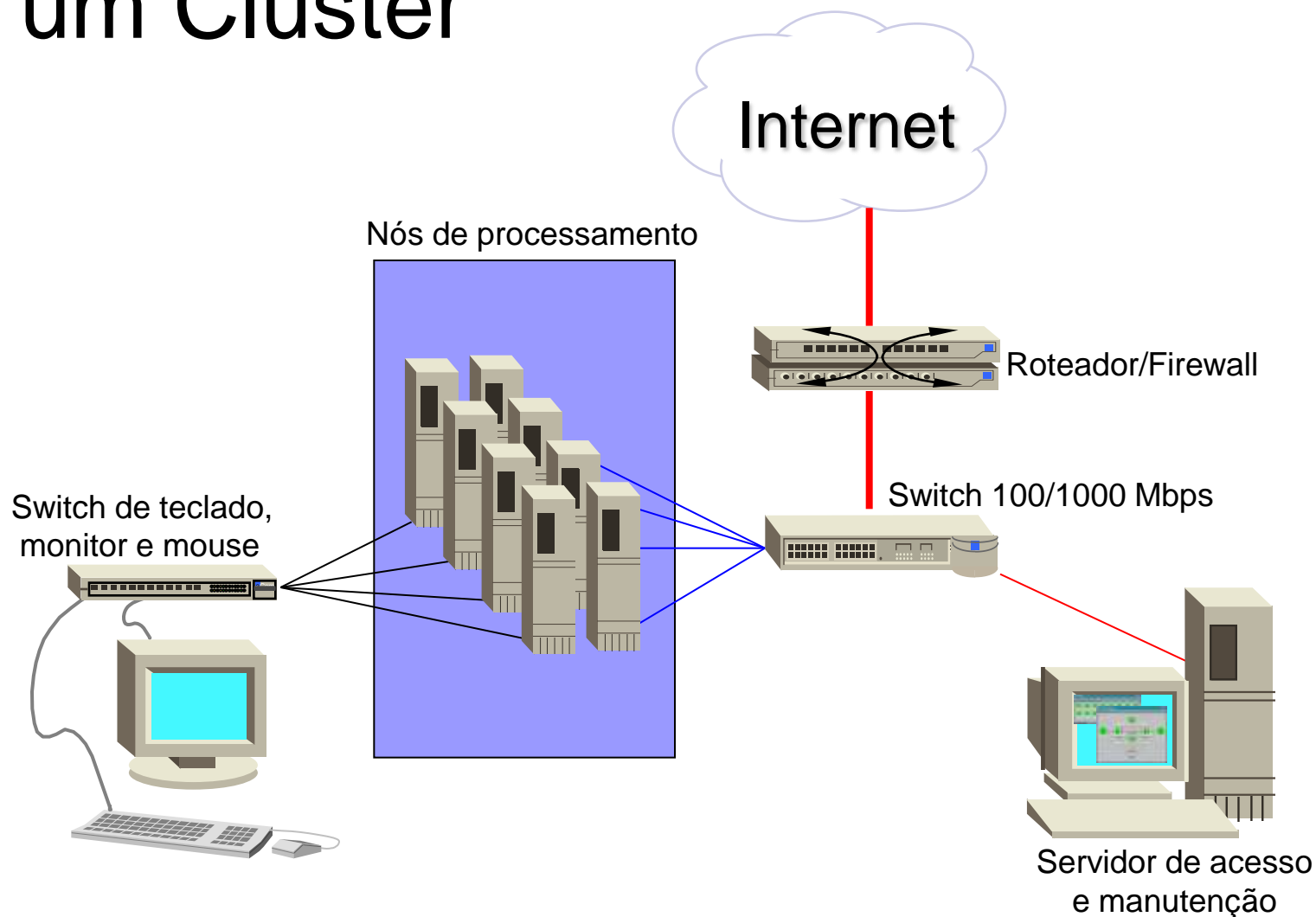
# Clusters

- Os supercomputadores tradicionais foram construídos por um pequeno número de fabricantes, com um alto orçamento destinado ao projeto
- Muitos clientes, como universidades, não podem arcar com os custos de um supercomputador, então o uso de clusters se torna uma alternativa interessante
- Com o uso de hardware mais barato e disponível no mercado, sistemas com desempenho similar aos supercomputadores podem ser construídos

# Arquitetura de um Cluster



# Exemplo de uma topologia física de um Cluster





# Maiores desafios

- Escalabilidade (física e de aplicação)
- Disponibilidade (gerenciamento de falhas)
- Imagem única do sistema (parece ao usuário como se fosse um único sistema)
- Comunicação rápida (redes e protocolos de comunicação)
- Balanceamento de carga (CPU, rede, memória, discos)
- Gerenciabilidade (administração e controle)
- Aplicabilidade (aplicações voltadas para o cluster)
- Segurança e encriptação (grid computing)



# Cluster Middleware

- Sistemas operacionais
  - Solaris MC
  - OpenMosix
  - Rocks
  - Sistema Operacional + OSCAR
- Gerenciamento de recursos e despacho
  - PBS
  - NQS
  - SGE
  - Condor
  - LSF





# Ambientes de programação

- Threads (PCs, SMPs, ...)
  - POSIX Threads
  - Java Threads
- MPI
- PVM
- Software DSMs (SHMEM da SGI)



# Cluster - Beowulf

## Definição:



- PC padrão nos nós
  - Pentium, Alpha, PowerPC, SMP
- Interconexão LAN/SAN
  - Ethernet, Myrinet, Giganet, ATM
- SO Unix
  - Linux, BSD
- Sistema de Troca de Mensagens
  - MPI, PVM
  - HPF



## Vantagens:

- Melhor relação preço-desempenho
- Menor custo inicial
- Configuração ajustável
- Invulnerável ao fornecedor
- Escalável
- Incorporação rápida de novas tecnologias



Facilitado pelo hardware, redes e sistemas operacionais de PC's padrão, alcançando a capacidade de estações de trabalho científicas a uma fração do preço e a disponibilidade de sistemas de troca de mensagens padronizados.



# O uso de Clusters

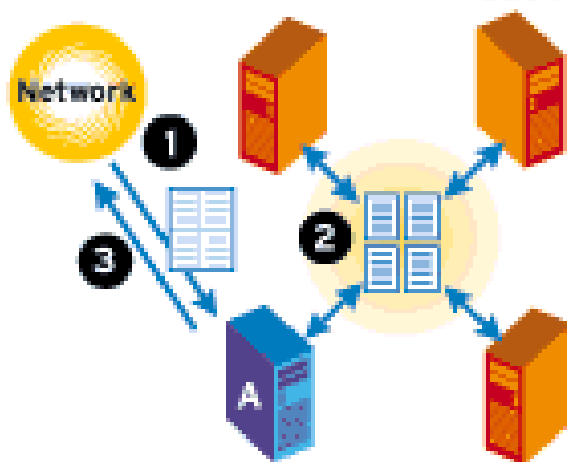
- Poder computacional com custo aceitável para aplicações científicas:
  - Restrições de tempo (aplicações de previsão de tempo)
  - Velocidade (aplicações de busca: Google – 15000 PCs)
  - Quantidade de memória (terabytes)
- Tolerância a falhas  $\Rightarrow$  Garantia de disponibilidade de serviço em caso de falha de um componente

# Beowulf

## ➤ “Do-it-yourself Supercomputing”

Conjunto de PCs interconectados por uma rede local de baixo custo rodando um sistema operacional Unix de código aberto e executando aplicações paralelas programadas a partir de uma biblioteca ou modelo de troca de mensagem

*How to Build a Beowulf*



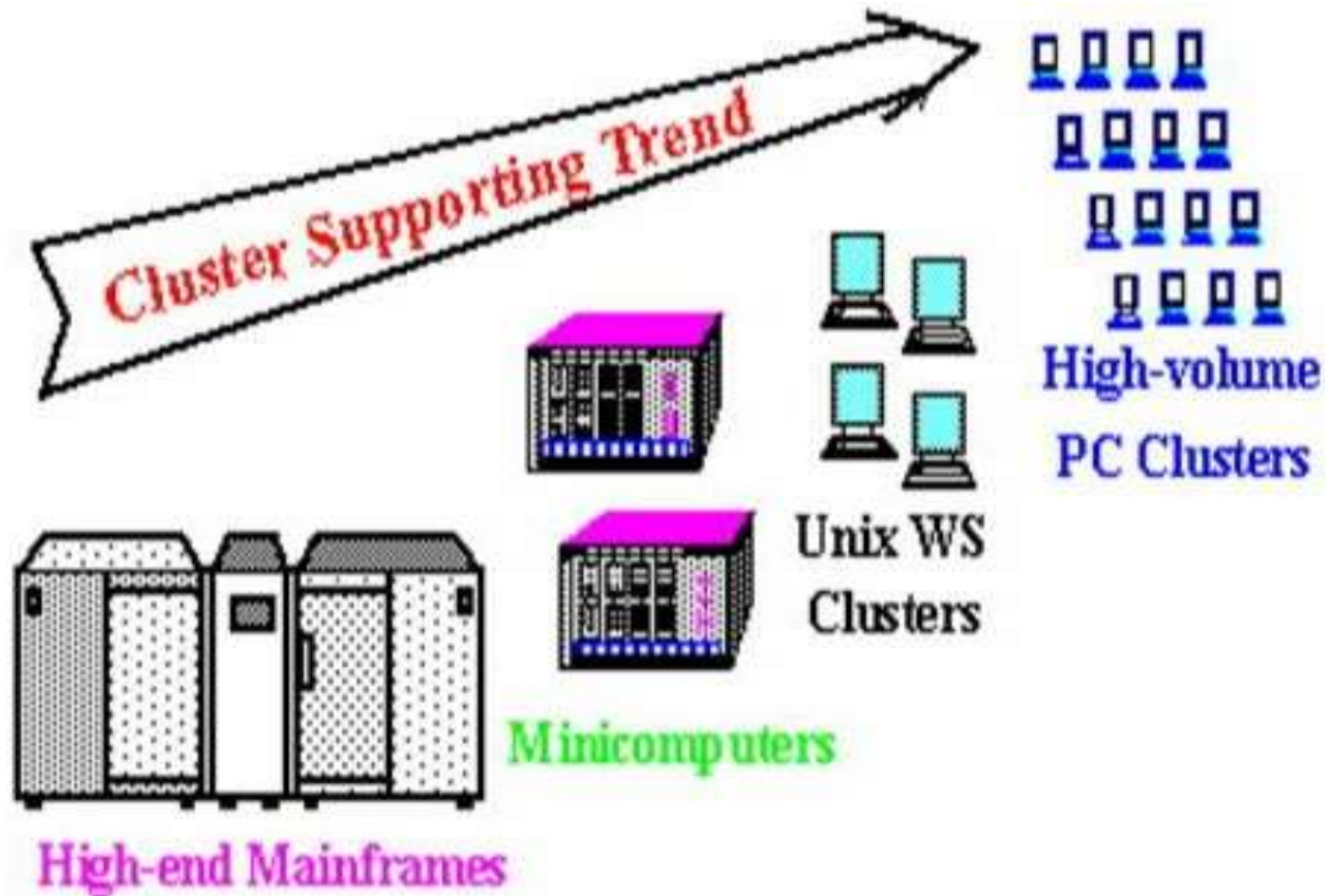
# Beowulf

- PCs M<sup>2</sup>COTS  $\Rightarrow$  mass-market commodity off the shelf
  - Economia de escala
- Redes de Interconexão comerciais
  - Ethernet, Fast Ethernet, Gigabit, Myrinet
- Sistema Operacional Unix de código aberto:
  - Sistemas Unix têm sido a escolha da comunidade científica desde estações de trabalho até supercomputadores
  - Uma cópia do sistema em cada nó  $\Rightarrow$  baixo custo ou custo zero  
 $\Rightarrow$  Linux ou FreeBSD
  - Especificação IEEE POSIX

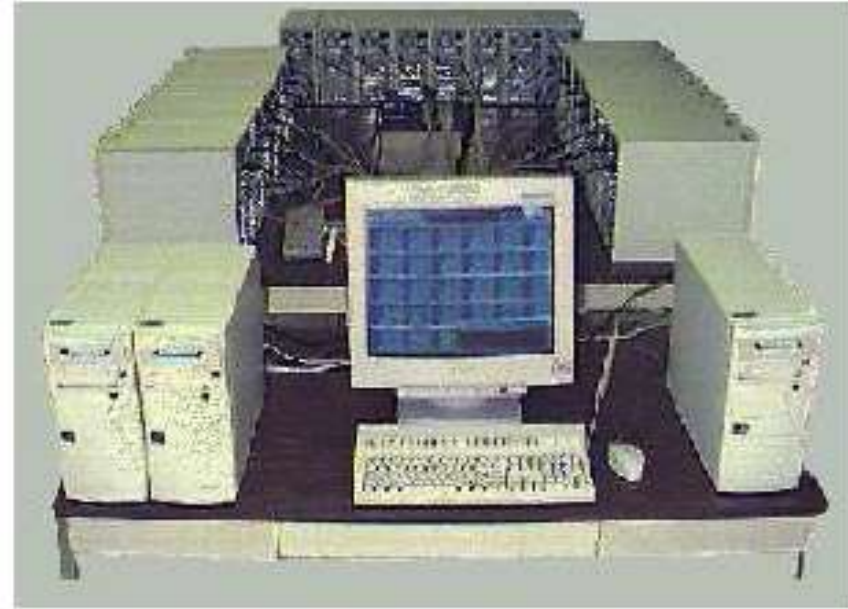
# Beowulf

- Aplicações devem executar mais rápido do que em estações de trabalho e a menor custo do que em supercomputadores MPP (massive parallel processors)
  - MPP – Massive Parallel Processors  $\Rightarrow$  sistemas proprietários de alto custo e baixa taxa preço/performance
  - O aproveitamento da concorrência do sistema leva a definição de uma estrutura lógica para o uso da camada de comunicação
  - Interface lógica entre o programador da aplicação e os recursos paralelos do sistema
  - Troca de mensagem suporta uma estrutura de comunicação de processos seqüenciais
  - PVM, MPI, OpenMP

# Beowulf



# Beowulf







# Beowulf

## ■ Vantagens:

- ☐ Não é suscetível a variações de mercado
- ☐ Não possui arquitetura específica
- ☐ Sistema escalável
- ☐ Mesmo paradigma de programação científica e comercial

## ■ Desvantagens:

- ☐ Execução limitada pelo servidor

# Beowulf

- Componentes do nó de administração:
  - Sistema computacional stand-alone com hardware completo e suporte de um sistema operacional para gerenciamento
  - Processador
  - Memória
  - Disco rígido compartilhado pelos nós de processamento
  - Rede
  - Outros dispositivos: CD-ROM, Floppy

# Beowulf

- Componentes do nó de processamento:
  - Sistema computacional stand-alone com hardware completo ou não (sem vídeo, sem teclado e sem mouse) e suporte de um sistema operacional para gerenciamento completo da execução de aplicativos em cada nó
  - Processador
  - Memória
  - Disco rígido
  - Rede
  - Outros dispositivos: CD-ROM, floppy



# Beowulf

- Rede de comunicação

- Hardware

- Interfaces: Hosts, Links e Dispositivos
    - Topologia

- Software

- Protocolos: Ethernet, IP, TCP, GM
    - Drivers

- Desempenho

- Hardware
    - Software

# Taxonomia das Aplicações Paralelas

*Pouca ou Nenhuma  
Comunicação*

*Embaraçosamente Paralela  
(EP)*

*Comunicação  
Explícita (Vizinhos)*

*Explícita  
Estruturada  
(EE)*

*Explícita  
Não-Estruturada  
(ENE)*

*Comunicação  
Implícita (Global)*

*Implícita  
Estruturada  
(IE)*

*Implícita  
Não-Estruturada  
(INE)*

*Comunicação  
Estruturada*

*Comunicação  
Não-Estruturada*

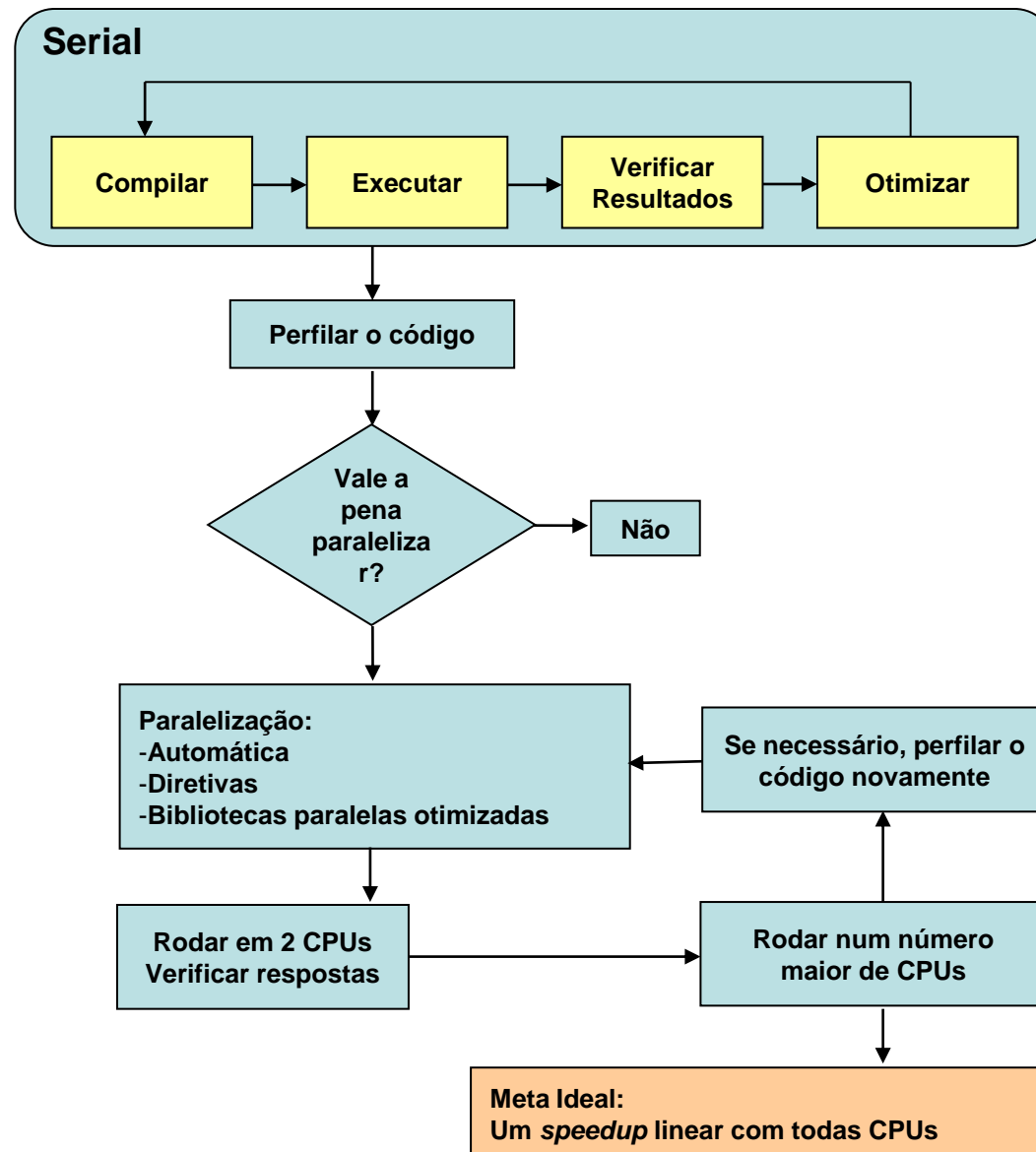


# Otimização e Programação

## Antes de paralelizar:

- Otimize o código para uma classe de processadores (vetoriais, RISC, Intel, etc )
- Responsável pela redução do tempo de CPU
- Use TODAS as ferramentas de otimização de código existentes em seu sistema
- Verifique se você está obtendo as respostas corretas
- Procure usar sempre que possível padronização, BLAS, LAPACK etc

# Algoritmo de como paralelizar um programa



# Ferramentas de Desenvolvimento

## Bibliotecas Numéricas

- Netlib: <http://www.netlib.org>
- ACTS (Advanced Computational Testing and Simulation) Toolkit <http://acts.nersc.gov/>
  - PETSc (Portable, Extensible Toolkit for Scientific Computation)
  - ScaLAPACK library extends LAPACK's high-performance linear algebra software to distributed memory





# Ementa

---

- Introdução
- Tópicos Básicos em arquitetura de Computadores
- Arquitetura de processadores e memória
- Medidas de desempenho e análise
- Clusters
- Comentários Finais

# Centros de Supercomputação no Brasil



SINAPAD



## **Centro Nacional de Supercomputação na Região Sul**

<http://www.cesup.ufrgs.br>

[super@cesup.ufrgs.br](mailto:super@cesup.ufrgs.br)

### **CENAPAD-SP**

Centro Nacional de Processamento de Alto Desempenho em São Paulo

<http://www.cenapad.unicamp.br>

[cenapadsp@cenapad.unicamp.br](mailto:cenapadsp@cenapad.unicamp.br)

### **CENAPAD-NE**

Centro Nacional de Processamento de Alto Desempenho no Nordeste

<http://www.cenapadne.br>

[cenapad@cenapadne.br](mailto:cenapad@cenapadne.br)

### **CENAPAD-RJ**

Centro Nacional de Processamento de Alto Desempenho no Rio de Janeiro

LNCC - Laboratório Nacional de Computação Científica

<http://www.lncc.br>

[cadastro@lncc.br](mailto:cadastro@lncc.br)

### **CENAPAD-MGCO**

Centro Nacional de Processamento de Alto Desempenho em Minas Gerais e Centro Oeste

<http://www.cenapad.ufmg.br>

[osvaldo@cenapad.ufmg.br](mailto:osvaldo@cenapad.ufmg.br)

### **CENAPAD-AMB**

Centro Nacional de Processamento de Alto Desempenho Ambiental

Instituto Nacional de Pesquisas Espaciais - INPE

<http://www.cptec.inpe.br>

[benicio@cptec.inpe.br](mailto:benicio@cptec.inpe.br)

### **NACAD**

Núcleo de Atendimento em Computação de Alto Desempenho-COPPE

<http://www.nacad.ufrj.br>

[nacad@nacad.ufrj.br](mailto:nacad@nacad.ufrj.br)

# Cluster de baixo custo montado no NACAD

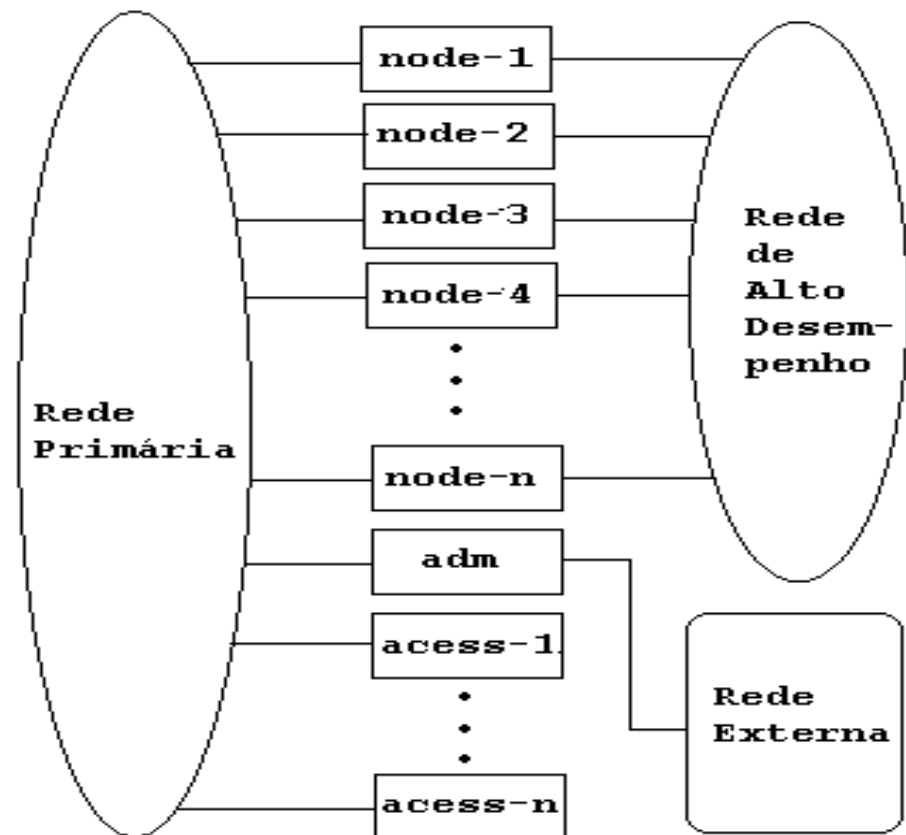


- 8 CPUs Intel Pentium III de 1GHz
- Memória por CPU: 512MB RAM e cache de 256KB
- Memória Total: 4.0 Gbytes RAM (distribuída)
- Rede: dedicada com tecnologia Fast-Ethernet (100 MBits/s)
- Sistema Operacional: Linux distribuição RedHat
- Compiladores: Fortran-77, Fortran-90 e C/C++
- Ferramentas para desenvolvimento de aplicações paralelas
- Sistema de fila PBS (Portable Batch System)

# Arquitetura do Cluster Mercury (NACAD)

## ➤ Cluster Inforserver Itaotec

- 16 nós duais de processamento (32 processadores)
- Pentium III - 1 GHz
- 512MB de memória principal por nó - 256KB cache



# SGI Altix 350

- 14 CPUs Intel Itanium2:  
palavra de 64 bits
- Pico Teórico de Performance:  
6 GFlop/s por CPU
- Memória: 28 Gbytes RAM  
(compartilhada – cc-NUMA)  
SGI NUMALink
- Sistema operacional: RedHat  
Enterprise Linux + SGI  
ProPack
- Compiladores: Intel e GNU  
(Fortran-90 e C/C++) com  
suporte OpenMP e MPI
- Sistema de Fila PBS-Pro



# Tendências

## ■ Grid Computing

