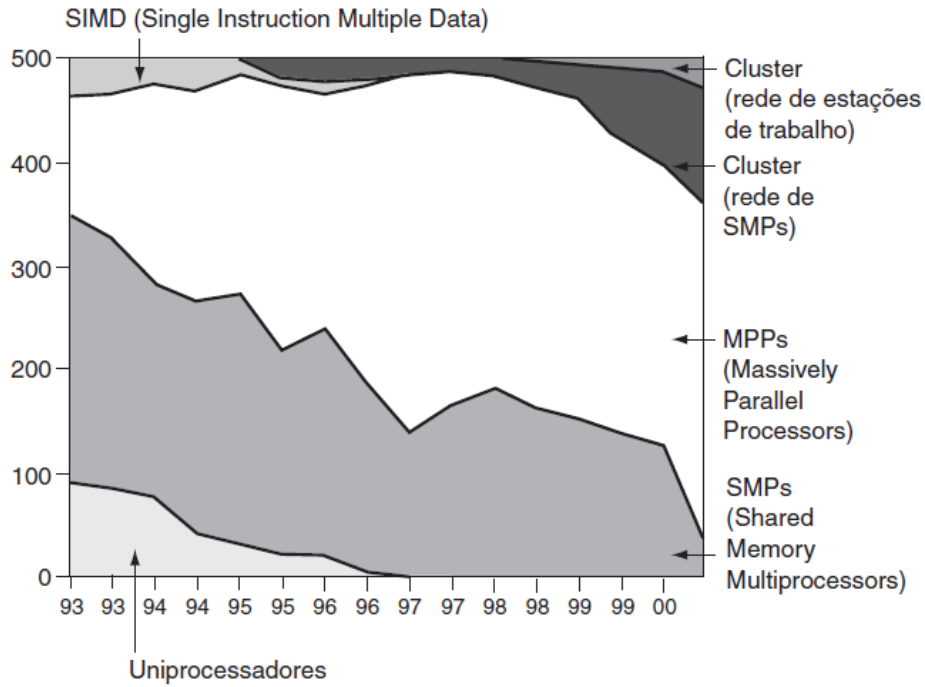
CAPÍTULO 6 MULTIPROCESSADORES E CLUSTERS

- Introdução
- Programando multiprocessadores
- O problema do speedup
- Multiprocessadores conectados por um único barramento
- Exemplo de programa paralelo
- Coerência de cache em multiprocessadores
- Sincronização
- Multiprocessadores Conectados por Rede
- Clusters
- Topologia de rede
- Multiprocessadores dentro de um Chip e Multithreading
- O Cluster Google
- Categorias de Paralelismo (Flynn,1966)
- Computadores SIMD
- Computadores Vetoriais
- Computadores MIMD

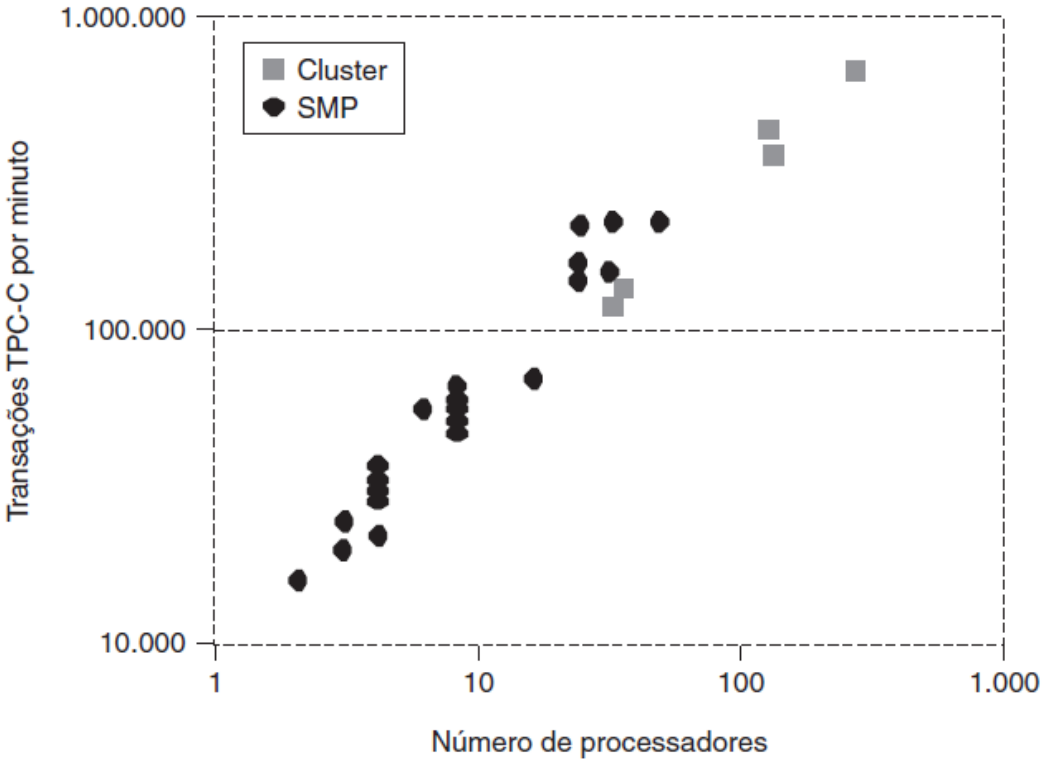
Introdução

- Sistemas que utilizam mais de um processador
  - Desempenho absoluto mais alto
  - Escalabilidade
  - Maior confiabilidade
- É mais econômico construir um cluster de processadores do que construir um uniprocessador potente para um dado workload



Introdução

- O gráfico abaixo mostra o desempenho versus o número de processadores.
- Os clusters possuem melhor desempenho tpmC a um custo mais baixo que os SMP



O TPC-C é um benchmark de processamento de transações online (OLTP). A medida do desempenho é dada em transações por minuto (tpmC)

Introdução

- Algumas preocupações no projeto de multiprocessadores são:
  - Mecanismos de sincronização
  - Lock/unlock de recursos
  - Gerenciamento de memória

- Principais perguntas que norteiam os projetos de multiprocessadores e clusters:
  - Como os processadores paralelos compartilham dados?
  - Como os processadores paralelos se coordenam?
  - Quantos processadores?

Introdução

- Os multiprocessadores de espaço de endereçamento único podem ser de dois tipos:
  - UMA (Uniform Memory Access) ou SMP (Symmetric Multiprocessors).
  - NUMA (Nonuniform Memory Access).
- Uma alternativa é a troca de mensagem

Multiprocessador	Ano de lançamento	SMP ou NUMA	Máximo de processadores	Rede de interconexão	Tempo típico de acesso à memória remota (ns)
Servidores Sun Starfire	1996	SMP	64	Barramentos de endereço múltiplos, switch de dados	500
SGI Origin 3000	1999	NUMA	512	Hipercubo largo	500
Cray T3E	1996	NUMA	2048	Toro 3D de duas vias	300
Série HP V	1998	SMP	32	Crossbar 8 × 8	1000
Compaq AlphaServer GS	1999	SMP	32	Barramentos comutados	400
Sun V880	2002	SMP	8	Barramentos comutados	240
HP Superdome 9000	2003	SMP	64	Barramentos comutados	275

Figura 9.1.3 Tempos de acesso remoto típicos para recuperar uma word de uma memória remota em multiprocessadores de memória compartilhada.

Programando multiprocessadores

- Nem sempre os ganhos dos multiprocessadores são aproveitados
- Poucos programas têm sido reescritos para executar mais rapidamente em multiprocessadores.
- A programação dos multiprocessadores é consideravelmente mais difícil
- Um dos problemas é o overhead de comunicação
- Em multiprocessadores é necessário conhecer melhor o hardware. O compilador não faz tudo sozinho e transparentemente.

O problema do speedup

- Suponha que você queira alcançar speedup linear com 100 processadores. Que fração da computação original pode ser seqüencial?

$$\text{Tempo de execução após melhoria} = \frac{\text{Tempo de execução afetado pela melhoria}}{\text{Quantidade de melhoria}} + \text{Tempo de execução não afetado}$$

$$\frac{\text{Tempo de execução antes da melhoria}}{100} = \frac{\text{Tempo de execução afetado pela melhoria}}{100} + \text{Tempo de execução não afetado}$$

Já que

$$\text{Tempo de execução antes da melhoria} = \text{Tempo de execução afetado pela melhoria} + \text{Tempo de execução não afetado}$$

$$\frac{\text{Tempo de execução afetado pela melhoria} + \text{Tempo de execução não afetado}}{100} = \frac{\text{Tempo de execução afetado pela melhoria}}{100} + \text{Tempo de execução não afetado}$$

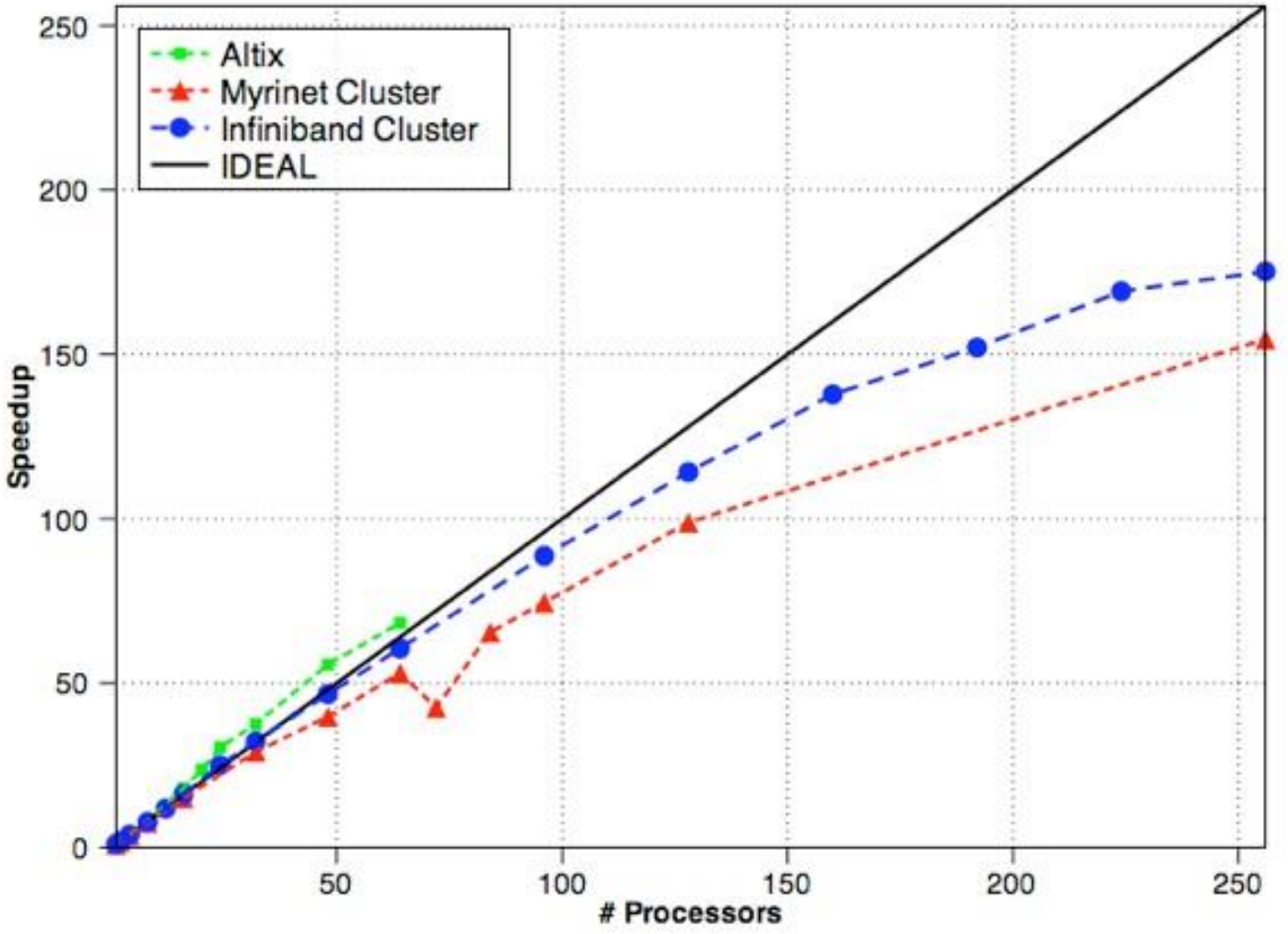
Simplificando, temos:

$$\frac{\text{Tempo de execução não afetado pela melhoria}}{100} = \text{Tempo de execução não afetado}$$

$$\text{Tempo de execução não afetado} = 0$$

O problema do speedup

- Exemplo de resultado prático com multiprocessamento:



Multiprocessadores conectados por um único barramento

- Sistema com vários processadores interconectados por um barramento e uma memória única
- As caches podem reduzir o tráfego de barramento e também o tempo de acessso a dados.
- Existem mecanismos para manter as caches e a memória consistentes.

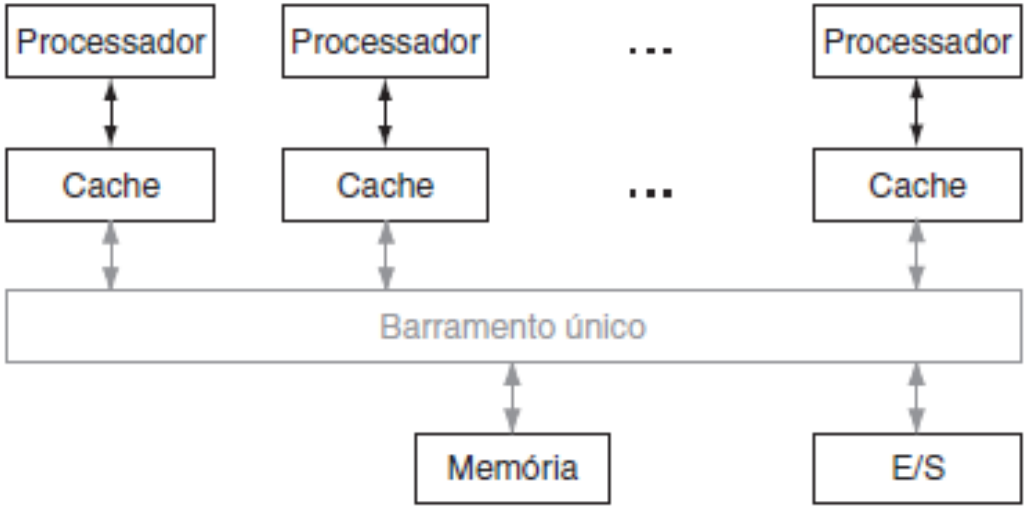


FIGURA 9.3.1 Um multiprocessador de barramento único. O tamanho típico é entre 2 e 32 processadores.

Exemplo de programa paralelo

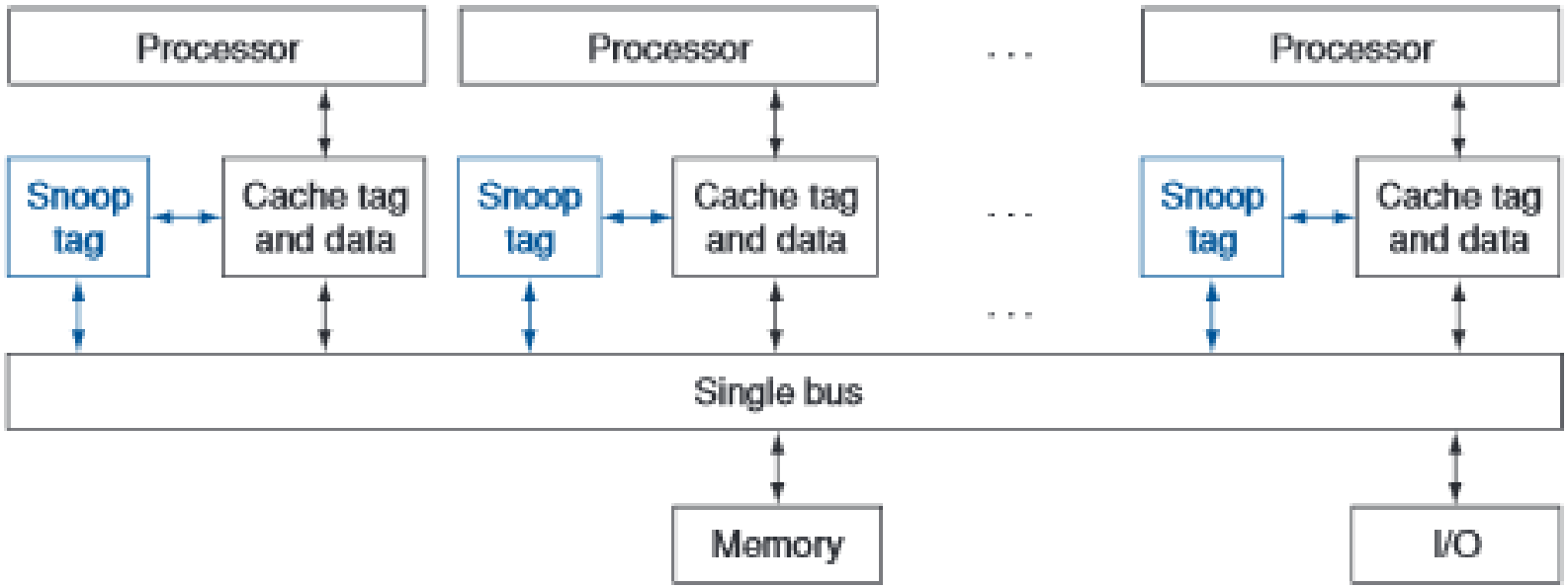
- Suponha que queremos somar 100.000 números em um computador com multiprocessador de barramento único. Vamos considerar que temos 100 processadores.

```
sum[Pn] = 0;
for (i = 1000*Pn; i < 1000*(Pn+1); i = i + 1)
   sum[Pn] = sum[Pn] + A[i]; /* soma as áreas atribuídas */
```

```
half = 100; /* 100 processadores no multiprocessador */
do
{
        synch( ); /* espera a conclusão da soma parcial */
        if (half%2 != 0 && Pn == 0)
              sum[0] = sum[0] + sum[half-1];
              /* soma condicional necessária quando half é
              ímpar; Processor0 obtém elemento ausente */
        half = half/2; /* linha divisora sobre quem soma */
        if (Pn < half) sum[Pn] = sum[Pn] + sum[Pn+half];
}
while (half == 1); /* sai com soma final em Sum[0] */
```

Coerência de cache em multiprocessadores

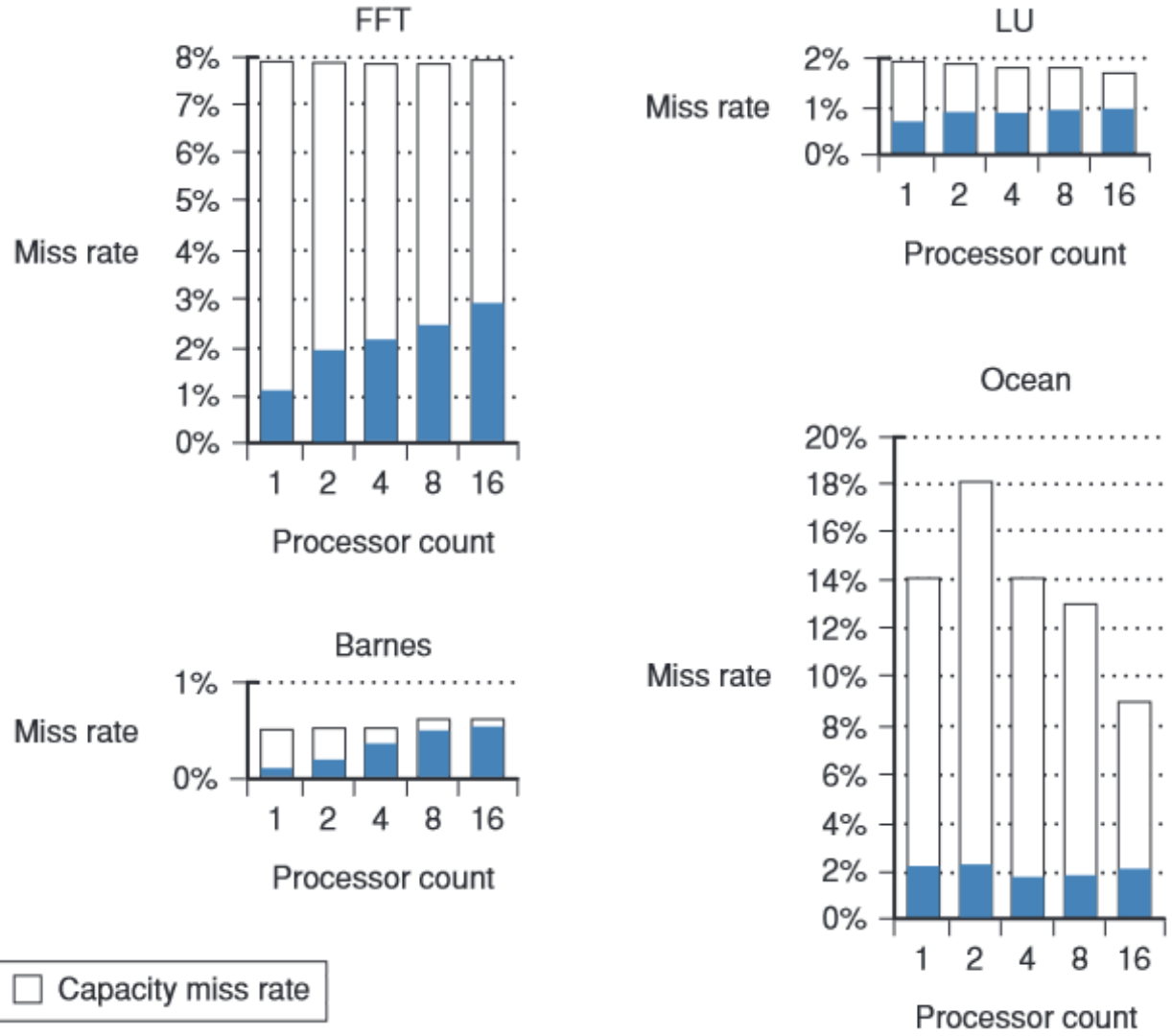
- O protocolo mais conhecido de coerência de cache é o **snooping**.
- Consiste em monitorar o barramento e tomar as ações necessárias em casos de escrita e leitura



Coerência de cache em multiprocessadores

- Dois problemas:
  - Processadores devem ter acesso exclusivo à memória no caso de escrita
  - Processadores devem ter em suas caches a cópia mais recente de um dado
- O protocolo deve localizar todas as caches que têm uma cópia de um dado que está sendo escrito
- Após uma escrita, as cópias podem ser atualizadas ou invalidadas
- O protocolo mais adotado comercialmente utiliza **write-back** e **write-invalidate**.

Coerência de cache em multiprocessadores



Coerência de cache em multiprocessadores

- Exemplo de protocolo de coerência de cache
- Diagrama de transição de estados para um protocolo write-invalidate/write-back.
- Cada bloco de cache está em um de três estados:
  - **Compartilhado** (apenas leitura): esse bloco de cache é limpo (não escrito) e pode ser compartilhado.
  - **Modificado** (leitura/escrita): esse bloco de cache é sujo (escrito) e não pode ser compartilhado.
  - **Inválido**: esse bloco de cache não possui dados válidos.

Coerência de cache em multiprocessadores

- Procedimento em caso de **read hit**:
  - Nada muda no estado da cache

- Procedimento em caso de **read miss**:
  - Processador adquire o barramento
  - Escreve na memória o bloco alvo se ele estiver no estado **Modified** (sujo).
  - Todas as caches nos outros processadores monitoram o cache miss para ver se o bloco está em sua cache.
  - Se algum processador tiver uma cópia e o bloco estiver no estado **Modified**, então ele é escrito novamente e seu estado é alterado para **Invalid**.
  - O bloco é lido da memória e o seu estado muda para **Shared**.

Coerência de cache em multiprocessadores

Procedimento em caso de **write hit**:

- Write hit em um bloco **Modified** não causa nenhuma ação
- Write hit em um bloco no estado **Shared**:
  - Cache adquire o barramento
  - envia sinal de invalidação para as outras cópias
  - escreve no bloco e muda o seu estado para **Modified**.

Coerência de cache em multiprocessadores

Procedimento em caso de **write miss:**

- Em um bloco no estado **Invalid** causa:
  - Cache adquire o barramento
  - lê o bloco faltante
  - escreve no bloco
  - muda seu estado para **Modified**

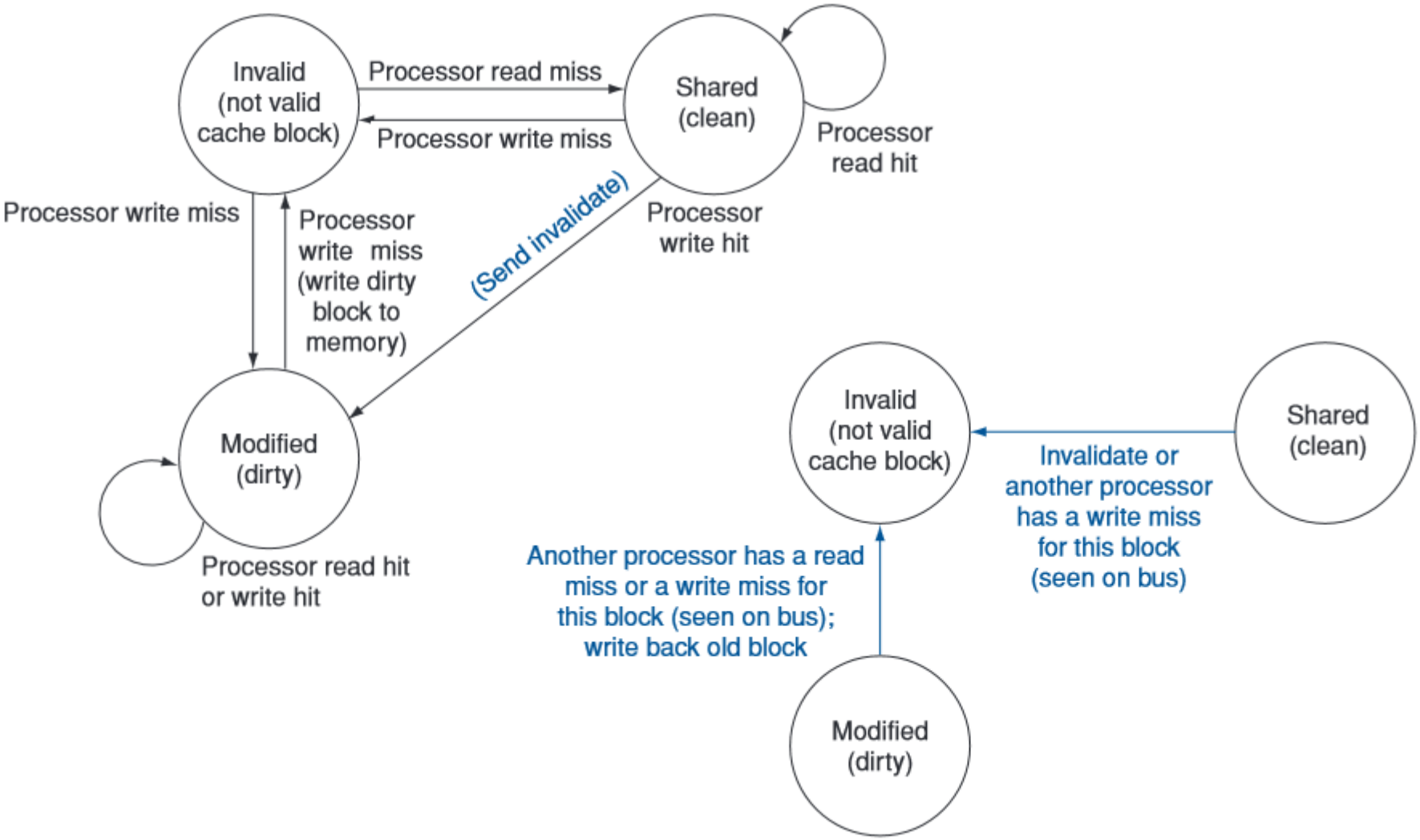
- Se o bloco estiver no estado **Shared** em qualquer outra cache:
  - Cache adquire o barramento
  - Envia um sinal para invalidar todas as cópias
  - lê o bloco faltante
  - escreve no bloco
  - muda seu estado para **Modified**

Coerência de cache em multiprocessadores

Erro de tradução do livro:

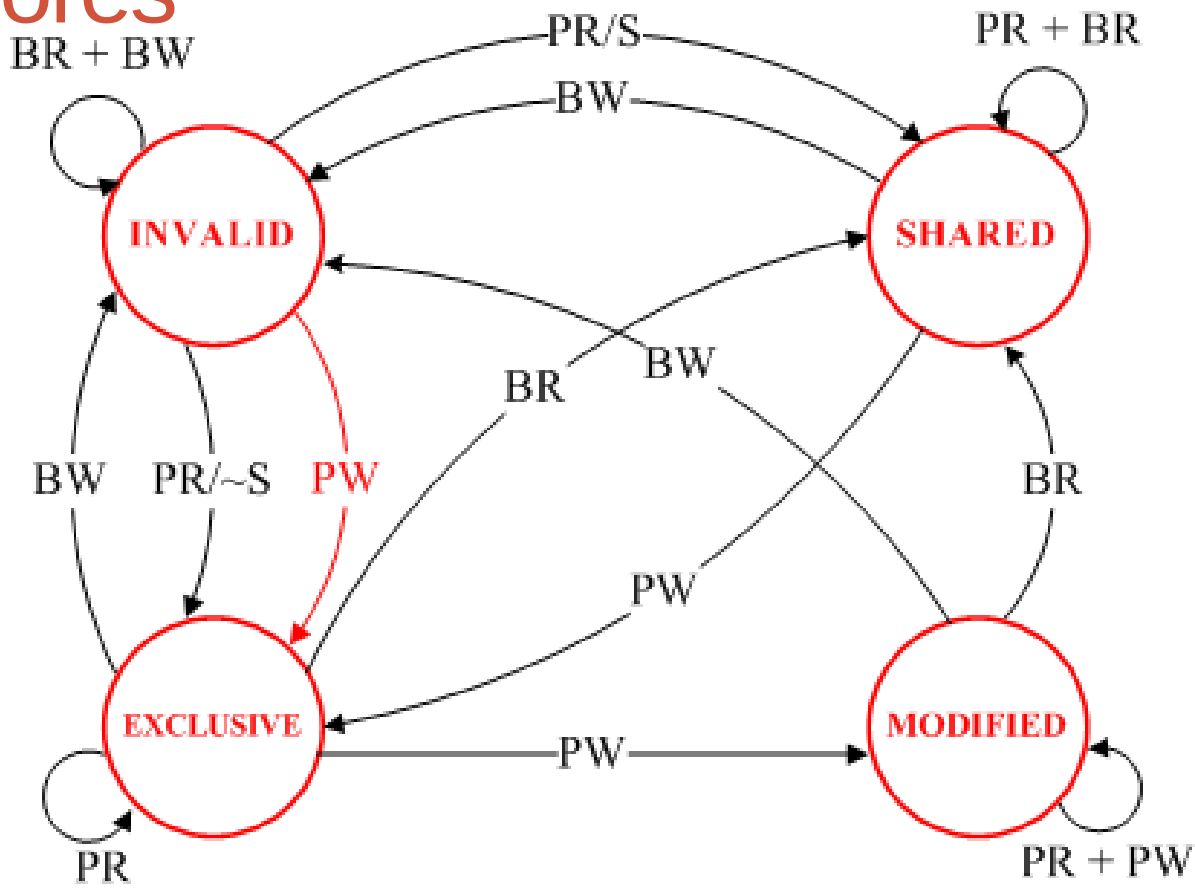
- **Tradução**: *“Por último estão as falhas de escrita. Uma falha de escrita em um bloco Shared em outra cache faz com que a cache adquira o barramento, envie um sinal de invalidação para bloquear todas as cópias,leia todo o bloco em que houve falha, modifique a parte do bloco sendo escrita e mude o estado para Modified.”*
- **Texto original**: *“Last is write misses. A write miss to an Invalid block causes the cache to acquire the bus, read the full missing block, modify the portion of the block being written, and change the state to Modified. A write miss to a Shared block in another cache causes the cache to acquire the bus, send an invalidate signal to knock out all copies, read the full missing block, modify the portion of the block being written, and change the state to Modified”*

Coerência de cache em multiprocessadores



Coerência de cache em multiprocessadores

- Há muitas variações de algoritmos de coerência de cache em multiprocessadores
- Uma variação é o MESI (Modified, Exclusive, Shared, Invalid), usado no Pentium 4 e processadores mais modernos



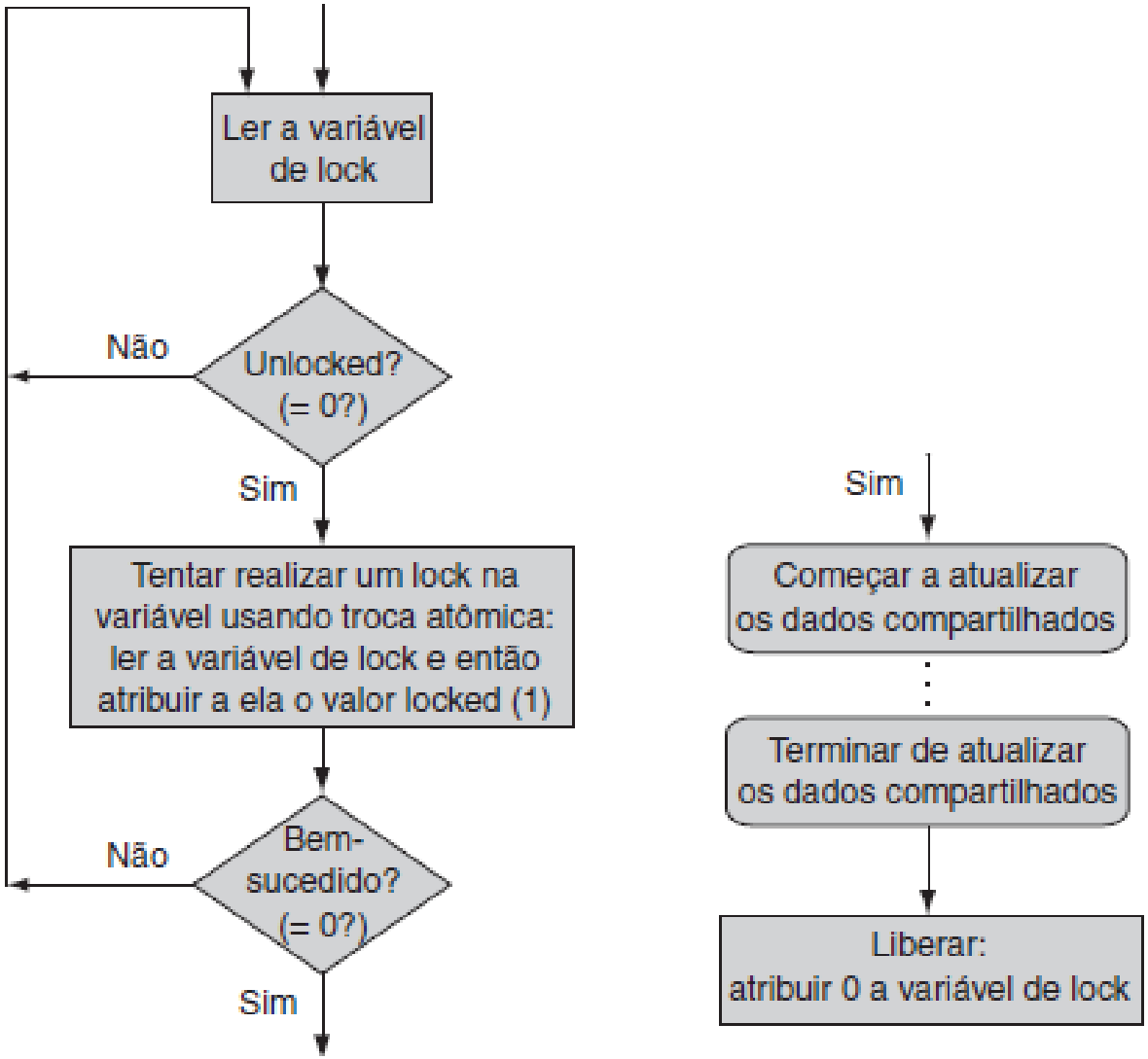
PR = processor read
PW = processor write
S/~S = shared/NOT shared

BR = observed bus read
BW = observed bus write

Sincronização

- É necessário coordenar processos que atuam numa mesma tarefa em um sistema multiprocessador.
- Lock variables (semáforos)
- O barramento em si já bloqueia que mais de um processo o utilize simultaneamente
- Mas é preciso um meio de dar o controle a apenas um processo
- É preciso ter uma **operação atômica de swap**

Sincronização



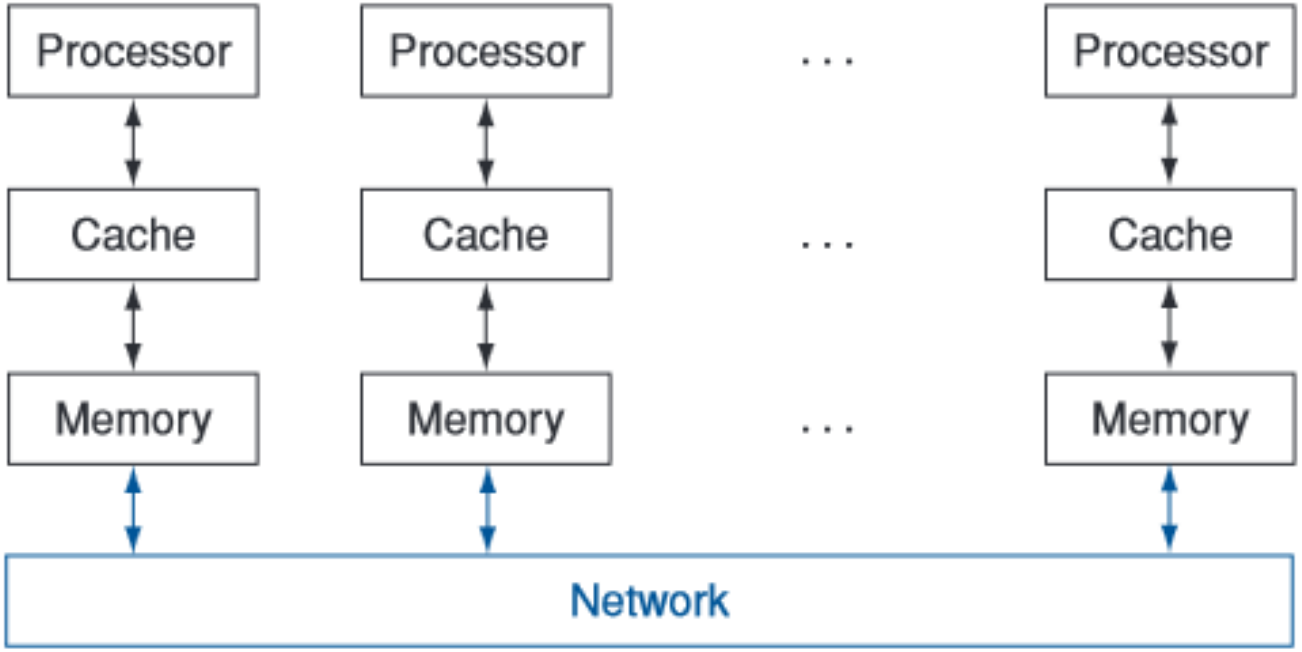
Sincronização

- O MIPS não tem uma instrução atômica de swap.
- Mas tem as instruções `load linked (ll)` e `store conditional (sc)`
- Exemplo de código (spin lock):

```
try:  mov   $t3,$t4     # move o valor de troca
      ll    $t2,0($t1)  # load linked
      sc    $t3,0($t1)  # store conditional altera $t3
      beqz  $t3,try     # desvia se store conditonal falhar (=0)
      nop               # (delayed branch)
      mov   $t4,$t2     # coloca o valor do load em $t4
```

Multiprocessadores Conectados por Rede

- A arquitetura de barramento tem limitações práticas: largura de banda, baixa latência e comprimento o maior possível são parâmetros incompatíveis.
- A conexão em rede não é mais entre processadores e memória. A rede é usada só para comunicação entre processadores



Multiprocessadores Conectados por Rede

- Os dois exemplos dados trazem dois conceitos comuns na arquitetura de computadores massivamente paralelos: memória compartilhada e memória distribuída.
- Espaçamento de endereços único ou múltiplo.
- Isso gera as seguintes categorias gerais de multiprocessadores:
  - UMA (Uniform Memory Access)
  - NUMA (Nonuniform Memory Access)
  - CC-NUMA (Cache-coherent nonuniform memory access)

Multiprocessadores Conectados por Rede

- Exemplo da soma de 100000 valores
- cada soma parcial está localizada em uma unidade de execução diferente.
- Primeiro, metade das unidades de execução envia suas somas parciais para a outra metade, onde duas somas parciais são somadas.
- Depois, um quarto das unidades de execução (metade da metade) envia essa nova soma parcial para o outro quarto das unidades de execução (a metade da metade restante) para a próxima rodada de somas.
- Essas divisões, envios e recepções continuam até haver uma única soma de todos os números.

Multiprocessadores Conectados por Rede

```
sum = 0;
for (i = 0; i<1000; i = i + 1) /* loop em cada array */
 sum = sum + A1[i]; /* soma os arrays locais */

limit = 100; half = 100; /* 100 processadores */
do
{
  half = (half+1)/2; /* linha divisória entre send e receive */
  if (Pn >= half && Pn < limit) send(Pn – half, sum);
  if (Pn < (limit/2)) sum = sum + receive( );
  limit = half; /* limite superior dos emissores */
}
while (half == 1); /* sai com a soma final */
```

Multiprocessadores Conectados por Rede

- Distributed Shared Memory (DSM)
- Consiste em se criar um endereçamento único de memória para processadores ligados em rede.
- O problema da coerência não pode ser resolvido por snooping
- Uso do conceito de diretórios
- Tem duas grandes desvantagens:
  - as “cache misses” levarão muito mais tempo para se resolver
  - A largura de banda da rede tem que ser muito alta

Clusters

- Muitas aplicações rodam bem em sistemas fracamente acoplados:
  - Bancos de dados,
  - Servidores de arquivos,
  - Servidores Web,
  - Simulações e multiprogramação/processamento
  - Com as redes de alta velocidade (gigabit ethernet) passou a ser comum o uso de clusters de workstations

Clusters

- Desvantagems:

1. Custo de administração
2. Largura de banda E/S (fracamente acoplado)
3. Menos memória disponível para cada processador

- Vantagens:

1. Baixo custo
2. Alta disponibilidade
3. Fácil de se expandir a rede

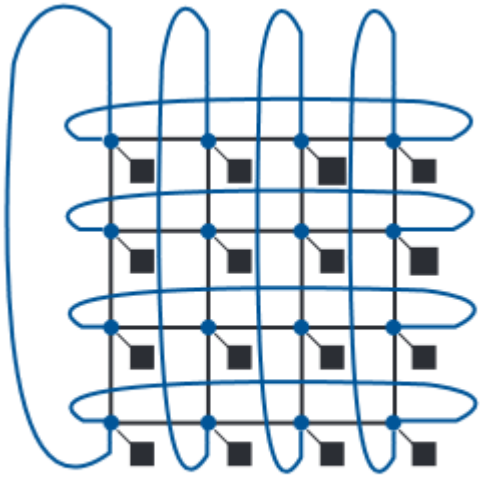
Para se obter as vantagens dos dois mundos, existem também os clusters de SMPs. Por exemplo, 8 máquinas em rede onde cada uma é um SMP de 4 processadores

Topologias de Rede

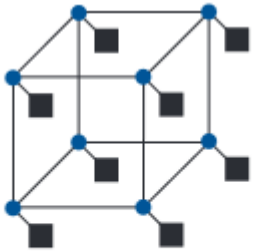
- Aspectos importantes da rede:
- Custo (número de switches, número de links em um switch, largura de banda, comprimento do link)
- Desempenho (latência, throughput, atrasos de contenção)
- Tolerância a falhas
- Topologia
  - Dois extremos: anel e totalmente ligada
  - A topologia em anel pode ter performance reduzida, mas a baixo custo.
  - A totalmente ligada tem performace ótima a um custo proibitivo.

Topologia de rede

- Algumas topologias possíveis

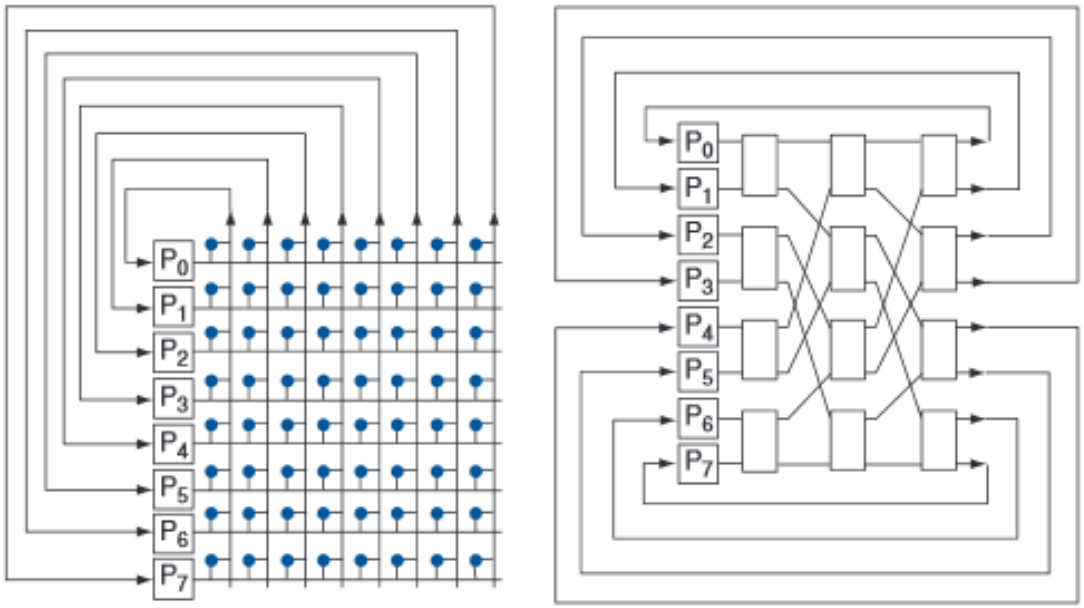


a. 2-D grid or mesh of 16 nodes



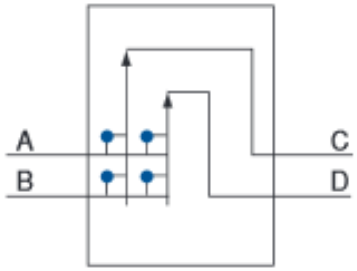
b. n-cube tree of 8 nodes ($8 = 2^3$ so $n = 3$)

Topologia de rede



a. Crossbar

b. Omega network



c. Omega network switch box

Multiprocessadores dentro de um Chip e Multithreading

- A solução que vem sendo aplicada recentemente é a de vários núcleos processadores no mesmo chip, ou multicore
- A grande vantagem é que não há latências entre chips
- As caches são também compartilhadas
- A idéia é aumentar a utilização de recursos simultâneos: Multithreading
- Cada thread deve ter um estado independente: PC próprio, banco de registradores, tabela de paginação.
- O hardware deve prover mecanismos de troca de contexto rápida

Multiprocessadores dentro de um Chip e Multithreading

- Abordagens de implementação:
- **Fine-grained**: comuta entre threads a cada instrução ou tempo (round-robin)
- **Coarse-grained**: comuta entre threads em paradas do processo que são mais onerosas, como um cache miss.
- **SMT** (Simultaneous Multi-Threading)
  - é uma variação de multithreading que usa os recursos de escalonamento dinâmico para explorar paralelismo em nível de thread ao mesmo tempo em que explora o paralelismo em nível de instrução.

Multiprocessadores dentro de um Chip e Multithreading

- Duas observações são importantes:
  - Em geral, o overhead devido ao multithreading é pequeno
  - A eficiência dos superescalares é baixa e o SMT parece ser o método mais promissor para melhoria.

- A Intel chama seu suporte de SMT no Pentium 4 de ***Hyper-Threading***.
- Dobrando o estado arquitetural do IA-32, ele suporta apenas duas threads, que compartilham todas as caches e unidades funcionais.

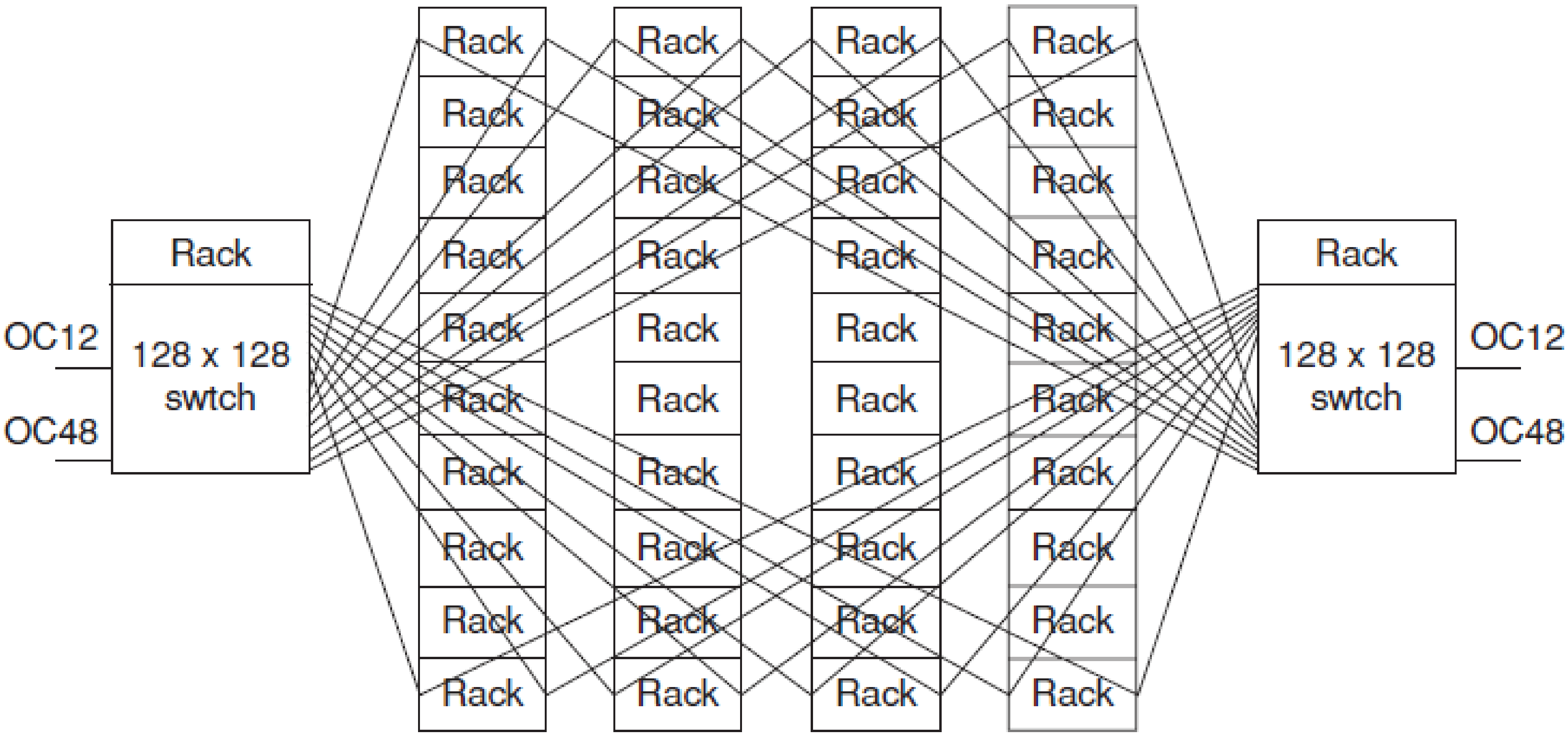
O Cluster Google

- Confiabilidade é importante
- 1000 consultas por segundo
- 3 bilhões de páginas indexadas
- Cada página é revisitada todo mês
- 6000 processadores, 12000 discos, 1 petabyte de armazenamento
- Sites redundantes (e não RAID), dois sites no Vale do Silício e dois na Virgínia
- Cada site conectado por um link OC48 (2488 Mbit/sec)
- Link OC12 conectando um par de sites (para redundância)

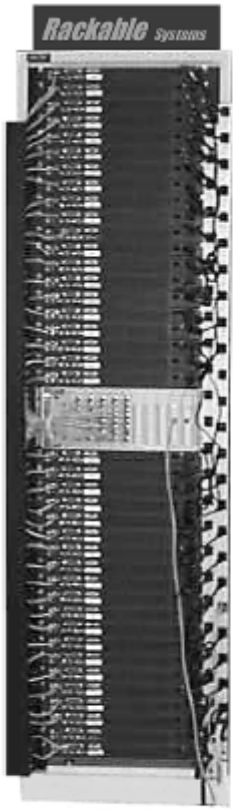
byte	B	10^0
Kilobyte	kB	10^3
megabyte	MB	10^6
gigabyte	GB	10^9
terabyte	TB	10^{12}
petabyte	PB	10^{15}
exabyte	EB	10^{18}
zettabyte	ZB	10^{21}
yottabyte	YB	10^{24}

O Cluster Google

- Cada rack contém 80 PCs, num total de 3200 PCs
- Links Gb Ethernet em cada PC com roteadores 128x128



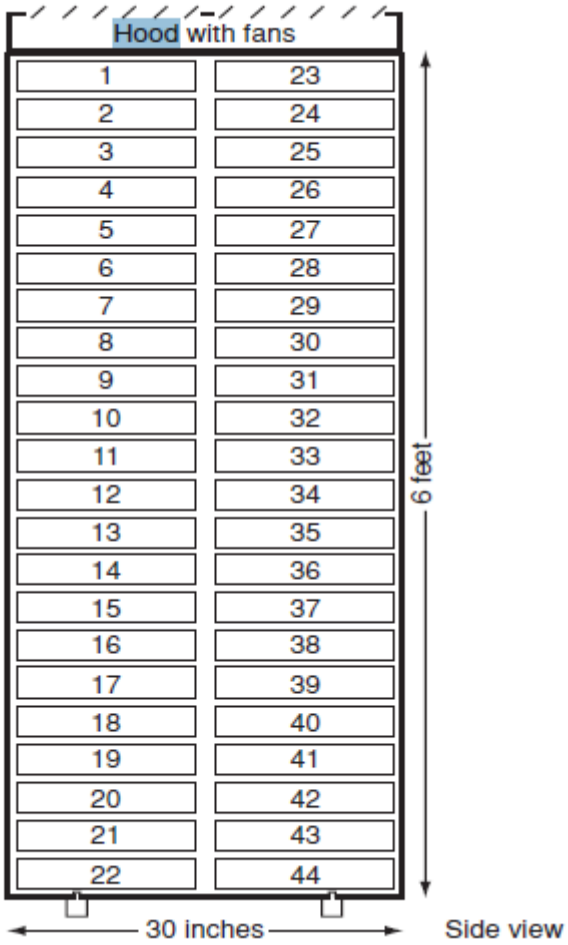
O Cluster Google



Front view (also back view)

19 inches

Close-up view of 1 RU PCs



Side view

O Cluster Google

- http://www.google.com/about/datacenters/

Categorias de Paralelismo (Flynn,1966)

1. *Fluxo de instruções único, fluxo de dados único* (SISD, o uniprocessador)
2. *Fluxo de instruções único, fluxos de dados múltiplos* (SIMD)
3. *Fluxos de instruções múltiplos, fluxo de dados único* (MISD)
4. *Fluxos de instruções múltiplos, fluxos de dados múltiplos* (MIMD)

Computadores SIMD

- Operam em vetores de dados.
- Por exemplo, quando uma única instrução SIMD soma 64 números, o hardware SIMD envia 64 fluxos de dados para 64 ALUs para formar 64 somas dentro de um único ciclo de clock.
- Os computadores SIMD reais possuem uma mistura de instruções SISD e SIMD.
- SIMD funciona bem quando há **paralelismo de dados**.

Computadores Vetoriais

- Um modelo relacionado ao SIMD é o **processamento vetorial**.
- Consideravelmente mais usado do que o SIMD.
- Os processadores vetoriais possuem operações de alto nível que atuam em arrays de números lineares, ou vetores. Um exemplo de operação vetorial é

$$A = B \times C$$

- onde A, B e C são vetores de 64 elementos de números de ponto flutuante de 64 bits.
- Diferença para o SIMD: processadores vetoriais dependem de unidades funcionais em pipeline, enquanto o SIMD opera em todos os elementos ao mesmo tempo.

Computadores Vetoriais

- As vantagens dos computadores vetoriais sobre os processadores SISD tradicionais são:
  - Cada resultado é independente dos resultados anteriores, o que permite pipelines profundos e altas velocidades de clock.
  - Uma única instrução vetorial realiza uma grande quantidade de trabalho, o que significa menos buscas de instruções e menos instruções de desvio e, portanto, menos desvios mal previstos.
  - Não precisam se basear em altas taxas de acerto das caches de dados para ter alto desempenho, devido ao acesso em blocos e menor latência

Computadores MIMD

- São os multiprocessadores com barramento e memória compartilhada

FIGURA 9.11.4 O IBM RS/6000 SP2 com 256 processadores. Esta máquina de memória distribuída é construída usando placas de computadores desktop inalteradas, além de um switch personalizado como a interconexão. Comparado com o SP2, a maioria dos clusters usa uma rede local comutada original. A foto é cortesia do Lawrence Livermore National Laboratory.