Gestão Eficiente de Elevadores: Uma Abordagem Concorrente com FIFO e SCAN

Carlos Eduardo da Silva Santos - 119065432
Ian de Andrade Camargo - 118089205
Julia Deroci Lopes - 117257871

Relatório Final

Programação Concorrente (ICP-361) — 2024/2

1. Descrição do Problema Geral

O problema em questão envolve a simulação de um sistema de elevadores que gerencia requisições de passageiros em um edifício. O objetivo é otimizar o movimento dos elevadores para atender a essas requisições de forma eficiente, utilizando duas abordagens de escalonamento: FIFO (First-In-First-Out) e SCAN.

1.1. Funcionamento da Solução FIFO

Diferente de como cogitamos inicialmente, o FIFO não seria um modelo sequencial, já que usaremos os múltiplos elevadores como sendo cada um uma thread. Nele teremos a ordem de requisições sendo imputadas de forma a escolher o número total de requisições seguido pelas requisições desejadas, como isso os elevadores deverão seguir um padrão similar ao padrão leito/escritor, onde após ter todas as requisições escritas, os elevadores poderão retirar do buffer as mesmas em ordem de chegada, first in first out. Com isso o processo seguiria até concluir a última requisição.

- ***Entrada:*** *Número de requisições, Andares de origem e destino das requisições, que serão gerados pelos passageiros. Neste caso, todas as requisições são conhecidas e definidas previamente. Exemplo:*

```
Quantas requisições você deseja fazer? (Máximo 10): 3
Digite a 1ª requisição no formato 'origem,destino': 2,5
Requisição: Passageiro 1 solicitou do andar 2 para o andar 5
Digite a 2ª requisição no formato 'origem,destino': 5,7
```

Figure 1. Entrada do FIFO.

- ***Saída:*** *Mensagens indicando a posição do elevador, requisição do passageiro, entrada e saída do elevador. Exemplo:*

```
Elevador 2: Passageiro 1 entrou no elevador no andar 2 com destino ao andar 5
Elevador 1: No andar 3
Elevador 1: No andar 4
Elevador 1: No andar 5
Elevador 1: Passageiro 2 entrou no elevador no andar 5 com destino ao andar 7
```

Figure 2. Saída do FIFO.

1.2. Funcionamento da Solução SCAN

A implementação SCAN permite que múltiplos elevadores processem requisições simultaneamente. Esta abordagem minimiza o tempo de espera dos passageiros, já que mais de uma requisição pode ser atendida ao mesmo tempo, reduzindo a sobrecarga em um único elevador e melhorando a experiência do usuário.

- ***Entrada:*** *O Número de requisições, Andares de origem e destino das requisições são gerados de forma automática pela função make_request, usando números aleatórios para determinar o andar de origem e destino. Além disso, as requisições surgem de forma aleatória , ao contrário da FIFO, que todas as requisições são conhecidas previamente*
- ***Saída:*** *Mensagens indicando a posição do elevador, requisição do passageiro, entrada e saída do elevador. Similar a saída do FIFO, porém optamos por colocar cores para ajudar na visualização do funcionamento do algoritmo. Exemplo:*

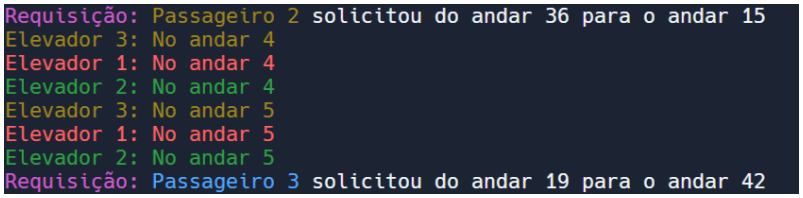

Figure 3. Saída do SCAN.

2. Projeto da Solução Concorrente

Divisão da Tarefa Principal: *A solução será dividida em múltiplas threads, onde cada thread representará um elevador. Cada elevador irá:*

- *Processar requisições em sua própria fila.*
- *Mover-se entre os andares conforme as requisições.*

2.1. FIFO

- ***Número Máximo de Passageiros por Elevador:*** *As requisições são tratadas individualmente, então o limite de passageiros é 1.*
- ***Número de Elevadores (Threads):*** *Utilizamos 2, 3 e 4 elevadores em nossos testes, configurados pela constante MAX_ELEVATORS.*
- ***Número de Andares:*** *Utilizamos 10 e 50 andares em nossos testes, configurados pela constante MAX_FLOORS*
- ***Número Máximo de Requisições:*** *Utilizamos 10 e 50 andares em nossos testes, configurados pela constante MAX_REQUEST*
- ***Movimentação:*** *O deslocamento entre os andares usa sleep(1) para simular o tempo necessário para o elevador se movimentar.*
- ***Embarque/Desemparque:*** *O embarque e desembarque é simulado utilizando sleep(4), para refletir a duração mais realista dessas ações.*
- ***Sincronização:*** *Utilizamos Mutex para proteger o acesso à fila centralizada request_queue. Apenas uma thread pode modificar a fila por vez. Além disso, utiliza variáveis de condição para coordenar o estado entre os elevadores e as requisições geradas.*

2.2. SCAN

- ***Número Máximo de Passageiros por Elevador:*** *Cada elevador pode transportar até 4 passageiros simultaneamente, definido pela constante MAX_PASSENGERS o início do código.*
- ***Número de Elevadores (Threads):*** *Utilizamos 2, 3 e 4 elevadores em nossos testes, configurados pela constante MAX_ELEVATORS.*
- ***Número de Andares:*** *Utilizamos 10 e 50 andares em nossos testes, configurados pela constante MAX_FLOORS*
- ***Número Máximo de Requisições:*** *Utilizamos 10 e 50 andares em nossos testes, configurados pela constante MAX_REQUEST*
- ***Movimentação:*** *O deslocamento entre os andares usa sleep(1) para simular o tempo necessário para o elevador se movimentar.*
- ***Embarque/Desemparque:*** *O embarque e desembarque é simulado utilizando sleep(4), para refletir a duração mais realista dessas ações.*
- ***Sincronização:*** *Cada andar possui um mutex individual. Isso garante que múltiplos elevadores não acessem as requisições do mesmo andar ao mesmo tempo. Além disso, protege o array floor_buffers e o contador request_counts de cada andar.*
- ***Temporizador de requisição:*** *Cada requisição poderá ocorrer no intervalo de 0 à 4 segundos, permitindo que as requisições possam ser mais espalhadas pelo processo, ou até mesmo serem simultânea, essa simulação é feita usando sleep(rand()%4+1).*

3. Testes de Corretude

Modificações na variação no número de requisições, para garantir que o gerenciamento da fila funcione adequadamente independente das requisições e elevadores utilizados. Para isso realizamos testes, variando a quantidade de requisições entre 3, 5, 7 por cinco vezes cada. Os testes foram feitos seguindo a ordem de testar o SCAN primeiro, onde seriam geradas requisições aleatórias, seguido da inserção dessas mesmas requisições no modelo FIFO, para que os testes tivessem a maior paridade possível.

É importante ressaltar que apesar de terem sido utilizadas as mesmas requisições para ambos os modelos, o teste como dito anteriormente não é totalmente parelho, visto as diferenças de como as requisições são chamadas no SCAN em comparação com o FIFO.

Abaixo as tabelas com os tempos de processamento obtidos nos testes:

Tempo de Execução (Processamento) em um Prédio de 10 andares - FIFO									
N° Elevadores (Threads)	2			3			4		
Requisições	3	5	7	3	5	7	3	5	7
1	58,777999	88,049381	98,886512	30,161919	61,137535	155,691248	34,276971	64,120105	75,843158
2	118,270543	74,651314	108,756536	27,562727	53,028199	70,000391	37,065556	65,018172	80,641560
3	101,177354	63,989955	115,227779	24,182495	53,363376	70,780225	35,004965	55,429486	80,425028
4	51,121862	134,533673	93,823068	28,070794	54,204198	70,874797	32,546352	65,153080	76,482090
5	68,030106	133,344793	89,213771	23,561350	64,836971	69,951872	29,493945	60,430221	74,447678
Média	79,475572 8	98,913823 2	101,181 5332	26,707857	57,314055 8	87,459 7066	33,677 5578	62,03 02128	77,5679 028

Figure 4. Tabela Tempo de Execução (FIFO 10 andares).

Tempo de Execução (Processamento) em um Prédio de 50 andares - FIFO			
N° Elevadores (Threads)	3		
Requisições	3	5	7
1	106,356039	194,359856	252,844906
2	123,066111	167,479778	229,172871
3	100,763395	111,527841	262,983515
4	78,158505	141,876924	198,438381
5	73,487243	113,358216	191,542927
Média	96,3662586	145,720523	226,99652

Figure 5. Tabela Tempo de Execução (FIFO 50 andares).

Tempo de Execução (Processamento) em um Prédio de 10 andares - SCAN									
N° Elevadores (Threads)	2			3			4		
Requisições	3	5	7	3	5	7	3	5	7
1	18,002554	33,011471	40,023830	21,002656	25,012876	53,024441	18,007643	29,003513	36,004219
2	24,003545	26,004743	37,010534	24,005091	28,004470	47,033446	19,004449	37,007479	38,013205
3	22,039290	22,012339	37,004033	25,004116	24,003174	30,008201	15,005040	35,003590	32,004799
4	15,003806	29,007610	33,006334	17,003081	36,024229	39,005899	22,007652	26,002358	33,022109
5	19,002141	34,008730	28,005196	23,002152	39,008926	33,004623	20,003315	25,008396	43,007111
Média	19,610267 2	28,808978 6	35,009985 4	22,003419 2	30,410735	40,415322	18,805 6198	30,40 50672	36,410 2886

Figure 6. Tabela Tempo de Execução (SCAN 10 andares).

Tempo de Execução (Processamento) em um Prédio de 50 andares - SCAN			
Nº Elevadores (Threads)	3		
Requisições	3	5	7
1	96,017006	96,016957	105,019178
2	94,011364	101,022296	108,024187
3	65,022953	75,008774	110,016417
4	57,010273	97,022542	109,041125
5	54,010880	76,011045	109,025091
Média	73,2144952	89,0163228	108,2251996

Figure 7. Tabela Tempo de Execução (SCAN 50 andares).

4. Avaliação de Desempenho

Para começarmos a avaliar o desempenho dos modelos, nós utilizamos o método de cálculo de aceleração, e eficiência que foi aprendida durante o período, para fazermos a comparação entre os modelo FIFO e SCAN. O código para cálculo de tempo, foi feito utilizando a biblioteca "timer.h" também utilizada durante o período em um dos laboratórios.

Com esses resultados obtidos, geramos gráficos que deixassem claros quais foram os ganhos, e qual modelo melhor servia ao nosso propósito. Como dito anteriormente, foram feitos testes para 2, 3, 4 threads/elevadores, onde cada thread será testada cinco vezes, com requisições variando entre 3, 5, 7.

4.1. Aceleração/eficiência SCAN

$$A(n,t) = \frac{T_{FIFO}(n,t)}{T_{SCAN}(n,t)}$$

$$E(n,t) = \frac{A(n,t)}{t}$$

4.2. Aceleração/eficiência FIFO

$$A(n,t) = \frac{T_{SCAN}(n,t)}{T_{FIFO}(n,t)}$$

$$E(n,t) = \frac{A(n,t)}{t}$$

4.3. Especificações da Máquina

A máquina utilizada na realização dos testes possui as seguintes configurações:

- *Sistema Operacional: Linux Mint 21 Cinnamon*
- *Versão do Cinnamon: 5.4.12*
- *Kernel do Linux: 5.15.0-101-generic*
- *Processador: 13ª geração Intel® Core™ i5-13500, com 14 núcleos*

5. Discussão dos Resultados

Nesta seção, discutem-se os resultados obtidos a partir das análises realizadas ao longo deste trabalho.

5.1. Análise de Aceleração

No sistema FIFO, as requisições são atendidas na ordem em que chegam, sem priorizar nenhum elevador sobre outro. A aceleração aqui é mais difícil de observar, pois o sistema FIFO pode levar a um desempenho subótimo quando o número de requisições aumenta e se tem um número muito grande de andares, já que as requisições podem ser atendidas de forma mais lenta, especialmente quando há congestionamento entre as threads/elevadores.

Analisando os valores do gráfico 9, podemos observar uma diminuição na aceleração conforme o número de requisições aumenta. Isso indica que, para um número maior de requisições e andares, o sistema FIFO começa a perder eficiência, provavelmente devido ao aumento da sobrecarga de controle e à possibilidade de congestionamento das requisições em um número limitado de threads.

Porém a aceleração aumenta ligeiramente à medida que o número de requisições cresce quando se tem poucos andares, como mostra o gráfico 8,, mas os valores são bastante baixos, indicando que o sistema FIFO não se beneficia muito da adição de mais threads.

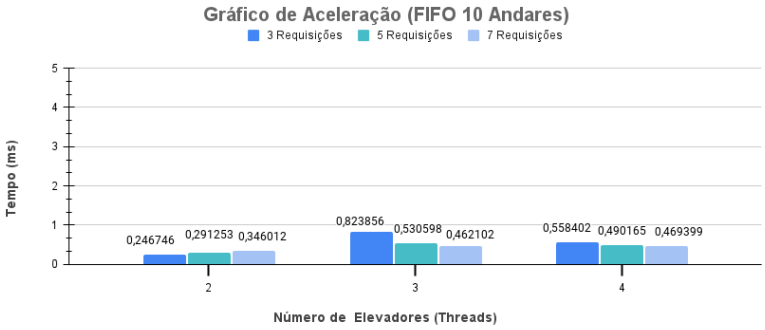

Figure 8. Gráfico de Aceleração (FIFO 10 andares).

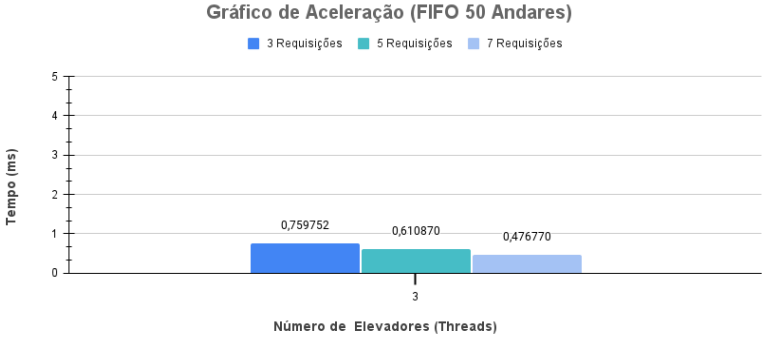

Figure 9. Gráfico de Aceleração (FIFO 50 andares).

O sistema SCAN permite que múltiplos elevadores processem requisições simultaneamente, o que pode resultar em um comportamento mais eficiente, pois as requisições são atendidas de forma mais dinâmica.

Observamos os gráficos 10 e 11 que a aceleração aumenta com o número de requisições e o número de andares. Esse comportamento é característico de sistemas com maior controle sobre a direção do movimento e onde a sobrecarga de gerenciar as requisições é mais bem distribuída. O sistema SCAN parece se beneficiar mais da adição de requisições e andares, pois a movimentação ordenada dos elevadores permite que o desempenho melhore com o aumento de andares e requisições.

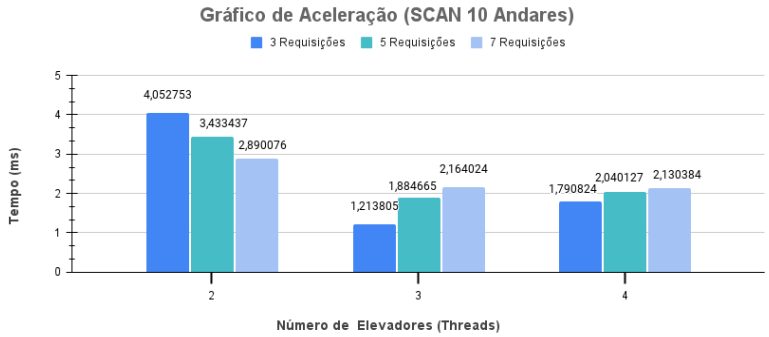

Figure 10. Gráfico de Aceleração (SCAN 10 andares).

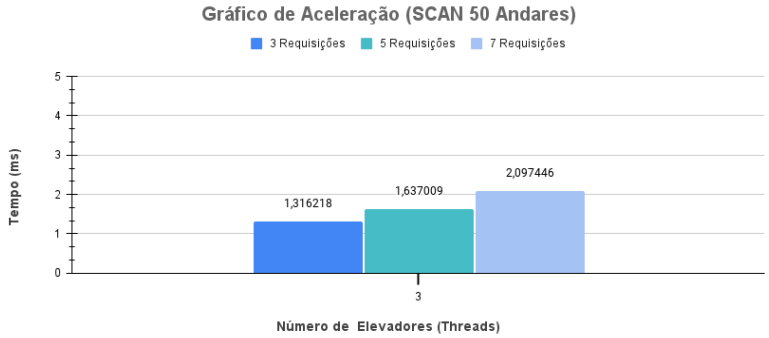

Figure 11. Gráfico de Aceleração (SCAN 50 andares).

5.2. Análise de Eficiência

No sistema FIFO a eficiência diminui à medida que o número de requisições aumenta. Isso indica que o sistema FIFO não está conseguindo aproveitar bem seus recursos com o aumento das requisições, provavelmente devido a uma sobrecarga na gestão das requisições em fila, que pode causar lentidão no processamento, como é possível observar nos gráficos 12 e 13

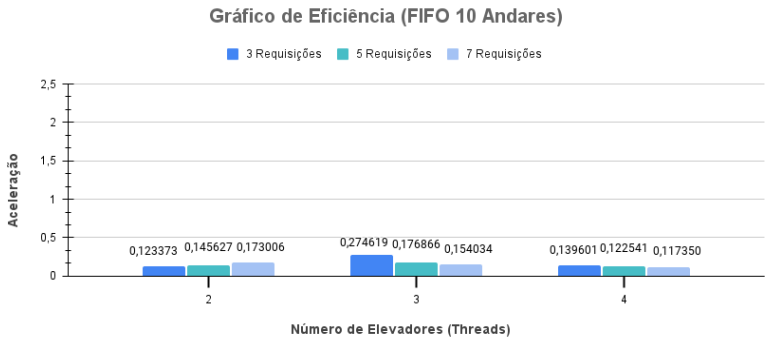

Figure 12. Gráfico de Eficiência (FIFO 10 andares).

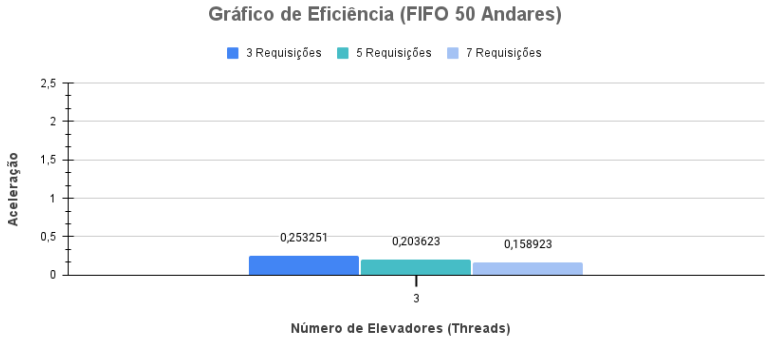

Figure 13. Gráfico de Eficiência (FIFO 50 andares).

No sistema SCAN a eficiência aumenta à medida que o número de requisições e andares aumentam. Isso sugere que o sistema SCAN está se tornando mais eficiente à medida que mais requisições são processadas e se tem um maior número de andares, aproveitando melhor os recursos, como é possível observar nos gráficos 14 e 15.

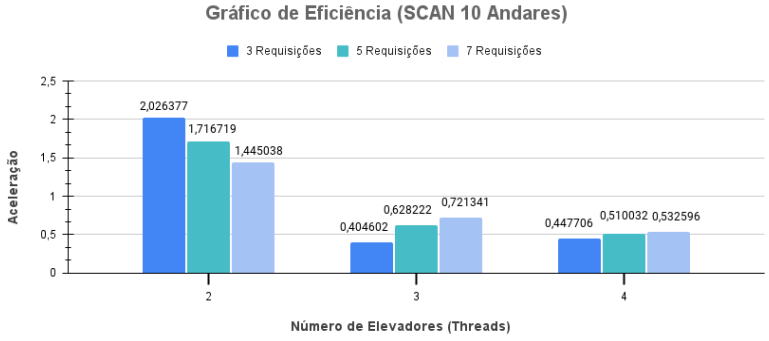

Figure 14. Gráfico de Eficiência (SCAN 10 andares).

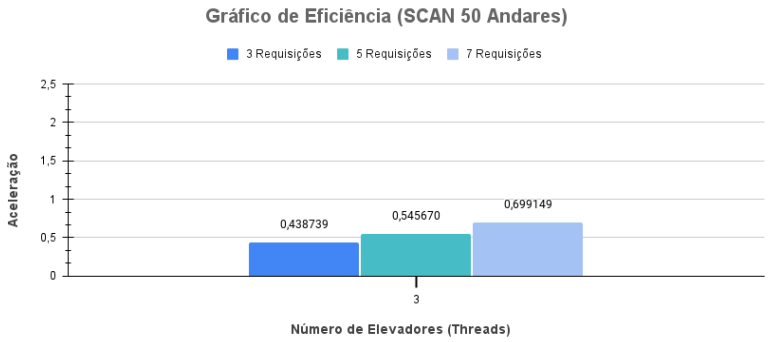

Figure 15. Gráfico de Eficiência (SCAN 50 andares).

6. Referências

Nosso trabalho toma como referência os livros [1], [2] e o vídeo mostrando a simulação de elevadores utilizando multithreads [3]. Além do código desenvolvido e disponibilizado por nós no repositório [4].

[1] Peter Pacheco. An Introduction to Parallel Programming. *Elsevier, 2011. s.l.*

[2] Randal E. Bryant and David R. O'Hallaron. Computer systems: a programmer's perspective. *Pearson, Boston, 3rd edition, 2016.*

[3] YouTube. Vídeo sobre elevadores com programação paralela. Disponível em: `https://www.youtube.com/watch?v=xKjRKND1ABg`*. Acesso em: 23 out. 2024.*

[4] Julia Deroci Carlos Eduardo and Ian Camargo. Elevadores. Disponível em: `https://github.com/iancbr/Elevadores/tree/main`*, 2024. Acessado em: 27 nov. 2024.*