UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Câmpus de São José do Rio Preto

RELATÓRIO DE PROGRAMAÇÃO CONCORRENTE

Multiplicação de Matrizes com o algoritmo SUMMA

Utilizando OpenMPI

Alunos:

Gabriel Henrique Martinez Saraiva

Leandro Moreira Barbosa

Professor:

Dr. Aleardo Manacero Jr.

Introdução

Neste trabalho, o algoritmo de multiplicação de matrizes SUMMA (Scalable Universal Matrix Multiplication Algorithm) foi analisado, implementado em linguagem C utilizando o OpenMPI, e foi testando utilizando matrizes quadradas para avaliar se existe ganho de desempenho ao utilizar MPI em ambientes distribuídos.

Objetivos

O objetivo do trabalho, principalmente é ter um contato mais intimo com a ferramenta MPI, e avaliar o ganho de desempenho que é obtido ao utilizar diversos computadores para a realização de uma mesma tarefa.

Metodologia

Para a execução desse projeto foi implementado o programa multiplicador de matrizes utilizando do algoritmo SUMMA de forma distribuiída com passagem de mensagens, utilizando a biblioteca OpenMPI.

Para a medição de tempo foram utilizadas duas ferramentas, a primeira foi a função ***time*** da biblioteca ***time.h***, na linguagem C que apresenta a resolução de tempo na

ordem de segundos. Utilizando essa ferramenta foi obtido a métrica “Tempo de Cálculo” que é a soma do tempo de comunicação para transmissão dos dados a serem computados, o tempo de da multiplicação das matrizes, e o tempo de devolução e redução dos resultados.

A segunda ferramenta utilizada para fazer a medição de tempo de execução foi o comando **date** do Gnu/Linux, que também fornece a resolução de tempo na ordem de segundos. Com essa ferramente foi obtida a métrica Tempo Total, que mostra o tempo que o processo todo levou para ser executado, desde sua criação até sua conclusão. Nesse tempo além dos valores do Tempo de Cálculo também estão inclusos o tempo de alocação e inicialização das matrizes e tratamento de parâmetros.

Para executar os testes foram feitas duas categorias diferentes de testes. A primeira utilizando um computador pessoal. Um notebook Microboard MUB342, com um Intel Core 2 Duo de 1.6GHz (T5450) e 4GB DDR2, utilizando o Arch Linux 64 bits com kernel 3.12.5-1 compilado com suporte a SMP, para analisar se existe ganho de desempenho em ambientes de memória convencional, como acontece no uso de outras técnicas como *threads*, por exemplo.

A segunda categoria foi utilizando um cluster heterogêneo composto de 8 nós rápidos (Core i7, 16GB de memória DDR3 cada) e 8 nós lentos (Intel Pentium D com 4GB de memória cada um).

Teste 1

Como pode-se ver abaixo nos gráficos o uso de OpenMPI em ambientes não distribuídos não se mostra tão vantajoso como o uso de Threads por exemplo, que foi objeto de estudo do primeiro projeto dessa disciplina.

No gráfico da figura 1 pode ser visto a variação dos tempos de execução da multiplicação de matrizes relativamente pequenas, não mostrando grandes ganhos no uso dessa ferramenta nesse caso.

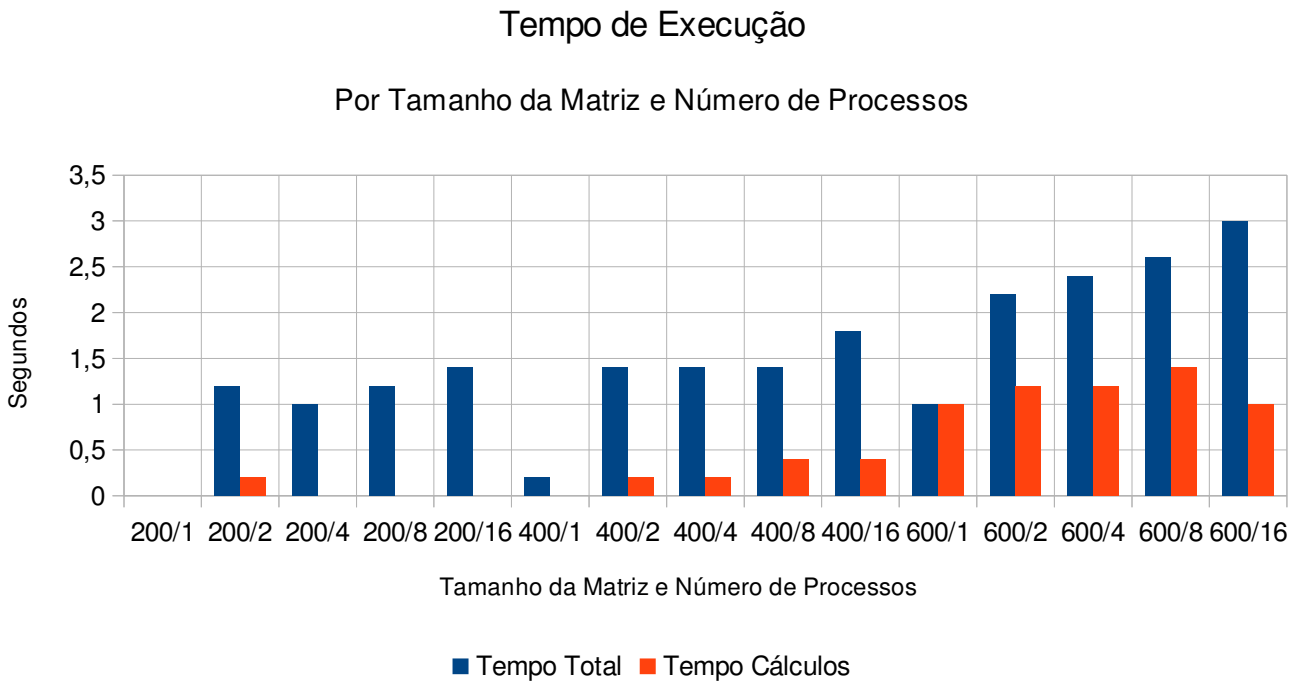

Figura 1: Execução da multiplicação das matrizes de ordem 200 a 600, com 1, 2, 4, 8 e 16 processos em sistema de memória convencional

É possível observar certas anomalias na figura 1 como por exemplo o primeiro

resultado, que apenas um processo foi utilizado para multiplicar duas matrizes de ordem 200, e foi executado em tempo 0s. Esse caso e outros semelhantes incluindo casos posteriores onde o Tempo Total e o Tempo de Cálculo são incompativeis, apresentando por exemplo o Tempo de Cálculo maior que o Tempo Total, são derivados das ferramentas de mensuração que foram utilizadas, devendo por tanto serem considerados frutos erros de precisão das ferramentas.

Na figura 2 é possível observar a execução de matrizes maiores, , onde é possível observar que ainda não existe grande vantagem no uso dessa ferramenta em ambientes de memória convencional.

Outro ponto que pode ter afetado em grande parte os resultados no ambiente de memória convencional é o equiapamento que foi utilizado, no primeiro teste, que é um notebook com apenas 2 núcleos.

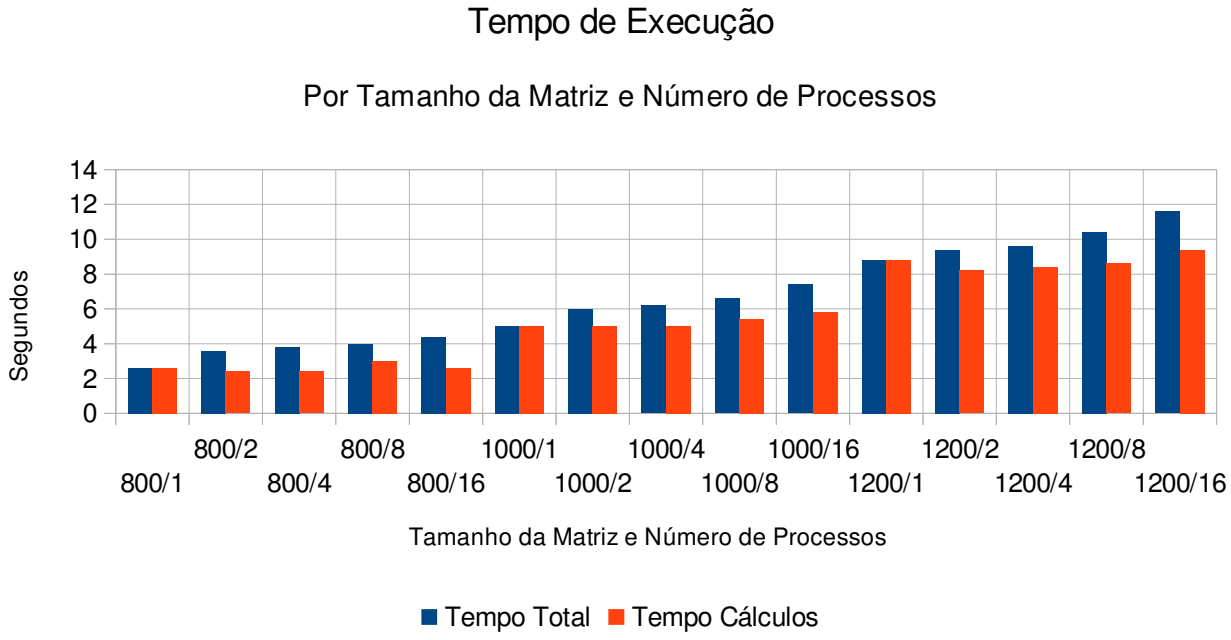

Figura 2: Execução da multiplicação das matrizes de ordem 800 a 1200, com 1, 2, 4, 8 e 16 processos em sistema de memória convencional.

Como na figura 2, os tempos apresentados são maiores que na figura 1, os erros de precisão das ferramentas de medição, são menos expressivos nos resultados finais.

Por fim, também pode ser observado na figura 3, a ultima bateria de testes dessa categoria, utilizando matrizes de odem de 1400 a 2000. Que também não apresenta vantagens no uso desse tipo de tecnologia em ambientes de memória compartilhada.

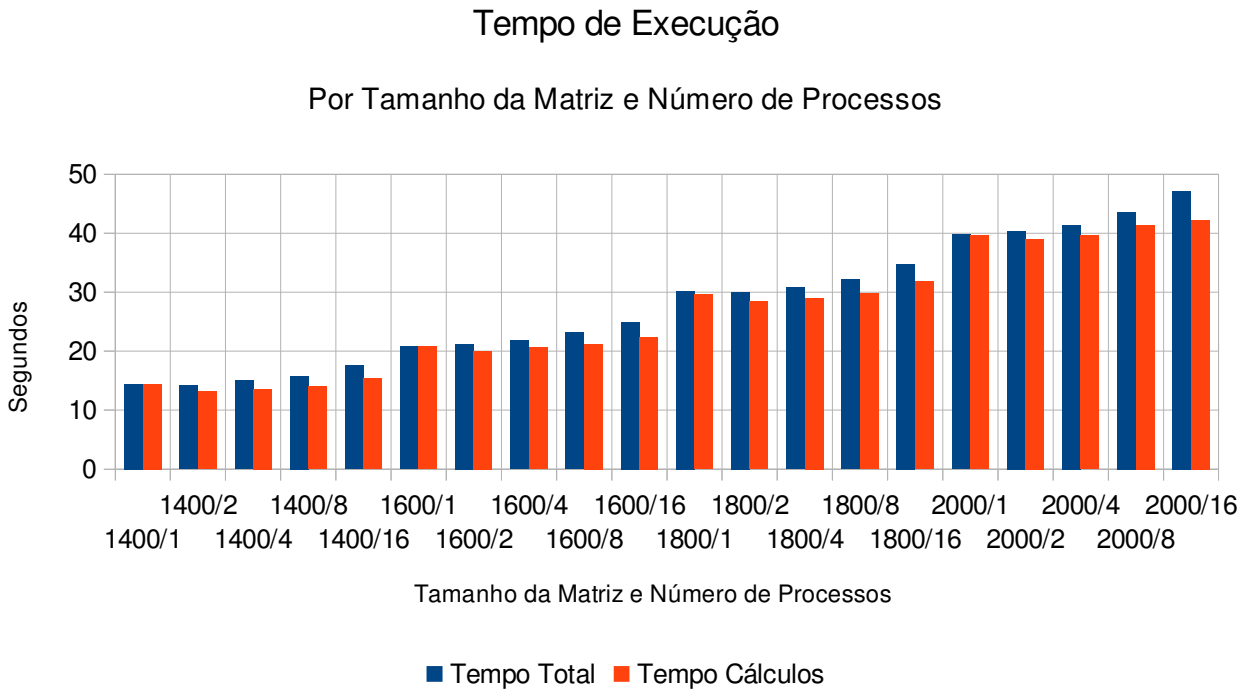

Figura 3: Tempo de execução da multiplicação das matrizes de 1400 a 2000 no embiante de memória convencional

Teste 2

Diferente do cenário do primeiro teste, quando passamos a utilizar ambientes de memória distribuída, que é para o qual o MPI foi desenvolvido, podemos notar grande ganho no tempo de execução das tarefas. É importante reforçar aqui que a diferença de tempo na execução é fruto não somente da mudança de ambiente, mas principalmente da capacidade dos equipamentos envolvidos.

Na figura 4 é possível já observar que os tempos são bem menores e o gráfico tem o formato de "serra" mostrando o ganho no uso de vários processos em diferentes computadores para a conclusão da multiplicação das matrizes.

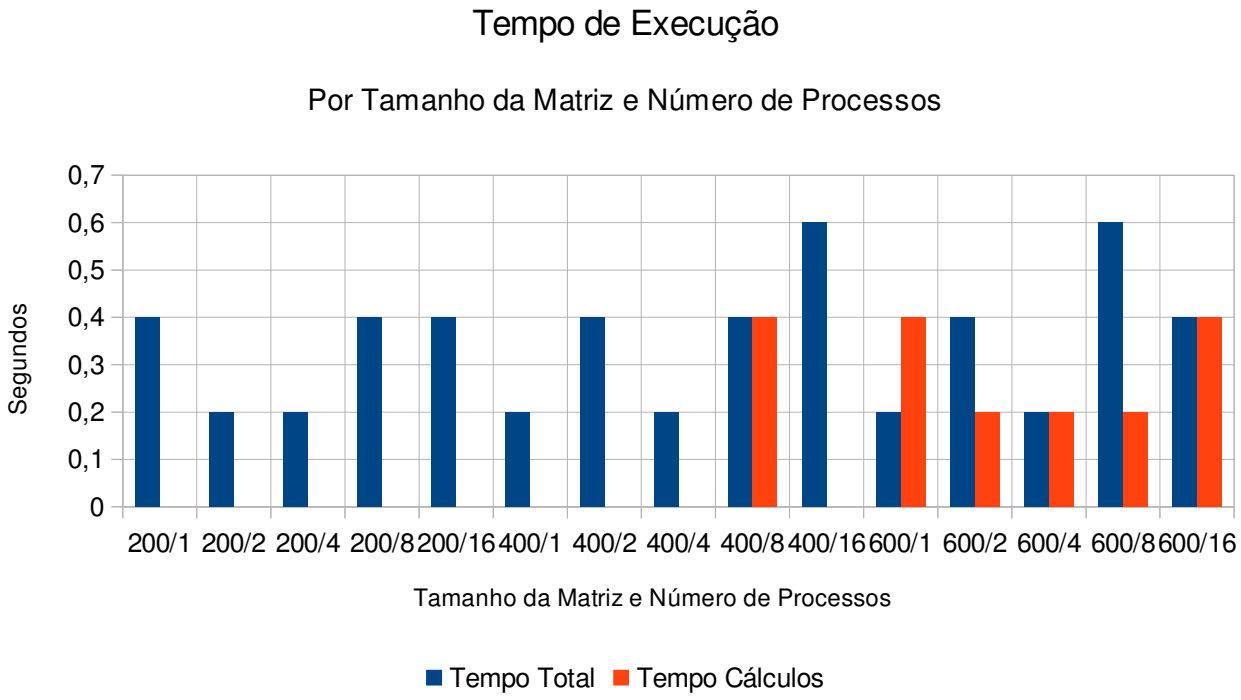

Figura 4: Execução da multiplicação das matrizes de ordem 200 a 600 em ambientes de memória distribuída.

Na figura 5 fica mais evidente ainda o formato de “serra” que o gráfico adquire. Vale uma ressaltar nesse ponto o acrescimo do tempo ao utilizar 16 processos. Isso é devido ao uso dos 8 nós lentos do cluster, o que obriga os outros 8 nós rápidos a esperarem para passarem pelas barreiras de sincronismo. Esse gargalo pode ser melhor explorado em uma outra oportunidade.

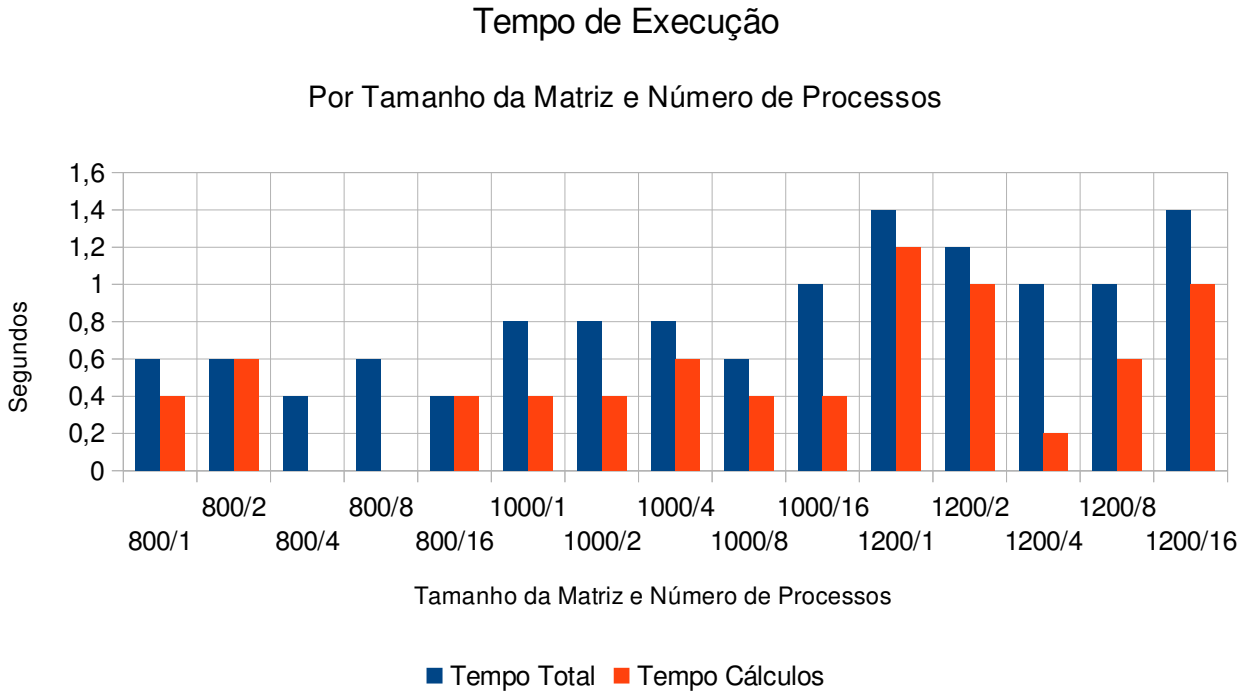

Figura 5: Execução da multiplicação das matrizes de ordem 800 a 1200 em ambientes de memória distribuída.

Na figura 4 e na figura 5 é possível notar que devído ao baixissimo tempo de exeução as ferramentas de medição de tempo não se mostram adequadas a esse tipo de cenário. Portanto não a utilizem em casos semelhantes. (Até mesmo um mal resultado é um

resultado)

Para completar esses resultados na figura 6 é possível observar o grafico que mostra o tempo de execução da multiplicação das maiores matrizes testadas, sendo elas de ordem de 1200 a 2000, não mostraram ainda grande desafio ao novo paradigma e nem ao equipamento utilizado.

Nota-se que no grafico da figura 6 é possível observar bem o ganho de desempenho ao se utilizar mais processos em computadores distintos e também o gargalo causado pelos 8 nós mais lentros do cluster.

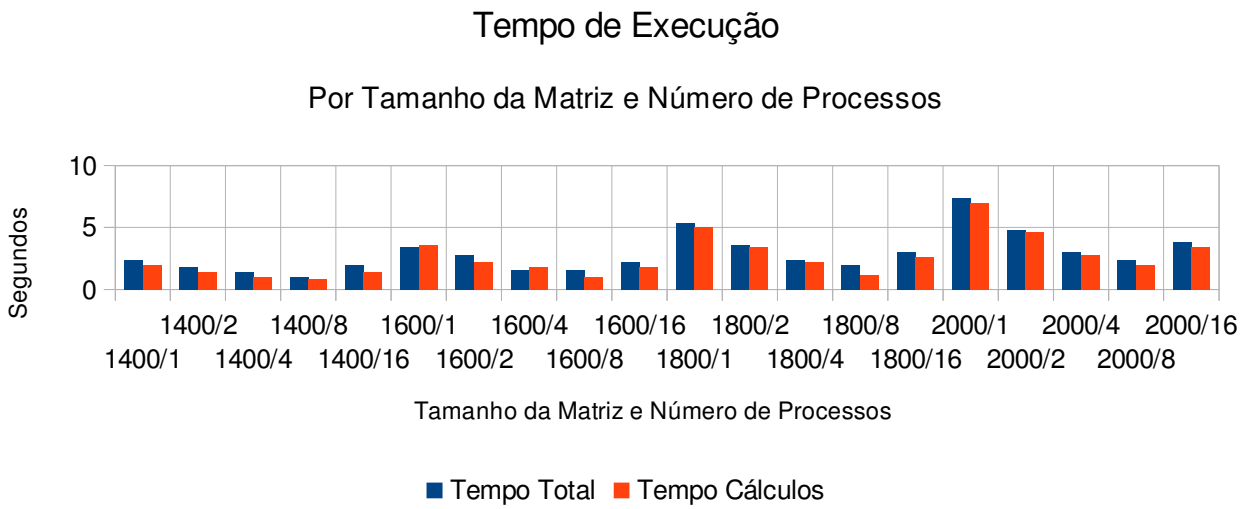

Figura 6: Mostra o tempo de execução das matrizes de ordem 1400 a 2000 no ambiente de memória distribuída.

Discussão

Com base em todos os testes feitos (e outros muitos ainda podem ser feitos como por exemplo o uso de um computador do Cluster, de forma isolada para executar os testes de MPI em memória convencional, que só foi pensador por nós na conclusão do trabalho) foi possível perceber que o uso dessa nova ferramenta facilita muito o processo de programação paralela e fornece um ótimo meio de conseguir explorar o potencial de clusters e grids, uma vez que oculta do programado quase toda parte de comunicação e uso de sockets, o que em linguagem C provavelmente desencorajaria muitos a se aventurar nesse nicho, que já se mostra muito deficitário em mão de obra.

Além disso essa ferramenta nos traz facilidade para explorar o poder computacional nas grades e em clusters, que muitas vezes podem ser muito mais fáceis de manter e escalar do que um computador com múitos núcleos e processadores individuais, onde não é possível inserir mais núcleos para executar mais threads ao mesmo tempo.