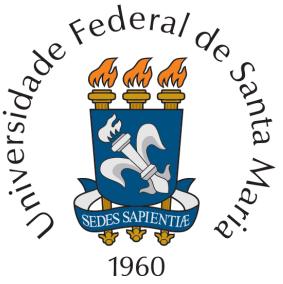
UNIVERSIDADE FEDERAL DE SANTA MARIA



RELATÓRIO TÉCNICO DE IMPLEMENTAÇÃO DE MEMÓRIA VIRTUAL

DISCIPLINA DE SISTEMAS OPERACIONAIS

Guilherme Meneghetti Einloft

Mathias Eckert Recktenvald

Leandro Oliveira Galbarino do Nascimento

SANTA MARIA- RS

2024

RESUMO

Com base nos conceitos apresentados na disciplina de Sistemas Operacionais, o trabalho foi essencial para consolidar os conhecimentos adquiridos. Assim como no Trabalho 1, que abordou o funcionamento dos processos e a implementação de escalonadores para permitir a execução simultânea de programas. No trabalho 2 exigiu a aplicação desses conhecimentos, combinados com os conceitos de memória virtual, que foram o foco principal desta etapa.

OBJETIVOS

- Aplicar os conceitos de memória virtual na construção de um sistema operacional mais robusto e eficiente.

Realizar testes para analisar a eficiência de memória virtual no sistema.

SUMÁRIO

INTRODUÇÃO

A memória virtual é um conceito fundamental em sistemas operacionais modernos, permitindo que programas utilizem mais memória do que a fisicamente disponível no computador. Ela cria uma abstração, onde o sistema operacional e o hardware trabalham juntos para simular uma quantidade maior de memória, utilizando o disco rígido (ou SSD) como uma extensão da memória RAM. A principal vantagem da memória virtual é que ela permite que programas e processos sejam executados sem se preocupar com os limites de memória física

1 MATERIAIS E MÉTODOS

Neste experimento, não foram utilizados hardwares físicos. Em vez de utilizar hardware físico, foram empregados códigos em C e Assembly que simulam o funcionamento de componentes de hardware, como processador e memória, porém com uma arquitetura simplificada, adequada para aprendizado e experimentação. A implementação do sistema operacional (SO) foi feita majoritariamente na linguagem de programação C.

1.1 Materiais Utilizados

- Ambiente de Desenvolvimento: Sistema Linux e Compilador GCC
- Simulador de Hardware: conjunto de funções em C e Assembly que imitam componentes como processador, memória e dispositivos de entrada/saída.
- Ferramentas de Análise: console para logs e métricas do SO, permitindo a depuração e avaliação do desempenho.

1.2 Métodos

1.2.1 Memória Virtual

1. Endereçamento Virtual e Físico:

Cada processo tem seu próprio espaço de endereçamento virtual, que é uma sequência de endereços de memória que ele pode acessar. O sistema operacional, por meio de um componente chamado *unidade de gerenciamento de memória* (MMU, *Memory Management Unit*), mapeia esses endereços virtuais para endereços físicos reais na memória RAM. Quando um processo tenta acessar um endereço virtual, a MMU converte esse endereço para um endereço físico correspondente.

2. Paginação:

A memória virtual é dividida em blocos de tamanho fixo chamados *páginas*. A memória física é dividida em blocos chamados *quadros de página*. Quando um programa tenta acessar dados que não estão na memória RAM, o sistema operacional usa um processo chamado *page fault* (falha de página), que carrega a página solicitada do disco para a memória. O gerenciamento de páginas é feito através de uma estrutura chamada *tabela de páginas*, que armazena o mapeamento entre endereços virtuais e físicos.

3. Segregação de Memória e Proteção:

A memória virtual também fornece uma camada de segurança, isolando os processos uns dos outros. Isso significa que um processo não pode acessar a memória de outro, o que ajuda a proteger os dados e prevenir falhas. O sistema operacional pode garantir que o acesso a áreas de memória não autorizadas seja bloqueado, evitando que um processo sobrescreva a memória de outro, o que poderia causar corrupção de dados ou falhas.

1.2.2 Benefícios da Memória Virtual:

1. Abstração de Memória:

Os programas podem ser escritos sem se preocupar com a quantidade de memória

física disponível. O sistema operacional lida com a alocação de memória, tornando o desenvolvimento mais simples.

2. Execução de Programas Maiores:

Mesmo que o sistema tenha pouca memória física, é possível rodar programas que exigem mais memória do que a disponível, graças ao uso do espaço de troca no disco.

3. Eficiência e Compartilhamento:

A memória virtual permite que múltiplos processos compartilhem partes de código e dados, economizando memória. Por exemplo, várias instâncias do mesmo programa podem compartilhar as mesmas páginas de código, reduzindo o consumo de memória.

4. Proteção e Isolamento de Processos:

Cada processo tem seu próprio espaço de memória isolado dos outros, o que aumenta a segurança e a estabilidade do sistema.

1.2.3 Desvantagens:

1. Desempenho:

O uso de memória virtual pode afetar o desempenho. Quando o sistema precisa trocar dados entre a RAM e o disco, isso gera um processo chamado thrashing, onde o sistema gasta mais tempo trocando dados entre o disco e a memória do que executando os processos, o que pode diminuir o desempenho consideravelmente.

2. Uso de Disco:

O uso de disco rígido (ou SSD) como memória virtual pode ser muito mais lento do que o acesso à memória RAM, o que pode afetar a eficiência do sistema, especialmente se o espaço de swap for usado com frequência.

2 TESTES E RESULTADOS

No total, foram realizados (número) testes, com 2 algoritmos de substituição de páginas e 2 diferentes tamanhos de página. Os testes iniciaram com tamanho de memória primária de 5000, para os testes subsequentes o tamanho da memória era a metade do teste anterior. Todos os testes foram feitos utilizando o escalonador prioritário com o tempo de bloqueio de disco de 10 instruções.

FIFO: O algoritmo FIFO (First In, First Out) é um dos mais simples entre os algoritmos de substituição de páginas. Ele funciona de acordo com o princípio em que a primeira página a entrar na fila será a primeira a ser removida, como indica sua tradução literal: 'primeiro a entrar, primeiro a sair'.

Segunda chance: O algoritmo Segunda Chance é uma variação do FIFO. Ele funciona utilizando um bit de referência para cada página, que indica se ela foi usada recentemente. Quando chega a vez de uma página ser removida, verifica-se o valor desse bit. Se o bit for igual a 1, a página recebe uma segunda chance: o bit é zerado, e a página é movida para o final da fila. Se o bit for igual a 0, a página é removida. Esse método busca evitar a substituição de páginas que ainda estão em uso.

2.1 Resultados FIFO - tamanho de página = 7

Teste 1 - tamanho memória primária = 5000

Métricas do SO:

Tempo total de execução	*Page fault*
37996	125

Métricas dos processos:

X	Tempo de retorno	*Page fault*	Tempo de *page fault*
Processo 1	37990	24	8889
Processo 2	21232	34	6014
Processo 3	18224	33	7629
Processo 4	28007	34	8056

Teste 2 - tamanho memória primária = 2500

Métricas do SO:

Tempo total de execução	*Page fault*
37996	125

Métricas dos processos:

X	Tempo de retorno	*Page fault*	Tempo de *page fault*
Processo 1	37990	24	8889
Processo 2	21232	34	6014
Processo 3	18224	33	7629
Processo 4	28007	34	8056

Teste 3 - tamanho memória primária = 1250

Métricas do SO:

Tempo total de execução	*Page fault*
37996	125

Métricas dos processos:

X	Tempo de retorno	*Page fault*	Tempo de *page fault*
Processo 1	37990	24	8889
Processo 2	21232	34	6014
Processo 3	18224	33	7629
Processo 4	28007	34	8056

Teste 4 - tamanho memória primária = 625

Métricas do SO:

Tempo total de execução	*Page fault*
41999	151

Métricas dos processos:

X	Tempo de retorno	*Page fault*	Tempo de *page fault*
Processo 1	41993	31	11409
Processo 2	23285	42	8364
Processo 3	20099	39	9728
Processo 4	29760	39	9318

Teste 5 - tamanho memória primária = 315

Métricas do SO:

Tempo total de execução	*Page fault*
83529	431

Métricas dos processos:

X	Tempo de retorno	*Page fault*	Tempo de *page fault*
Processo 1	83523	33	12487
Processo 2	40163	99	26711
Processo 3	58556	145	45661
Processo 4	71290	154	47485

2.2 Resultados FIFO - tamanho de página = 28

Teste 1 - tamanho memória primária = 5000

Métricas do SO:

Tempo total de execução	*Page fault*
27046	34

Métricas dos processos:

X	Tempo de retorno	*Page fault*	Tempo de *page fault*
Processo 1	27030	7	2655
Processo 2	20023	9	2141
Processo 3	14056	9	2440
Processo 4	23330	9	2055

Teste 2 - tamanho memória primária = 2500

Métricas do SO:

Tempo total de execução	*Page fault*
27046	34

Métricas dos processos:

X	Tempo de retorno	*Page fault*	Tempo de *page fault*
Processo 1	27030	7	2655
Processo 2	20023	9	2141
Processo 3	14056	9	2440
Processo 4	23330	9	2055

Teste 3 - tamanho memória primária = 1250

Métricas do SO:

Tempo total de execução	*Page fault*
27046	34

Métricas dos processos:

X	Tempo de retorno	*Page fault*	Tempo de *page fault*
Processo 1	27030	7	2655
Processo 2	20023	9	2141
Processo 3	14056	9	2440
Processo 4	23330	9	2055

Teste 4 - tamanho memória primária = 625

Métricas do SO:

Tempo total de execução	*Page fault*
30884	58

Métricas dos processos:

X	Tempo de retorno	*Page fault*	Tempo de *page fault*
Processo 1	30878	12	4320
Processo 2	20821	16	3264
Processo 3	16004	14	4636
Processo 4	25838	16	4959

Teste 5 - tamanho memória primária = 315

Métricas do SO:

Tempo total de execução	*Page fault*
90529	486

Métricas dos processos:

X	Tempo de retorno	*Page fault*	Tempo de *page fault*
Processo 1	90523	18	6806
Processo 2	43657	76	28804
Processo 3	78049	183	62800
Processo 4	85017	209	59847

2.3 Resultados Segunda Chance - tamanho de página = 7

Teste 1 - tamanho memória primária = 5000

Métricas do SO:

Tempo total de execução	*Page fault*
37996	125

Métricas dos processos:

X	Tempo de retorno	*Page fault*	Tempo de *page fault*
Processo 1	37990	24	8889
Processo 2	21232	34	6014
Processo 3	18224	33	7629
Processo 4	28007	34	8056

Teste 2 - tamanho memória primária = 2500

Métricas do SO:

Tempo total de execução	*Page fault*
37996	125

Métricas dos processos:

X	Tempo de retorno	*Page fault*	Tempo de *page fault*
Processo 1	37990	24	8889
Processo 2	21232	34	6014
Processo 3	18224	33	7629
Processo 4	28007	34	8056

Teste 3 - tamanho memória primária = 1250

Métricas do SO:

Tempo total de execução	*Page fault*
37996	125

Métricas dos processos:

X	Tempo de retorno	*Page fault*	Tempo de *page fault*
Processo 1	37990	24	8889
Processo 2	21232	34	6014
Processo 3	18224	33	7629
Processo 4	28007	34	8056

Teste 4 - tamanho memória primária = 625

Métricas do SO:

Tempo total de execução	*Page fault*
41939	136

Métricas dos processos:

X	Tempo de retorno	*Page fault*	Tempo de *page fault*
Processo 1	41933	31	11558
Processo 2	20181	34	5807
Processo 3	17783	34	7679
Processo 4	29700	37	9450

Teste 5 - tamanho memória primária = 315

Métricas do SO:

Tempo total de execução	*Page fault*
56952	302

Métricas dos processos:

X	Tempo de retorno	*Page fault*	Tempo de *page fault*
Processo 1	56946	31	11607
Processo 2	38014	94	22969
Processo 3	45246	144	33600
Processo 4	35344	63	14720

2.3 Resultados Segunda Chance - tamanho de página = 28

Teste 1 - tamanho memória primária = 5000

Métricas do SO:

Tempo total de execução	*Page fault*
27046	34

Métricas dos processos:

X	Tempo de retorno	*Page fault*	Tempo de *page fault*
Processo 1	27030	7	2655
Processo 2	20023	9	2141
Processo 3	14056	9	2440
Processo 4	23330	9	2055

Teste 2 - tamanho memória primária = 2500

Métricas do SO:

Tempo total de execução	*Page fault*
27046	34

Métricas dos processos:

X	Tempo de retorno	*Page fault*	Tempo de *page fault*
Processo 1	27030	7	2655
Processo 2	20023	9	2141
Processo 3	14056	9	2440
Processo 4	23330	9	2055

Teste 3 - tamanho memória primária = 1250

Métricas do SO:

Tempo total de execução	*Page fault*
27046	34

Métricas dos processos:

X	Tempo de retorno	*Page fault*	Tempo de *page fault*
Processo 1	27030	7	2655
Processo 2	20023	9	2141
Processo 3	14056	9	2440
Processo 4	23330	9	2055

Teste 4 - tamanho memória primária = 625

Métricas do SO:

Tempo total de execução	*Page fault*
29257	51

Métricas dos processos:

X	Tempo de retorno	*Page fault*	Tempo de *page fault*
Processo 1	29251	12	4420
Processo 2	21594	14	3593
Processo 3	14827	12	3331
Processo 4	24211	13	3202

Teste 5 - tamanho memória primária = 315

Métricas do SO:

Tempo total de execução	*Page fault*
68737	329

Métricas dos processos:

X	Tempo de retorno	*Page fault*	Tempo de *page fault*
Processo 1	68731	16	6170
Processo 2	31466	56	17394
Processo 3	52774	124	40290
Processo 4	63228	133	40511

CONCLUSÃO

Com base nos testes realizados é possível concluir que, em situações onde a memória primária tem espaço suficiente para alocar todos os processos que devem ser executados, uma abordagem com páginas maiores se mostra mais efetiva. Contudo, no momento em que a memória primária vai diminuindo, tanto páginas pequenas quanto grandes demonstram uma piora no desempenho, mas essa piora é mais significativa em páginas maiores, fazendo com que nesse caso, uma abordagem com páginas menores, seja mais efetiva.

Ademais, independente do tamanho da memória primária, o algoritmo de substituição de páginas de segunda chance demonstrou um melhor desempenho que o FIFO para todos os casos.

REFERÊNCIAS

TANENBAUM, Andrew S.; BOS, Herbert. **Sistemas Operacionais Modernos**. 4. ed. São Paulo: Pearson Education, 2016. ISBN 978-85-4301-818-8.