Capítulo 9: Memória Virtual

Sobre a apresentação (About the slides)

Os slides e figuras dessa apresentação foram criados por Silberschatz, Galvin e Gagne em 2005. Esse apresentação foi modificada por Cristiaino Costa (cac@unisinos.br). Basicamente, os slides originais foram traduzidos para o Português do Brasil.

É possível acessar os slides originais em http://www.os-book.com

Essa versão pode ser obtida em http://www.inf.unisinos.br/~cac

The slides and figures in this presentation are copyright Silberschatz, Galvin and Gagne, 2005. This presentation has been modified by Cristiano Costa (cac@unisinos.br). Basically it was translated to Brazilian Portuguese.

You can access the original slides at http://www.os-book.com

This version could be downloaded at http://www.inf.unisinos.br/~cac

Capítulo 9: Memória Virtual

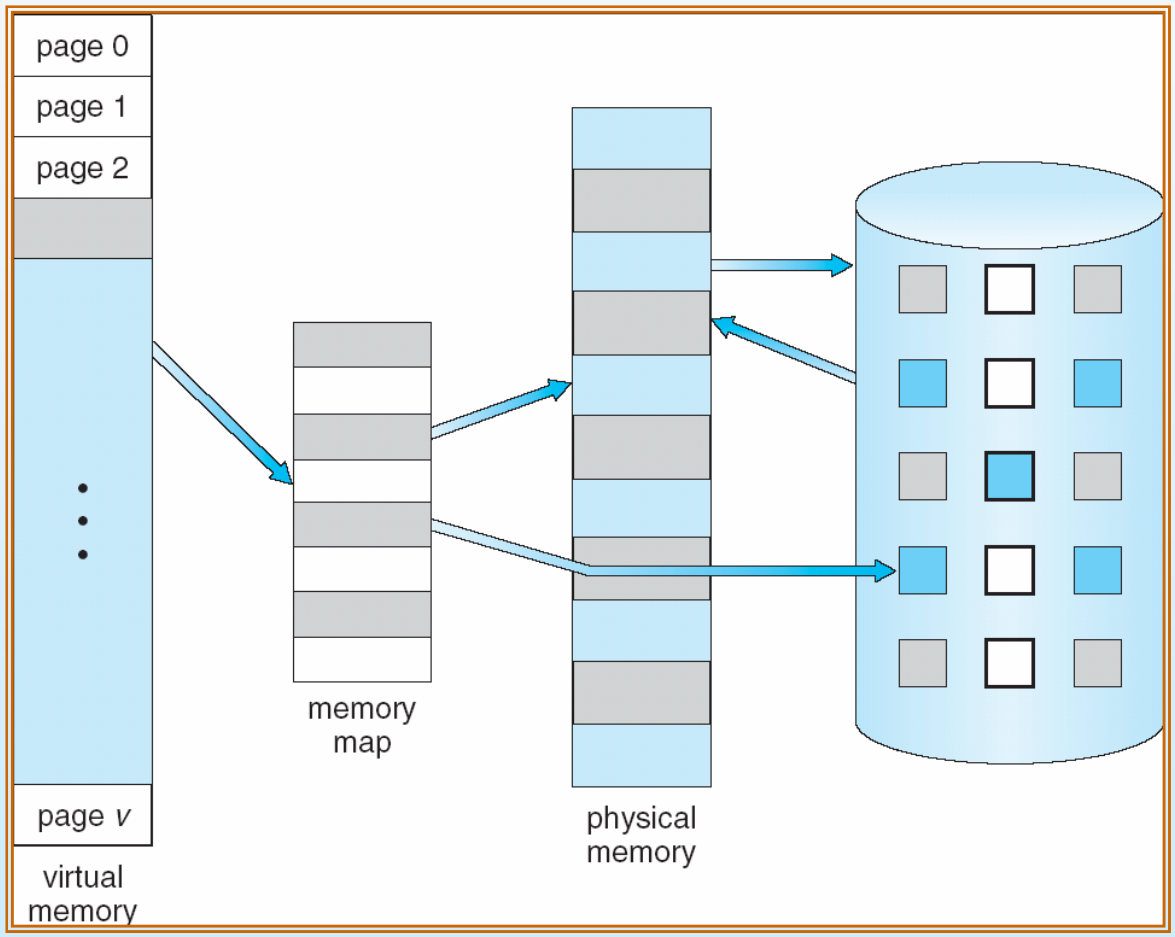
- Fundamentos
- Paginação sob Demanda
- Criação de Processos
- Substituição de Páginas
- Alocação de Blocos (*frames*)
- Paginação Excessiva (*Thrashing*)
- Segmentação sob Demanda
- Exemplos de Sistemas Operacionais

Fundamentos

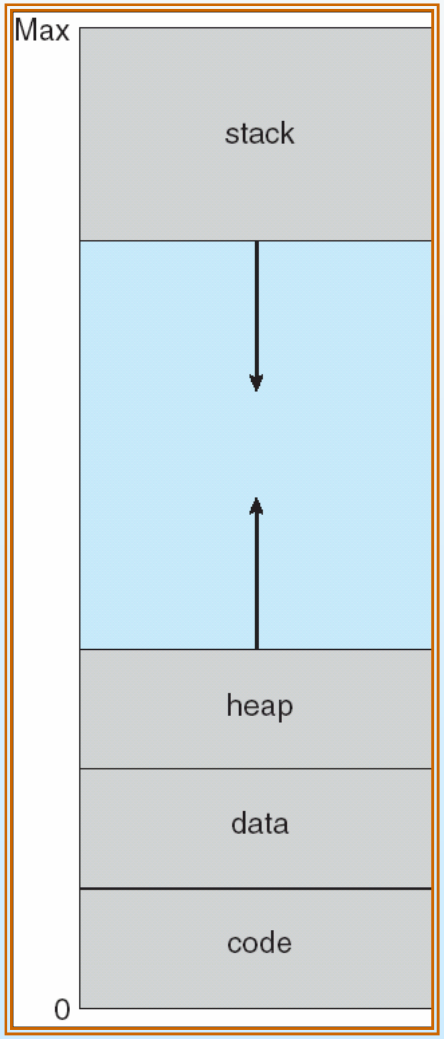
- **Memória virtual** – separação da memória lógica do usuário da memória física.
  - Somente parte do programa precisa estar na memória para execução.
  - Espaço de endereçamento lógico pode ser bem maior que o espaço de endereçamento físico.
  - Permite espaços de endereçamento serem compartilhados por vários processos.
  - Permite a criação de processos mais eficiente.

- Memória Virtual pode ser implementada via:
  - Paginação sob demanda
  - Segmentação sob demanda

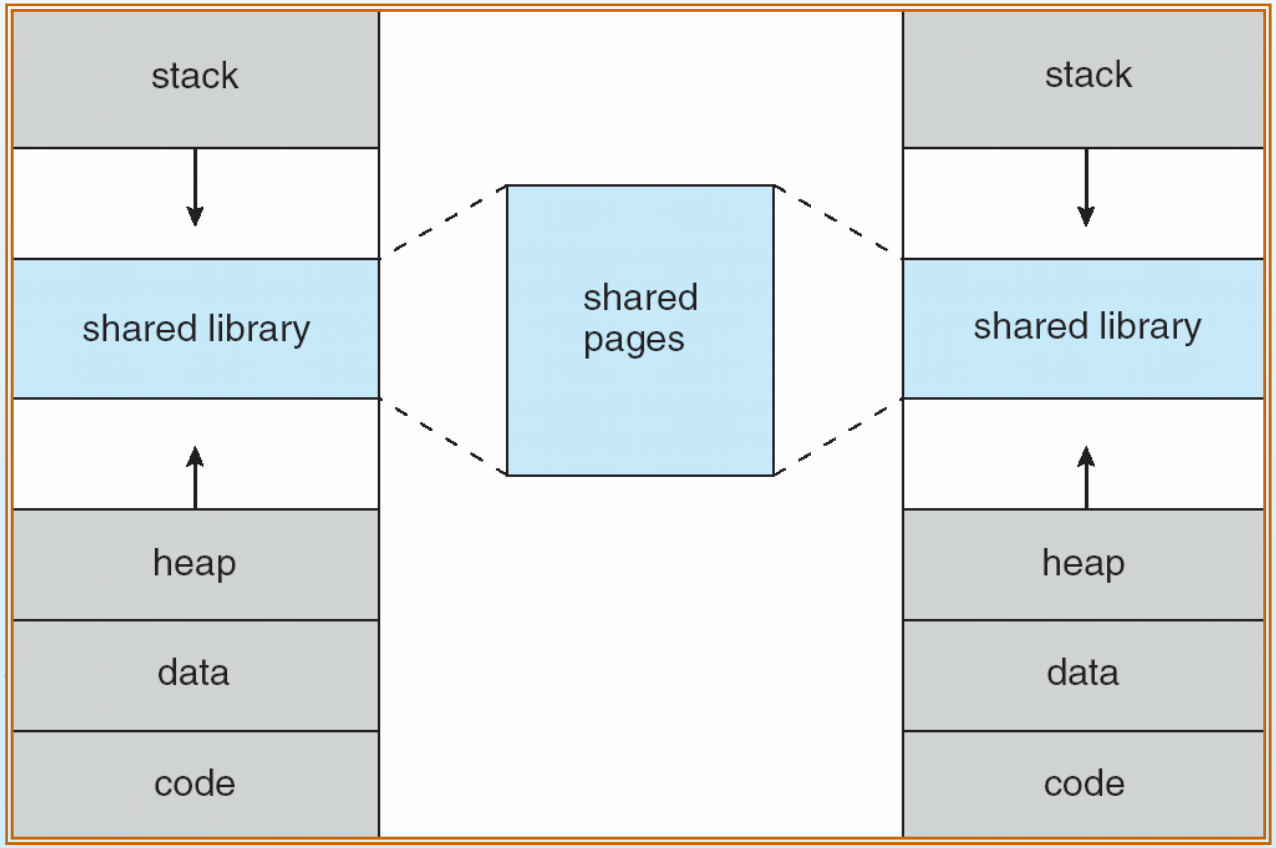
Memória Virtual que é Maior do que a Memória Física



Espaço de Endereçamento Virtual



Biblioteca Compartilhada Usando Memória Virtual

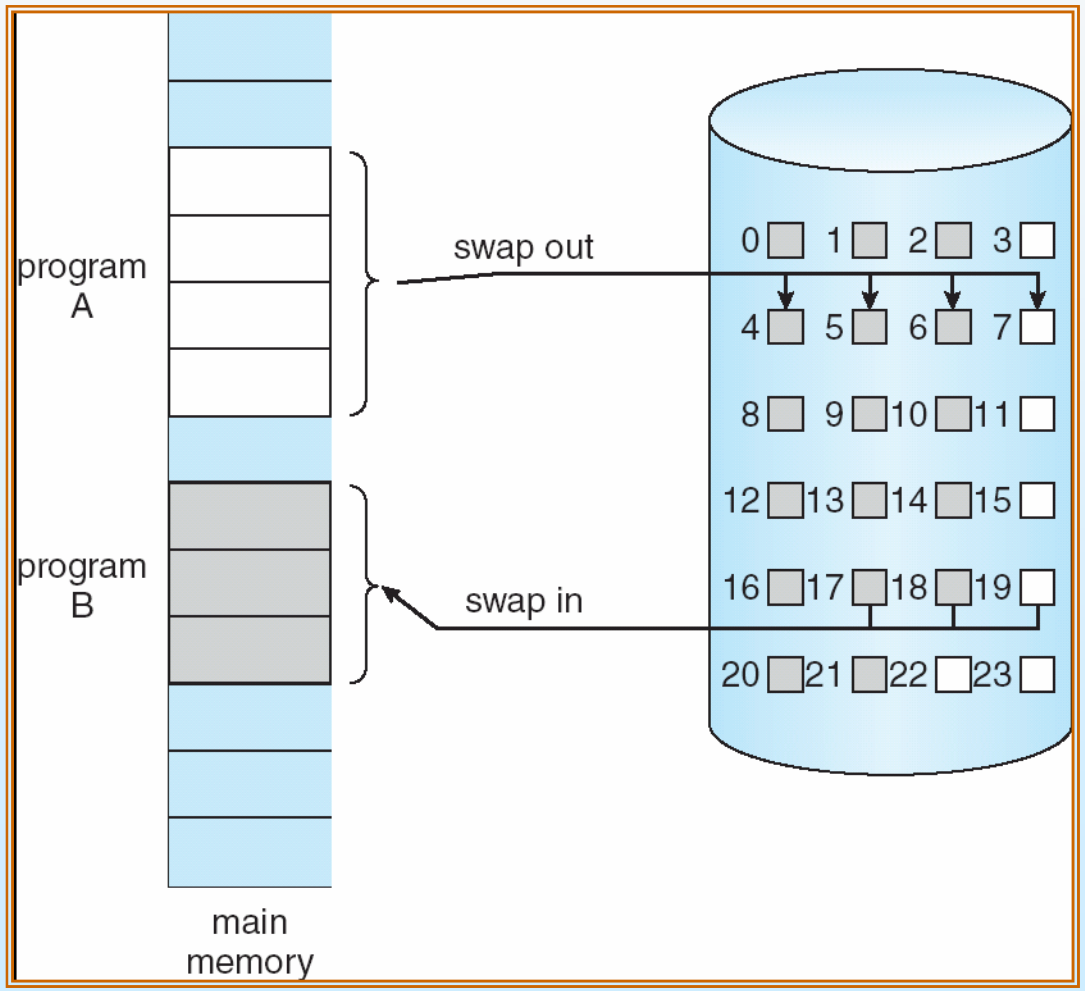


Paginação sob Demanda

- Traz uma página para a memória somente quando ela é necessária.
  - Necessita de menos E/S
  - Ocupa menos memória
  - Resposta mais rápida
  - Permite mais usuários

- Página é necessária ⇒ referencia ela
  - referência invalida ⇒ aborta
  - Não presente na memória ⇒ traz para a memória

Transferência de uma Memória Paginada para o Espaço Contíguo de Disco



Bit Válido-Inválido

- Com cada entrada na tabela de páginas é associado um bit válido-inválido (**v** ⇒ na memória, **i** ⇒ não está na memória)
- Inicialmente bit válido-inválido é **i** em todas entradas da tabela.
- Exemplo de uma tabela de páginas.

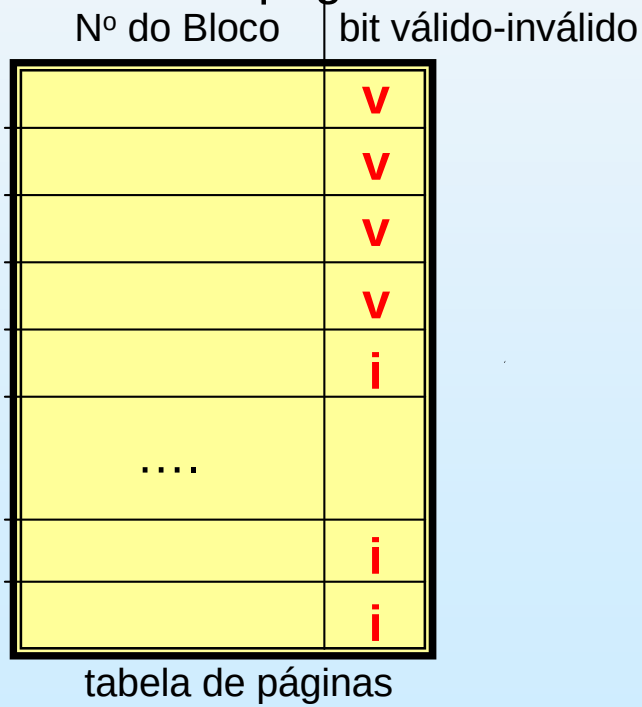
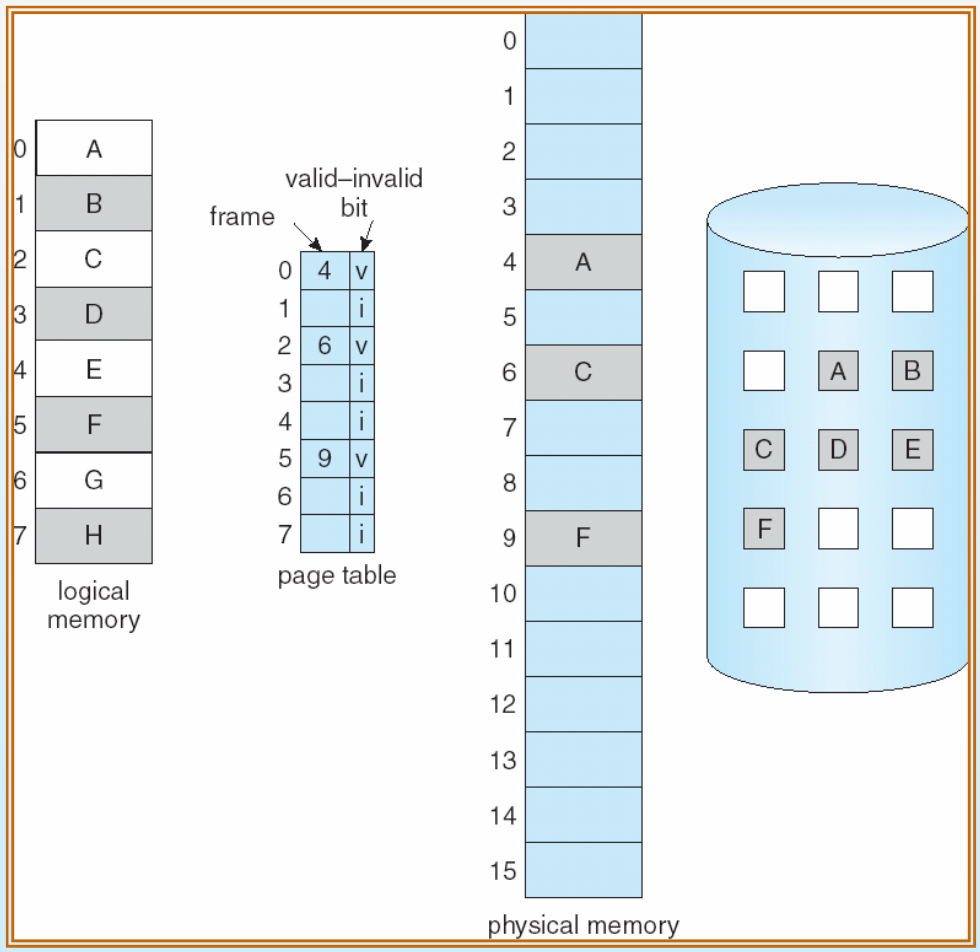


Tabela de Páginas Quando Algumas Páginas não estão na Memória Principal



Página Ausente (*Page Fault*)

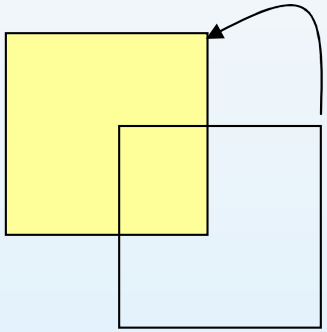
- Se existe uma referência para uma página, a primeira referência para esta página irá causar uma *trap* no sistema operacional:

  página ausente

- SO verifica uma outra tabela para decidir:
  - Referência inválida ⇒ aborta.
  - Não está na memória.
- Obtém bloco livre na memória.
- Traz página do disco para o bloco alocado.
- Atualiza tabelas
- Bit de validação= v.
- Reinicia execução da Instrução que causou a página ausente.

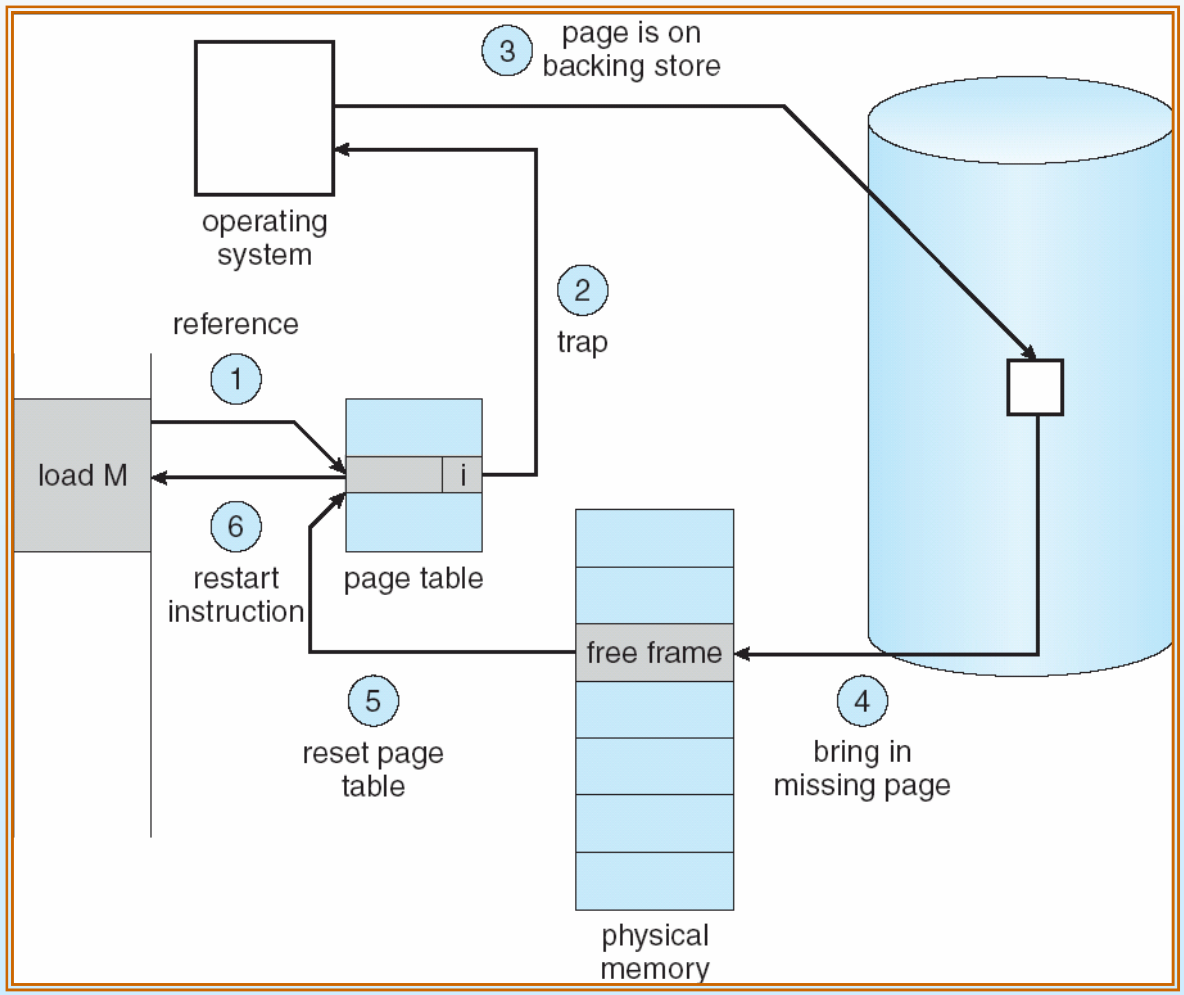
Página Ausente (Cont.)

- Reinicia execução da Instrução: Menos recentemente usada
  - movimenta blocos



  - auto-incremento/auto-decremento

Passos no Tratamento de uma Página Ausente



O que ocorre se não existe bloco livre?

- Substituição de Página – encontrar alguma página na memória, que não esteja em uso, e transferir para o armazenamento secundário.
  - algoritmo
  - desempenho – é desejado um algoritmo que resulte em um número mínimo de páginas ausentes.
- A mesma página pode ser trazida para memória diversas vezes.

Desempenho da Paginação sob Demanda

- Taxa de paginas ausentes $0 \le p \le 1.0$
  - Se $p = 0$ nenhuma página ausente
  - Se $p = 1$, cada referência gera uma página ausente

- Tempo Médio de Acesso (TMA)

$$\begin{aligned} \text{TMA} = (1 - p) \times &\text{ tempo de acesso à memória} \\ &+ p \text{ (sobrecarga da página ausente} \\ &+ [\text{tempo para gravar }] \\ &+ \text{tempo para ler} \\ &+ \text{sobrecarga de reinicialização)} \end{aligned}$$

Exemplo de Paginação sob Demanda

- Tempo de Acesso à Memória = 1 microssegundo
- 50% das vezes a página que está sendo substituída foi modifica e deve ser gravada no meio de acesso secundário.
- Tempo de troca de página = 10 ms = 10.000 microssegundos

$$TMA = (1 – p) \times 1 + p\ (15000)$$

$$1 + 15000P \quad \text{(em microssegundos)}$$

Criação de Processos

- Memória virtual traz outros benefícios durante a criação de processos:

  - Copy-on-Write (Cópia na Escrita)

  - Arquivos Mapeados na Memória (depois)

Copy-on-Write

- Copy-on-Write (COW) permite que tanto o processo pai como filho inicialmente *compartilhem* as mesmas páginas na memória

  Se qualquer processo modificar uma página compartilhada, osomente então esta página será copiada

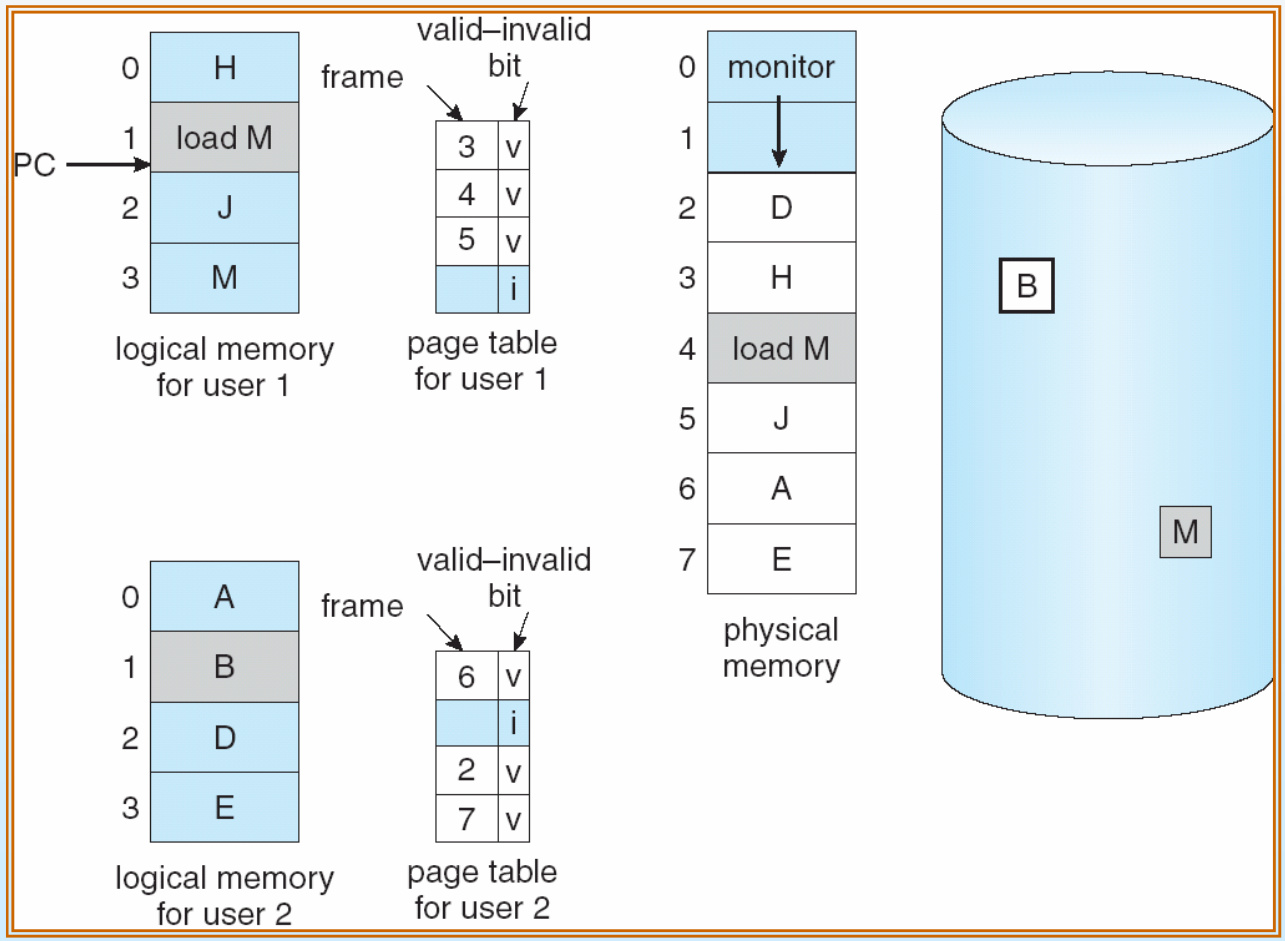
- COW permite criação de processos mais eficiente uma vez que somente páginas modificadas são copiadas

- Páginas livres são alocadas de um **pool** de páginas zeradas

Substituição de Páginas

- Prevenir super alocação da memória modificando a rotina de serviço da página ausente para incluir um algoritmo de substituição de página.
- Usar um **bit de modificação** (*modify* / *dirty bit*) para reduzir a sobrecarga da transferência de página – somente páginas modificadas são gravadas no disco.
- Substituição de páginas completa a separação entre a memória lógica e a memória física – grande área de memória virtual pode ser obtida com memória física reduzida.

Necessidade de Substituição de Página



Algoritmo Básico de Substituição de Página

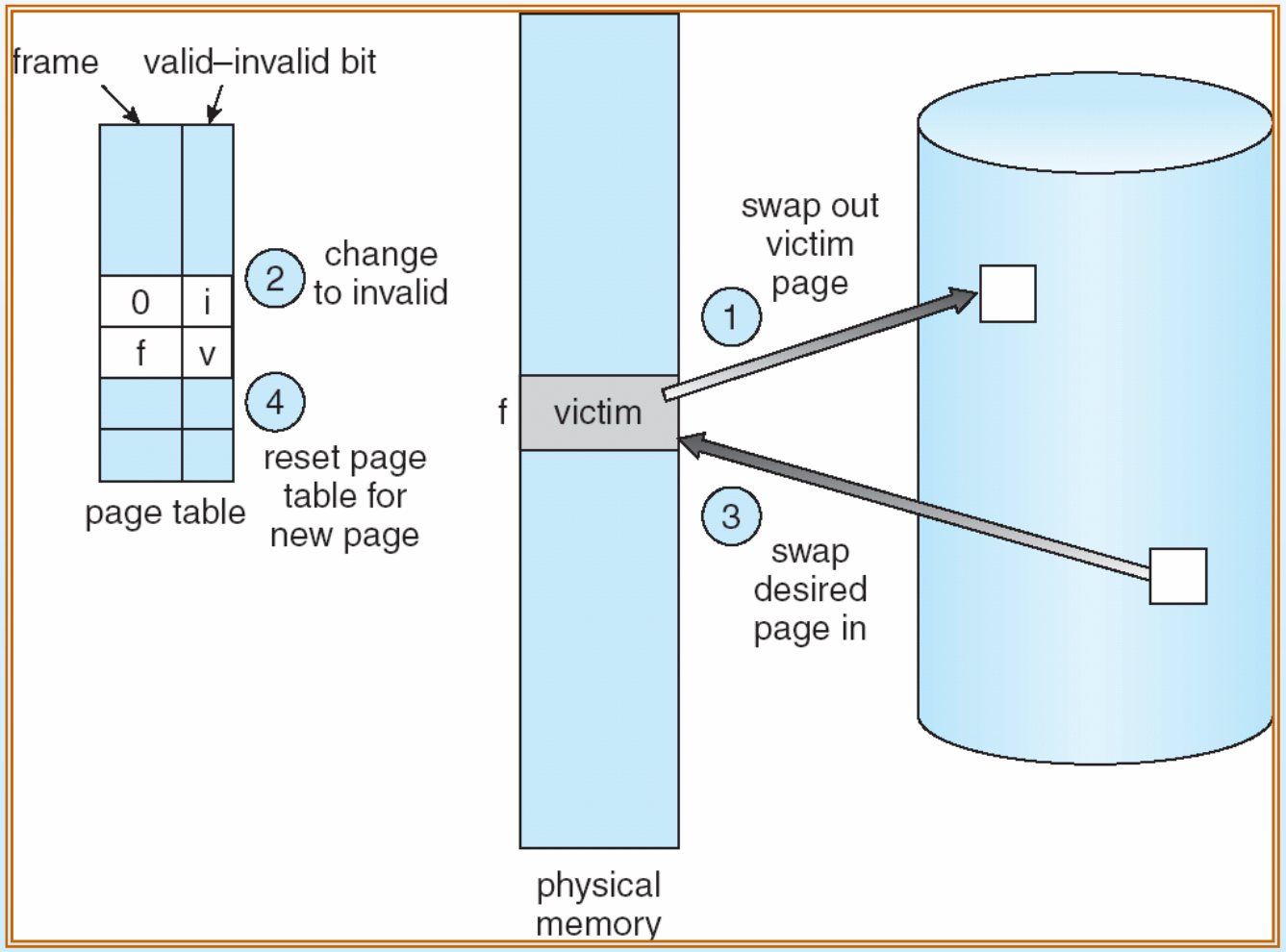
1. Encontre a localização da página desejada no disco

2. Encontre um bloco livre:
   - Se existe um bloco livre, use-o
   - Se não existe um bloco livre, use um algoritmo de substituição de página para selecionar um bloco **vitima**

3. Leia a página desejada no (novo) bloco livre. Atualize as tabelas de página e de blocos.

4. Reinicie o processo

Substituição de Página

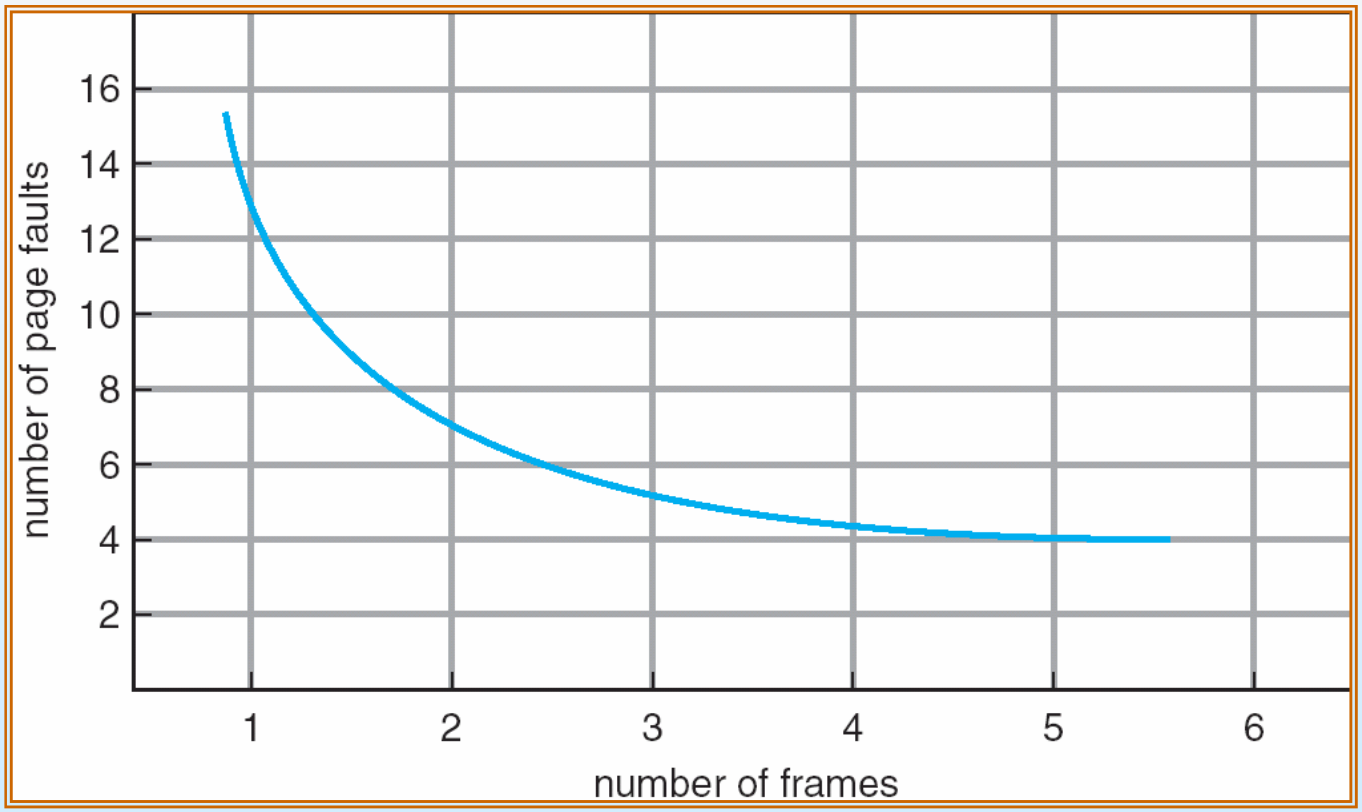


Algoritmos de Substituição de Páginas

- Desejável a menor taxa de páginas ausentes
- Um algoritmo é avaliado pela execução em uma seqüência particular de referências de memória (string de referência) e é computado o número de páginas ausentes nessa string
- Em todos os exemplos, a seqüência de referência é

  1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

Gráfico de Páginas Ausentes Versus Número de Blocos



Algoritmo FIFO (*First-In-First-Out*)

- Seqüência de Referência: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
- 3 blocos (3 páginas podem estar na memória ao mesmo tempo para cada processo)

1	1	4	5	
2	2	1	3	9 páginas ausentes
3	3	2	4	

- 4 blocos

1	1	5	4	
2	2	1	5	10 páginas ausentes
3	3	2		
4	4	3		

Substituição de Página FIFO

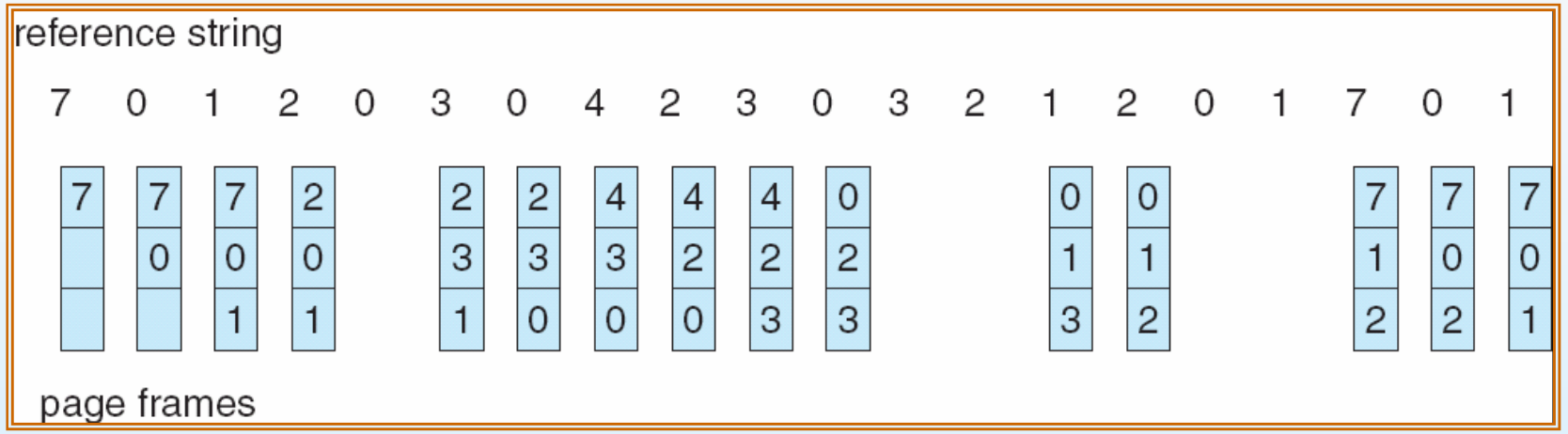
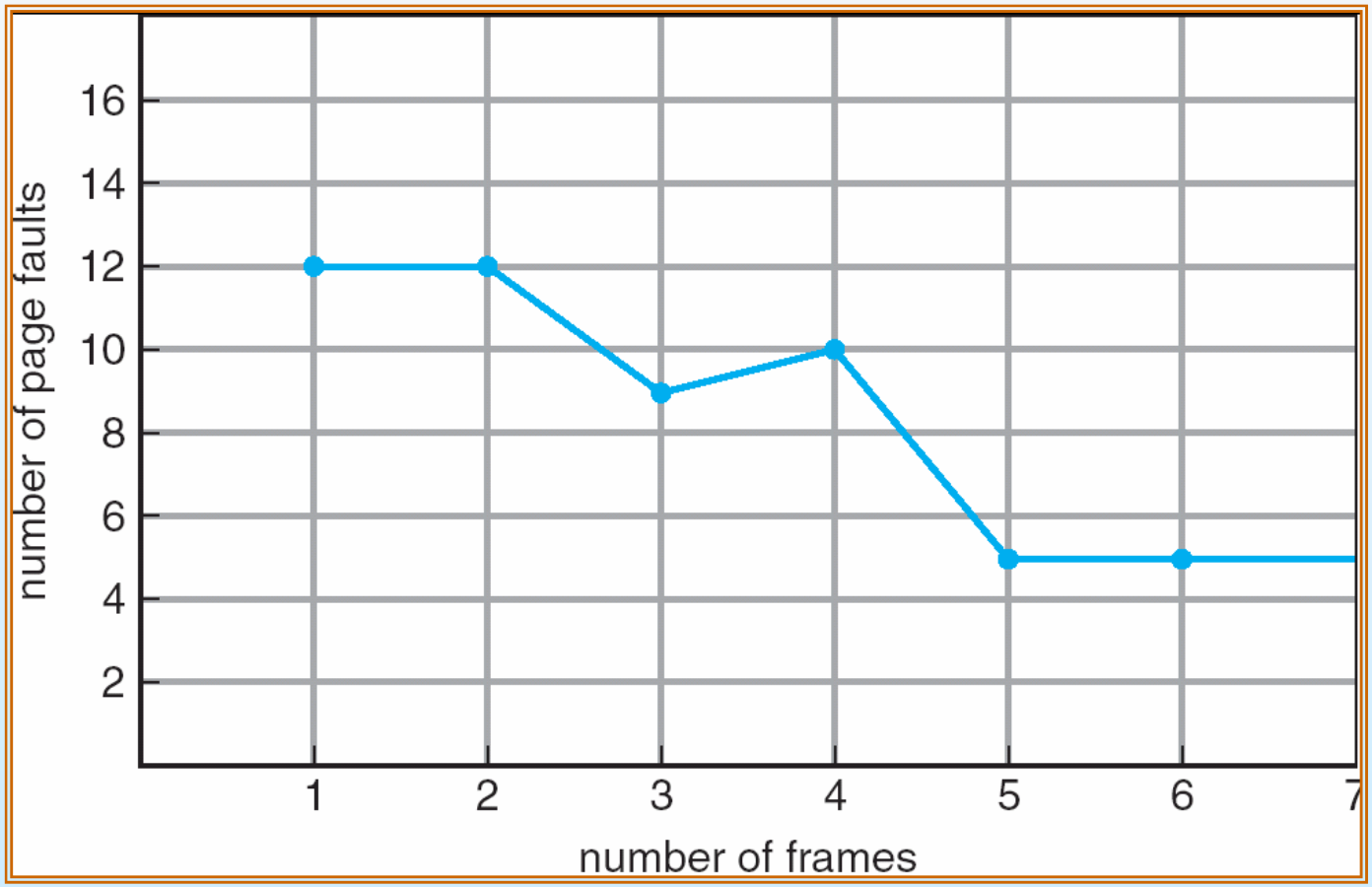


Ilustração da Anomalia de Belady no FIFO



Algoritmo Ótimo

- Substituir a página que não será usada pelo maior período de tempo.
- Exemplo com 4 blocos

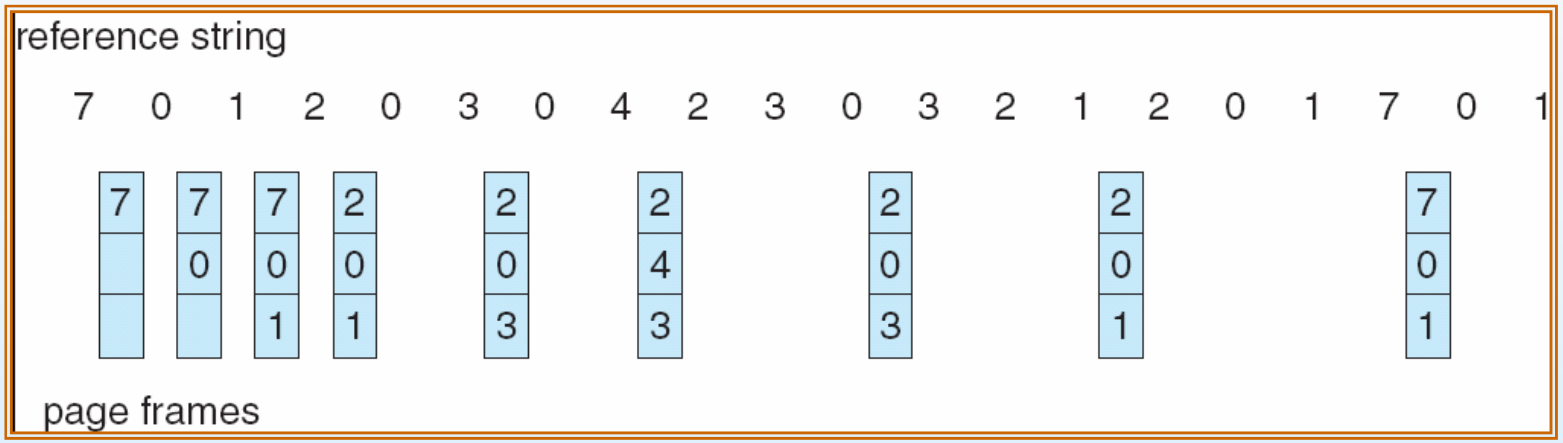
1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

1	4
2	
3	
4	5

6 page faults

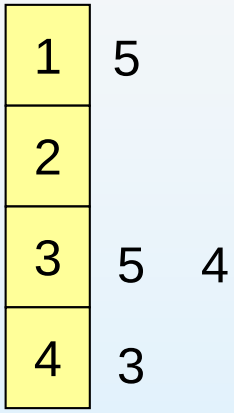
- Como você sabe qual página será referenciada?
- Usado para medir quão bom é o desempenho de determinado algoritmo.

Substituição de Página Ótima



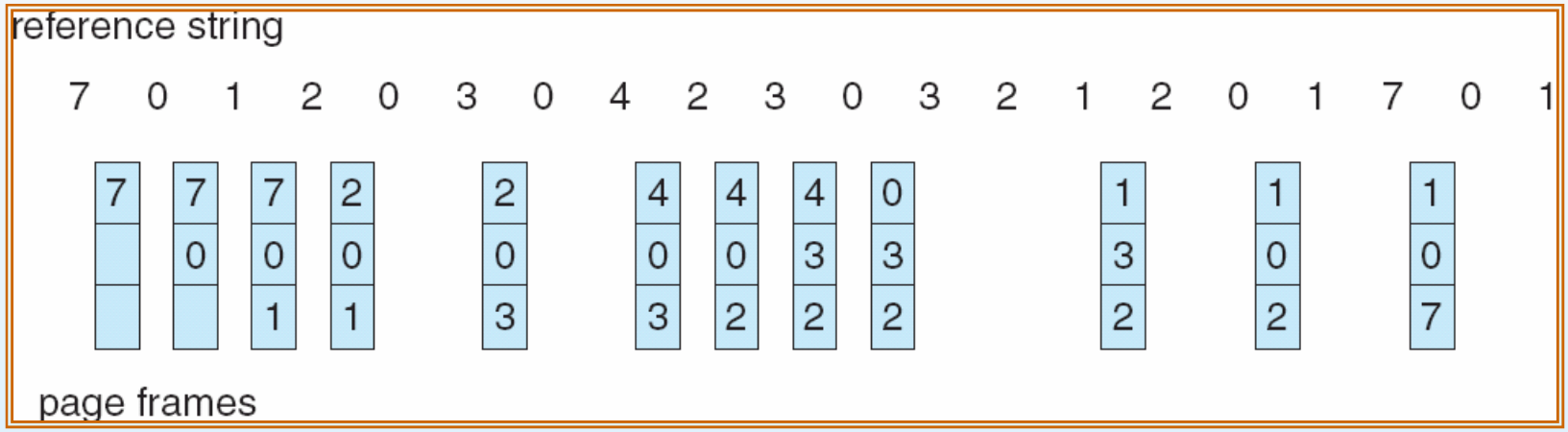
Algoritmo Menos Recentemente Usado (LRU - *Least Recently Used*)

- Seqüência de Referência: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5



- Implementação com contadores
  - Cada entrada na tabela de página tem um contador; cada vez que a página é referenciada, o valor do *clock (*contador de ciclos da CPU*)* é copiado no contador da entrada.
  - Quando uma página necessita ser substituída, através do valor do contador é determina qual página deve sair da memória.

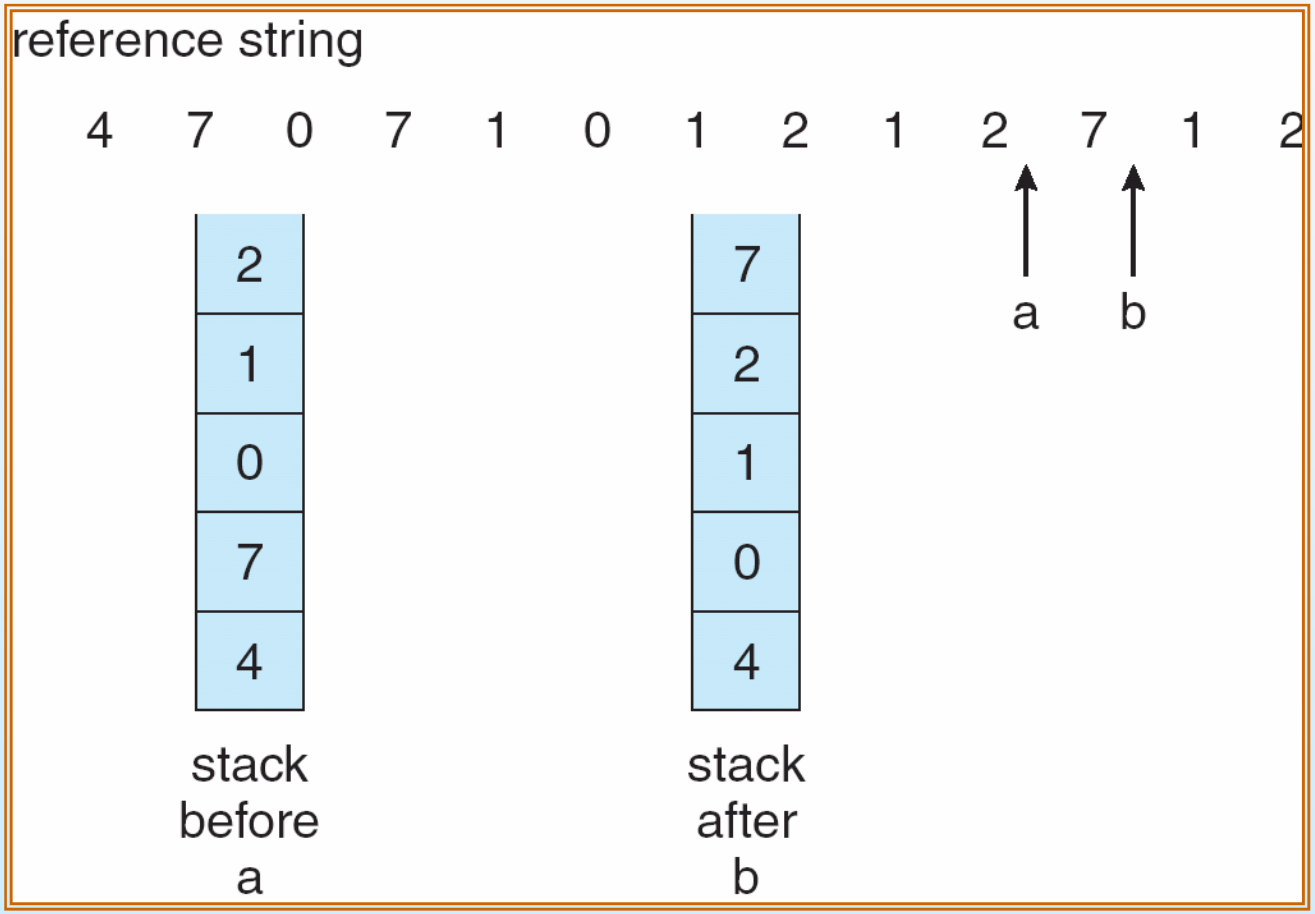
Substituição de Página LRU



Algoritmo LRU (Cont.)

- Implementação com pilha – manter uma pilha dos números de páginas de uma forma duplamente encadeada:
  - Página referenciada:
    - Mova ela para o topo
    - Requer a alteração de 6 ponteiros
  - Sem busca para substituição

Uso de uma Pilha para Armazenar a referência da Página mais Recente

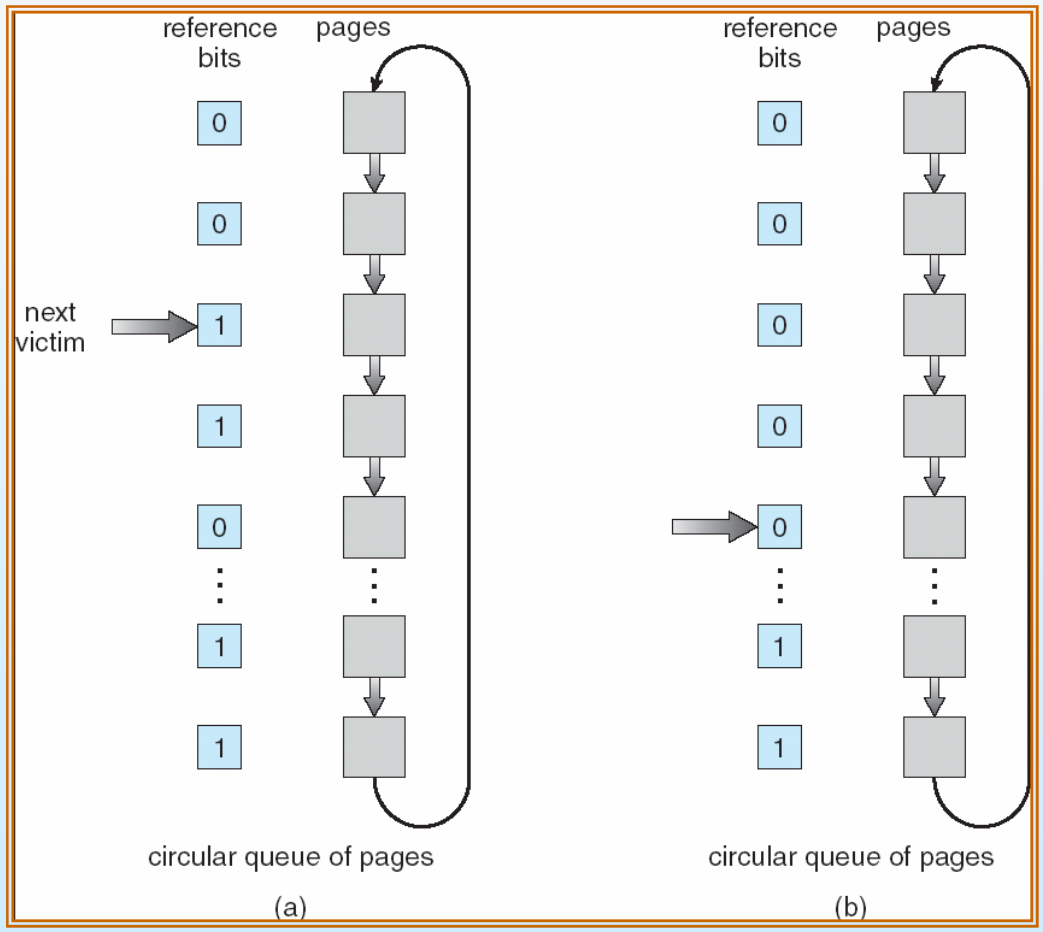


Algoritmos LRU Aproximados

- **Bit de Referência**
  - Com cada pagina é associado um bit, inicialmente = 0
  - Quando a página é referenciada o bit é alterado para 1.
  - Substitui a página que o bit seja 0 (se existir). Entretanto, não se sabe a ordem de referência.

- **Segunda Chance**
  - Necessita do bit de referência.
  - Substituição circular (implementada em uma fila circular).
  - Se uma página a ser substituída (no sentido horário) tem o bit de referência= 1, então:
    - altera o bit de referência para 0.
    - deixa a página na memória.
    - substitui a próxima página (no sentido horário), sujeita as mesmas regras.

Algoritmo de Substituição de Página Segunda Chance



Algoritmos com Contadores

- Mantém um contador do número de referências que foram feitos para cada página
- **Algoritmo Menos Freqüentemente Usada** (*Least Frequently Used* – LFU) : substitui a página com o menor contador.
- **Algoritmo Mais Freqüentemente Usada** (*More Frequently Used* – MFU): baseado no argumento que a página com o menor contador foi provavelmente recém carregado na memória e ainda não foi usada.

Alocação de Blocos (*frames*)

- Cada processo necessita um *número mínimo* de páginas.
- Exemplo: IBM 370 – 6 páginas para manipular instrução SS MOVE:
  - instrução é de 6 bytes, pode estar contida em 2 páginas.
  - 2 páginas para manipular o endereço do bloco a ser movido (*from*).
  - 2 páginas para manipular o endereço da área destino (*to)*.
- Dois esquemas de alocação são comumente usados.
  - Alocação fixa
  - Alocação por prioridade

Alocação Fixa

- Alocação Equânime – ex.: se 100 blocos e 5 processos, dar para cada 20 páginas.
- Alocação Proporcional – Aloca de acordo com o tamanho do processo.
  - $s_i =$ tamanho do processo p_i
  - $S = \sum s_i$
  - $m =$ número total de blocos
  - $a_i =$ alocação para $p_i = \frac{s_i}{S} \times m$

$$m = 64$$

$$s_i = 10$$

$$s_2 = 127$$

$$a_1 = \frac{10}{137} \times 64 \approx 5$$

$$a_2 = \frac{127}{137} \times 64 \approx 59$$

Alocação por Prioridade

- Usa um esquema de alocação proporcional empregando prioridades ao invés do tamanho.

- Se processo P_i gera uma página ausente,
  - Seleciona para substituição um de seus blocos.
  - Seleciona para substituição um bloco de um processo com uma prioridade menor.

Alocação Global vs. Local

- **Substituição Global** – processo seleciona um bloco para substituição do conjunto de todos os blocos; um processo pode pegar um bloco de outro.

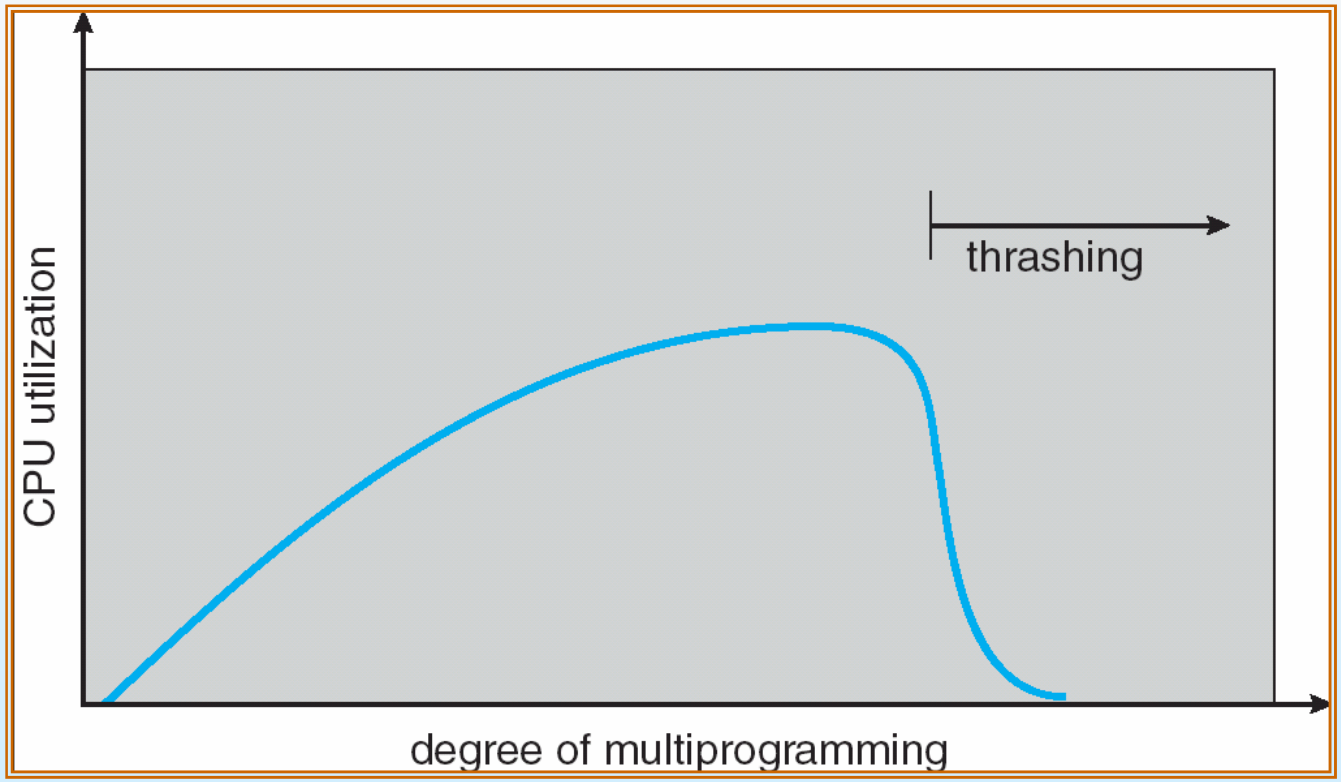
- **Substituição Local** – cada processo seleciona somente de seu próprio conjunto de blocos alocados.

Paginação Excessiva (*Thrashing*)

- Se um processo não possui páginas “suficientes”, a taxa de páginas ausentes é bastante alto. Isto leva a:
  - Baixa utilização da CPU.
  - Sistema operacional pensa que é necessário aumentar o grau de multiprogramação.
  - Outro processo é adicionado ao sistema.

- **Paginação Excessiva** ≡ um processo está ocupado trocando páginas de e para o meio de armazenamento secundário.

Paginação Excessiva - *Thrashing* (Cont.)

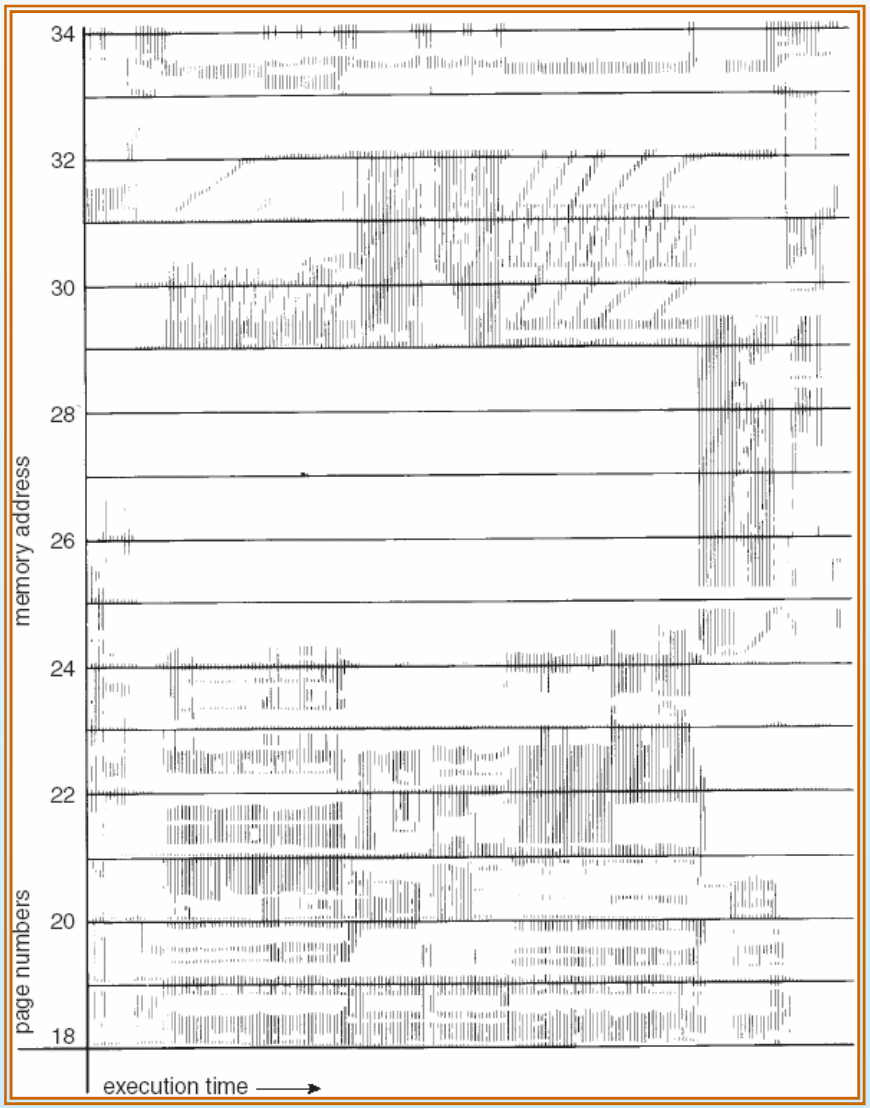


Paginação sob Demanda e Thrashing

- Por que paginação sob demanda funciona?
  Modelo de Localidade
  - Processo migra de uma localidade para outra
  - Localidades podem se sobrepor

- Por que ocorre paginação excessiva?
  Σ tamanho da localidade > quantidade total de memória

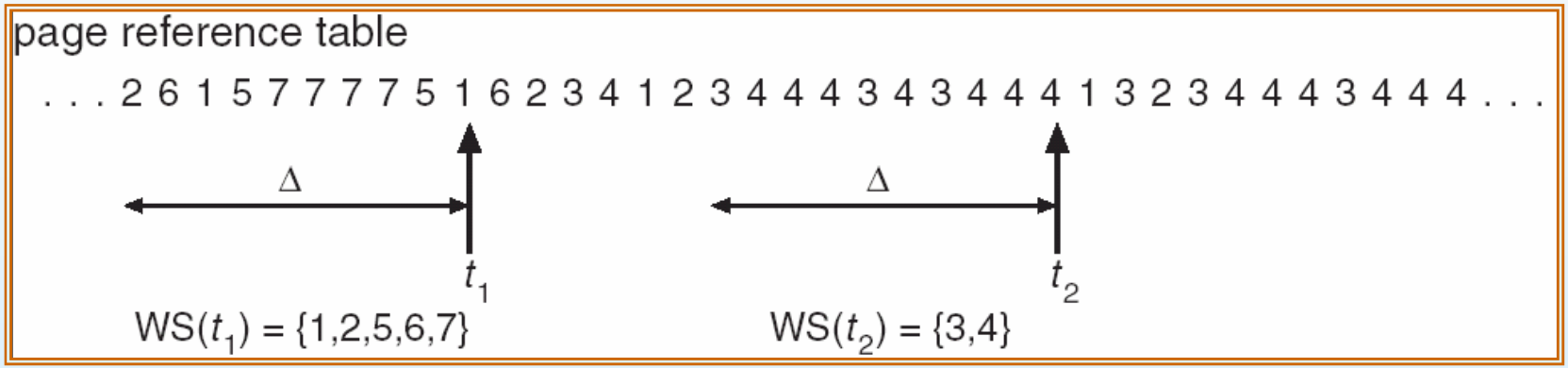
Localidade em um Padrão de Referências a Memória



Modelo do Conjunto-de-trabalho (*working set*)

- $\Delta \equiv$ *janela do conjunto-de-trabalho* $\equiv$ um número fixo de referências a páginas
  Exemplo: 10,000 instruções
- WSS_i (conjunto de trabalho do processo P_i) = número total de páginas referenciadas na Δ mais recente (varia no tempo)
  - Se Δ é muito pequeno não será suficiente para abranger toda a localidade.
  - Se Δ é muito grande conterá várias localidades sobrepostas.
  - Se $\Delta = \infty \Rightarrow$ irá conter todo o processo.
- $D = \Sigma\ WSS_i \equiv$ total de blocos necessários
- if $D > m \Rightarrow$ Paginação Excessiva
- Política: se $D >$ m, então suspende um dos processos.

Modelo do Conjunto-de-trabalho

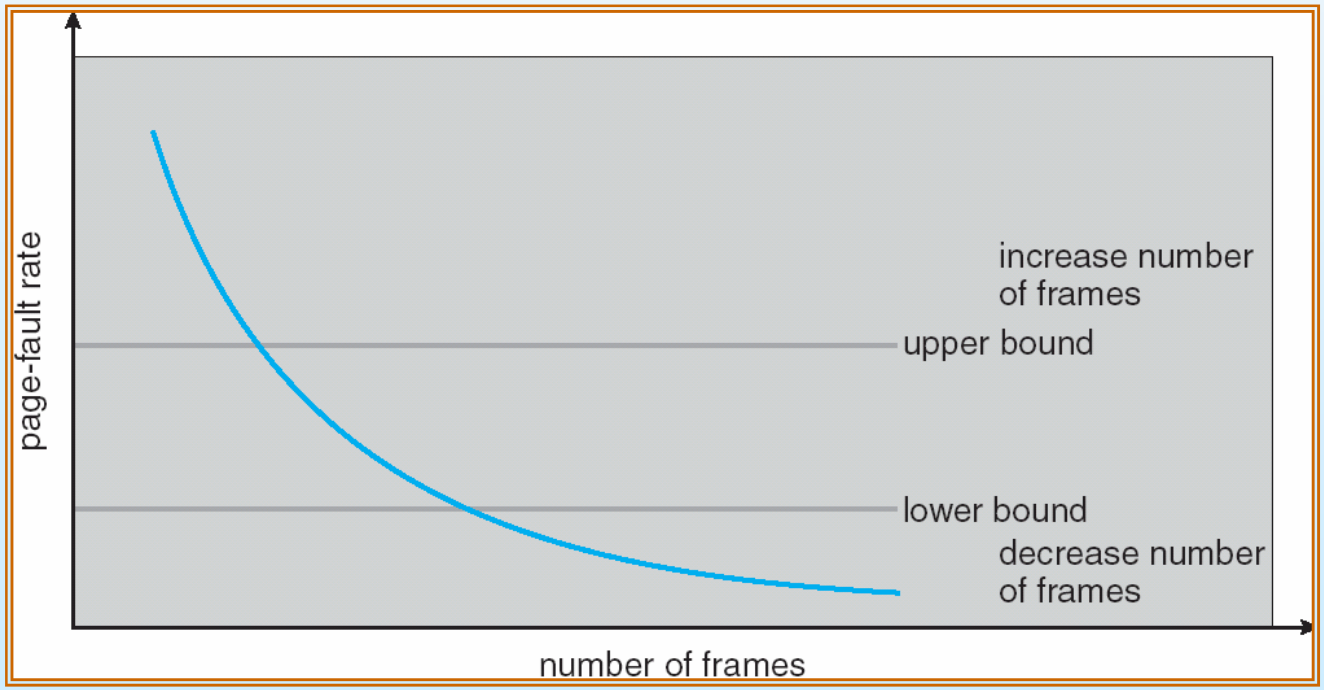


Mantendo o Conjunto-de-trabalho

- Aproximação com intervalos de tempos regulares + bit de referência
- Exemplo: Δ = 10,000
  - Interrupções de tempo após cada 5000 unidades de tempo.
  - Manter na memória 2 bits para cada página.
  - Sempre que ocorrer a interrupção copiar e alterar o valor de todos os bits de referência para 0.
  - Se um dos bits em memória = 1 $\Rightarrow$ página está no conjunto-de-trabalho.
- Por que esta técnica não é completamente precisa?
- Melhora = 10 bits e interrupção a cada 1000 unidades de tempo.

Esquema de Freqüência de Página Ausente

- Estabelecer uma taxa “aceitável” de páginas ausentes.
  - Se taxa atual é muito baixa, processos perdem blocos.
  - Se taxa atual é muito alta, processos ganham blocos.



Arquivos Mapeados na Memória

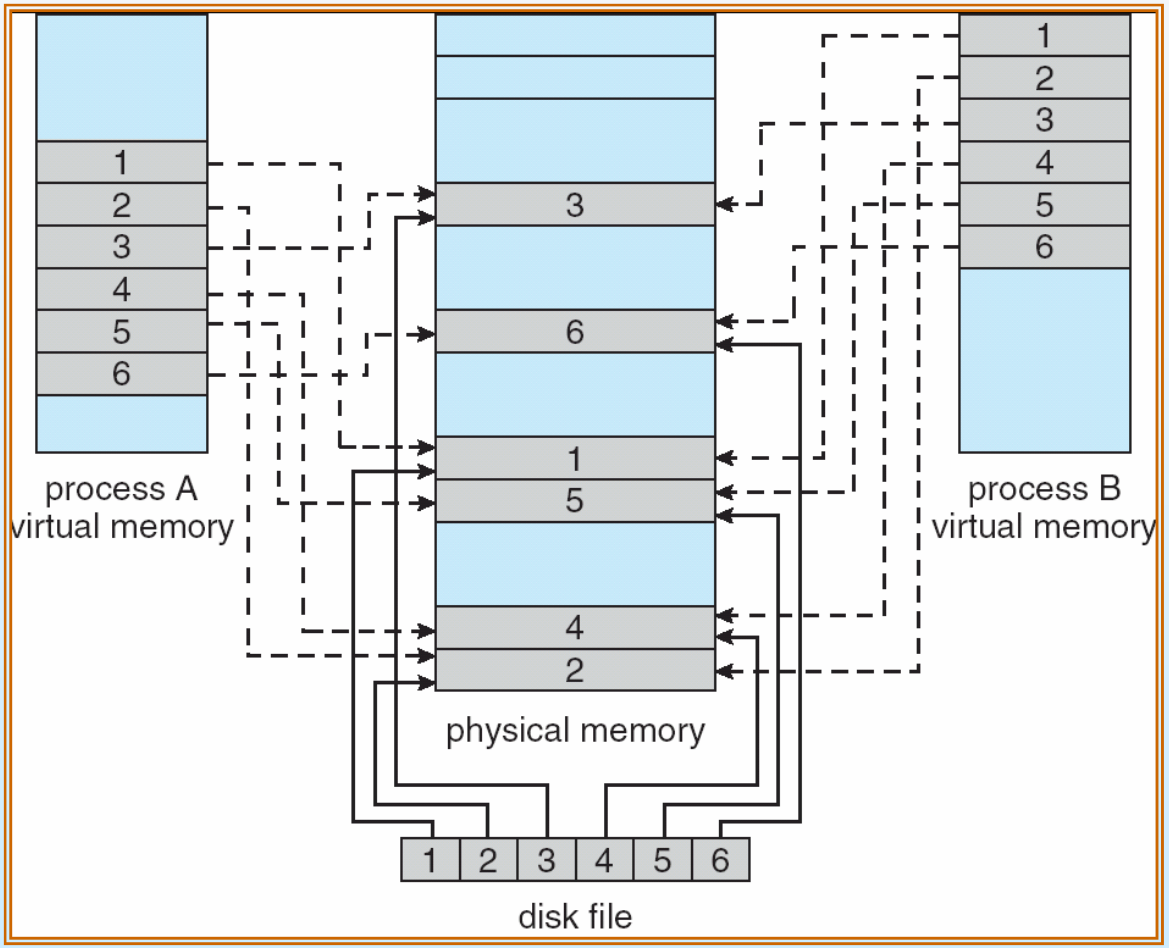
- E/S de arquivos mapeados na memória permite que acessos sejam tratados como acesso a rotinas de memória pelo **mapeamento** de blocos do disco em páginas na memória

- Um arquivo é lido inicialmente usando paginação sob demanda. Uma porção do arquivo é lida do sistema de arquivos em páginas físicas. Leituras e escritas subseqüentes de/para o arquivo são tratados como acessos a memória comuns.

- Simplifica o acesso a arquivos tratando E/S através da memória ao invés do uso de chamadas de sistemas **read() write()**

- Também permite que vários processos mapeiem o mesmo arquivo através do compartilhamento de páginas na memória

Arquivos Mapeados na Memória (Cont.)



Arquivos Mapeados na Memória em Java

```
import java.io.*;
import java.nio.*;
import java.nio.channels.*;
public class MemoryMapReadOnly
{
    // Assume the page size is 4 KB
    public static final int PAGE SIZE = 4096;
    public static void main(String args[]) throws IOException {
        RandomAccessFile inFile = new RandomAccessFile(args[0],"r");
        FileChannel in = inFile.getChannel();
        MappedByteBuffer mappedBuffer =
          in.map(FileChannel.MapMode.READ ONLY, 0, in.size());
        long numPages = in.size() / (long)PAGE SIZE;
        if (in.size() % PAGE SIZE > 0)
            ++numPages;
```

Arquivos Mapeados na Memória em Java (Cont.)

```
            // we will "touch" the first byte of every page
            int position = 0;
            for (long i = 0; i < numPages; i++) {
                    byte item = mappedBuffer.get(position);
                    position += PAGE SIZE;
            }
            in.close();
            inFile.close();
    }
}
```

- A API para o método map() é:

```
map(mode, position, size)
```

Outras Questões – Pré-paginação

- Pré-paginação
  - Para reduzir o alto nível de paginas ausentes que ocorrem na inicialização do processo
  - Pré-paginar todas ou algumas páginas que o processo irá precisar, antes de serem referenciadas
  - Mas se páginas pré-paginadas não são usadas, E/S e memória foram desperdiçados
  - Assuma que *s* páginas são pré-paginadas e *α* delas são usadas
    - Se o custo de $s * \alpha$ falta páginas evidados > ou < que o custo da pré-paginação
      $s * (1- \alpha)$ páginas desnecessárias?
    - α próximo de zero $\Rightarrow$ perdas na pré-paginação

Outras Questões – Tamanho da Página

- Escolha do tamanho da página deve levar em consideração:
  - fragmentação
  - tamanho da tabela
  - sobrecarga de E/S
  - localidade

Outras Questões – Alcance da TLB

- Alcance da TLB – A quantidade de memória acessível a partir da TLB
- Alcance da TLB = (Tamanho da TLB) X (Tamanho da Página)
- Idealmente, o conjunto-de-trabalho de cada processo é armazenado na TLB. Em caso contrário existirá um alto grau de páginas ausentes.
- Aumentar o tamanho da página. Isso pode causar um aumento na fragmentação uma vez que nem todas as aplicações necessitam de páginas de tamanho grande
- Fornecer vários tamanhos de página. Isso possibilita aos aplicativos que necessitam de páginas maiores a oportunidade de usá-las sem aumentar a fragmentação.

Outras Questões – Estrutura de Programas

- Estrutura de programa
  - Int[128,128] data;
  - Cade linha é armazenada em uma página
  - Programa 1

```
for (j = 0; j <128; j++)
    for (i = 0; i < 128; i++)
        data[i,j] = 0;
```

128 x 128 = 16,384 páginas ausentes

  - Programa 2

```
for (i = 0; i < 128; i++)
    for (j = 0; j < 128; j++)
        data[i,j] = 0;
```

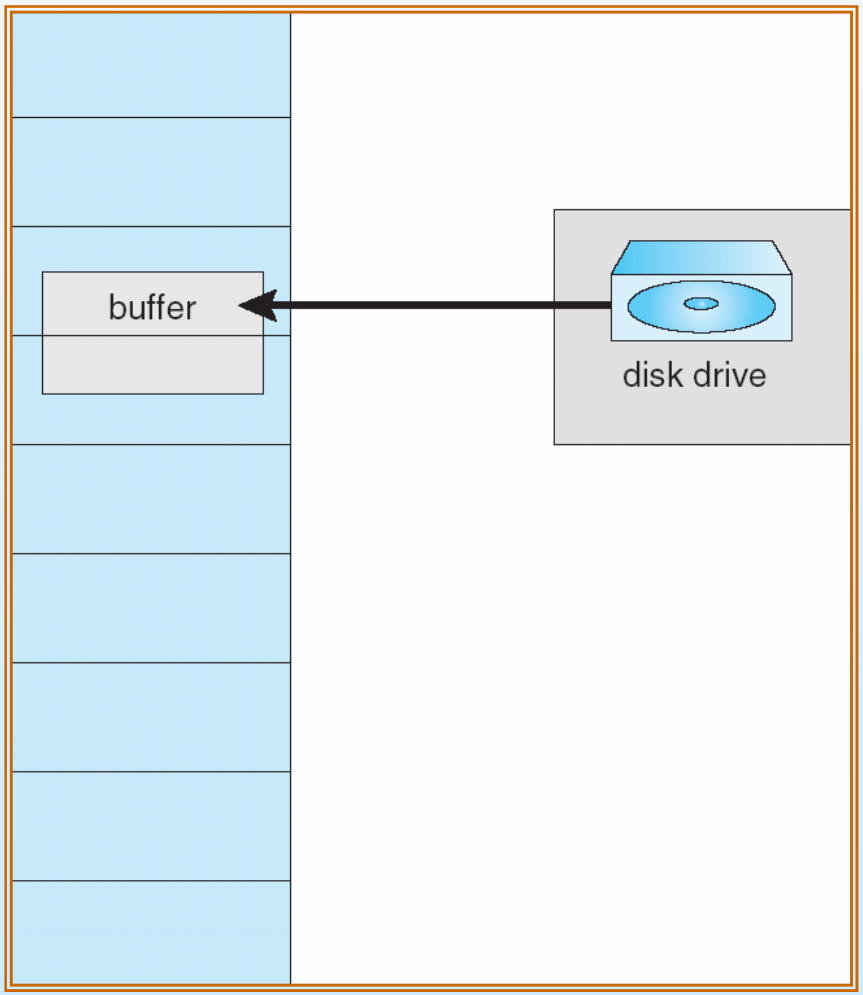
128 páginas ausentes

Outras Questões – E/S *interlock*

- **E/S Interlock** – Páginas algumas vezes devem ser travadas (*lock*) na memória

- Considere E/S. Páginas que são usadas para copiar um arquivo de um dispositivo devem ser travadas para não serem selecionadas para despejo por um algoritmo de substituição de página

Razão porque blocos usados para E/S devem estar na memória



Exemplos de Sistemas Operacionais

- Windows XP
- Solaris

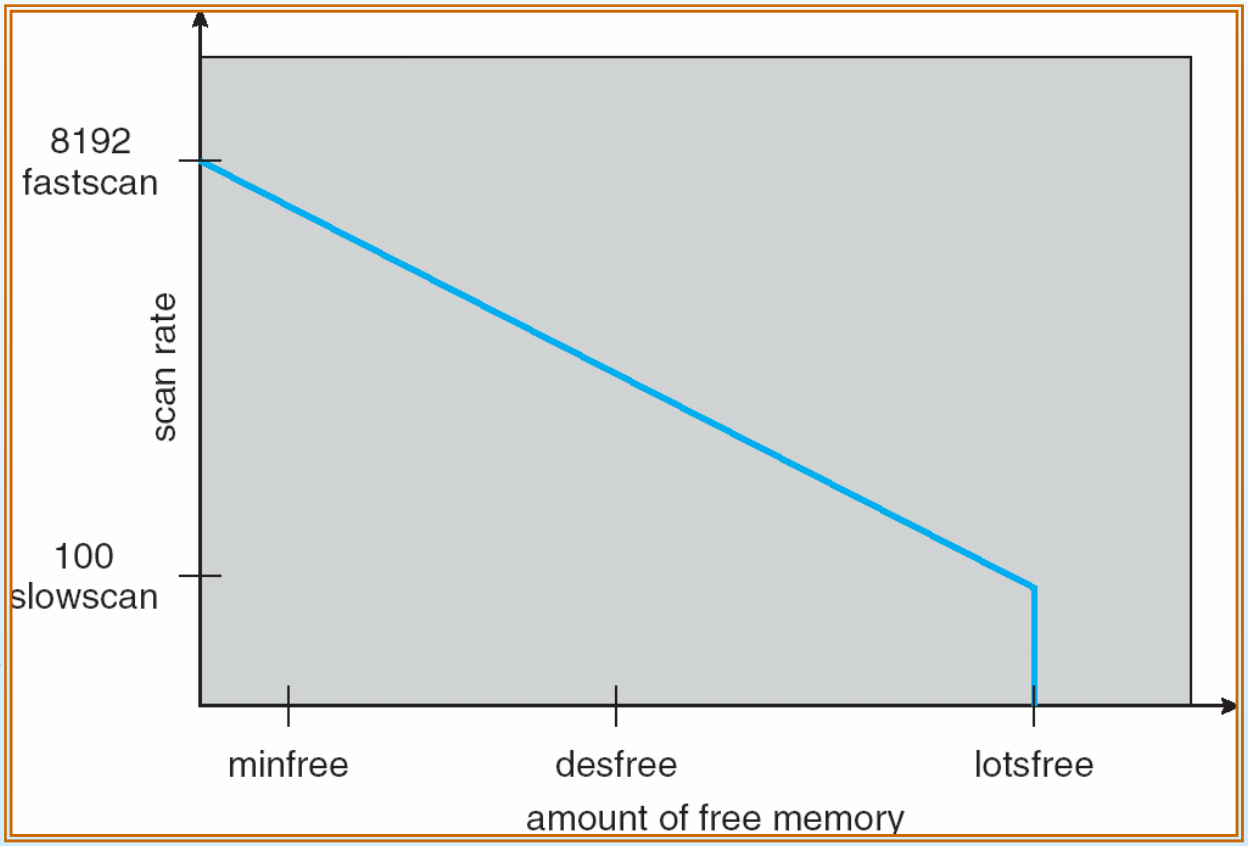
Windows XP

- Usa paginação sob demanda com ***clustering***. Clustering traz as páginas ao redor da página faltante.
- Processos possuem um **conjunto-de-trabalho mínimo** e um **conjunto-de-trabalho máximo**
- Conjunto-de-trabalho mínimo é o menor número de páginas que o processo tem garantido na memória
- Um processo pode ter associado tantas páginas quanto o seu conjunto-de-trabalho máximo
- Quando a quantia de memória livre no sistema cai abaixo de um limiar, um corte automático do conjunto-de-trabalho (**automatic working set trimming**) é realizado para restaurar a quantidade de memória livre
- O corte automática remove páginas de processos que têm excesso no conjunto-de-trabalho mínimo

Solaris

- Mantém uma lista de páginas livres para associar a processos faltantes
- *Lotsfree* – parâmetro limiar (quantidade de memória livre) para começar a paginar
- *Desfree* – parâmetro limiar para aumentar paginação
- *Minfree* – parâmetro limiar para fazer *swapping*
- Paginação é realizada pelo processo *pageout*
- Pageout procura páginas usando algoritmo circular modificado
- *Scanrate* é a taxa na qual páginas são procuradas. Isso se altera de *slowscan* (lento) até *fastscan* (rápido)
- Pageout é chamado mais freqüentemente dependendo da quantidade de memória livre disponível

Procura de Página no Solaris 2



Fim do Capítulo 9