UNIVERSIDADE DE SÃO PAULO
SISTEMAS DE INFORMAÇÃO

GABRIEL BERNARDINI SCHMIDT - 12873188
LUCAS SARTOR CHAUVIN - 11796718
TAMYRIS AYUMI NASCIMENTO ONODA - 12731401

ACH2044 - SISTEMAS OPERACIONAIS

RELATÓRIO EXERCÍCIO-PROGRAMA 2

SÃO PAULO
2022

1. Solução adotada para o problema de Leitores e Escritores

A solução adotada para o problema de leitores e escritores pode ser conferida no seguinte link. O mesmo código e ideia de resolução foram utilizados neste projeto.

Essa solução permite que várias threads concorrentes leiam e escrevam na região crítica ao mesmo tempo, desde que não haja nenhuma thread escrevendo. Além disso, as threads de leitura têm prioridade sobre as threads de escrita, ou seja, as threads de escrita ficam "em espera" até que todas as threads de leitura tenham sido finalizadas.

2. Estrutura e definição de classes do projeto

Tanto a solução com leitor e escritor quanto a sem leitor e escritor foram desenvolvidas no projeto para usar as mesmas classes, as quais possuem os mesmos atributos e métodos - o que muda é a implementação e código nos métodos. Como decisão de projeto, foram implementadas as seguintes classes:

2.1 Database

A classe Database é responsável por armazenar as palavras lidas do arquivo "bd.txt" em um ArrayList. Como a estrutura é uma região crítica que pode ser acessada por diferentes threads concorrentemente, a classe implementa um mecanismo de controle de acesso. O tipo de mecanismo implementado depende da solução escolhida (com leitor e escritor ou sem). A classe também é responsável por ler o arquivo e inicializar e carregar os dados no ArrayList.

A classe possui os atributos "list" (um ArrayList que armazena as palavras) e "leitores" (que armazena a quantidade de leitores que acessam a região crítica no momento). Ela também possui os métodos "carregaEstruturaRAM" (que lê o arquivo e carrega os dados no ArrayList), "read" (que permite a leitura dos dados do ArrayList) e "write" (que permite a atualização dos dados do ArrayList por apenas uma thread por vez).

2.2 Writer

A classe Writer (de escritor) é responsável por representar uma thread de escrita no ArrayList de palavras (região crítica).

Ela possui o atributo "database" que armazena uma instância da classe que armazena o ArrayList de palavras. Como a classe estende a classe Thread do Java, ela possui apenas o método "run" que faz uma chamada ao método "read" da classe Database.

2.3 Reader

A classe Reader (de leitor) é responsável por representar uma thread de leitura dos dados armazenados no ArrayList de palavras (região crítica).

Ela possui o atributo "database" que armazena uma instância da classe que armazena o ArrayList de palavras. Como a classe estende a classe Thread do Java, ela possui apenas o método "run" que faz uma chamada ao método "write" da classe Database.

2.4 Sistema

A classe Sistema é responsável por representar a thread principal que executará todas as threads leitoras e escritoras necessárias para os experimentos. Ela armazena um array com todas as 100 threads necessárias e o popula com threads leitoras e escritoras de forma aleatória, seguindo as proporções especificadas para os experimentos.

Para isso, a classe possui os atributos "numeroLeitores" e "numeroEscritores" (que são usados para controlar a proporção de threads leitoras e escritoras), "mediaTempo" (que é usado para calcular o tempo médio de execução para cada proporção de threads leitoras e escritoras), "threads" (um array que armazena as threads) e "database" (uma instância da classe que armazena o ArrayList de palavras).

Ela também possui os métodos "buscaPosicao", que busca uma posição aleatória no array de threads que ainda não foi preenchida, "populaObjetoThreads", que popula o array de threads com instâncias de threads leitoras ou escritoras de forma aleatória e de acordo com uma certa proporção para os experimentos, "executaThreads", que executa cada thread armazenada no array de threads, e "run", que por estender a classe Thread do Java, implementa o método "run" que, nesse projeto, realiza a medição dos tempos de execução dos experimentos.

3. Execução do Projeto

Para a correta execução das soluções, siga os seguintes passos:

1. Descompacte o arquivo .zip;
2. Acesse, em um terminal, o diretório do projeto;
3. a solução desejada MUDAR ISSO (**semLeitorEscritor** ou **comLeitorEscritor**;
4. Compile a classe com o comando:
    a. Solução sem leitor-escritor: **javac semLeitorEscritor/Main.java**
    b. Solução com leitor-escritor: **javac comLeitorEscritor/Main.java**
5. Execute a classe compilada com o comando:
    a. Solução sem leitor-escritor: **java semLeitorEscritor/Main.java**
    b. Solução com leitor-escritor: **java comLeitorEscritor/Main.java**

6. O resultado da execução será apresentado na tela do terminal.

4. Resultados

Rodando as duas soluções (com leitor e escritor e sem leitor e escritor) obteve-se os seguintes resultados:

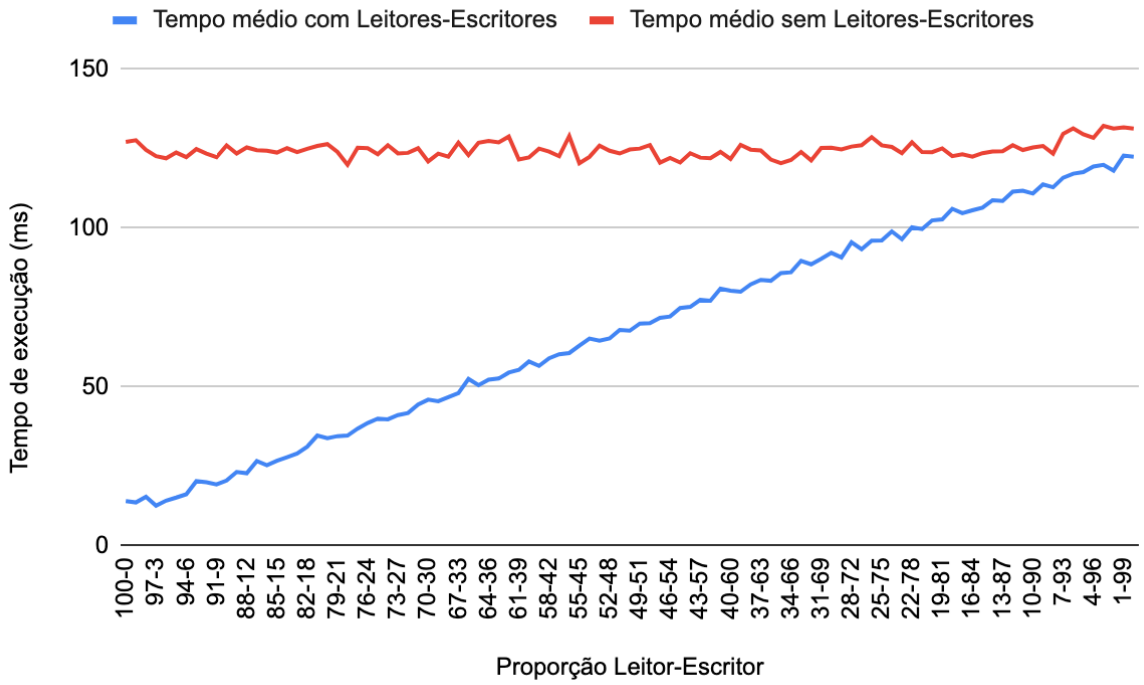
Proporção Leitor-Escritor	Tempo médio com Leitores-Escritores	Tempo médio sem Leitores-Escritores
100-0	14,02	127,14
99-1	13,6	127,66
98-2	15,38	124,64
97-3	12,56	122,68
96-4	14,16	121,98
95-5	15,12	123,80
94-6	16,18	122,34
93-7	20,3	124,88
92-8	19,94	123,48
91-9	19,26	122,32
90-10	20,5	126,06
89-11	23,2	123,50
88-12	22,8	125,40
87-13	26,62	124,54
86-14	25,34	124,38
85-15	26,72	123,80
84-16	27,84	125,16
83-17	29,02	123,92
82-18	31,14	124,98
81-19	34,66	125,88
80-20	33,86	126,46
79-21	34,48	124,00
78-22	34,66	119,92
77-23	36,84	125,30
76-24	38,62	125,14
75-25	40,00	123,28
74-26	39,78	126,10
73-27	41,12	123,54
72-28	41,76	123,70

71-29	44,40	125,14
70-30	45,96	121,00
69-31	45,46	123,46
68-32	46,74	122,52
67-33	48,04	126,92
66-34	52,48	123,00
65-35	50,50	126,90
64-36	52,24	127,44
63-37	52,62	127,00
62-38	54,50	128,84
61-39	55,34	121,62
60-40	58,00	122,24
59-41	56,60	125,04
58-42	59,02	124,10
57-43	60,26	122,64
56-44	60,66	128,98
55-45	63,00	120,44
54-46	65,22	122,44
53-47	64,54	125,98
52-48	65,24	124,36
51-49	67,94	123,50
50-50	67,68	124,76
49-51	69,94	125,04
48-52	70,04	126,14
47-53	71,72	120,62
46-54	72,16	122,12
45-55	74,84	120,66
44-56	75,18	123,58
43-57	77,42	122,20
42-58	77,12	122,00
41-59	80,94	124,00
40-60	80,30	121,80
39-61	79,96	126,22
38-62	82,28	124,72
37-63	83,70	124,46
36-64	83,42	121,60
35-65	85,82	120,48

34-66	86,08	121,46
33-67	89,70	123,98
32-68	88,60	121,32
31-69	90,32	125,26
30-70	92,20	125,34
29-71	90,76	124,82
28-72	95,52	125,68
27-73	93,36	126,10
26-74	96,04	128,66
25-75	96,12	126,08
24-76	98,96	125,58
23-77	96,48	123,60
22-78	100,26	127,06
21-79	99,70	123,94
20-80	102,42	123,92
19-81	102,76	125,12
18-82	106,08	122,66
17-83	104,72	123,22
16-84	105,60	122,50
15-85	106,44	123,60
14-86	108,80	124,12
13-87	108,56	124,22
12-88	111,46	126,14
11-89	111,78	124,60
10-90	110,90	125,40
9-91	113,78	125,88
8-92	112,86	123,46
7-93	115,86	129,70
6-94	117,14	131,42
5-95	117,62	129,54
4-96	119,40	128,42
3-97	119,92	132,20
2-98	118,10	131,34
1-99	122,86	131,74
0-100	122,48	131,32

5. Análise e conclusão dos resultados

Através dos resultados anteriores, foi possível gerar o seguinte gráfico:



A análise do gráfico mostra que o tempo médio de execução sem leitores e escritores na solução se mantém estável com variações sutis para todas as proporções de leitores-escritores testadas nos experimentos. Isso era esperado, pois enquanto acessam a região crítica, tanto leitores quanto escritores a mantêm bloqueada, evitando o acesso concorrente.

No entanto, isso não ocorre nos testes realizados com leitores e escritores na solução. Através do gráfico, é possível observar que a proporção de leitores-escritores influencia diretamente no tempo de execução: quanto mais escritores, mais lenta a execução, pois eles não permitem que múltiplos leitores acessem os dados da região crítica de forma concorrente. Portanto, a partir da proporção 13-87 (13 leitores e 87 escritores), o tempo de execução com leitores-escritores na solução (108,56 ms) tende a igualar o tempo de execução sem eles na solução (124,22 ms). A tabela de resultados apresentada anteriormente mostra justamente isso.

Portanto, em um sistema que utilize principalmente operações de escrita, com uma proporção de leitor-escritor superior a 14-86, é possível que a implementação sem a solução com leitores e escritores seja mais vantajosa por ser mais fácil de implementar.

Para verificar se há outliers nos tempos medidos, foi registrado o tempo mínimo e máximo atingido durante as 50 iterações para cada proporção de leitor-escritor nas duas soluções. Isso permitiu a criação dos seguintes gráficos, que mostram que a presença de

outliers é mínima, o que confirma a precisão dos valores calculados para as médias dos tempos de execução.