Universidade Federal do Rio Grande do Norte

Instituto Metrópole Digital

Análise e comparação de speedup e eficiência do código serial e paralelo no problema da multiplicação de matrizes quadradas

João Vitor Venceslau Coelho

Natal/RN
2020

Introdução

Este relatório consiste na explicação da ***Multiplicação de Matrizes Quadradas***, essa explicação busca deixar claros todos os detalhes do problema e das soluções utilizadas, sendo elas: *multiplicar* ***sequencialmente*** as linhas da primeira matriz com as colunas da segunda matriz, multiplicar ***aleatoriamente*** as linhas da primeira matriz com as colunas da segunda matriz e multiplicar sequencialmente as linhas da primeira matriz pelas linhas da segunda matriz ***transposta***. As soluções implementadas também serão explicadas, isto é, a versão serial e a paralela dos códigos implementados para cada uma das soluções dos problemas abordados será explanada, em seguida os resultados obtidos serão mostrados e comentados. Será também brevemente comentada a corretude dos algoritmos implementados, serão expostas as análises do speedup, da eficiência e da escalabilidade dos algoritmos paralelos para os seriais. Ao final, um resumo das atividades e análises realizadas será relatado, sintetizando os principais pontos do relatório.

Multiplicação de Matrizes Quadradas

A multiplicação de matrizes é uma operação entre matrizes que só é definida caso o número de colunas da primeira matriz seja igual ao número de linhas da segunda matriz. Dadas duas matrizes A_{mxn} e B_{nxo}, cada elemento c_{ij} da matriz resultante C_{mxo} é obtido com a soma das multiplicações dos elementos correspondentes da linha i da matriz A com os elementos da coluna j da matriz B. A multiplicação de matrizes quadradas é igual a multiplicação das matrizes não quadradas, desde que respeite a condição para a definição, dada acima, da operação. Logo a multiplicação de matrizes quadradas só está definida para matrizes quadradas de mesma ordem (onde ordem seria o número de linhas e colunas da matriz).

Desenvolvimento

Abaixo são explicados os algoritmos implementados, tanto as versões seriais quanto as paralelas das soluções utilizadas, sendo as soluções:

- Multiplicar matrizes quadradas sequencialmente nas linhas da primeira matriz;
- Multiplicar matrizes quadradas aleatoriamente nas linhas da primeira matriz;
- Multiplicar matrizes quadradas sequencialmente nas linhas da primeira matriz transpondo a segunda matriz.

O funcionamento dos dois primeiros algoritmos é bem similar, o segundo só tem uma etapa a mais em relação ao primeiro, que é definir a sequência aleatória em que as linhas da primeira matriz serão usadas, e o terceiro algoritmo possui mais diferenças, pois nele a

transposição da segunda matriz tem que ser calculada e a multiplicação ocorrerá entre as linhas da primeira com as linhas da segunda transposta.

Soluções implementadas

Multiplicar Matrizes Quadradas Sequencialmente nas linhas (Serial)

O algoritmo inicia com a leitura dos argumentos a serem utilizados, sendo eles:

1. Flag que indica se as matrizes devem ser exibidas ao fim da multiplicação (0 ou 1);
2. Seed a ser utilizada para preencher a primeira matriz;
3. Seed a ser utilizada para preencher a segunda matriz;
4. Quantidade de linhas da primeira matriz;
5. Quantidade de colunas da primeira matriz;
6. Quantidade de linhas da segunda matriz;
7. Quantidade de colunas da segunda matriz;

Dito isso, o algoritmo foi feito de forma que realiza a multiplicação de matrizes mesmo quando as mesmas não são quadradas, desde que satisfaçam a condição para definição da operação de multiplicação definida anteriormente, caso essa condição não seja satisfeita, uma mensagem é exibida e o programa finaliza. Após a verificação da condição, com base nas quantidades de linhas e colunas informadas, as devidas matrizes são alocadas e depois utilizando as seeds, obtidas anteriormente, preenche-se as matrizes a serem multiplicadas. Após realizar essas etapas, a multiplicação de fato se inicia. Para cada linha da primeira matriz é calculada a linha correspondente da matriz resultante da multiplicação, sendo cada elemento dessa linha calculado, somando o resultado das multiplicação de cada elemento j da linha atual com o elemento correspondente na coluna j da segunda matriz. Após percorrer todas as linhas da primeira matriz, todas as linhas da matriz resultante da multiplicação são devidamente calculadas. Após ter calculado a multiplicação, caso a flag que indica a exibição esteja igual a 1, é mostrado as matrizes usadas na multiplicação e o resultado obtido. Por fim, libera-se a memória alocada para armazenar as matrizes.

Multiplicar Matrizes Quadradas Sequencialmente nas linhas (Paralelo)

A versão paralela do algoritmo descrito anteriormente, possui um único argumento a mais em relação aos que foram listados, que é a quantidade de threads a serem utilizadas pelo programa, e, vale salientar que esse argumento é informado antes da flag de exibição. Uma nova condição imposta no algoritmo implementado é que o número de threads solicitadas deva dividir sem resto o número de linhas da primeira matriz. A tarefa alvo da paralelização é o cálculo das linhas equivalentes da matriz resultante da multiplicação, a abordagem utilizada foi da delegação para cada thread criada calcular o resultado da multiplicação utilizando um determinado número de linhas da primeira matriz. Sendo que cada thread

calcula quais linhas deve utilizar baseando-se no seu rank, obtido no momento de sua criação. A thread principal do programa não participa do cálculo, apenas cria as threads e as espera finalizar, para então seguir com as mesmas tarefas da versão serial.

Multiplicar Matrizes Quadradas Aleatoriamente nas linhas (Serial)

Esta versão serial tem as mesmas etapas da versão serial anterior, sendo que no momento de percorrer as linhas da primeira matriz e calcular o resultado da multiplicação na linha equivalente na matriz resultante, esse percurso não é realizado *sequencialmente* a partir do índice 0. Previamente, é criado um array com os índices das linhas da primeira matriz e então ele é desordenado, utilizando um sequência de substituições. A ordem final utilizada para calcular as linhas da matriz resultante é a indicada pelos valores desse array. Essa é a única diferença entre os algoritmos.

Multiplicar Matrizes Quadradas Usando Transposição (Serial)

Na versão serial com transposição são feitas algumas modificações nas etapas descritas na primeira versão serial, são elas: a alocação de uma nova matriz para armazenar o resultado da transposição da segunda matriz, a transposição da segunda matriz antes de realizar a multiplicação, a multiplicação entre as matrizes agora ocorre entre as linhas da primeira matriz e as linhas da segunda matriz transposta, e ocorre a liberação da memória alocada para armazenar a segunda matriz transposta.

A memória usada para armazenar a segunda matriz transposta é alocada após a alocação das outras matrizes. A transposição é realizada da seguinte maneira: para cada elemento e_{ij} , sendo i o índice da linha na matriz original e j o índice da coluna, o valor do elemento t_{ji} na matriz alocada para armazenar a transposta, será igual ao elemento e_{ij} . Assim, após percorrer todos os elementos da matriz original e os posicionar na matriz alocada para armazenar a transposição, inicia-se a multiplicação entre as matrizes. Para isso são percorridas as linhas da primeira matriz e é calculada a linha correspondente da matriz resultante da multiplicação. Ressalta-se que, em relação a versão serial anterior, a diferença ocorre no momento de calcular cada elemento, pois em vez de multiplicar os elementos da linha da primeira matriz com a coluna da segunda matriz, realiza-se a multiplicação entre os elementos da linha da primeira matriz com os elementos das linhas da segunda matriz, uma vez que, por ela ter sido transposta, as linhas dela correspondem às colunas da matriz original. Por conseguinte, o restante do algoritmo é o mesmo da versão serial apresentada inicialmente, com a adição da liberação da memória da matriz transposta junto da liberação das outras matrizes ao fim do código.

Multiplicar Matrizes Quadradas Usando Transposição (Paralelo)

Essa versão segue a mesma estrutura da versão paralela, à priori apresentada, com a adição dos elementos descritos na versão serial com transposição, isto é, a alocação da matriz transposta, a transposição da segunda matriz, a alteração na multiplicação para ocorrer

entre linhas da primeira matriz e linhas da segunda matriz transposta e a liberação da matriz transposta. O ponto que vale destacar é: a paralelização da operação de transposição. Essa paralelização permite que a troca das posições dos elementos de uma linha sejam realizados paralelamente, onde cada thread troca um certo números de linhas por colunas. Assim, uma nova condição é imposta ao algoritmo, essa condição é que a quantidade de linhas das matrizes deve ser divisível pelo número de threads solicitadas, pois cada thread calcula quais colunas deve trocar por linhas baseando-se no seu rank e no resultado dessa divisão.

Resultados encontrados

Corretude dos algoritmos implementados (Todos)

A corretude dos algoritmos implementados pode ser verificada com a seguinte instância do problema, usando os seguintes argumentos:

- Nas versões seriais: **1 0 42 2 2 2 2**
- Nas versões paralelas: **2 1 0 42 2 2 2 2**

Obtém-se o seguinte resultado em todas as implementações feitas:

Primeira Matriz

0.840188	0.394383
0.783099	0.798440

Segunda Matriz

0.033470	0.329964
0.690636	0.422487

Matriz Resultante

0.300496	0.443853
0.577641	0.595725

Ao verificar as contas realizadas, é visto que o resultado difere apenas em alguns dígitos, isso é facilmente explicável, afinal os números exibidos na tela são apenas uma aproximação dos números que são de fato usados internamente, muitas vezes arredondando os valores, pois apenas 6 dígitos de precisão foram utilizados. Segue a verificação das contas:

$$e_{11} = 0.840188 \cdot 0.033470 + 0.394383 \cdot 0.690636 = 0.30049618994$$
$$e_{12} = 0.840188 \cdot 0.329964 + 0.394383 \cdot 0.422487 = 0.44385348375$$
$$e_{21} = 0.783099 \cdot 0.033470 + 0.798440 \cdot 0.690636 = 0.57764173137$$
$$e_{22} = 0.783099 \cdot 0.329964 + 0.798440 \cdot 0.422487 = 0.59572499871$$

Análise de Speedup, Eficiência e Escalabilidade

Abaixo estão as tabelas com as médias dos tempos válidos obtidos, sendo que para cada instância analisada, foram coletados 15 tempos, em que os dois maiores e os dois menores foram desconsiderados como válidos para o cálculo dessa média.

Multiplicar Matrizes Quadradas Sequencialmente nas Linhas

Comparando a versão serial que percorre *sequencialmente* as linhas da primeira matriz com a versão serial que percorre *aleatoriamente* essas mesmas linhas, pode-se observar um comportamento não esperado, o tempo obtido é muito semelhante, enquanto o esperado é que pelo princípio da localidade, o custo de percorrer as linhas de forma aleatória seria maior, visto que os dados estariam mais distantes na estrutura da hierarquia de memória do que quando percorre-se as linhas de forma sequencial, onde os dados costumam estar sempre próximos, seguem os tempos obtidos:

Tamanho do Problema	Serial	Serial Random Line
1440	32,285556	32,304444
1600	47,364444	47,326667
1760	69,073333	68,745556
1920	98,676667	98,552222
2080	144,020000	140,810000

Tabela 1: Média dos tempos das versões seriais sequencialmente nas linhas e aleatoriamente nas linhas em cada um dos tamanhos de problema utilizados.

Em relação ao ganho de tempo com o uso da versão paralela pode-se perceber uma diminuição considerável no tempo gasto para o cálculo da multiplicação. Seguem agora os tempos entre a versão serial e a paralela, seguindo a ordem das linhas e sem transposição:

Tamanho do Problema	Serial	4	8	16	32
1440	32,285556	7,782344	4,042211	2,065465	1,118365
1600	47,364444	11,176804	5,750048	2,944502	1,527971
1760	69,073333	14,901650	7,945884	4,012879	2,129984
1920	98,676667	20,883314	10,779094	5,431646	2,811986
2080	144,020000	27,305537	14,697001	7,470707	4,022237

Tabela 2: Apresenta os tempos médios obtidos nos experimentos por tamanho do problema e algoritmo utilizado, assim como a quantidade de cores na versão paralela.

Abaixo seguem as análises e comparações, de speedup e eficiência, das versões paralelas do algoritmo de multiplicação das matrizes percorrendo *sequencialmente* as linhas, sem utilizar transposição na segunda matriz, e seus devidos gráficos.

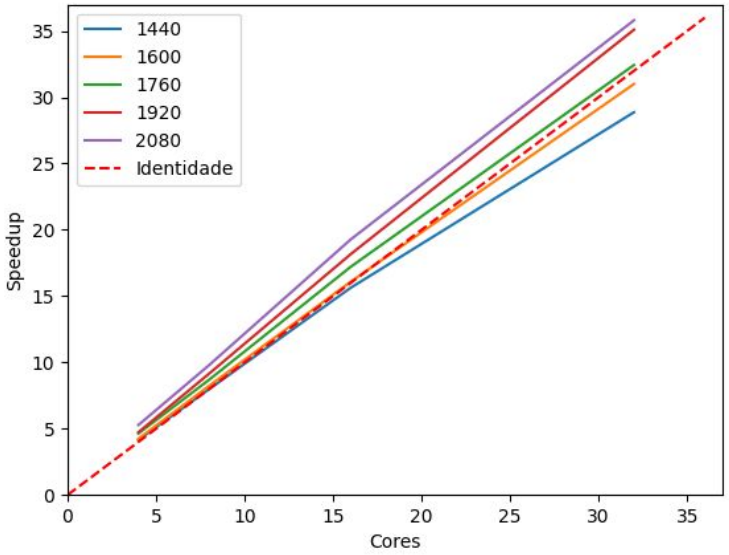

Figura 1: Comparações do speedup em cada um dos tamanhos do problema com o speedup ideal. É interessante que, com o aumento do tamanho do problema e o aumento do número de cores a diferença entre os speedups para cada problema foi aumentando.

Esse comportamento de speedup superlinear não é o esperado, pode ter sido causado por algum detalhe do nó utilizado no supercomputador durante a execução do experimento. Novos testes precisam ser executados para validar esse comportamento. Seguem agora os gráficos sobre a eficiência.

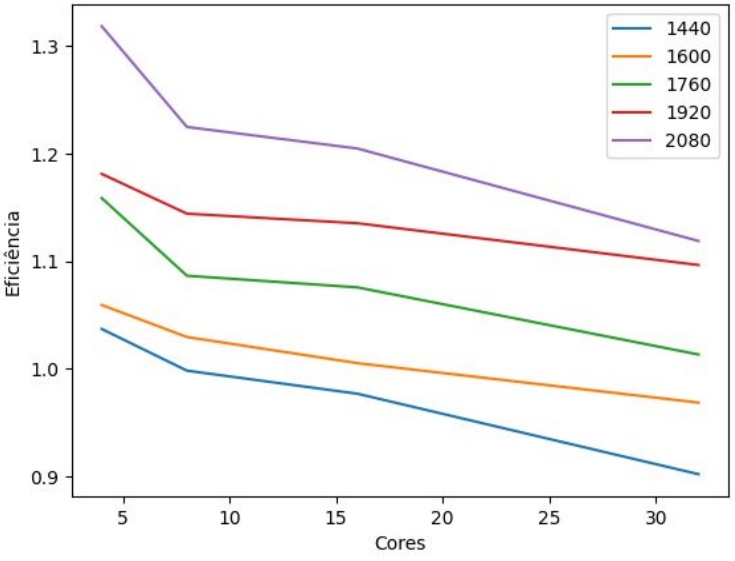

Figura 2: Apresenta a eficiência por cores em cada tamanho de problema utilizado e nele pode-se observar um comportamento comum, a diminuição da eficiência com o aumento do número de cores usados.

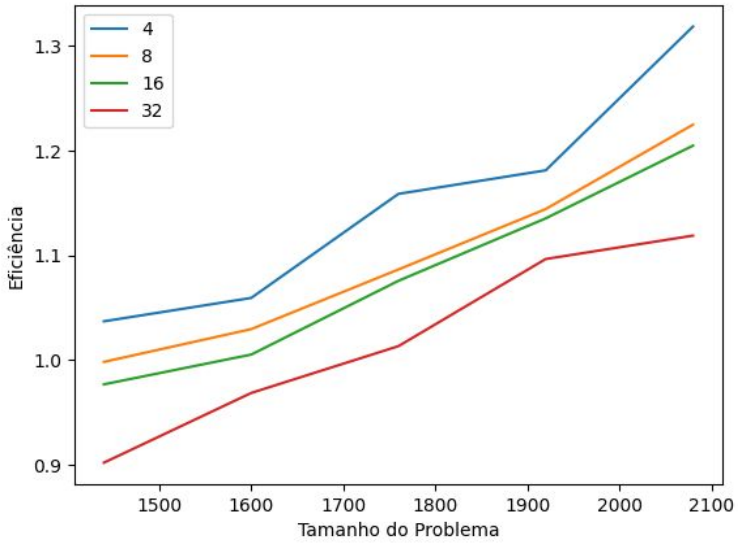

Figura 3: Apresenta a eficiência por tamanho de problema em cada número de cores utilizados, pode-se perceber que com o aumento do tamanho do problema, a eficiência aumenta, principalmente quando a quantidade de cores é menor.

Diante desses dados, é concluído que a versão da multiplicação sequencialmente nas linhas se categoriza como **Fracamente Escalável**, pois aumentando o tamanho do problema e o número de cores em proporções iguais, a eficiência se comporta de forma crescente, com algumas pequenas flutuações.

Multiplicar Matrizes Quadradas Usando Transposição

Ao realizar a *transposição* da segunda matriz e adaptando a lógica da multiplicação para linhas com linhas, obtém-se uma diminuição considerável no tempo gasto para realizar a multiplicação das matrizes, pois o princípio da localidade é melhor explorado, visto que durante a multiplicação são acessados apenas elementos que estão próximos uns dos outros e, como uma matriz é armazenada de forma contígua na memória, acessar elementos em linhas diferentes, mas numa mesma coluna é mais custoso, pois estão mais distantes do que o elemento na mesma linha, porém em outra coluna. Isso se dá pela forma como a hierarquia de memória é estruturada. Essa diminuição no tempo gasto é visível nos tempos exibidos na tabela abaixo.

Tamanho do Problema	Serial Transpose	4	8	16	32
1440	13,634444	3,493499	1,808268	0,957964	0,525259
1600	18,590000	4,762809	2,445943	1,296578	0,744688
1760	25,557778	6,313952	3,248057	1,695804	0,937431
1920	33,611111	8,235728	4,195999	2,194620	1,179988
2080	44,483333	10,369769	5,298842	2,752346	1,485041

Tabela 3: Apresenta os tempos médios obtidos nos experimentos com a versão utilizando *transposição* por tamanho do problema e algoritmo empregado, assim como a quantidade de cores na versão paralela.

Abaixo seguem as análises e comparações de speedup e eficiência das versões paralelas do algoritmo de multiplicação utilizando a transposição da segunda matriz e seus devidos gráficos.

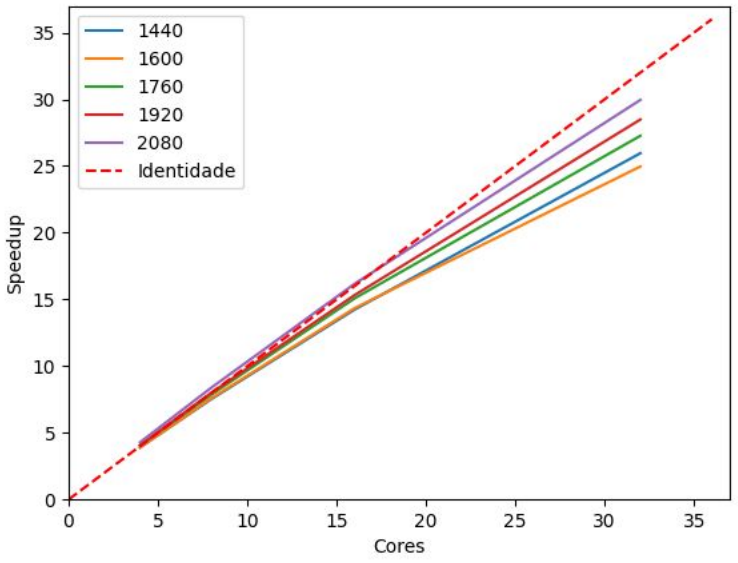

Figura 4: Comparação do speedup em cada um dos tamanhos do problema com o speedup ideal. Interessante que, assim como na versão de multiplicação sequencialmente nas linhas, com o aumento do tamanho do problema e o aumento do número de cores a diferença entre os speedups para cada problema foi aumentando.

O speedup nessa versão se comportou como o esperado, se mantendo abaixo do speedup ideal e curiosamente os valores mantiveram uma consistência bem semelhante no aumento conforme o número de cores, mesmo com tamanho de problema diferentes, com exceção do tamanho 1600, que obteve speedup menor que o de tamanho 1440. Seguem agora os gráficos sobre a eficiência.

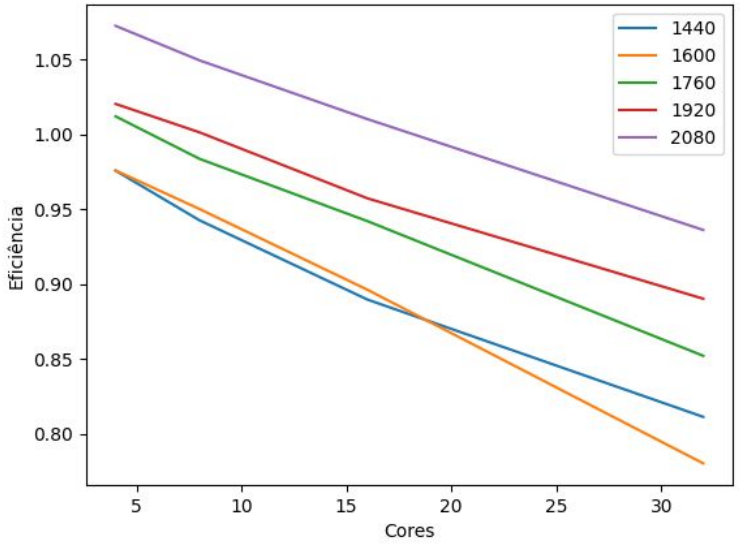

Figura 5: Apresenta a eficiência por cores em cada tamanho de problema utilizado, assim como no gráfico equivalente do algoritmo anterior, o comportamento obtido é o esperado, diminuição da eficiência com o aumento dos cores, tirando a eficiência do problema 1600, que acabou sendo menor que a do 1440, a eficiência se comportou como esperado.

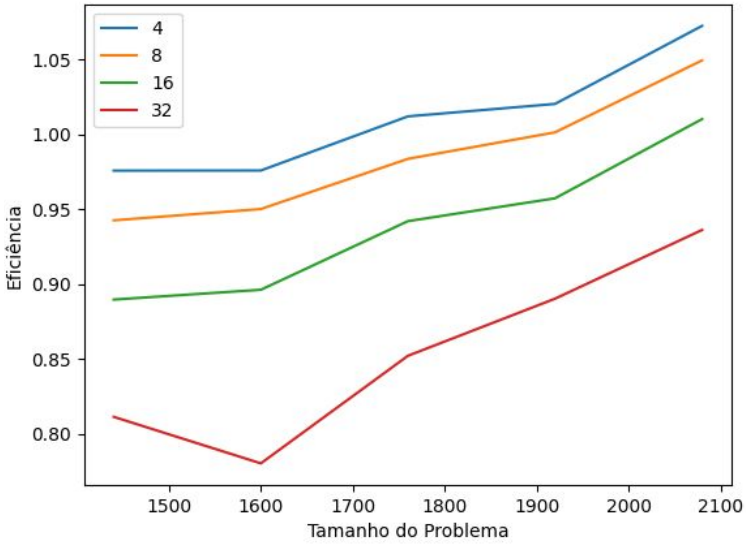

Figura 6: Apresenta a eficiência por tamanho de problema em cada número de cores utilizados, o comportamento geral foi o mesmo do algoritmo anterior, porém o aumento da eficiência ocorreu de maneira diferente, principalmente ao utilizar 32 cores, onde surgiu uma queda na eficiência com o tamanho de problema 1600. Vale destacar que a maior eficiência obtida foi levemente maior que 1.0 nessa versão, enquanto que na anterior chegou a 1.3.

Assim, analisando esses gráficos, essa versão com a transposição pode ser classificada como **Fracamente Escalável**, visto que assim como a versão anterior, com o aumento do tamanho do problema e do número de cores em proporções iguais, a eficiência se mantém crescente.

Considerações Finais

Neste relatório apresentou-se como é definida a multiplicação de duas matrizes, quais as condições para a execução dessa operação, como essa operação é feita e como ela se aplica para matrizes quadradas. Foram explanadas três versões seriais e duas versões paralelas para a multiplicação de matrizes quadradas, sendo que duas das seriais diferem apenas na sequência em que as linhas são percorridas (se é algo aleatório ou não) e a outra versão serial realiza a *transposição* da segunda matriz antes de realizar a multiplicação linha a linha. A primeira versão paralela implementada têm como base a versão serial sem sequência aleatória de percurso das linhas e a segunda versão paralela usa como base a versão serial com transposição da segunda matriz. A versão serial com sequência aleatória de percurso das linhas não teve versão paralela implementada. Foram abordadas apenas as implementações realizadas, focando nos principais pontos das mesmas, com maior detalhe para a versão serial sem aleatoriedade no percurso das linhas e na versão serial com transposição. Em seguida, foi realizada uma breve apresentação da corretude dos algoritmos implementados, exemplificando a execução dos mesmos por meio de uma instância de execução dos algoritmos implementados e depois verificando o resultado realizando os cálculos manualmente, constatando que o resultado é correto, relevando o erro por aproximação dos números de ponto flutuante. Por conseguinte, a análise e comparação do speedup e eficiências dos algoritmos paralelos implementados foi feita, expondo os tempos médios obtidos nos experimentos realizados e discutindo os gráficos de speedup e eficiência gerados por esses tempos. É facilmente perceptível o ganho no tempo gasto na versão com transposição em relação a versão sem transposição, mesmo que as métricas obtidas sobre speedup e escalabilidade sejam menores na versão com transposição. Por fim foi realizada a categorização das versões paralelas em relação a escalabilidade delas, sendo ambas as versões **Fracamente Escaláveis**.

Referências

Introdução a Sistemas Paralelos

Produto de matrizes

Matrix multiplication algorithm

How can I shuffle the contents of an array?