Relatório do Desempenho de Aplicações Paralelas

Miguel Nunes

02/09/2022

1 Algoritmos e Descrição do Ambiente

Foram analisados os algoritmos de Mandelbrot e Multiplicação de Matrizes, com apenas leves modificações na entrada de dados para facilitar a automatização dos testes. Para o algoritmo de Mandelbrot, foram utilizadas os valores:

```
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.

Já para o algoritmo de Multiplicação de Matrizes, a entrada `n = 2340` foi utilizada, pois, análogo ao caso anterior, foi observado que essa entrada levava, em média, 54 segundos para ser processada.

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.

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 `mandelbrot16` se refere a segunda execução do algoritmo de mandelbrot com 6 *threads*, já o arquivo `matrix58` se refere a sexta execução do algoritmo de multiplicação de matrizes com 8 *threads*.

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	64.7158	65.21128	64.73953	64.79531	64.67935	64.68465	64.69427	64.66029	64.88569	64.7099
2	33.13408	32.81293	32.88506	32.70995	33.0712	32.92352	33.03085	32.69934	33.02866	32.67746
3	40.49595	40.4585	40.4149	40.46098	40.49784	40.55031	40.53603	40.51463	40.49988	40.4008
4	27.44845	27.21293	27.45597	27.43875	27.26834	27.3451	27.28704	27.28093	27.49284	27.22875
5	27.18402	27.11974	27.0774	27.13122	27.01274	27.07804	27.01961	27.07903	27.11221	27.14181
6	20.75757	20.77389	20.68474	20.81157	20.73202	20.79684	20.76261	20.72167	20.68593	20.75927
7	19.67249	19.67188	19.65971	19.64016	19.71861	19.69955	19.70114	19.65318	19.73501	19.70659
8	16.75083	16.78174	16.77071	16.78651	16.81145	16.74864	16.74882	16.75016	16.79915	16.80264

Table 1: Dados da execução do algoritmo de Mandelbrot

Threads	Exec 0	Exec 1	Exec 2	Exec 3	Exec 4	Exec 5	Exec 6	Exec 7	Exec 8	Exec 9
1	58.19308	58.85921	57.25952	56.50984	57.08937	56.59573	56.55776	56.27447	56.69656	57.30469
2	27.69368	27.36266	27.68316	27.6043	27.63449	27.59407	27.48514	27.69325	27.65521	27.64803
3	18.03399	18.17427	18.11412	18.14828	18.11417	18.18025	17.98689	18.18972	18.00595	18.09148
4	13.51867	13.47603	13.47634	13.47873	13.49899	13.45625	13.61785	13.45564	13.40626	13.47542
5	10.70563	10.71371	10.68173	10.70417	10.70313	10.71331	10.70657	10.74149	10.69732	10.77864
6	8.78966	8.85618	8.8318	8.81042	8.83356	8.88258	8.92194	8.83724	8.88398	8.8047
7	7.48969	7.52113	7.53303	7.58157	7.54399	7.46355	7.46856	7.58963	7.50519	7.47868
8	6.41058	6.43281	6.48943	6.41859	6.42211	6.36786	6.42151	6.43479	6.47839	6.37533

Table 2: Dados da execução do algoritmo de Multiplicação de Matrizes

3 Análise do Dados

Sobre os dados apresentados acima, podemos obter as seguintes informações:

	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

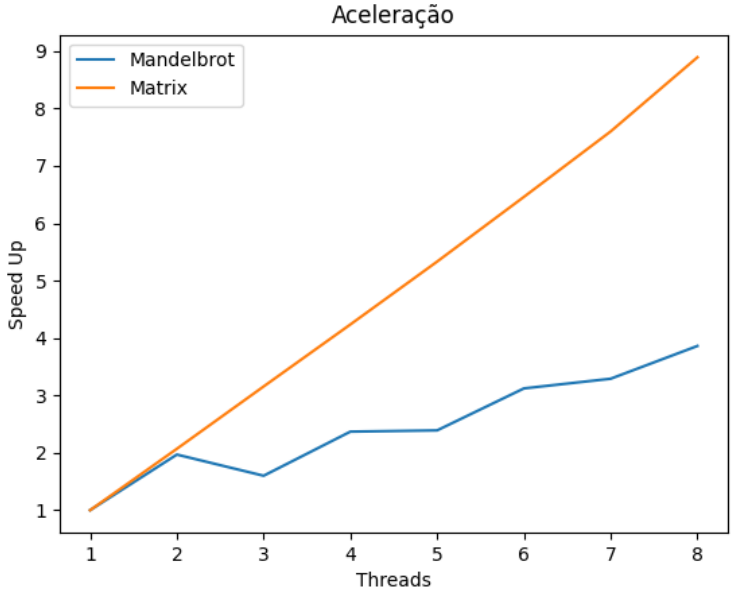
Table 3: Médias e desvio padrão do algoritmo de Mandelbrot

	Média do Tempo de Execução	Desvio Padrão
1 Thread	57.134023	0.8223537073013178
2 Threads	27.605399	0.10552352749242376
3 Threads	18.103912	0.07352478279086958
4 Threads	13.486018	0.054917365964187136
5 Threads	10.71457	0.027061583677070934
6 Threads	8.8452060	0.0411281758517063
7 Threads	7.517502	0.04472313639369324
8 Threads	6.4251400	0.03823902369743929

Table 4: Médias e desvio padrão do algoritmo de Multiplicação de Matrizes

É evidente que o aumento de threads diminui significativamente o tempo de execução dos algoritmos sem causar aumento significativo no desvio padrão.

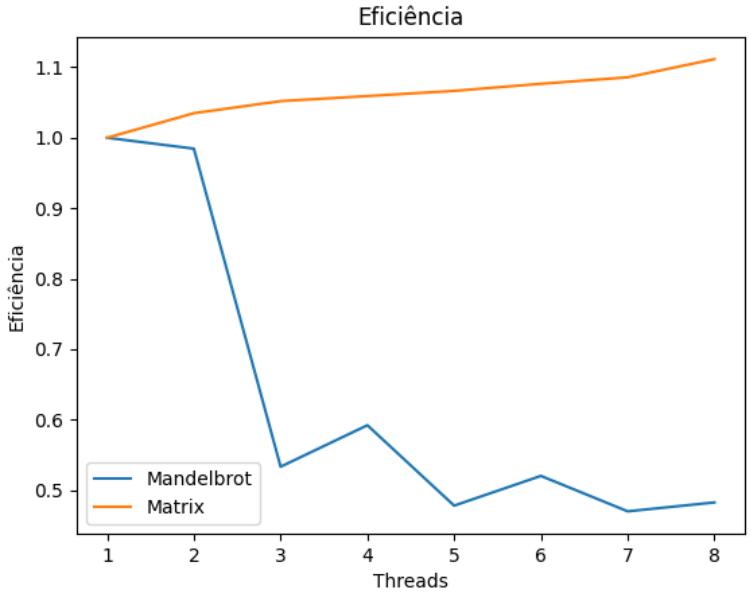
Ademais, os seguinte gráficos de aceleração e eficiência foram elaborados:



É possível observar que o algoritmo de Mandelbrot 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.

Já no caso do algoritmo de Multiplicação de Matrizes, temos um caso ideal, onde o desempenho do programa cresce linearmente com o aumento do número de threads utilizadas.



O algoritmo de Mandelbrot 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.

Já o algoritmo de Multiplicação de Matrizes teve um desempenho peculiar. Ao contrário do algoritmo de Mandelbrot, sua eficiência continuamente aumentou com a adição de novas threads, não só isso, mas ela esteve acima de 100% em todo os momentos, o que pode ser um indicador de que a implementação, no computador onde foi testada, tem comportamento superlinear.