FUNDAÇÃO CENTRO DE ANÁLISE, PESQUISA E INOVAÇÃO TECNOLÓGICA – FUCAPI

ORDENAÇÃO EXTERNA POR INTERCALAÇÃO (MERGE-SORT) MULTIVIAS

RAFAEL DA SILVA ARAUJO – 072188
ROSIANE BRITO DE LIMA – 051334
SUZAN NELY CARDOSO OLIVEIRA – 135763
WILLIAN RODRIGUES ALVES - 102326

Manaus
2016

1. INTRODUÇÃO

O trabalho consistiu na implementação de um algoritmo de ordenação externa por intercalação (Merge-Sort) multivias, onde foram realizados experimentos para ordenar um arquivo de dados fornecido como entrada.

2. OBJETIVO GERAL

Implementar um algoritmo de ordenção externa por intercalação (Merge-Sort) multivias para ordenar um arquivo de dados fornecido como entrada.

3. OBJETIVOS ESPECÍFICOS

Ordenar um arquivo de dados fornecido como entrada
Apresentando a complexidade de tempo do algoritmo em memoria secundária
Implementar um software para ordenar arquivos cujos registros são inteiros de 32 bits.

4. PROCEDIMENTOS EXPERIMENTAIS

Da ordenação externa

Quando é preciso ordenar uma base de dados muito grande, que não cabe na memoria principal, assume-se que os dados deve ser recuperados a partir de dispositivos externos, como o acesso á memoria secundaria é muito mais lento, a maior preocupação passa a minimizar a quantidade de leituras e escritas nesses dispositivos, basicamente,preocupa-se com a quantidade de operações envolvendo a transferencia de blocos de registros entre as memorias primaria e secundaria, (operações de leitura e escrita) sendo possivel ordenar um arquivo grande em disco *separando-o em k arquivos ordenados em RAM* e fazendo a fusão intercalada desses arquivos, esse K e o tamanho dos arquivos são determinados pela memoria primaria disponivel e pelo tamanho de arquivo a ser ordenado, operando com 4 arquivos, o registros são lidos de 2 arquivos de origem e rescritos de forma parcialmente ordenada em arquivo de destino, então resumindo tivemos com estrategia:

Quebrar o arquivo em blocos que caibam na memoria disponivel
Ordenar cada bloco na RAM
Intercalar os blocos ordenados, gerando um o arquivo ordenado

Das bibliotecas utilizadas

<stdio.h> Para manipulção de entrada e saida dos dados,
<stdlib.h> Para alocação de memória,
<stdint.h> Definição de tipos de dados inteiros.
<time.h> Conversão de tipos de dado de data e horário.
<math.h> Para calculos matemáticos,

<string.h> Tratamento de cadeia de caracteres,
<unistd.h> cabeçalho com a definição das chamadas de sistema.

Do Hardware, Sistema Operacional e Compilador:

Processador Intel Core i7 1.8Ghz 64bits 8GB de memoria, SSD de 120GB
Linux Ubuntu 14.04 LTS
gcc (Ubuntu 4.8.4-2ubuntu1~14.04.1) 4.8.4

Da Compilação

make

ou

gcc ./src/gerar_arquivo.o ./src/gerar_arquivo_main.o -o gerador
gcc ./src/ordenar_arquivo.o ./src/ordenar_arquivo_main.o -o ordenar

Da Execução

make gerar

ou

./gerador nome_arquivo tamanho_arquivo

Make ordenar
./ordenar entrada_arquivo saida_arquivo memoria K

Dos Testes

Durante os testes foram coletados :

- Entrada e saida de dados de diferentes tamanhos de arquivo.
- Contagem do tempo (s) gasto durante a execução do algoritmo.
- Quantidade total de memoria disponivel para ordenção.
- Numeros de vias (K) usada pelo Merge-Sort.

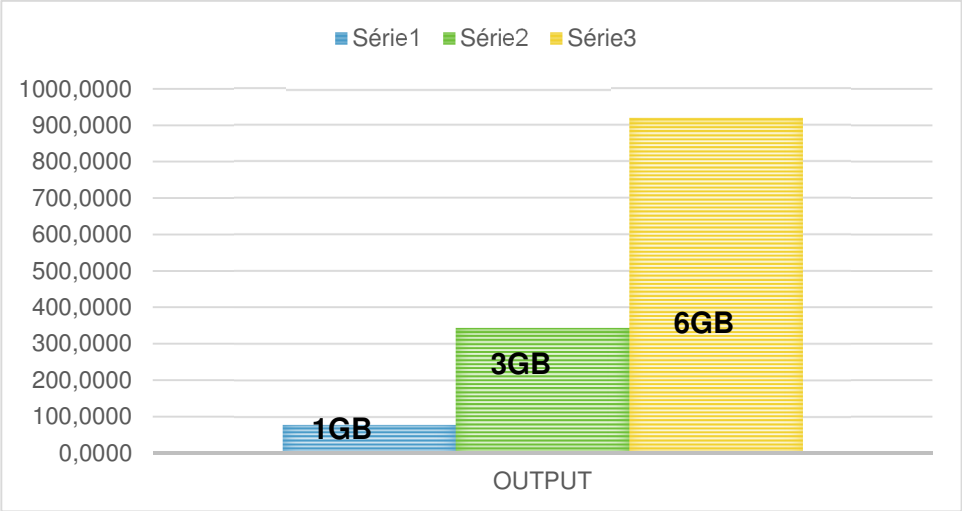

Grafico 1: Entada de dados tempo de execução por tamanho de arquivo.

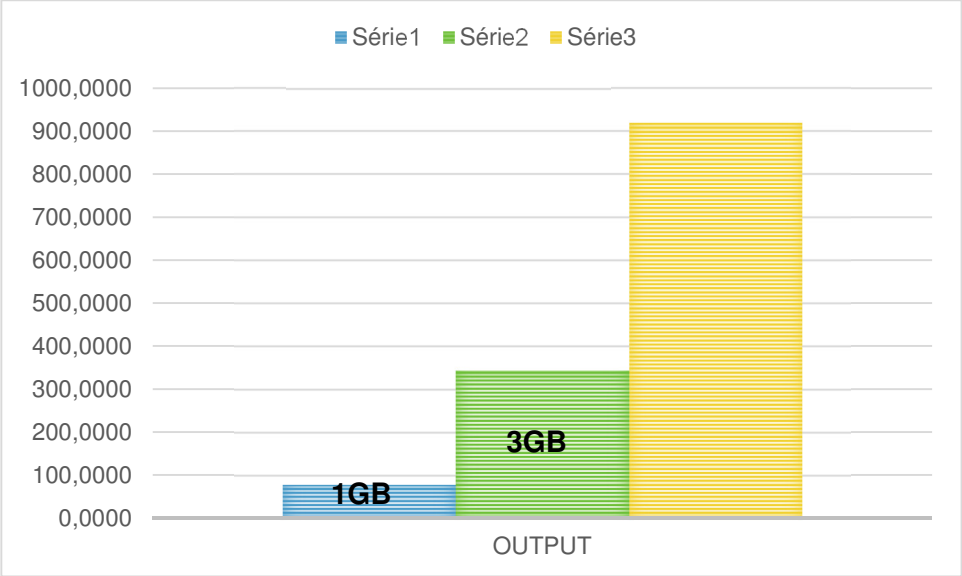

Grafico 2 : Saída de dados tempo gasto por tamanho de arquivo.

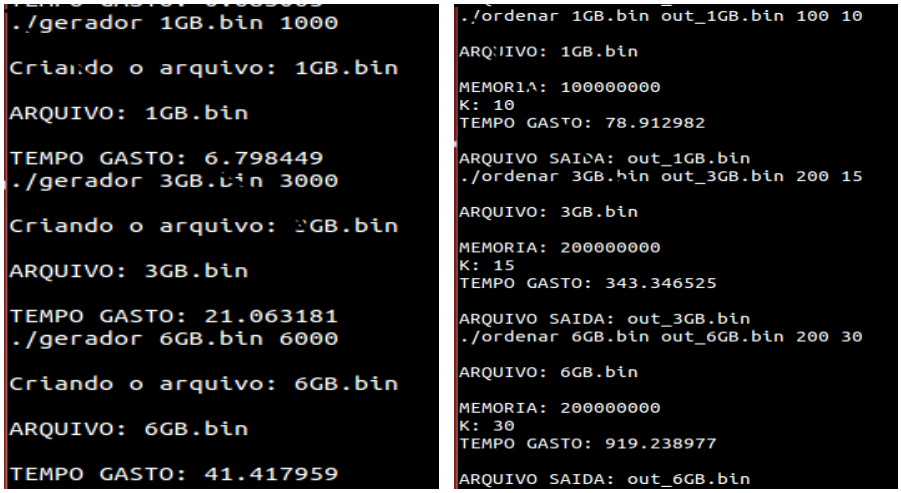

Figura 1:Detalhes de geração e ordenação.

Tem-se que cada passagem requer a leitura e escrita de 2 arquivos, cada um com aproximadamente n/2 registros, Numero de acessos em cada passagem ~= 4(n/2) ~= 2n. O numero de leituras e escritas de blocos em cada passagem é em torno de2n/b, onde b é o tamanho do bloco, então, o numero total de leitura e escritas de blocos em todo o processo de ordenação é em torno de (2n log n) / b, ou seja é excelente ordem O(n log n).

5. CONCLUSÃO

Em nossa implemetação foram realizados utilizando ordenação externa por intercalação (Merge-Sort) multivias, cujo o arquivo de entrada são registros de inteiros de 32 bits, utilizando uma função randômica de números aleatórios não ordenados de até 6Gb, Podemos concluir que esgota-se completamente o potencial de ordenação em memoria interna e aplica-se ordenação externa apenas em arquivos cujas rodadas superam a capacidade interna da memoria que quando existe a necessidade de classificar registros em sistemas e a memória principal não possui espaço que comporte todos os registros, utilizar a ordenação externa é uma solução, porém ao utilizar memória auxiliar o consumo de memória é muito alto e muito mais lento também.

A implementação do trabalho prático também se encontra disponivel no repositorio GitHub (https://github.com/WilAlves/ordenacaoExterna)