ACH2044 - Sistemas Operacionais

Prof. Norton Trevisan Roman

Relatório EP 02

Felipe Oliveira do Espirito Santo – 11925242

Fernando José Germinio Fenley - 11207630

Guilherme Jimenes - 11911021

Jalmir Iago Alves da Silva - 11808999

1. Escolha de solução para o problema de Leitores e Escritores

A solução escolhida para o problema de leitores e escritores utilizada como base para elaboração do projeto foi a da York University, que pode ser acessada por meio deste **link**.

Essa solução utiliza o seguinte princípio: há muitas threads concorrentes que desejam ler e escrever na região crítica. Assim, é aceitável ter vários processos de leitura lendo os dados da região crítica ao mesmo tempo. Porém, se uma thread está escrevendo, nenhum outro processo pode escrever ou ler ao mesmo tempo. Além disso, há uma priorização forte para as threads de leitura, ou seja, espera-se que todas as threads de leitura sejam executadas para que então as threads de escrita possam ser executadas. Em outras palavras, as threads de escrita ficam “em espera” até que todas as threads de leitura sejam finalizadas.

2. Definição das classes do projeto

Tanto a solução com leitor e escritor quanto a sem foram desenvolvidas no projeto para funcionarem com as mesmas classes que, por sua vez, possuem os mesmos atributos e métodos. O que se altera é o código presente nos métodos. Assim, como decisão de projeto, foram implementadas as seguintes classes:

Database

Responsável por armazenar um ArrayList contendo, em cada um de seus elementos, uma palavra do arquivo “bd.txt”. Pelo fato dessa estrutura se tratar de uma região crítica, ou seja, uma estrutura que será acessada por diferentes threads concorrentemente, a classe Database implementa também um mecanismo de controle de acesso a essa estrutura. Tal mecanismo difere de acordo com a solução(com leitor e escritor ou sem) e pode ser analisado por meio da leitura do código-fonte. Por fim, a classe também é responsável por realizar a leitura do arquivo “bd.txt” bem como inicialização e carga dos dados no ArrayList.

Com isso, a classe possui os seguintes atributos:

- private List<String> list;
- private int leitores; // armazena a quantidade de leitores que estão acessando a região crítica no momento.

A classe possui também os seguintes métodos:

- public void carregaEstruturaRAM() throws IOException: responsável por realizar a leitura do arquivo “bd.txt” bem como inicialização e carga dos dados no ArrayList. Lança uma exceção do tipo IOException caso não seja possível realizar a leitura do arquivo.
- public void read() throws InterruptedException: responsável por permitir a leitura de dados do ArrayList de palavras, ou seja, da região crítica. Lança uma exceção do tipo InterruptedException.
- public synchronized void write() throws InterruptedException: responsável por permitir a atualização dos dados do ArrayList de palavras, ou seja, da região crítica, por apenas uma thread por vez. Lança uma exceção do tipo InterruptedException.

Escritor

Classe responsável por representar uma thread de escrita ao ArrayList de palavras(região crítica).

Com isso, a classe possui o seguinte atributo:

- private Database database; // armazena uma instância da classe que armazena o ArrayList de palavras.

Por estender a classe Thread do Java, a classe possui apenas o método run() que faz uma chamada ao método read() da classe Database.

Leitor

Classe responsável por representar uma thread de leitura dos dados armazenados no ArrayList de palavras(região crítica).

Com isso, a classe possui os seguintes atributos:

- private Database database; // armazena uma instância da classe que armazena o ArrayList de palavras.

Por estender a classe Thread do Java, a classe possui apenas o método run() que faz uma chamada ao método write() da classe Database.

Sistema

Classe responsável por representar a thread principal que irá executar todas as threads leitoras e escritoras necessárias para os experimentos. Assim, a classe armazena um array com todas as 100 threads necessárias bem como popula tal array com threads leitoras e escritora de forma aleatória, seguindo as proporções especificadas para os experimentos.

Para isso, a classe possui os seguintes atributos:

- private static int numeroLeitores = 100; // atributo estático que será utilizado para controlar a proporção de threads leitoras para os experimentos
- private static int numeroEscritores = 0; // atributo estático que será utilizado para controlar a proporção de threads escritoras para os experimentos
- public static float mediaTempo; // atributo estático que será utilizado para calcular o tempo médio de execução para cada proporção de threads leitoras e escritoras
- private Thread[] threads; // array que armazena as 100 threads que serão utilizadas nos experimentos.
- private Database database; // instância da classe que armazena o ArrayList de palavras(região crítica).

Por fim, a classe possui os seguintes métodos:

- private int buscaPosicao(ThreadLocalRandom generator); // método responsável por buscar uma posição aleatória no array de threads que ainda não foi preenchida.
- private void populaObjetoThreads(int qntdLeitores, int qntdEscritores); // método responsável por popular o array de threads com instâncias de threads leitoras ou escritoras de forma aleatória e de acordo uma certa proporção para os experimentos.
- private void executaThreads(); // método responsável por executar cada thread armazenada no array de threads
- public void run(); // por estender a classe Thread do Java, a classe implementa o método run() que, no escopo desse projeto, realiza a medição dos tempos de execução dos experimentos.

3. Execução do projeto

Para executar o projeto de forma correta, realize os seguintes passos:

1. Descompacte a pasta 11207630.zip;
2. Mude para o diretório desejado (com_readers_writers ou sem_readers_writers);
3. Com o terminal aberto no diretório escolhido que contém o arquivo Sistema.java, execute os seguintes comandos:

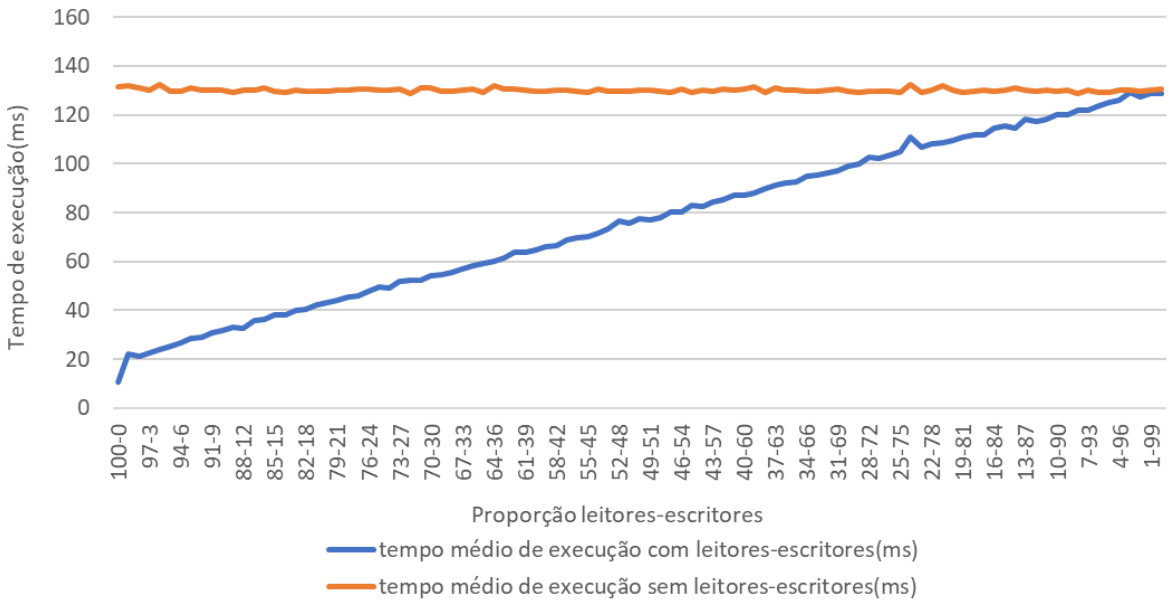
   javac Main.java

   java Main

4. Análises dos experimentos

Os resultados dos experimentos podem ser consultados na seção 5 em que é possível analisar os valores exatos dos tempos médios de execução para cada proporção com e sem leitores-escritores.

Para uma representação mais visual dos valores obtidos, gerou-se o seguinte gráfico:



A partir da análise do gráfico, observa-se que o tempo médio de execução sem leitores e escritores na solução mantém-se estável com variações sutis para todas as proporções de leitores-escritores testadas nos experimentos. Tal fato era esperado, uma vez que, enquanto acessam a região crítica, tanto leitores como escritores a mantém bloqueada, evitando o acesso concorrente à mesma.

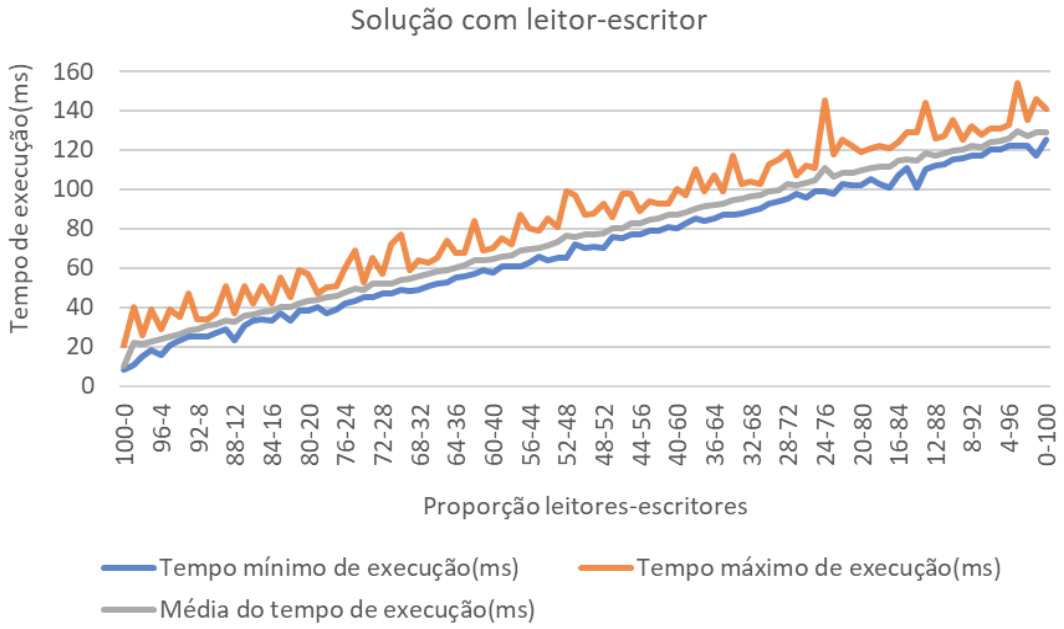
Em contrapartida, o mesmo não ocorre nos testes realizados com leitores e escritores na solução. Pela análise do gráfico, observa-se que a proporção de leitores-escritores influencia diretamente no tempo de execução, ou seja, à medida que a quantidade de escritores aumenta, a execução é mais lenta, pois eles não permitem que múltiplos leitores possam acessar concorrentemente os dados da região crítica. Assim, a partir da proporção de 14 leitores e 86 escritores (14-86), notou-se que o tempo de execução com leitores-escritores na solução(114,42 ms) tende a igualar o tempo de execução sem eles na solução(131,2 ms). A tabela de resultados apresentada na seção 5 evidencia esse fato.

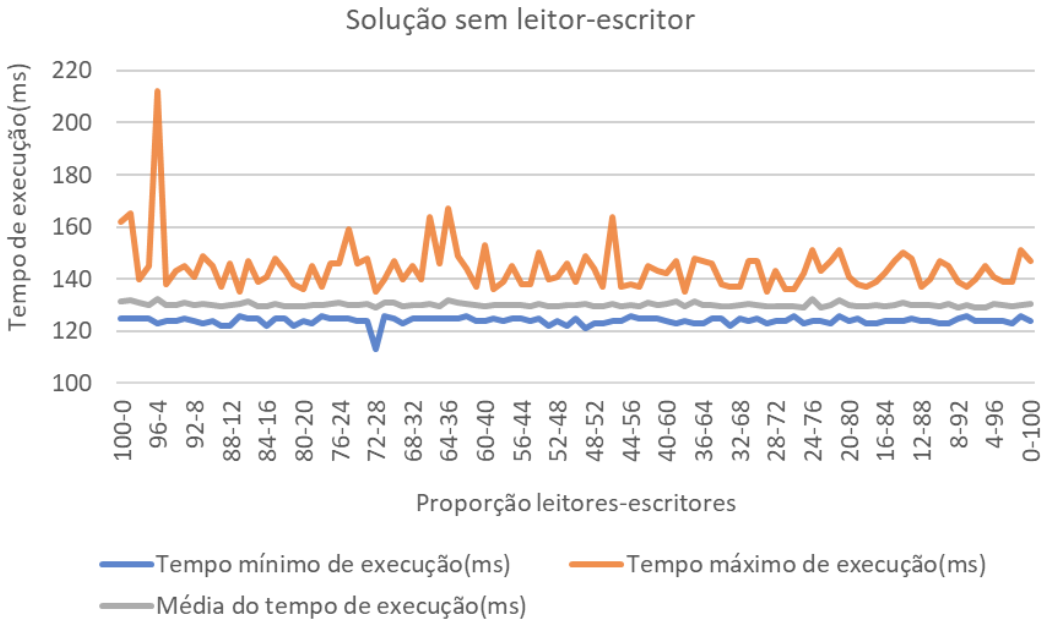
A tabela abaixo mostra a perda de performance da solução com leitor-escritor à medida que a quantidade de escritores aumenta:

Proporção leitor-escritor	Perda de performance (%)
100-0 para 85-15	263%
85-15 para 70-30	42%
70-30 para 55-45	30%
55-45 para 30-70	41%
30-70 para 15-85	16%
15-85 para 0-100	12%

Portanto, em um sistema que utilize majoritariamente operações de escrita, de forma que a proporção leitor-escritor seja superior a 14-86, é factível que a implementação sem a utilização da solução com leitores e escritores seja mais vantajosa por ser fácil implementação.

Por fim, para avaliar a presença de outliers nos tempos medidos, foi registrado o tempo mínimo e máximo atingido durante as 50 iterações para cada proporção de leitor-escritor em ambas as soluções. Deste modo, os seguintes gráficos foram criados:





Como observado pela análise dos gráficos acima, a presença de outliers nos tempos medidos para o cálculo da média é mínima, o que atesta a precisão dos valores calculados para as médias dos tempos de execução.

5. Resultados obtidos

Especificação do computador utilizado para executar os experimentos:

```
Processador: Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz (8 CPUs), ~1.2GHz
    Memória: 8192MB RAM
```

Proporção leitores-escritores	Tempo médio de execução com leitores-escritores(ms)	Tempo médio de execução sem leitores-escritores(ms)
100-0	10,44	131,58
99-1	21,98	132,08
98-2	21,1	131,06
97-3	22,48	129,96
96-4	23,72	132,32
95-5	25,44	129,92
94-6	26,56	129,8
93-7	28,4	130,96
92-8	29,08	130,1

91-9	30,7	130,32
90-10	31,56	129,98
89-11	33,24	129,36
88-12	32,66	130,18
87-13	35,7	130,34
86-14	36,44	131,3
85-15	37,92	129,56
84-16	38,24	129,42
83-17	40,1	130,32
82-18	40,26	129,54
81-19	42,22	129,78
80-20	43,18	129,7
79-21	43,86	130,16
78-22	45,46	130,08
77-23	45,68	130,42
76-24	47,72	130,76
75-25	49,52	130,16
74-26	49,16	129,98
73-27	51,78	130,7
72-28	52,2	128,88
71-29	52,16	131,2
70-30	53,98	130,86
69-31	54,42	129,76
68-32	55,68	129,84
67-33	57,1	130,08
66-34	58,14	130,58

65-35	59,02	129,4
64-36	60,14	132
63-37	61,34	130,74
62-38	63,78	130,48
61-39	63,82	130
60-40	64,76	129,78
59-41	66,08	129,86
58-42	66,56	130,1
57-43	68,66	130,26
56-44	69,6	129,92
55-45	70,22	129,42
54-46	71,62	130,58
53-47	73,42	129,72
52-48	76,64	129,62
51-49	75,56	129,88
50-50	77,34	130,12
49-51	77,06	130,36
48-52	78,02	129,72
47-53	80,22	129,36
46-54	80,38	130,54
45-55	82,82	129,4
44-56	82,66	130,16
43-57	84,34	129,56
42-58	85,28	130,76
41-59	87,24	130,22
40-60	86,94	130,64

39-61	88,14	131,34
38-62	90	129,4
37-63	91,46	131,3
36-64	92,2	130,02
35-65	92,46	129,94
34-66	94,84	129,7
33-67	95,46	129,62
32-68	96,44	130,18
31-69	97,2	130,48
30-70	99,2	129,84
29-71	99,72	129,36
28-72	102,76	129,64
27-73	102,04	129,58
26-74	103,5	129,58
25-75	104,8	129,14
24-76	110,78	132,52
23-77	106,6	129,18
22-78	108,26	130,1
21-79	108,64	131,9
20-80	109,4	130
19-81	111,02	129,38
18-82	111,62	129,7
17-83	111,62	129,98
16-84	114,42	129,6
15-85	115,42	130,06
14-86	114,4	131,2

13-87	118,14	130,06
12-88	117,46	129,9
11-89	118,12	129,96
10-90	119,96	129,78
9-91	120,3	130,36
8-92	122	128,88
7-93	121,82	129,96
6-94	123,84	129,3
5-95	124,9	129,32
4-96	125,82	130,32
3-97	129,36	130,22
2-98	127,36	129,48
1-99	128,82	130,26
0-100	128,88	130,68