Hendrick Felipe Scheifer

Memória Virtual

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

Universidade Tecnológica Federal do Paraná – UTFPR

Departa mento Acadêmico de Computação – DACOM

Bacharelado em Ciência da Computação – BCC

Campo Mourão

Dezembro / 2024

Resumo

Os objetivos deste trabalho envolvem a compreensão de conceitos de memória virtual, paginação e algoritmos de substituição a partir da configuração e análise de diferentes arquiteturas e execuções em memória virtual. Os procedimentos se deram por meio do uso do simulador Amnesia para simular, analisar e comparar arquiteturas que utilizam diferentes algoritmos de substituição aplicando diferentes execuções. Os resultados foram analisados e os diferentes algoritmos foram discutidos. Esta atividade proporcionou uma melhor compreensão dos conceitos envolvidos na utilização de memória virtual.

Palavras-chave: memória virtual; sistemas operacionais; simulador.

Sumário

1 Introdução

A memória RAM (*Random Access Memory*) é um recurso escasso na computação, devido ao seu alto custo e consume de energia. Porém, para armazenar quantidades maiores de informações, os computadores possuem um disco de armazenamento, que são maiores e mais baratos, mas, em contrapartida, são mais lentos (MAZIERO, 2019).

Devido ao grande espaço que pode ser ocupado por processos, estes não podem ser totalmente carregados na memória principal da máquina antes de sua execução. Para executar processos que não estão completamente na memória principal, utiliza-se a técnica da memória virtual. A memória virtual consiste na abstração da memória principal em um *array* de armazenamento muito maior, separando a memória lógica da memória física (SILBERSCHATZ, 2015).

No estudo da memória virtual é importante compreender a paginação por demanda, que garante que as páginas serão carregadas conforme a necessidade dos processos em execução. Como a memória ainda é limitada, deve-se carregar para a memória principal apenas os quadros essenciais (SILBERSCHATZ, 2015). Caso haja a falta de uma página específica, deve-se carregá-la, mas é importante notar que se os quadros estiverem cheios, há a necessidade de substituição, mas, qual página substituir? Existem diferentes abordagens para a política de substituição, algumas delas serão explicadas nas seções posteriores deste trabalho.

2 Objetivos

Os objetivos deste trabalho são compreender os conceitos de memória virtual, paginação e TLB (*Translation Lookaside Buffers*), configurar e analisar diferentes arquiteturas e execuções em memória virtual usando simulador, além de compreender e comparar políticas de substituição de páginas.

3 Fundamentação

Devido ao tamanho elevado que processos, impossibilita-se carregá-los por completo para a memória principal. Como processos são divididos em quadros, mapeados por páginas, é possível manter em memória apenas as páginas essenciais para a execução do processo, e é neste procedimento que se baseia a memória virtual.

Os quadros da memória representam páginas do processo, e o número de quadros disponíveis é limitado, logo, nem todas as páginas do processo estão carregadas na memória, quando se referencia uma página que não está presente na memória acontece uma *page-fault* (falta de página). Uma falta de página gera uma exceção no sistema operacio-

nal, portanto o sistema operacional irá tratar de forma síncrona, bloqueando o processo, e iniciando a operação de E/S (Entrada/Saída), e quando a página estiver carregada o processo voltará a ser executado.

Toda vez que uma página precisa ser carregada e todos os quadros estão ocupados, é necessário selecionar uma página vítima para ser substituída. Para evitar uma quantidade excessiva de falta de páginas, busca-se explorar diferentes técnicas para esta seleção de página vítima, pois, se substituir uma página comumente referenciada, ela pode ser referenciada logo em seguida, gerando uma nova falta de páginas. Estas técnicas são conhecidas como algoritmos de substituição, e serão melhor explorados nas seções posteriores.

4 Materiais

Simulador Amnesia (http://amnesia.lasdpc.icmc.usp.br/).

5 Procedimentos e Resultados

Nesta seção, serão abordados diferentes algoritmos para a política de substituição de páginas, para cada um dos algoritmos apresentados, será detalhado seu critério para seleção de página vítima e serão executados testes no simulador proposto. Com os testes, os resultados obtidos em tempo, frequência de falta de página e desempenho serão analisados.

Para a apresentação do resultado das simulações serão apresentadas figuras contendo tabelas que demonstram o estado dos páginas na memória. Esta tabela apresenta o passo de execução atual, o endereço recebido neste passo, a página referenciada e o o endereço de cada página na memória, caso a página não esteja na memória será indicado por um x, vale ressaltar que o endereço na memória será apresentado em números decimais para uma melhor visualização.

A simulação é realizada no simulador Amnesia, proposto pela atividade. Para cada algoritmo de substituição apresentado, carregamos uma arquitetura de memória, que contém as informações de qual algoritmo será utilizado, além da quantidade de quadros na memória e tamanhos de cada quadro. Para o teste utilizando a arquitetura são utilizados "*traces*", que são sequências de referências de acesso à memória contendo um indicador de operação a ser realizada e o endereço lógico desejado.

Para cada acesso realizado, uma cor é atribuída ao acesso na tabela, caso a célula seja marcada com verde, significa que a página foi encontrada na memória, caso seja marcada com vermelho, significa que houve uma falta de páginas e foi necessária uma substituição.

5.1 Algoritmo de substituição FIFO

O algoritmo FIFO (*first in, first out*) é bem simples de compreender. As páginas são substituídas na ordem que foram carregadas, ou seja, caso haja uma falta de página, a página carregada primeiro na lista de páginas atual será substituída, como o próprio nome do algoritmo diz: primeira a entrar, primeira a sair. É importante ressaltar que a página mais antiga não é necessariamente inútil, pois ela pode ser referenciada logo em seguida gerando uma nova falta de páginas, por isso, o algoritmo FIFO em sua forma mais pura não é muito utilizado (TANENBAUM, 2016)

Para este algoritmo, foram simulados dois *traces* diferentes, os resultados de acesso do primeiro *trace* podem ser vistos na figura 1.

Passo	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Endereço (Hex.)	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	10	11	12	13
Página	0	0	0	0	1	1	1	1	2	2	2	2	3	3	3	3	4	4	4	4
0	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
1	0	0	0	0	0	0	0	0	x	x	x	x	x	x	x	x	x	x	x	x
2	x	x	x	x	x	x	x	x	0	0	0	0	0	0	0	0	0	0	0	0
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figura 1 – Tabela da simulação do primeiro *trace* com FIFO.

É possível visualizar que houve apenas uma falta de páginas, no passo 9, com esta falta de página foi necessário realizar uma substituição. Conforme o algoritmo FIFO determina, a página vítima deve ser a página carregada primeiro, que era a página 1, podemos observar que ela estava no endereço 0, o primeiro a ser carregado na inicialização da execução.

Na execução do segundo *trace*, foram obtidos os acessos representados na figura 2.

Passo	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Endereço (Hex.)	b	a	0	1	c	d	10	11	7	6	0	1	9	8	4	5	6	9	c	d
Página	2	2	0	0	3	3	4	4	1	1	0	0	2	2	1	1	1	2	3	3
0	0	0	0	0	0	0	0	0	x	x	1	1	1	1	1	1	1	1	1	1
1	x	x	x	x	x	x	x	x	0	0	0	0	0	0	0	0	0	0	0	0
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	x	x
3	1	1	1	1	1	1	1	1	1	1	x	x	x	x	x	x	x	x	2	2
4	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3

Figura 2 – Tabela da simulação do segundo *trace* com FIFO.

Diferente do primeiro *trace*, este segundo não possui um acesso sequencial, tornando próximo acesso sempre menos previsível. É possível visualizar que ocorreram três faltas de páginas, no passo 9, a página 1 não estava carregada na memória, logo, foi reali-

zada uma substituição, como o algoritmo FIFO seleciona a página mais antiga, a vítima foi a página 0. O que não foi a melhor opção, pois dois passos depois, no passo 11, a página 0, que acabou de ser substituída, foi referenciada e ocorreu uma nova falta de páginas, sendo necessária uma substituição, a vítima selecionada foi a página 3, que havia sido carregada há mais tempo. A terceira falta de páginas ocorre no passo 19, onde a página 3 é referenciada e precisa ser carregada, e havendo uma substituição, a página vítima foi a página 2, que era atualmente a mais antiga carregada.

Comparando a execução dos dois *traces* pode-se notar que o algoritmo FIFO pode ter um desempenho menos, obtendo menos falta de páginas, em acessos sequenciais a memória. Mas para o acesso de páginas mais aleatório, o próximo acesso se torna menos previsível e difícil de garantir que a página mais antiga é a melhor vítima, pois, em determinadas situações, ela ainda pode ser útil e ser referenciada instantes após a substituição, gerando mais uma falta de páginas.

5.2 Algoritmo de substituição NRU

A memória virtual atribui às páginas carregadas dois bits indicadores muito relevantes, o bit R para indicar se a página foi referenciada, ou seja, se ela foi necessária, e o bit M para indicar se ela foi modificada durante a execução. O algoritmo NRU (*Not Recently Used*) se baseia nestes bits para realizar sua escolha de vítima, para a análise, são observadas quatro possíveis classes:

- 0: R = 0 e M = 0, não referenciada e não modificada;
- 1: R = 0 e M = 1, não referenciada e modificada;
- 2: R = 1 e M = 0, referenciada e não modificada;
- 3: R = 1 e M = 1, referenciada e modificada;

O algoritmo NRU busca substituir uma página da classe mais baixa possível, note que é preferível manter uma página referencia do que uma alterada (TANENBAUM, 2016).

Para este algoritmo, foram simulados dois *traces* diferentes, os resultados de acesso do primeiro *trace* podem ser vistos na figura 3.

Passo	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Endereço (Hex.)	0	4	8	c	d	10	11	14	15	c	d	10	11	14	15	c	d	10	11	14	15	c	d	10	11	14	15	16	16	17
Página	0	1	2	3	3	4	4	5	5	3	3	4	4	5	5	3	3	4	4	5	5	3	3	4	4	5	5	5	5	5
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
3	x	x	x	3	3	x	x	x	x	3	3	x	x	x	x	3	3	x	x	x	x	3	3	x	x	x	x	x	x	x
4	x	x	x	x	x	3	3	x	x	x	x	3	3	x	x	x	x	3	3	x	x	x	x	3	3	x	x	x	x	x
5	3	3	3	x	x	x	x	3	3	x	x	x	x	3	3	x	x	x	x	3	3	x	x	x	x	3	3	3	3	3

Figura 3 – Tabela da simulação do primeiro *trace* com NRU.

Este *trace* simula um laço de repetição que acessa quatro vezes as páginas 3, 4 e 5. É possível observar que houve uma grande quantidade e falta de páginas, 12 no total, 3 a cada laço de repetição. As faltas de páginas acontecem ao tentar acessar as páginas referenciadas no laço de repetição, pois como as páginas 0, 1 e 2 haviam sido referenciadas e alteradas, pertenciam à classe 3 citada anteriormente, e não seriam substituídas, e embora as páginas do laço de repetição tivessem sido referenciadas, pertencendo à classe 2, ainda possuíam prioridade inferior às páginas da classe 3.

Na execução do segundo *trace*, foram obtidos os acessos representados na figura 4.

Passo	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Endereço (Hex.)	b	a	0	1	c	d	10	11	7	6	0	1	9	8	4	5	6	9	c	d
Página	2	2	0	0	3	3	4	4	1	1	0	0	2	2	1	1	1	2	3	3
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	x	x	x	x	x	x	x	x	3	3	3	3	3	3	3	3	3	3	3	3
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
4	3	3	3	3	3	3	3	3	x	x	x	x	x	x	x	x	x	x	x	x

Figura 4 – Tabela da simulação do segundo *trace* com NRU.

Este *trace* possui acesso mais aleatório, sem a simulação de repetições ou acessos sequenciais. Para este *trace* foram obtidas apenas uma falta de páginas, no passo 7, onde a página 1 foi referenciada e não estava carregada na memória, a página vítima foi a 4, que embora houvesse sido referenciada recentemente, não havia sido alterada, pertencendo a uma classe inferior as outras páginas.

Comparando a execução dos *traces* é possível observar que o algoritmo NRU não irá necessariamente ser eficiente em laços de repetição, pois embora as páginas tenham sido carregadas recentemente e referenciadas recentemente, caso ela não seja alterada, não irá possuir a maior classe e obter prioridade. Mas para acessos aleatórios, como no segundo *trace*, o resultado pode ser melhor que outros algoritmos, pois utiliza critérios que valorizam páginas referenciadas e modificadas recentemente.

5.3 Algoritmo de substituição LRU

O algoritmo LRU (Least Recently Use) visa remover a página que não foi referenciada a mais tempo. Este algoritmo embora simples de compreender, possui obstáculos de implementação, pois seria necessário manter uma lista com todas as páginas da memória e manter ordenada por ordem de acesso a todo momento. Esta implementação é mais viável com o uso de *hardware* especial (TANENBAUM, 2016).

Para este algoritmo, foi simulado o mesmo *trace* utilizado anteriormente, na subseção 5.2, o qual simula um laço de repetição e obteve uma grande quantidade de falta de páginas com o algoritmo NRU. Os acessos simulados com o algoritmo LRU podem ser vistos na figura 5.

Passo	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Endereço (Hex.)	0	4	8	c	d	10	11	14	15	c	d	10	11	14	15	c	d	10	11	14	15	c	d	10	11	14	15	16	16	17
Página	0	1	2	3	3	4	4	5	5	3	3	4	4	5	5	3	3	4	4	5	5	3	3	4	4	5	5	5	5	5
0	1	1	1	1	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
1	0	0	0	0	0	0	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
3	x	x	x	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
4	x	x	x	x	x	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
5	3	3	3	x	x	x	x	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figura 5 – Tabela da simulação do *trace* com LRU.

É possível notar que houve apenas três faltas de páginas, no passo 4, a página 3 não está carregada na memória, e há uma substituição da página 5 que havia sido utilizada há mais tempo (não havia sido referenciada ainda no *trace*), posteriormente, a página 4 não estava carregada e houve mais uma falta de páginas, ocasionando a substituição da página 0, que foi a página referenciada há mais tempo, no passo 8, a página 5 foi referenciada, mas havia sido substituída no passo 4, houve uma substituição e a página vítima foi a página 2, pelo critério citado anteriormente. Como todas as páginas a serem referenciadas no laço de repetição estavam carregadas, foi possível realizar todas as repetições e a finalização sem falta de páginas.

Comparando o resultado obtido na execução do *trace* com laço de repetição utilizando o algoritmo LRU (5) com o a execução do mesmo utilizando o algoritmo NRU (3), é possível notar que a eficiência do algoritmo LRU foi superior. Isto se deve ao fato de que em um laço de repetição, as páginas recentemente referenciadas tendem a ser referenciadas novamente, isto é o princípio da localidade temporal, então utilizar um algoritmo que preserva as páginas utilizadas mais recentemente, substituindo as páginas utilizadas menos recentemente, é a melhor escolha.

6 Discussão dos Resultados

Os resultados obtidos permitem visualizar a diferença na eficiência de cada algoritmo em determinadas situações. É possível dizer que para acessos sequenciais, o algoritmo FIFO proporciona um bom resultado, mas com a necessidade de acessar páginas de forma menos previsível, onde páginas mais antigas podem ser úteis a qualquer momento, seu desempenho decai, gerando uma quantidade maior de falta de páginas. O algoritmo NRU, embora tenha uma proposta fácil de compreender e um desempenho razoável em diversas condições, possui fatores limitantes para seu desempenho, pois, mesmo que uma página tenha sido acessada recentemente, caso ela não tenha sido modificada, pode perder prioridade para outras páginas e ser substituída, o que traz um baixo desempenho em situações como o laço de repetição simulado. O algoritmo LRU apresenta uma solução para a implementação rudimentar proposta pelo NRU, já que busca manter páginas que foram acessadas recentemente em memória, proporcionando um ótimo desempenho em laços de repetição, pois explora o princípio da localidade temporal.

7 Conclusões

O objetivo do trabalho de compreender os conceitos de memória virtual, paginação e TLB (*Translation Lookaside Buffers*), configurar e analisar diferentes arquiteturas e execuções em memória virtual usando simulador e compreender e comparar políticas de substituição de páginas foram parcialmente atingidos. Embora os conceitos de memória virtual, paginação e memória virtual tenham sido devidamente explorados no trabalho e a simulação de diferentes arquiteturas e execuções em memória virtual tenha sido realizada com sucesso, houve uma limitação temporal durante a realização da atividade proposta e não foi possível explorar e simular os conceitos relacionados à TLB. O trabalho foi realizado com poucas dificuldades, as quais envolviam a complexidade do simulador, que apresenta muitos dados e torna-se difícil se acostumar a localizar os dados desejados com precisão, tomando um grande tempo durante a realização da atividade. Apesar das dificuldades, foi possível compreender melhor os conceitos a partir da prática.

8 Referências

MAZIERO, C. A. *Sistemas Operacionais: Conceitos e Mecanismos*. [S.l.]: UFPR, 2019. Citado na página 4.

SILBERSCHATZ, A. *Fundamentos de Sistemas Operacionais*. [S.l.]: LTC, 2015. Citado na página 4.

TANENBAUM, H. B. A. S. *Sistemas Operacionais Modernos*. [S.l.]: Pearson, 2016. Citado 3 vezes nas páginas 6, 7 e 9.