Universidade de São Paulo - ICMC - BCC
SSC0903 - Computação de Alto Desempenho (2022/2)
Primeiro Trabalho Prático (TB1) - Resolução em Grupo

Turma: A

Grupo 7

Nomes dos integrantes deste grupo que resolveram o trabalho:

11796893 Luiz Fernando Rabelo
11031663 Marcus Vinicius Medeiros Pará
4769989 Rafael Corona
11795526 Tulio Santana Ramos

Resposta para Q01:

Particionamento

Na etapa de particionamento a primeira tarefa é a geração aleatória e sequencial das notas. Sendo C o número de cidades por região e R o número de regiões, na sequência são geradas 4*C*R tarefas, o que corresponde a quatro tarefas por cidade, que operarão sobre as notas de cada aluno da cidade, buscando:

- A menor nota;
- A maior nota;
- Construir a contagem de frequência das notas;
- A soma das notas;

A contagem da cidade será utilizada na tarefa subsequente de calcular a mediana da cidade. A partir da soma da cidade, se desencadearão, obrigatoriamente nesta ordem, as tarefas responsáveis pelo cálculo da média das notas e do desvio padrão. Ao finalizarem, escreverão seus resultados em vetores, seja para realizar o print sequencial dos resultados.

Com exceção da tarefa responsável pelo cálculo da mediana todas serão concluídas a partir de suas respectivas reduções.

Neste ponto, os cálculos por cidade estarão prontos e a tarefa de descobrir a cidade de maior média pode iniciar e printar o resultado após seu término. Além disso, serão desencadeadas estas tarefas, que buscarão, a nível de região:

- A menor nota;
- A maior nota;
- Somar as contagens de frequência das notas das cidades;
- A soma das notas;

Estas tarefas serão realizadas a partir de reduções dos vetores de resultados das cidades. Similarmente ao nível das cidades, serão feitas então as tarefas responsáveis pela mediana e desvio padrão e poderá ser calculada a melhor região.

Por fim, a fim de se obter as mesmas métricas, a nível nacional, mais uma vez serão realizadas reduções sobre os vetores resultados das tarefas anteriores, com exceção do desvio padrão, que sempre é uma redução sobre o vetor de notas, e a mediana, que é o cálculo do elemento central do vetor de contagens.

Dessa forma, os resultados podem então serem printados sequencialmente;

Grafo de dependências:

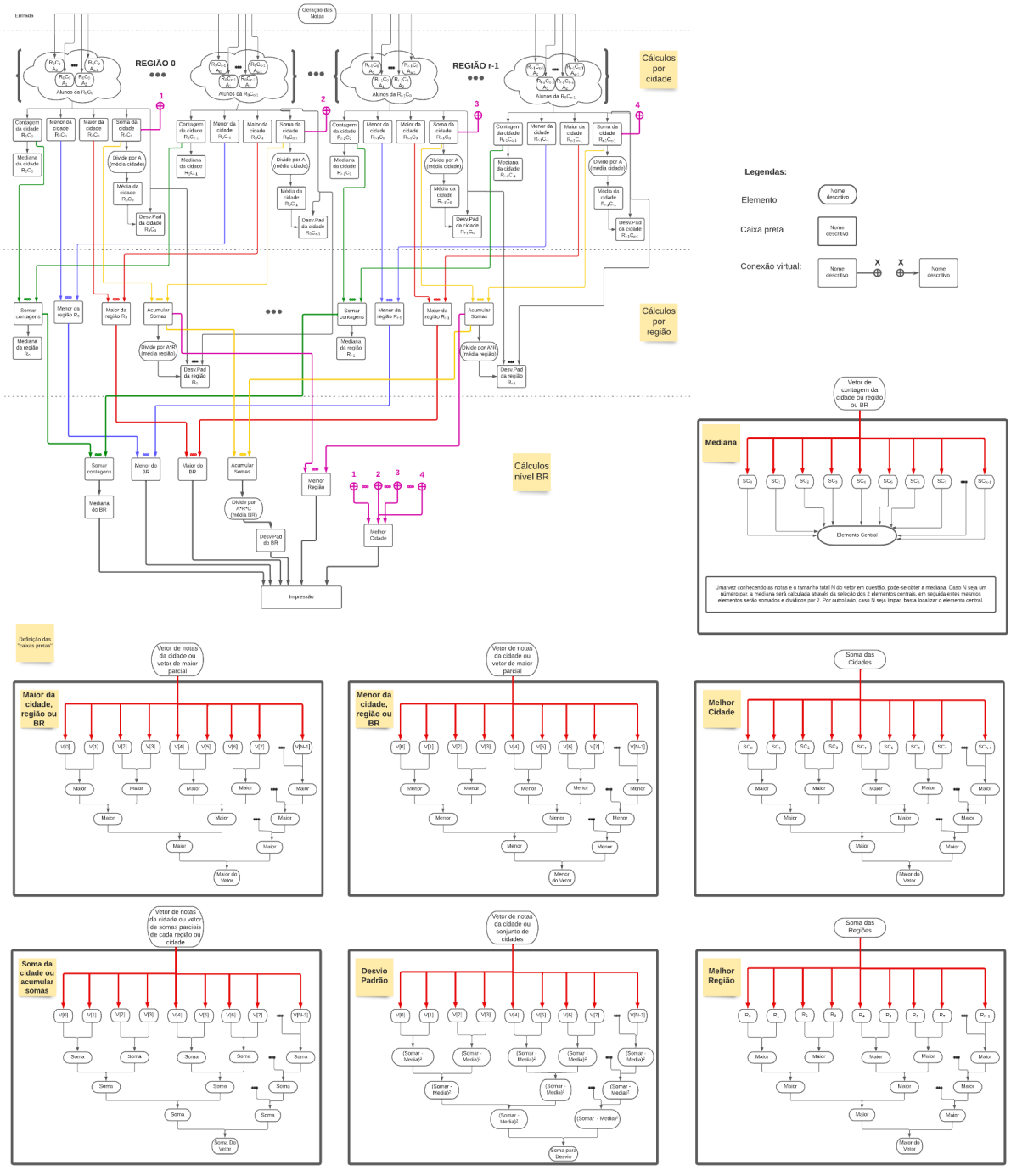


Figura 1 - Grafo de Dependências

Observação: O grafo de dependências também foi anexado como “dependência.svg” caso exista alguma dificuldade de visualização.

Comunicação

O fluxo de dados neste problema é bem semelhante ao grafo de dependências, com apenas algumas novas conexões que representam o broadcast de determinadas variáveis.

Os dados recebidos na entrada, assim como as notas geradas a partir deles, são enviados para suas tarefas correspondentes que operam sobre cada cidade. O resultado de cada uma das tarefas, se utilizado futuramente, é então escrito em sua respectiva posição no vetor que será enviado a tarefa correspondente, que opera agora em nível regional. Analogamente, o nível nacional também opera com reduções sobre os resultados do nível anterior e escreve o novo output em variáveis concentradoras dos resultados, que serão posteriormente printadas. Note que sempre ocorre um broadcast das notas e das médias para as tarefas de cálculo do desvio padrão.

Grafo de Comunicação:

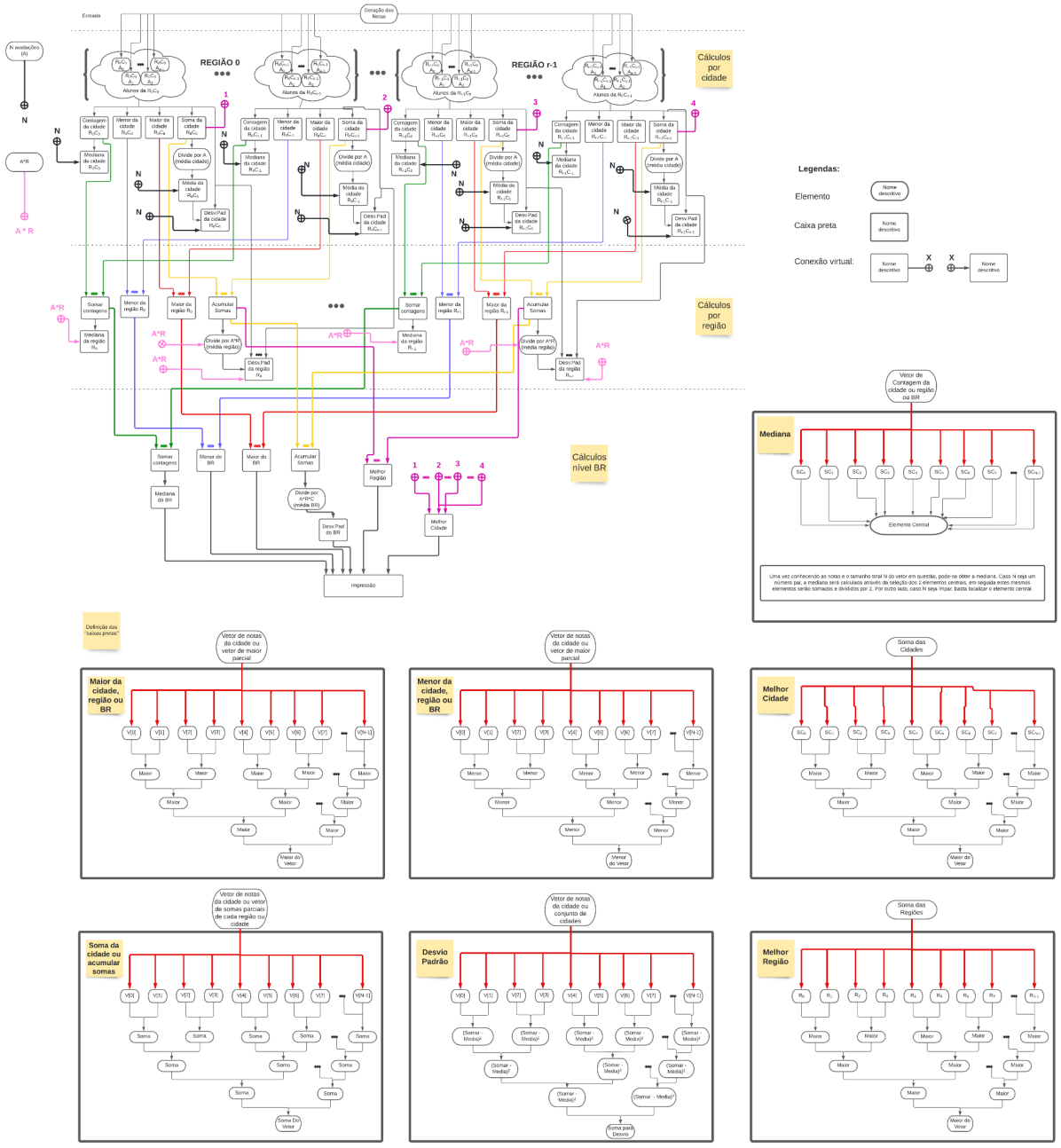


Figura 2 - Grafo de Comunicação

Observação: O grafo de comunicação também foi anexado como “comunicação.svg” caso exista alguma dificuldade de visualização.

Aglomeração

Dada a plataforma alvo deste algoritmo, máquina MIMD com 4 núcleos, devemos agrupar as 4*C*R tarefas da seguinte em 4 processos.

Cada um deles deverá realizar os cálculos referentes a C*R/4 cidades e uma cidade a mais será adicionada às Q primeiras tarefas, onde Q é o resto da divisão.

Quando todos finalizarem sua carga, cada um será responsável por R/4 regiões e uma região a mais será adicionada às Q primeiras tarefas, onde Q é o resto da divisão.

Por fim, restará o cálculo de 6 seções que podem ser realizadas simultaneamente:

- Maior nacional
- Menor nacional
- Melhor região
- Melhor cidade
- Soma + média + desvio padrão nacionais

É esperado que a última seja a mais longa e portanto deve ser exclusiva de uma tarefa. As demais podem englobar uma seção cada, sendo que uma tarefa será responsável por duas seções.

Mapeamento

O mapeamento é dinâmico. Cada elemento de processamento deve receber um processo (thread). Essa gerência caberá ao Sistema Operacional da máquina. Espera-se a distribuição boa distribuição (aproximadamente uniforme) dos processos nos núcleos disponíveis.

Resposta para a Q02:

Essa questão foi respondida através da implementação dos códigos sequencial e paralelo. Respectivamente, os programas se encontram nas pastas **codigo-sequencial** e **codigo-paralelo** enviadas no mesmo arquivo **.zip** deste documento. Estes códigos são divididos entre seus programas principais (que contém a função “main”), **studentsseq.c** e **studentspar.c**, e seus arquivos de funções, **funcoes-seq.h** e **funcoes-par.h**. Para a execução dos programas, basta o uso do **Makefile** externo às pastas mencionadas anteriormente. Os comandos disponíveis pelo Makefile são:

- *make*: compila o sequencial e o paralelo com as flags padrão para execução;
- *make debug*: compila o sequencial e o paralelo com o caso de teste apresentado na especificação da atividade;
- *make time*: compila o sequencial e o paralelo para a coleta de tempo de execução;
- *make sequencial*: executa o sequencial;
- *make paralelo*: executa o paralelo.

Para alterar a entrada do algoritmo, deve-se alterar o arquivo “entrada.txt”, localizado na raíz do arquivo zip.

Quando compilados com make time, o tempo de execução coletado é escrito em uma arquivo cujo nome é definido por:

- Código sequencial:
  - seq<numero_de_execucoes>RCA<regioes>-<cidades>-<alunos>
- Código paralelo
  - par-<numero_de_threads>threadRCA<regioes>-<cidades>-<alunos>

Resposta para a Q03:

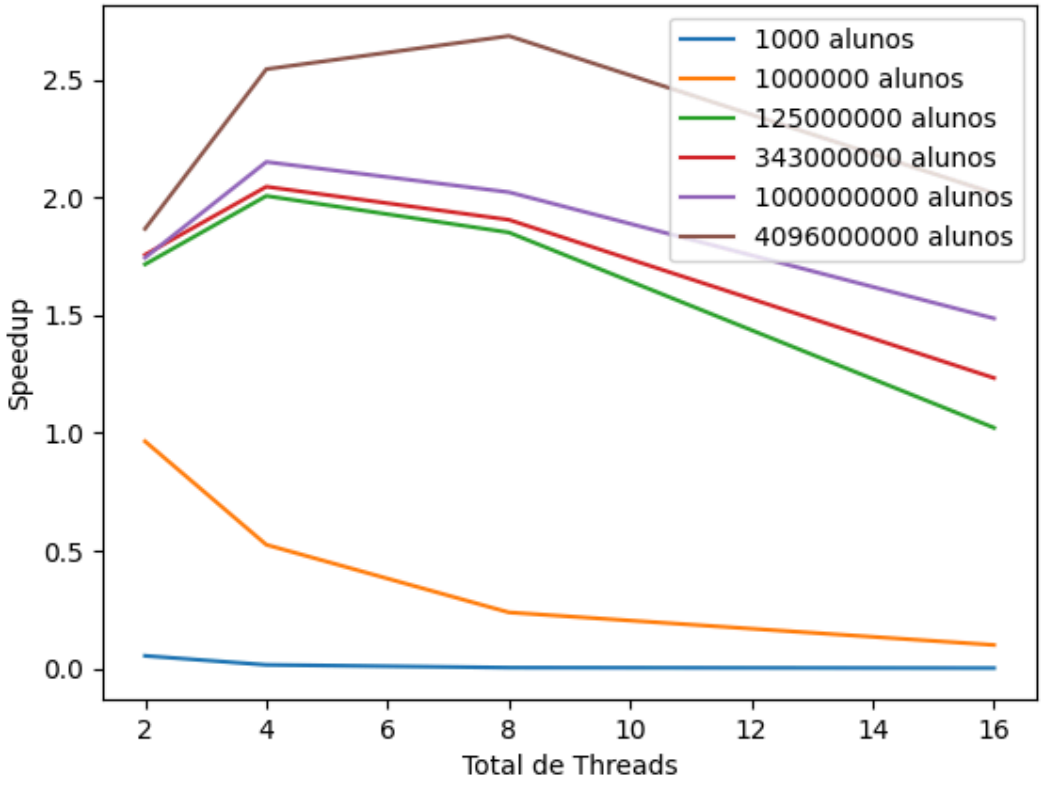
Quando comparamos o tempo de execução dos códigos sequenciais e paralelo observamos diferentes valores de Speed-up e Eficiência, conforme a variação no tamanho da entrada e número de threads. As medidas e valores utilizados nos cálculos e gráficos são a média de ao menos 50 execuções, com desvio padrão e margem de erro aceitáveis. Notamos porém que por vezes, o tempo de execução repentinamente subia por um fator fixo e depois de dadas execuções retornava ao mesmo valor menor. Isso muito provavelmente se deve a outros experimentos realizados simultaneamente no mesmo nó do cluster.
Para cada combinação de tamanho de entrada e número de threads as métricas de Speed-up e eficiência foram calculadas por:

$$Sp = T_{seq} / T_{par}$$
$$Eficiência = Sp / p$$

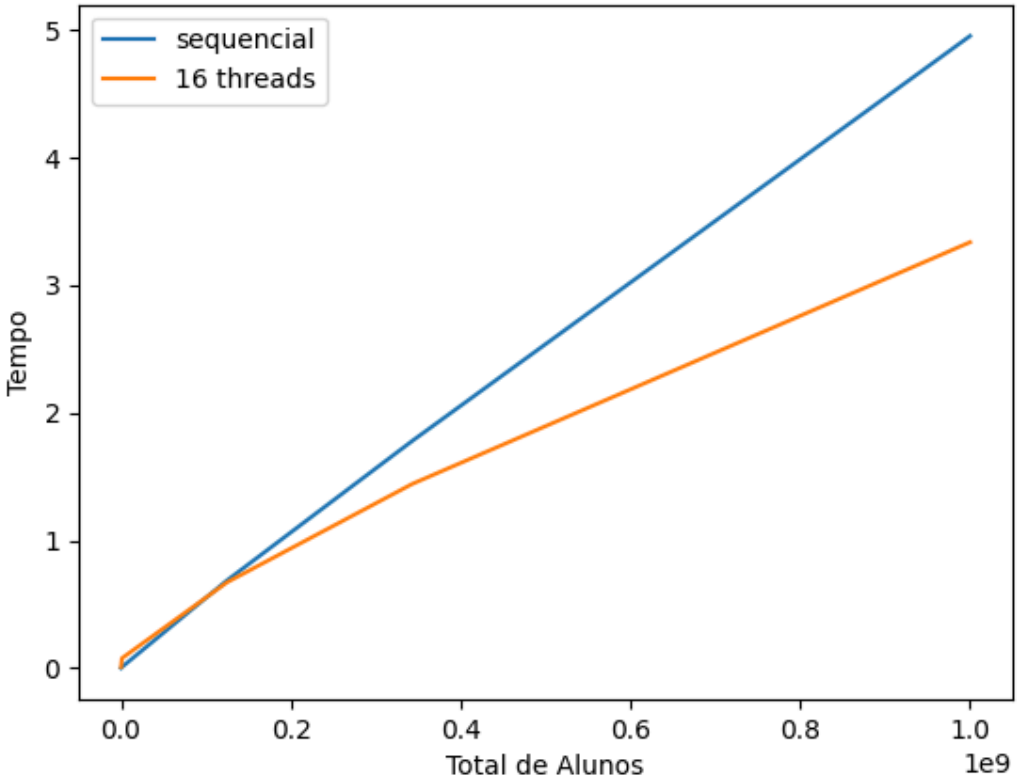
(colocar e explicar gráficos)
Abaixo, temos tabelas e gráficos apresentando resultados de nossas comparações.

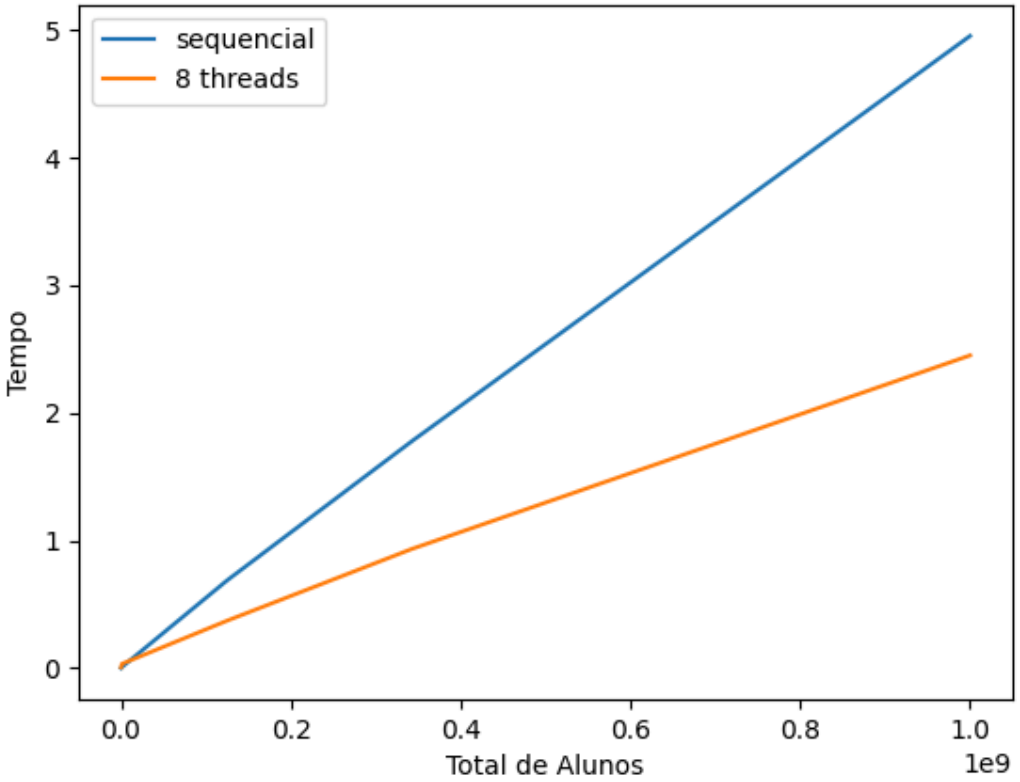
Speedups Calculados				
Total Alunos	2 Threads	4 Threads	8 Threads	16 Threads
1000	0.055166	0.0168739	0.00508	0.003275
1000000	0.96489	0.52631	0.239317	0.10115
125000000	1.71582	2.00596	1.85089	1.02246
343000000	1.75628	2.04506	1.90475	1.23357
1000000000	1.7439	2.15023	2.02180	1.48646
4096000000	1.8666	2.54362	2.68445	2.0169

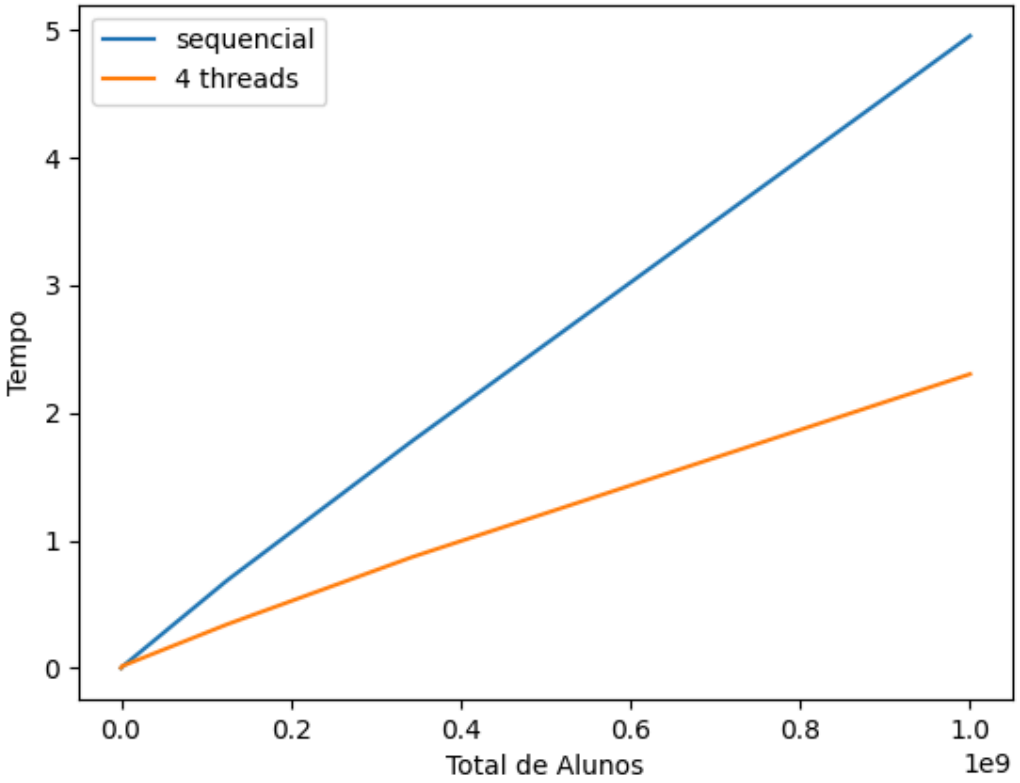
Eficiências Calculadas				
Total Alunos	2 Threads	4 Threads	8 Threads	16 Threads
1000	2.75%	0.4%	0.06%	0.02%
1000000	48.2%	13.1%	2.99%	0.63%
125000000	85.7%	50.1%	23.1%	6.39%
343000000	87.8%	51.1%	23.8%	7.70%
1000000000	87.1%	53.7%	25.2%	9.29%
4096000000	93.3%	63.5%	33.5%	12.6%

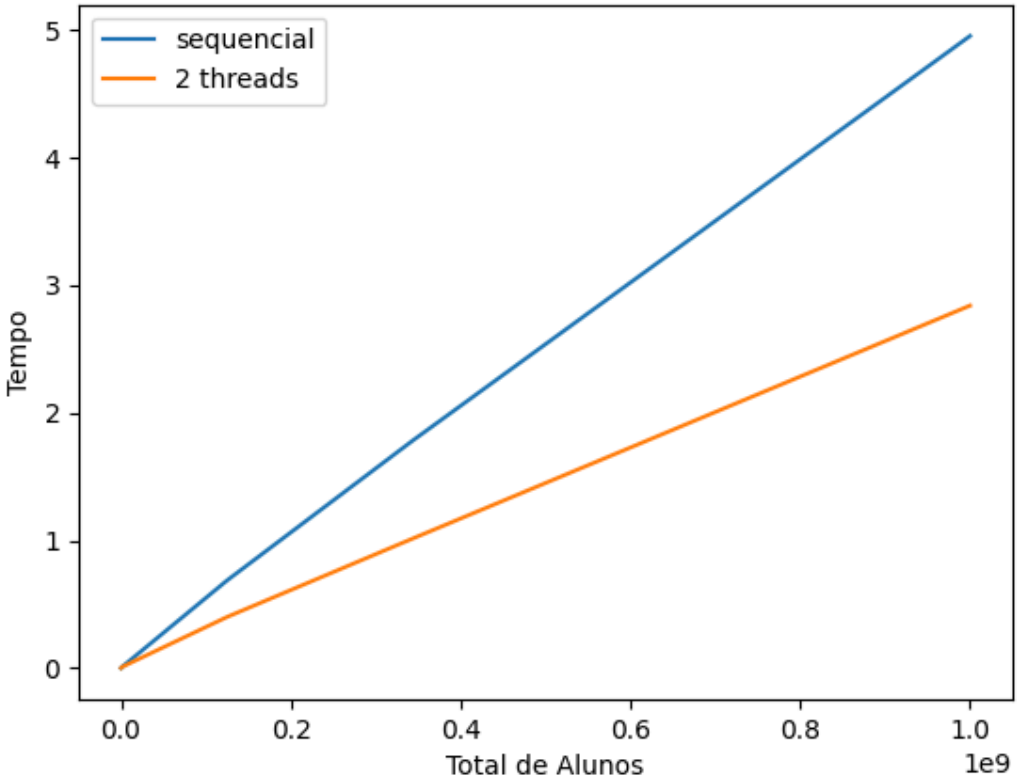


Percebe-se claramente um speedup sublinear e, que, conforme o número de threads cresce, a diferença entre o tempo sequencial e paralelo diminui.


sequencial
16 threads
Tempo
0
1
2
3
4
5
0.0
0.2
0.4
0.6
0.8
1.0
1e9
Total de Alunos


sequencial
8 threads
Tempo
0
1
2
3
4
5
0.0
0.2
0.4
0.6
0.8
1.0
1e9
Total de Alunos


sequencial
4 threads
Tempo
0
1
2
3
4
5
0.0
0.2
0.4
0.6
0.8
1.0
1e9
Total de Alunos


sequencial
2 threads
Tempo
0
1
2
3
4
5
0.0
0.2
0.4
0.6
0.8
1.0
1e9
Total de Alunos