Estrutura de Dados

Módulo: Memória Secundária

Prof: Olga Goussevskaia

Conteúdo

- Localidade de Referência
- Memória Virtual e Paginação
- Ordenação Externa
  - Intercalação balanceada
  - Quicksort externo
- Árvores B

Livro de Referência

- **Projeto de Algoritmos** com implementação em Pascal e C. Nivio **Ziviani**. 3a edição.
- Localidade de Referência (não está no livro)
- Memória Virtual e Paginação (6.1)
- Ordenação Externa (4.2)
- Árvores B, B* (6.2, 6.3)

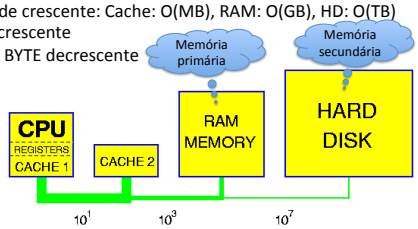
EVOLUÇÃO

SISTEMAS DE MEMÓRIA

Perspectiva Histórica

- Primeiros computadores: processador + fita
- Processadores modernos:

+ Capacidade crescente: Cache: O(MB), RAM: O(GB), HD: O(TB)
+ Latência crescente
- Custo por BYTE decrescente

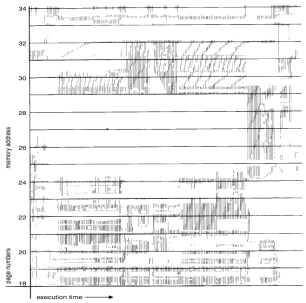


Indicated are approximate numbers of clock cycles to access the various elements of the memory hierarchy

O QUE É?

LOCALIDADE DE REFERÊNCIA

Localidade em padrões de acesso



Localidade de Referência

- Padrões de acesso a dados observados desde os primeiros sistemas de computação:
  - **Temporal**: se um dado é acessado uma vez, há uma probabilidade grande dele ser acessado novamente num futuro próximo.
  - **Espacial**: se um dado é acessado uma vez, há uma probabilidade grande do seu vizinho ser acessado em breve.

Localidade de Referência

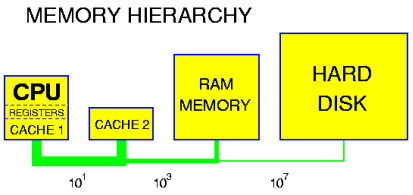
- Estrutura mais clássica de programação:
  - Loop:

```
for (i = 0; i < n; i++) {
    vetor[i] = fun(vetor[i]);
}
```

- Localidade de referência espacial:
  - vetor[0], vetor[1], ..., vetor[n-1]
- Localidade de referência temporal:
  - As instruções compondo o corpo do loop
- Mais um exemplo: espacial ou temporal?
  - Árvores: os nós são sempre acessados pela raíz

Localidade de Referência Temporal

- Quanto maior, melhor!
- Uma vez trazido para a o topo da hierarquia, o custo de acessos seguintes (tempo para ler o dado) a um dado é bem mais baixo.



Indicated are approximate numbers of clock cycles to access the various elements of the memory hierarchy

Localidade de Referência Espacial

- Como pode ser explorada?
  - Estruturas baseadas em paginação e segmentação
    - Em hardware, a partir da arquitetura 386 (segmentation fault)
  - Paginação: Acessos em blocos, de tamanho:
    - L1: 128 bytes, L2: um pouco maior
    - RAM: 4 a 8 KB
    - Disco: 4 a 8 KB

Isso torna os sistemas de memória modernos eficientes!

Custo alto: ex: buscar na memória

Custo mais baixo: os vizinhos de a1 são trazidos junto com a1

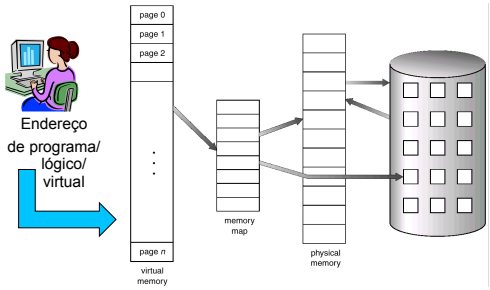
Localidade de referência: resumo

- Localidade de Referência Temporal:
  - Sistemas de hierarquia de memória (caching)
- Localidade de Referência Espacial:
  - Estruturas de paginação
- Projeto de um sistema de memória secundária
  - Dimensionamento de cada nível de memória
    - Benefício x custo
  - Dimensionamento de páginas
    - Pequenas demais: muitos "misses", poucos "hits"
    - Grande demais: fragmentação: memória ocupada por dados que não serão acessados

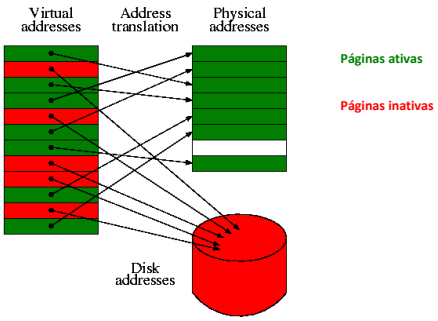
Computação em Memória Secundária

- O que fazer quando os dados (e.g. vetor a ser ordenado) não cabem na memória principal?
- Como implementar de forma eficiente a interface entre dois níveis de memória, e.g., RAM e disco?
- Solução: VIRTUALIZAÇÃO DE MEMÓRIA

Sistema de Memória Virtual



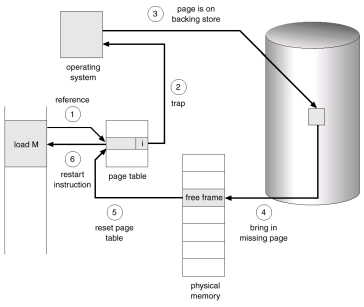
Memória Virtual



Falha de página (*page fault*)

- Na primeira referência a uma página: TRAP
- S.O. consulta tabela de paginação virtual
  - Referência inválida → aborta o processo
  - Pág. ausente → localiza a página no disco
- Obtém um quadro/frame vazio
- Carrega a página no quadro
- Atualiza a tabela de páginas
- Reinicia a instrução que gerou a página

Falha de página (*page fault*)



E se não houver um quadro vazio?

- Substituição/reposição de páginas
  - Encontre uma página que não esteja em uso
  - Libere o quadro (pode exigir escrita no disco)
- Desempenho:
  - Página retirada pode vir a ser acessada de novo
  - Deseja-se um algoritmo que minimize as falhas de paginas (traps)
- Ideal: remover a página que não será referenciada pelo período de tempo mais longo no futuro
- Tentamos inferir o futuro a partir do comportamento passado

Memória Virtual: Políticas de Reposição de Páginas

- **Menos Recentemente Utilizada (LRU):**
  - um dos algoritmos mais utilizados,
  - remove a página menos recentemente utilizada,
  - parte do princípio que o comportamento futuro deve seguir o passado recente.
- **Menos Freqüentemente Utilizada (LFU):**
  - remove a página menos feqüentemente utilizada,
  - inconveniente: uma página recentemente trazida da memória secundária tem um baixo número de acessos e pode ser removida.
- **Ordem de Chegada (FIFO):**
  - remove a página que está residente há mais tempo,
  - algoritmo mais simples e barato de manter,
  - desvantagem: ignora o fato de que a página mais antiga pode ser a mais referenciada.

Sistema de memória virtual: resumo

- Cria a ilusão de um espaço de memória maior e contíguo
- Normalmente implementado como uma função do sistema operacional
- Modelo de armazenamento em 2 níveis: pequena qntde de mem. principal e grande qntde de mem. secundária
- Tarefas básicas:
  - Mapeamento de endereços entre dois níveis de memória:
  - Transferência eficiente de páginas entre os dois níveis de memória: “qual página remover qdo não há espaço na memória?”
- Funciona bem para algoritmos que possuem uma localidade de referência espacial pequena.