Memória Virtual

MCTA026-13 - Sistemas Operacionais

Emilio Francesquini
e.francesquini@ufabc.edu.br

2019.Q1

Centro de Matemática, Computação e Cognição
Universidade Federal do ABC

UFABC

Disclaimer

UFABC

- Estes slides foram preparados para o curso de **Sistemas Operacionais na UFABC**.
- Este material pode ser usado livremente desde que sejam mantidos, além deste aviso, os créditos aos autores e instituições.
- Este material foi baseado nas ilustrações e texto preparados por Silberschatz, Galvin e Gagne [SGG] e Tanenbaum e Bos [TB] detentores do copyright.

CC BY SA

Introdução

Roteiro

UFABC

- Fundamentos
- Paginação por demanda
- Cópia-na-gravação (*Copy-on-write*)
- Substituição de página
- Alocação de *Frames*
- Thrashing
- Arquivos mapeados em memória
- Alocando a memória do kernel
- Outras considerações
- Exemplos do sistema operacional

Objetivos

UFABC

- Descrever os benefícios de um sistema de memória virtual
- Explicar os conceitos de paginação sob demanda, algoritmos de substituição de páginas e alocação de frames
- Para discutir o princípio do modelo de conjunto de trabalho
- Examinar a relação entre memória compartilhada e arquivos mapeados na memória
- Explorar como a memória do kernel é gerenciada

Fundamentos

UFABC

- O código precisa estar na memória para ser executado, mas o programa completo raramente é usado
  - Código de erro, rotinas incomuns, grandes estruturas de dados
- Todo o código do programa não é necessário ao mesmo tempo

UFABC

- Considere a capacidade de executar programas parcialmente carregados
  - Programa não seriam mais limitados pelos por limites físicos da memória RAM
  - Cada programa precisa de menos memória durante a execução: assim mais programas podem executar ao mesmo tempo
    - Maior utilização da CPU e taxa de transferência sem aumento no tempo de resposta ou no tempo de retorno
  - Menos E/S necessário para carregar ou trocar programas em memória: logo cada programa do usuário é executado mais rapidamente

Memória virtual

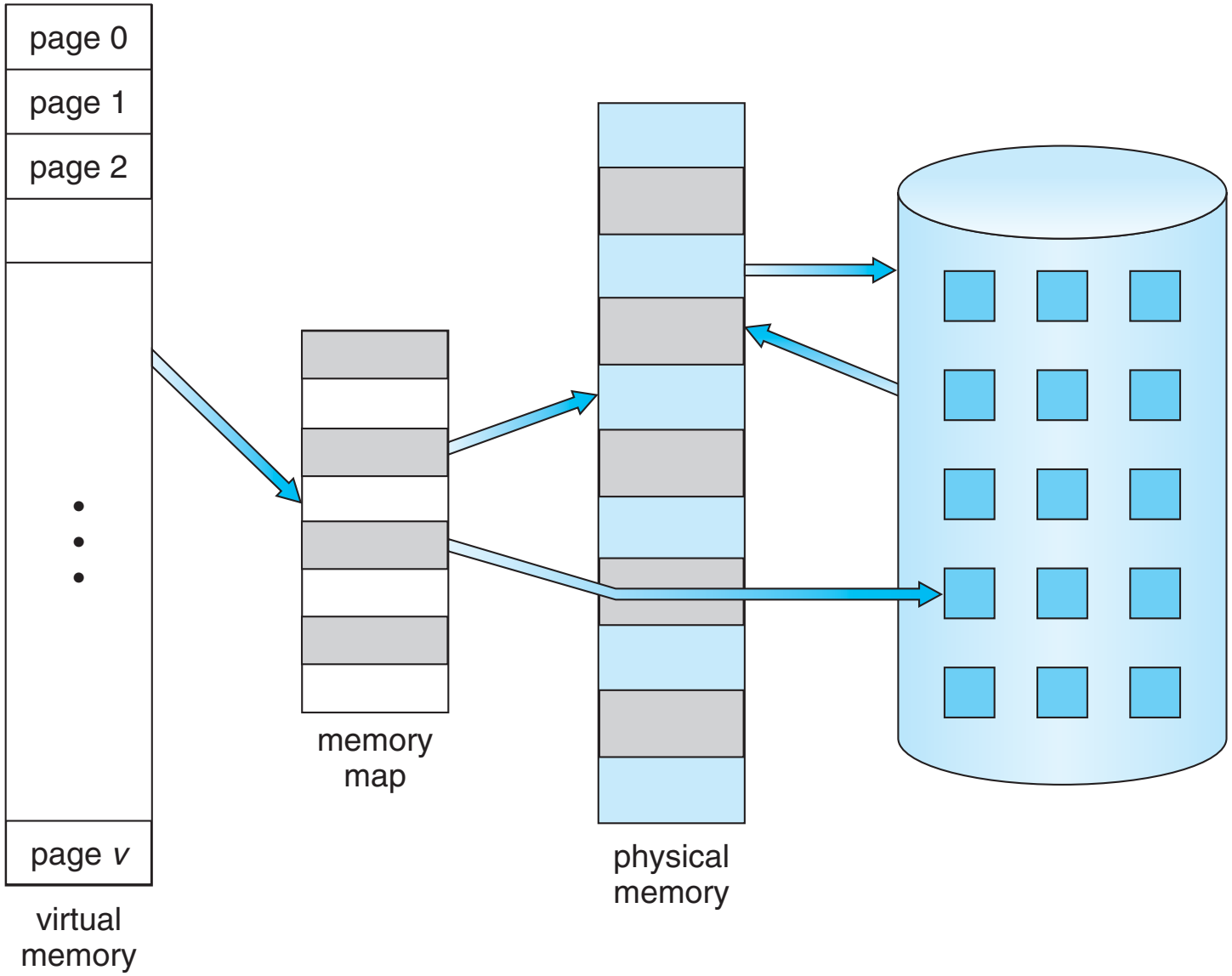
Memória virtual

UFABC

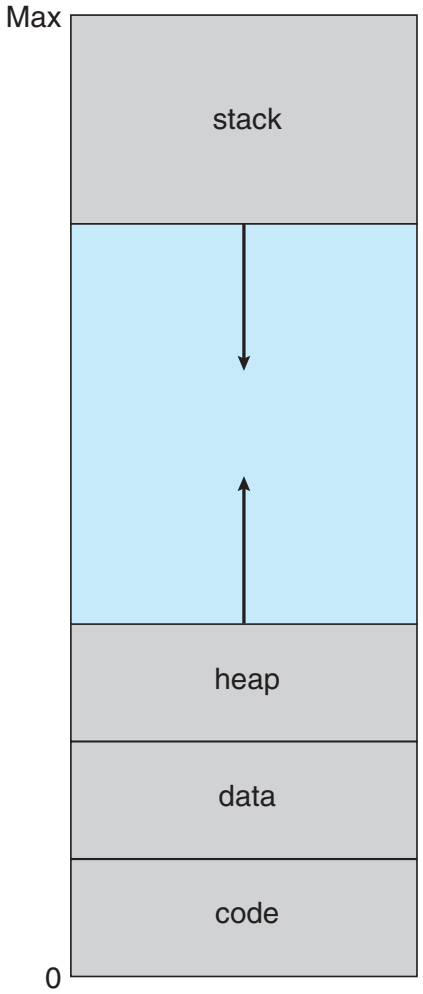
Memória virtual - separação da memória lógica do usuário da memória física

- Apenas parte do programa precisa estar na memória para execução
- O espaço de endereçamento lógico pode, portanto, ser muito maior do que espaço de endereçamento físico
- Permite que os espaços de endereço sejam compartilhados por vários processos
- Permite uma criação de processo mais eficiente
- Mais programas em execução simultaneamente
- Menos E/S necessário para carregar ou trocar processos
- Memória virtual pode ser implementada via:
  - Paginação por demanda
  - Segmentação por demanda

Memória virtual que é maior que a memória física



Espaço de endereço virtual



Espaço de endereço virtual

UFABC

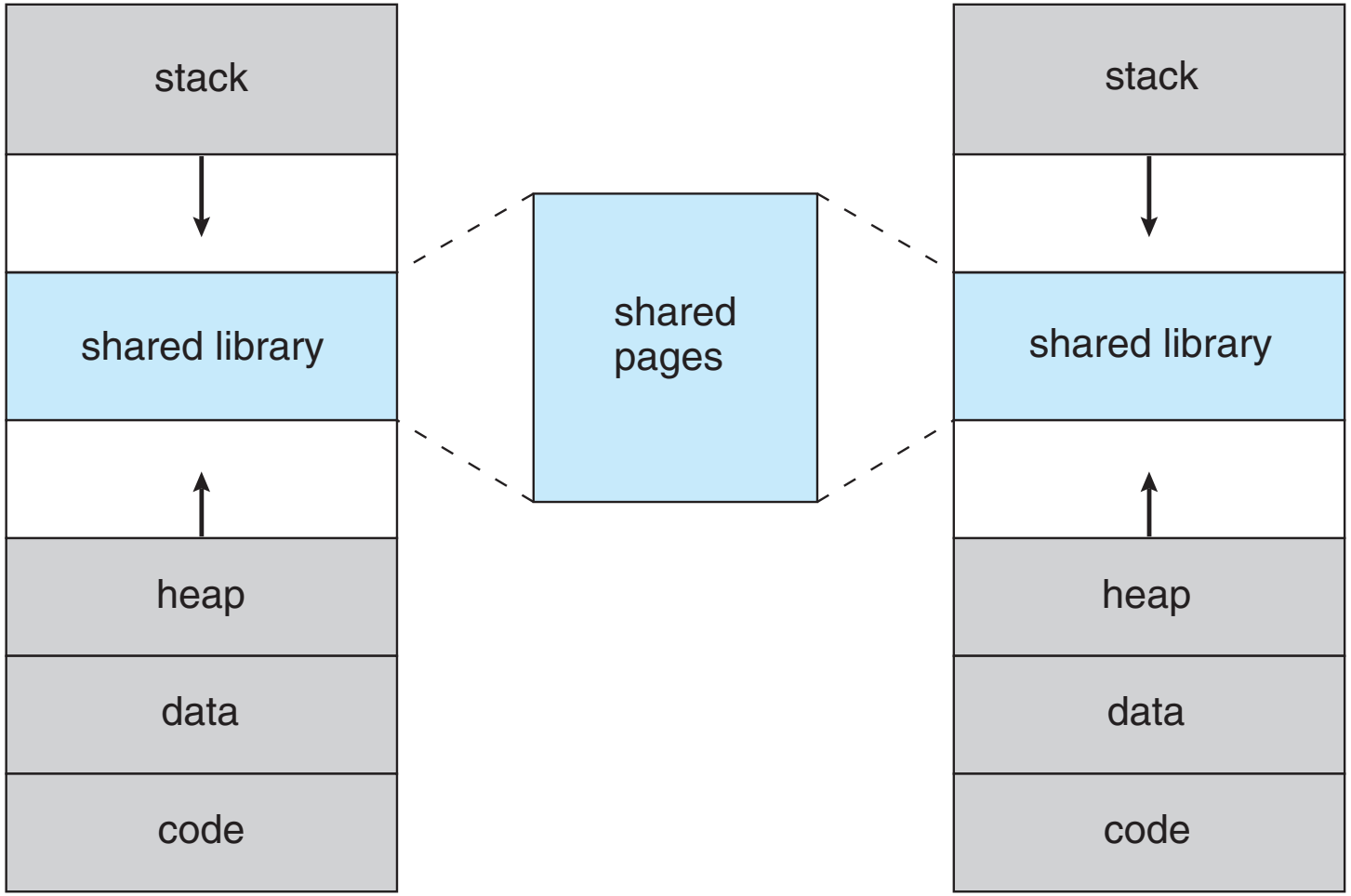
- Geralmente projeta-se o espaço de endereço lógico da pilha para começar no endereço lógico máximo e crescer ”para baixo”enquanto heap cresce ”para cima”
  - Maximiza o uso do espaço de endereçamento
  - Espaço de endereço não utilizado entre os dois é buraco
    - Nenhuma memória física necessária até a pilha ou heap crescerem até uma nova página

Espaço de endereço virtual

UFABC

- Habilita espaços de endereços esparsos com buracos deixados para crescimento, bibliotecas ligadas dinamicamente, etc.
- Bibliotecas compartilhadas via mapeamento no espaço de endereçamento virtual
- Memória compartilhada com páginas mapeadas em modo de leitura-escrita no espaço de endereço virtual
- As páginas podem ser compartilhadas durante o `fork()`, acelerando a criação de processos

Biblioteca compartilhada usando memória virtual



Paginação por demanda

UFABC

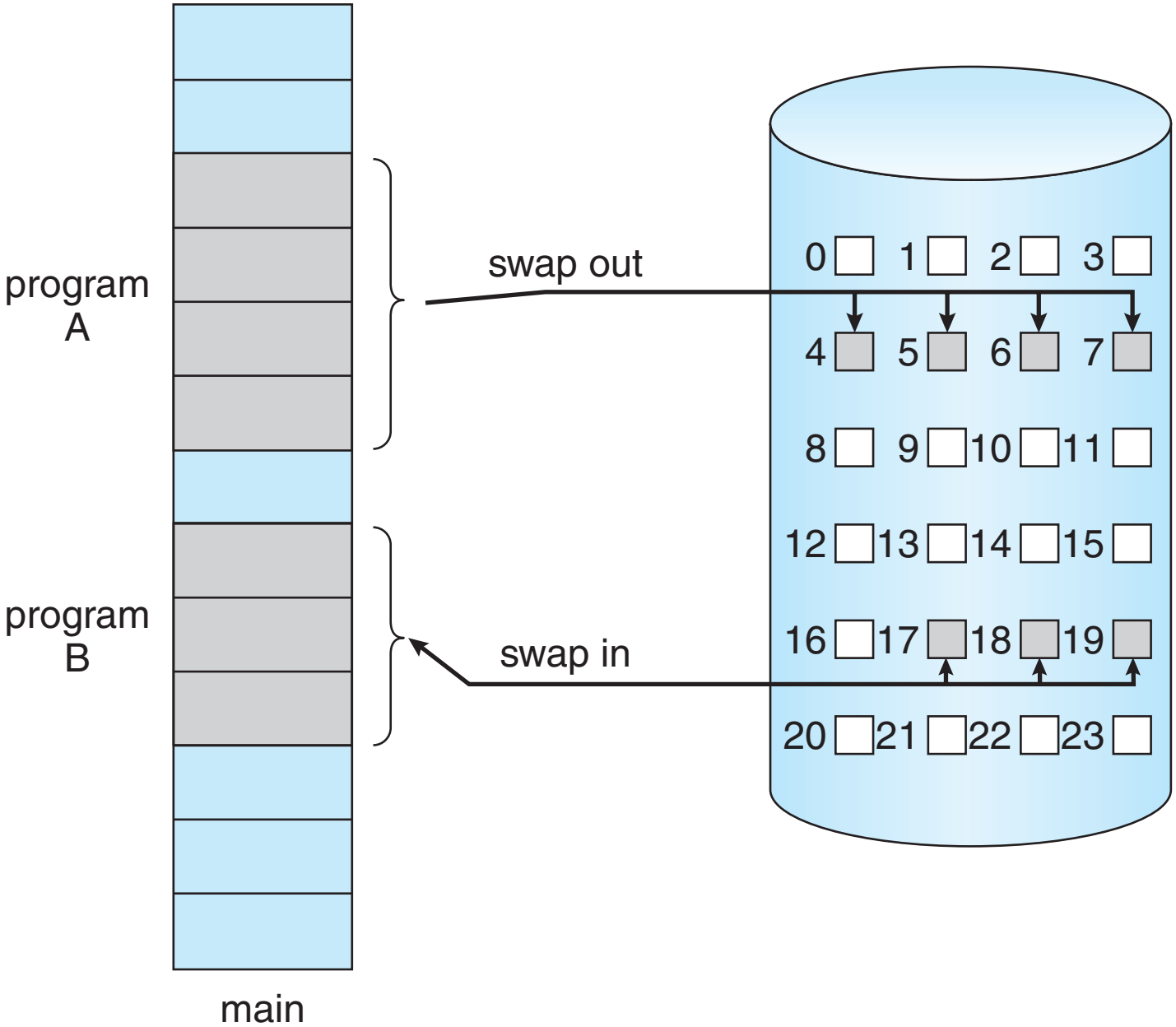
- Poderia trazer todo o processo para a memória em tempo de carregamento
- Ou pode-se trazer uma página para a memória somente quando for necessário
  - Menos E/S necessário, sem E/S desnecessária
  - Menos memória necessária
  - Resposta mais rápida
  - Mais usuários
- Semelhante ao sistema de paginação com troca (diagrama à direita)

Paginação por demanda

UFABC

- Página é necessária $\Longrightarrow$ referência à página
  - referência inválida $\Longrightarrow$ aborta
  - Não está na memória $\Longrightarrow$ trazer para a memória
- *Lazy swapper* - nunca troca uma página para a memória menos que a página seja necessária
  - Swapper que lida com páginas é um paginador

Paginação por demanda



Conceitos básicos

UFABC

- Quando um processo está para ser trocado, o paginador adivinha quais páginas serão usadas antes que o processo seja trocado novamente
- Em vez de trocar em um processo inteiro, o paginador traz apenas aquelas páginas “adivinhadas” na memória
- Precisa de nova funcionalidade na MMU para implementar a paginação por demanda. È preciso distinguir entre as páginas que estão na memória e as páginas que estão no disco. Usa uma variação da marcação de páginas como válidas e inválidas para proteção de memória

Conceitos básicos

UFABC

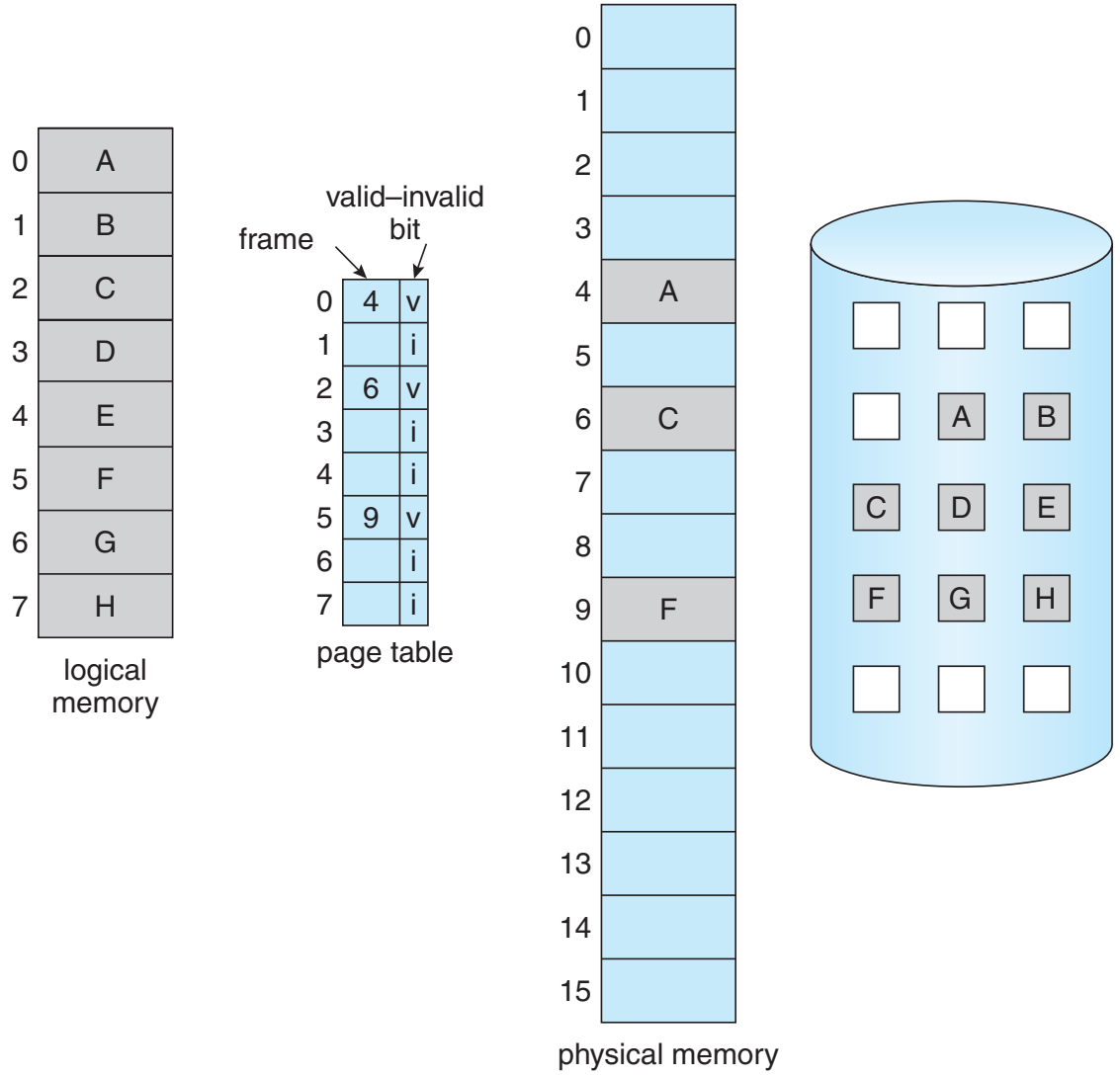
- Se as páginas necessárias já residirem na memória
  - Nenhuma diferença de paginação sob demanda
- Se a página necessária não residir na memória, precisa detectar e carregar a página na memória a partir do disco
  - Sem alterar o comportamento do programa
  - Sem programador precisar alterar o código

Bit Válido-Inválido

UFABC

- Com cada entrada da tabela de páginas, um bit válido e inválido é associado (v $\implies$ na memória, i $\implies$ fora da memória)
- O bit válido-inválido é definido como i em todas as entradas
- Exemplo de um estado da tabela de páginas:
- Durante a tradução do endereço pela MMU, se bit válido-inválido na entrada da tabela de páginas for i $\implies$ falha de página

Tabela de Páginas Quando Algumas Páginas Não Estão na Memória Principal



Falha de página

UFABC

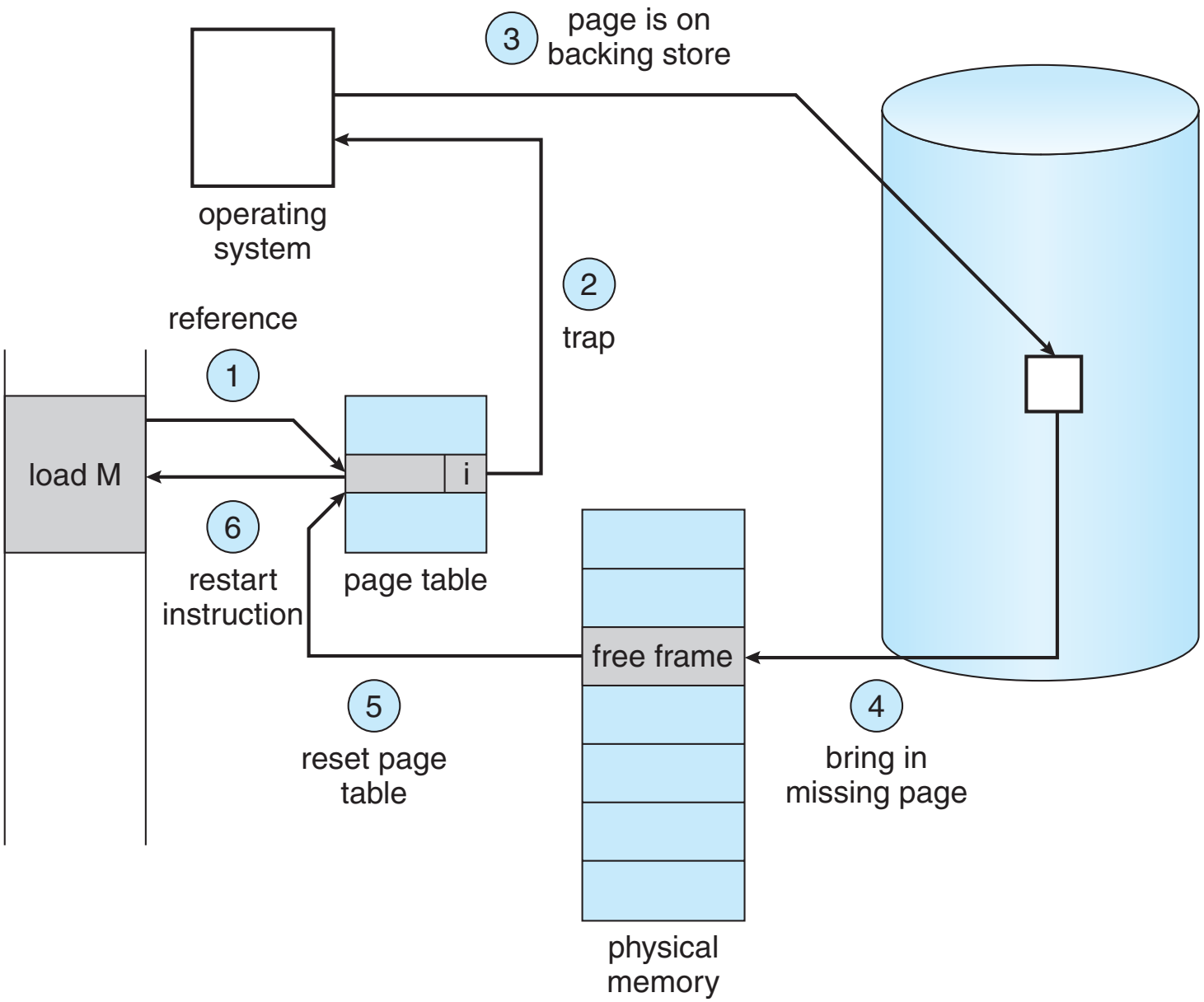
Se houver uma referência a uma página, a primeira referência a essa página será interceptada pelo sistema operacional:

falha de página

Ações quando ocorre uma falha de página

1. O sistema operacional examina outra tabela para decidir:
   - Referência inválida $\Longrightarrow$ abortar
   - Apenas não está na memória. Vá para o passo (2).
2. Encontre um frame livre
3. Troca a página para o frame através de um agendamento de uma operação de disco
4. Atualiza tabelas para indicar que a página agora está na memória Definir bit de validação = v
5. Reinicia a instrução que causou a falha na página

Etapas no tratamento de uma falha de página



Aspectos da paginação por demanda

UFABC

- Paginação por demanda pura - processo de início sem páginas na memória
  - SO define o ponteiro de instruções para a primeira instrução do processo, não residente na memória -> falha de página
  - E para todas as outras páginas de processo no primeiro acesso
- Na verdade, uma determinada instrução pode acessar várias páginas $\rightarrow$ falhas de várias páginas
  - Considere a instrução de busca e decodificação de uma instrução que adicione 2 números da memória e armazene o resultado de volta à memória
  - Os dois números podem residir em duas páginas diferentes

Aspectos da paginação por demanda

UFABC

- Suporte de hardware necessário para paginação por demanda
  - Tabela de páginas com bit válido / inválido
  - Memória secundária (dispositivo de troca com espaço de troca)
  - Reiniciação da instrução

Reinício de Instrução

UFABC

- Considere uma instrução que possa acessar vários locais diferentes
  - Mover bloco
  - Falha de página durante o movimento
  - Reinicie toda a operação?
- E se a origem e o destino se sobrepuserem?
- Várias soluções. O mais simples é descobrir todas as páginas que são necessárias para executar a instrução até o fim e garantir que essas páginas estejam na memória antes da instrução começa a executar.

Estágios na paginação por demanda

UFABC

Estágios na paginação por demanda (pior caso)

1. *Trap* para o sistema operacional
2. Salva os registradores e o estado do processo de usuário
3. Determina se a interrupção foi uma falha de página
4. Verifica se a referência da página era legal e determina a localização da página no disco
5. Emite uma E/S de leitura do disco para um frame livre:
    1. Aguarda em uma fila por este dispositivo até a solicitação de leitura ser atendida
    2. Aguarda a busca do dispositivo e/ou o tempo de latência
    3. Começa a transferência da página para um frame livre
6. Enquanto espera, aloca a CPU para algum outro usuário

Estágios na paginação por demanda (cont.)

UFABC

7. Recebe uma interrupção do subsistema de E/S de disco (E/S concluído).
8. Salva os registradores e o estado do processo de usuário
9. Determina se a interrupção foi do disco
10. Corrige a tabela de páginas e outras tabelas para mostrar que a página está agora na memória
11. Aguarda a CPU ser alocada para este processo novamente
12. Restaura os registradores, o estado do processo do usuário e a nova tabela de páginas e retoma a instrução interrompida

Desempenho da paginação por demanda

UFABC

- Três atividades principais
  - Manutenção da interrupção - codificação cuidadosa significa apenas algumas centenas de instruções necessárias
  - Leitura na página - muito tempo
  - Reinicilização do processo - apenas uma pequena quantidade de tempo
- Taxa de falha de página $0 \leq p \leq 1$
  - se p = 0 sem falhas de página
  - se p = 1, toda referência é uma falha
- Tempo de Acesso Efetivo (EAT) EAT = (1 - p) x acesso à memória + p (sobrecarga de falha de página + troca de página (out) + troca de página (in))

EAT Exemplo

UFABC

- Tempo de acesso à memória = 200 nanossegundos
- Tempo médio de atendimento de falha de página = 8 milissegundos
- EAT = (1 - p) x 200 + p (8 milissegundos) = (1 - p x 200 + p x 8.000.000 = 200 + p x 7.999.800
- Se um acesso de 1.000 causar uma falha de página, EAT = 8,2 microssegundos. Esta é uma perda de desempenho de 40x!
- Se quiser degradação de desempenho < 10%
  - 220 > 200 + 7.999.800 x p 20 > 7.999.800 x p
  - p < 0.0000025
  - Menos que uma falha de uma página a cada 400.000 acessos de memória

Otimizações de paginação de demanda

UFABC

- Espaço de swap para mais rápido que a E/S do sistema de arquivos, mesmo se no mesmo dispositivo
  - Espaço de troca alocada em trechos maiores, menos gerenciamento necessário do que o sistema de arquivos
- Copie toda a imagem do processo para o espaço de troca no momento do carregamento do processo
  - Em seguida, paginar do espaço de troca
  - Usado no antigo BSD Unix
- Fazer paginação do binário a partir do disco, mas descarta a página em vez de paginar quando for liberar o frame
  - Usado no Solaris e no BSD atual
  - Ainda precisa escrever no espaço de troca
    - Páginas não associadas a um arquivo (como pilha e heap) - memória anônima
    - Páginas modificadas na memória, mas ainda não gravadas no sistema de arquivos

Paginação por demanda em sistemas móveis

UFABC

- Normalmente não suportam swapping
- Em vez disso, solicitam a página do sistema de arquivos e recuperam apenas páginas de leitura (como código)

Cópia-na-gravação

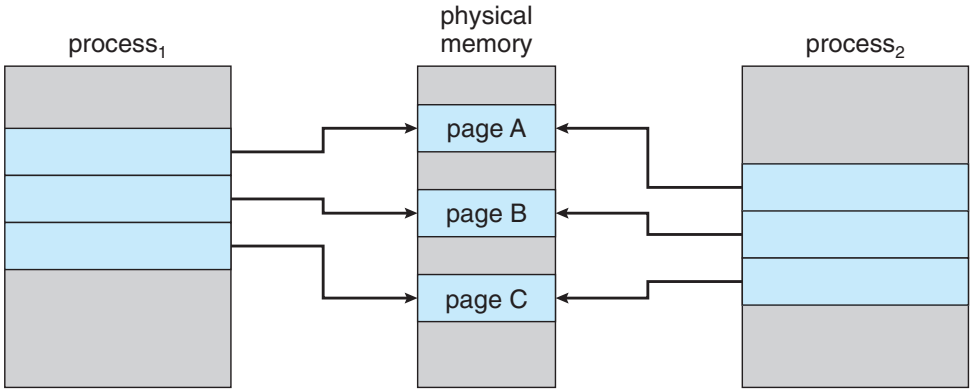
UFABC

- Copy-on-Write (CoW) permite que os processos pai e filho compartilhem inicialmente as mesmas páginas na memória
  - Se um dos processos modificar uma página compartilhada, somente então a página será copiada
- COW permite a criação de processos mais eficientes, pois apenas as páginas modificadas são copiadas
- Em geral, as páginas livres são alocadas de um pool de páginas que são preenchidas com zero sob demanda
  - O pool deve sempre ter frames livres para execução rápida
    - Não desejamos ter que liberar um frame quando estamos no meio do processamento de falhas de página
- `vfork()` variação na chamada do sistema `fork()` faz o pai suspender a execução e o filho utilizar CoW no espaço de endereçamento do pai
  - Projetado para que o filho faça a chamada do `exec()`
  - Muito eficiente

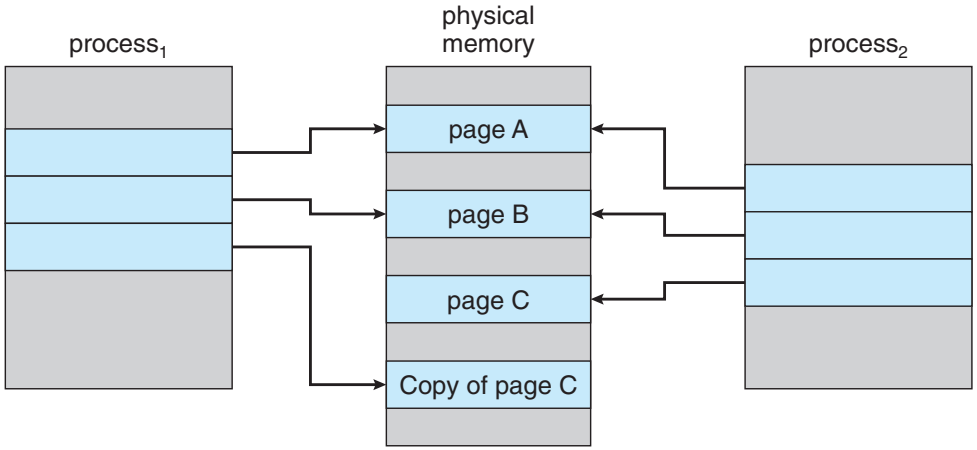
Antes e depois do processo 1 modifica a página C

UFABC

Antes



Depois



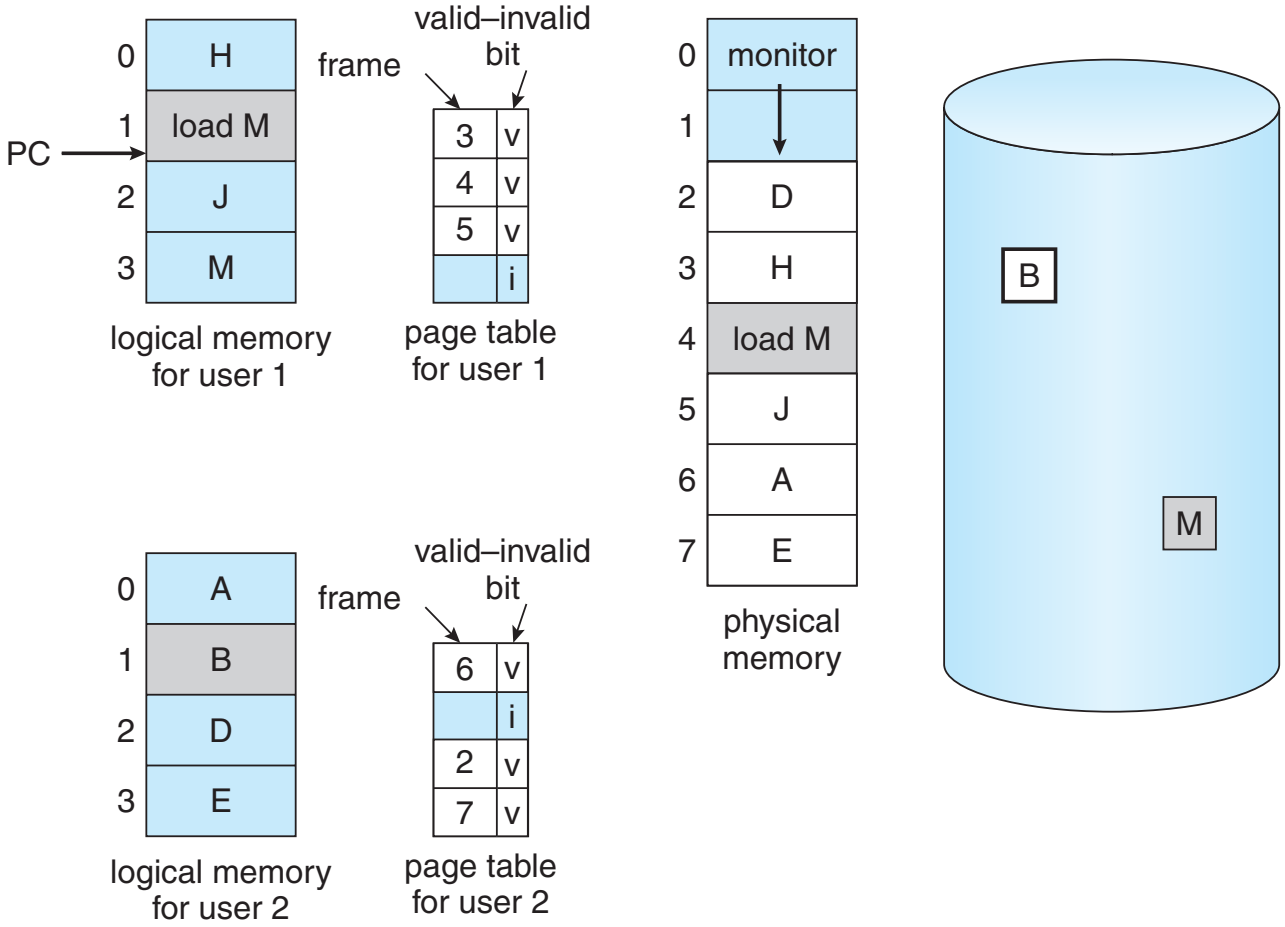
Substituição de Página

UFABC

- Substituição de página ocorre quando:
  - Ocorre uma falha de página e precisamos trazer a página desejada à memória
  - Não há frames livres.
- Substituição de página - encontre alguma página na memória, que não está em uso e a grave no espaço de swap
  - Algoritmo - decide qual frame liberar
  - Desempenho - queremos um algoritmo que irá resultar em número mínimo de falhas de página
- A mesma página pode ser trazida para a memória várias vezes

Necessidade de substituição de página

Todos os frames estão em uso. Nenhum frame livre. Usuário 2 precisa de B



Substituição de página (cont.)

UFABC

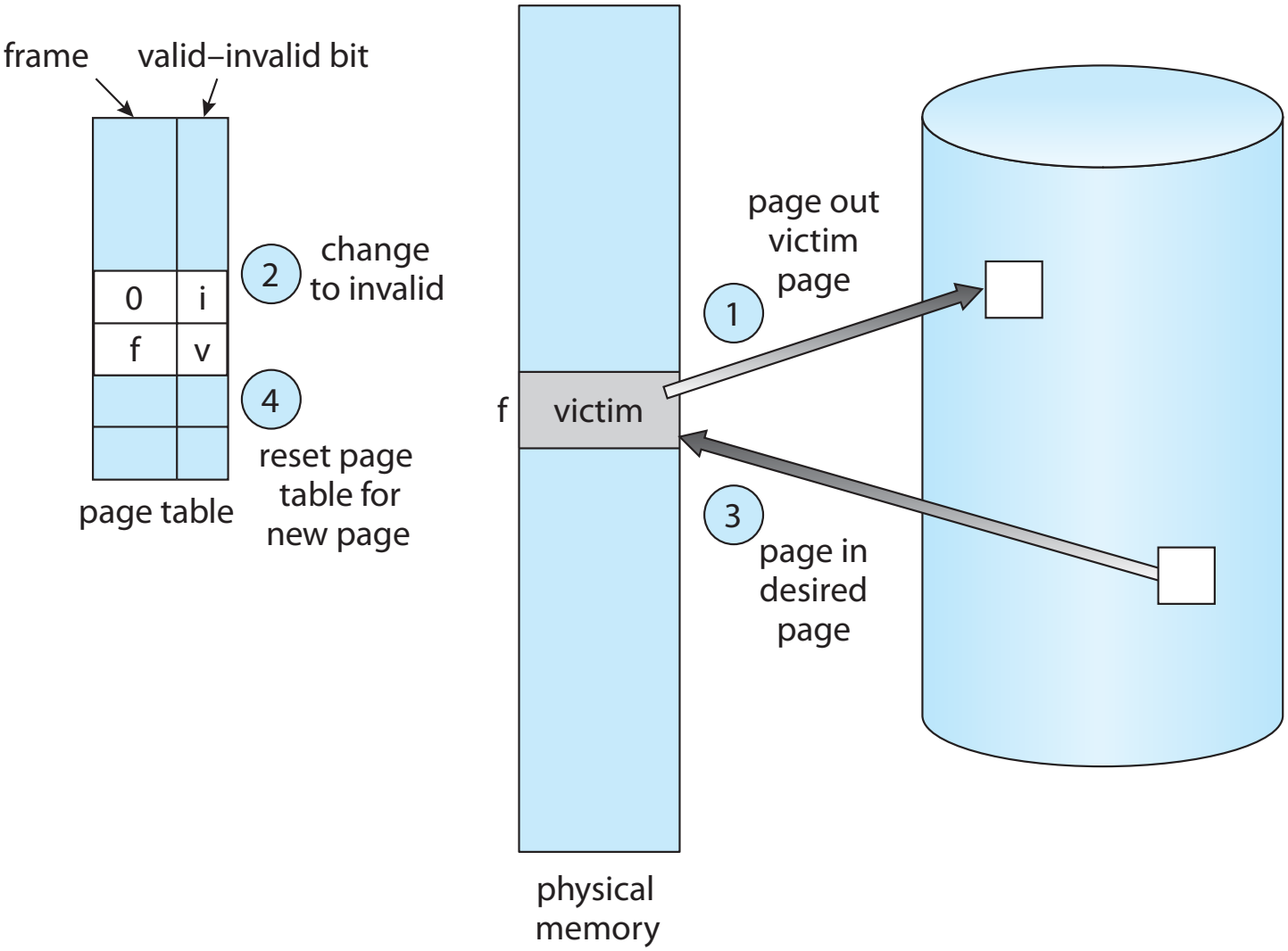
- Use bit de modificação (sujo) para reduzir a sobrecarga das transferências de página - apenas páginas modificadas são escritas de volta para disco
- Substituição de página completa a separação entre memória lógica e memória física - grande memória virtual pode ser fornecida em uma memória física menor

Substituição de Página Básica

UFABC

1. Encontre a localização da página desejada no disco
2. Encontre um frame livre:
   - Se houver um frame livre, use-o
   - Se não houver frame livre, use um algoritmo de substituição de páginas:
     - Selecione um frame vítima
     - Escreva o frame vítima no disco, se estiver sujo
3. Traga a página desejada para o (novo) frame livre; atualize a tabela de páginas e frames
4. Continue o processo reiniciando a instrução que causou o trap

Substituição de Página



Algoritmos de Substituição de Página e Quadro

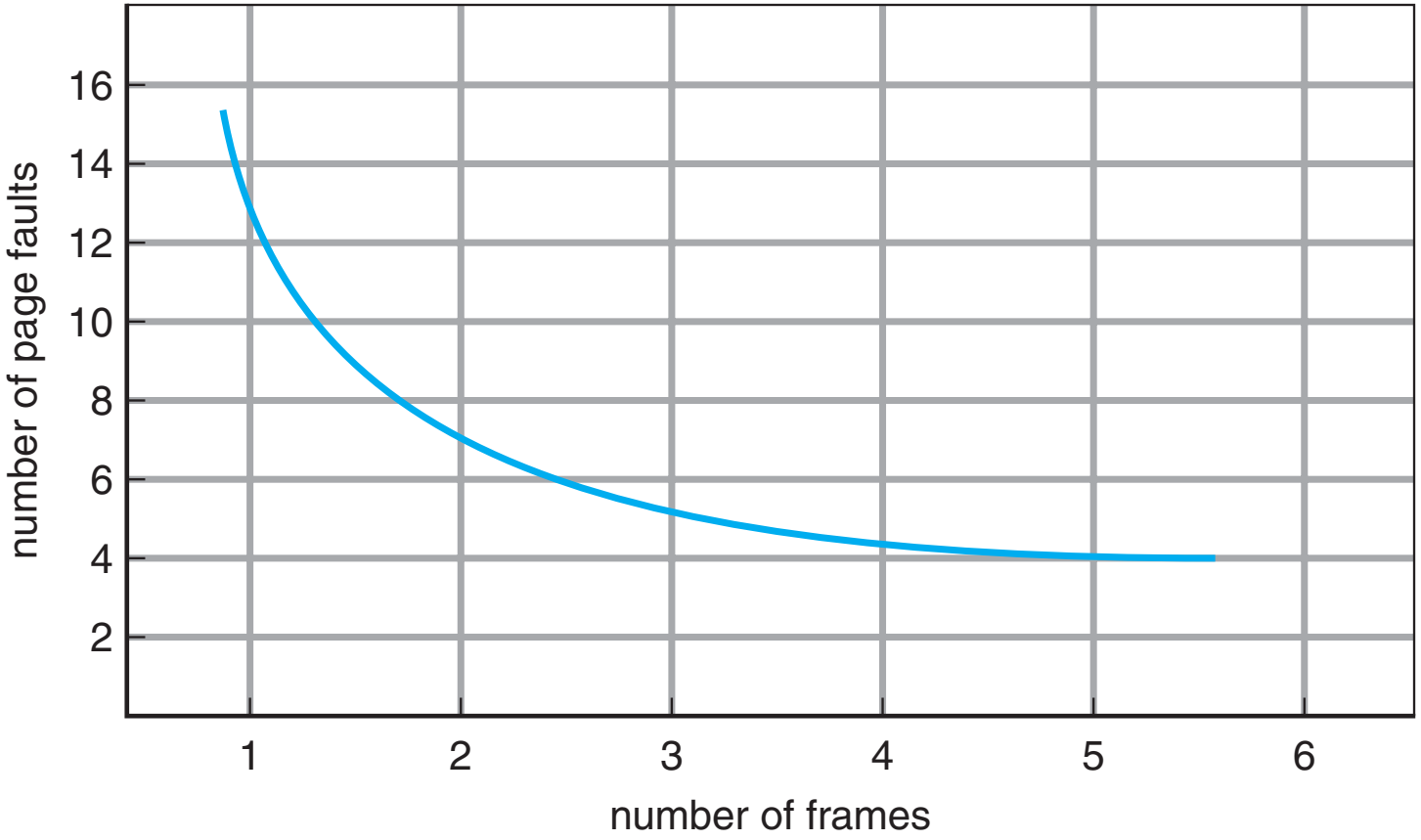
UFABC

- O algoritmo de alocação de frames determina
  - Quantos frames para cada processo
  - Quais frames substituir
- Algoritmo de substituição de páginas
  - Queremos a menor taxa de falha de página no primeiro acesso e nos reacessos

- Avalie o algoritmo executando-o em uma sequência de acessos à memória específica (string de referência) e calcule o número de falhas de páginas nessa cadeia
  - A sequência consiste apenas de números de página, não endereços completos
  - O acesso repetido à mesma página não causa uma falha de página
  - Os resultados dependem do número de frames disponíveis
- Em todos os nossos exemplos, a string de referências às páginas é

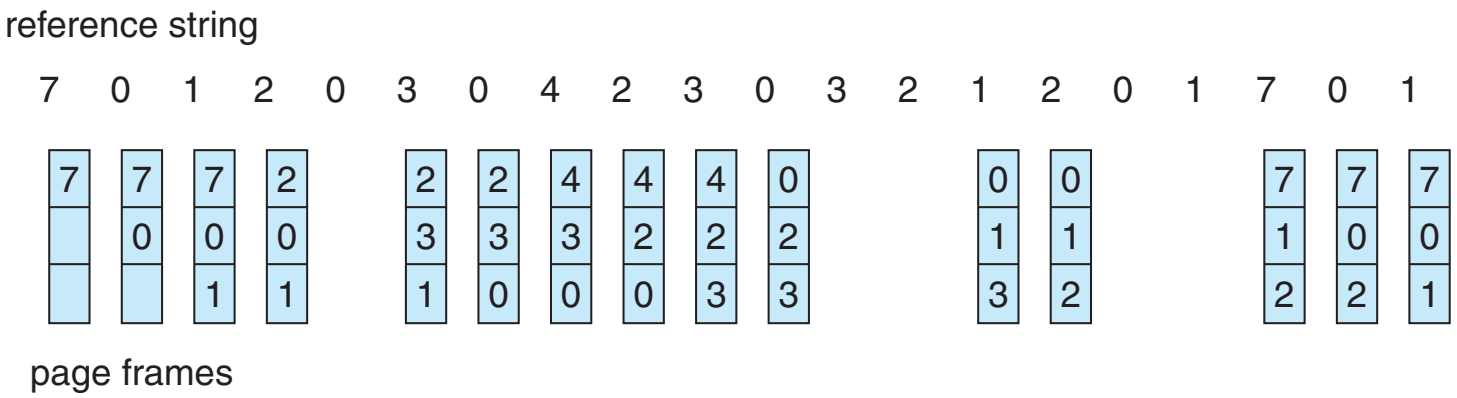
  `7,0,1,2,0,3,0,4,2,3,0,3,0,3,2,1,2,0,1,7,0,1`

Gráfico de falhas de página versus o número de frames



Algoritmo First-In-First-Out (FIFO)

- String de referências:
  `7,0,1,2,0,3,0,4,2,3,0,3,0,3,2,1,2,0,1,7,0,1`
- 3 frames (3 páginas podem estar na memória por vez por processo)



15 falhas de página

- Como rastrear idades de páginas?
  - Basta usar uma fila FIFO

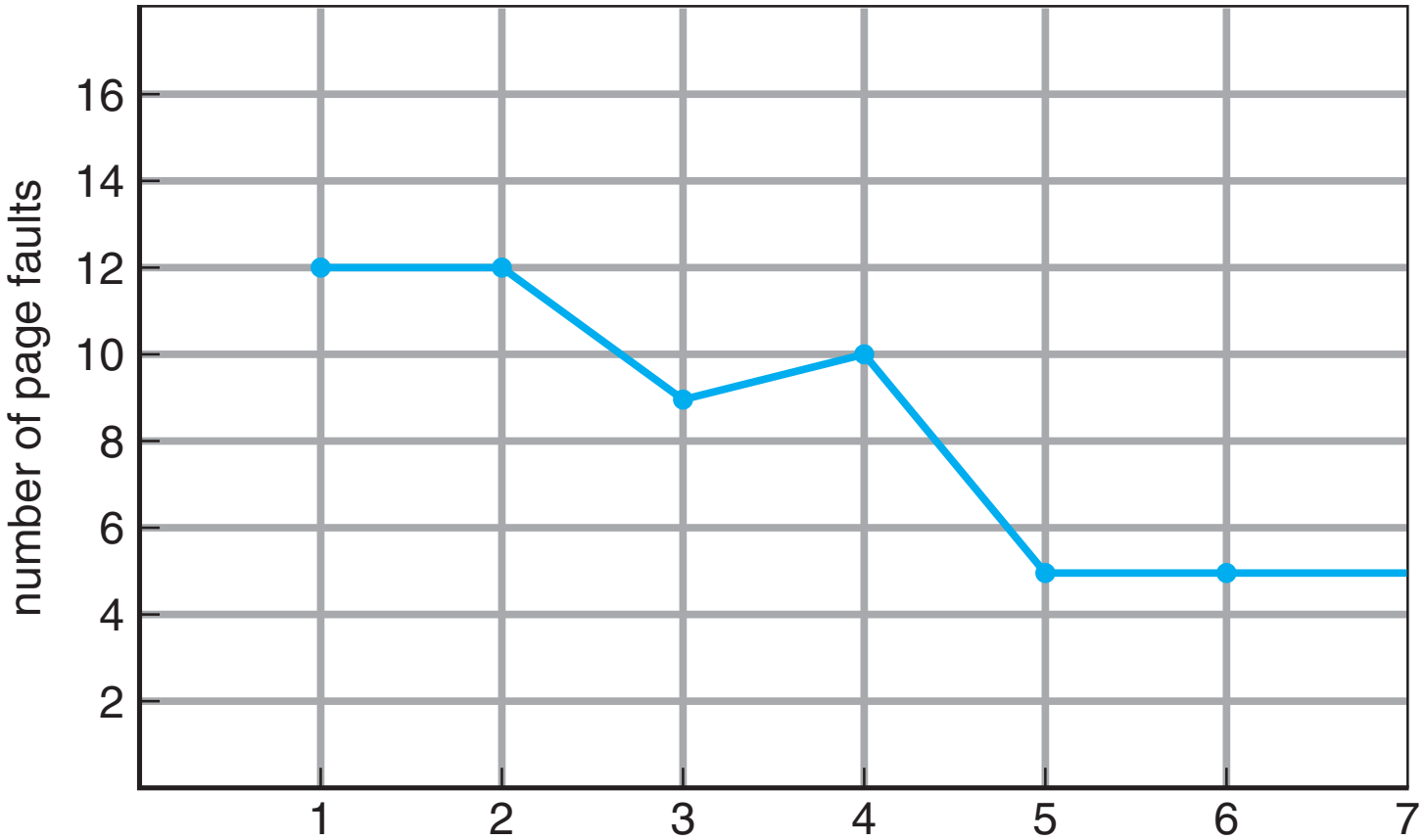
Número de frames vs falhas de página

UFABC

- Seria de esperar que quanto mais frames forem alocados para um processo, menos falhas de página
- Considere a string de referência:
  `1,2,3,4,1,2,5,1,2,3,4,5`
- Quantas falhas de página, se tivermos 3 frames?
- Quantas falhas de página, se tivermos 4 frames?

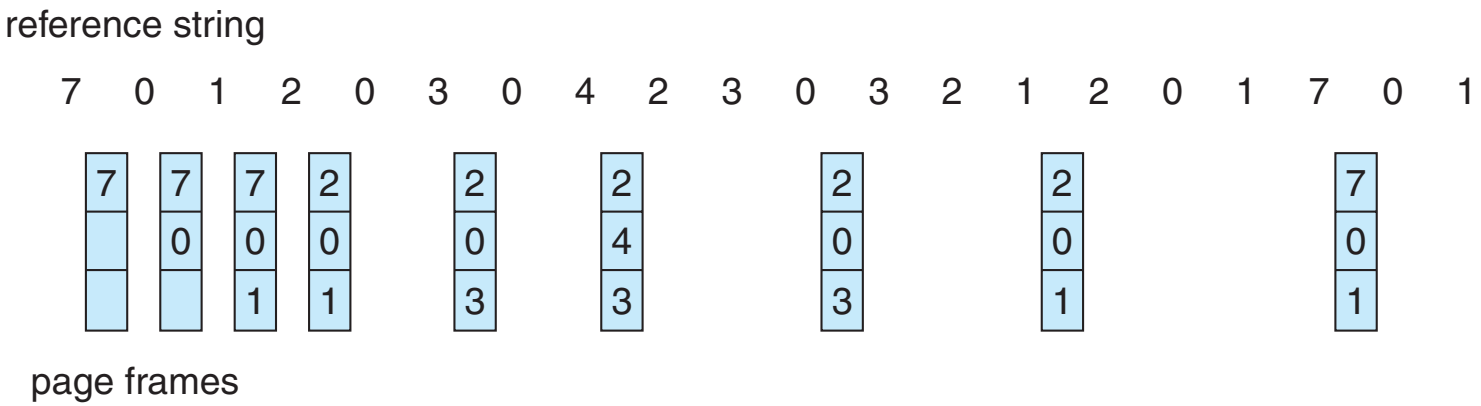
Anomalia de Belady

- Se usarmos FIFO para substituição de página, poderemos encontrar anomalia que mais frames podem levar a mais falhas de página
- FIFO ilustrando a anomalia de Belady



Algoritmo Ótimo (OPT)

- Substitua a página que não será usada por um período de tempo mais longo
  - Com 3 frames, 9 é ideal para a referência de exemplo.

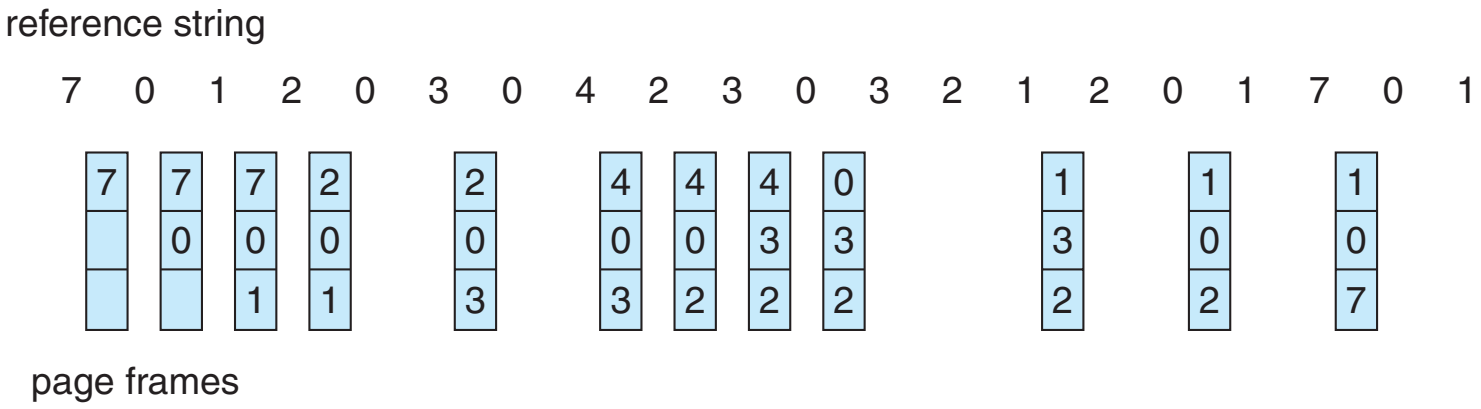


- Como você sabe qual página não será usada por período mais longo de tempo?
  - Não consigo prever o futuro
- Usado principalmente para medir o desempenho de um determinado algoritmo

Algoritmo Menos Utilizado Recentemente (LRU)

UFABC

- Usa conhecimento passado ao invés de futuro
- Substitui a página que não foi usada pelo maior período de tempo.
- Associa o tempo do último uso a cada página



- 12 falhas - melhor que FIFO, mas pior que OPT
- Geralmente bom algoritmo e é frequentemente usado
- Mas como implementamos?

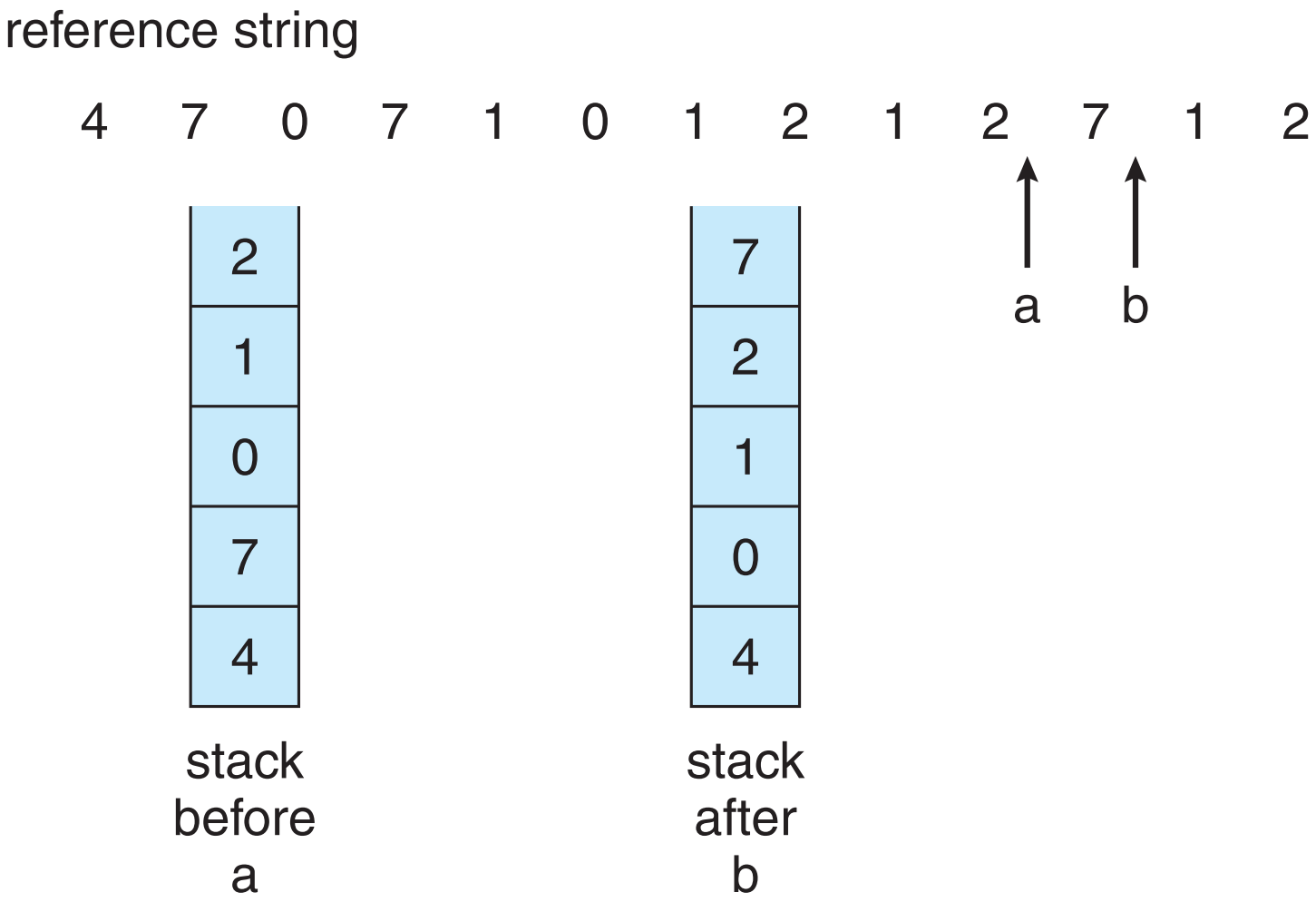
Algoritmo LRU (Cont.)

UFABC

- Implementação do contador (tempo de uso)
  - Toda entrada de tabela de páginas possui um contador associado a ela; toda vez uma página é referenciada através desta entrada, o conteúdo do relógio é copiado no contador
  - Substituímos a página com o menor valor de tempo.
    - Pesquisa através da tabela necessária
- Implementação de pilha
  - Mantenha uma pilha de números de página em um formato duplamente encadeado:
  - Página referenciada:
    - mova-a para o topo
    - requer que 6 ponteiros sejam alterados
  - Nenhuma busca para fazer uma substituição
- LRU precisa de hardware especial e ainda é lenta
- LRU e OPT são casos de algoritmos de pilha que não sofrem de Anomalia de Belady

Exemplo de Algoritmo de Pilha

Uso de uma pilha para gravar as referências de páginas mais recentes



Algoritmos de Aproximação LRU

UFABC

Para obter uma implementação eficiente, usamos uma aproximação de LRU

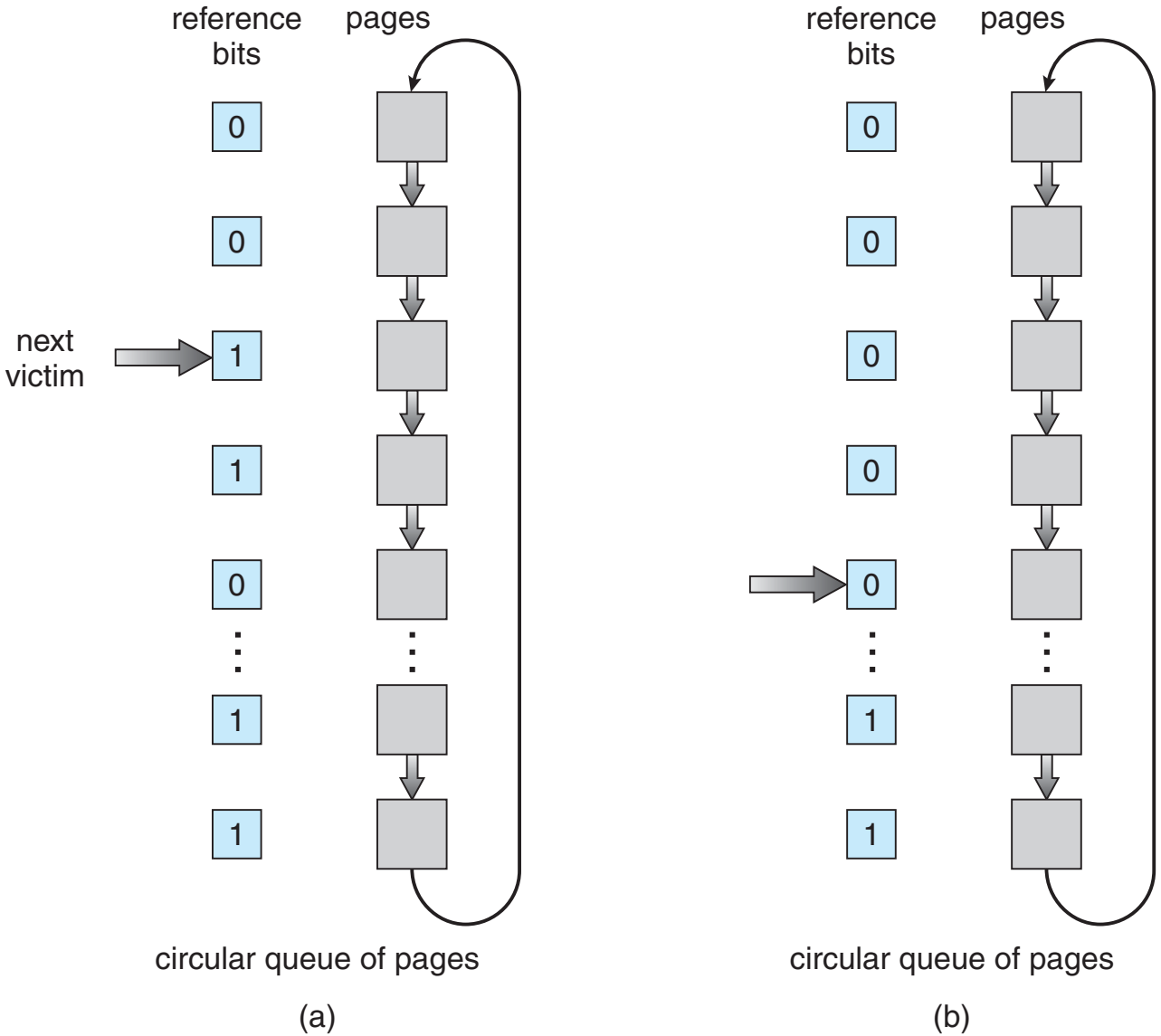
- Bit de referência
  - Associe um bit fornecido pelo hardware para cada página; inicialmente = 0
  - Quando uma página é referenciada, o bit associado é definido como 1
  - Substitua qualquer página pelo bit de referência = 0 (se existir)
    - Nós não sabemos a ordem, entanto no

UFABC

Algoritmo de segunda chance

- Esquema FIFO, além de bit de referência fornecido pelo hardware
- Se a página a ser substituída tiver
  - Bit de referência = 0 $\rightarrow$ substitua
  - Bit de referência = 1 então:
    - definir bit de referência 0, deixar a página na memória
    - substituir a próxima página, sujeita às mesmas regras

Algoritmo de substituição de página de segunda chance (relógio)



Algoritmo Aprimorado de Segunda Chance

UFABC

- Melhora o algoritmo usando o bit de referência e modifica o bit (se disponível) em conjunto
- Tome um par ordenado (referência, modificação)
  1. (0, 0) nem recentemente usado não modificado - melhor página para substituir
  2. (0, 1) não usado recentemente, mas modificado - não tão bom, deve escrever antes da substituição
  3. (1, 0) usado recentemente mas limpo - provavelmente será usado novamente em breve
  4. (1, 1) recentemente usado e modificado - provavelmente será usado novamente em breve e precisa escrever antes da substituição
- Quando for solicitada a substituição da página, use o esquema do relógio, mas as quatro classes substituem a página na menor classe não vazia
  - Pode precisar procurar fila circular várias vezes

Algoritmos de Contagem

UFABC

- Mantenha um contador do número de referências que foram feitas para cada página
- Algoritmo **Least Frequently Used (LFU)**: substitui a página com menor contagem
- Algoritmo **Most Frequently Used (MFU)**: baseado no argumento de que a página com a menor contagem foi provavelmente carregada e ainda não foi utilizada
- Tanto o LFU quanto o MFU são caros e não são comumente usados.

Algoritmos de buffer de páginas

UFABC

Outros procedimentos são frequentemente usados além de um algoritmo de substituição de páginas específico. Várias organizações são possíveis:

- Mantenha um pool de frames livres
  - Quando ocorre uma falha de página, uma página vítima é escolhida como antes.
  - A página desejada que precisa ser trazida para a memória é lida em um dos frames livres do pool antes que a página da vítima seja escrita. Inicia o processo imediatamente.
  - Quando a página da vítima é finalmente escrita, o frame é adicionado ao poll de frames livres

Algoritmos de buffer de páginas

UFABC

- Manter uma lista de páginas modificadas
  - Sempre que o disco estiver ocioso, uma página modificada é selecionada e gravada no disco e seu bit modificado é redefinido.
- Possivelmente, mantenha um conjunto de frames livres e lembre em qual página estava cada frame
  - Se a página for referenciada novamente antes que o frame seja reutilizado, não é necessário carregar conteúdo novamente do disco
  - Geralmente útil para reduzir a penalidade se o frame de vítima errado for selecionado

Aplicações e Substituição de Páginas

UFABC

- Todos esses algoritmos fazem suposições do sistema operacional sobre futuros acessos às páginas
- Algumas aplicações têm melhor conhecimento - por exemplo, bancos de dados
- Aplicativos com uso intensivo de memória podem acabar criando double buffering
  - O SO mantém cópia da página na memória como buffer de E/S
  - Aplicação mantém página na memória para seu próprio trabalho
- Sistema operacional pode ter acesso direto ao disco, saindo do caminho das aplicações
  - Modo de disco cru (raw)
- Ignora buffering, travas, etc

Alocação de Frames

UFABC

- Cada processo precisa de um número mínimo de frames
- Exemplo: IBM 370 - 6 páginas para lidar com a instrução SS MOVE:
  - instrução é de 6 bytes, pode abranger duas páginas
  - 2 páginas para lidar com *from*
  - 2 páginas para lidar com *to*
- Máximo, é claro, é o total de frames no sistema
- Dois grandes esquemas de alocação
  - alocação fixa
  - atribuição de prioridade
- Muitas variações

Alocação de Quadros

UFABC

- Alocação igual - divide m frames entre n processos igualmente - m / n Por exemplo, se houver 100 frames (após a alocação de frames para o OS) e 5 processos, cada processo recebe 20 frames
  - Manter alguns no pool de frames livres
  - Alocação proporcional - aloca de acordo com o tamanho do processo
    - Um processo “grande” precisa de mais frames? Localidade?

Alocação global vs. local

UFABC

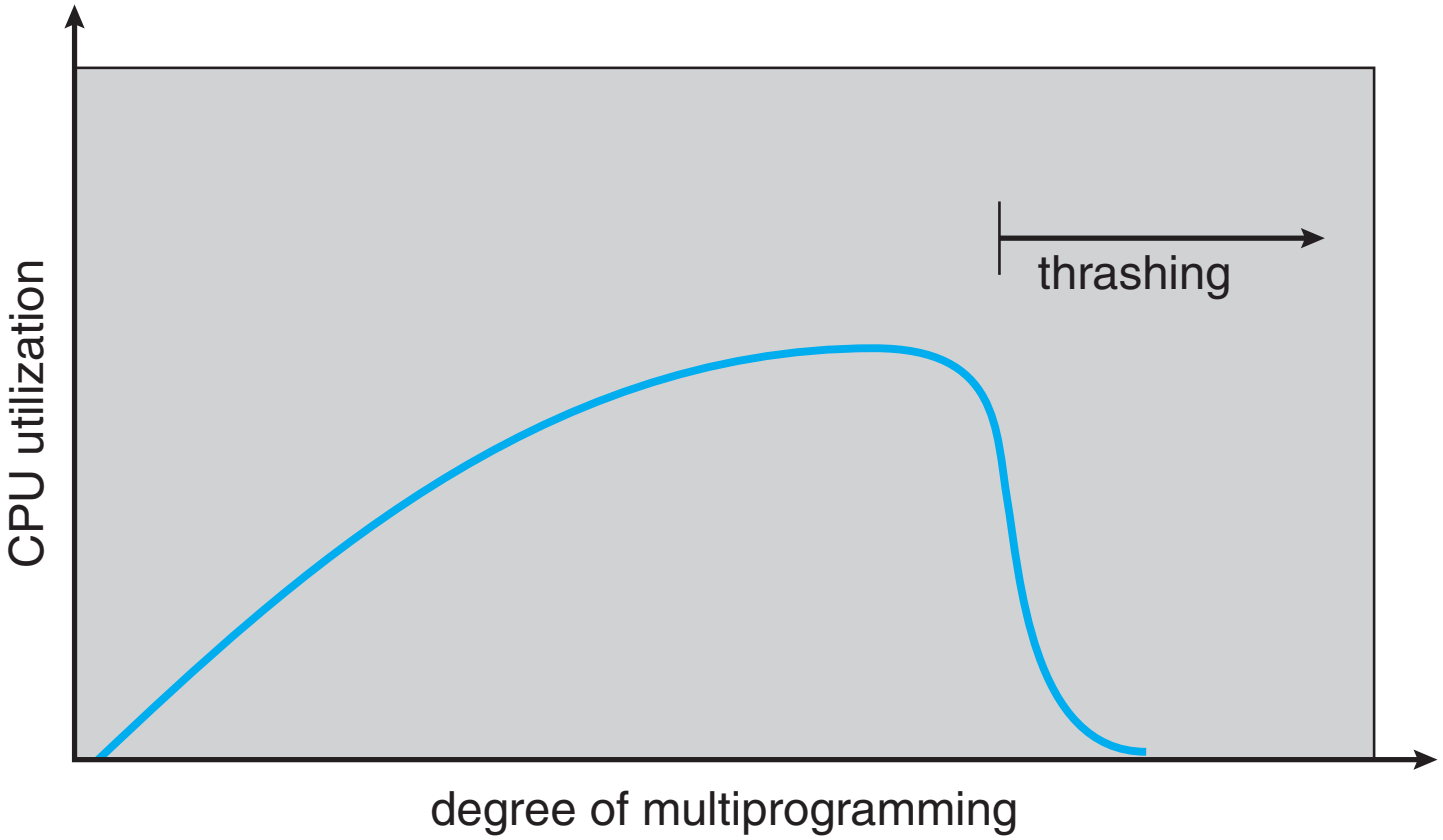
- Substituição global - processo seleciona um frame de substituição do conjunto de todos os frames; um processo pode tomar um frame de outro
  - O tempo de execução do processo pode variar muito
  - Um processo não pode controlar sua própria taxa de falha de página
  - Maior rendimento, portanto, mais comum
- Substituição local - cada processo seleciona apenas o seu próprio conjunto de frames alocados
  - Desempenho mais consistente por processo
  - Se um processo não tiver número suficiente de frames alocado a ele, o processo sofrerá muitas falhas de página (thrashing).
  - Possivelmente memória subutilizada

Thrashing

UFABC

- Se um processo não tiver páginas “suficientes”, a taxa de falhas de página pode ser muito alta
  - Falha de página para obter a página
  - Substituir o frame existente
  - Mas rapidamente precisa de frame substituído de volta
  - Isso leva a:
    - Baixa utilização da CPU
    - Sistema operacional que pensa que precisa aumentar o grau de multiprogramação
    - Outro processo adicionado ao sistema
- Thrashing $\equiv$ um processo está ocupado trocando páginas

Thrashing (cont.)



Paginação de demanda e thrashing

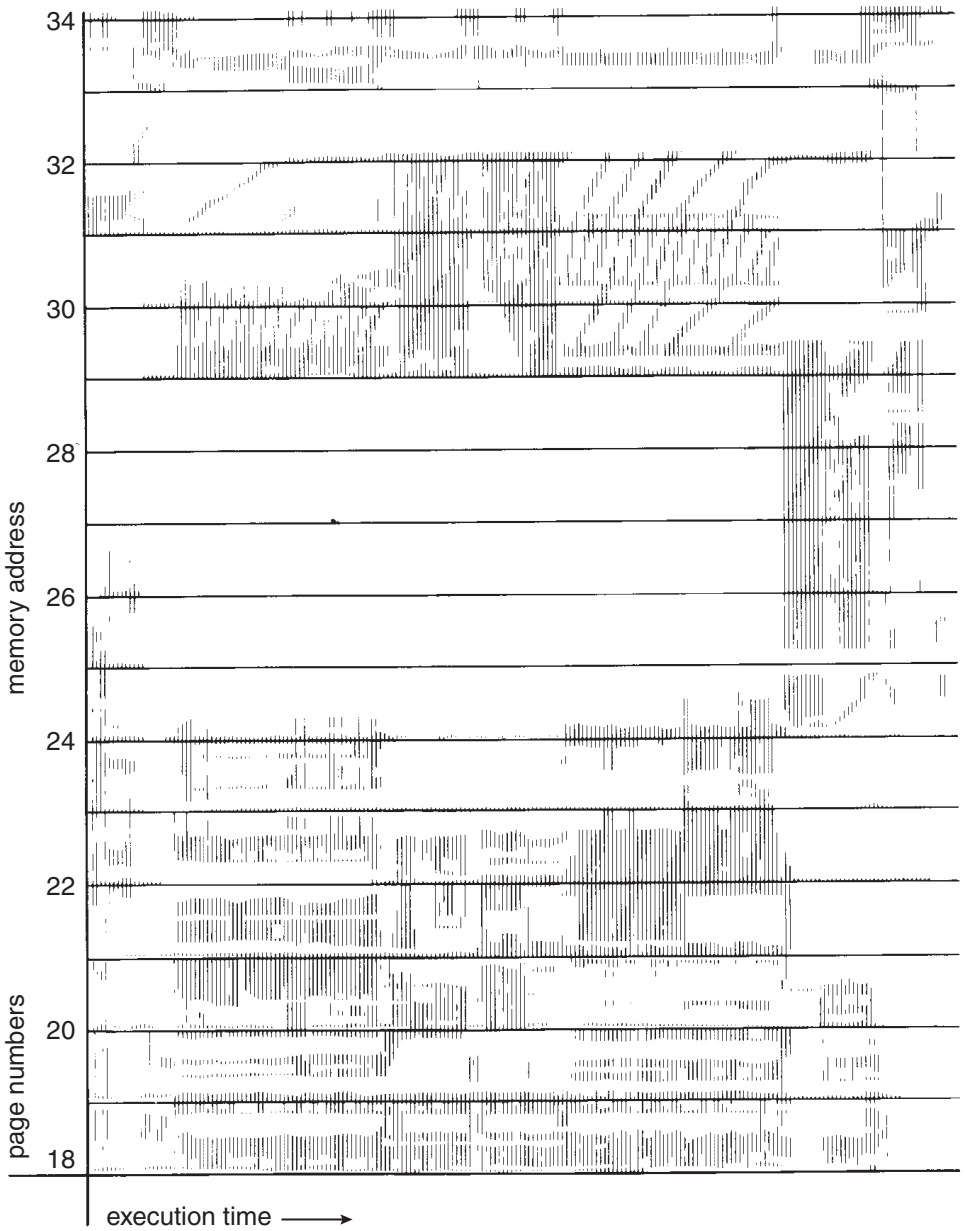
UFABC

- Por que a demanda de paginação funciona? **Modelo de localidade**
  - O processo migra de uma localidade para outra
  - As localidades podem se sobrepor
- Por que thrashing ocorre?

Σ tamanho da localidade> tamanho total da memória

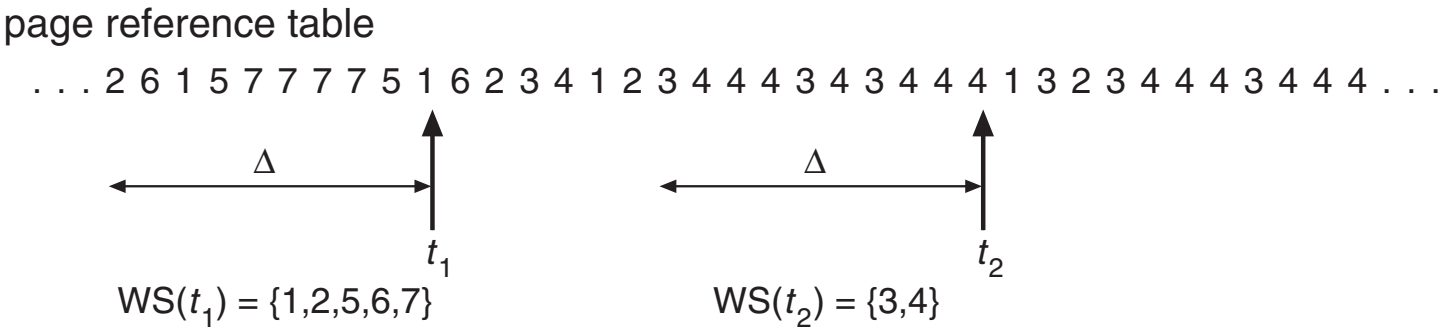
- Pode-se limitar os efeitos de thrashing usando:
  - Substituição de página local
  - Substituição de página baseada em prioridades - substitui uma página de um processo com a menor prioridade.

Localidade em um padrão de referência de memória



Modelo de Conjunto de Trabalho

- Definir Δ para ser uma janela de conjunto de trabalho. Δ é um número fixo de referências de página Por exemplo: 10.000 instruções
- WSS_i (conjunto de trabalho do Processo P_i) é definido como o total número de páginas referenciadas no mais recente Δ (varia no tempo)
- Exemplo com Δ = 10



Modelo de Conjunto de Trabalho (Cont.)

UFABC

- WSS_i - tenta aproximar o tamanho da localidade de processo P_i
  - se Δ muito pequeno não abrangerá localidade inteira
  - se Δ muito grande abrangerá várias localidades
  - se $\Delta = \infty \implies$ abrangerá todo o programa
- $D = \Sigma\ WSS_i \equiv$ aproximar o total de frames de demanda
  - Aproximação de TODAS as localidades
- m = número total de frames.
- se D> m $\implies$ Thrashing
- Política: se D > m, então suspender ou trocar um dos processos

Mantendo o controle do conjunto de trabalho

UFABC

- Manter as informações exatas sobre cada conjunto de trabalho é impraticável já que a janela do conjunto de trabalho é uma janela móvel.
- Aproximado com temporizador de intervalo + um bit de referência

Mantendo o controle do conjunto de trabalho

UFABC

- Exemplo: Δ = 10.000
  - Temporizador interrompe a cada 5000 unidades de tempo
  - Mantenha na memória 2 bits para cada página.
  - Veja o bit de referência e os bits de memória de 2 entradas como um registrador (3-bits), com o bit de referência sendo o bit mais significativo
  - Sempre que um temporizador interromper, faça um shift para a direita de um lugar nos 3 bits e defina os valores de todos os bits de referência para 0.
  - Se ocorrer uma falha de página, podemos examinar os 3 bits para determinar se uma página foi usada nos últimos 10.000 a 15.000 referências (pelo menos um bit está ativado). Em caso afirmativo, a página está no conjunto de trabalho

Mantendo o controle do conjunto de trabalho

UFABC

- Por que isso não é completamente preciso?
- Melhoria = 10 bits e interromper a cada 1000 unidades de tempo

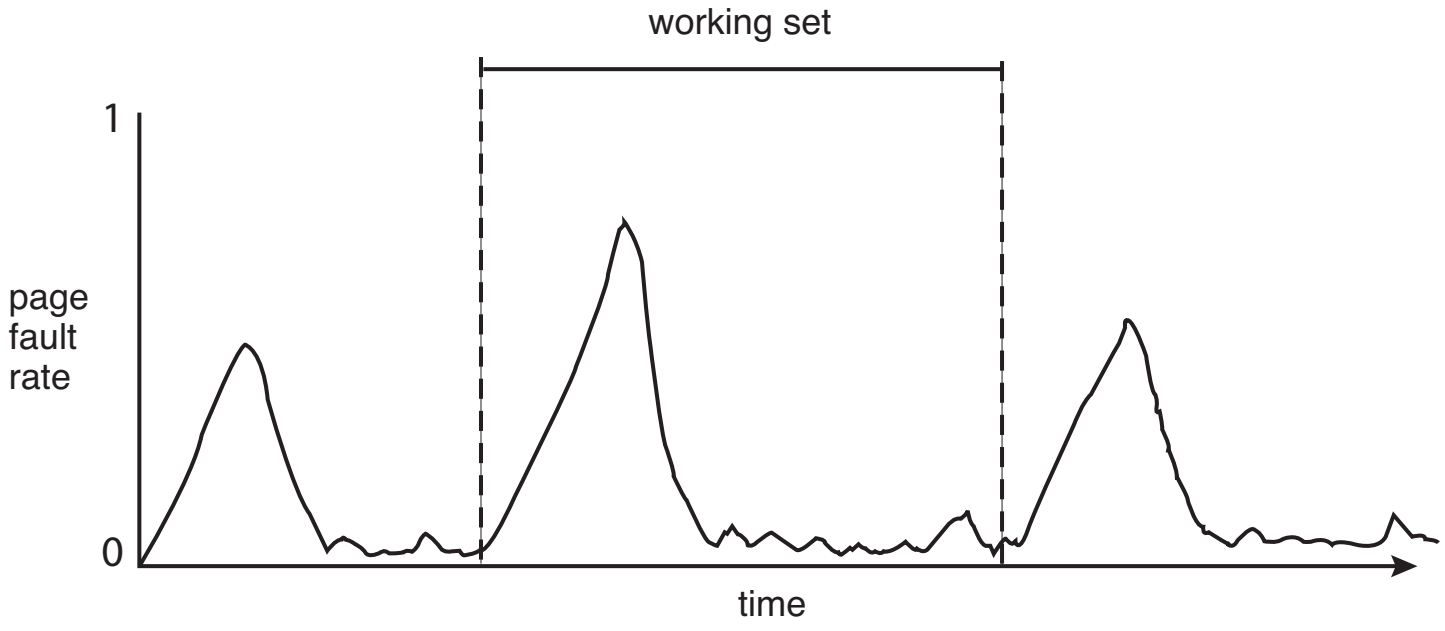
Frequência de falha de página

UFABC

- Abordagem mais direta que o WSS
- Estabelecer taxa de frequência de falha de página (PFF) "aceitável" e usar a política de substituição local
  - Se a taxa real for muito baixa, o processo perde o frame
  - Se a taxa real for muito alta, processo ganha o frame

Conjuntos de trabalho e taxas de falha de página

- Relacionamento direto entre o conjunto de trabalho de um processo e sua taxa de falha de página
- Alterações no conjunto de trabalho ao longo do tempo
- Picos e vales ao longo do tempo



Outras considerações

UFABC

- Prepaging
- Tamanho da página
- Alcance de TLB
- Tabelas de páginas invertidas
- Estrutura do programa
- Inter Bloqueio de E/S e bloqueio de página

Outras considerações - Prepaging

UFABC

- Prepaging usado para reduzir o grande número de falhas de página que ocorre na inicialização do processo
- Prepagina todas ou algumas das páginas que um processo precisará, antes que elas sejam referenciadas
- Mas se as páginas pré-paginadas não são usadas, a E/S e a memória foram desperdiçadas
- Suponha que: As páginas são pré-paginadas e
  - Uma fração α destas páginas é realmente usada (α entre 0 e e 1)
  - Pergunta - O custo de falhas de páginas economizado s $*$ é maior menor que o custo de pré-paginação de páginas desnecessárias s $*$ $(1 -)$?
    - Se α próximo de zero $\Longrightarrow$ prepaging perde
    - Se α perto de um $\Longrightarrow$ prepaging ganha

Outros problemas - tamanho da página

UFABC

- Às vezes, os projetistas de sistemas operacionais têm uma escolha
  - Especialmente se estiver executando em CPU customizada
- A seleção do tamanho da página deve levar em consideração:
  - Fragmentação
  - Tamanho da tabela de páginas
  - Resolução
  - Sobrecarga de E/S
  - Número de falhas de página
  - Localidade
  - Tamanho e efetividade do TLB
- Sempre potência de 2, geralmente na faixa de 2^{12} (4.096 bytes) a 2^{22} (4.194.304 bytes)
  - Em média, crescendo com o tempo

Outros problemas - Alcance do TLB

UFABC

- Alcance de TLB - A quantidade de memória acessível a partir do TLB
- Alcance de TLB = (Tamanho de TLB) X (Tamanho da Página)
- Idealmente, o conjunto de trabalho de cada processo é armazenado no TLB
  - Caso contrário, há um alto grau de falhas de página
- Aumentar o tamanho da página
  - Isso pode levar a um aumento na fragmentação, pois nem todos aplicativos exigem um tamanho de página grande
- Fornecer vários tamanhos de página
  - Isso permite que aplicativos que exigem tamanhos de página maiores tenham oportunidade de usá-los sem um aumento na fragmentação

Outras questões - estrutura do programa

UFABC

- Estrutura do programa
  - ~int [128,128] data;
  - Cada linha é armazenada em uma página
  - Programa 1

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

    128 x 128 = 16.384 falhas de página

  - Programa 2

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

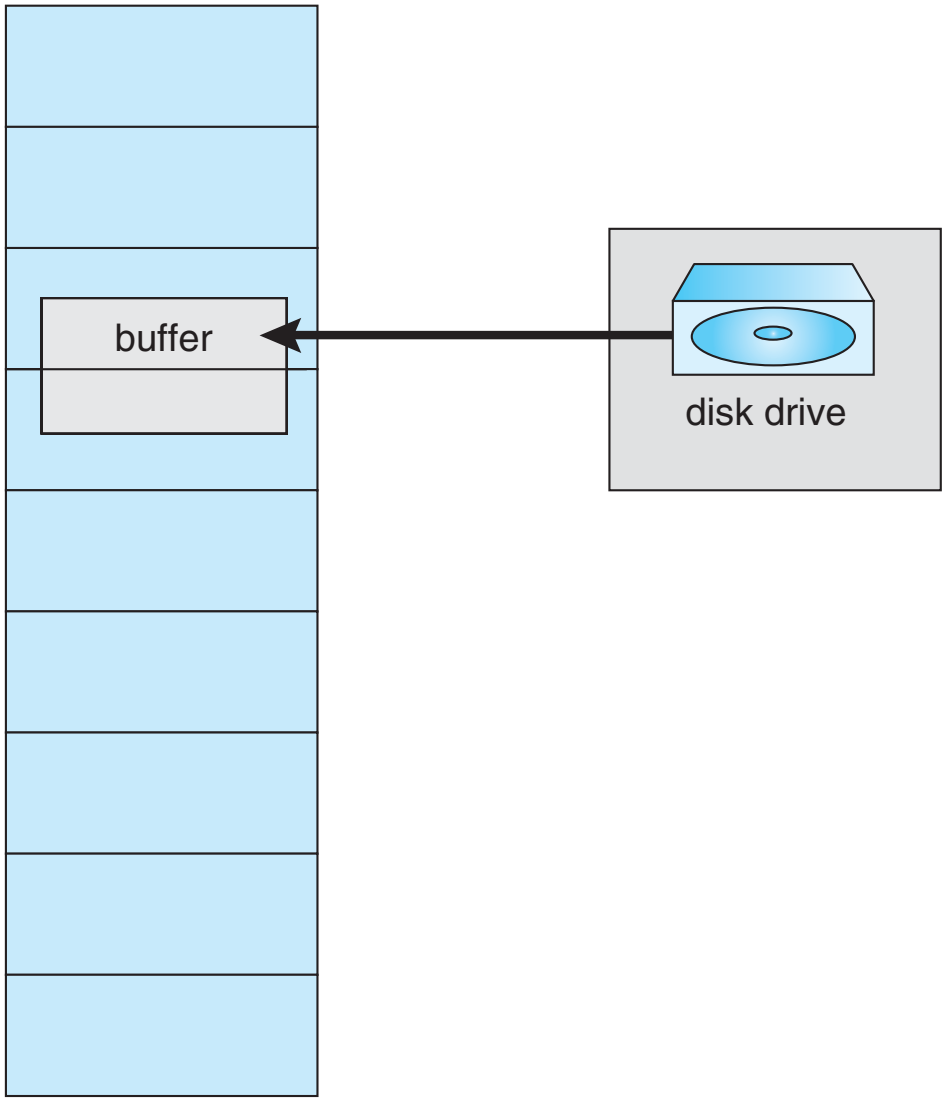
    128 falhas de página

Outros problemas - Intertravamento de E/S

UFABC

- Inter Intertravamento E/S - As páginas devem às vezes, ficar fixas na memória
- Considere E/S - Páginas que são usadas para copiar um arquivo de um dispositivo deve ser bloqueado de ser selecionado para substituição pelo algoritmo de substituição de páginas
- Fixação de páginas (pinning) para travar páginas na memória

Outros problemas - Intertravamento de E/S



Exemplos de sistema operacional

UFABC

- Windows
- Solaris

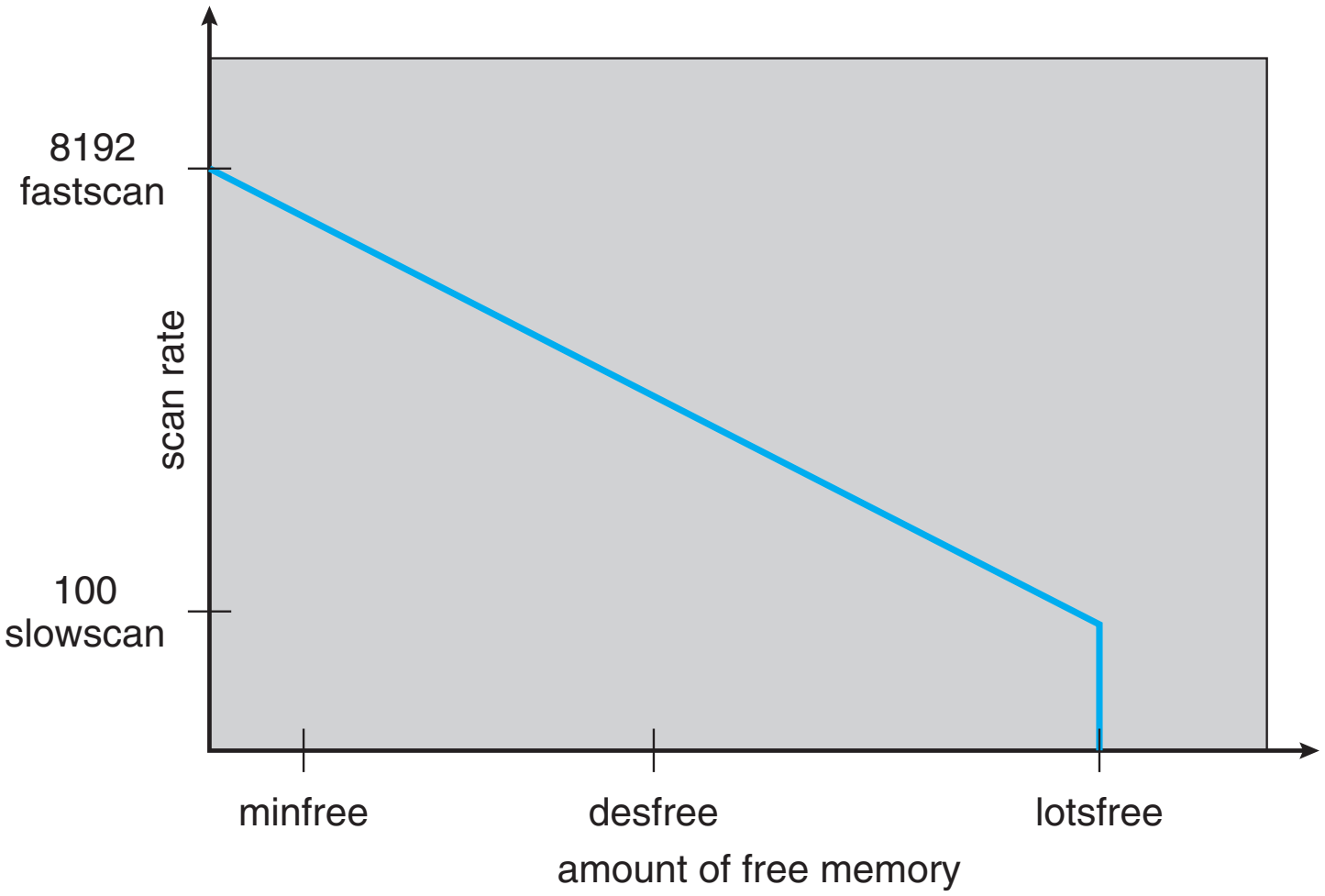
Windows

UFABC

- Usa a paginação de demanda com clustering. Clustering traz páginas próximas à página com falhas
- Processos são designados ao um conjunto de trabalho mínimo e a um conjunto de trabalho máximo
- O conjunto mínimo de trabalho é o número mínimo de páginas que um processo tem como garantia de que estarão na memória
- Um processo pode ter atribuído a ele a um número de páginas até o seu conjunto de trabalho máximo
- Quando a quantidade de memória livre no sistema cai abaixo de um limiar, um ajuste automático do conjunto de trabalho é executado para restaurar a quantidade de memória livre
  - O ajuste do conjunto de trabalho remove páginas de processos que excedem o tamanho do conjunto de trabalho mínimo

Solaris

UFABC

- Mantém uma lista de páginas livres para receber processos com falhas de página
- Lotsfree - parâmetro de limiar (quantidade de memória livre) para começar paginação
- Desfree - parâmetro de limite para aumentar a paginação
- Minfree - parâmetro limiar para começar a paginação
- A paginação é executada pelo processo que está sendo retirado da memória
- Pageout varre páginas usando o algoritmo de relógio modificado
- Scanrate é a taxa em que as páginas são varridas. Isso varia de slowscan a fastscan
- Pageout é chamado com mais freqüência, dependendo da quantidade de memória livre disponível
- A paginação prioritária dá prioridade às páginas de código

Solaris 2 Page Scanner