Relatório do Desempenho de Aplicações Paralelas Usando OpenMP

Miguel Nunes

22/09/2022

1 Algoritmos e Descrição do Ambiente

Foi analisado o algoritmos de Mandelbrot paralelizado usando *pthreads* e Open MP, com apenas leves modificações na entrada de dados e alocação de memória para facilitar a automatização dos testes. Foram utilizadas os valores como entrada:

```
max_row    = 340
max_column = 1000
max_n      = 4600
```

Pois foi observado experimentalmente que, em execuções sequenciais do algoritmo, essas entradas levavam, em média, 64 segundos para serem processadas.

Os testes foram realizados num computador com as seguintes especificações:

CPU `AMD Ryzen 7 5700G with Radeon Graphics (16) @ 4.673GH`

Memória `63588MiB`

OS `Fedora Linux 36 (MATE-Compiz) x86_64`

Kernel `5.18.19-200.fc36.x86_64`

Os testes foram executados automaticamente a partir de um script em bash, com o auxílio de um makefile. As entradas para os testes eram geradas pelo makefile, ao início de uma nova rodada de testes, todos os arquivos gerados no teste anterior são deletados.

Apenas foi testada a versão do algoritmo em Open MP, já que a versão com *pthreads* havia sido testada no trabalho anterior.

2 Coleta dos Dados

Ao fim da execução dos algoritmos, o tempo levado para realizar seu cálculo e a quantidade de *threads* utilizada é é imprimida para `stdout`. Nos testes `stdout` era redirecionado para um arquivo de texto por meio de operações de *pipe* do bash. O nome dos arquivos de saída segue o formato `algoritmo_númeroTeste_númeroThreads`, onde _ é apenas um separador para visualização e não estava de fato no nome dos arquivos. Por exemplo, o arquivo `mandelbrotStatic16` se refere a segunda execução do algoritmo de mandelbrot com 6 *threads* usando o *schedule* estático.

Foi feito um script em *Python* versão 3.10 para processar esses dados. Os arquivos são lidos "em massa" e seus dados são carregados em estruturas *dict* para serem manipulados. Os dados obtidos são os seguintes:

Threads	Exec 0	Exec 1	Exec 2	Exec 3	Exec 4	Exec 5	Exec 6	Exec 7	Exec 8	Exec 9
1	64.62736	64.67839	64.63666	64.61962	64.6403	64.65312	64.65453	64.63046	64.72165	64.70816
2	32.83472	32.69233	32.79807	32.73635	32.83295	32.82157	32.83873	32.60417	32.87441	32.87749
3	40.2882	40.34767	40.39883	40.27371	40.35982	40.36073	40.38627	40.41132	40.40303	40.34727
4	27.2599	27.19623	27.28722	27.11876	27.28726	27.16293	27.29407	27.29752	27.36482	27.31697
5	26.93793	26.9792	26.91464	26.98675	26.98278	26.94668	27.02919	27.03469	27.01199	27.05143
6	20.63402	20.64316	20.71388	20.6398	20.71072	20.70096	20.65364	20.65875	20.66586	20.67334
7	19.56035	19.57419	19.60095	19.54903	19.59881	19.58438	19.59719	19.62478	19.61138	19.59587
8	16.68858	16.63405	16.70305	16.71212	16.7054	16.70179	16.67909	16.74911	16.75054	16.69679

Table 1: Dados da execução do algoritmo de Mandelbrot com *pthreads*

Threads	Exec 0	Exec 1	Exec 2	Exec 3	Exec 4	Exec 5	Exec 6	Exec 7	Exec 8	Exec 9
1	9.78393	9.80667	9.80489	9.81746	9.80339	9.81952	9.78686	9.8202	9.82036	9.82429
2	5.01427	4.93913	4.9759	4.95032	4.99144	4.9761	4.96398	4.97508	4.9792	4.98763
3	6.13053	6.11679	6.11678	6.14326	6.12834	6.12032	6.10977	6.11775	6.12596	6.14628
4	4.15169	4.14156	4.1114	4.13331	4.1311	4.15066	4.15062	4.12374	4.12873	4.1409
5	4.08908	4.09408	4.09256	4.09621	4.08099	4.0762	4.10196	4.08305	4.08978	4.08311
6	3.13102	3.13283	3.13701	3.1082	3.12533	3.13172	3.1306	3.12288	3.12566	3.13176
7	3.04048	3.00827	3.04431	3.02491	3.04205	3.01365	3.02013	3.02774	3.0265	3.02325
8	2.56239	2.55556	2.57713	2.56433	2.5605	2.56472	2.56242	2.56517	2.57673	2.56543

Table 2: Dados da execução do algoritmo de Mandelbrot com *schedule* estático

Threads	Exec 0	Exec 1	Exec 2	Exec 3	Exec 4	Exec 5	Exec 6	Exec 7	Exec 8	Exec 9
1	9.83482	9.80162	9.80322	9.80485	9.80949	9.84028	9.82023	9.82385	9.81263	9.83629
2	4.98459	5.00203	4.99481	4.9835	4.96478	4.95368	4.9838	4.96962	4.96588	4.96698
3	6.122	6.11685	6.13562	6.1481	6.09653	6.13217	6.13106	6.10061	6.12665	6.13536
4	4.15264	4.14629	4.12521	4.13607	4.12597	4.1286	4.11322	4.13178	4.13293	4.11002
5	4.07857	4.0968	4.10565	4.08905	4.09445	4.1129	4.09365	4.09427	4.1033	4.08883
6	3.13595	3.13061	3.13882	3.12771	3.13159	3.1098	3.11586	3.10823	3.13267	3.11275
7	3.03186	3.03353	3.04228	3.03629	3.00831	3.03027	3.02419	3.03084	3.01781	3.03072
8	2.56125	2.55367	2.5611	2.55851	2.55608	2.55921	2.56632	2.56237	2.54864	2.56663

Table 3: Dados da execução do algoritmo de Mandelbrot com *schedule* dinâmico

Threads	Exec 0	Exec 1	Exec 2	Exec 3	Exec 4	Exec 5	Exec 6	Exec 7	Exec 8	Exec 9
1	9.76654	9.80699	9.82049	9.80722	9.81139	9.81294	9.82094	9.82568	9.82483	9.82965
2	4.96658	4.99545	4.99377	4.97251	4.96081	4.96774	4.95027	4.98712	4.97739	4.97696
3	6.1409	6.10176	6.13289	6.12336	6.1221	6.13359	6.13511	6.13349	6.1235	6.13184
4	4.14677	4.13439	4.15167	4.13562	4.12293	4.12498	4.14187	4.11693	4.11899	4.14177
5	4.09641	4.09522	4.08594	4.08461	4.08063	4.10762	4.08842	4.10045	4.08878	4.09802
6	3.1333	3.12916	3.13164	3.10829	3.13986	3.13015	3.13029	3.11742	3.12848	3.11916
7	3.01587	3.0235	3.03445	3.0419	3.03224	3.02162	3.01682	3.02251	3.0192	3.01478
8	2.56642	2.56725	2.57673	2.55449	2.56545	2.57011	2.5644	2.57417	2.56103	2.55936

Table 4: Dados da execução do algoritmo de Mandelbrot com *schedule* guiado

3 Análise do Dados

Sobre os dados apresentados acima, podemos obter as seguintes informações, onde o tempo é medido em segundos:

	Média do Tempo de Execução	Desvio Padrão
1 Thread	64.777607	0.16630814071409003
2 Threads	32.897305	0.16732219181035715
3 Threads	40.482982	0.04887138853948963
4 Threads	27.345909	0.10432392396335134
5 Threads	27.095582	0.05337008271390313
6 Threads	20.748611	0.04259400204254115
7 Threads	19.685831	0.03082819193170025
8 Threads	16.775065	0.02461598543404037

Table 5: Médias e desvio padrão do algoritmo de Mandelbrot com *pthreads*

	Média do Tempo de Execução	Desvio Padrão
1 Thread	9.808757	0.01430773997993158
2 Threads	4.975305	0.02110255184674231
3 Threads	6.125578	0.01186485735270328
4 Threads	4.136371	0.01319911566903059
5 Threads	4.088702	0.00786163087406173
6 Threads	3.127701	0.00801616623524678
7 Threads	3.027129	0.01203302441893417
8 Threads	2.565438	0.00671966070572017

Table 6: Médias e desvio padrão do algoritmo de Mandelbrot com *schedule* estático

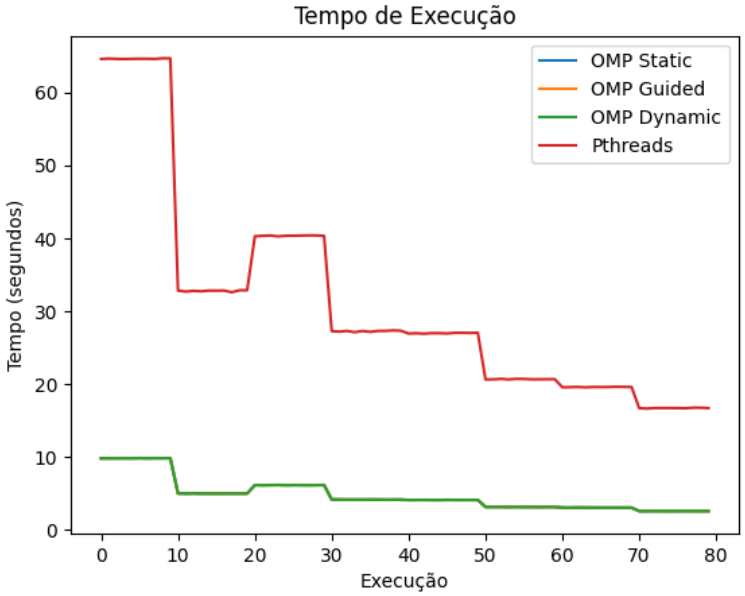
	Média do Tempo de Execução	Desvio Padrão
1 Thread	9.818728	0.01456274836698083
2 Threads	4.976967	0.01513441336087318
3 Threads	6.124495	0.01606536734026911
4 Threads	4.130273	0.01311360790256527
5 Threads	4.095747	0.00969797579108356
6 Threads	3.124399	0.01151921626384943
7 Threads	3.028609	0.00966705286584861
8 Threads	2.559377	0.00553523019695965

Table 7: Médias e desvio padrão do algoritmo de Mandelbrot com *schedule* dinâmico

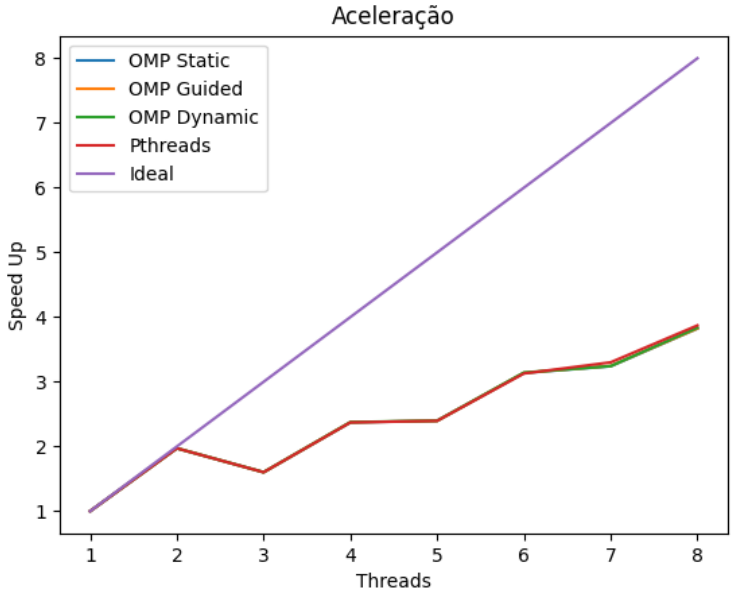
É evidente que o aumento de threads diminui significativamente o tempo de execução dos algoritmos sem causar aumento significativo no desvio padrão, assim como que a utilização do Open MP, mesmo nos casos com apenas 1 thread, significativamente diminui o tempo de execução do algoritmo, como pode ser observado no gráfico abaixo, onde as execuções [0,. . . ,9] tem 1 thread, as execuções [10,. . . ,19] tem duas threads e assim por diante:

	Média do Tempo de Execução	Desvio Padrão
1 Thread	9.812667	0.01803282319056608
2 Threads	4.974860	0.014412573985547702
3 Threads	6.127850	0.010964194452854417
4 Threads	4.133592	0.01210745298474367
5 Threads	4.092610	0.008310102285772432
6 Threads	3.126775	0.00917122098983795
7 Threads	3.024289	0.009009298221528924
8 Threads	2.565941	0.006704982310027237

Table 8: Médias e desvio padrão do algoritmo de Mandelbrot com *schedule* guiado

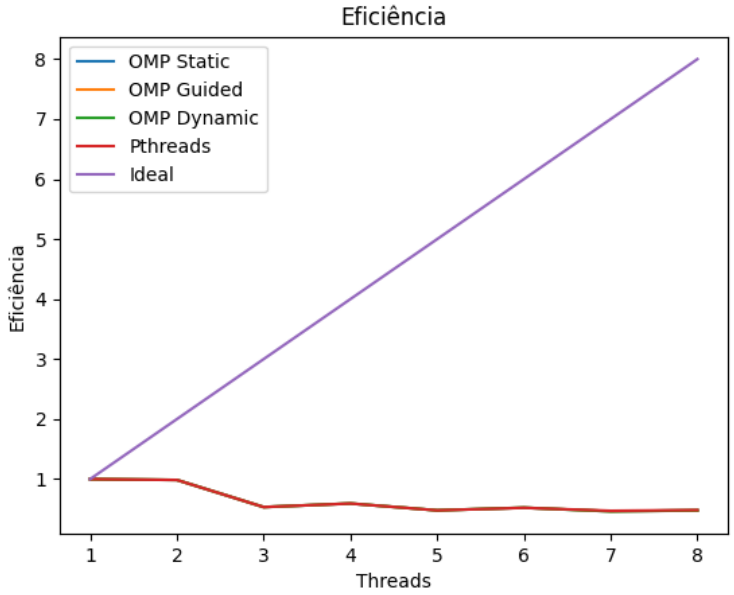


Ademais, os seguinte gráficos de aceleração e eficiência foram elaborados:



É possível observar que o algoritmo de Mandelbrot com *pthreads* não teve bom desempenho ao ser paralelizado, apesar de ter um aumento de performance significativo em uma execução com 8 threads se comparada com a execução sequencial, esse aumento não é tão grande se comparado com uma execução com apenas 6 threads. Mais ainda, o desempenho não é consistente, isto é, houveram situações onde execuções com menos threads tiveram desempenho melhor que execuções com mais threads.

Isso também ocorre, porém de maneira menos significativa nos casos de uso do Open MP, onde, apesar do tempo absoluto ser significativamente menor que nos casos de *pthreads*, as taxas de aceleração se mantiveram as mesmas.



O algoritmo de Mandelbrot com *pthreads* novamente teve um desempenho ruim. O gráfico torna evidente que este algoritmo, na sua implementação testada, não lida bem com paralelização além de duas threads. A baixa eficiência do algoritmo nos casos com mais de duas threads pode ser interpretada como uma indicação que a implementação de paralelismo testada não é a ideal e pode ser melhorada.

O mesmo pode ser dito para os casos de uso de Open MP, e específico a estes, o uso de diferentes *schedules* de paralelismo não tiverem impacto significativo nas taxas de aceleração ou eficiência do algoritmo.