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

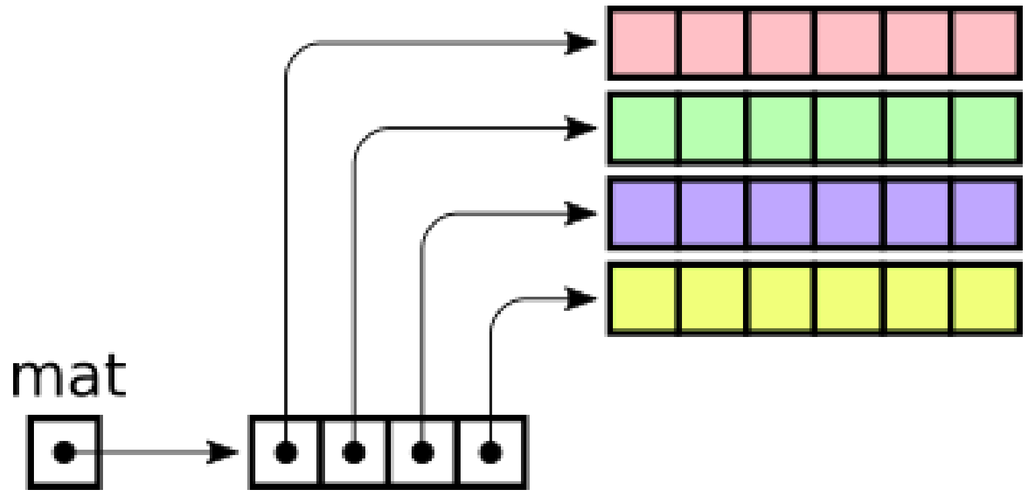
- Implementado em linguagem C;
- Possui 3 modos de multiplicação:

  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" (v):*** aloca a matriz B de maneira contígua na memória.

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

Algoritmos de Alocação das Matrizes

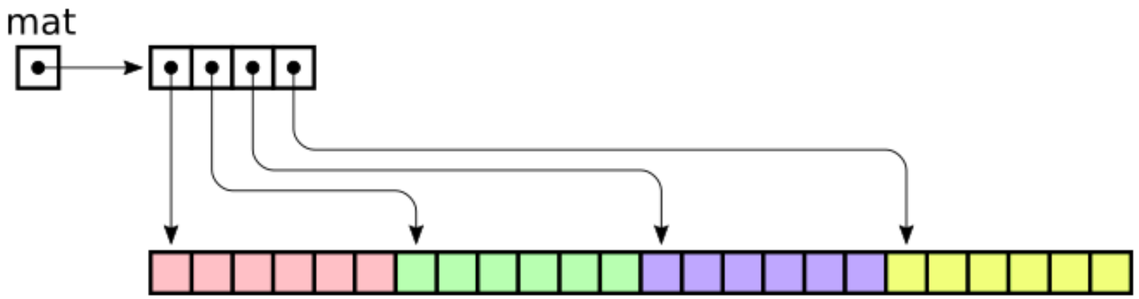
- Vetor de ponteiros de linhas separadas:




```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] + i + c;
    return M;
}
```

Algoritmo de Transposição

- Aloca a matriz transposta apenas se necessário

```c
double** matrizT(double **M,  int l, int c)
{
    int i, j;
    double **Mt;
    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 v out >> $DIRETORIO/v$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="Notebook03"

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

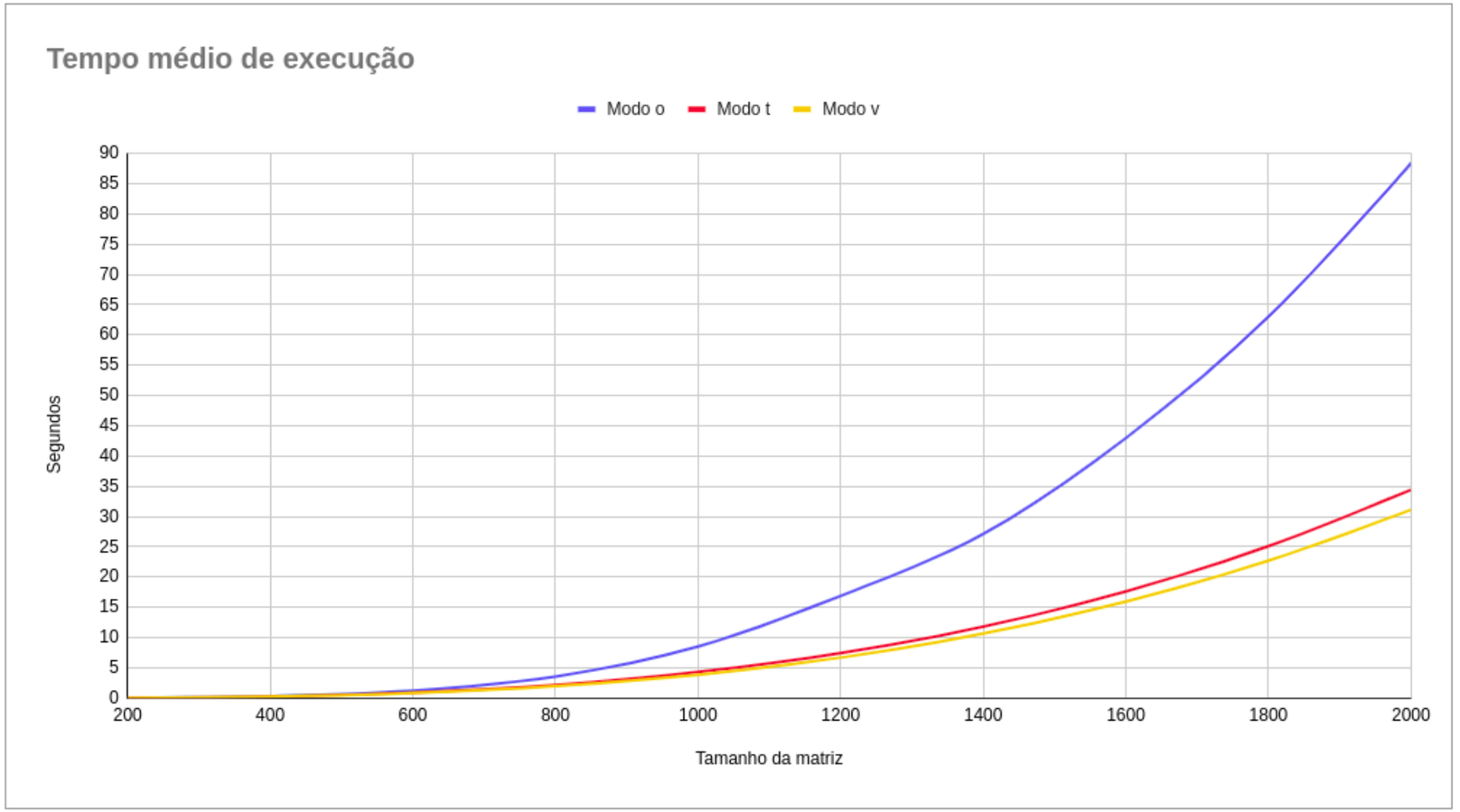
Resultados

Compilador sem otimização

	Tempo médio de execução (seg)					
Tamanho da matriz	**Modo o**	**Modo t**	**Modo v**	**Speedup t/o**	**Speedup v/o**	**Speedup v/t**
200	0,03637	0,03386	0,03196	93,10%	87,87%	94,39%
400	0,31679	0,27061	0,24775	85,42%	78,21%	91,55%
600	1,22797	0,91033	0,83203	74,13%	67,76%	91,40%
800	3,57023	2,18586	1,96873	61,22%	55,14%	90,07%
1000	8,51347	4,29998	3,84611	50,51%	45,18%	89,44%
1200	16,88843	7,42735	6,68901	43,98%	39,61%	90,06%
1400	27,10006	11,79696	10,67645	43,53%	39,40%	90,50%
1600	42,95819	17,63377	15,94094	41,05%	37,11%	90,40%
1800	63,01296	25,10047	22,70667	39,83%	36,03%	90,46%
2000	88,36005	34,43184	31,12419	38,97%	35,22%	90,39%

Resultados

Compilador sem otimização



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

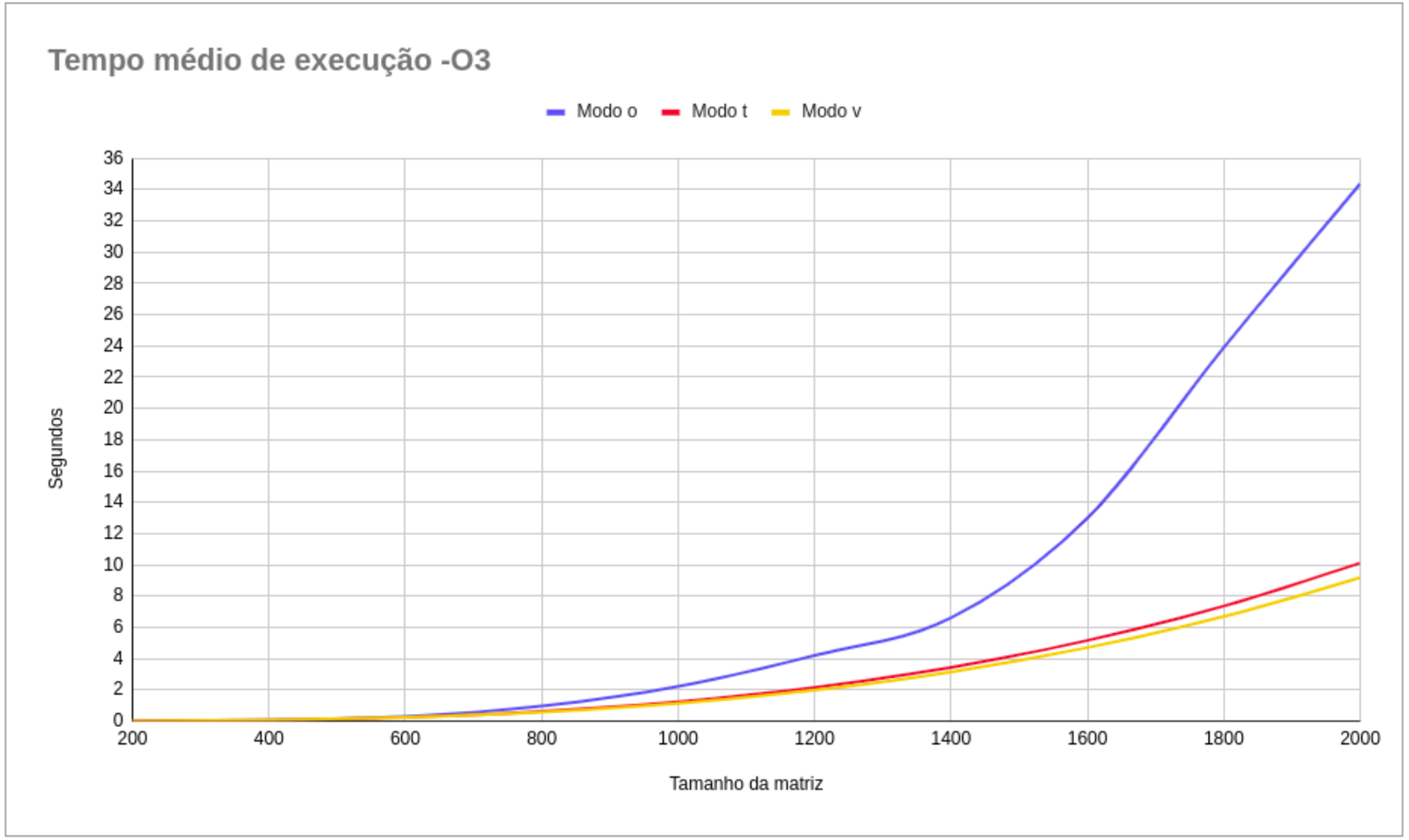
Resultados

Compilador com otimização

	Tempo médio de execução (seg)					
Tamanho da matriz	**Modo o**	**Modo t**	**Modo v**	**Speedup t/o**	**Speedup v/o**	**Speedup v/t**
200	0,00914	0,00870	0,00888	95,19%	97,16%	102,07%
400	0,07479	0,07141	0,07248	95,48%	96,91%	101,50%
600	0,29920	0,24954	0,24581	83,40%	82,16%	98,51%
800	0,96219	0,62180	0,58350	64,62%	60,64%	93,84%
1000	2,21586	1,24264	1,14216	56,08%	51,54%	91,91%
1200	4,19774	2,15882	1,97695	51,43%	47,10%	91,58%
1400	6,60480	3,44205	3,13945	52,11%	47,53%	91,21%
1600	12,98341	5,15988	4,69998	39,74%	36,20%	91,09%
1800	23,88930	7,36359	6,68777	30,82%	27,99%	90,82%
2000	34,37358	10,12024	9,17922	29,44%	26,70%	90,70%

Resultados

Compilador com otimização



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 gcc -O3 / gcc		
Modo o	**Modo t**	**Modo v**
25,13%	25,69%	27,78%
23,61%	26,39%	29,26%
24,37%	27,41%	29,54%
26,95%	28,45%	29,64%
26,03%	28,90%	29,70%
24,86%	29,07%	29,56%
24,37%	29,18%	29,41%
30,22%	29,26%	29,48%
37,91%	29,34%	29,45%
38,90%	29,39%	29,49%

Resultados

Valgrind

	Modo o	Modo t	Modo v	Compilador
D1 Miss Rate	5,40%	0,60%	0,00%	**Sem otimização**
D Refs	2.895.424.451	2.898.714.137	2.895.394.098	
D1 MIsses	124.571	16.160.772	98.496	
D1 Miss Rate	30,20%	4,10%	0,00%	**Com otimização**
D Refs	515.158.002	391.191.678	515.126.156	
D1 MIsses	155.437.492	16.162.123	98.479	

Referências

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