MAC 0219/5742 – Introdução à Computação Concorrente, Paralela e Distribuída

Prof. DR. Alfredo Goldman,
Guilherme Souza S.

1

1. Relatório Geral

O algoritmo *matrix_dgemm_1* teve uma melhora enorme comparado a *matrix_degemm_0*. Sendo possível ver que haveria um boa melhora em sua implementação, durante sua implementação houve o cuidado de fazer um melhor aproveitamento da cache, ou seja fez-se uso da localização espacial, ao buscar um dado acabamos por trazer alguns adjacentes a ele, fazendo um melhor aproveitamento disso mudou-se a ordem da implementação trivial, realizando cálculos parciais para utilizar os dados já disponíveis, assim a quebra na matriz B (nesse caso específico a matriz se chama B, mas considere qualquer matriz que tiver o acesso por coluna), pode-se observar que todo acesso que foi feito na matriz com um *for* sobre o outro (de forma a garantir melhor uso das posições da matrz) assim garantindo um melhor acesso ao dados a matriz .

Comparando *matrix_dgemm_1* com *matrix_dgemm_2*, teve uma melhora circunstancial com matrizes pequenas porem ao aumentar bastante a matriz, pode-se observar uma boa melhora. Dividindo uma matriz grande em blocos de tamanho x, faz com que o uso da cache seja melhor aproveitamento também, porém a realizar a blocagem, começa-se a realizar calculo de matrizes menores, podendo fazer o que foi feito no algoritmo 1, logo usando melhor a cache também, tendo que ter o cuidado para andar de forma correta na matriz, alem de andar de forma à aproveitar os dados trazidos com a localização espacial.

2. Relato Pessoal

No inicio do trabalho parecia algo mais trivial, pois quando se fala em aproveitamento de cache sabemos que temos que fazer um aproveitamento melhor da localidade espacial, porém a realizar isso na pratica tive um pouco de dificuldade em ver isso em código, apos algumas execuções no papel a consulta na monitoria tornou-se mais fácil de ver o que estava ocorrendo no código, em blocagem o conceito e de fácil entendimento porém o controle de andar corretamente nos blocos certo para que ocorra a multiplicação da matriz corretamente era um pouco mais difícil do que o conceito em si.

Em um primeiro momento fiz uso do primeiro código escritor por mim, em um conjunto com o conceito estendido, teve-se um código funcional porém não tão eficiente quanto o primeiro, a quantidade de *for* era uma questão importante, primeiramente implementado com 6 *for*, após uma análise do tempo e a sua não melhora, decidi obter um realização da matriz com 5 *for*, mas de novo, encontrar a ordem dos *for* sem que interfira nas mudanças do calculo e acesso da matriz. Apos algumas tentativas e mais umas rabiscadas no papel e uma conversa com os monitores, foi possível mais um ganho no tempo

do algoritmo, depois de ver algumas posições que eram-se usada mais de uma vez sem se ter alteração realizei o uso de variável auxiliar (o que me evitava o acesso a matriz) e o acesso utilizando o *define* ouve-se mais ganho no tempo podendo-se demonstrar um melhora superior comparado com *matrix_dgemm_1*.

3. Hardware

O computador no qual os testes foram rodados conta com as configurações apresentadas abaixo, tais informações foram obtidas com o comando *dmidecode type number*.

```
1
2  Handle 0x0006, DMI type 7, 19 bytes
3  Cache Information
4    Socket Designation: L1-Cache
5    Configuration: Enabled, Not Socketed, Level 1
6    Operational Mode: Write Through
7    Location: Internal
8    Installed Size: 32 kB
9    Maximum Size: 32 kB
10   Supported SRAM Types:
11     Unknown
12   Installed SRAM Type: Unknown
13   Speed: Unknown
14   Error Correction Type: Parity
15   System Type: Instruction
16   Associativity: 8-way Set-associative
17
18 Handle 0x0007, DMI type 7, 19 bytes
19 Cache Information
20   Socket Designation: L2-Cache
21   Configuration: Enabled, Not Socketed, Level 2
22   Operational Mode: Write Through
23   Location: Internal
24   Installed Size: 256 kB
25   Maximum Size: 256 kB
26   Supported SRAM Types:
27     Unknown
28   Installed SRAM Type: Unknown
29   Speed: Unknown
30   Error Correction Type: Multi-bit ECC
31   System Type: Unified
32   Associativity: 8-way Set-associative
33
34 Handle 0x0008, DMI type 7, 19 bytes
35 Cache Information
36   Socket Designation: L3-Cache
37   Configuration: Enabled, Not Socketed, Level 3
38   Operational Mode: Write Back
39   Location: Internal
40   Installed Size: 3072 kB
41   Maximum Size: 3072 kB
42   Supported SRAM Types:
43     Unknown
44   Installed SRAM Type: Unknown
45   Speed: Unknown
```

```
    Error Correction Type: Multi-bit ECC
    System Type: Unified
    Associativity: 12-way Set-associative
```

Listing 1. Informacoes Cache

```

Handle 0x0037, DMI type 17, 34 bytes
Memory Device
    Array Handle: 0x0035
    Error Information Handle: Not Provided
    Total Width: 64 bits
    Data Width: 64 bits
    Size: 4096 MB
    Form Factor: SODIMM
    Set: None
    Locator: Channel1B-DIMM0
    Bank Locator: BANK 2
    Type: DDR3
    Type Detail: Synchronous
    Speed: 1600 MHz
    Manufacturer: Micron
    Serial Number: 44297152
    Asset Tag: 9876543210
    Part Number: 8KTF51264HZ-1G6E1
    Rank: Unknown
    Configured Clock Speed: 1600 MHz
```

Listing 2. Informacoes Memoria

```
Handle 0x0004, DMI type 4, 42 bytes
Processor Information
    Socket Designation: CPU Socket - U3E1
    Type: Central Processor
    Family: Core i5
    Manufacturer: Intel(R) Corporation
    ID: A9 06 03 00 FF FB EB BF
    Signature: Type 0, Family 6, Model 58, Stepping 9
    Flags: #omitido
    Version: Intel(R) Core(TM) i5-3230M CPU @ 2.60GHz
    Voltage: 0.9 V
    External Clock: 100 MHz
    Max Speed: 2600 MHz
    Current Speed: 2600 MHz
    Status: Populated, Enabled
    Upgrade: Socket rPGA988B
    L1 Cache Handle: 0x0006
    L2 Cache Handle: 0x0007
    L3 Cache Handle: 0x0008
    Serial Number: To Be Filled By O.E.M.
    Asset Tag: To Be Filled By O.E.M.
    Part Number: To Be Filled By O.E.M.
    Core Count: 2
    Core Enabled: 2
    Thread Count: 4
    Characteristics:
```

```
27      64-bit capable
```

Listing 3. Informacoes Processador

4. Resultados

Em um primeiro momento executou-se 30 vezes cada algoritmo com uma matriz de N=2048, após tal execução realizou-se a média para uma melhor qualidade da amostra. Nessa etapa a grande diferença pode se observar do *matrix_degemm_0* para os demais, quanto ao *matrix_degemm_1* comparado ao *matrix_degemm_2* o ganho foi relativamente pouco, mas tal motivo se da pelo fato da matriz ser pequena a ponto do método de blocagem não se mostrar tão necessário, mas ainda sim demonstrando ganho, tais resultados podem ser vistos na Tabela1 e Gráfico1 ao fim do relatório.

Tendo tal ponto em vista, pudemos ter noção que com o aumento de N ou seja o tamanho da matriz, a diferença se demonstre maior, pois ai começa a se tornar mais complicado o controle sobre as posições da cache, onde a blocagem iria se demonstrar superior ainda sim com um ganho pequeno como visto na Tabela2 e Gráfico2.

Dando continuidade aos testes de ganho, por motivos de tempo habil a execução dos testes, optouse por diminuir a quantidade de execuções, passando de 30 para 10, pelo tempo de duração de cada teste, principalmente o *matrix_degemm_0* que se demonstra menos eficiênte, logo usou-se em compração somente mais uma vez ja que a distancia de eficiência se tornou muito grande.

Em um último teste em uma matriz 8112, demonstrando somente o *matrix_degemm_1* e *matrix_degemm_2* Na Tabela 3 e Gráfico, onde pode ser visto uma melhora na eficiência, mas creio que para tamanho ainda maiores essa distância fique mais perceptível, porém tem a necessidade de se ter máquinas com poder computacional para tal execução, além de tempo disponível, para melhor demonstação usou-se a formula padrão do intervalo de confiança junto aos limites superiores e inferiores nos dando uma melhor visibilidade dos resultados, até por não seguir muito uma normal.

Tabela1: Dados quantitativos dos algoritmos para tamanho de matriz 2048

Algotirmo e Tamanho N de 2048	Média	Desvio Padrao	Limite Superior	Limite Inferior	Intervalo de confiança	Primeiro Quartil	Terceiro Quartil
Algoritmo 0	92	36,12925406	128	56	12,92844997	90,054842	97,5478685
Algoritmo 1	7	0,04043287251	7	7	0,01446845175	7,08434	7,10490375
Algoritmo 2	6	0,01297317928	6	6	0,004642307281	5,653624	5,6614355

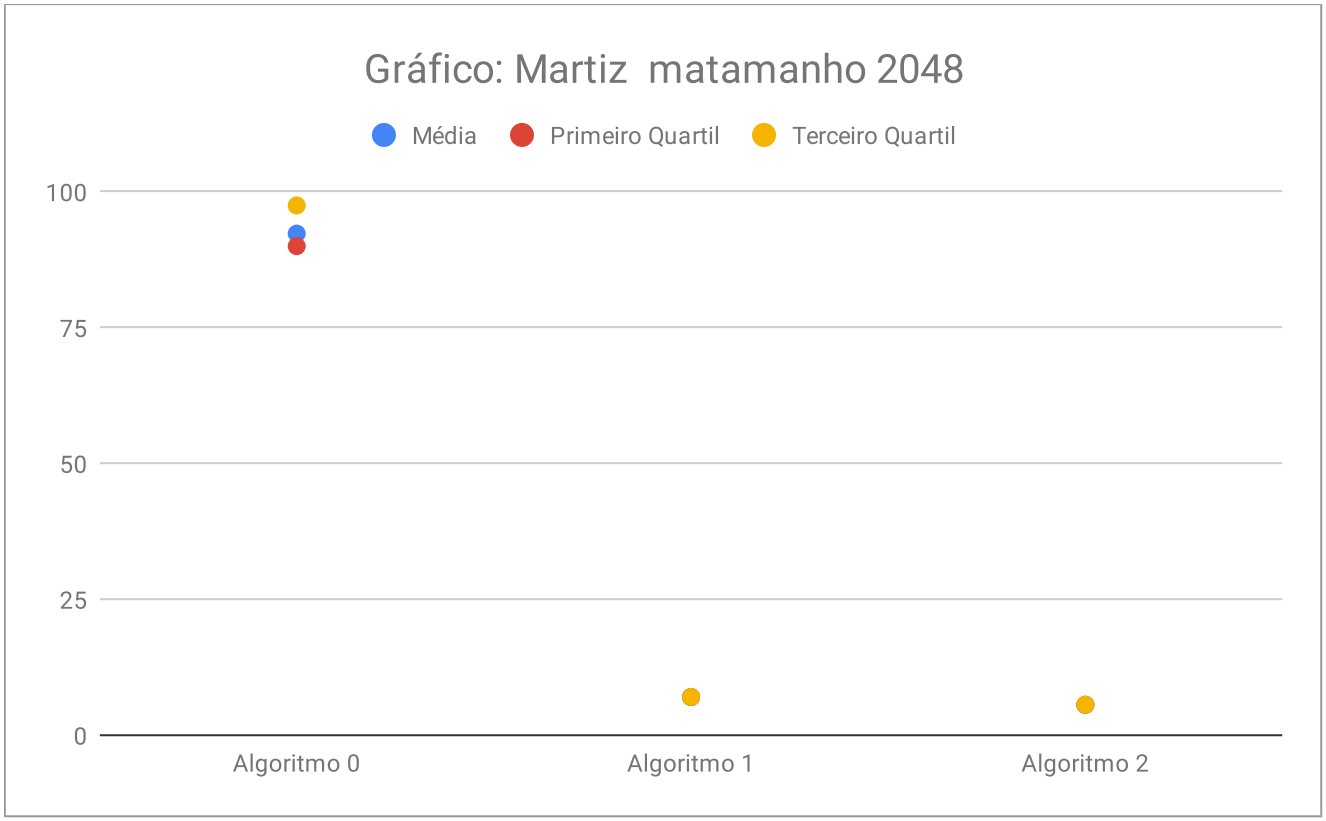

Tabela2: Dados quantitativos dos algoritmos para tamanho de matriz 4096

Algotirmo e Tamanho N de 4096	Média	Desvio Padrao	Limite Superior	Limite Inferior	Intervalo de confiança	Primeiro Quartil	Terceiro Quartil
Algoritmo 0	956	76,22678649	1032	880	47,24498363	950,98283	1092,085702
Algoritmo 1	60,1152295	0,1456814191	60	59,96954808	0,09029261994	60,01191225	60,27756225
Algoritmo 2	48,4850905	0,09127326673	49	48,39381723	0,05657071735	48,425224	48,552474

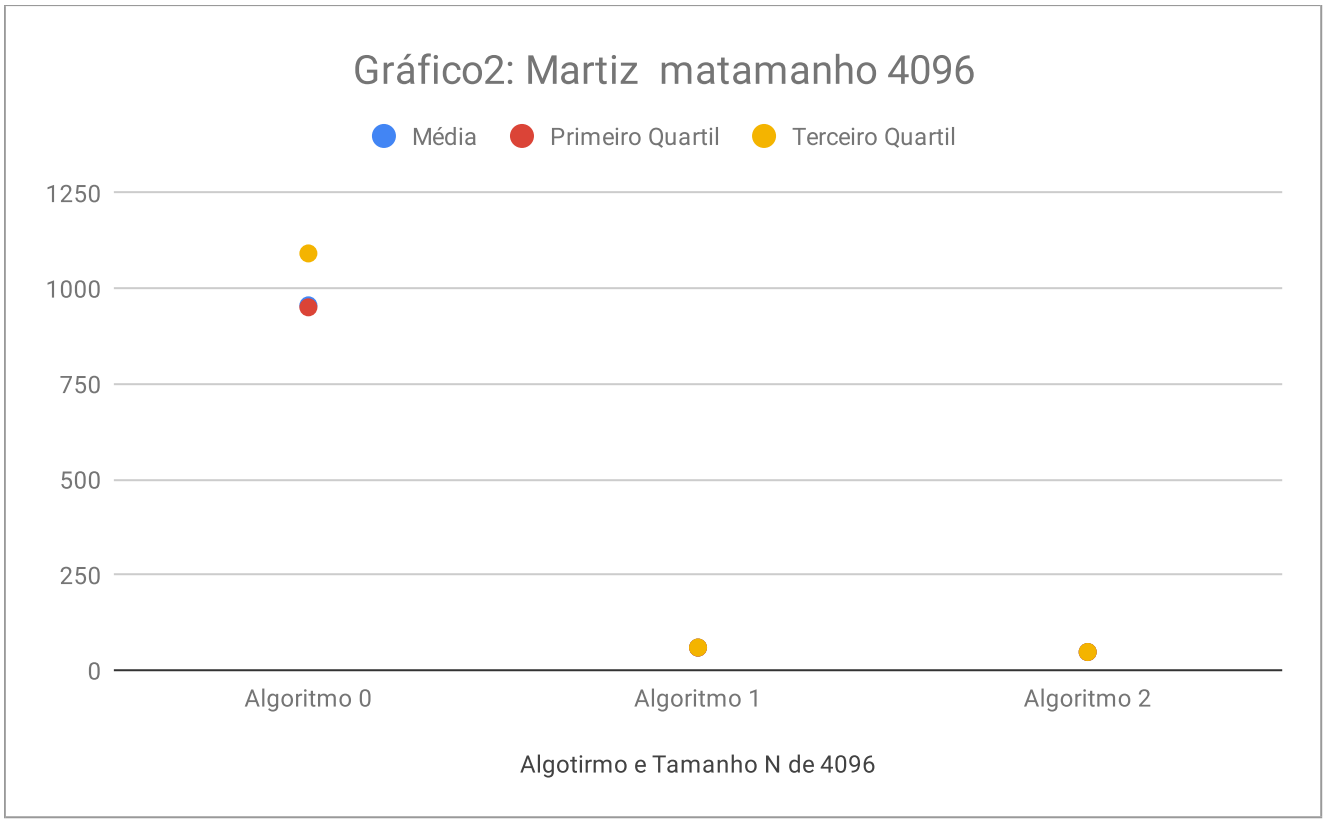

Tabela3: Dados quantitativos dos algoritmos para tamanho de matriz 8192

Algotirmo e Tamanho N de 8192	Média	Desvio Padrao	Limite Superior	Limite Inferior	Intervalo de confiança	Primeiro Quartil	Terceiro Quartil
Algoritmo 0	-	-	-	-	-	-	-
Algoritmo 1	492,441444	7,04630819	499	485,3951358	4,367266816	489,577384	498,31959
Algoritmo 2	487,015825	11,26203859	498	475,7537864	6,980155579	485,43304	494,5236953