OAC PRÁTICA 3

Memória CACHE

Vinicius de Oliveira Silva

Objetivo

- Demonstrar os efeitos da memória cache no acesso aos dados de um programa, através de um algoritmo que multiplica duas matrizes, A e B.

O Programa

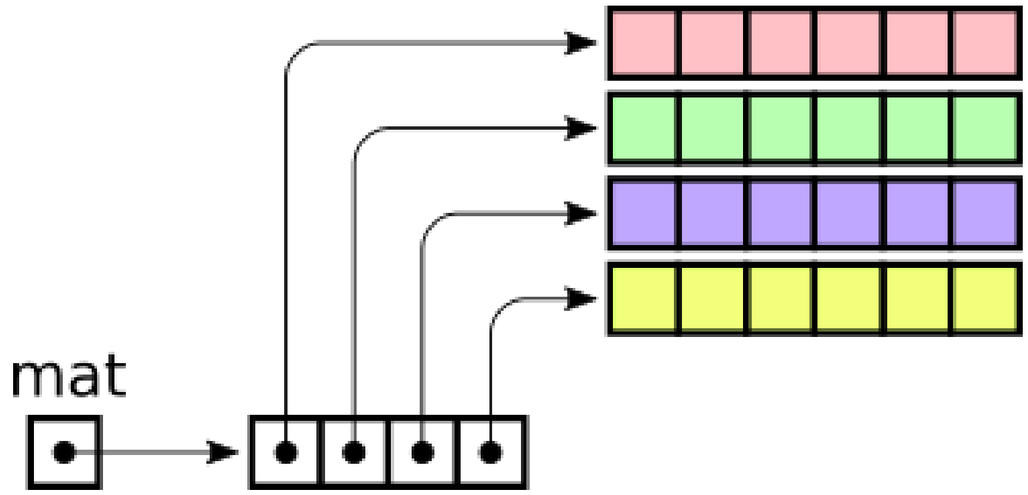
- Implementado em linguagem C;
- Possui 4 modos de multiplicação:

  I - ***Modo original (o):*** não faz alterações nas matrizes antes de multiplicar;
  II - ***Modo transposta (t)***: transpõe a matriz B antes da multiplicação;
  III - ***Modo "vetor" original (vo):*** aloca a matriz B de maneira contígua na memória;
  IV - Modo "vetor" transposta (vt): aloca a matriz transposta de B de maneira contígua.

- A saída do programa é o tempo de execução da multiplicação.

Algoritmos de Alocação das Matrizes

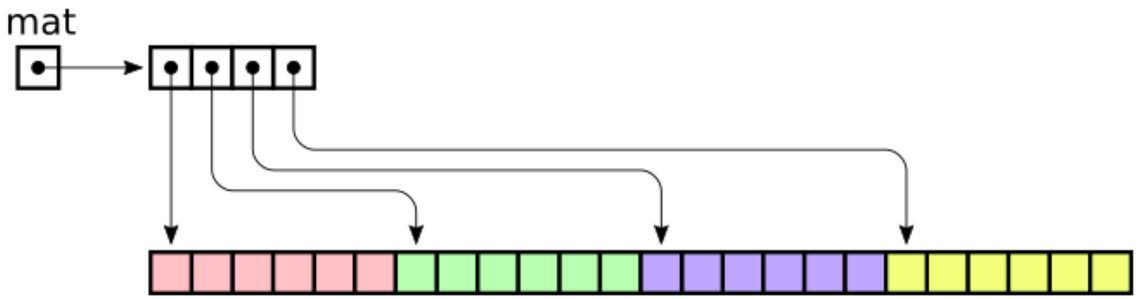
- Vetor de ponteiros de linhas separadas:




```c
double** newMatriz(int l, int c)
{
    double **M;
       M = malloc (l * sizeof (double*)) ;
    int i;
    for (i=0; i < l; i++)
        M[i] = malloc (c * sizeof (double)) ;
    return M;
}
```

Algoritmos de Alocação das Matrizes

- Vetor de ponteiros de linhas contíguas:




```c
double** newVetor(int l, int c)
{
    double **M;
    int i;
    M = malloc (l * sizeof (double*)) ;
    M[0] = malloc(l * c * sizeof(double));
    for (i=1; i < l; i++)
        M[i] = M[0] + c * i;
    return M;
}
```

Algoritmo de Transposição

- Aloca a matriz transposta de acordo com o modo.

```c
double** matrizT(double **M,  int l, int c, char *mode)
{
    int i, j;
    double **Mt;
    if (!strcmp(mode, "vt"))
        Mt = newVetor(c,l);
    else
        Mt = newMatriz(c, l);
    for(i = 0; i < c; i++)
        for(j = 0; j < l; j++)
            Mt[i][j] = M[j][i];
    return Mt;
}
```

Testes e Resultados

Testes

Configurações da máquina:

- Intel Core i7-7500U @ 2.7GHz;
- 2 núcleos 4 threads;
- Intel Smart Cache: L1 128 kB, L2 512 kB, L3 4 MB;
- 8GB DDR4;
- 240GB SSD 500MB/s;

Testes

- Foram utilizadas duas compilações do código para os testes, com otimização do compilador e sem otimização do compilador:
    I - gcc -O3 multmat.c -o multmat.x
    II - gcc multmat.c -o multmat.x
- Para cada um dos executáveis gerados pelas compilações, foram seguidos os seguintes passos:
    1 - O programa foi executado com matrizes quadradas;
    2 - O tamanho das matrizes foi variado entre 200 e 2000, com incremento de 200;
    3 - Para cada tamanho de matriz foram realizadas 10 execuções em cada modo de multiplicação.
- Um shell script foi utilizado para executar os passos acima.

Script de Execução

```
#!/bin/bash
#gcc -O3 multmat.c -o multmat.x
gcc multmat.c -o multmat.x
DIRETORIO="Notebook"
for ((j = 200; j <= 2000; j+=200))
    do
    for ((i=0; i < 10; i++))
        do
        ./multmat.x $j $j $j $j o out >> $DIRETORIO/o$j.csv
        ./multmat.x $j $j $j $j t out >> $DIRETORIO/t$j.csv
        ./multmat.x $j $j $j $j vo out >> $DIRETORIO/vo$j.csv
        ./multmat.x $j $j $j $j vt out >> $DIRETORIO/vt$j.csv
        done
    done
```

Resultados

- Para calcular o tempo médio de execução de todos os tamanhos de matriz, o código "media.c" e o shell script a seguir foram utilizados:

```
#!/bin/bash
gcc media.c -o media.x
DIRETORIO="Notebook"

echo "Tam Matriz,Mode o,Mode t,Mode vo, Mode vt" >> $DIRETORIO/medias.csv
for ((i = 200; i <= 2000; i+=200))
    do
    echo -n "$i," >> $DIRETORIO/medias.csv
    ./media.x $DIRETORIO/o$i.csv >> $DIRETORIO/medias.csv
    ./media.x $DIRETORIO/t$i.csv >> $DIRETORIO/medias.csv
    ./media.x $DIRETORIO/vo$i.csv >> $DIRETORIO/medias.csv
    ./media.x $DIRETORIO/vt$i.csv >> $DIRETORIO/medias.csv
    echo >> $DIRETORIO/medias.csv
    done
```

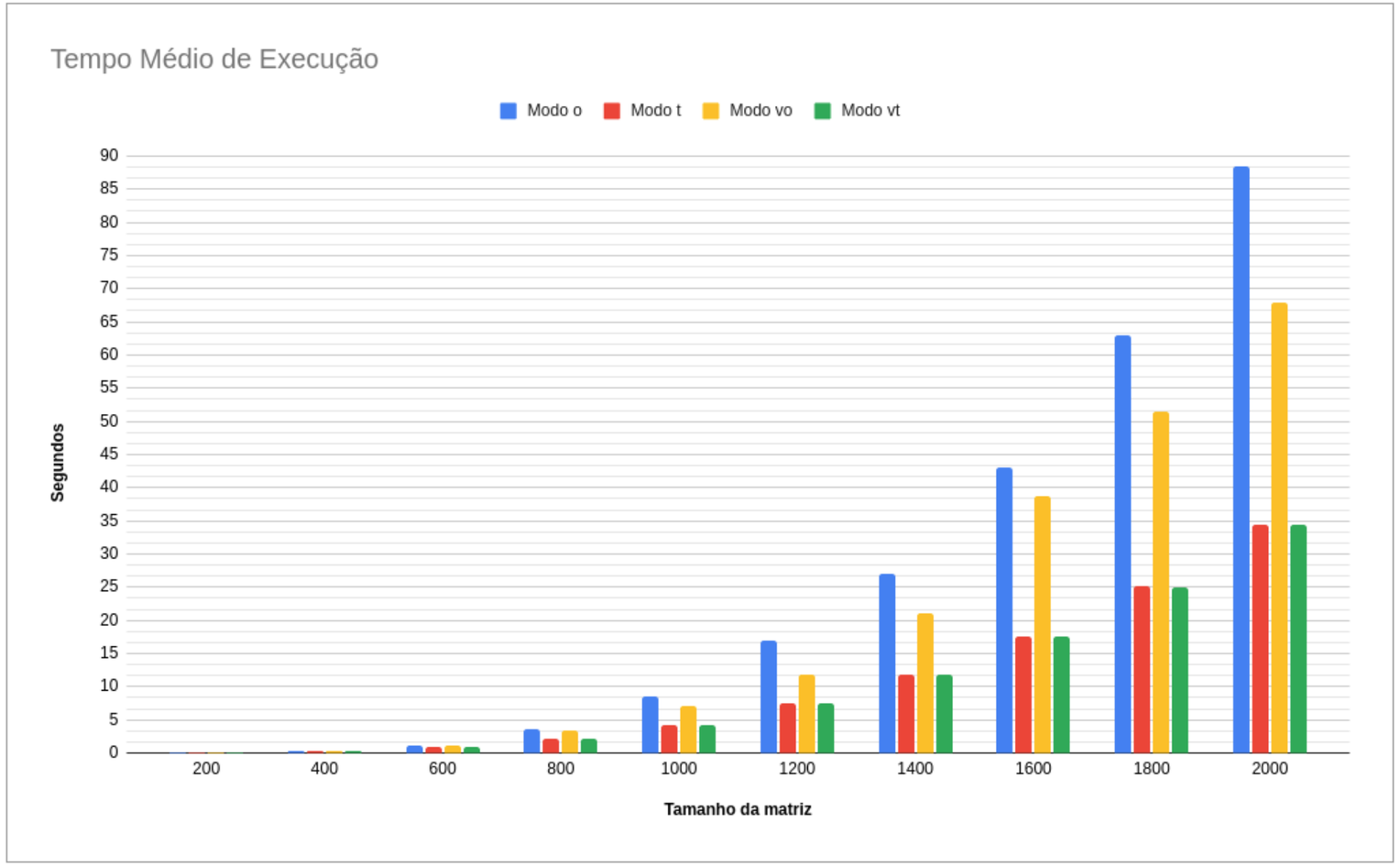
Resultados

Compilador sem otimização

	Tempo médio de execução (seg)			
Tamanho da matriz	**Modo o**	**Modo t**	**Modo vo**	**Modo vt**
200	0,03637	0,03386	0,03512	0,03397
400	0,31679	0,27061	0,31738	0,27121
600	1,22797	0,91033	1,14607	0,90799
800	3,57023	2,18586	3,36757	2,17287
1000	8,51347	4,29998	7,05905	4,27194
1200	16,88843	7,42735	11,87615	7,41014
1400	27,10006	11,79696	21,14709	11,73450
1600	42,95819	17,63377	38,73014	17,52437
1800	63,01296	25,10047	51,37159	24,95320
2000	88,36005	34,43184	67,92963	34,39872

Resultados

Compilador sem otimização



Resultados

Compilador sem otimização

	Speedup's		
Tamanho da matriz	**Speedup t/o**	**Speedup vo / o**	**Speedup vt / t**
200	93,10%	96,56%	100,32%
400	85,42%	100,19%	100,22%
600	74,13%	93,33%	99,74%
800	61,22%	94,32%	99,41%
1000	50,51%	82,92%	99,35%
1200	43,98%	70,32%	99,77%
1400	43,53%	78,03%	99,47%
1600	41,05%	90,16%	99,38%
1800	39,83%	81,53%	99,41%
2000	38,97%	76,88%	99,90%

Resultados

Compilador sem otimização

Tamanho da matriz	Tempo médio de transposição (seg)
200	0,00013
400	0,00058
600	0,00195
800	0,00368
1000	0,00646
1200	0,01081
1400	0,01421
1600	0,01929
1800	0,02571
2000	0,03322

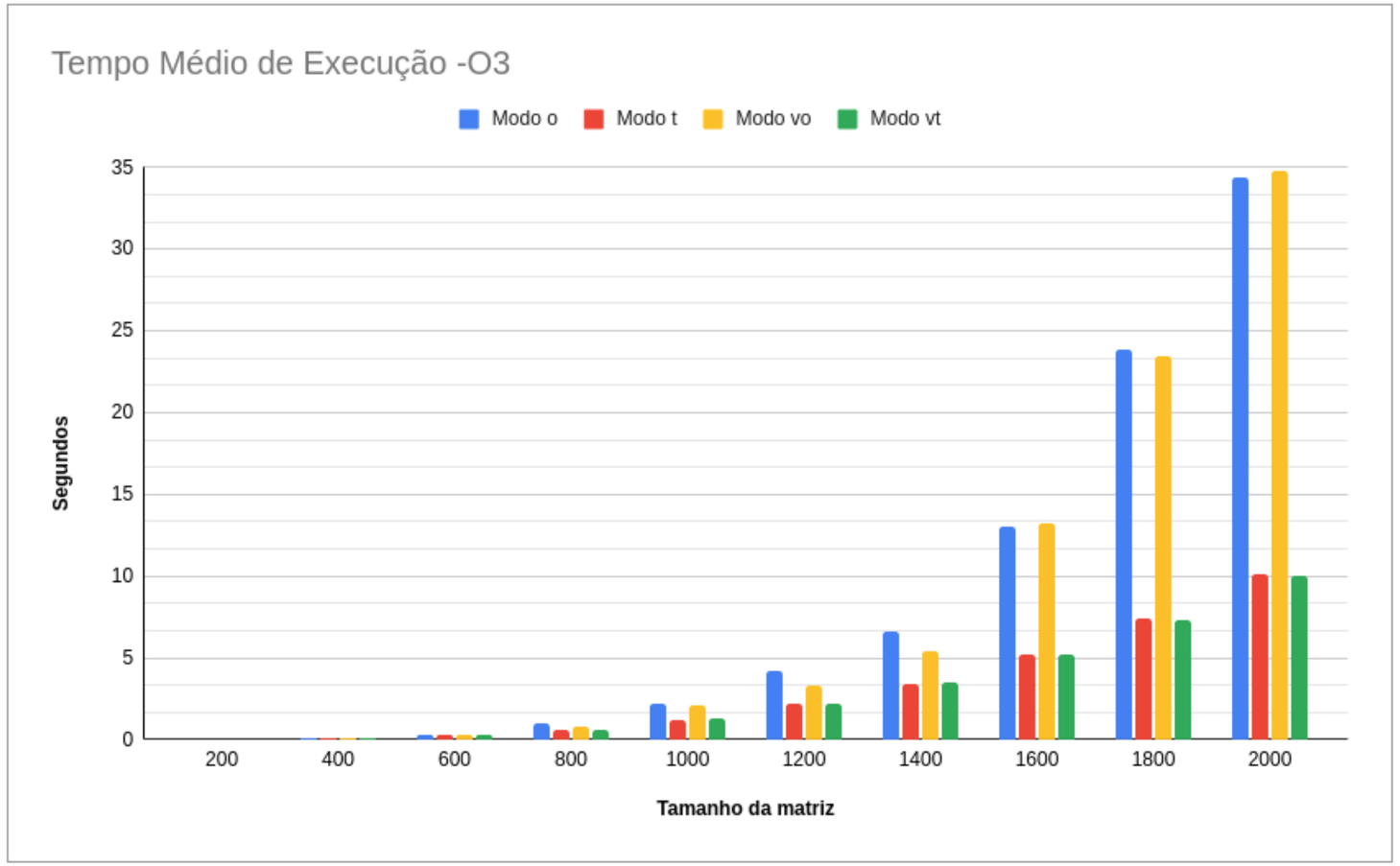
Resultados

Compilador com otimização

	Tempo médio de execução (seg)			
Tamanho da matriz	**Modo o**	**Modo t**	**Modo vo**	**Modo vt**
200	0,00914	0,00870	0,00904	0,00883
400	0,07479	0,07141	0,07813	0,07306
600	0,29920	0,24954	0,28396	0,25024
800	0,96219	0,62180	0,81871	0,63211
1000	2,21586	1,24264	2,04718	1,26251
1200	4,19774	2,15882	3,32147	2,18459
1400	6,60480	3,44205	5,42833	3,48093
1600	12,98341	5,15988	13,25395	5,22891
1800	23,88930	7,36359	23,43156	7,32813
2000	34,37358	10,12024	34,83607	10,06633

Resultados

Compilador com otimização



Resultados

Compilador com otimização

	Speedup's		
Tamanho da matriz	**Speedup t / o**	**Speedup vo / o**	**Speedup vt / t**
200	95,19%	98,91%	101,49%
400	95,48%	104,47%	102,31%
600	83,40%	94,91%	100,28%
800	64,62%	85,09%	101,66%
1000	56,08%	92,39%	101,60%
1200	51,43%	79,13%	101,19%
1400	52,11%	82,19%	101,13%
1600	39,74%	102,08%	101,34%
1800	30,82%	98,08%	99,52%
2000	29,44%	101,35%	99,47%

Resultados

Compilador com otimização

Tamanho da matriz	Tempo médio de transposição (seg)
200	0,00005
400	0,00016
600	0,00052
800	0,00101
1000	0,00183
1200	0,00279
1400	0,00375
1600	0,00802
1800	0,01375
2000	0,01739

Resultados

Speedup entre os compiladores

	Speedup gcc -O3 / gcc			
Tamanho da matriz	**Modo o**	**Modo t**	**Modo vo**	**Modo vt**
200	25,13%	25,69%	25,74%	25,99%
400	23,61%	26,39%	24,62%	26,94%
600	24,37%	27,41%	24,78%	27,56%
800	26,95%	28,45%	24,31%	29,09%
1000	26,03%	28,90%	29,00%	29,55%
1200	24,86%	29,07%	27,97%	29,48%
1400	24,37%	29,18%	25,67%	29,66%
1600	30,22%	29,26%	34,22%	29,84%
1800	37,91%	29,34%	45,61%	29,37%
2000	38,90%	29,39%	51,28%	29,26%

Resultados

Valgrind

	Modo o	Modo t	Modo vo	Modo vt	Compilador
D1 Miss Rate	5,40%	0,60%	5,40%	0,60%	**Sem otimização**
D Refs	2.895.547.901	2.898.878.037	2.895.476.227	2.898.806.314	
D1 Misses	155.237.180	16.163.033	155.321.302	16.099.686	
D1 Miss Rate	30,20%	4,10%	30,20%	4,10%	**Com otimização**
D Refs	515.273.478	391.346.045	515.202.515	391.274.590	
D1 Misses	155.439.202	16.164.397	155.527.198	16.101.046	

Referências

- https://www.inf.ufpr.br/roberto/ci067/14_alocmat.html
- https://matrixcalc.org/pt/