Trabalho de Programação Paralela e Distribuída - Mandelbrot Set com MPI

Daniela Kuinchtner – 152064@upf.br

Implementação

O algoritmo foi dividido em *chunks* (grão = 48) e foi utilizado o modelo mestre-escravo para processar eles, onde o processo 0 (mestre) controlará a comunicação entre os processos escravos e o que cada um executará.

Dificuldades encontradas

Compreender os parâmetros das funções disponibilizadas pelo MPI.

Recursos utilizados

No início da função *main* foi iniciada a sincronização com todos os processos utilizando o MPI_Init e obtido a quantidade de processos disponíveis e o *rank* de cada um deles com os comandos MPI_Comm_size e MPI_Comm_rank, respectivamente. Foi utilizado MPI_Send e MPI_Recv para enviar e receber mensagens entre os processos e, também, MPI_ANY_SOURCE e MPI_ANY_TAG para que seja possível receber um valor no MPI_Recv de qualquer processo e *tag*. E, por último, o MPI_Finalize é chamado no final da função *main* para encerrar a paralelização. Não foi realizado nenhum teste de comparação com outras formas de comunicação coletiva, a escolha de usar MPI_Send e MPI_Recv foi por opção.

Resultados

Foram utilizados dezesseis computadores com processador Intel® Core™ i7-2600 CPU @ 3.40GHz × 8, 4 núcleos físicos e 8 *threads*, 8 GB de memória RAM e o sistema operacional Ubuntu 16.04 LTS, 64-bit. A Tabela 1 mostra os resultados obtidos (média, *speedup*, eficiência e custo) das 5 execuções paralelas realizadas com as entradas de 1024 linhas, 768 colunas e 18000 iterações para 2, 4, 8, 16, 32 e 64 processos. O Anexo 1 mostra todos os tempos das 5 execuções paralelas.

Processos	Tempo em seg.	*Speedup*	Eficiência	Custo em seg.
1	582,0	1	100%	582,0
2	573,4	1,0150	50,74%	1146,8
4	201,0	2,8955	72,39%	804,0
8	87,2	6,6743	83,42%	697,6
16	41,0	14,1951	88,71%	656,0
32	20,0	29,1000	90,93%	640,0
64	10,8	53,8889	84,20%	691,2

Tabela 1 – Resultados

Análise

Conforme a Tabela 1, nota-se que foram obtidos ganhos, exceto com 2 processos, isto porque, o algoritmo usado é um mestre-escravo, onde o mestre não processa nada além da comunicação, com isso, somente um processo, o escravo, fica responsável de executar, ficando parcialmente sequencial. Outro ponto interessante a se notar é que não há um aumento no custo de processamento relativo com o aumento de processos, consequentemente, isso causa uma queda no tempo, devido à baixa comunicação necessária entre os processos.

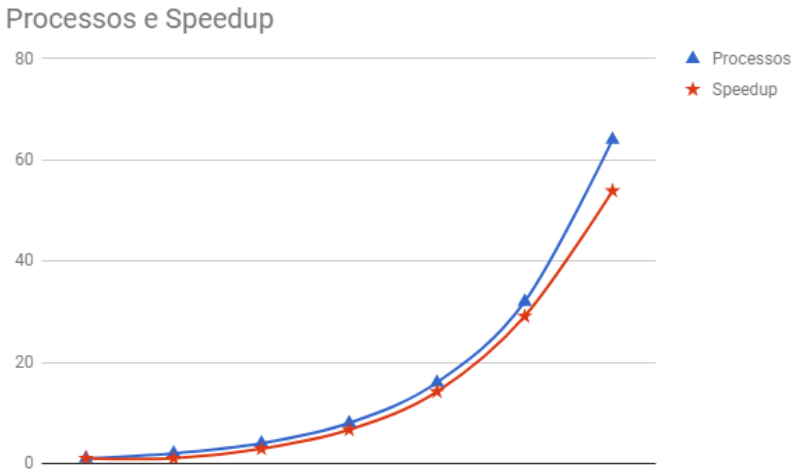
Já a eficiência aumentou, gradativamente, em relação ao aumento do número de processos, apenas começou a diminuir com 64 processos, pelo fato da alta de comunicação entre eles. E para uma melhor visão referente ao *speedup* e à eficiência, os anexos 2 e 3 mostram gráficos dos mesmos, respectivamente.

Considerações finais

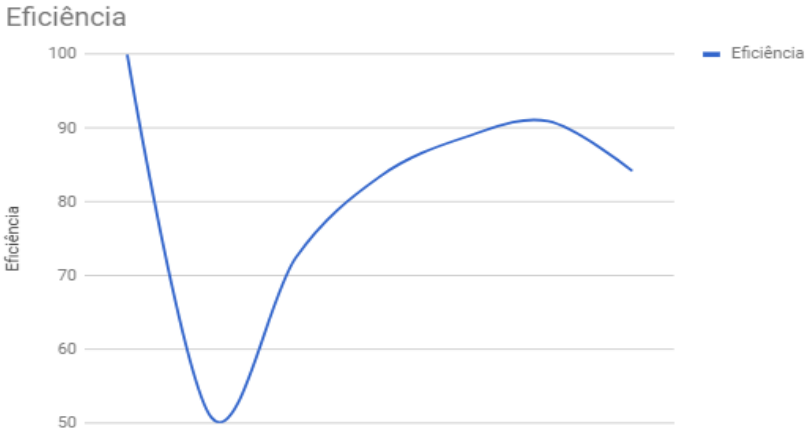
Conclui-se que a utilização do MPI para a paralelização do Mandelbrot Set foi considerada satisfatória, pois não possui um alto custo computacional e possui um tempo de execução baixo dividindo a carga de trabalho em 64 processos.

Nº Processos	Execução 1	Execução 2	Execução 3	Execução 4	Execução 5
2	9m33.691s	9m32.509s	9m32.540s	9m32.551s	9m38.374s
4	3m21.190s	3m21.707s	3m22.025s	3m20.831s	3m21.233s
8	1m28.449s	1m27.051s	1m27.300s	1m27.482s	1m27.048s
16	0m41.079s	0m41.003s	0m41.041s	0m41.393s	0m41.102s
32	0m20.433s	0m20.416s	0m20.262s	0m20.461s	0m20.633s
64	0m14.871s	0m10.234s	0m10.231s	0m10.289s	0m10.266s

Anexo 1 – Tabela com os tempos das 5 execuções paralelas



Anexo 2 – Gráfico de *speedup* com número de processos



Anexo 3 – Gráfico de eficiência