Multiprocessamento

Patterson & Hennessy – Capítulo 9

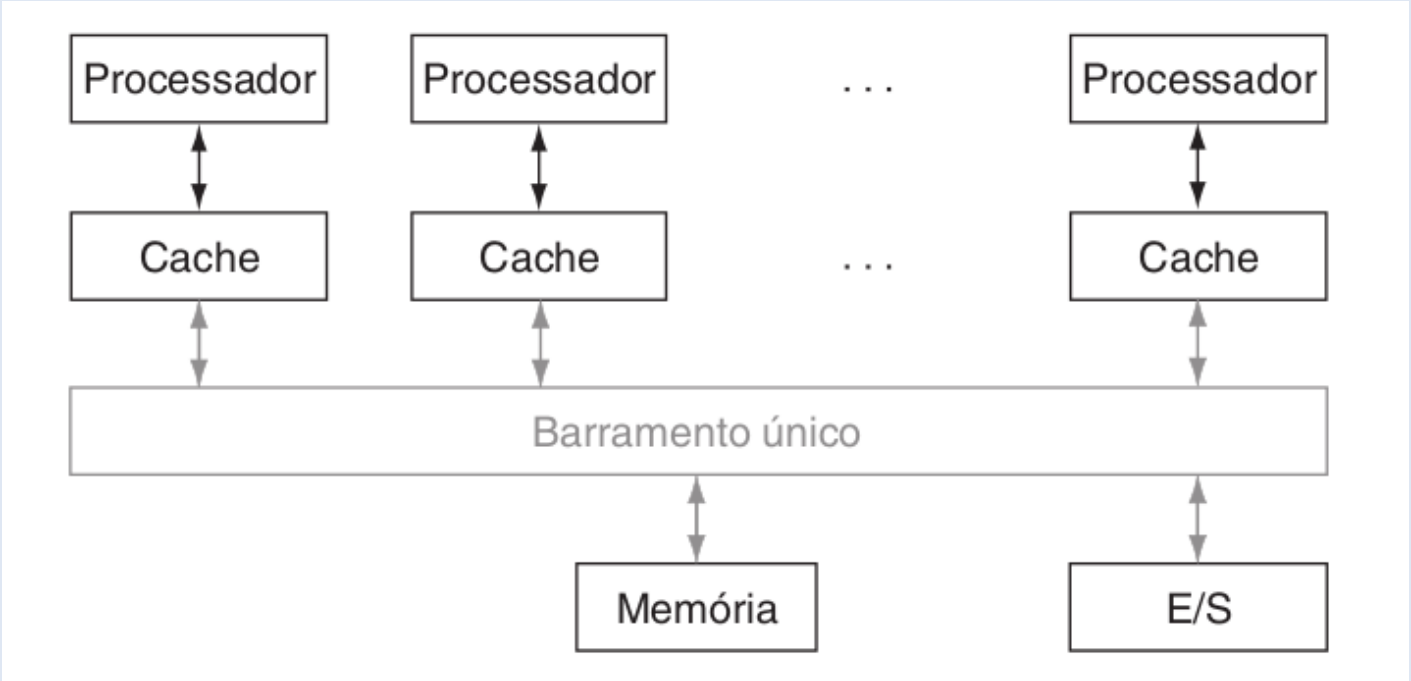
Arquitetura e Organização de Computadores
Juliano M. Vieira (c) 2011

Tópicos Abordados

- Tipos comuns
  - SMP (processamento paralelo)
  - NUMA (placas de alto desempenho)
  - Clusters (computação distribuída e nuvem)
- Comunicação
  - Memória Compartilhada
  - Mensagens
- Coerência de cache
- Multithreading

Symmetrical Multiprocessing (SMP)

- Típico de Multicore
- Um chip, vários núcleos
- Sem memória privada para o µP

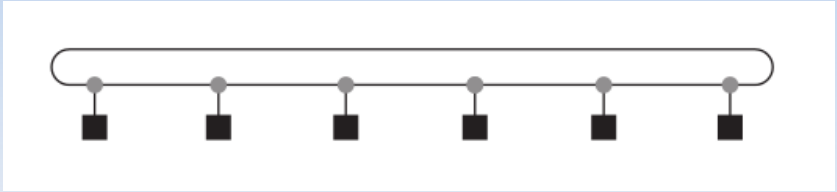


SMP

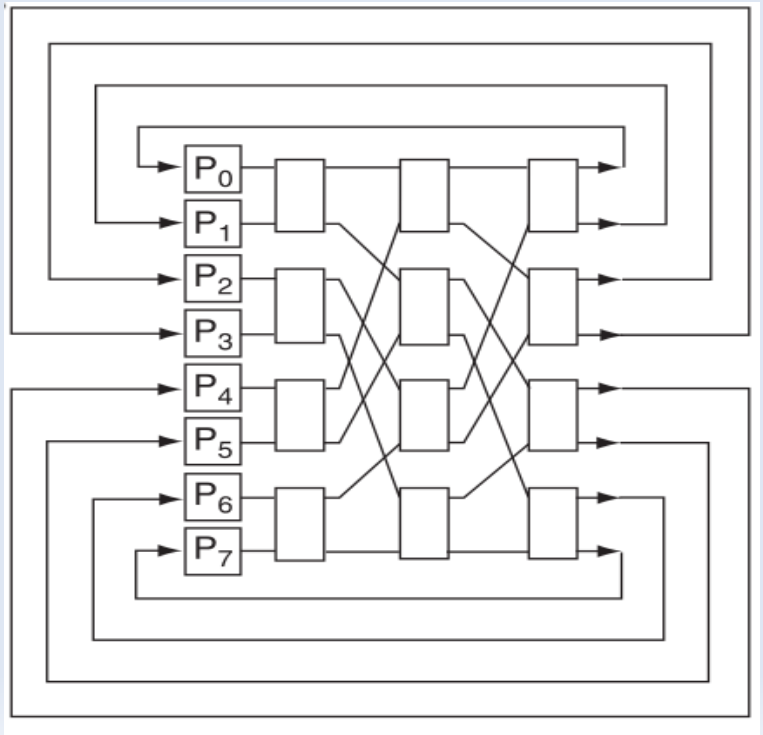
- Taxonomia de Flynn
  - SISD (Single Instruction Single Data)
  - SIMD (Single Instruction Multiple Data)
  - MIMD (Multiple Instruction Multiple Data)
  - MISD (Multiple Instruction Multiple Data)
- SIMD: típico de processamento paralelo
  - Elementos idênticos
  - Mesmo programa em todos os μP
  - Algoritmos: O(O(single) / núm μP)

Algumas Topologias SIMD

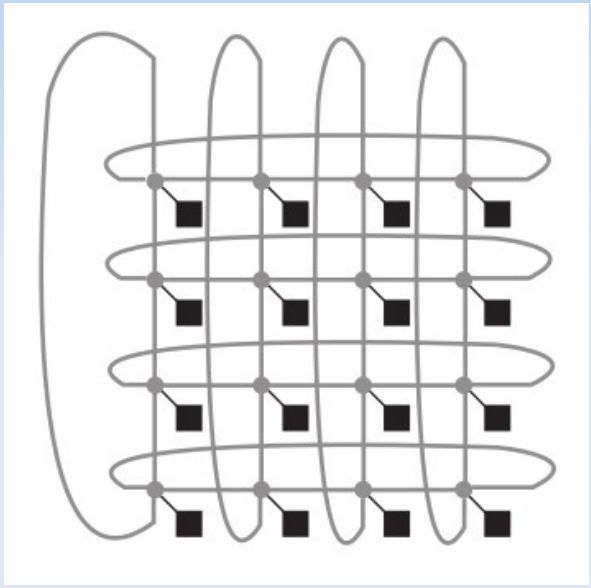
- Anel



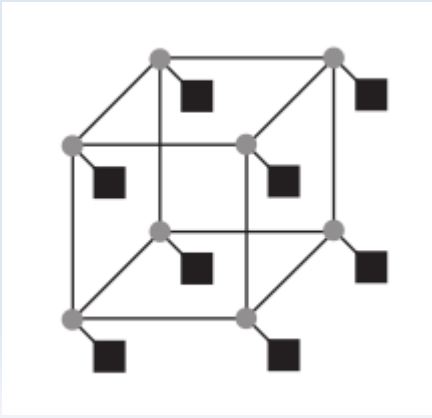
- Rede ômega



- Grade 16 nós



- n-cubo (n=3)

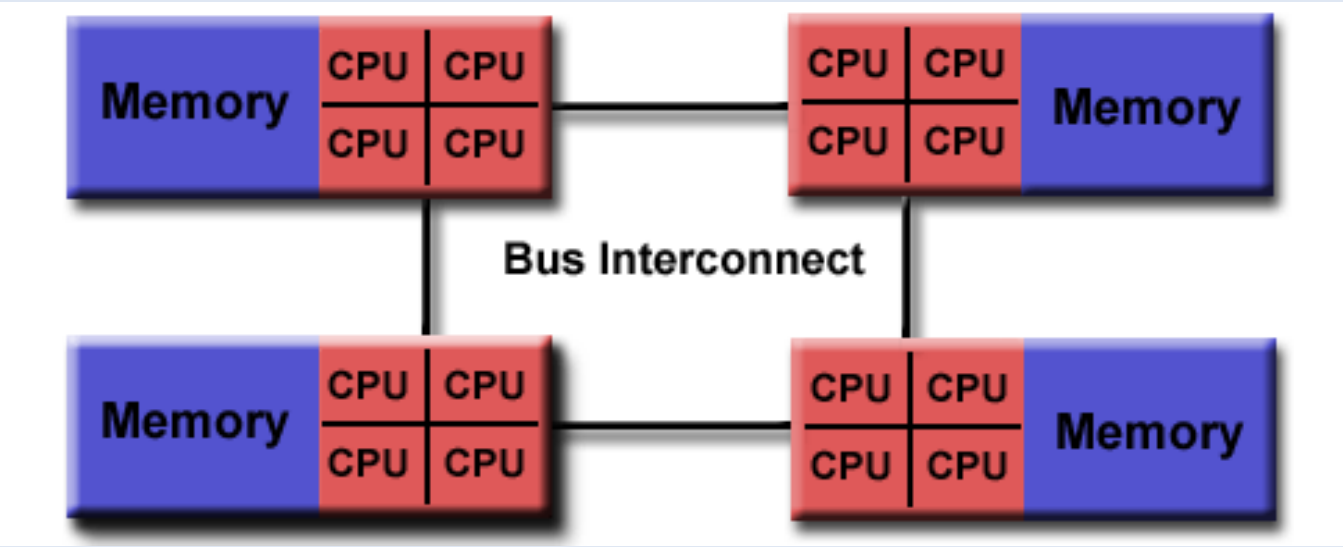


NUMA

- *Non-Uniform Memory Access*
- Tempo de acesso à memória
  - Varia conforme a localização do dado
- Um µP pode ter vantagens sobre outro
- Típico de “multiprocessamento”
  - Uma placa, mais de um chip
- Rede de memória compartilhada distribuída
  - Load/Store remotos: menos *overhead* que send/receive

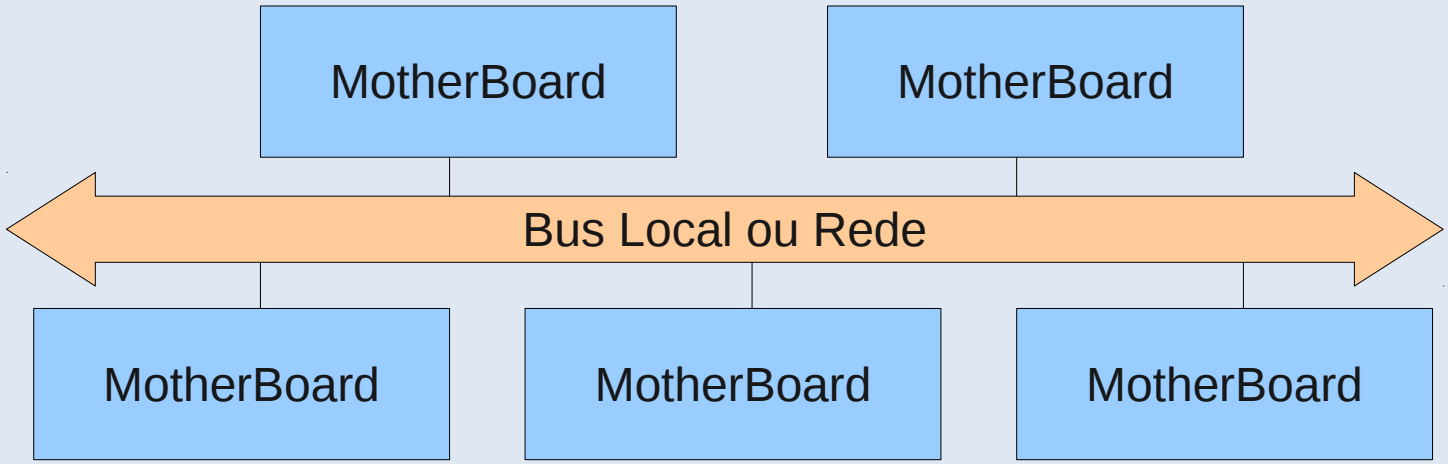
Exemplo de NUMA

- Cada CPU tem uma memória local
- Acesso à memória de outro nó:
  - Indireto, via barramento
  - Bem mais lento que acesso local



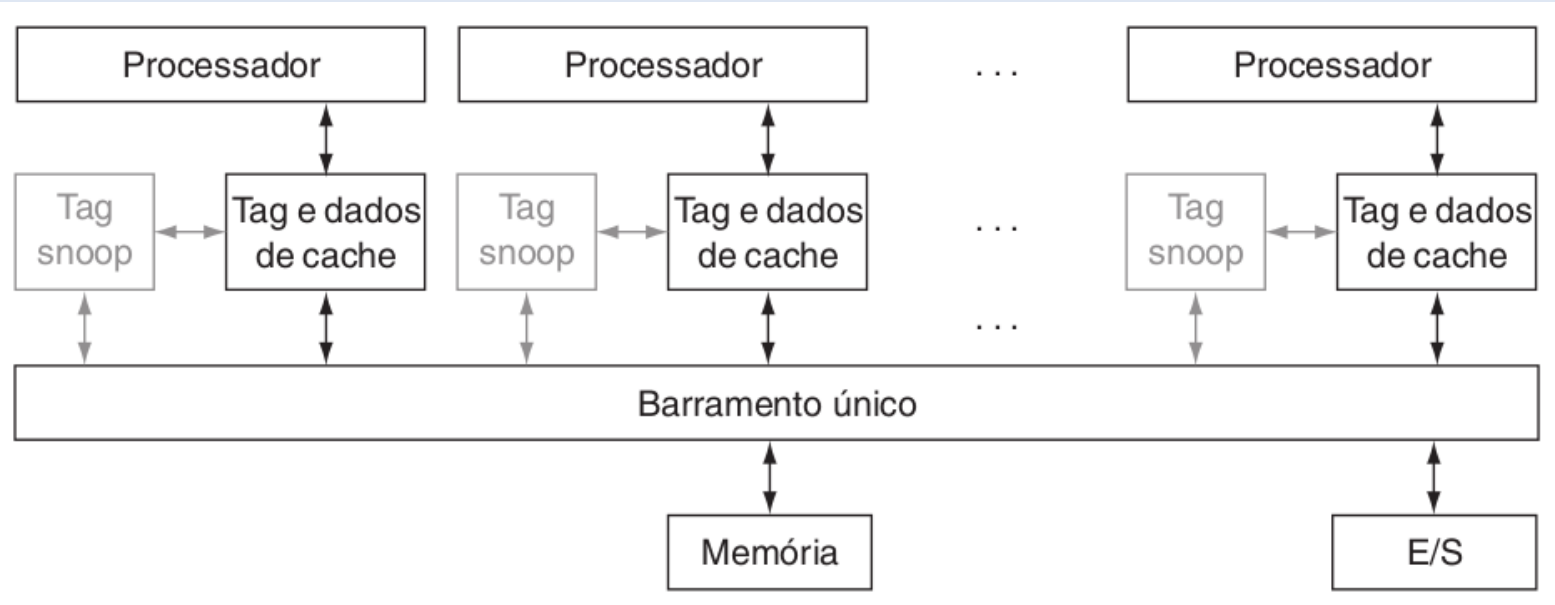
Cluster

- Tipicamente, várias placas mãe
  - Conexão via bus local ou rede
- As placas possuem memórias privadas
- Comunicação via mensagens, usualmente



Coerência de Cache

- Quando um µP altera um dado, outro µP pode estar com uma cópia antiga
  - Problema de consistência dos dados
- Protocolo MESI
- Modified, Exclusive, Shared, Invalid



Exemplo com Memória Compartilhada

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


half = 100; /* 100 processadores */
do{
  synch(); /* espera 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 - quem soma */
  if (Pn < half)
    sum[Pn] = sum[Pn] + sum[Pn+half];
}while (half == 1); /* soma final em Sum[0] */
```

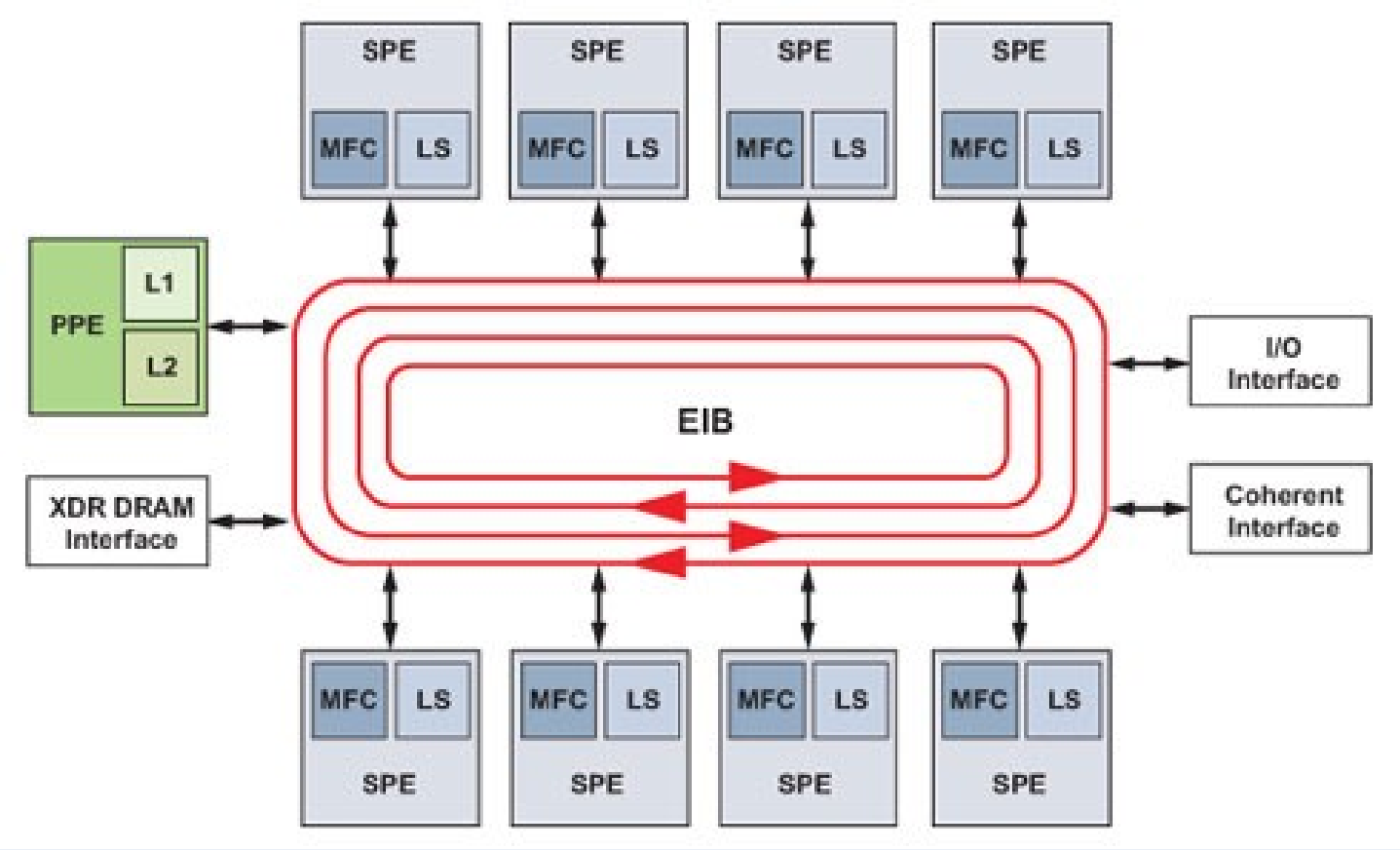
Exemplo com Troca de Mensagens

```
sum = 0;
for (i = 0; i<1000; i = i + 1)
  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 emissores */
}
while (half == 1); /* sai com a soma final */
```

Processador Cell (derivado do PowerPC)

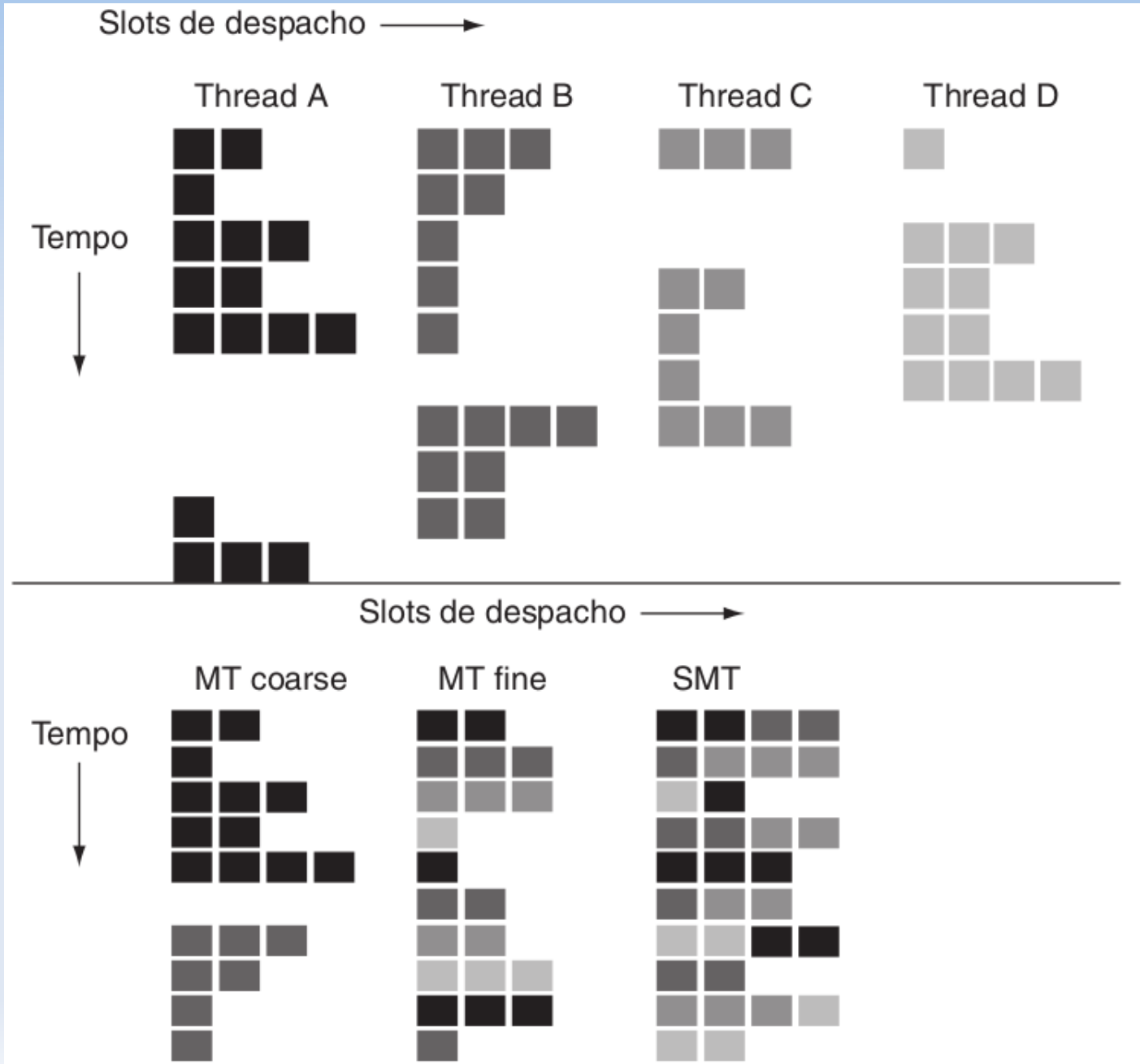
- Usado em videogames e supercomputação
- PPE: processador genérico, controla os outros
- SPE: processadores específicos, subtarefas



Multithreading

- Um processo é um programa
  - Memória privada, comunicação com outros processos (IPC) “complexa”
  - Vários processos simultâneos num Sist. Op.
- Uma thread é um “processo light”
  - Um processo pode ter várias threads
  - Todos acessam a mesma área de memória
  - Comunicação mais fácil
  - Podem ser executadas em simultâneo
- Várias formas de multihtreading (SW, HW)

Suporte do µP a Multithreading



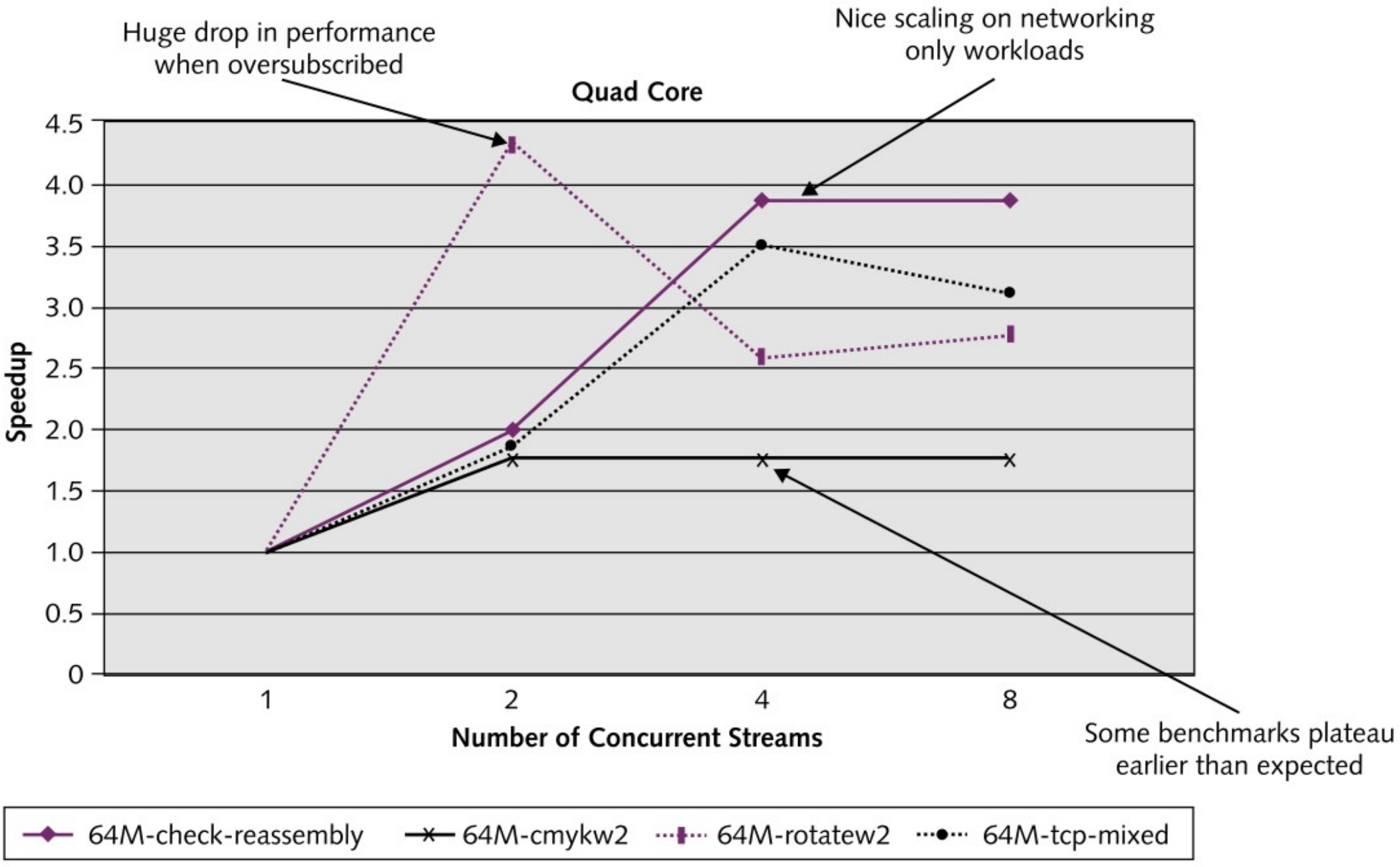
No Meu Laptop!

- Intel Core i5-460M
- Dois núcleos no encapsulamento (dois μP)
- Intel Hyperthreading: Multithreading SMT
- Tanto Windows como Linux enxergam 4 processadores
  - No chaveamento, o processo/thread vai para o processador mais desocupado
  - Threads são praticamente processos (Linux)
  - Duas threads de um programa não necessariamente acessam a mesma cache!

Benchmarking

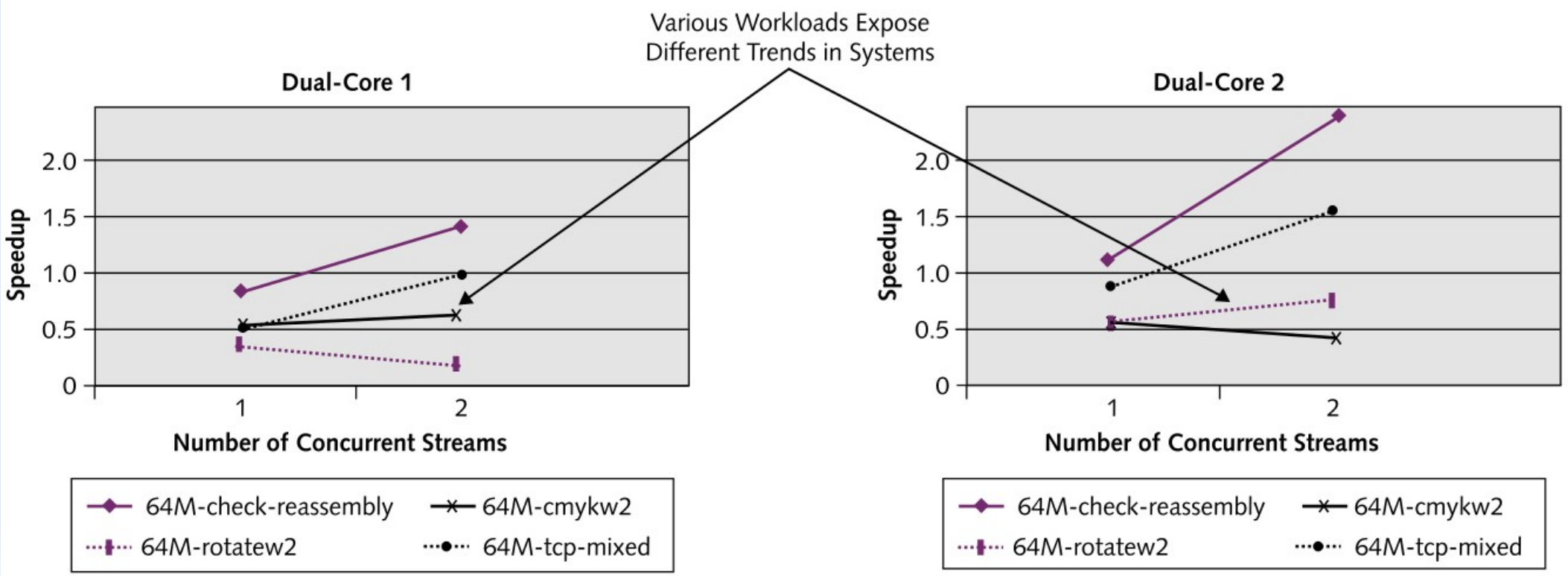
- Desempenho agora depende de:
  - Quantidade de processadores
  - Escalabilidade do algoritmo
    - Problema divisível?
    - Memória compartilhada
  - Sistema operacional vs Processador vs Algoritmo
  - Velocidade? Consumo? Trade off?
- EEMBC (*embassy*)
  - Consórcio para benchmarking de sistemas embarcados

Dimensionamento vs. Algoritmo



Dimensionamento vs. Processador

- Benchmarks: coleção de tarefas
- Desempenho depende também do processador!



Futuro

- A Intel e a Universidade da Califórnia prevêem aumento significativo
  - Centenas ou até milhares (!) de cores num PC comum
- Exigirá algoritmos especializados
- Limites inerentes de paralelização?
  - A lei de Amdahl não tem extensão clara