CAP-387(2016) – Tópicos Especiais em Computação Aplicada: Construção de Aplicações Massivamente Paralelas

Aula 6: Desempenho e Memórias Cache

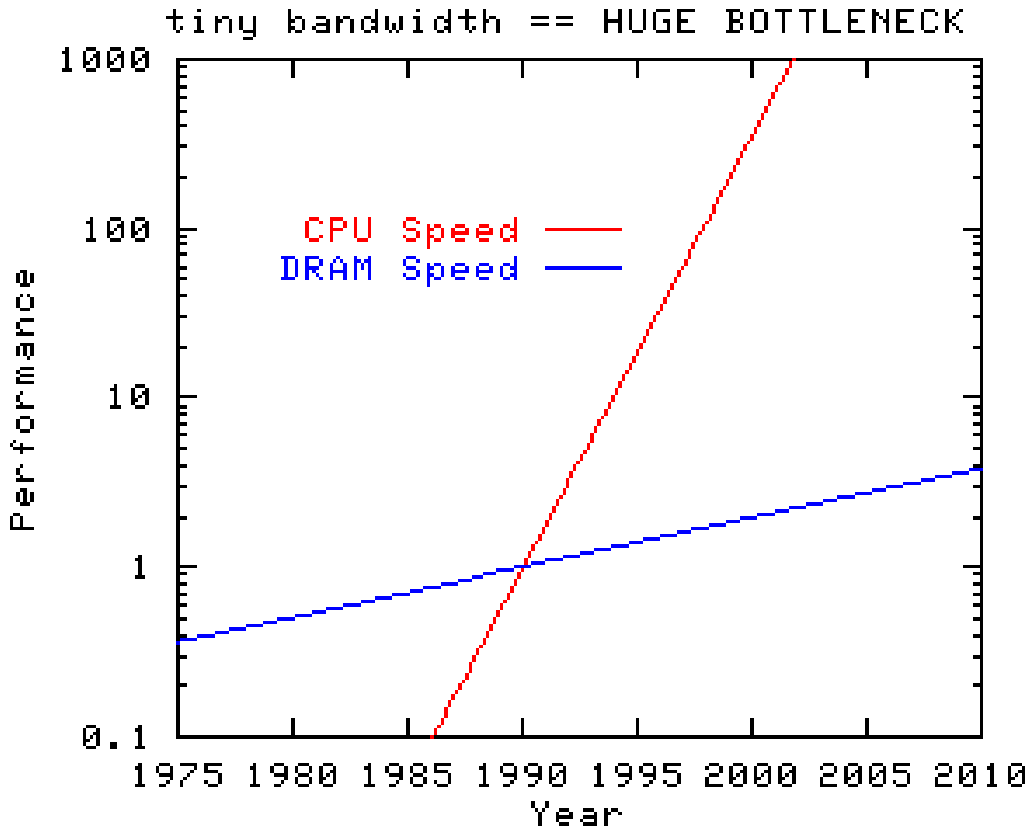
Celso L. Mendes, Stephan Stephany

LAC /INPE

Emails: celso.mendes@inpe.br, stephan.stephany@inpe.br

Desempenho e Memória

- Tendências Atuais de Desempenho



→ Desempenho de CPU evolui bem mais rapidamente que o de DRAM (memória principal)

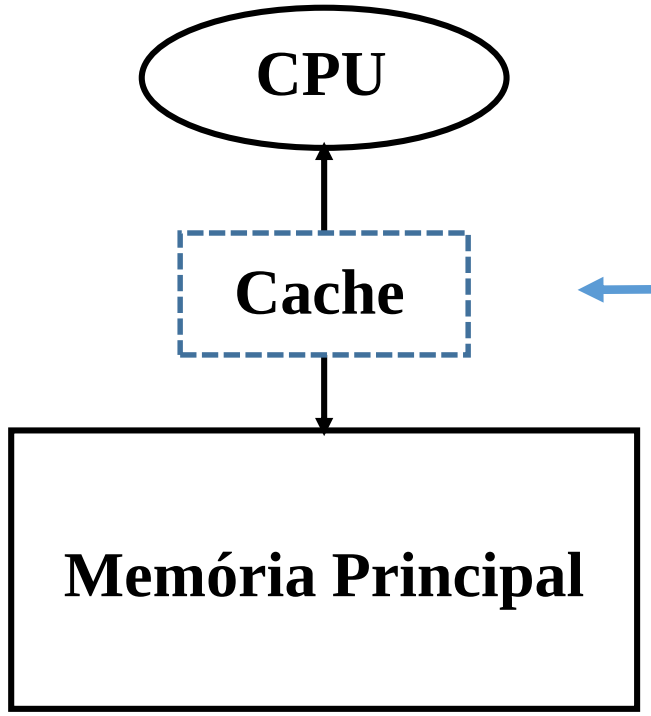
→ Cenário ainda válido nos dias de hoje!

Fonte: John McCalpin / Stream website

Desempenho e Memória

- Problema:
  - Evolução do desempenho das CPUs maior que a evolução de desempenho da memória principal
    - Efeito: desempenho da memória limita o desempenho final
    - Fenômeno conhecido com *memory-wall*
- Solução Universal (popularizada nos anos 80/90):
  - Implementar hierarquia de memória com *caches*
  - "Transparente" aos programas existentes
    - Pode ser ignorada quando o desempenho não for crítico, mas…
    - É essencial para permitir atingir maior desempenho

Nova Arquitetura com Cache

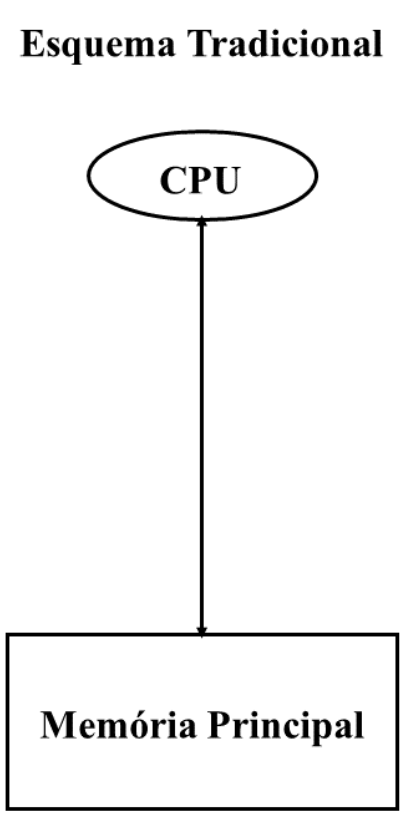


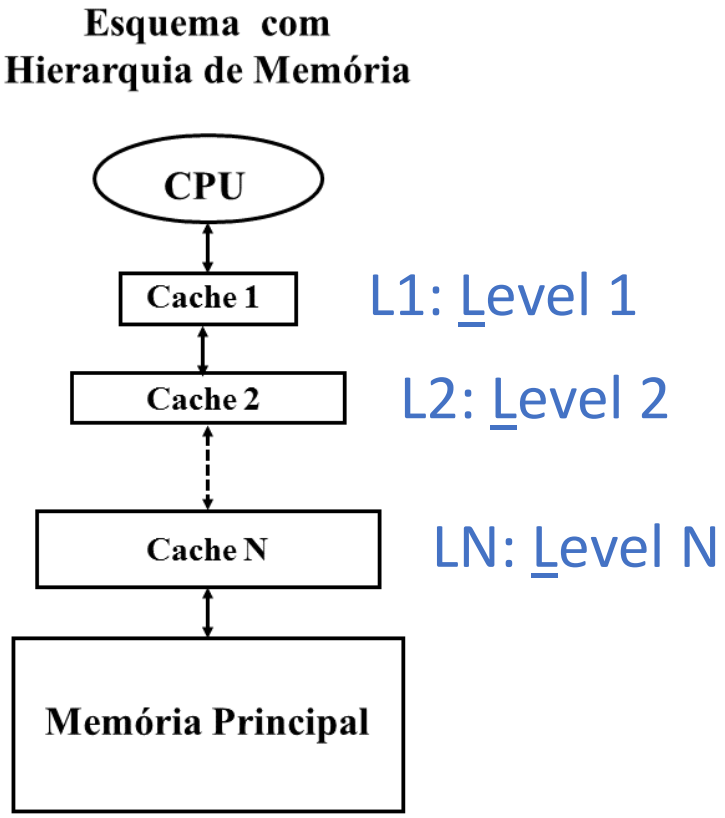
Novo componente:
maior velocidade, menor capacidade que a memória principal

Hierarquia de Memória

- Sistema de memória ideal:
  - Tamanho “ilimitado”, velocidade mais alta possível
- Solução economicamente viável:
  - Criar hierarquia de memória
    - Memórias de diversos tamanhos e velocidades, em níveis
    - Velocidade efetiva próxima à do nível mais alto
    - Tamanho efetivo é o do nível mais baixo
    - Cada nível contém um subconjunto dos dados do nível abaixo
    - Transferência de dados entre níveis é controlada por hardware

Hierarquia de Memória (cont.)

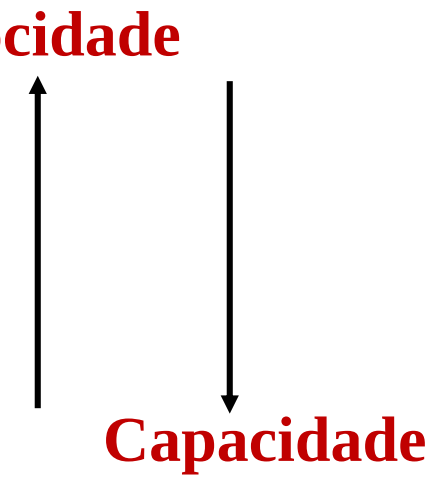




L1: Level 1

L2: Level 2

LN: Level N



Hierarquia de Memória (cont.)

- Tempos de Acesso Típicos em Sistemas Atuais

Nível	Tempo (Ciclos)	Razão p/ Anterior
Registrador na CPU	1	-
Cache L1	alguns	1-3 ×
Cache L2	~ 10	3-10 ×
Memória Principal	~ 250	25 ×
Memória Remota	~ 2500-5000	10-20 ×

Hierarquia de Memória (cont.)

- Efetividade da cache: Princípios da *Localidade*
- Localidade Temporal
  - Se um dado é acessado, deverá ser acessado novamente em futuro próximo
- Localidade Espacial
  - Se um dado é acessado, dados próximos em memória deverão ser acessados em futuro próximo
  - Razão para estruturar a cache em blocos (linhas)
- Ambos os princípios são observados na prática
  - Caches existem em todos os sistemas atuais

Hierarquia de Memória (cont.)

- Processadores modernos
  - Um ou mais níveis de cache são internos ao chip da CPU
    - Velocidade interna bem maior que a externa
  - Pode haver caches separadas para dados e instruções
    - Em geral, caches de instruções são menores
  - Otimização: instruções específicas de pré-carga (*prefetch*)
- Possível problema: consistência dos dados em cache
  - Exemplo: periféricos ligados diretamente à memória podem atualizar a memória em posições que estão na cache
    - Solução: hardware de monitoramento, invalidação dos dados em cache no caso de atualizações da memória

Estrutura Básica de Caches

- Estrutura de armazenamento dos dados:
  - Unidade básica: bloco de dados (conjunto contíguo de bytes)
    - Algumas vezes também chamado de linha
  - Endereço do bloco = subconjunto dos bits de endereço dos bytes
  - Transferências entre níveis sempre ocorrem por blocos
- Política de controle em cada nível:
  - Onde colocar um certo bloco?
  - Como saber se um certo bloco está presente na cache?
  - Qual bloco deve ser retirado ao se trazer um novo bloco?
  - O que acontece durante um *write*?

Desempenho de Caches

- Resultados possíveis em um acesso à memória:
  - *cache-hit*: dado está na cache (acesso rápido)
  - *cache-miss*: dado não está na cache; buscar na memória principal (acesso lento)
- Tempo médio de acesso à memória: $T_m = h + r \times p$
  - h: tempo de um hit (acesso ao dado na cache)
  - r: *miss-rate* (fração de acessos a dados fora da cache)
    - Definido pelo número de *misses* dividido pelo total de acessos
  - p: *miss-penalty* (tempo adicional gasto em um miss)

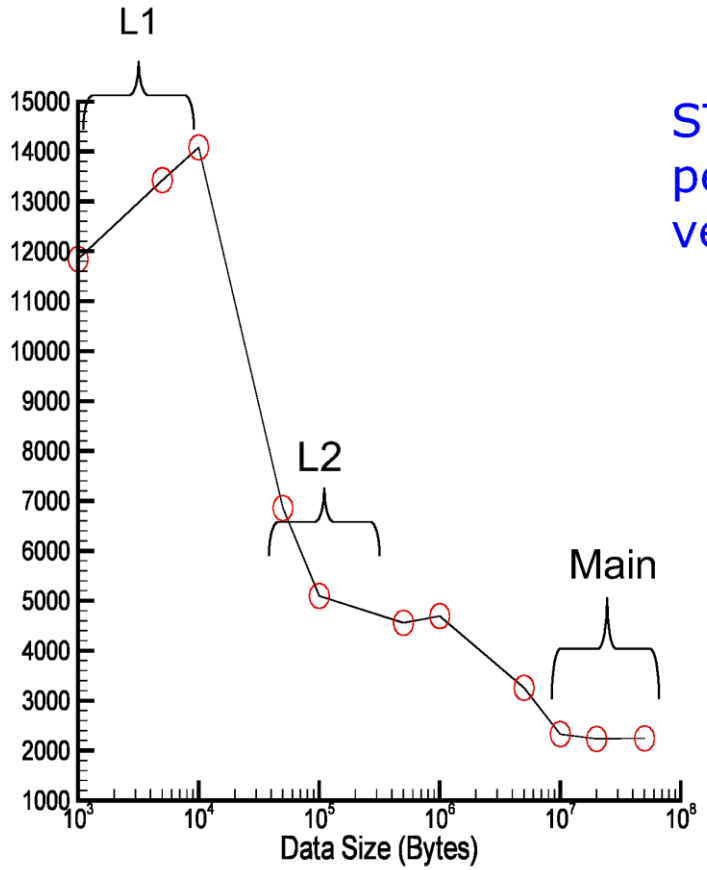
h e p podem ser expressos em segundos ou em ciclos

Otimizações com Caches

- Objetivo básico: minimizar $T_m = h + r \times p$
  - *h* e *p:* em geral, são definidos pelo hardware
  - *r:* tipicamente definido pelo programa (programador!)
- Duas formas principais de reduzir o *miss-rate r*
  - Instruções de *prefetch:* carregar dados na cache antes que tais dados sejam realmente necessários
  - Rearranjo do programa – pelo programador, ou via compilador
    - Requer um conhecimento mínimo da estrutura das caches
    - Não exige nenhum hardware especial
    - Pode causar enorme impacto no desempenho do programa

Stream - Desempenho com Caches

- Desempenho Medido:



STREAM
performance in MB/s
versus data size