Organização de Computadores II
DCC007

Aula 14 – Introdução às Máquinas Paralelas

Prof. Omar Paranaiba Vilela Neto

Motivações para Arquiteturas Paralelas

- Desempenho
- Custo
- Disponibilidade e confiabilidade

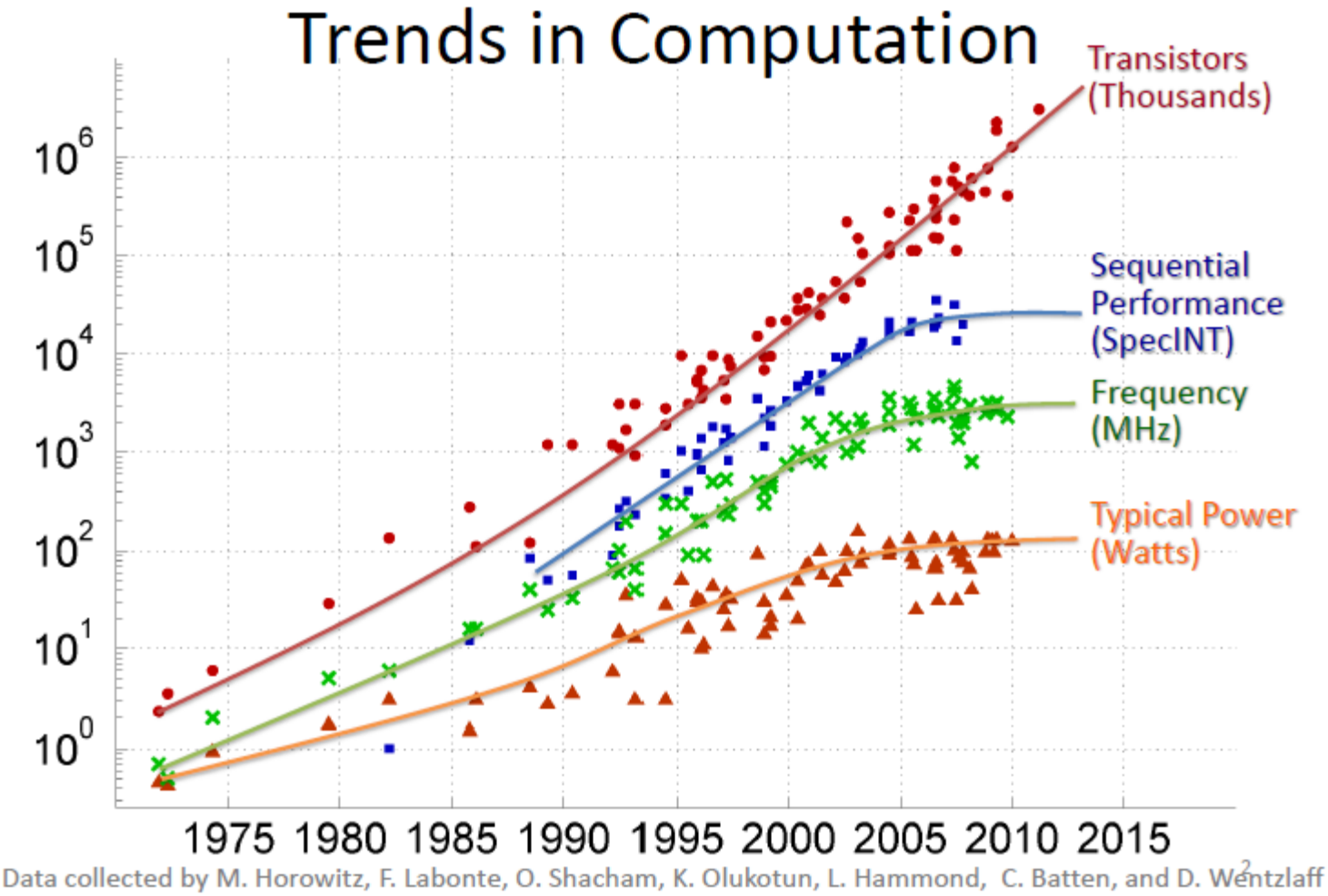
Arquitetura Paralela

- Máquina Paralela: conjunto de elementos de processamento (PEs) que cooperam para resolver problemas computacionalmente difíceis rapidamente
  - Quantos PEs?
  - Poder computacional de cada PE?
  - Quanta memória em cada PE?
  - Como o dado é transmitido entre PEs?
  - Quais as primitivas para cooperação?
  - Qual o desempenho?
  - Como a máquina escala?

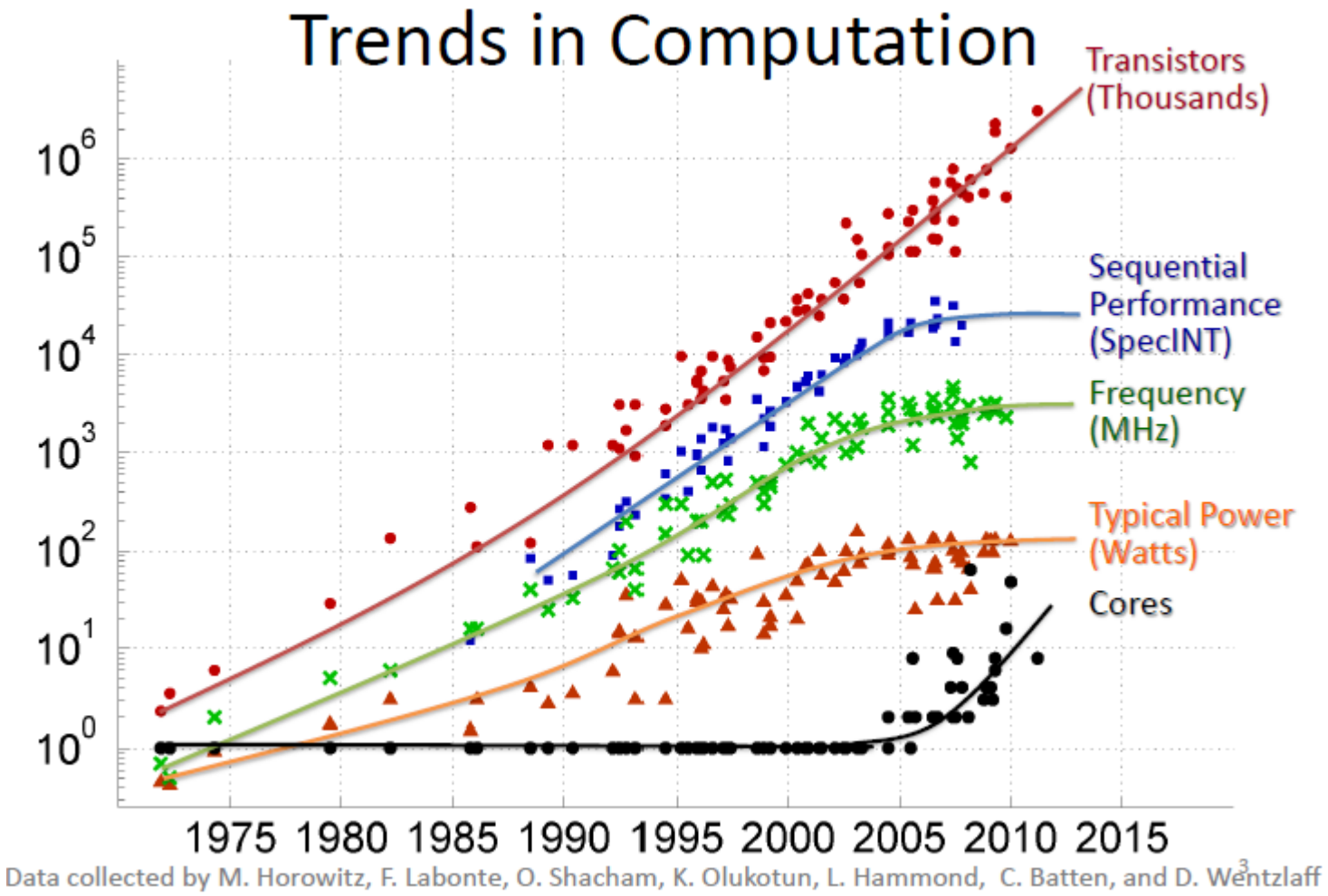
Por que Estudar Arquiteturas Paralelas

- Alternativa para clocks mais rápidos para melhorar desempenho
  - já vimos isso também em processadores superescalares
- Aplica-se em todos os níveis do projeto de sistemas
- Abriu novas perspectivas para **arquitetura de computadores**
  - Por que não colocar múltiplos processadores em um um único chip ao invés de colocar um processador superescalar?

Por que Estudar Arquiteturas Paralelas



Por que Estudar Arquiteturas Paralelas



Arquiteturas Paralelas

- Demanda das aplicações: progresso contínuo de software paralelo
  - Explorar ao máximo o tempo de CPU
  - Computação científica: Biologia, Química, Física, **Nanotecnologia**
  - Computação de uso geral: Vídeo, Computação Gráfica, CAD, Banco de Dados, **Big Data, IA**

- Tendências da tecnologia e das arquiteturas
  - Tecnologia
    - # de transistores crescendo **rapidamente**
    - Frequência crescendo **moderadamente**
  - Arquitetura
    - Limites de ILP

- Paralelismo de granularidade grossa, mais viável

Tendências Arquiteturais ILP

- Speedups reportados para processadores superescalares
  - Horst, Harris, Jardine [1990] 1,37
  - Wang, Wu [1988] 1,70
  - Smith, Johnson, Horowitz [1989] 2,30
  - Murakami, ... [1989] 2,55
  - Jouppi, Wall [1989] 3,20
  - Lee, Kwok, Briggs [1991] 3,50
  - Melvin, Patt [1991] 8,00
  - Butler, ... [1991] 17+

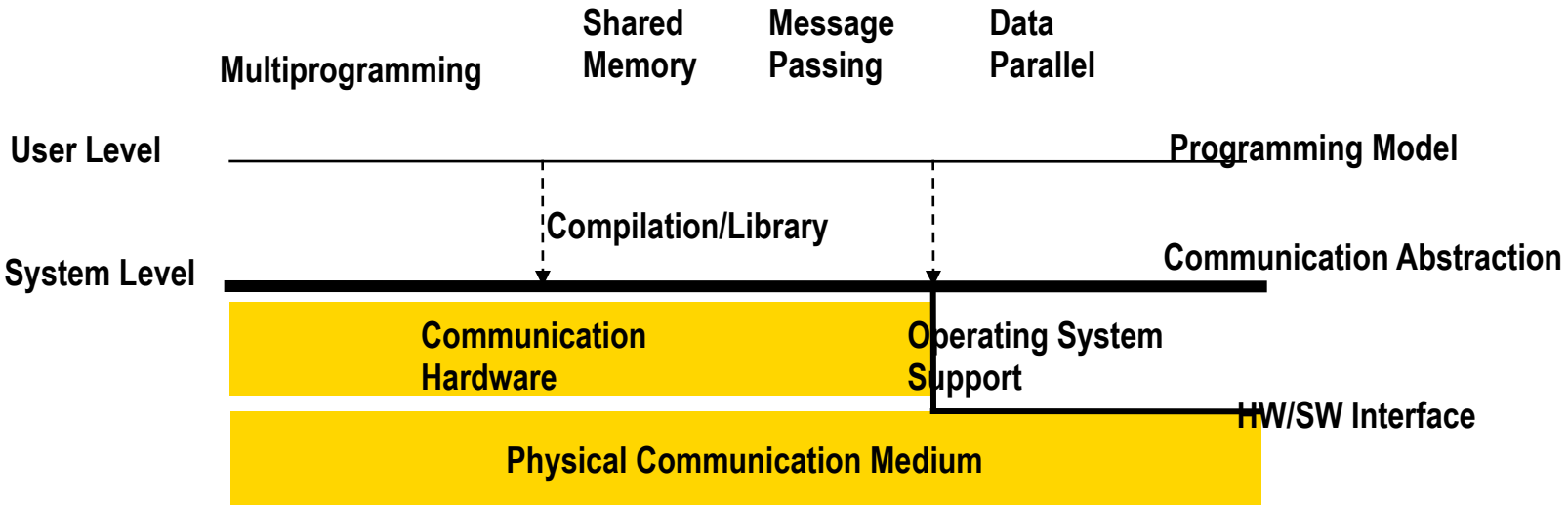
- Melhorias em ILP continuarão a prevalecer nos próximos anos?

Arquiteturas Paralelas e Computação Científica

- Computação Científica
  - uPs conquistaram ganhos altos em desempenho de FP
    - clocks
    - FPUs com pipeline (mult-add todo ciclo)
    - Uso efetivo de caches
  - uPs são relativamente baratos
    - Custo de desenvolvimento de dezenas de milhões de dólares amortizado sobre os milhões de componentes vendidos
- Multiprocessadores usando uPs de grande-escala **substituíram** supercomputadores tradicionais

Arquiteturas Paralelas Hoje

- Extensão de arquitetura de computadores para suportar comunicação e cooperação
  - ANTIGAMENTE: Instruction Set Architecture
  - HOJE: Arquitetura de comunicação



Arquitetura de Comunicação

= Abstração de Comunicação + Implementação

- Abstração:
  - Primitivas de comunicação de HW/SW para o programador (ISA)
  - Modelo de memória compartilhada, baseado em mensagens, ...
  - Primitivas devem ser eficientemente implementadas
- Implementação
  - Onde interface de rede e controlador de comunicação integram no nó?
  - Rede de interconexão
- Objetivos:
  - Aplicações de uso geral (custo/aplicação)
  - Programabilidade
  - Escalabilidade

Considerações Sobre Escalabilidade

- Tanto máquinas small-scale quanto large-scale tem seu lugar no mercado

- Custo-desempenho-complexidade são diferentes para cada máquina

Questões de Projeto

- ***Espaço de endereço***: Como dados compartilhados e/ou comunicação são nomeados?
- ***Latência***: Qual é a latência da comunicação?
- ***Bandwidth***: Quanto dado pode ser comunicado por segundo?
- ***Sincronização***: Como a transferência de dados pode ser sincronizada?
- ***Granularidade***:

  Silício = processador + memória
- ***Aplicabilidade***: Propósito geral ou específico?

Taxonomia Histórica
Classificação de Flynn

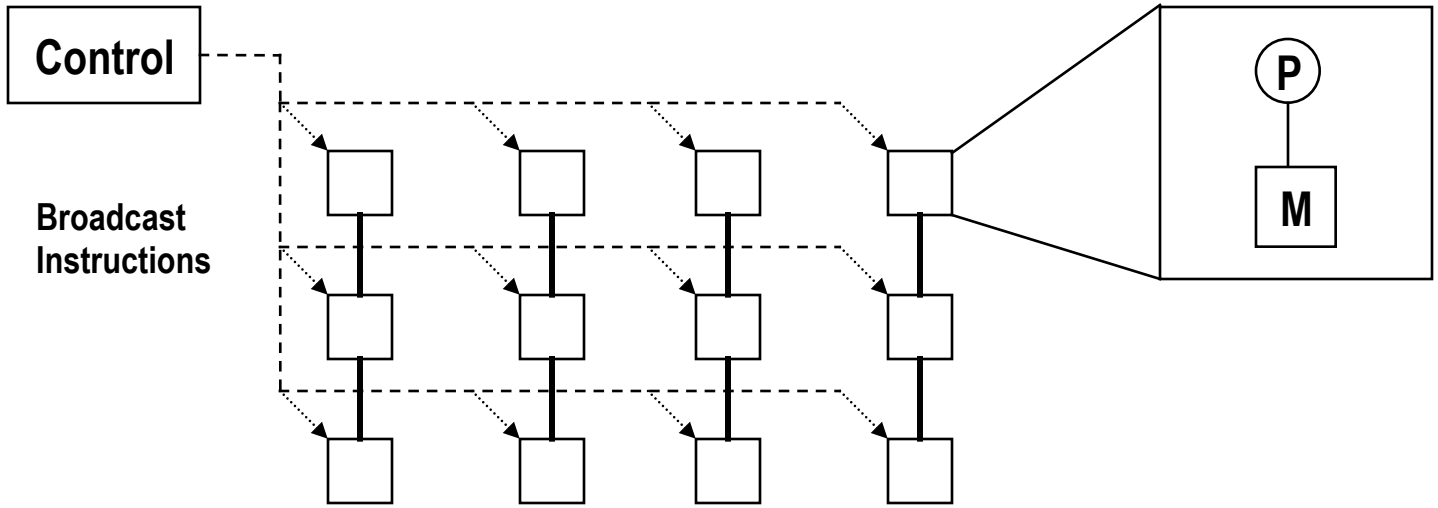
- Baseada nas noções de
  - ***Instruction streams***
  - ***Data streams***

- Organização da máquina é dada pela multiplicidade do hardware para manipular sequências independentes de dados e instruções

  - ***SISD***: Single Instruction and Single Data Stream
  - ***SIMD***: Single Instruction and Multiple Data Streams
  - ***MISD***: Multiple Instructions and Single Data Stream
  - ***MIMD***: Multiple Instructions and Multiple Data Streams

Máquinas SIMD

- Exemplos: GPGPU



- Vantagens: simplicidade de controle, custo, fácil de depurar, baixa latência
- Desvantagens: modelo restrito, pode desperdiçar recursos se somente poucos PEs são utilizados por instrução

SIMD

- Arquitetura SIMD pode explorar significativamente o **paralelismo em nível de dados** para:
  - matrix-oriented computação científica
  - media-oriented processadores de imagem e som

- SIMD é **energeticamente mais eficiente** que MIMD
  - Precisa apenas buscar uma instrução por operação de dados
  - É mais atrativo para aplicações embarcadas

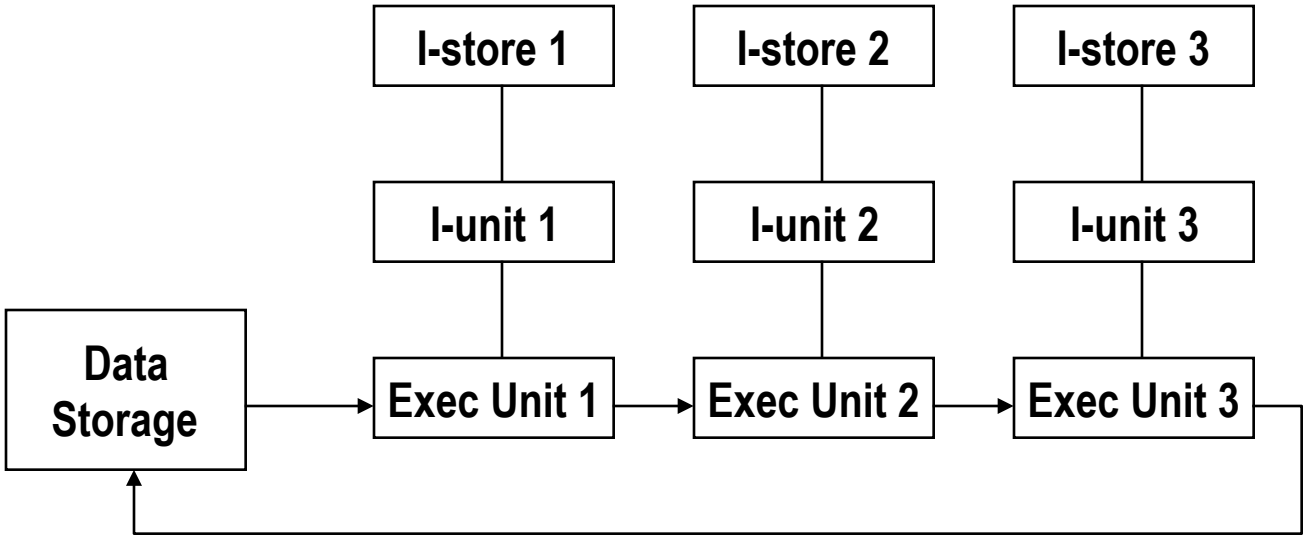
- SIMD permite que o programador continue **pensando sequencialmente**

Paralelismo SIMD

- Arquiteturas Vetoriais
- Extensões SIMD
- Graphics Processor Units (GPUs)

Máquinas MISD

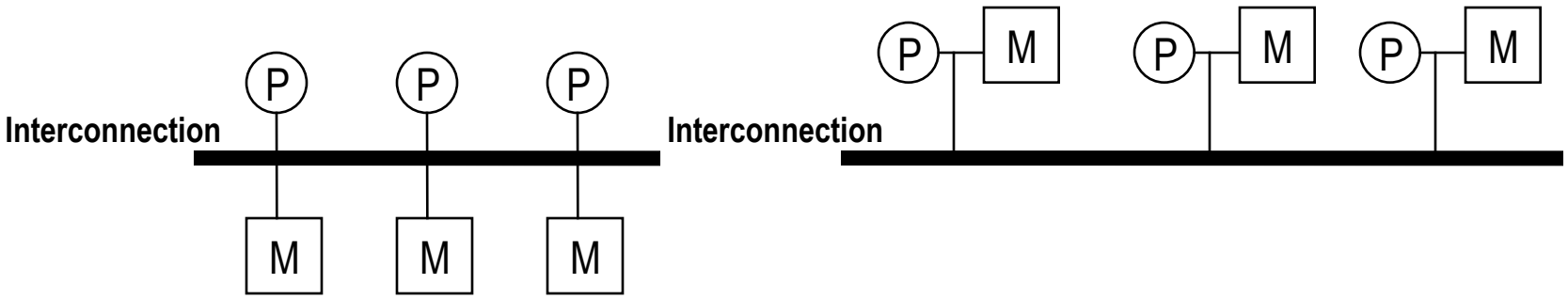
- Exemplo: arrays sistólicos (Warp de CMU)



- Vantagens: simples de projetar, custo-desempenho alto quando pode ser utilizado (processamento de sinais)
- Desvantagens: aplicabilidade limitada, difícil de programar

Máquinas MIMD

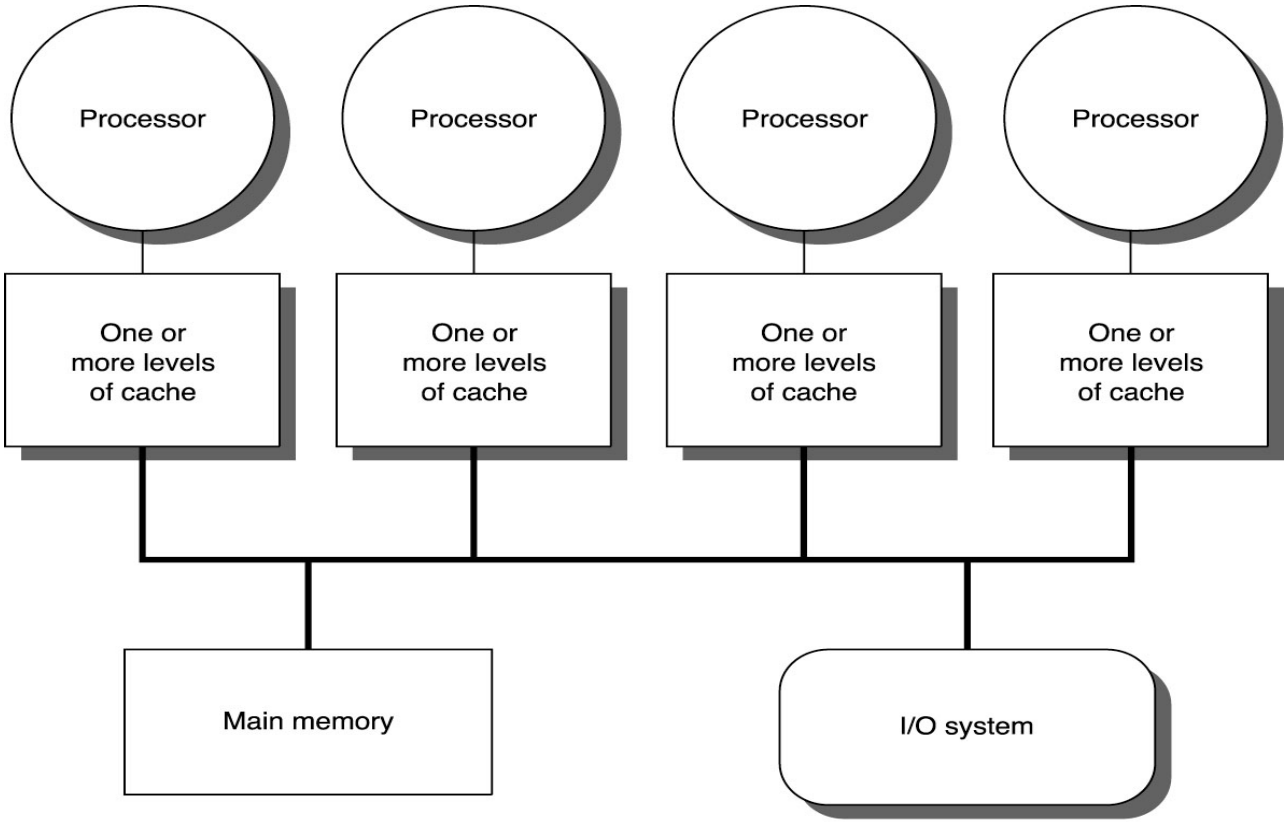
- Exemplo: SGI Challenge, SUN SparcCenter, Cray T3D, Multicores, Clusters....



- Vantagens: aplicabilidade
- Desvantagens: difícil de projetar bem, overhead de sincronização pode ser alto, programação pode ser difícil

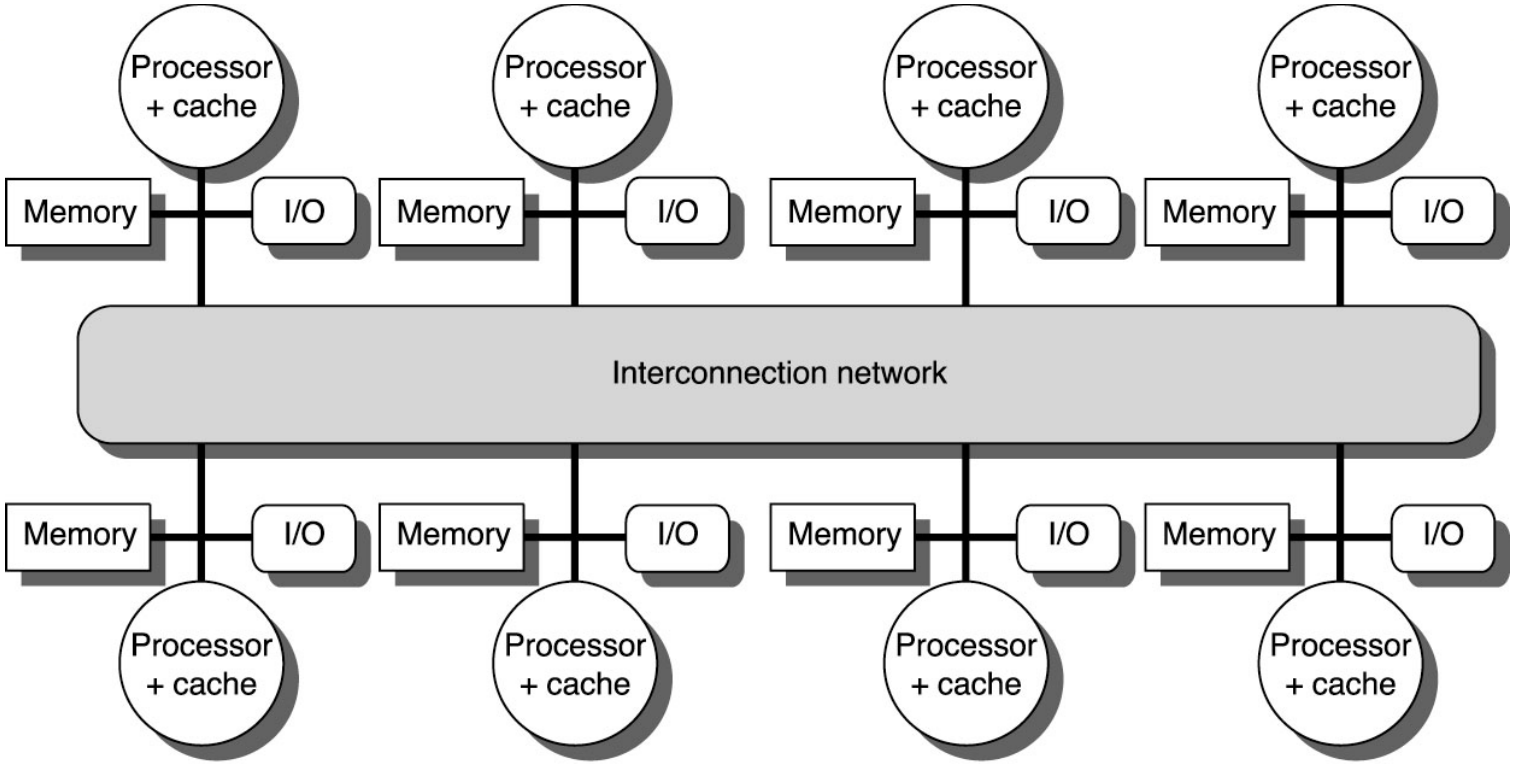
Small-Scale MIMD

- Memória: centralizada com uniform access time (“uma”) e conexão por barramento



Large-Scale MIMD

- Memória: distribuída com nonuniform access time (“numa”) interconexão escalável (distributed memory)

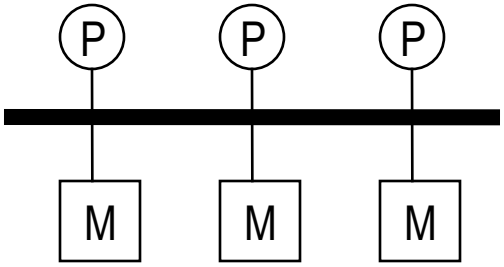


Modelo de Comunicação

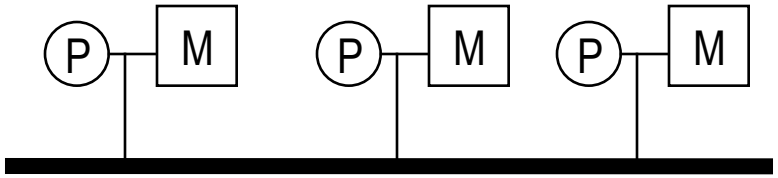
- ***Shared Memory*** (centralizada ou distribuída)
  - Processadores comunicam com espaço de endereçamento compartilhado
  - Fácil em máquinas small-scale
  - Vantagens:
    - Escolhido para uniprocessadores, MPs small-scale
    - Fácil de programar
    - Baixa latência
    - Mais fácil para usar hardware de controle de cache
- ***Message passing (RPC)***
  - Processadores possuem memórias privadas e comunicam-se via mensagens
  - Vantagens:
    - Menos hardware, fácil de projetar
    - Escalabilidade
- HW pode suportar os dois modelos

Arquiteturas com Espaço de Endereçamento Compartilhado

- Todos os processadores podem **acessar** todas as posições de memória do sistema, provendo um mecanismo conveniente e rápido para comunicação



Centralizada
(small-scale)



Distribuída
(large-scale)

- Conhecidas também como shared memory

Modelo de Compartilhamento de Memória

- Comunicação, compartilhamento e sincronização existem via load/store em variáveis compartilhadas

- Modelo de programação relativamente fácil (uniprocessador + sincronização)

- Desvantagem: potencial e escalabilidade

Arquiteturas Baseadas em Troca de Mensagens

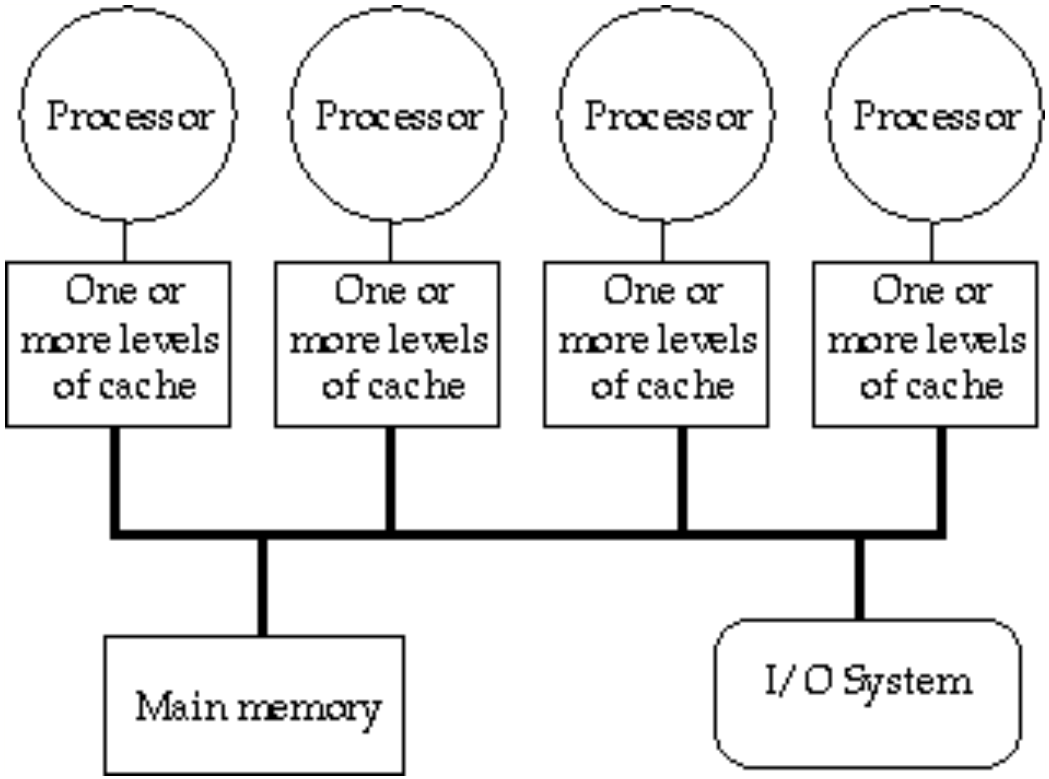
- Processadores só podem acessar memória local, e toda a comunicação e sincronização ocorrem via troca de mensagens explícita

- Fácil de se construir (máquinas uniprocessadoras + redes de comunicação)

- Escalabilidade é alta

Small-Scale—Shared Memory

- Caches servem para:
  - Reduzir necessidade de bandwidth alto entre processador/memória
  - Reduzir latência de acesso
  - Bom para dados privados e compartilhados



- Qual o problema de caches em MPs?

Coerência de Cache

Coerência de Caches

Tempo	Evento	Cache CPU A	Cache CPU B	Memória (X)
0				1
1	A Lê (X)	1		1
2	B Lê (X)	1	1	1
3	A Escreve 0 em (X)	0	1	1

Coerência de Caches

- Sistema de memória é coerente se o dado retornado após uma leitura é o dado que foi escrito mais recentemente

  - *Coerência:* Quais valores podem ser retornados durante uma leitura

  - *Consistência:* Quando valor escrito vai ser retornado durante uma escrita

Mecanismos para Garantir Coerência de Caches

- *Snooping*: Cada cache que possui cópia própria do dado compartilhado possui também o estado do bloco, e nenhum estado centralizado é mantido.
  - Os **controladores da cache ficam** "espionando" o **barramento** compartilhado para verificar o estado atual do bloco

- *Directory based:* O estado do bloco é mantido em um diretório em uma única localização

Arquiteturas Intel

Figure 4. CPU Internals of the Intel® Core™ 2 Duo Processor

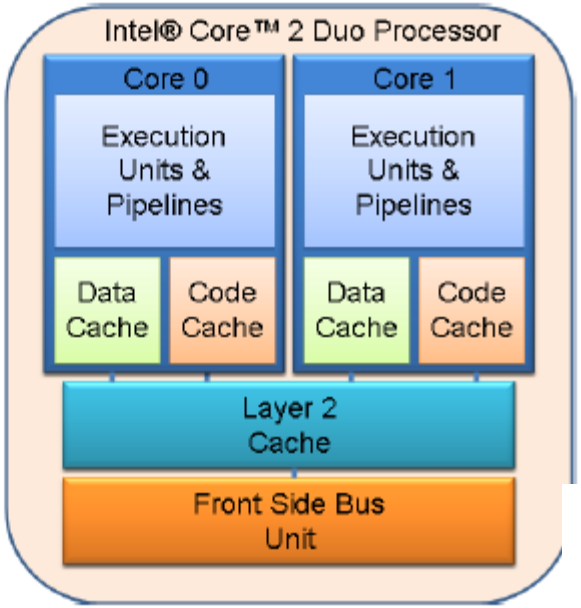


Figure 8. Intel® Core™ i7 Internals