UFGD - Universidade Federal da Grande Dourados
Faculdade de Ciências Exatas e suas Tecnologias
Engenharia de Computação

Alexsander Barbosa Siqueira
Roberto Haruo Honda Junior

Programação Paralela usando MPI

Multiplicação de Matriz por Vetor

Dourados - MS
20 de março de 2017

Introdução

Neste trabalho foi desenvolvido a multiplicação de uma matriz quadrada por um vetor seguindo três modos de implementação uma sequencial, uma paralela por decomposição de linha e outra paralela por decomposição de colunas, comparando-os quanto ao desempenho, algumas características comuns a todas implementação são que todas necessitaram utilizar alocação dinâmica para execução de uma matriz quadrada maior ou igual á 1024 x 1024 e que a contagem do tempo ocorre somente do tempo de computação da multiplicação e da comunicação sem qualquer interferência contagem de quando é gerada ou apresentado os dados.

Multiplicação Matriz por Vetor modo Sequencial:

Para o produto de uma matriz A_{nxn} por um vetor X_n de forma sequencial foram necessários apenas quatro laços de repetição, sendo os dois primeiro para percorrer e gerar os elementos da matriz e do vetor e os dois últimos propriamente para realizar o produto, sendo que nesses dois últimos foram utilizado um vetor para armazenar o resultado e uma variável chamada “soma” para armazenar a soma da multiplicação dos elementos de uma linha da matriz A pelo vetor X que é propriamente um elemento do vetor resultado, essa variável “soma” é atribuída á um elemento do vetor que armazena o resultado da multiplicação da uma matriz A_{nxn} pelo vetor X_n, repetindo n vezes até a multiplicação da matriz pelo vetor gerar um novo vetor, o vetor resultado Y_n, .

Multiplicação Matriz por Vetor modo Linha por Linha:

No produto de uma matriz A_{nxn} por um vetor X_n de forma paralela utilizando a decomposição por linhas, os elementos da matriz e do vetor são gerados no processo 0, o mesmo verifica quantos elementos cada processo terão que calcular, chamado de balanceamento de carga, e para saber esse balanceamento de carga é necessário dividir o total de elementos que a matriz principal possui pela quantidade de processos presentes. Com essa informação de quantos elementos cada processo terá que calcular, tomando o processo 0 como raiz, realiza-se um “MPI_Scatter” para divisão da matriz principal em sub matrizes para cada processo, assim cada processo poderá manipular sua própria matriz local, em seguida também é realizado um “MPI_Bcast” tomando o processo 0 como raiz, onde o vetor X_n é enviado para todos processos que o utilizam para calcular localmente parte da multiplicação da matriz A_{nxn} pelo vetor X_n, gerando assim um novo vetor de resultados locais de cada processo que é enviado para o processo 0 pelo “MPI_Allgather” que irá reunir todos os resultados locais e distribuir o resultado final para todos os processos.

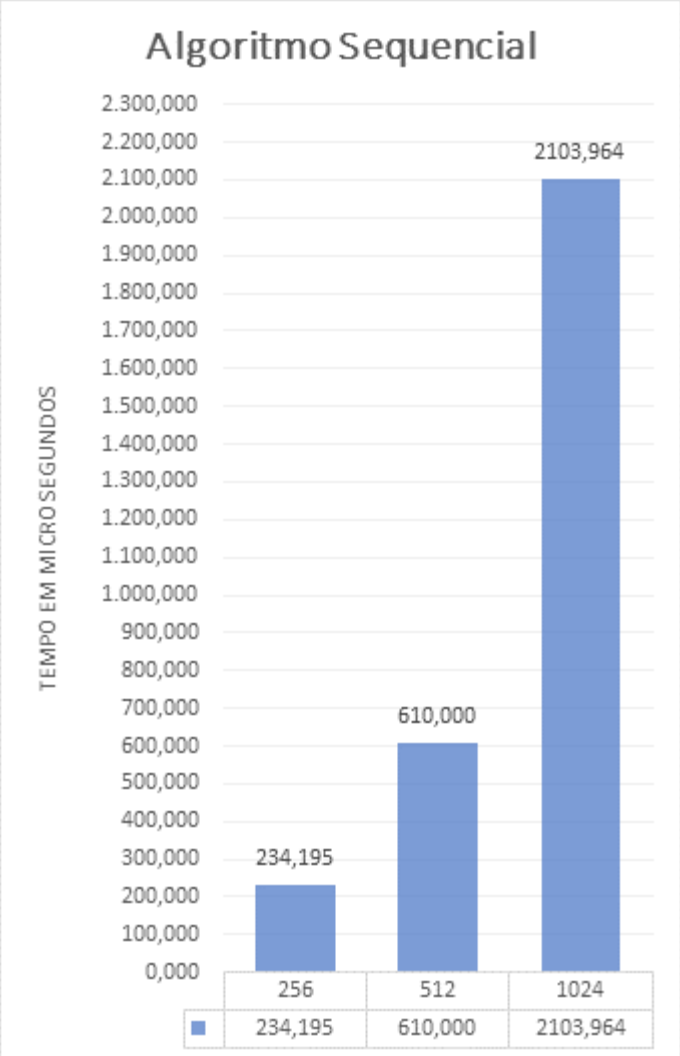
Multiplicação Matriz por Vetor modo coluna por coluna:

Nessa multiplicação, também de forma paralela como a anterior no entanto utilizando a decomposição por colunas, onde os elementos da matriz e do vetor também são gerados no processo 0, e o mesmo faz o balanceamento de carga utilizando a função “MPI_Send” para distribuir os elementos da matriz entre todos processos participantes, utiliza a função “MPI_Scatter” para dividir os elementos do vetor entre os processos, além de também realizar uma multiplicação local assim como os outros processos. Em outro processos, eles se utilizam do “MPI_Recv” para receber a uma matriz local e o “MPI_Scatter” para receber um vetor local, advindos do processo 0, onde estes processos realizam sua multiplicação local da matriz e vetor recebidos. Após estas operações tanto o processo 0 quanto os demais processos utilizam a função “MPI_Allreduce” para o obter o resultado final em cada processo.

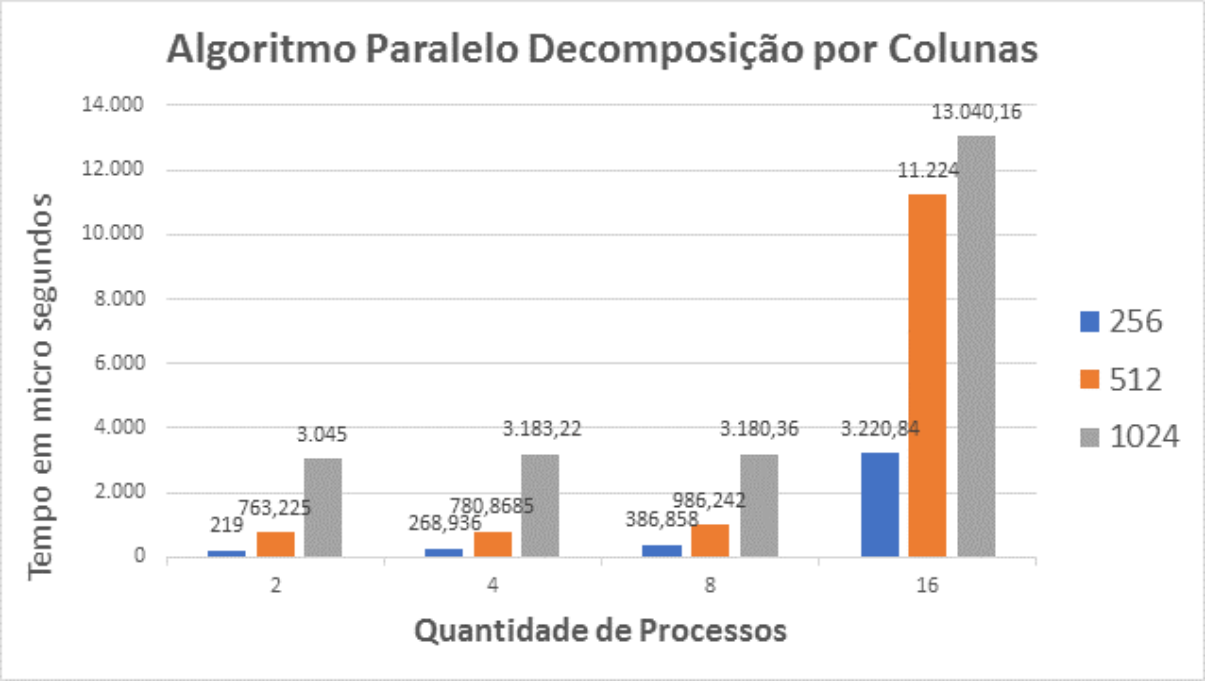
Conclusão - {gráficos e tabelas de testes}

Os tempos das tabelas e gráficos a seguir estão em microssegundos.

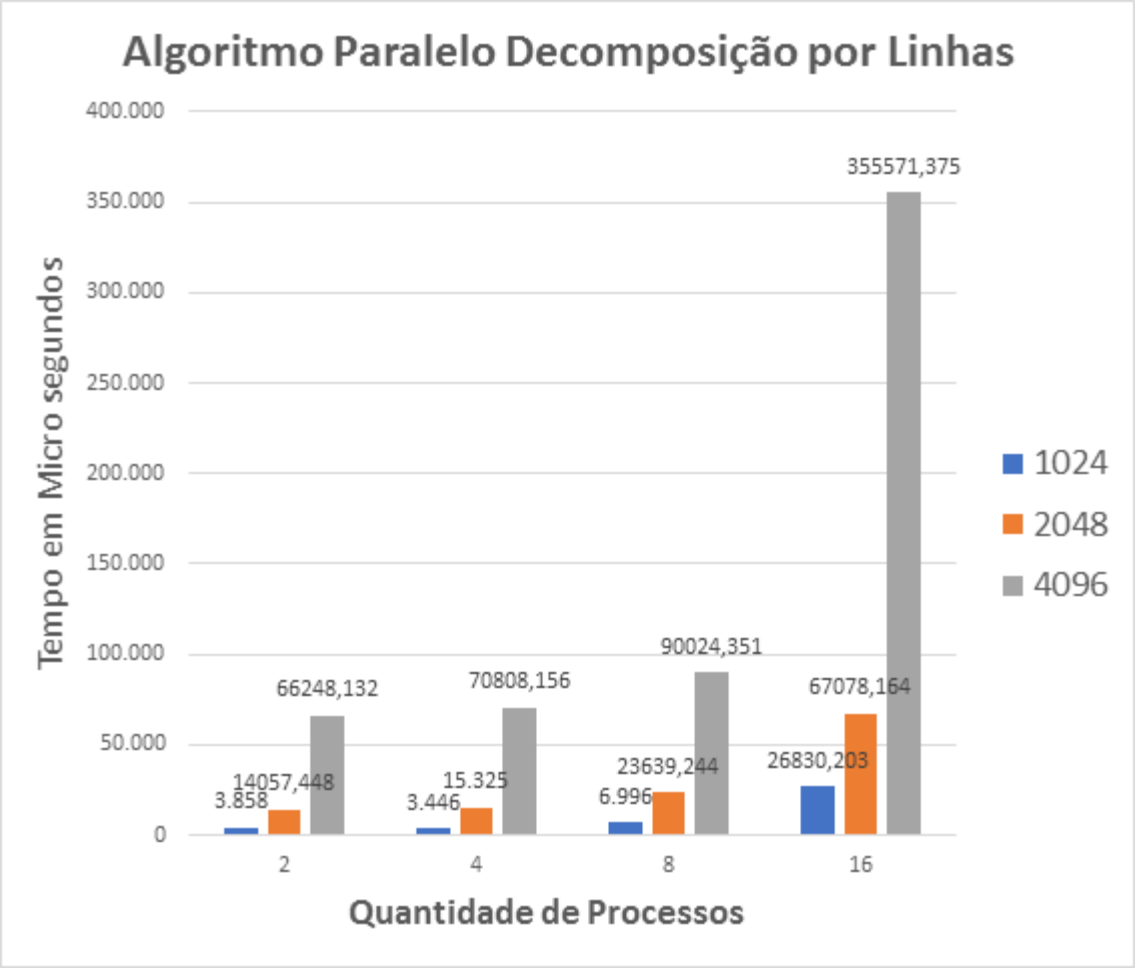
Algoritmo Sequencial	
Tamanho Matriz	Tempo
256	234,195
512	610,000
1024	2103,964



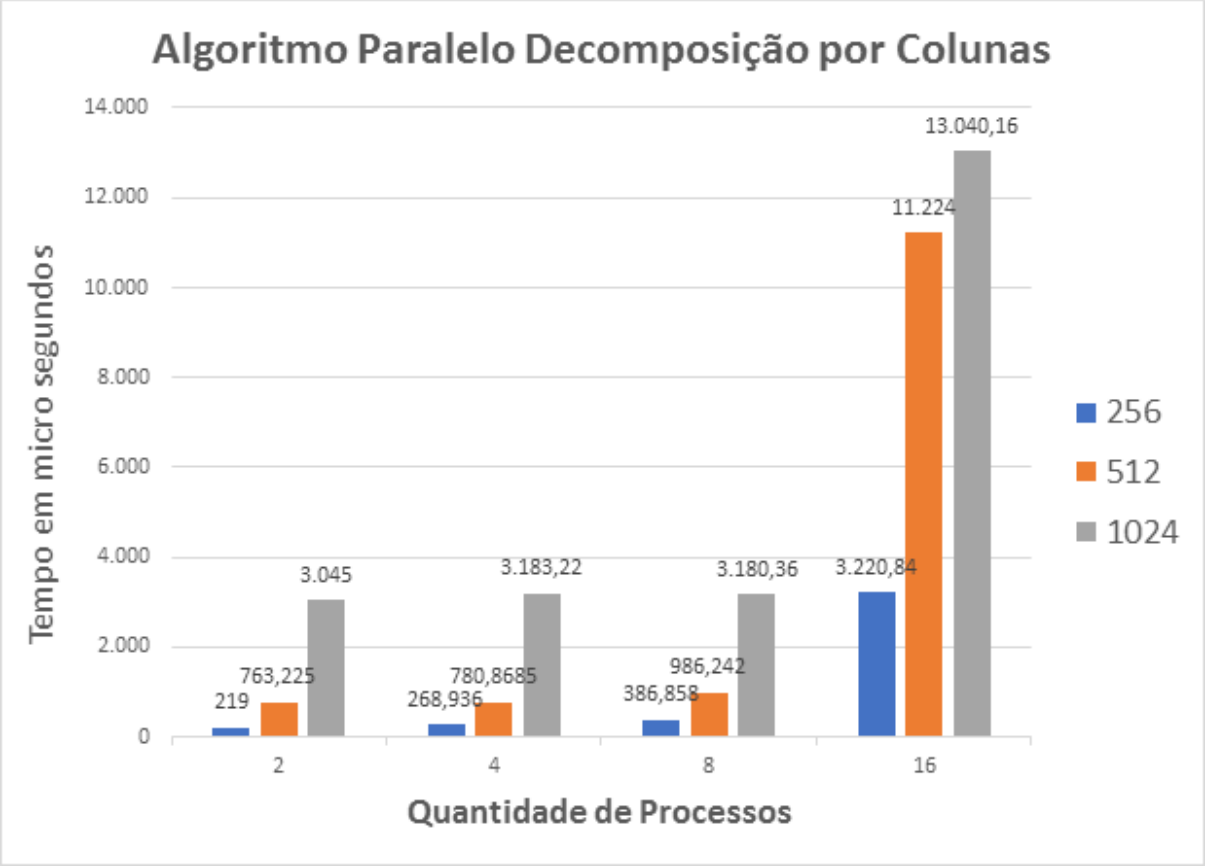
Algoritmo Paralelo decomposição por colunas				
	2	**4**	**8**	**16**
256	219	268,936	386,858	3.220,84
512	763,225	780,8685	986,242	11.224
1024	3.045	3.183,22	3.180,36	13.040,16



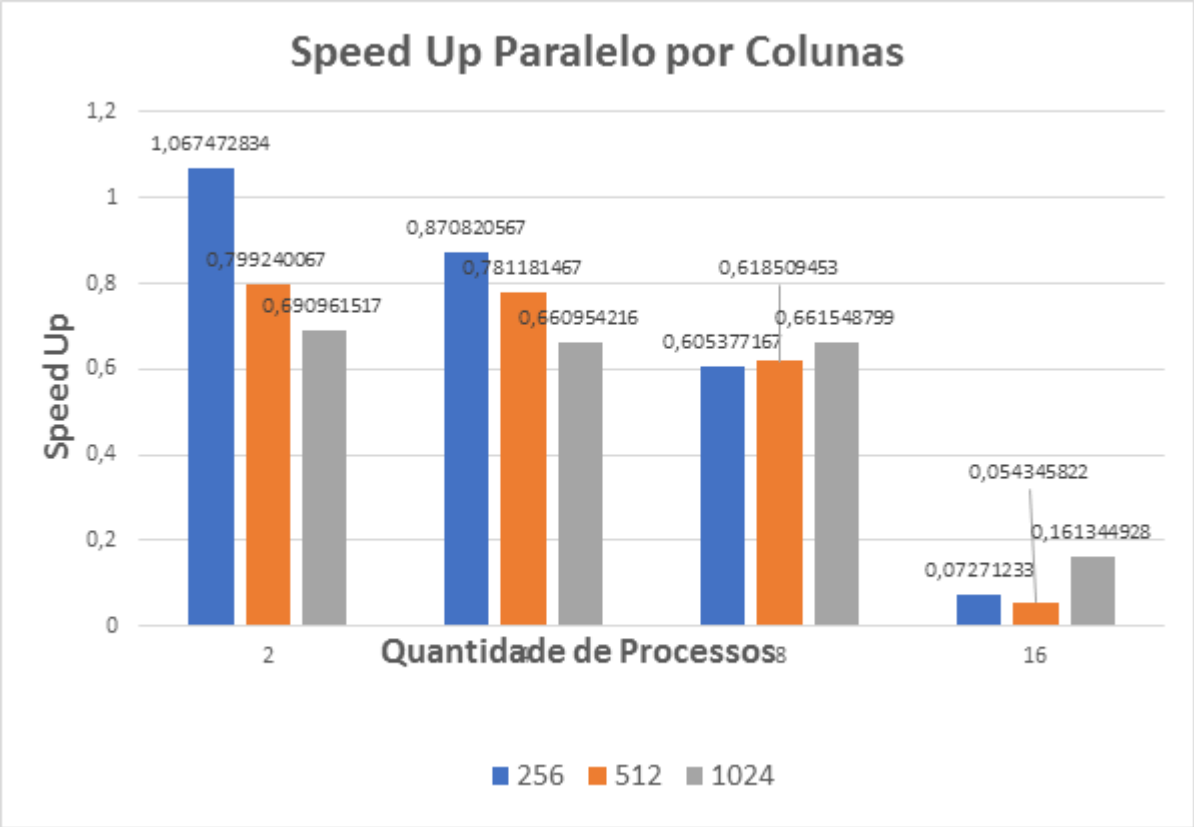
Algoritmo Paralelo decomposição por Linhas				
	2	4	8	16
256	3.858	3.446	6.996	26830,2
512	14057,45	15.325	23639,24	67078,16
1024	66248,13	70808,156	90024,35	355571,4



Algoritmo Paralelo decomposição por Colunas				
	2	4	8	16
256	219	268,936	386,858	3.220,84
512	763,225	780,8685	986,242	11.224
1024	3.045	3.183,22	3.180,36	13.040,16



	Speed Up Paralelo por Colunas			
	2	4	8	16
256	1,067472834	0,870820567	0,605377167	0,07271233
512	0,799240067	0,781181467	0,618509453	0,054345822
1024	0,690961517	0,660954216	0,661548799	0,161344928



	Speed Up Paralelo por Linhas			
	2	4	8	16
256	0,060700837	0,067961286	0,033473677	0,008728782
512	0,043393367	0,039802948	0,025804548	0,009093868
1024	0,031758843	0,029713583	0,023371054	0,005917135