Autores: Dhruv Babani, Eduardo Bregalda, Eduardo Barcellos

T3-Programa Paralelo mestre/escravo usando MPI

1 – Solução Geral

O trabalho se basea na compilação, execução e analise de dois programas de ordenação de vetores. Há duas versões, a sequencial e a paralela. A paralela usa o algoritmo de mestre e escravo. Ambas versões possuem a implementação de dois algortimos de ordenação, Quick Sort e Bubble Sort.Além disso, ela foi compilada tanto pelas máquinas do LAD e pela maquina pessoal(8/16)

2 – Versão Sequencial

2.1 Quick Sort

Utilizando o algoritmo de ordenação Quick Sort, como uma carga de trabalho de 5000 arrays e cada array com 100000 elementos. O Algoritmo demorou 28.1374s para completar.

2.2 Bubble Sort

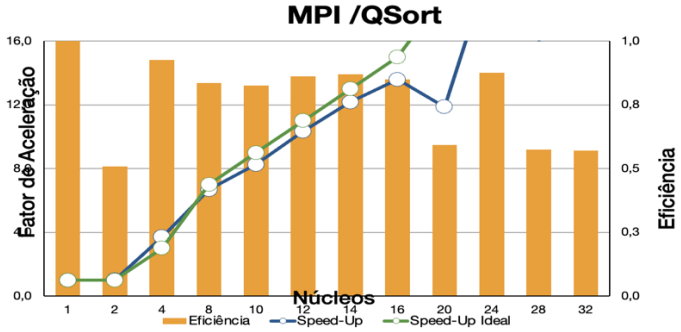
Já usando o algoritmo Bubble Sort, não foi possível terminar sua execução com os mesmo parametros da execução em Quick Sort, com uma leve alteração do numero de elementos para 10k. Considerando que o aumento de tempo e a quantidade de arrays são proporcionais. Chegamos a conclusão que demoraria cerca de 30 minutos para executar o algoritmo.

3 – Versão Paralela

Para a versão paralela usando MPI, podemos, então, controlar a quantidade de núcleos que serão usados na execução. Para isso, foi executado o mesmo programa utilizando quantidades de núcleos diferentes (2 a 32 núcleos).

3.1 QuickSort

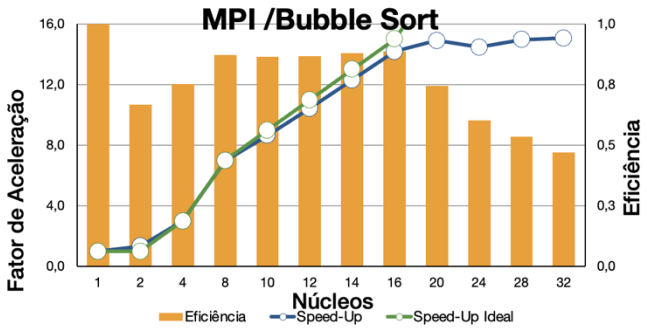
O Resultado para versão em Quick Sort se dá pelo gráfico de Spped Up e Eficiência.



Concluímos que, as execuções até 16 núcleos, houve um aumento expressivo de Speed-Up, que por sua vez estão acompanhando o Speed-Up ideal. A partir de 20, o speed-up teve uma queda a 12 e depois começou ter uma oscilação para cima e baixo, no final resultando um Speed up a 18. A partir de 16, também, começou a haver uma queda na eficiência. Além disso, o tempo levado na minha máquina foi de 7,2 segundos

3.2 BubbleSort

O resultado para versão em BubbleSort se dá pelo gráfico de Speed Up e Eficiência.



O resultado utilizando BubbleSort foi bem diferente ao QuickSort em termos de Speed-Up.Até os 14 núcleos, o Speed-up aumentou consideralvemente, acompanhando próximo ao Speed-Up ideal com o aumento do número de Núcleos.Considerando que a apartir de 16 núcleos, que serão usados pela tecnologia de Huper Threading. Podemos concluir que não houve uma melhora significativa de Speed-up, e alguns casos houve piora. Também a partir de 16 núcleos, começou haver uma queda na Eficiência. Ademais, na minha maquina teve uma performance muito baixa, levando cerca de 404,8 segundos.

Tabelas do Bubble Sort e Quick Sort

1.Bubble Sort

Núcleos	Tempo de Execução	Speed Up	Speed Up Ideal	Eficiência
1	2239,02	1,0	1	1,0
2	1680	1,3	1	0,7
4	744,97	3,0	3	0,8
8	320,77	7,0	7	0,9
10	259,02	8,6	9	0,9
12	214,82	10,4	11	0,9
14	181,86	12,31177829	13	0,879412735
16	157,83	14,18627637	15	0,886642273
20	150,16	14,91089505	19	0,745544752
24	154,83	14,46115094	23	0,602547956
28	149,49	14,97772426	27	0,534918724
32	148,6	15,06742934	31	0,470637765

2.Quick Sort

Núcleos	Tempo de Execução	Speed Up	Speed Up Ideal	Eficiência
1	28	1,0	1	1,0
2	27,68	1,0	1	0,5
4	7,59	3,7	3	0,9
8	4,21	6,7	7	0,8
10	3,41	8,3	9	0,8
12	2,72	10,3	11	0,9
14	2,31	12,18069264	13	0,870049474
16	2,07	13,59294686	15	0,849559179
20	2,37	11,87232068	19	0,593616034
24	1,339	21,0137416	23	0,875572567
28	1,75	16,07851429	27	0,574232653
32	1,54	18,27103896	31	0,570969968