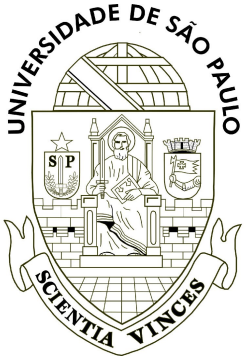
Universidade de São Paulo

Instituto de Matemática e Estatística

MAC0219/5742 - Introdução à Computação Paralela e Distribuída

2023



Exercício-programa 1: Simulação paralela de um gás ideal usando Pthreads e OpenMP

Beatriz Viana Costa - 13673214
Mariana Tiemi Silva Misu - 12542842
Maysa Cristina Claudino da Silva - 11878806

Conteúdo

1 Análise dos experimentos

Os testes de tempo foram realizados 10 vezes para cada permutação de número de *threads* e tamanhos de *grids* para cada uma das implementações apresentadas.

Tais testes foram realizados usando o programa *time_test* fornecido e foram feitos na mesma máquina para que não houvesse mudança de desempenho por conta das características do processador. O computador utilizado para os testes possui Inter(R) Core(TM) i7-11390H com 4 núcleos.

Para a apresentação dos dados de desempenho dos programas utilizando *OpenMP* e *Pthreads* foram confeccionados gráficos que apresentam o desempenho de diferentes quantidades de *threads* em um mesmo tamanho de *grids*. Já no programa sequencial, que não há mudança de número de *threads*, foi feito apenas um gráfico mostrando a diferença de desempenho entre os diferentes *grids_sizes*.

Nos gráficos, o eixo das abcissas estão dispostos os diferentes valores de *grid_size*, já nas ordenadas apresenta o tempo de execução do programa em segundos. Apresentamos os intervalos de tempos máximos e mínimos calculados de acordo com a média e variâncias encontradas nos testes realizados nos gráficos, com exceção do programa sequencial.

1.1 Análise do programa sequencial

A primeira parte do exercício-programa se refere a tomar as medidas de execução do código inicial fornecido, que foi escrito de maneira sequencial.

Os dados obtidos foram usados para as análises de desempenho posteriores dos códigos com diferentes implementações de programação paralela.

Como esperado no programa sequencial, a média de tempo para a execução do programa aumenta à medida que o tamanho da *grid* de entrada aumenta.

O acréscimo de tempo se torna mais evidente a partir do momento em que a *grid_size* assume o valor 1024.

O maior valor de tempo de execução foi apresentada quando a grade apresentou seu maior tamanho, 4096, e o tempo foi igual à 12 segundos.

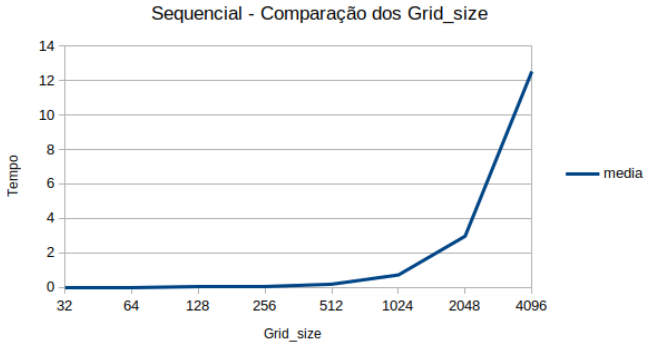


Figura 1: Dados de desempenho do programa sequencial

Como as variações na medição do tempo foram mínimas, colocamos apenas a média no gráfico. Mas para a apresentação dos dados, segue os valores máximos e mínimos calculados com 95% de confiança, como indicado no início da seção.

Grid_size	Tempo máx. (s)
32	0,0022105
64	0,0057088
128	0,0152232
256	0,0478589
512	0,1760571
1024	0,7203576
2048	2,9953374
4096	12,6735798

(a) Tempos máximos calculados.

Grid_size	Tempo mín. (s)
32	0,0020154
64	0,0052099
128	0,0145837
256	0,0464598
512	0,1736090
1024	0,7050119
2048	2,9331809
4096	12,3776697

(b) Tempos mínimos calculados.

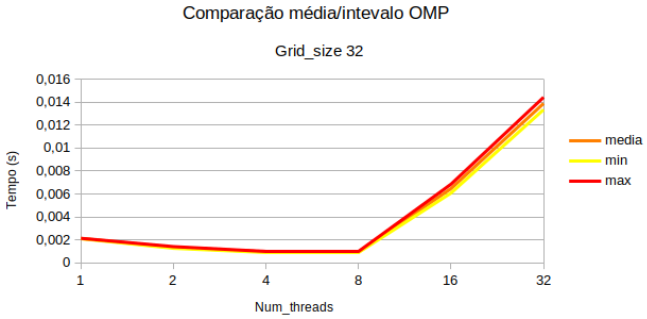
Tabela 1: Intervalos de tempo de execução calculados para o programa sequencial.

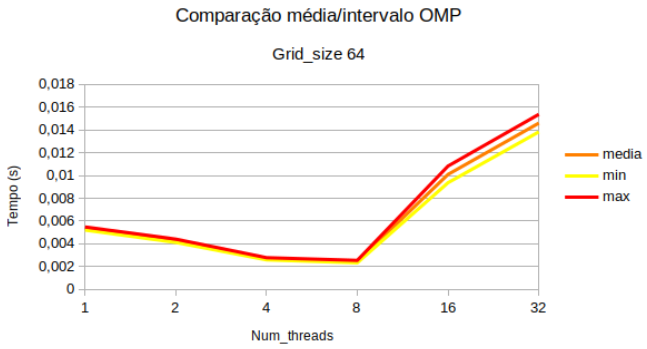
1.2 Análise do programa em OpenMP

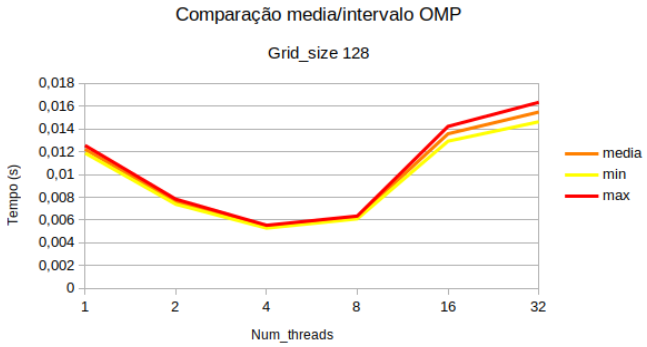
A segunda parte do exercício-programa consistia em realizar a modificação do código utilizando a linguagem de programação paralela OpenMP.

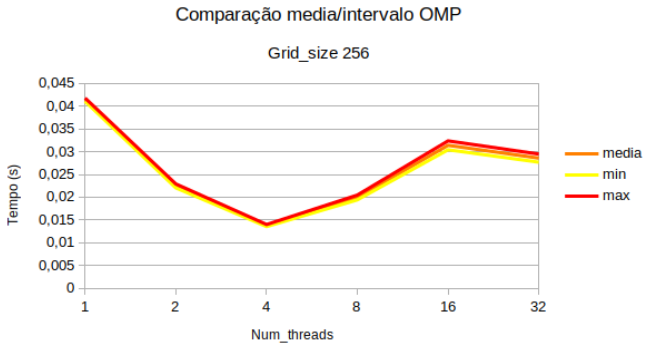
Para esta implementação foi modificada apenas a função *update()*, apenas paralelizando o *for* mais externo, o que não gera conflitos, pois o local de leitura e escritura são dois arquivos distintos e também torna o programa mais eficiente, pois este *for* percorre toda a altura da *grid*.

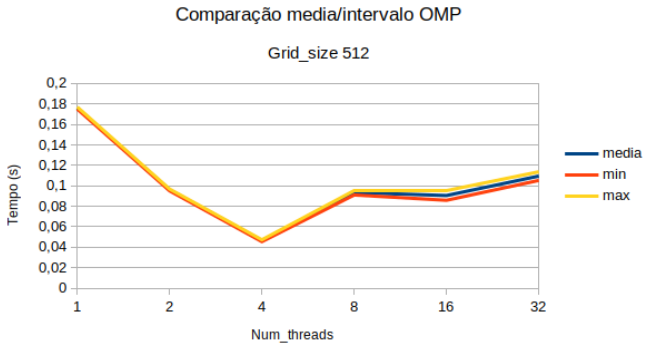
Segue tabelas com médias e intervalos calculados. Como dito anteriormente, cada gráfico apresenta o desempenho de um tamanho de *grid* com diferentes números de *threads*.

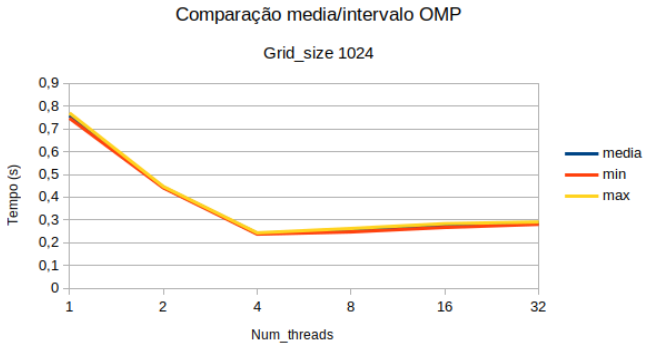


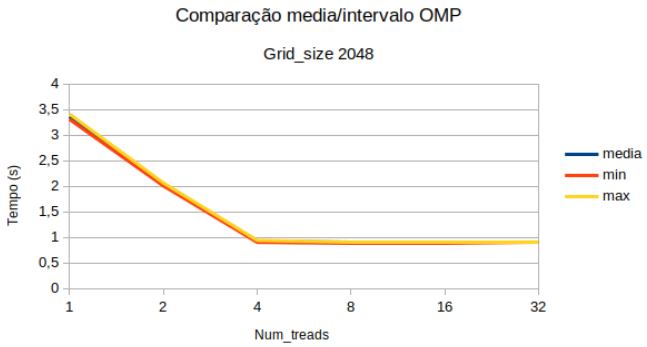


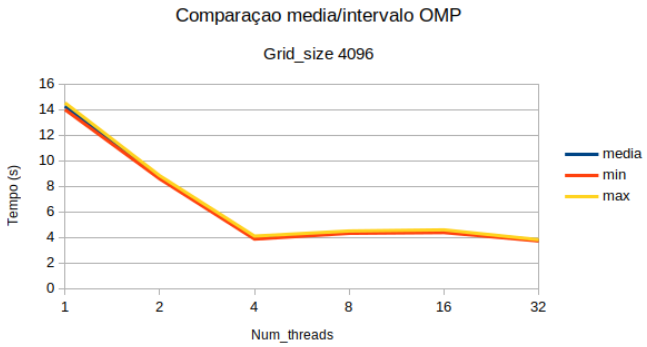












É possível notar que a quantidade de *threads* que apresentou melhor desempenho em todos os tamanhos de *grid* foi 4 *threads*, seguido de 8.

Podemos notar também que, até a *Grid_size* de tamanho 512, todos os testes realizados com a quantidade de *threads* maior do que 8 apresentaram resultados piores em relação ao tempo gasto na execução do programa; ademais, a partir do tamanho 1024, os testes com maior quantidade de *threads*, mais especificamente a partir de 4, apresentaram resultados melhores. Contudo, ainda sim 4 *threads* se mostra a quantidade ideal em todos os testes, por conta das especificações do processador utilizado nos testes.

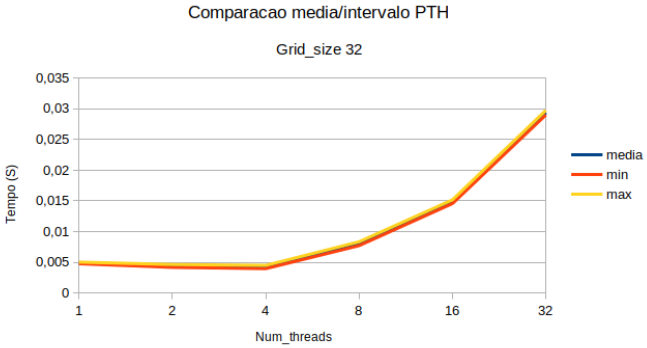
Além disso, é importante ressaltar que os tempos apresentados em todos os testes em que a quantidade de *threads* era igual à 1 são similares aos resultados apresentados no programa sequencial.

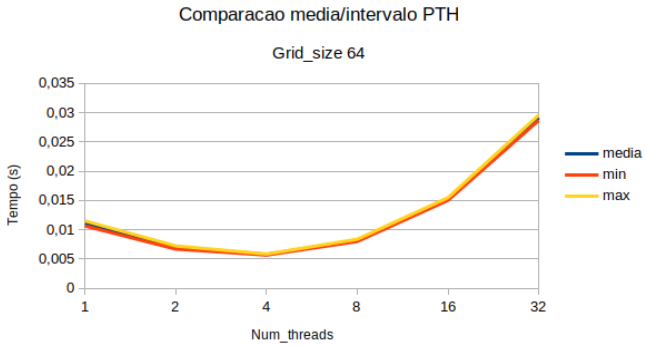
1.3 Análise do programa em Pthreads

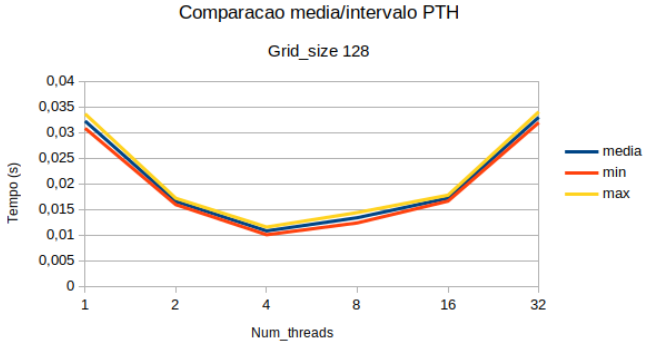
A terceira e última parte era referente a paralelização do código fornecido utilizando *Pthreads*.

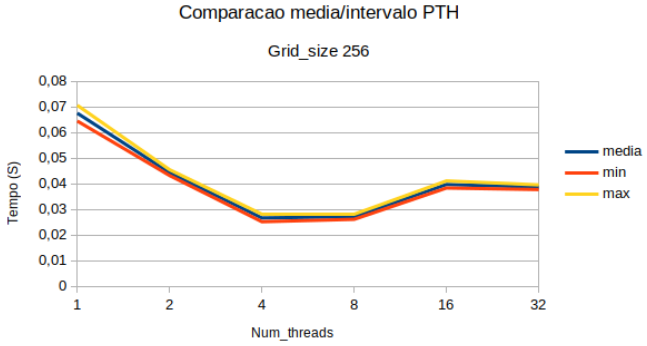
A implementação desta paralelização seguiu a mesma ideia da anterior, ocorrendo no *for* mais externo da função *update*.

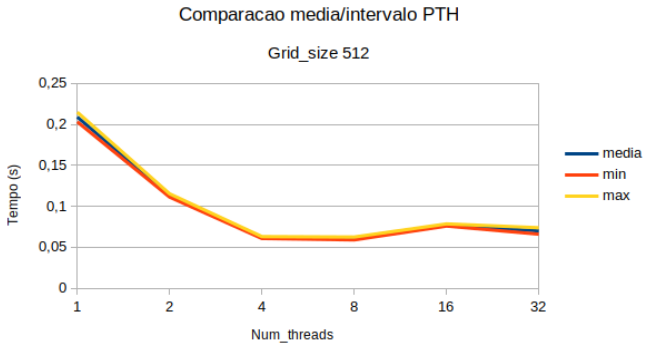
Abaixo estão os gráficos com as médias com resultados obtidos dos testes de diferentes tamanhos de *grid* e diferentes números de *threads* em função do tempo.

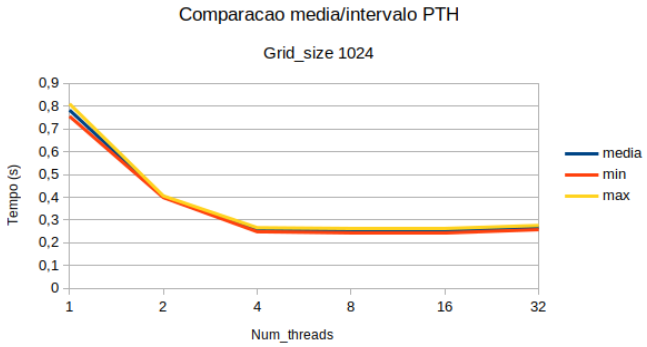


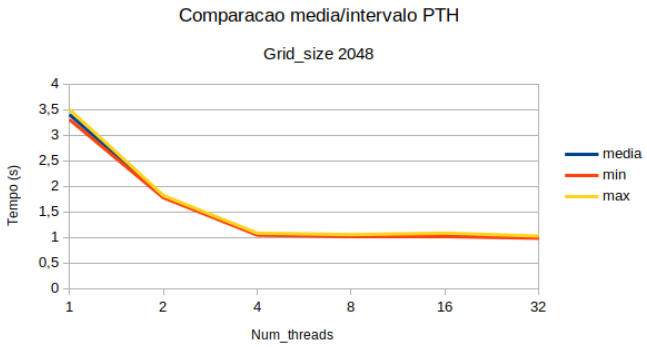


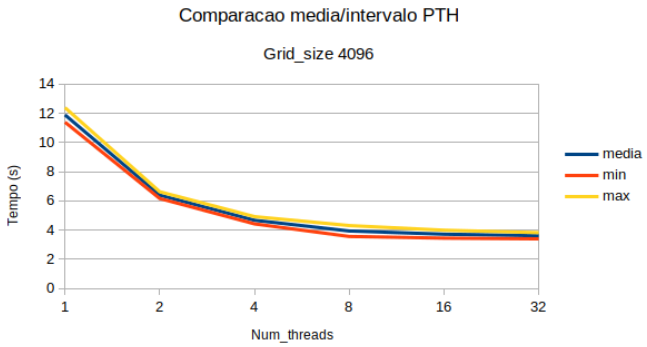












Para *grid* de tamanho 32 o resultado é parecido com a implementação sequencial pois o tempo aumenta junto ao número de *threads*.

O cenário começa a mudar a partir do *grid* de tamanho 64, onde o tempo apresenta uma queda no intervalo (1, 4) antes de subir. No tamanho de *grid* 128 acontece o mesmo fenômeno porém dessa vez é atingido o menor valor de amplitude entre a menor e maior média obtida, conforme apresentado na tabela abaixo:

grid_size	**maior média**	**menor média**	**amplitude**
32	0,0294	0,0042	0,0251
64	0,0291	0,0057	0,0233
128	0,0330	0,0108	**0,0222**
256	0,0676	0,0266	0,0409
512	0,2090	0,0607	0,1483
1024	0,7831	0,2527	0,5303
2048	3,4062	1,0042	2,401
4096	11,8974	3,6087	8,2887

Tabela 2: Tempo em segundos das maiores e menores médias de cada teste e sua amplitude.

A partir do tamanho de *grid* 256 observamos a tendência do tempo iniciar alto para 1 única *thread*, diminuir até atingir 4 *threads* e se estabilizar para as demais quantidades de *threads*. Além disso, tal como na implementação em *OpenMP* todos os testes apresentaram melhores resultados para 4 *threads* se configurando como a melhor opção para a máquina em questão.

2 Conclusão final

A partir dos experimentos realizados foi possível notar que em ambas as implementações paralelizadas, *OpenMP* e *Pthreads*, obteve-se um melhor resultado ao utilizar 4 *threads* isso ocorre pois o processador do computador onde foram feitos os testes é um Inter(R) Core(TM) i7-11390H com 4 núcleos, assim utilizando o dispositivo em sua maior capacidade.

Apesar desa semelhança, em geral a solução com *OpenMP* é mais rápida em relação à implementação com *Pthreads*, a não ser quando o tamanho da *grid* é muito grande, por exemplo a solução em *Pthreads* para o tamanho de *grid* 2048 demorou tempo similar ao *OpenMP* e levou menos tempo para tamanho de *grid* 4096.

Logo, os valores ideais de quantidade de *threads* vai depender das especificações do processador do computador usado e da dimensão da *grid*.