Trabalho de Sistemas Operacionais “A”

Rafael Vales e Juliano Soares

rvales@inf.ufsm.br

jlsoares@inf.ufsm.br

1. Introdução

Este é um trabalho referente a disciplina de Sistemas Operacionais A, ele foi implementado em C e possui quatro partes. A primeira delas é a implementação de um buffer compartilhado por várias threads, que farão requisições para adicionar e remover dados dele. A segunda é a medição do desempenho deste buffer, cujo resultados serão analisados a seguir neste relatório. A terceira parte é a criação de uma simulação de disquete, e a quarta faz a junção dos códigos implementados anteriormente para fazer acessos ao disquete simulado no modo cliente-servidor.

2. Desempenho do buffer

O buffer implementado na parte 1 foi submetido a várias execuções com diferentes situações para medir seu desempenho, variando o número de threads escritoras e leitoras, a quantidade de dados que passaram pelo buffer durante a execução e a capacidade do buffer, que variou entre 512, 32K e 1M bytes. As escritas variam em tamanhos aleatórios em dois intervalos: de 0 a 25% e 20 a 50% do tamanho do buffer.

Segue a tabela com os resultados obtidos:

Tamanho dos dados: de 0 a 25% do tamanho do buffer						
	THREADS		800 DADOS		32000 DADOS	
TAM DO BUFFER	ESCRITORAS	LEITORAS	TEMPO (s)	FLUXO (dados/s)	TEMPO (s)	FLUXO (dados/s)
512 B	1	1	0,015	53.333,3	0,074	432.432,4
	1	8	0,018	44.444,4	0,180	177.777,8
	8	1	0,022	36.363,6	0,182	175.824,2
	8	4	0,023	34.782,6	0,124	258.064,5
	4	8	0,015	53.333,3	0,134	238.806,0
	16	16	0,021	38.095,2	0,148	216.216,2
	8	32	0,023	34.782,6	0,154	207.792,2
	32	8	0,030	26.666,7	0,203	157.635,5
64 KB	1	1	0,122	6.557,4	1,832	17.467,2
	1	8	0,091	8.791,2	1,996	16.032,1
	8	1	0,116	6.896,6	2,090	15.311,0
	8	4	0,106	7.547,2	2,145	14.918,4
	4	8	0,132	6.060,6	2,046	15.640,3
	16	16	0,091	8.791,2	2,085	15.347,7
	8	32	0,095	8.421,1	2,339	13.681,1
	32	8	0,129	6.201,6	2,131	15.016,4

1 MB	1	1	0,689	1.161,1	27,180	1.177,3
	1	8	0,719	1.112,7	27,414	1.167,3
	8	1	0,726	1.101,9	27,374	1.169,0
	8	4	0,691	1.157,7	27,651	1.157,3
	4	8	0,715	1.118,9	27,698	1.155,3
	16	16	0,744	1.075,3	27,640	1.157,7
	8	32	0,787	1.016,5	27,985	1.143,5
	32	8	0,738	1.084,0	28,202	1.134,7

Figura 1. Execuções com dados variando de 0 a 25% do tamanho do buffer

Tamanho dos dados: de 10 a 50% do tamanho do buffer						
	THREADS		800 DADOS		32000 DADOS	
TAM DO BUFFER	ESCRITORAS	LEITORAS	TEMPO (s)	FLUXO (dados/s)	TEMPO (s)	FLUXO (dados/s)
512 B	1	1	0,015	53.333,3	0,141	226.950,4
	1	8	0,014	57.142,9	0,225	142.222,2
	8	1	0,031	25.806,5	0,253	126.482,2
	8	4	0,027	29.629,6	0,219	146.118,7
	4	8	0,028	28.571,4	0,269	118.959,1
	16	16	0,030	26.666,7	0,183	174.863,4
	8	32	0,014	57.142,9	0,278	115.107,9
	32	8	0,032	25.000,0	0,285	112.280,7
64 KB	1	1	0,154	5.194,8	4,330	7.390,3
	1	8	0,144	5.555,6	4,436	7.213,7
	8	1	0,170	4.705,9	4,379	7.307,6
	8	4	0,241	3.319,5	4,481	7.141,3
	4	8	0,205	3.902,4	4,491	7.125,4
	16	16	0,254	3.149,6	4,497	7.115,9
	8	32	0,160	5.000,0	4,573	6.997,6
	32	8	0,156	5.128,2	4,655	6.874,3
1 MB	1	1	1,704	469,5	64,423	496,7
	1	8	1,689	473,7	65,426	489,1
	8	1	1,725	463,8	65,515	488,4
	8	4	1,664	480,8	65,190	490,9
	4	8	1,715	466,5	65,418	489,2
	16	16	1,693	472,5	65,906	485,5
	8	32	1,709	468,1	66,042	484,5
	32	8	1,727	463,2	66,061	484,4

Figura 2. Execuções com dados variando de 10 a 50% do tamanho do buffer

Quando o número de threads escritoras e leitoras são iguais, o fluxo de dados é maior em comparação com os outros casos para tamanhos de buffer menores. Com isso, pode-se perceber que um fator de 1 pra 1 nas threads, obtém o melhor desempenho. Pois para cada thread que inserir um dado, terá outra para remover esse dado e assim faz com que não acumule grandes filas de threads esperando enquanto o buffer está cheio, com muitas threads inserindo e poucas removendo, ou ao contrário, deixando-o vazio.

Além do mais, o tamanho dos dados é proporcional ao tamanho do buffer, assim, quanto maior o buffer, maior são os dados que passarão por ele. Com isso, o tempo para inserção e remoção se torna maior e faz com que o fluxo de dados que entra e sai do buffer a cada segundo, diminua. Isso pode ser visto na Tabela 2, onde em um buffer de 512 bytes o fluxo de dados por segundo fica na casa das dezenas de milhares, mas ao passar os testes para um buffer de 64 KB o valor diminui significativamente para poucas

unidades de milhar. Com o maior tamanho de capacidade, 1 MB, cai ainda mais para apenas algumas centenas. Assim, mostra que como já esperado, quanto maior o tamanho do dado inserido/removido, mais lento é para fazer a transferência e mais lento o buffer fica.

Os testes foram realizados em um computador com sistema Ubuntu 18.04.1 LTS, processador Intel Core i5-7200U, 8 GB de memória RAM e placa gráfica GeForce 920MX. O processador possui 2 núcleos com 4 threads, trabalhando a uma frequência de 2.50 Ghz e caches L1 de 32K para dados e para instruções.

3. Unidade de disquete

A parte 3 deste trabalho se baseou na unidade de disquete IBM PC 360 KB. Dupla face, ela possui 40 cilindros e 9 setores por trilha, totalizando 720 setores de 512 bytes que resultam em uma capacidade total de 360 KB. Tem uma velocidade de rotação de 200ms por volta, e demora 250ms para iniciar ou parar o motor. Possui um tempo de transferência de um setor de 22ms e o tempo de troca do cabeçalho de uma trilha para outra adjacente de 6ms, com o tempo médio de busca de 77ms. Essas informações foram retiradas do livro Sistemas Operacionais Modernos de Tanenbaum[1].

Com a implementação de uma simulação para este disquete, foram testados os vários fatores de entrelaçamento. Se fez testes de gravação e leitura, com acesso sequencial e aleatório. Foram realizados pedidos de 50 blocos para cada caso.

Os testes de acesso aleatório possuíram praticamente o mesmo tempo de acesso independente do fator de entrelaçamento, ficando com uma média por volta dos 190ms por bloco requisitado, tanto para leitura como para gravação.

Quanto aos acessos de escrita e leitura sequencial, o desempenho foi bastante superior. Cada acesso ao disquete demorou cerca de 35ms sem entrelaçamento, e a cada fator de entrelaçamento adicional o tempo de acesso aumentou em cerca de 20ms.

Com isso, o melhor fator de entrelaçamento de setores para esse caso é 0, ou seja, sem entrelaçamento. Os resultados completos podem ser vistos no gráficos abaixo ou no arquivo saída *disquete_desempenho.txt* presente junto aos arquivos fonte.

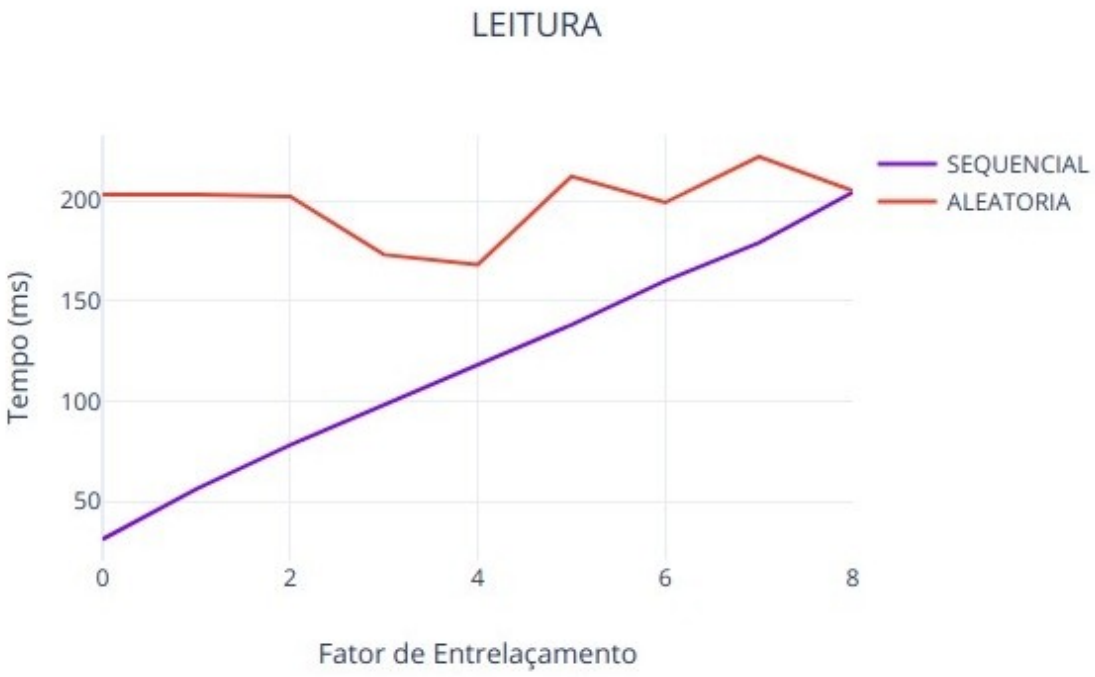

Figura 3. Desempenho da leitura sequencial e aleatória

Figura 4. Desempenho da escrita sequencial e aleatória

Referências

[1] Tanenbaum, A. (2015), Sistemas Operacionais Modernos, 3ª edição.