Arquitectura de Computadores II

LESI - 3º Ano

Multiprocessadores

João Luís Ferreira Sobral
Departamento do Informática
Universidade do Minho

Maio 2002

Multiprocessadores

- Utilização de vários processadores numa arquitectura
  => aumento do desempenho

- N processadores => ganho de N vezes?

  - capacidade de processamento superior a qualquer sistema uniprocessador

- Escaláveis: o desempenho do sistema pode ser melhorado adicionando mais unidades de processamento.

- Populares em servidores de ficheiros e de bases de dados

- Utilizados para executar uma só aplicação (p. ex. simulação do tempo) ou para suportar a carga de vários utilizadores (p. ex. servidores Web)

- Classificação dos sistemas (Taxonomia de Flynn)

  - Classificação de acordo com o número de fluxos de instruções e de dados processados (Exemplo para MIPS)

1. **SISD - Single Instruction Single Data** – Um só fluxo de instruções e de dados (processadores convencionais)

IF	ID	EX	MEM	WB

2. **SIMD - Single Instruction Multiple Data –** Um só fluxo de instruções processa vários fluxos de dados

		EX	MEM	WB
IF	ID	EX	MEM	WB
		EX	MEM	WB

3. **MIMD - Multiple Instruction Multiple Data** – Vários fluxos de instruções processam vários fluxos de dados

IF	ID	EX	MEM	WB

IF	ID	EX	MEM	WB

IF	ID	EX	MEM	WB

Multiprocessadores

- MIMD
  - Tem prevalecido por ser mais flexível e poder ser desenvolvido com base em processadores comerciais
  - O hardware pode suportar um modelo de **memória partilhada** ou de **memória distribuída**

- MIMD com Memória partilhada centralizada:
  - um só espaço de endereçamento partilhado por todos os processadores
  - primitivas para sincronizar os acesso às zonas de memória partilhada
  - velocidade de acesso à memória pode ser uniforme (UMA)

- MIMD com Memória distribuída:
  - cada processador possui o seu espaço de endereçamento
  - existem primitivas para o envio de informação entre processadores

- Sistemas híbridos de memória partilhada e distribuída
  - velocidade de acesso à memória não uniforme (NUMA) variando em função do endereço acedido.

- Vantagens de memória distribuída:
  - A largura de banda de acesso à memória escala mais facilmente
  - A latência dos acessos à memória é menor

- Desvantagens de memória distribuída:
  - A comunicação entre máquina é mais complexa e a latência de comunicação entre máquinas é superior

- É possível simular por SW um sistema de memória partilhada numa sistema de memória distribuída (*Virtual Shared Memory*)

Multiprocessadores

- Programação mais complexa que a dos sistemas uniprocessador:

  - condicionada à obtenção de bons ganhos
    => forte envolvimento do programador

  - complexidade de aplicações sequenciais + divisão da aplicação em tarefas paralelas + identificação das interacções entre tarefas.

  - ganho condicionado pela fracção sequencial da aplicação

    Se *s* for a fracção sequencial do algoritmo, então 1/*s* é o ganho máximo que pode ser obtido com essa aplicação:

    Exemplo:

    Se 10% (s=0,1) da aplicação for sequencial qual o ganho máximo que pode ser obtido?

$$ganho_p = \frac{1}{\frac{0,9}{p} + 0,1} \underset{p \to \infty}{=} 10$$

    Se for utilizado um número *infinito* de processadores o ganho é 10 x! (8,5x c/ 50 processadores, 9,2x c/ 100 processadores)

  - impacto negativo no desempenho da aplicação da latência dos acessos a memória remota

    Exemplo:

    acesso a uma memória remota = 2000 ns
    Tcc = 10 ns
    acessos remotos = 0,5%
    CPI sem acessos remotos = 1

    Qual o CPI se forem considerados os acessos remotos?

    CPI = CPI *Base* + % acessos remotos x custo de cada acesso

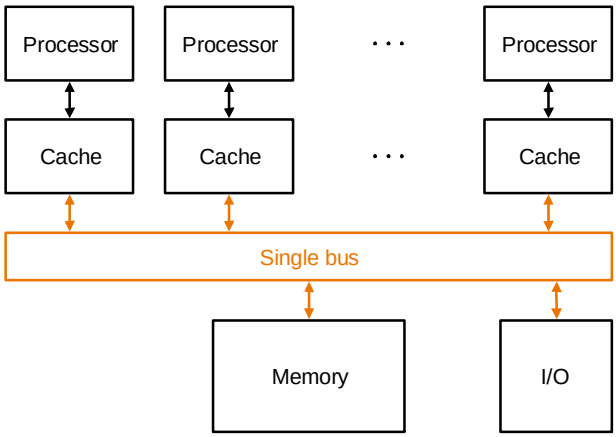
    CPI = 1,0 + 0,05 x 2000/10 = 2,0

    Uma máquina apenas com acessos locais é 2x mais rápida!

Multiprocessadores

- Conectados por um único barramento
  (arquitecturas com memória partilhada centralizada):

  - Vários processadores partilham um barramento de acesso à memória
  - As *caches* de cada processador contribuem para reduzir o tráfego no barramento e a latência dos acessos à memória
  - um valor pode estar replicado em vários sítios => são necessários mecanismos para assegurar a coesão entre as *caches* dos vários processadores e a memória
  - A largura de banda de acesso à memória é partilhada pelos vários processadores => limitação à escalabilidade deste tipo de arquitectura



  - Exemplos (1997):

Nome	Nº máx. de CPU	CPU	Freq. Relógio	Memória Máxima	Débito de comunicação
Compaq Proliant 5000	4	Pentium Pro	200 MHz	2,0 GB	540 MB/s
Digital AlphaServer 8400	12	Alpha 21164	440 MHz	28,6 GB	2150 MB/s
HP 9000 k460	4	PA-8000	180 MHz	4,0 GB	960 MB/s
IBM RS/6000 R40	8	PowerPC 604	112 MHz	2,0 GB	1800 MB/s
SGI Power Challenge	36	MIPS R10000	195 MHz	16,3 GB	1200 MB/s
SUN Enterprise 6000	30	UltraSPARC 1	167 MHz	30,7 GB	2600 MB/s

  - Exemplos (2002 – Boards para Intel/AMD):

Nome	Nº CPU	CPU	Freq. Rel.	Mem. Max	BW Memória	Outros
Tyan Thunder K7X	2	AMD Athlon MPX	1,73 GHz	4,0 GB	2100 MB/s	2 x 64 bit PCI Dual Ultra 160 SCSI
Tyan Thunder i860	2	Intel Pentium 4 Xeon	2,20 GHz	4,0 GB	3200 MB/s	2 x 64 bit PCI Dual Ultra 160 SCSI
Tyan Thunder i7500	2	Intel Pentium 4 Xeon	2,20 GHz	12,0 GB	3200 MB/s	2 x 64 bit PCI-X Dual Ultra 160 SCSI

Multiprocessadores

Coesão de *caches*

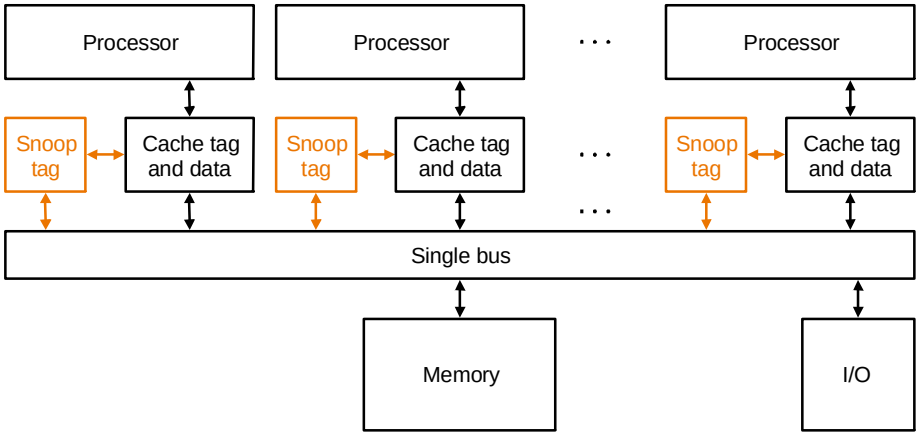
- O facto de os processadores poderem utilizar protocolos de *cache writeback*, pode originar valores inconsistentes entre a *cache* e a memória principal partilhada com outros processadores.

- Os protocolos que asseguram a coesão entre os valores das *caches* dos vários processadores e da memória central são designados por **protocolos de coesão de *caches***

- **Coesão baseada em *Snooping*** – cada bloco da *cache* contém informação indicando se é partilhado. O processador monitoriza a actividade de barramento verificando se esta envolve um bloco local:

  - ***Write-invalidate***: Quando um processador escreve um valor na *cache* todas as outras cópias são tornadas inválidas através de um sinal do barramento.

  - ***Write-update***: Quando um processador escreve um valor na *cache* todas as outras cópias são também actualizadas.



  - *write-invalidate* tende a gerar menos tráfego no barramento, uma vez várias escritas numa mesma linha da *cache* não geram tráfego.

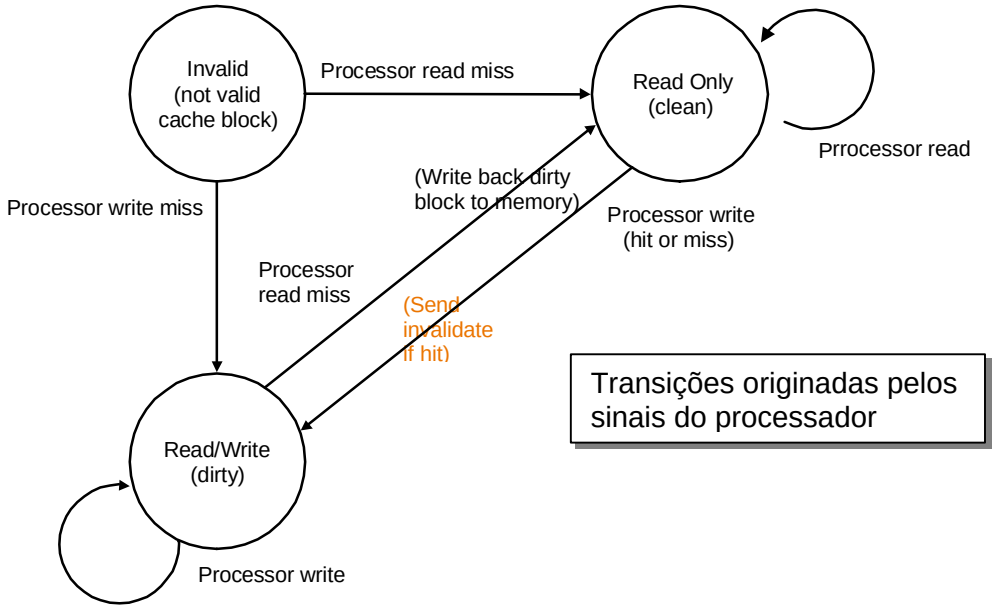
  - *write-update* tende a reduzir a latência nos acessos, porque as cópias estão sempre actualizadas

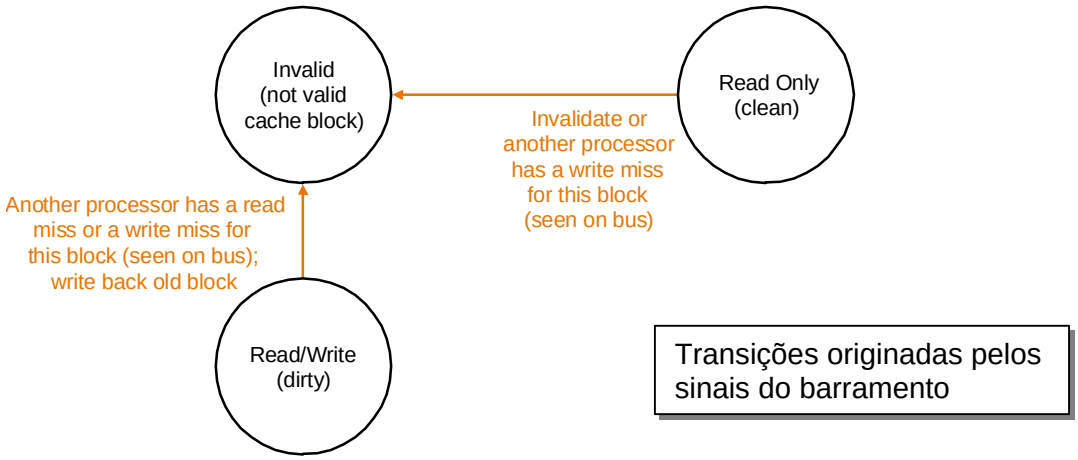
  - A generalidade dos sistemas comerciais utiliza o esquema *write-invalidate*.

  - As linhas de *cache* de elevada dimensão dão origem ao problema da falsa partilha:
    variáveis utilizadas por processadores diferentes e colocadas num mesmo bloco => o bloco é invalidado pelos dois processadores

Multiprocessadores

- Exemplo de um protocolo de coesão de *caches* (tipo *write-invalidate*)
  - Cada linha da cache pode estar num de três estados:
    1. Apenas de leitura
    2. Leitura/escrita – quando o processador escreve na cache
    3. Inválido - quando outro processador invalida o bloco





- Pentium Pro e o PowerPC utilizam uma variante deste protocolo (MESI), onde o estado *Read Only* está dividido em dois (Exclusive/Shared), indicando se o bloco é partilhado ou não.

Multiprocessadores

Implementação de operações de sincronização

- Com uma operação atómica de leitura e atribuição de um valor

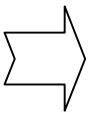
Implementação clássica

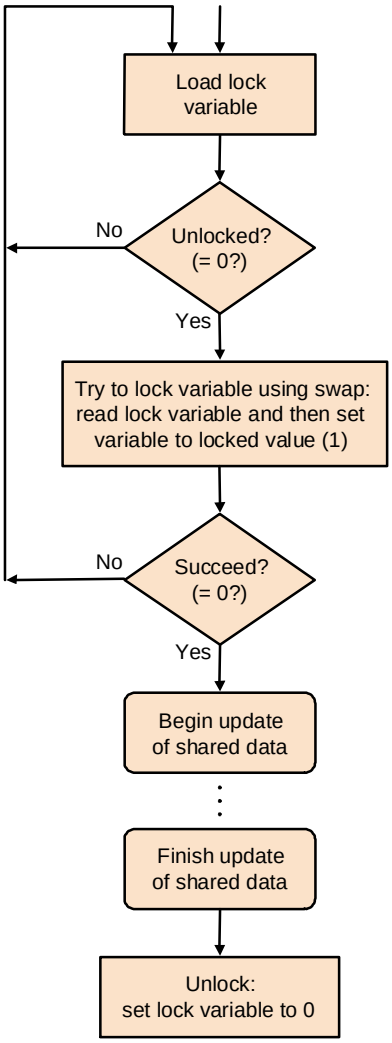
```
        li      R2, 1
lock:   exch    R2, 0 (R1)      # troca atómica
        bnez    R2, lock
```

- a troca atómica gera *write-miss* e *read-miss*!

Implementação tirando partido do protocolo de coesão de caches

```
lock:   lw      R2, 0 (R1)      # spin-lock
        bnez    R2, lock        # spin-lock
        li      R2, 1
        exch    R2, 0(R1)       # troca atómica
        bnez    R2, lock
```





- apenas gera *write-miss* quando o valor da variável é alterado!

- Este esquema tira partido do protocolo de coesão de *caches*, uma vez que a leitura das variáveis (*spin lock*) não gera tráfego no barramento

- O *spin-lock* desperdiça ciclos de CPU, não escalando com o número de processadores

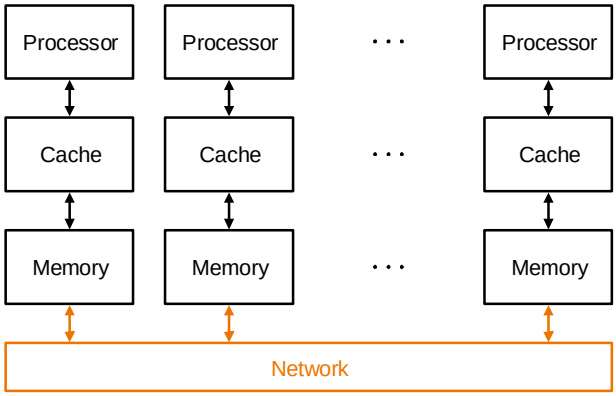
- Exemplo com 3 processadores

Passo	Processador P0	Processador P1	Processador P2	Actividade no barramento
1	possui o fecho	testando se fecho==0	testando se fecho==0	nenhuma
2	fecho=0, envia 0 pelo barr/	testando se fecho==0	testando se fecho==0	*write-invalidate* de fecho em P0
3		*cache miss*	*cache miss*	barramento serve P2
4		espera pelo barramento	fecho =0	serve *cache miss* de processador P2
5		fecho =0	lê fecho e atribui fecho=1	serve *cache miss* de processador P1
6		lê fecho e obtém fecho=1	fecho local =0 e envia 1 pelo barr/	*write-invalidate* de fecho em P2
7		fecho local =1 e envia 1 pelo barr/	possui o acesso ao fecho	*write-invalidate* de fecho em P1
8		testando se fecho==0		nenhuma

Multiprocessadores

Conectados por uma rede de interligação

- Os sistemas interligados por um barramento tendem a limitar o número de processadores que efectivamente podem ser ligados
- A alternativa reside na utilização de uma rede dedicada à interligação dos vários processadores, possuindo cada processador a sua memória dedicada



- Exemplos (1997):

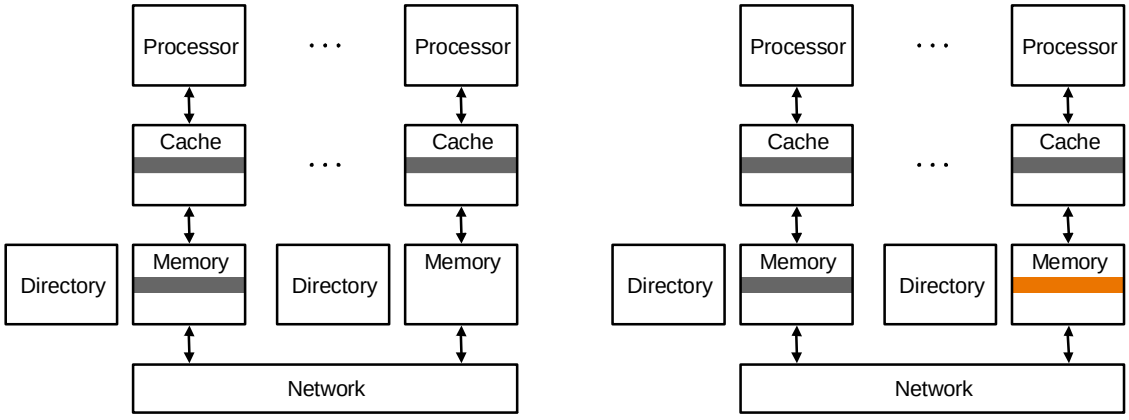
Nome	Nº máx. de CPUs	CPU	Freq. Rel.	Memória Máxima	Largura de Banda de comunicação	Tipo de Nodos	Topologia
Cray Research T3E	2048	Alpha 21164	450 MHz	524 GB	1200 MB/s	4-*way*	3-D torus
HP/Convex Exemplar X-class	64	PA-8000	180 MHz	65 GB	980 MB/s	2-*way*	8-way crossbar + anel
Sequent NUMA Q	32	Pentium Pro	200 MHz	131 GB	1024 MB/s	4-*way*	anel
SGI Origin 2000	128	MIPS R10000	195 MHZ	131 GB	800 MB/s	2-*way*	6-cube
Sun Enterprise 10000	64	Ultra SPARC 1	250 MHz	65 GB	1699 MB/s	4-*way*	16-way crossbar

Multiprocessadores

- Conectados por uma rede de interligação

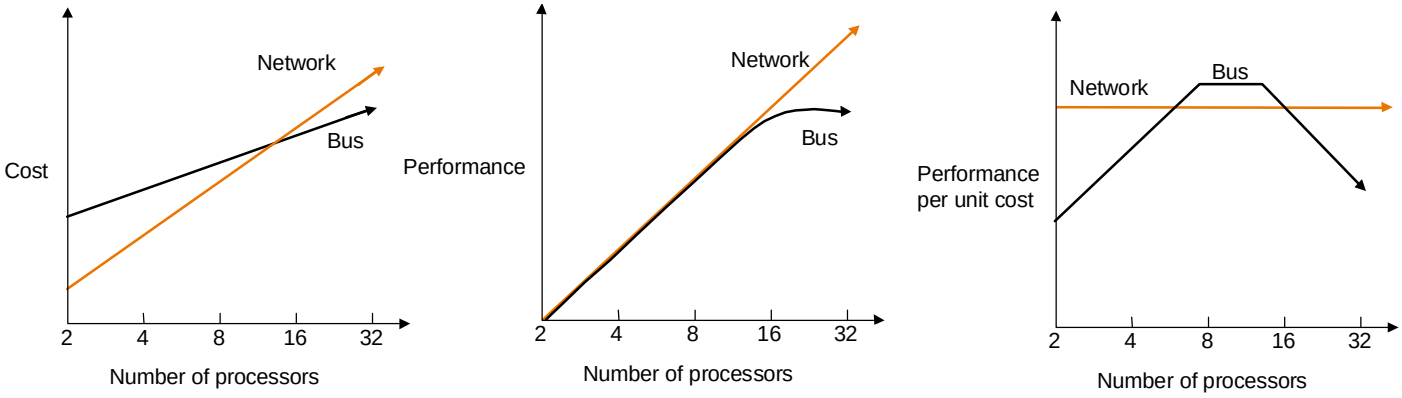
  Implementação de um endereçamento partilhado

  - A implementação mais simples é não fornecer HW para suporte a coesão de *cache* => a cache não é utilizada em acessos remotos
  - **Serviço de directoria** que mantém a informação sobre a partilha de cada bloco local de memória:
    - A directoria é distribuída pelos processadores
    - Cada entrada indica os processadores que contêm uma cópia do bloco de memória
    - A invalidação dos blocos em *caches* é efectuada través de mensagens, com base em informação na directoria



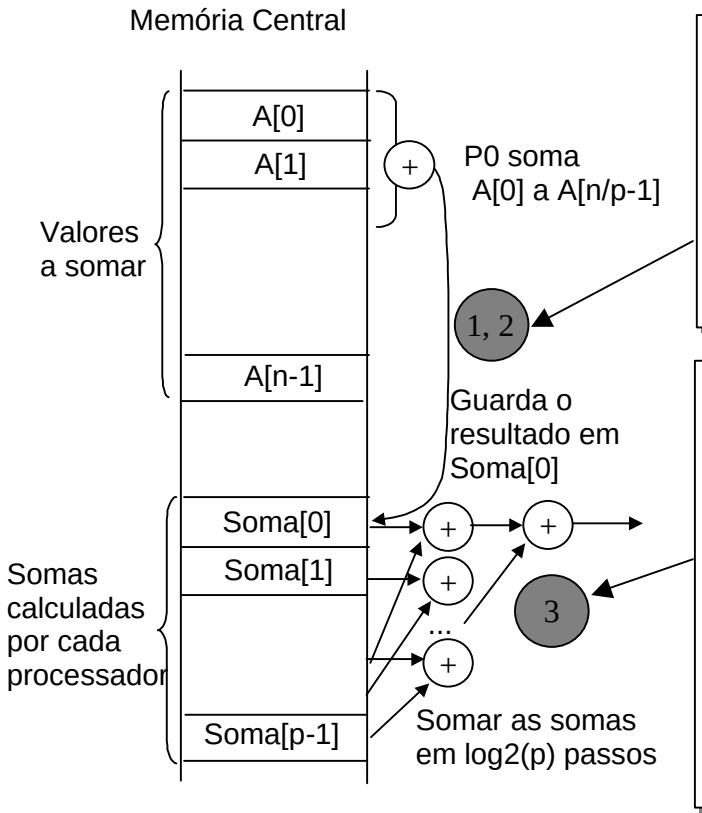
a. b.

- Comparação entre os sistemas conectados por um barramento e os sistema conectados por uma rede de interligação



Multiprocessadores

- Programação de sistemas multiprocessadores com memória partilhada
  - Os processadores comunicam através da memória, efectuando *load* e *store*
  - As primitivas de sincronização asseguram que os acessos a regiões críticas são efectuados com exclusividade
- Somar uma lista de números
  1. Dividir os elementos da lista pelos processadores disponíveis
  2. Efectuar a soma parcial em cada processador
  3. Efectuar a soma das várias somas parciais
- Exemplo: somar 100000 números com 10 processadores:



Passos 1 e 2: calcular somas parciais

```
Soma[Pn]=0;

for (i=10000*Pn, i<10000*(Pn+1); i++)
     Soma[Pn]= Soma[Pn] + A[i];
// i é uma variável privada
// em cada processador
```

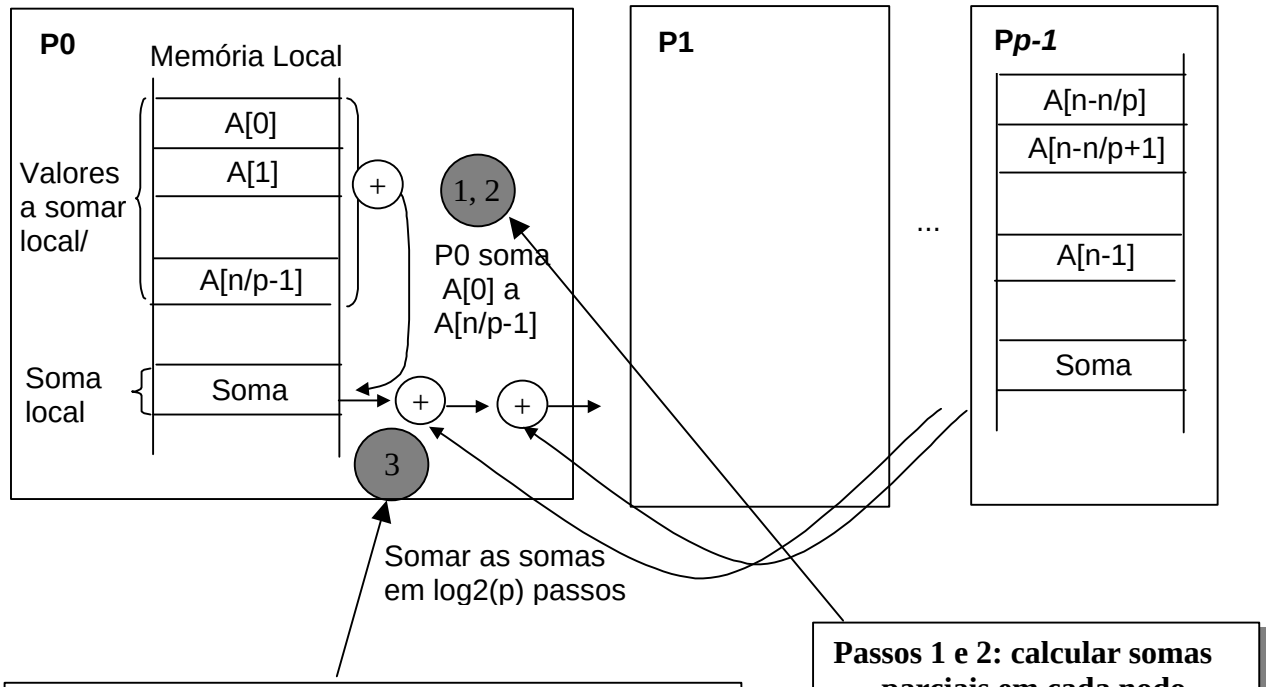
Passo 3: somar as somas parciais

```
Num = 10;
do   {
     synch(); // espera por soma parcial
     if (num%2 != 0 && Pn==0)  // nº impar
          Soma[0]= Soma[0] + Soma[Num-1];
     Num = Num/2;
     if (Pn<Num)
          Soma[Pn] = Soma[Pn] +
                              Soma[Pn+Num];
} while (Num>1);
```


Multiprocessadores

- Programação de sistemas multiprocessadores com memória distribuída
  - Os processadores comunicam através da memória de passagem de mensagens (send, receive)

- Exemplo: somar 100000 números com 100 processadores:



Passo 3: somar as somas parciais

```
Num = 100; // 100 Processadores
Lim = Num
do   {
     Num=(Num+1)/2;
     if (Pn>=Num && Pn<Lim) send(Pn-Num,Soma);
     if (Pn < (Lim/2-1) )
          Soma = Soma + receive();
     Lim = Num;
} while (Num>1);
```

Passos 1 e 2: calcular somas parciais em cada nodo

```
Soma=0;

for (i=0, i<10000; i++)
     Soma= Soma + A[i];
```

- A principal diferença reside no passo 3, porque as somas parciais se encontram distribuídas por vários processadores, sendo necessário enviar mensagens entre processadores com a soma parcial

- Adicionalmente, é necessário uma fase antes dos passos 1,2 para distribuir os valores a somar pelas memórias de cada nodo

Multiprocessadores

Clusters de máquinas

- Constituídos pode HW "normal", interligados por uma rede de alta velocidade
- Cada nodo de processamento pode ser uma máquina de memória partilhada com vários processadores
- Cada nodo possui uma cópia do SO
- Alta disponibilidade: quando falha uma máquina basta substituí-la por outra
- Custos de administração na ordem do número de máquinas
- Exemplos (1997)

Nome	Número máximo de processadores	Processador	Freq. Relógio	Memória Máxima	Largura de Banda de comunicação	Nodos	Máximo de nodos
HP 9000 EPS21	64	PA-8000	180 MHz	65 GB	532 MB/s	*4-way*	16
IBM RS/6000 HACMP R 40	16	PowerPC 604	112 MHz	4 GB	12 MB/s	*8-way*	2
IBM RS/6000 SP2	512	Power2 SC	135 MHz	1000 GB	150 MB/s	*16-way*	32
Sun Enterprise Cluster 6000 HA	60	Ultra SPARC 1	167 MHz	61 GB	100 MB/s	30-*way*	2
Tandem NonStop Himalaya S70000	4096	MIPS R10000	195 MHz	1000 GB	40 MB/s	*16 way*	256

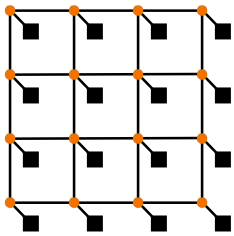
- Exemplo (Cluster investigação da Universidade do Minho - 1999)

Nº máx. CPU	Processador	Freq. Relógio	Memória Máxima	Largura de Banda de comunicação	Nodos	Máximo de nodos
16	Pentium III	400 MHz	6,1 GB	150 MB/s (1,2 Gbit/s)	*2-way*	8

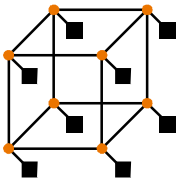
Multiprocessadores

Topologia de redes de interligação

- Completamente conectada
- Anel
- Matriz e hipercubo

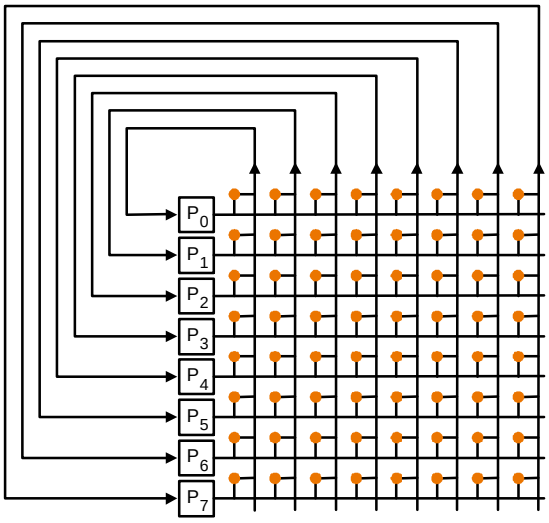


a. 2D grid or mesh of 16 nodes

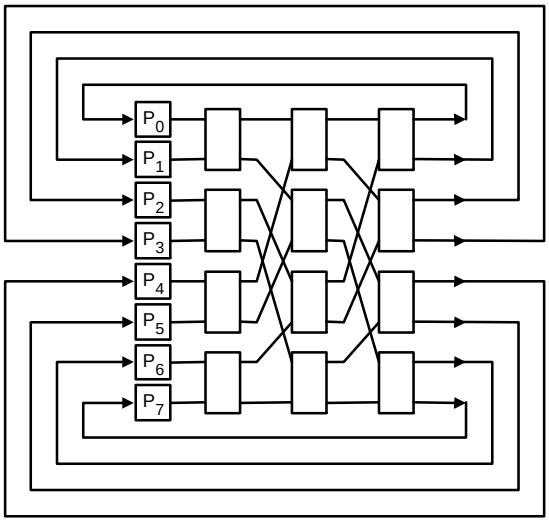


b. n-cube tree of 8 nodes (8 = 2^3 so n = 3)

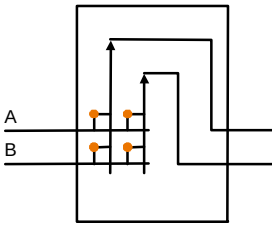
- Multi-nível



a. Crossbar



b. Omega network



c. Omega network switch bc

Métricas de redes de interligação

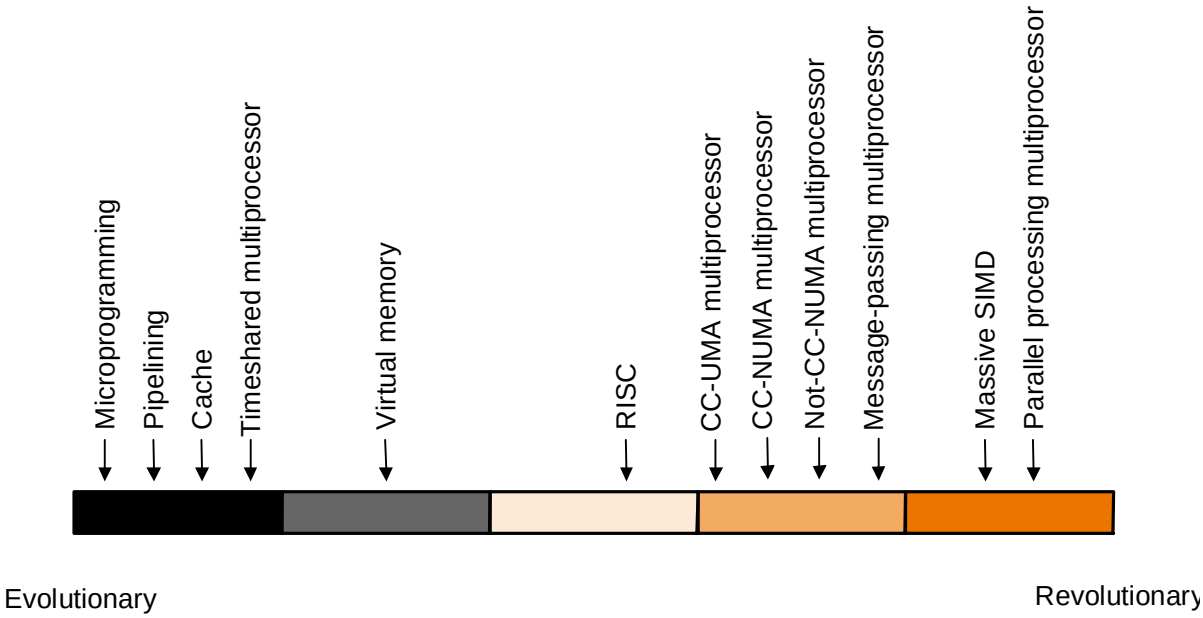
- Largura de banda por ligação
- Largura de banda agregada – soma a largura de banda de todas as ligações que podem ser realizadas simultaneamente
- Largura de banda de bisecção – obtida dividindo a máquina em dois tipos de nodos: receptores e emissores

Futuro da Arquitectura de Computadores

- TOP 500 (os mais potentes computadores do mundo www.top500.org)

Rank	Manufacturer	Computer	R_{max}(GFlops)	Installation Site	Country	Year	Processors
1	IBM	ASCI White,SP Power3 375 MHz	7226.00	Lawrence Livermore National Laboratory	USA	2000	8192
2	Compaq	AlphaServer SC ES45/1 GHz	4059.00	Pittsburgh Supercomputing Center	USA	2001	3024
3	IBM	SP Power3 375 MHz 16 way	3052.00	NERSC/LBNL	USA	2001	3328
4	Intel	ASCI Red	2379.00	Sandia National Labs	USA	1999	9632
5	IBM	ASCI Blue-Pacific SST,IBM SP 604e	2144.00	Lawrence Livermore National Laboratory	USA	1999	5808

- Evolução versus revolução



Bibliografia

- **Secções 9.1, 9.2, 9.3, 9.4, 9.5, 9.6, 9.9, 9.10** de *Computer Organization and Design: the hardware/software interface*, D. Patterson and J. Hennessy, Morgan Kaufmann, 2ª edição, 1998.

- **Adicional:**

  - Secções 8.1, 8.3, 8.4, 8.5 de *Computer Architecture: A Quantitative Approach*, J. Hennessy and D. Patterson, Morgan Kaufmann, 2ª edição, 1996.

