Relatório do Trabalho de Sistemas Operacionais

Lucas Micol [1], **Claiton Neisse**[1]

[1]Ciência da Computação – Universidade Federal de Santa Maria (UFSM)
Santa Maria – RS – Brazil

{lmpolicarpo,chneisse}@inf.ufsm.br

1. Informações Gerais

Este relatório apresenta uma análise do que foi desenolvido no trabalho final da disciplina de sistemas operacionais.

Como foi determinado na parte dois deste trabalho, o relatório contém uma análise de desempenho do buffer implementado em C, assim como uma explicação de suas funções. Na parte três, a implementação de disco, e por fim a parte quatro onde as implementações são usadas em uma simulação de servidor-cliente.

2. Ambiente de Testes

Todos os teste foram realizados em um notebook Dell Inspiron 5558, com processador Intel Dual Core i7-5500U com 4MB de cache e 3.00 GHz e 8GB de RAM. De acordo com o site da Intel, o processador possui espaço para 4 threads por núcleo.

3. Buffer

3.1. Implementação do Buffer

O buffer foi implementado de maneira circular com exclussão mútua, exatamente como foi pedido na descrição do trabalho. Os semáforos bloqueiam as remoções quando não há dado a remover. Quando não há espaço no buffer um aviso é retornado de que não foi possível inserir o dado, e ao final da inserção ocorre um post no semáforo de remoção. Todas as funções foram implementadas em detalhes.

3.2. Análise de Desempenho do Buffer

Nosso buffer compartilhado utiliza um TAD para armazenamento de dados em paralelo, além do próprio vetor e das variáveis de controle e semáforos, implementamos uma lista que armazena o tamanho dos dados, a lista é implementada em FIFO, assim só podendo ser retirado os dados em ordem.

Quanto ao desempenho, efetuamos alguns teste, primeiro setamos o tamanho do buffer para 720 KBytes, e testamos diferentes entradas, a tabela a seguir amostra os resultados obtidos. Os números em vermelho representam (”Threads Insersoras-¿ ”Threads Remosoras”), os em azuis o tamanho em Bytes. Todos os tempos estão em microssegundos.

Todos os testes foram realizados com o arquivo ”main_teste_buffer.c, enviado em anexo com o resto do trabalho”

Com os dados acima podemos perceber algumas coisas interessantes, como o fato de que mesmo quando temos 20 threads inserindo e uma removendo (coluna 3), o tempo

Tamanho das Inserções	1 -> 1	1 -> 20	20 -> 1	20 -> 20
500 B	278	2.292	1.634	3.158
5000 B	558	6.062	3.351	5.140
13000 B	870	10.160	6.146	9.505
35000 B	2.096	16.014	11.709	16.183

Figura 1. Resultados dos testes

acaba sendo muito menor do que no caso oposto, uma thread inserindo e 20 removendo (coluna 2), isso se deve ao fato das inserções serem disparadas primeiro pelo CPU, devido a forma como o código está escrito.

Outra situação interessante de comentar é a de inserção 1 -¿ 1 com 35KB, onde a maior parte do tempo de processamento foi usado para colocar dados no buffer para inserção, as threads em sí ocuparam muito pouco tempo de CPU.

E uma última comparação que gostaríamos de fazer é relativo a segunda coluna com a última, o tempo de execução de 20 -¿ 20 foi muito melhor, visto que na situação onde temos apenas uma thread para inserir esta threads acaba demorando para ser escalonada já que concorre com as outras até estas bloquerem no semáforo de remoção, essa situação fez o tempo de execução disparar na frente dos outros.

4. Disco

4.1. Implementação

Para implementação de um disco simulado foram utilizados os seguintes dados sobre uma unidade de disquete:

Disquete 3.5’	
Capacidade:	1.44MB
Velocidade de rotação:	300 RPM
Número de trilhas:	80
Setores por trilha:	18
Trilhas por cilindro:	2
Tamanho do setor:	512 bytes
Total de setores:	2880
Tempo de rotação:	200 ms
Tempo transferência de um setor:	11,11 ms
Tempo troca de trilha adjacente:	6 ms
Tempo troca de trilha médio:	77 ms
Taxa de transferência:	500 Kbits/s ou 62500 bytes/s

A simulação utiliza uma imagem do disco armazenada em um arquivo e recebe pedidos de leitura e de gravação de blocos do disco, demorando para responder cada

pedido o tempo aproximado ao que o disco real demoraria.

Nesse tempo estão incluídos o tempo de busca pelo cilindro certo, considerado o tempo médio de troca de trilha, o tempo de seleção do cabeçote, considerado 1/10 do tempo de busca de uma trilha, o tempo de espera rotacional, o tempo de transferência do disco para a controladora e o tempo de transferência para a memória. O tempo de troca do cabeçote só é considerado quando o identificador do setor estiver na segunda superfície.

A interface de acesso é a função ”disco_simulador”que recebe a identificação do setor (linear e começando em 0), o tipo de acesso (leitura ou gravação) e um buffer para os dados (um vetor de 512 bytes).

Essa interface utiliza o fator de entrelaçamento calculado a partir dos testes feitos com a função ”disco_entrelacamento”. A função ”disco_simulador_sem_entrelacamento”implementa a interface sem entrelaçamento.

4.2. Testes do Disco

Para medir o desempenho do disco simulado foi escrito um programa (disponivel em ”main_teste_disco.c”) que varia o fator de entrelaçamento e a quantidade de setores gravados (predefinidos em 25%, 50%, 75% e 100% da quantidade de total de setores), alternando entre leitura e gravação, sequencial ou aleatório. Foi utilizada ”ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghijklmnopqrstuvwxyz0123456789”como string de teste. A escolha do entrelaçamento se deu pelo menor tempo médio, considerando a totalidade das possibilidades para cada fator de entrelaçamento. O arquivo ”resultados_disco_entrelacamento.csv”contem os resultados dos testes realizados.

A tabela a seguir apresenta o tempo médio, em microssegundos e por fator de entrelaçamento, entre leitura e gravação, sequencial ou aleatória, consideradas diferentes quantidades de setores gravados. O fator de entrelaçamento escolhido foi o 16, por apresentar o menor tempo médio nos testes.

Fator de Entrelacamento	**Tempo Médio (μ segundos)**
0	14824211,9375
1	14812024,4375
2	14855107,625
3	14831405,25
4	14809573,8125
5	14829922,25
6	14814464,625
7	14807231,6875
8	14803135,0625
9	14814800,375
10	14820474,8125
11	14808207,4375

Fator de Entrelacamento	Tempo Médio (μ segundos)
12	14802088,5625
13	14803747,8125
14	14804634,1875
15	14801825,875
16	14801197,375

5. Cliente - Servidor

Por último o arquivo ”main.c”une as partes anteriores em um programa que simula uma comunicação ”Cliente¡-¿Servidor”, como número de clientes é variável, no inicio do programa a variável definida ”T_CLIENTES”, modifica o número de threads clientes do programa inteiro. As threads têm apenas uma função, as clientes executam as requisições e a servidor as respostas. O programa possui um único semáforo para a thread do servidor não executar caso não tenha o que retirar do próprio buffer.

A função ”request”insere no buffer do servidor pacotes de 512 bytes, partes desses bytes são comprometidos com o cabeçalho do request, que guarda as informações necessárias para a resposta do servidor. A função ”response”acessa o buffer do servidor e responde as requisições com o método FIFO, não implementamos o algoritmo do elevador.

Por último, os clientes têm um ”sleep”aleatório entre 2 a 4 segundos, apenas para facilitar a leitura das respostas e requisições. O servidor responde imediatamente após a simulação do disco.

6. Executar

Para compilar o código basta compilar com o seguinte comando:

$ gcc -o cliente-servidor main.c buffer.c lista.c disco.c -pthread -Wall -Wextra -pedantic

E para executar basta usar o comando:

$ ./cliente-servidor