


# Arquitetura e Organização de Computadores II

Multiprocessamento

Prof. Nilton Luiz Queiroz Jr.

# Introdução

- Alguns detalhes fizeram com que o multiprocessamento se tornasse cada vez mais importante:
    - Entre 2000 e 2005 os ganhos alcançados com ILP se tornaram cada vez menos eficientes em relação ao uso de energia e silício;
      - Custos do silício e energia cresceram mais que a performance;
    - Interesse em servidores de alto nível;
    - Interesse em aplicações “data-intensive” uma vez que a quantidade de dados na Internet se tornou muito grande;
    - Melhor conhecimento de como usar multiprocessadores de maneira eficiente;
      - Especialmente para servidores, onde o paralelismo é natural;
        - Códigos científicos;
        - Muitas requisições;
    - O reuso de projetos, reduzindo a necessidade de investimentos para criação de projetos exclusivos;
- 

# Introdução

- Podemos definir multiprocessadores como:
  - Computadores com processadores fortemente acoplados coordenados por um único S.O. que compartilham memória por meio de um espaço compartilhado;



# Introdução

- Multiprocessadores tem como objetivo explorar o paralelismo em nível de thread (TLP) e fazem isso de duas maneiras:
  - Por meio de conjuntos de threads fortemente acopladas colaborando em uma única tarefa;
    - Processamento paralelo;
  - Múltiplos processos relativamente independentes que atendem um ou mais usuários;
    - Processamento a nível de requisição;
      - Podendo ser explorado por :
        - Uma única aplicação executando em diversos processadores;
          - Um sistema de banco de dados atendendo diversas requisições;
        - Várias aplicações rodando independentemente;
          - Multiprogramação;
  - Multiprocessadores podem ser classificados, na taxonomia de Flynn como MIMD;

# Introdução

- Para melhor proveito de multiprocessadores deve-se ter uma quantidade de threads, ou processos, no mínimo igual a quantidade de processadores;
  - A quantidade de threads é geralmente especificada pelo programador ou criada pelo S.O.
    - Diferente o paralelismo em nível de instrução, que é “transparente” para o programador
  - É importante saber explorar o paralelismo em nível de thread;
    - Tamanho da granularidade;
      - Dividir muito a tarefa pode causar overhead;
- Threads podem ser usadas para explorar paralelismo em nível de dados;
  - Tem um overhead maior que computadores SIMD ou GPUs;



# Comunicação

- Multiprocessadores se comunicam e se coordenam através do compartilhamento de memória;
  - Os endereços de memória são compartilhados;
    - Não quer dizer que exista uma única memória física;
- Multiprocessadores podem ser:
  - Um único chip contendo diversos núcleos;
  - Múltiplos chips, cada um contendo seu design multicore



# Tipos de memória

- Os processadores de memória compartilhada existentes se dividem em duas classes:
  - Memória compartilhada centralizada;
  - Memória compartilhada distribuída;

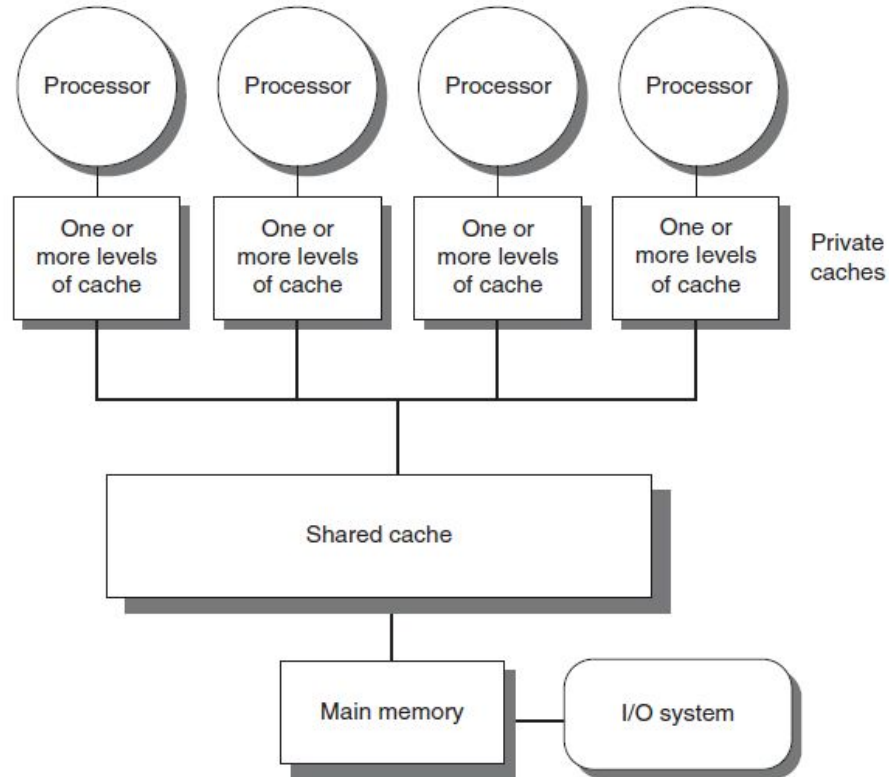


# Memória compartilhada centralizada

- Multiprocessadores de memória compartilhada centralizada são também chamados de :
  - Multiprocessadores simétricos (symmetric multiprocessors - SMP);
  - Multiprocessadores acesso a memória uniforme (uniform memory access - UMA);
- Geralmente compostos por poucos núcleos;
  - Geralmente oito ou menos;
- Devido à pequena quantidade de processadores é possível compartilhar uma memória única e centralizada;
  - Todos processadores têm acesso igual a essa memória;
  - Esse compartilhamento ocorre inclusive com memória organizada em múltiplos bancos;



# Memória compartilhada centralizada



# Memória compartilhada distribuída

- Multiprocessadores de memória distribuída (distributed shared memory - DSM) são também chamados de:
  - Multiprocessadores de acesso não uniforme a memória (nonuniform access memory - NUMA)
- Para suportar mais processadores é necessário que a memória seja distribuída entre eles;
  - Muitos processadores requerem uma largura de banda maior para transportar os dados, o que o sistema de memória não iria suportar a demanda sem causar latências de acesso muito grandes;
- A quantidade de processadores com essas características continua a diminuir com o aumento das capacidades de largura de banda da memória;

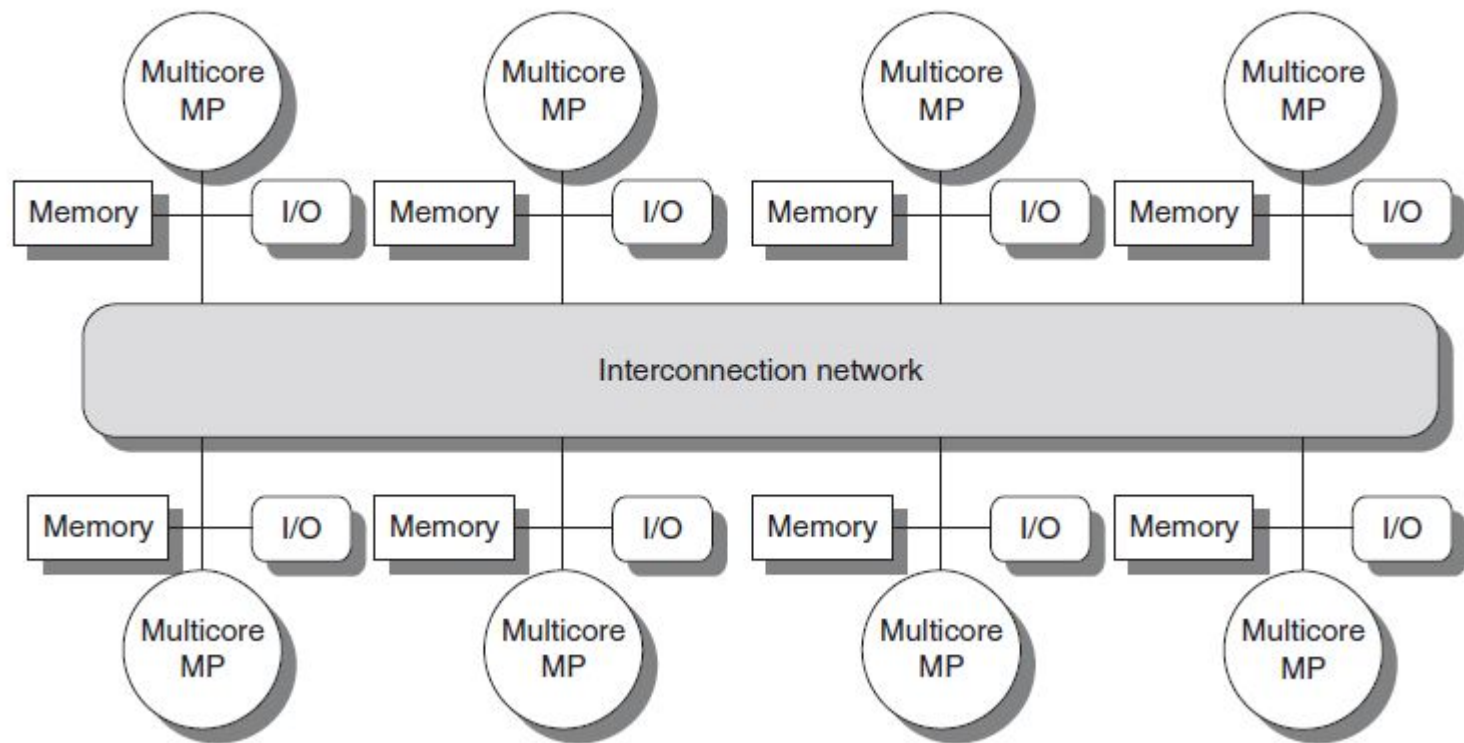


# Memória compartilhada distribuída

- Para a interconexão desses multiprocessadores são usadas redes;
- Distribuir a memória entre os nós aumenta a largura de banda e diminui a latência local;
- O tempo de acesso é não uniforme pois depende da localização dos dados;
- Memórias compartilhadas distribuídas têm comunicação entre os processadores mais complexa;
  - Requer mais esforço em software para tirar vantagem do aumento da largura de banda;



# Memória compartilhada distribuída



# Memória compartilhada

- Tanto em arquiteturas SMP como DSM a comunicação das threads ocorre num espaço compartilhado;
  - Qualquer processador pode fazer acesso a qualquer endereço de memória, desde que tenha permissões;
- Se diferencia dos clusters pois em clusters um processador não tem acesso direto a memória de outro processador;
  - O acesso é feito por meio de protocolos de envios de mensagem;



# Multiprocessamento

- Multiprocessadores podem ser aplicados para se obter desempenho em tarefas independentes (rodando simultaneamente) e tarefas que usam threads.
- Existem dois importantes obstáculos que tornam o processamento paralelo desafiador:
  - O grau de paralelismo disponível nas aplicações;
  - O custo da comunicação nas aplicações;



# Multiprocessamento

- O grau de dificuldade desses obstáculos é dependente de diversas coisas, entre elas:
  - Arquitetura do computador usada na solução do problema;
    - Algumas arquiteturas são mais específicas e se dão melhor em alguns tipos de problemas;
  - A aplicação que se busca paralelizar;
    - Algumas aplicações têm um nível de paralelismo maior que outras;
- O paralelismo disponível nos programas torna difícil alcançar bons speedups em alguns processadores paralelo;



# Exemplo

- Suponha que se deseja alcançar um speedup de 80 com 100 processadores. Qual a fração de computação paralela é necessária?
- Para resolver iremos usar a lei de Amdahl

$$\text{Speedup} = \frac{1}{\frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}} + (1 - \text{Fraction}_{\text{enhanced}})}$$





# Exemplo

- Resolução

$$80 = \frac{1}{\frac{Fraction_{enhanced}}{100}} + (1 - Fraction_{enhanced})$$

$$\left[ \frac{Fraction_{enhanced}}{100} + (1 - Fraction_{enhanced}) \right] \times 80 = 1$$

$$79,2 Fraction_{enhanced} + 80 = 1 \rightarrow Fraction_{enhanced} = \frac{79}{79,2} = 0,9975$$

Obs: na prática os programas usam menos que o complemento total de processadores ao executar no modo paralelo

# Multiprocessadores

- O custo das comunicações envolve a alta latência de acesso em processadores paralelos;
  - No mesmo chip:
    - 30 a 50 ciclos de clock entre os núcleos;
  - Em chips separados:
    - 100 até 500 ciclos, ou mais;
      - Depende do mecanismo de comunicação;
      - Tipo de rede de interconexão;
      - Escala dos multiprocessadores



# Exemplo

- Suponha uma aplicação executando em um multiprocessador com 32 processadores, com tempo de 200ns para lidar com referência a uma única memória remota. Para essa aplicação, considere que todas as referências de memória local. Os processadores ficam em stall em uma solicitação remota, e a frequência do processador é 3,3 GHz. Se o CPI base é 0,5, o quão mais rápido será o multiprocessador quando não faz comunicação se comparado a o multiprocessador quando 0,2% de suas instruções faz referência a comunicação remota?



# Exemplo


- Para resolver devemos calcular a quantidade de ciclos que leva uma solicitação remota:

$$\frac{\text{Remote access cost}}{\text{Cycle time}}$$

- Em seguida, calcular o CPI com solicitação:

$$\text{CPI} = \text{Base CPI} + \text{Remote request rate} \times \text{Remote request cost}$$

- E por fim o speedup:

$$\frac{\text{Base CPI}}{\text{Remote CPI}}$$


# Exemplo


- Ciclos para solicitação remota:

$$\frac{200ns}{\frac{1}{3,3 \times 10^9}} = \frac{200ns}{0,3} = 666 \text{ ciclos}$$

- CPI com solicitação remota:

$$0,5 + 0,002 \times 666 = 1,8$$

- Speedup:

$$\frac{1,8}{0,5} = 3,6$$


# Multiprocessamento

- Para superar tais desafios, os problemas de insuficiência de paralelismo e longas latências podem ser atacados de diferentes maneiras:
  - Insuficiência de paralelismo:
    - Desenvolvimento de algoritmos que podem ter melhor desempenho em paralelo;
    - Sistemas de software que maximizem o tempo gasto na execução com todos processadores;
  - Longa latências:
    - Por software e programador:
      - Reestruturação de dados para maior desempenho com acessos locais
    - Hardware
      - Reduzir frequência dos acessos remotos com caching de dados compartilhados;
      - Pré-busca
        - Busca de mais blocos consecutivos em miss na memória

# Referências

HENNESSY, John L.; PATTERSON, David A. PATTERSON, David A. Computer architecture: a quantitative approach. Elsevier, 2011.

