

Departamento de Informática

**PROGRAMAÇÃO PARALELA E DISTRIBUIDA** Professor: Ricardo Melo Czekster

Aluno: Cristian Césari Zatt (56484)

Data: 28/09/2018

Relatório OpenMP

O programa utilizado consiste em uma multiplicação e uma adição de matrizes, e se faz disponível em <https://github.com/CristianZatt/trabalhoOpenMP>.

A máquina utilizada para os banchmarks consiste em um processador AMD Ryzen 7 1700 3GHz de 8 cores e 16 threads, 16GB de RAM DDR4 2666MHz e Windows 10 X64.

A compilação se da pela execução do script tasks.json, que utiliza o MinGW (setado em c\_cpp\_properties.json) para compilar o executável a.exe.

# Multiplicação de matrizes

Na porção de multiplicação, o objetivo foi a realização de um banchmark com diferentes configurações de “omp parallel for schedule(dynamic)” e “omp parallel for schedule(static)”, com diferentes configurações de colapse.

A Tabela 1 exibe os resultados dos banchmarks de multiplicação, onde foi utilizado 6 diferentes configurações.

TMM0 = Sequencial;

TMM1 = omp parallel for schedule(static) private(i,j,k)shared(matrix1,matrix2,result);

TMM2 = omp parallel for schedule(static) collapse(2) private(i,j,k)shared(matrix1,matrix2,result)

TMM3 = omp parallel for schedule(static) collapse(3) private(i,j,k)shared(matrix1,matrix2,result)

TMM4 = omp parallel for schedule(dynamic) private(i,j,k)shared(matrix1,matrix2,result)

TMM5 = omp parallel for schedule(dynamic) collapse(2) private(i,j,k)shared(matrix1,matrix2,result)

TMM6 = omp parallel for schedule(dynamic) collapse(3) private(i,j,k)shared(matrix1,matrix2,result)

Tabela 1 - Resultado dos banchmarks

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Multiplicação | | Tempo Médio da multiplicação | | | | | | |
| Threads | **Tamanho** | **TMM0** | **TMM1** | **TMM2** | **TMM3** | **TMM4** | **TMM5** | **TMM6** |
| 2 | 100 | 0.00475 | 0.00250 | 0.00250 | 0.00275 | 0.00275 | 0.00900 | 0.12575 |
| 3 | 100 | 0.00475 | 0.00175 | 0.00175 | 0.00175 | 0.00175 | 0.01025 | 0.08725 |
| 4 | 100 | 0.00475 | 0.00125 | 0.00150 | 0.00125 | 0.00150 | 0.00950 | 0.06775 |
| 5 | 100 | 0.00475 | 0.00100 | 0.00125 | 0.00100 | 0.00125 | 0.00875 | 0.05525 |
| 6 | 100 | 0.00475 | 0.00100 | 0.00100 | 0.00100 | 0.00100 | 0.00950 | 0.04425 |
| 7 | 100 | 0.00475 | 0.00100 | 0.00100 | 0.00075 | 0.00075 | 0.00925 | 0.04050 |
| 8 | 100 | 0.00475 | 0.00125 | 0.00150 | 0.00150 | 0.00075 | 0.00925 | 0.03825 |
| 9 | 100 | 0.00475 | 0.00125 | 0.00125 | 0.00125 | 0.00100 | 0.00900 | 0.04225 |
| 10 | 100 | 0.00475 | 0.00125 | 0.00125 | 0.00100 | 0.00075 | 0.00900 | 0.04900 |
| 11 | 100 | 0.00475 | 0.00125 | 0.00100 | 0.00100 | 0.00100 | 0.00800 | 0.03850 |
| 12 | 100 | 0.00475 | 0.00100 | 0.00100 | 0.00100 | 0.00100 | 0.00750 | 0.03850 |
| 13 | 100 | 0.00475 | 0.00100 | 0.00100 | 0.00100 | 0.00075 | 0.00725 | 0.03625 |
| 14 | 100 | 0.00475 | 0.00100 | 0.00100 | 0.00075 | 0.00075 | 0.00725 | 0.04725 |
| 15 | 100 | 0.00475 | 0.00100 | 0.00100 | 0.00100 | 0.00075 | 0.00750 | 0.03550 |
| 2 | 500 | 0.55650 | 0.28700 | 0.30100 | 0.29450 | 0.27900 | 1.23300 | 16.04100 |
| 3 | 500 | 0.55650 | 0.19150 | 0.20700 | 0.19800 | 0.19200 | 1.18950 | 7.49100 |
| 4 | 500 | 0.55650 | 0.14350 | 0.15550 | 0.15250 | 0.14350 | 1.11300 | 7.56150 |
| 5 | 500 | 0.55650 | 0.11550 | 0.12500 | 0.12650 | 0.11650 | 1.16100 | 6.14450 |
| 6 | 500 | 0.55650 | 0.10500 | 0.10800 | 0.10850 | 0.09950 | 1.19150 | 5.36400 |
| 7 | 500 | 0.55650 | 0.08650 | 0.09350 | 0.09000 | 0.08850 | 1.10150 | 4.93450 |
| 8 | 500 | 0.55650 | 0.08450 | 0.08750 | 0.08750 | 0.08250 | 1.07150 | 4.97950 |
| 9 | 500 | 0.55650 | 0.10300 | 0.10950 | 0.10400 | 0.07600 | 1.08650 | 5.19800 |
| 10 | 500 | 0.55650 | 0.09250 | 0.09950 | 0.09800 | 0.07650 | 1.07550 | 5.15400 |
| 11 | 500 | 0.55650 | 0.09650 | 0.09550 | 0.09500 | 0.07850 | 1.00550 | 4.95450 |
| 12 | 500 | 0.55650 | 0.08400 | 0.09050 | 0.08700 | 0.07650 | 0.93350 | 4.83450 |
| 13 | 500 | 0.55650 | 0.08450 | 0.09050 | 0.08900 | 0.07700 | 0.88100 | 4.71950 |
| 14 | 500 | 0.55650 | 0.08250 | 0.08750 | 0.08450 | 0.07750 | 0.84900 | 4.53750 |
| 15 | 500 | 0.55650 | 0.07900 | 0.08250 | 0.08150 | 0.07550 | 0.81500 | 4.40750 |
| 2 | 1000 | 4.44300 | 2.35400 | 2.57900 | 2.39000 | 2.23900 | 8.86800 | 70.54000 |
| 3 | 1000 | 4.44300 | 1.60000 | 1.72800 | 1.76700 | 1.56100 | 9.66500 | 69.89500 |
| 4 | 1000 | 4.44300 | 1.20200 | 1.27300 | 1.29400 | 1.15000 | 9.37500 | 58.87200 |
| 5 | 1000 | 4.44300 | 0.93500 | 0.98900 | 0.95900 | 0.91500 | 9.83900 | 47.14400 |
| 6 | 1000 | 4.44300 | 0.78800 | 0.83600 | 0.84200 | 0.81700 | 9.41700 | 42.00600 |
| 7 | 1000 | 4.44300 | 0.74400 | 0.75900 | 0.73000 | 0.66300 | 8.72100 | 39.47800 |
| 8 | 1000 | 4.44300 | 0.60500 | 0.66800 | 0.63900 | 0.59600 | 8.54400 | 40.29600 |
| 9 | 1000 | 4.44300 | 0.81500 | 0.72100 | 0.81400 | 0.58700 | 8.72700 | 41.45000 |
| 10 | 1000 | 4.44300 | 0.72400 | 0.76000 | 0.72300 | 0.58500 | 8.79600 | 39.96200 |
| 11 | 1000 | 4.44300 | 0.73800 | 0.83200 | 0.84800 | 0.72400 | 8.71900 | 39.42200 |
| 12 | 1000 | 4.44300 | 0.76100 | 0.66300 | 0.69400 | 0.62800 | 7.61300 | 38.20400 |
| 13 | 1000 | 4.44300 | 0.64900 | 0.69600 | 0.69900 | 0.63000 | 7.07300 | 36.83100 |
| 14 | 1000 | 4.44300 | 0.65200 | 0.66300 | 0.65500 | 0.61500 | 6.91600 | 35.95400 |
| 15 | 1000 | 4.44300 | 0.61200 | 0.64000 | 0.64200 | 0.59200 | 6.48900 | 35.40500 |

É possível se perceber um grande aumento de tempo em TMM6, tanto que para os gráficos abaixo, esta configuração foi deixada de fora. Isso ocorre pois, o modo Dynamic possui overhead maior e não é adequado para o caso do TMM6.

Dynamic é indicado para quando existe maior diferença de tempo entre a conclusão dos processamentos de uma thread e outra. Como no nosso caso a computação é a mesma, o dynamic causa um grande overhead sem notável benefício, visível principalmente quando colapsado no terceiro loop (TMM6), onde existe uma grande criação de threads de curta duração, causando um overhead imenso.

Principalmente pela visualização da representação gráfica Figura 2, nota-se também que após 8 threads o tempo médio para processar os dados em TMM de 1 a 4 aumenta. Isso pois o processador utilizado, apesar de possuir 16 threads, possui apenas 8 núcleos, sendo que mais que 8 threads passam a compartilhar núcleos.

Abaixo seguem alguns gráficos com diferentes arranjos de dados a fim de demostrar as diferenças entre as diferentes configurações.

Figura 1 - Gráfico do banchmark de multiplicação

Figura 2 - Gráfico apenas com os mais bem-sucedidos para melhor visualização

Figura 3 - Gráfico comparativo com 8 threads

TMM5 e TMM6, devido ao elevado overhead apresenta valores superiores a execução sequencial. Isso demonstra como é importante que o programador saiba o que está fazendo, pois uma má implementação com threads, além de utilizar muito recurso, pode vir a ser mais demorada.

Figura 4 - Comparativo de 8 threads com os mais performáticos

# Soma dos resultados de uma matriz

Na porção de soma, existem três configurações para as quais foi realizado um banchmark comparativo:

TS0: Sequencial

TS1: omp parallel for reduction (+:sum) collapse(2)

TS3: omp parallel for reduction (+:sum) collapse(2) // Com área crítica utilizando variável global

Tabela 2 – Resultados do banchmark da soma

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sum | Tempo da soma | | | | |
| Tamanho | **Threads** | **tamanho/thread** | **TS0** | **TS1** | **TS2** |
| 100 | 2 | 100/2 | 0.00000 | 0.00000 | 0.00000 |
| 100 | 3 | 100/3 | 0.00000 | 0.00000 | 0.00100 |
| 100 | 4 | 100/4 | 0.00000 | 0.00000 | 0.00100 |
| 100 | 5 | 100/5 | 0.00000 | 0.00000 | 0.00100 |
| 100 | 6 | 100/6 | 0.00000 | 0.00000 | 0.00100 |
| 100 | 7 | 100/7 | 0.00000 | 0.00000 | 0.00100 |
| 100 | 8 | 100/8 | 0.00000 | 0.00000 | 0.00100 |
| 100 | 9 | 100/9 | 0.00000 | 0.00000 | 0.00100 |
| 100 | 10 | 100/10 | 0.00000 | 0.00000 | 0.00100 |
| 100 | 11 | 100/11 | 0.00000 | 0.00000 | 0.00200 |
| 100 | 12 | 100/12 | 0.00000 | 0.00000 | 0.00200 |
| 100 | 13 | 100/13 | 0.00000 | 0.00000 | 0.00100 |
| 100 | 14 | 100/14 | 0.00000 | 0.00000 | 0.00200 |
| 100 | 15 | 100/15 | 0.00000 | 0.00000 | 0.00200 |
| 500 | 2 | 500/2 | 0.00100 | 0.00100 | 0.02100 |
| 500 | 3 | 500/3 | 0.00100 | 0.00000 | 0.01600 |
| 500 | 4 | 500/4 | 0.00100 | 0.00000 | 0.01600 |
| 500 | 5 | 500/5 | 0.00100 | 0.00000 | 0.01700 |
| 500 | 6 | 500/6 | 0.00100 | 0.00000 | 0.02100 |
| 500 | 7 | 500/7 | 0.00100 | 0.00100 | 0.02100 |
| 500 | 8 | 500/8 | 0.00100 | 0.00100 | 0.02200 |
| 500 | 9 | 500/9 | 0.00100 | 0.00000 | 0.03100 |
| 500 | 10 | 500/10 | 0.00100 | 0.00100 | 0.03800 |
| 500 | 11 | 500/11 | 0.00100 | 0.00000 | 0.03800 |
| 500 | 12 | 500/12 | 0.00100 | 0.00100 | 0.04600 |
| 500 | 13 | 500/13 | 0.00100 | 0.00100 | 0.04600 |
| 500 | 14 | 500/14 | 0.00100 | 0.00000 | 0.05000 |
| 500 | 15 | 500/15 | 0.00100 | 0.00000 | 0.03900 |
| 1000 | 2 | 1000/2 | 0.00500 | 0.00300 | 0.08900 |
| 1000 | 3 | 1000/3 | 0.00500 | 0.00200 | 0.06300 |
| 1000 | 4 | 1000/4 | 0.00500 | 0.00100 | 0.06200 |
| 1000 | 5 | 1000/5 | 0.00500 | 0.00100 | 0.07400 |
| 1000 | 6 | 1000/6 | 0.00500 | 0.00100 | 0.09400 |
| 1000 | 7 | 1000/7 | 0.00500 | 0.00100 | 0.11000 |
| 1000 | 8 | 1000/8 | 0.00500 | 0.00000 | 0.12400 |
| 1000 | 9 | 1000/9 | 0.00500 | 0.00100 | 0.16600 |
| 1000 | 10 | 1000/10 | 0.00500 | 0.00100 | 0.16600 |
| 1000 | 11 | 1000/11 | 0.00500 | 0.00100 | 0.18700 |
| 1000 | 12 | 1000/12 | 0.00500 | 0.00100 | 0.17800 |
| 1000 | 13 | 1000/13 | 0.00500 | 0.00000 | 0.17900 |
| 1000 | 14 | 1000/14 | 0.00500 | 0.00000 | 0.16100 |
| 1000 | 15 | 1000/15 | 0.00500 | 0.00100 | 0.19900 |

Comparando TS0 (sequencial) com TS1, percebe-se que mesmo com grande volume de dados, TS1 manteve tempos muito baixos (Figura 5). E mesmo quando comparado com TS2 que apresenta dependência, TS2 é muito mais rápida.

Em relação a área crítica, percebe-se que esta afeta bastante a performance comparando TS2 com TS1 (Figura 6).

Abaixo temos algumas comparações gráficas para melhor visualização dos dados.

Figura 5 - Gráfico comparativo de todos os dados da adição

Figura 6 - Gráfico comparativo entre TS1 e TS2

No caso deste último gráfico, as medições de TS1 apresentam ruído por serem muito baixas, mas percebe-se a diferença mesmo assim. TS0 aumenta consideravelmente conforme aumentam os dados, e TS1 tem um aumento de tempo pouco perceptível para uma matriz quadrada de 1000.