Organização e Recuperação de Dados

Profa. Valéria

5ª Lista de Exercícios

1. O desempenho de um processo de ordenação interna é, geralmente, medido em função do número de comparações necessárias. Explique por que o número de comparações não é uma medida adequada para avaliar o desempenho de métodos de ordenação de grandes arquivos que não cabem em memória.

   Comparações são operações realizadas em memória e qualquer operação em memória tem custo irrelevante quando comparado ao custo de uma operação em disco. Desse modo, quando se estima o desempenho de métodos de ordenação externa (em disco), o que se mede é o número de leituras necessárias.

2. Escreva um algoritmo que faça o seguinte:
   a. Examine o conteúdo de dois arquivos ordenados, M1 e M2;
   b. Produza um terceiro arquivo chamado COMUM contendo a cópia dos registros que existam nos dois arquivos (AND);
   c. Produza um quarto arquivo chamado DIFERENTE contendo os registros dos dois arquivos que existam em apenas M1 ou M2 (XOR).

   (exercícios de programação não serão incluídos no gabarito
   por haver mais de uma forma de resolvê-los)

3. Suponha que um arquivo de dados com 6.000 registros, mantido em disco, deve ser ordenado em um computador cuja memória interna acomoda no máximo 600 registros por vez, usando o procedimento de *merge sort* externo. Essa área de memória interna é usada como *buffer* de entrada para leitura de dados do disco e o sistema conta com um *buffer* de saída adicional que acomoda 200 registros.

   a. Durante o *merge*, quantos registros serão lidos de cada partição cada vez que ela é acessada? Justifique.

      Como o buffer de entrada acomoda 600 registros e há 6.000 registros no arquivo, serão geradas 6.000/600 = 10 partições.
      Durante o merging, o buffer de entrada será então subdividido em 10 buffers que acomodarão 60 registros cada. Dessa forma, durante a etapa de merging, serão lidos 60 registros de cada partição cada vez que elas forem acessadas.

   b. Quantos *seeks* serão realizados para a leitura dos registros durante a etapa de *merge* (excluindo a fase de geração de partições)?

      Cada partição terá que ser acessada 10 vezes para ser lida por completo. Como temos 10 partições, o número de seeks será 10 × 10 = 100 seeks.

   c. Quantos *seeks* serão realizados para escrever o arquivo ordenado no disco? Justifique.

      O buffer de saída acomoda 200 registros e terão que ser escritos 6.000 registros. Desse modo, o número de seeks para a escrita do arquivo ordenado será 6.000/200 = 30 seeks.

4. O *merge sort* em múltiplos passos requer mais processamento interno e operações de E/S extras. Então por que, em geral, é mais rápido do que o *merge sort* em passo único?

Porque ele permite alocar buffers maiores para o passo de merging, que é o passo mais custoso do merge. Com buffers maiores, o número de seeks no passo de merging será menor e essa redução compensará o custo das passadas extras.

5. Considerando o arquivo de 800 MB usado como exemplo nas aulas, um *buffer* de entrada de 4 MB e *buffers* de saída de 200 KB, calcule o custo total e de cada fase do *merge sort* (em termos de bytes transferidos e número de *seeks*) quando realizado em dois passos: primeiro 20×10-vias, depois 20-vias.

   Arquivo de 800MB ÷ 4MB = 200 partições

   Passo 1: 200 seeks de leitura + transferência de 800MB

   Passo 2: 200 seeks de escrita + transferência de 800MB

   1ª passada (20 merges de 10-vias)

   Passo 3-1: Nos merges de 10-vias, cada partição alocará 1/10 do buffer e, portanto, será lida 10 vezes

   Como temos 10 partições, teremos 10 × 10 = 100 seeks de leitura para um merge de 10-vias

   Como temos 20 merges de 10-vias, 20 × 100 = 2.000 seeks de leitura + transf. de 800MB

   Passo 4-1: 800MB ➜ 800.000KB ÷ 200KB = 4.000 seeks de escrita + transf. de 800MB

   2ª passada (1 merge de 20-vias)

   Passo 3-2: Cada partição alocará 1/20 do buffer, o que corresponde a 1/200 da partição (uma vez que as partições são maiores do que o buffer). Portanto, cada partição será lida 200 vezes.

   Como temos 20 partições, 20 × 200 = 4.000 seeks de leitura + transf. de 800MB

   Passo 4-2: 4.000 seeks de escrita + transf. de 800MB

   Custo total = 200 + 200 + 2.000 + 4.000 + 4.000 + 4.000 = **14.400 seeks + transf. de 4.800MB ~ 4,8 GB**

6. Considerando um arquivo de 10 GB, um *buffer* de entrada de 20MB e *buffers* de saída de 250 KB, calcule o custo total e de cada fase do *merge sort* (em termos de bytes transmitidos e número de *seeks*) em um passo. Em seguida, calcule o custo do *merge* quando realizado em dois passos: primeiro 25×20-vias, depois 25-vias.

   Arquivo de 10GB = 10.000MB ÷ 20MB = 500 partições

   Passo 1: 500 seeks de leitura + transferência de 10GB

   Passo 2: 500 seeks de escrita + transferência de 10GB

   1ª passada (25 merges de 20-vias)

   Passo 3-1: Nos merges de 20-vias, cada partição alocará 1/20 do buffer e, portanto, será lida 20 vezes

   Como temos 20 partições, teremos 20 × 20 = 400 seeks de leitura para um merge de 20-vias

   Como temos 25 merges de 20-vias, 25 × 400 = 10.000 seeks de leitura + transf. de 10GB

   Passo 4-1: 10.000MB ➜ 10.000.000KB ÷ 250KB = 40.000 seeks de escrita + transf. de 10GB

2ª passada (1 merge de 25-vias)

Passo 3-2: Cada partição alocará 1/25 do buffer, o que corresponde a 1/500 de uma partição (uma vez que as partições são maiores do que o buffer). Portanto, cada partição será lida 500 vezes.

Como temos 25 partições, 25 × 500 = 12.500 seeks de leitura + transf. de 10GB

Passo 4-2: 40.000 seeks de escrita + transf. de 10GB

Custo total = 500 + 500 + 10.000 + 40.000 + 12.500 + 40.000 = **103.500 seeks + transf. de 60 GB**