Sistemas Operacionais

Algoritmo sequencial de gerenciamento dinâmico de memória

Discentes:
Otávio Farias
Thiago Pfeifer
Gabriel Louzada

O que deveria ser feito?

- Implementar um gerador de requisições aleatórias, com tais confinado entre um valor mínimo (16 bytes) e um máximo (1 KB)
- **First-fit** foi escolhido
- Cada requisição deve ter um ID exclusivo e o valor deste ID deve ser escrito em todas as posições da heap ocupadas pela variável correspondente ao ID.
- O sistema deve apresentar um resumo das principais informações resultantes da execução do programa

Main

- Inicia o programa, valida os argumentos da linha de comando.
- Cria uma instância de Buffer e gera requisições aleatórias.
- Mede o tempo de execução e salva os resultados no CSV.
  Validação dos parâmetros.
- Loop principal que insere requisições.
- Uso do CsvLogger para registrar dados da execução.

Main

```java
public class Main {
    public static void main(String[] args) {
        if (args.length != 6) {
            System.out.println("Uso correto: java Main <tamanho_buffer> <min_requisicao> <max_requisicao> <num_requisicoes> <garbagePercent> <tamanhoBloco>");
            return;
        }

        int tamanhoBuffer = Integer.parseInt(args[0]);
        int min = Integer.parseInt(args[1]);
        int max = Integer.parseInt(args[2]);
        int numeroDeRequisicoes = Integer.parseInt(args[3]);
        int garbagePercent = Integer.parseInt(args[4]);
        int bloco = Integer.parseInt(args[5]);
        int id = 0;

        if(garbagePercent > 100 || garbagePercent < 0) garbagePercent = 30;
        if(bloco < 1) bloco = 1;
        if(min < 16) min = 16;
        if(tamanhoBuffer < min) tamanhoBuffer = min + 1;
        if(max > tamanhoBuffer) max = tamanhoBuffer;
        if(max < min) max = min + 1;
        if(bloco > tamanhoBuffer) bloco = tamanhoBuffer;
```

Buffer

- Armazenar as requisições em uma "heap" simplificada (vetor).
- Gerenciar inserções, remoções e compactação de memória.
- heap, fila, size, MaxSize, numCompac, numRemocao.

- **Principais métodos:**
  - **insert(Requisicao item):** Tenta inserir uma requisição.
  - **remove():** Remove a mais antiga (FIFO).
  - **garbageCollector(int tamanho):** Remove elementos até ter espaço.
  - **compactador():** Reorganiza a heap eliminando espaços vazios

Buffer

- Garbage Collector e compactador

```
85      public void garbageCollector(int tamanho){//verificar se if para chamar garbage não é mais eficiente
86
87          while(100-size*100/MaxSize < garbagePercent || (MaxSize - size) < tamanho){
88              remove();
89          }
90          numCompac++;
91          compactador();
92      }
93
94      public void compactador(){  1 usage
95
96          for(int i = MaxSize - 1; i > 0; i--){
97              if(heap[i] == null){
98
99                  heap[i] = heap[i-1];
100                 heap[i-1] = null;
101
102             }
103         }
```

Buffer

- Inserir e Inserir na Heap

```
44  @   private int inserirHeap(Requisicao item){  1 usage
45          int count = 0;
46          int i = 0;
47          if(item.tamanho > MaxSize - size) return 0;
48          while(true){
49              if(heap[i] == null && i < MaxSize){
50                  count++;
51                  if(count == item.tamanho){
52                      fila.add(item.id);      // adiciona o id à fila
53                      while(count > 0){
54                          heap[i] = item.id;
55                          i--;
56                          count--;
57                      }
58                      return 1;
59                  }
60              }
61              else{
62                  count = 0;
63              }
64              i++;
65              if (i == MaxSize) return 0;
66          }
```

Buffer

- Inserir e Inserir na Heap

```
30      public void insert(Requisicao item) {  2 usages
31          if (inserirHeap(item) == 1){
32              tamanhoTotalBloco += item.tamanho;
33              tamanhoTotalReq += item.tamanhoR;
34              size += item.tamanho;
35              System.out.println("id " + item.id + " tamanho " + item.tamanho + " adicionado com sucesso!");
36              System.out.println("Espaço usado da heap:" + size);
37          } else {
38              System.out.println("Buffer cheio");
39              garbageCollector(item.tamanho);
40              insert(item);
41          }
```

Requisição

- Armazenar os dados de uma requisição: ID, tamanho original e tamanho convertido para blocos
- id, tamanho, tamanhoR

```
3
4       public Integer id;  8 usages
5       public int tamanho;  9 usages
6       public int tamanhoR;  2 usages
7
8       public Requisicao(int id, int tamanhoR, int bloco){  no usages
9           this.id = id;
10          this.tamanhoR = tamanhoR;
11          this.tamanho = ((tamanhoR + bloco - 1)/bloco ) * bloco;
12      }
```

CSV Logger

- Grava os resultados da execução em um arquivo CSV.
- Permite uma melhor análise posterior de desempenho.
- Tamanho do buffer, parâmetros da requisição, contagem de remoção e compactação, tempo, tamanhos médios.

```
8   private static final String FILE_NAME = "tempos.csv";  1 usage
9   private static final String HEADER = "tamanhoBuffer,min,max,numExecucao,numRemocao,numCompactacao,garbagePercent,bloco,tamanhoMedioReq,tamanhoMedioBloco,tempo";  1 usage
10
11  public static void salvarDados(int tamanhoBuffer, int min, int max, int numExecucao, int numRemocao, int numCompactacao, int garbagePercent, int bloco, int tamanhoMedi
12      File file = new File(FILE_NAME);
13      boolean fileExists = file.exists();
14
15      try (FileWriter writer = new FileWriter(file,  append: true)) {
16          // Escreve o cabeçalho apenas se o arquivo ainda não existir
17          if (!fileExists) {
18              writer.write( str: HEADER + "\n");
19          }
20
21          // Adiciona a nova linha de dados
22          writer.write(String.format("%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d\n", tamanhoBuffer, min, max, numExecucao, numRemocao, numCompactacao,garbagePercent, bloco, tamanh
23      } catch (IOException e) {
24          e.printStackTrace();
```

Como foi testado?

```
1   JAVA_CLASS=tools.Main
2
3   run:
4           @for i in 0 10 20 30 40 50 60 70 80 90 100; do \
5               for k in 1024 2048 4096 8192 16384; do \
6                 for r in 1 2 3 4 5 6 7 8 9 10; do \
7                   echo "Execução $$r: java $(JAVA_CLASS) $$k 16 1024 1000 $$i 1"; \
8                   java $(JAVA_CLASS) $$k 16 1024 1000 $$i 1; \
9               done; \
10            done; \
11          done
```

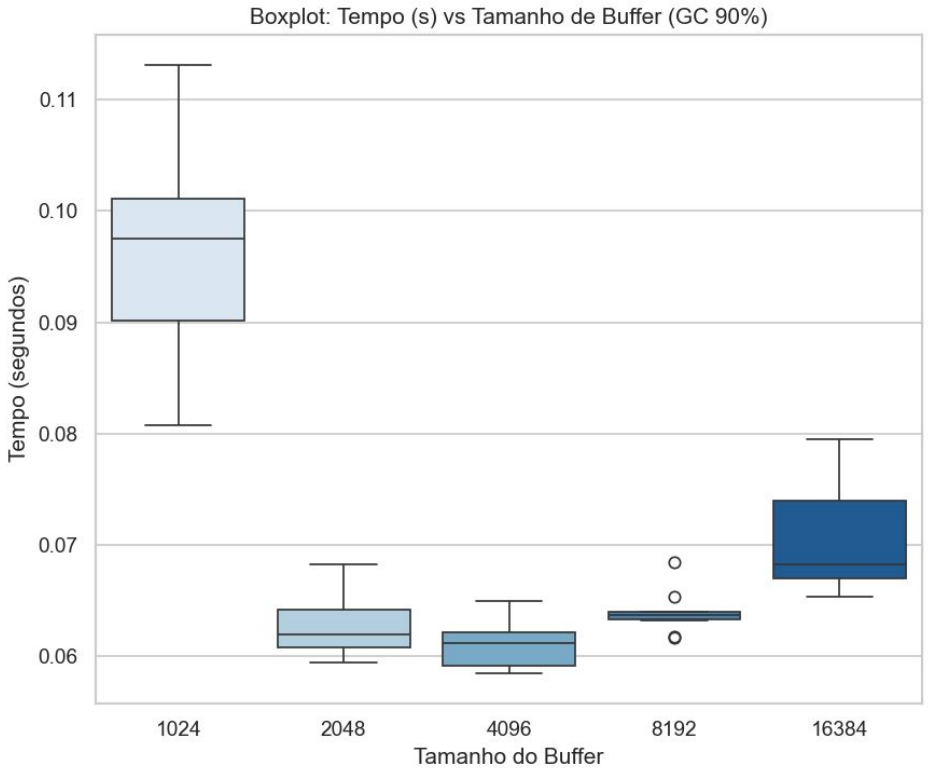
Tabelas de resultados

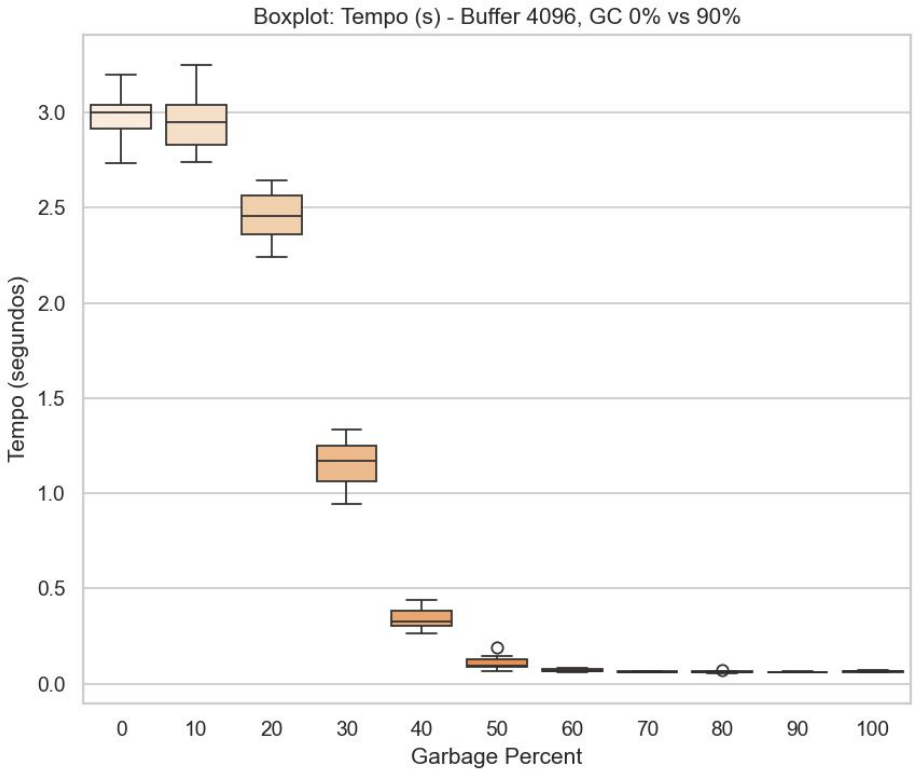
tamanhoBuffer	min	max	numRemocao	numCompactacao	garbagePercent	tamanhoMedioReq	tamanhoMedioBloco	tempo
4096	16	1024	146,5	995	90	519,3	519,3	61231893
4096	16	1024	171,5	995,2	80	517,8	517,8	61670604,1
4096	16	1024	207,3	993,8	70	517,4	517,4	62226184
2048	16	1024	303,8	996,9	100	522,9	522,9	62491000,2
2048	16	1024	339,9	996,1	90	520,9	520,9	62645065
8192	16	1024	73	990	90	517,8	517,8	63958127,6
4096	16	1024	137,7	994,8	100	518,2	518,2	64586430
8192	16	1024	99,1	986,8	70	520,3	520,3	66482932,8
8192	16	1024	66,1	991,3	100	521,2	521,2	67753194,3
8192	16	1024	115,6	988	60	517,3	517,3	67819000,6
1024	16	1024	678,7	997,7	100	519	519	69476391,3
16384	16	1024	55,2	976,7	60	520,6	520,6	69524333
16384	16	1024	40,6	979,1	80	516,5	516,5	69566892,5
8192	16	1024	84,6	991,6	80	517,5	517,5	69973855,2
16384	16	1024	36,1	981,2	90	521,2	521,2	70332306,3
16384	16	1024	32,1	985	100	521,4	521,4	70436339,6
4096	16	1024	630,8	993,3	60	522,8	522,8	70628526,9
2048	16	1024	1174,4	997	80	521,5	521,5	70640868,7
16384	16	1024	46	977,9	70	512,7	512,7	71798637,5
8192	16	1024	150	987,6	50	521,7	521,7	72959190,5
16384	16	1024	68,8	977,3	50	519,8	519,8	75055140,2
16384	16	1024	88,8	972,9	40	518,6	518,6	75544933,9

tamanhoBuffer	min	max	numRemocao	numCompactacao	garbagePercent	tamanhoMedioReq	tamanhoMedioBloco	tempo
16384	16	1024	128,9	972,1	30	519,8	519,8	76190878,1
8192	16	1024	471,1	984,8	40	519,1	519,1	78728578,1
1024	16	1024	5576,7	997,5	90	527,9	527,9	96747994,2
2048	16	1024	4783,4	996,8	70	517,4	517,4	106074346
4096	16	1024	2997,4	993,3	50	521	521	108991331
1024	16	1024	9508	997,3	80	522	522	123596966,1
1024	16	1024	10582,8	997,5	70	516,3	516,3	127795444,7
1024	16	1024	13544,1	997,2	60	520,7	520,7	150491459,1
1024	16	1024	17108,7	997,5	40	518,8	518,8	163936986,3
16384	16	1024	2511	969,7	20	516,1	516,1	167269123,3
1024	16	1024	17612,4	997,9	0	524,2	524,2	168162160,1
1024	16	1024	15058,1	997,2	50	519	519	171469338
1024	16	1024	18338,3	996,7	10	522	522	172922904,5
1024	16	1024	18262,3	997,2	20	513,2	513,2	175131926,3
1024	16	1024	18468,2	997,4	30	520,5	520,5	182372939,7
2048	16	1024	14806,9	995,7	60	514,4	514,4	193744513,5
8192	16	1024	7278	985,4	30	519,1	519,1	232593409,8
4096	16	1024	18195,1	992,8	40	514,1	514,1	339914539,9

tamanhoBuffer	min	max	numRemocao	numCompactacao	garbagePercent	tamanhoMedioReq	tamanhoMedioBloco	tempo
2048	16	1024	40482,1	995,8	50	524	524	456626480,8
2048	16	1024	64505,6	995,6	40	521	521	681976829,1
2048	16	1024	82941,8	994,8	30	518,9	518,9	861233390,5
2048	16	1024	94032,7	995,9	20	517	517	996723393
2048	16	1024	96981,5	995,5	0	515,6	515,6	997964288,2
2048	16	1024	99737,9	995,7	10	525	525	1014708434
4096	16	1024	69838,2	992	30	521	521	1159890795
8192	16	1024	80718,8	984,3	20	524,9	524,9	1972793503
4096	16	1024	152251,4	992,2	20	517,9	517,9	2450748164
4096	16	1024	188118,8	991,4	10	517,1	517,1	2949056665
4096	16	1024	186480,5	991,5	0	518,3	518,3	2986206973
16384	16	1024	117450,3	968,4	10	517,1	517,1	4739925288
8192	16	1024	231377,8	982,9	10	516,4	516,4	5529771921
8192	16	1024	243610,9	984,1	0	523,3	523,3	6043202408
16384	16	1024	223870,6	968	0	518,9	518,9	8698202056

Resultados


Boxplot: Tempo (s) vs Tamanho de Buffer (GC 90%)
Tempo (segundos)
0.11
0.10
0.09
0.08
0.07
0.06
1024
2048
4096
8192
16384
Tamanho do Buffer


Boxplot: Tempo (s) - Buffer 4096, GC 0% vs 90%
Tempo (segundos)
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
10
20
30
40
50
60
70
80
90
100
Garbage Percent

Planejamento para o algoritmo paralelo

- Semáforos
- Inserções e remoções
- Sincronização do buffer
- Paralelismo no Garbage Collector
- Ferramenta JAVA

Referências Bibliográficas:

SILBERSCHATZ, A.; Galvin, P. B.; Gagne, G. Sistemas Operacionais com Java. 7. ed. Rio de Janeiro: Campus, 2007.

Obrigado pela atenção!