Gabriela Paola Sereniski - 2349345
João Victor Salvi Silva - 2304350

Simulador para Memória Virtual e Algoritmos de Substituição de Páginas

Relatório técnico do projeto solicitado pelo professor Rodrigo Campiolo na disciplina de Sistemas Operacionais de Bacharelado em Ciência da Computação da Universidade Tecnológica Federal do Paraná.

Universidade Tecnológica Federal do Paraná – UTFPR

Departamento Acadêmico de Computação – DACOM

Bacharelado em Ciência da Computação – BCC

Campo Mourão

Julho / 2023

Resumo

Este relatório descreve o desenvolvimento de um simulador para algoritmos de substituição de páginas em sistemas com memória virtual de paginação pura. Foram implementados três algoritmos: FIFO, LRU e NRU. O objetivo do projeto é analisar o comportamento do sistema ao gerenciar a RAM com base em uma sequência de acessos à memória. A metodologia utilizada envolveu a implementação da memória virtual e das estruturas usadas por cada algoritmo. Os algoritmos implementados são descritos detalhadamente. Cada algoritmo apresenta vantagens e limitações. O projeto foi útil para compreender o funcionamento da memória virtual e das estratégias de gerenciamento de memória.

Palavras-chave: algoritmos de substituição. simulador. sistemas operacionais. paginação de memória.

Sumário

1 Introdução

A eficiência do gerenciamento de memória é fundamental para o bom funcionamento de um sistema operacional (SO). Um dos principais desafios de seu desenvolvimento é a otimização da utilização da memória primária.

Neste contexto, algoritmos de substituição de páginas desempenham um papel crucial para garantir o melhor desempenho possível em sistemas que utilizam memória virtual. Esses algoritmos são responsáveis por decidir quais páginas devem ser mantidas na memória primária e quais devem ser removidas para dar espaço a novas páginas. Dessa forma, o desempenho do sistema depende diretamente da eficiência destes algoritmos.

Este relatório tem como objetivo apresentar os métodos empregados no desenvolvimento de um simulador para memória virtual e seus algoritmos de substituição de páginas, detalhando seu funcionamento, suas estruturas de controle e o comportamento dos algoritmos *first in first out* (FIFO), *least recently used* (LRU) e *not recently used* (NRU).

A memória virtual por paginação pura é um método em que a memória primária é dividida em blocos fixos chamados de páginas, e o espaço de endereçamento do processo é dividido em blocos correspondentes chamados de quadros de página (*frames*). Quando um processo precisa acessar uma página que não está presente na memória primária, ocorre uma falta de página e o sistema operacional precisa tomar uma decisão sobre qual página deve ser substituída.

Ao implementar e comparar diferentes algoritmos de substituição de páginas, poderemos avaliar seu desempenho e identificar aquele que melhor se adéqua às necessidades do sistema.

2 Descrição da atividade

Neste projeto, o objetivo foi desenvolver um simulador para algoritmos de substituição de páginas em um sistema com memória virtual de paginação pura. Foram implementadas três políticas de substituição de páginas, sendo elas a FIFO, LRU e NRU. A partir da implementação, é possível analisar qual seria o desempenho de um sistema ao gerir a RAM, dado uma sequência de acessos à memória. A atividade requereu conhecimentos em programação, Sistemas Operacionais e suas técnicas de gerenciamento de memória.

3 Metodologia

Nesta seção, apresenta-se detalhes sobre a implementação do simulador, e as estruturas empregadas para tal.

3.1 *A Memória Virtual*

Quando um processo precisa acessar um endereço da memória, o gerenciador deve realizar uma série de verificações para garantir a segurança, validade e eficiência desse acesso. O processador de um computador, em sua maioria, não realiza acessos diretamente à memória física, mas sim, gera endereços para um espaço virtual, que é gerenciado pela Unidade de Gerenciamento de Memória, ou do inglês *Memory Management Unit* (MMU) em conjunto com o núcleo do SO.

A memória virtual pode ser organizada de diversas maneiras, como por partições, segmentos, ou como é o caso da implementada neste trabalho, paginação. A paginação pode ser utilizada na RAM, em disco, ou até mesmo em ambos. O *software* desenvolvido para este projeto simula a paginação entre a RAM e o disco a partir de uma tabela de páginas, que armazena informações sobre os endereços utilizados por um processo.

Quando um processo tenta acessar uma página da tabela, verifica-se se ela está ou não presente. Caso ela esteja, o processo pode acessar esse endereço imediatamente, sem necessidade de carregá-lo do disco. Caso não esteja, acontece o que é conhecido como falta de página, termo que indica que um quadro do disco deverá ser carregado para a memória física e o processo deve ser suspenso. Nota-se que a suspensão de um processo é prejudicial ao seu desempenho, visto que carregar instruções do disco é muito lento se comparado ao tempo de acesso à RAM, e por isso a implantação de técnicas que diminuam ao máximo o número de ocorrência de faltas de página é tão importante.

Quando há falta de página, verifica-se se há espaço disponível na RAM. Se houver, o processo é carregado, caso contrário, aplica-se uma política que escolhe qual página presente na RAM deverá ser substituída. Todo o processo descrito pode ser visualizado com mais facilidade no fluxograma da Figura 1.

A memória RAM do simulador desenvolvido é representada por um vetor, assim como o disco. Todo o espaço de endereçamento virtual do processo fica armazenado no disco, a partir de seu primeiro endereço. A decisão de inicializar os dados no disco dessa maneira, além da facilidade de implementação, foi baseada no comportamento do simulador Amnesia (LASDPC, 2012), que carrega o que é possível na RAM, e o restante sequencialmente no disco. Como desejava-se contabilizar as faltas de página que ocorrem quando um processo é inicializado, todo ele é carregado no disco.

A estrutura das entradas da tabela de página consiste em um ponteiro para um

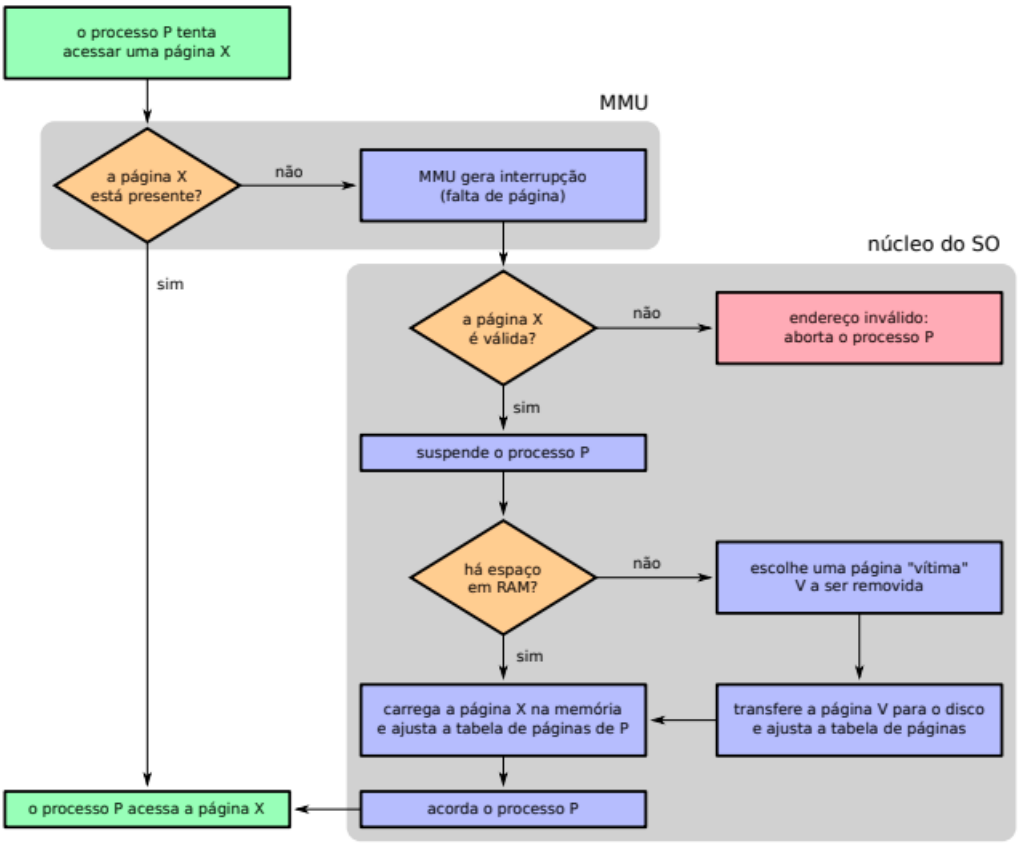

Figura 1 – Ações do mecanismo de paginação em disco. (MAZIERO, 2020)

quadro da memória física, e mais quatro campos de controle:

- `V`: serve para verificar se um campo é válido ou não, ou seja, se ele está carregado em memória ou deverá ser buscado no disco.
- `R`: é ativo sempre que uma página é acessada. Por decisão de projeto, este campo é desativado caso o processo tenha sido referenciado há muito tempo. O tempo de ativação deste campo depende do espaço na RAM - caso tenham sido processados mais instruções que o número de páginas da memória física, ele é desativado.
- `M`: é ativo quando há operação de escrita no endereço.
- `age`: é usado para controle dos algoritmos LRU e NRU, que serão descritos a seguir.

Como mencionado, o processador não realiza acesso direto à memória, mas sim, acessa endereços virtuais que precisam ser convertidos pela MMU para o endereço físico correspondente. Usualmente, os primeiros *bits* do endereço virtual representam o número da entrada da tabela de páginas e os últimos o deslocamento dentro dessa página, uma vez que as páginas e os quadros da RAM tem o mesmo tamanho. Como a entrada da tabela armazena qual o quadro da RAM em que o endereço está armazenado, é possível realizar

operações *bitwise* para adquirir a página correta, e concatenar os bits de deslocamento obtidos do endereço virtual com os bits que representam o quadro armazenado pela página.

A fim de simplificar a implementação dessa conversão de endereços, optou-se por realizar cálculos de divisão e resto para obter os valores desses bits. O índice da página é obtido a partir da divisão do endereço lógico pelo tamanho da página, O deslocamento é o resultado da operação de resto entre o endereço lógico e o tamanho da página, e o endereço físico é obtido pela multiplicação do valor do *frame* encontrado na entrada da tabela de páginas pelo tamanho da página, mais o deslocamento. Quando ao endereçamento no disco, optou-se por definir que os blocos do disco tem o mesmo tamanho que os da RAM. Como o processo é carregado a partir do início do disco, pode-se obter o endereço do início de um quadro no disco como o próprio endereço lógico, menos o resto da divisão entre o endereço lógico e o tamanho da página.

3.2 *Algoritmo FIFO*

O algoritmo FIFO (First-In, First-Out) é um dos algoritmos mais simples de substituição de páginas. Sua ideia é substituir a página que entrou primeiro na memória, ou seja, a página mais antiga em termos de tempo de chegada.

Para implementação dessa política, cada página que entra na memória é adicionada ao final de uma fila. Quando ocorre uma falta de página e é necessário substituir uma página, a página no início da fila é escolhida para ser removida.

Uma das principais vantagens do algoritmo FIFO é a sua simplicidade de implementação. Não é necessário realizar cálculos complexos ou manter informações adicionais sobre o histórico de uso das páginas. Além disso, é fácil de entender e não requer muitos recursos computacionais.

Porém, o algoritmo FIFO apresenta algumas limitações. Um dos principais problemas é conhecido como "anomalia de Belady", que ocorre quando aumentar o número de quadros de página resulta em um aumento da taxa de falta de página. Isso significa que, em certos casos, o desempenho do algoritmo pode piorar à medida que a quantidade de memória disponível aumenta. Essa limitação ocorre porque o algoritmo FIFO não considera o padrão de acesso às páginas. Mesmo que uma página seja frequentemente usada, ela pode ser removida pelo algoritmo FIFO se tiver entrado na memória antes de outras páginas. Isso leva a um maior número de faltas de página e a um desempenho inferior em comparação com algoritmos mais sofisticados, como o LRU.

Apesar das limitações, o algoritmo FIFO ainda é usado em alguns sistemas, especialmente em situações em que a simplicidade é mais valorizada do que a otimização de desempenho. É um bom ponto de partida para entender os conceitos básicos de substituição de páginas e pode ser útil em certos cenários com requisitos específicos.

3.3 Algoritmo LRU

O algoritmo LRU é um dos algoritmos mais populares e utilizados para substituição de páginas em sistemas com memória virtual. Sua ideia é substituir a página que foi menos recentemente usada, ou seja, a página que não foi acessada por mais tempo.

O LRU baseia-se na suposição de que as páginas que foram acessadas recentemente têm maior probabilidade de serem acessadas novamente no futuro próximo. Portanto, ao substituir a página menos recentemente usada, o algoritmo tenta maximizar o aproveitamento da memória primária e minimizar a taxa de faltas de página.

Para implementar esse comportamento, foi feito uso do campo `age` mantido na tabela de páginas. Sempre que um endereço é carregado, sua idade é atualizada para zero. A cada instrução do processo, essa idade é incrementada. Quando uma substituição é necessária, percorre-se a tabela de páginas verificando qual, dentre as entradas válidas, tem a maior idade. Essa página é escolhida para ser removida da RAM.

Uma das principais vantagens do algoritmo LRU é sua capacidade de aproveitar padrões de acesso às páginas. Ele tende a manter na memória as páginas que são frequentemente usadas, reduzindo assim o número de faltas de página e melhorando o desempenho do sistema como um todo. Além disso, o LRU não sofre da anomalia de Belady, ou seja, ao aumentar a quantidade de memória disponível, a taxa de faltas de página tende a diminuir.

No entanto, o algoritmo LRU também apresenta algumas desvantagens. Uma delas é a complexidade de implementação, se comparada ao FIFO, e a necessidade de manter, de alguma forma, o registro do histórico de acesso às páginas, o que pode exigir recursos adicionais. Além disso, em sistemas com muitas páginas, a manutenção do registro de histórico pode causar gastos grandes em termos de espaço e aumentar o tempo de execução.

3.4 Algoritmo NRU

O algoritmo NRU é outro algoritmo utilizado para substituição de páginas. Sua abordagem é baseada em uma classificação das páginas em categorias, levando em consideração se elas foram recentemente usadas e se foram modificadas.

O NRU divide as páginas em quatro classes, como descritas por (MAZIERO, 2020):

- Classe 0 (*Not Recently Used, Not Modified*): páginas que não foram referenciadas recentemente e cujo conteúdo não foi modificado. São as melhores candidatas à substituição, pois podem ser simplesmente retiradas da memória.
- Classe 1 (*Not Recently Used, Modified*): páginas que não foram referenciadas recentemente, mas cujo conteúdo já foi modificado. Não são escolhas tão boas, porque

terão de ser gravadas no disco antes de serem substituídas.

- Classe 2 (*Recently Used, Not Modified*): páginas referenciadas recentemente, cujo conteúdo permanece inalterado. São provavelmente páginas de código que estão sendo usadas ativamente e serão referenciadas novamente em breve.
- Classe 3 (*Recently Used, Modified*): páginas referenciadas recentemente e cujo conteúdo foi modificado. São a pior escolha, porque terão de ser gravadas na área de troca e provavelmente serão necessárias em breve.

No simulador, quando uma substituição é necessária, é realizado um escaneamento da tabela de páginas, verificando-se os campos `R` e `M`, para encontrar qual página deverá ser substituída. A página escolhida será a que pertencer à classe com o menor valor possível. Ou seja, a página da classe 0 será escolhida primeiro, seguida pela classe 1, classe 2 e, por fim, classe 3.

Com o intuito de evitar o fenômeno conhecido como "*aging*", em que páginas que não são recentemente usadas e não foram modificadas permanecem na memória por um longo período de tempo, ocupando espaço desnecessariamente, verifica-se também a idade do processo. Assim, caso haja processos de mesma categoria, o que fora acessado há mais tempo dentre eles será escolhido para ser substituído.

Uma das vantagens do algoritmo NRU é sua simplicidade de implementação, se comparado a outros algoritmos mais complexos. No entanto, o algoritmo NRU também possui algumas limitações. Uma delas é a necessidade de realizar o escaneamento da tabela de páginas, o que pode ser custoso.

4 Conclusão

O desenvolvimento deste projeto fora um exercício valioso para esclarecer o funcionamento da memória virtual e de diferentes estratégias de gerência de memória.

Cada algoritmo implementado possui suas características e critérios de escolha de páginas para substituição. O FIFO é simples de implementar, mas pode resultar na anomalia de Belady. O LRU considera o padrão de acesso às páginas, mantendo as páginas mais recentemente usadas na memória. O NRU, por sua vez, classifica as páginas em categorias com base em seu uso recente e *status* de modificação.

É importante ressaltar que existem outros algoritmos de substituição de páginas disponíveis, cada um com suas vantagens e desvantagens. A escolha do algoritmo mais apropriado dependerá da análise cuidadosa das características do sistema e do equilíbrio entre desempenho e complexidade de implementação.

Por fim, a capacidade de visualizar o comportamento das estruturas de memória proporciona facilidade de compreensão do conteúdo de gerência de memória, que pode ser um tanto abstrato para um aluno em seu primeiro contato com o assunto. A implementação deste projeto auxiliou não só no entendimento dos algoritmos de substituição em si, mas também em questões como o espaço de endereçamento de um processo, e tudo que envolve os mecanismos de paginação, desde o *hardware* disponível à lógica utilizada para geri-lo.

5 Referências

LASDPC. *User Manual for glossaries.sty*. [S.l.], 2012. Disponível em: <http://amnesia-.lasdpc.icmc.usp.br/>http://amnesia.lasdpc.icmc.usp.br/. Acesso em: 11.3.2013. Citado na página 5.

MAZIERO, C. *Sistemas Operacionais: Conceitos e Mecanismos*. [S.l.: s.n.], 2020. 456 p. ISBN 978-85-7335-340-2. Citado 2 vezes nas páginas 6 e 8.