Decomposição de Cholesky OpenMP - Análises

Larissa Trindade

1. Contexto

Definição

Decomposição de Cholesky ou Fatoração de Cholesky é um método de álgebra linear para *resoluções de sistemas lineares.*

Para utilizar este método e necessário que a matriz do sistema linear seja **quadrada** (n x n), **simétrica** e **definida positiva.**

Para utilizar a decomposição de Cholesky é utilizado a equação (1) onde **A** é a *matriz inicial* e **L** é uma *matriz triangular inferior* com elementos da diagonal principal estritamente positivos.

$$A = LL^T$$

Especificações da máquina

Intel® Core™ i5-7200	
Core	2
Threads	4
Cache	3 MB Intel® Smart Cache
Memória	16 GB

Algoritmo

Diagonal Principal

$$l_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} l_{ik}^2}$$

Abaixo da Diagonal

$$l_{ji} = \frac{1}{l_{ii}} a_{ji} - \sum_{k=1}^{j-1} l_{jk} l_{ik}$$

novo

```
void cholesky(double** A, int n) {
  double s = 0;
  int diagonal = 0;
  int i, j, k;
  for (i = 0; i < n; i++) { //coluna
     s = 0;
     if (diagonal == 0) {
        diagonal = 1;
        for (k = 0; k < i; k++)  s += A[i][k] * A[i][k];
        A[i][i] = sqrt(A[i][i] - s);
     }
     for (j = i + 1; j <  n; j++) { //linha
        for (k = 0; k < i; k++)  s += A[j][k] * A[i][k];
        A[j][i] = (1.0 / A[i][i] * (A[j][i] - s));
        A[i][j] = A[j][i];
     }
     diagonal = 0;
   }
}
```

Algoritmo paralelizado

```
void cholesky(double** A, int n) {
  double s = 0;
  int diagonal = 0;
  int i, j, k;
  for (i = 0; i < n; i++) { //coluna
      s = 0;
      if (diagonal == 0) {
          diagonal = 1;
          for (k = 0; k < i; k++)  s += A[i][k] * A[i][k];
          A[i][i] = sqrt(A[i][i] - s);
      }
    #pragma omp parallel for shared(A, i, n) private(j, k, s)
    for (j = i + 1; j <  n; j++) { //linha
        for (k = 0; k < i; k++)  s += A[j][k] * A[i][k];
        A[j][i] = (1.0 / A[i][i] * (A[j][i] - s));
        A[i][j] = A[j][i];
    }
      diagonal = 0;
  }
}
```

$$l_{ji} = \frac{1}{l_{ii}} a_{ji} - \sum_{k=1}^{j-1} l_{jk} l_{ik}$$

Objetivo

- Análise dos tempos separados (diagonal principal e abaixo da diagonal)
- Verificar se paralelizando a diagonal principal melhora o tempo
- Analisar a aplicação de schedule para paralelizar o for

2. Análise

3 entradas analisadas e definidas pelo trabalho 1:

- *cholesky_5000.in*
- *cholesky_7000.in*
- *cholesky_10000.in*

Verificação: Próprio programa da Decomposição de Cholesky (caso algo esteja errado e a matriz não esteja correta, o arquivo .out possui algumas linhas com **#INDO**)

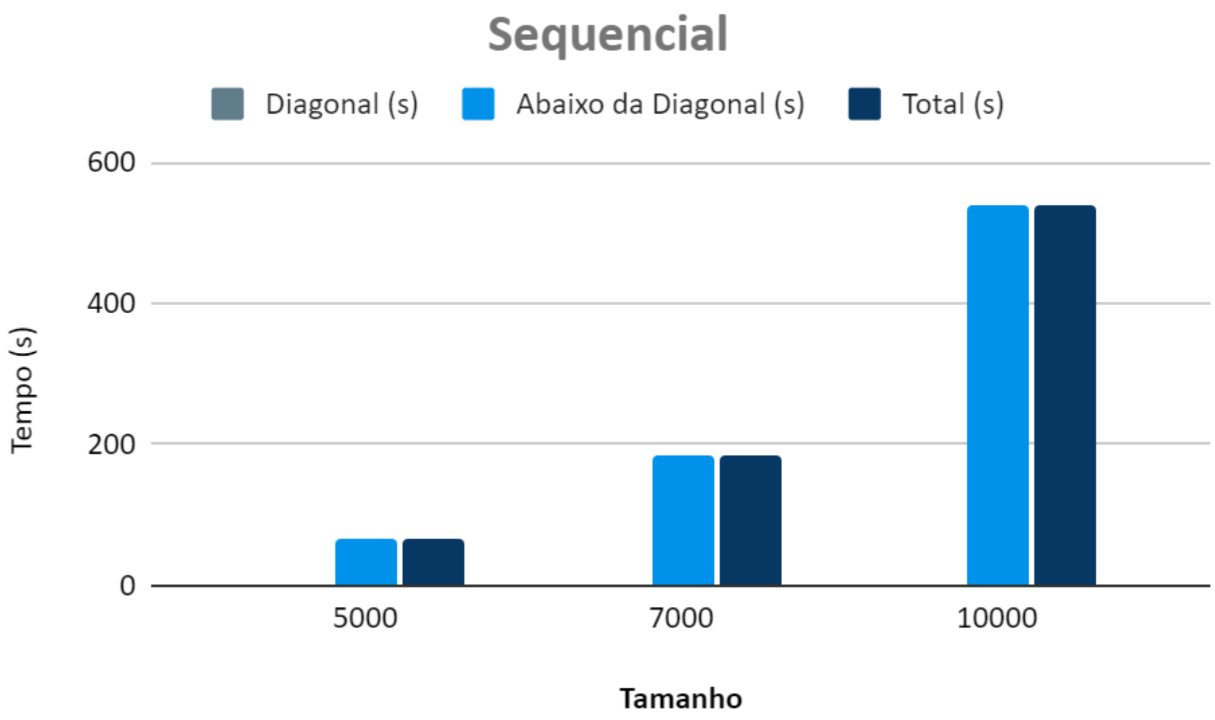
Análise - tempos separados

Sequencial

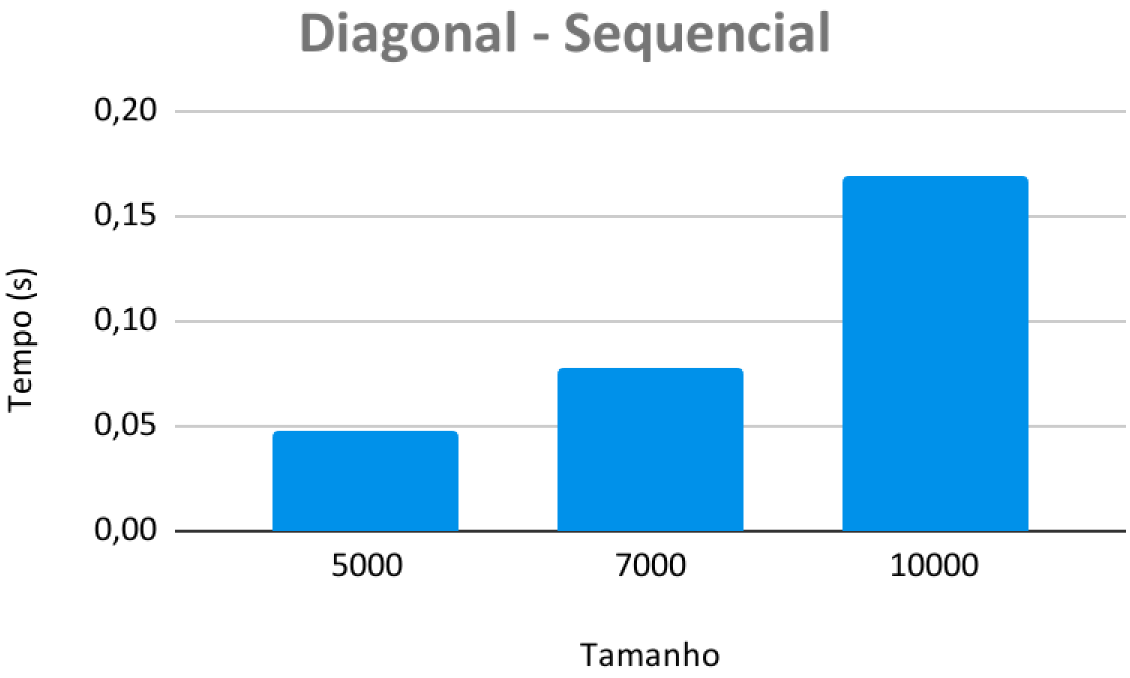
Tamanho	Diagonal (s)	Abaixo da Diagonal (s)	Total (s)
5000	0,048	68,09	68,14
7000	0,078	185,12	185,93
10000	0,169	540,03	540,20

Média realizada com a remoção dos outliers

Análise- tempos separados



Análise- tempos separados



3. Utilizando Reduction

Usando Reduction

```c
void cholesky(double** A, int n) {
  double s = 0;
  int diagonal = 0;
  int i, j, k;
  for (i = 0; i < n; i++) { //coluna
     s = 0;
     if (diagonal == 0) {
        diagonal = 1;
        for (k = 0; k < i; k++)  s += A[i][k] * A[i][k];
        A[i][i] = sqrt(A[i][i] - s);
     }
    #pragma omp parallel for shared(A, i, n) private(j, k, s)
    for (j = i + 1; j <  n; j++) { //linha
       for (k = 0; k < i; k++)  s += A[j][k] * A[i][k];
       A[j][i] = (1.0 / A[i][i] * (A[j][i] - s));
       A[i][j] = A[j][i];
     }
     diagonal = 0;
  }
}
```

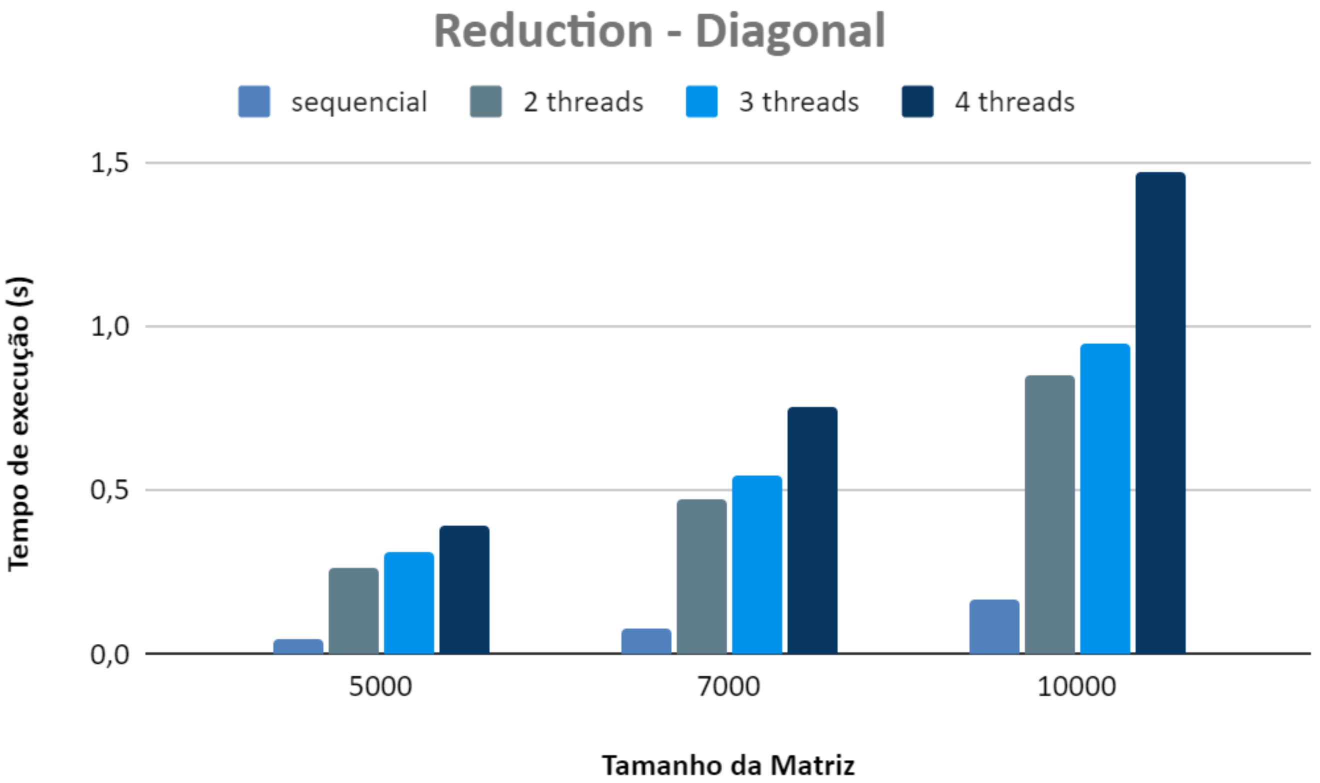
Reduction

```c
int csum = 0;
gettimeofday(&tstart, NULL);
#pragma omp parallel private(k) reduction
(+:csum)
for (k = 0; k < i; k++){
   csum += A[i][k] * A[i][k];
}
gettimeofday(&tend, NULL);
 A[i][i] = sqrt(A[i][i] - csum);
```

Tempo médio (aplicando Reduction) - Diagonal

Tamanho	Sequencial (s)	2 Threads (s)	3 Threads (s)	4 Threads (s)
5000	0,048	0,26	0,31	0,39
7000	0,08	0,47	0,52	0,75
10000	0,17	0,85	0,95	1,47

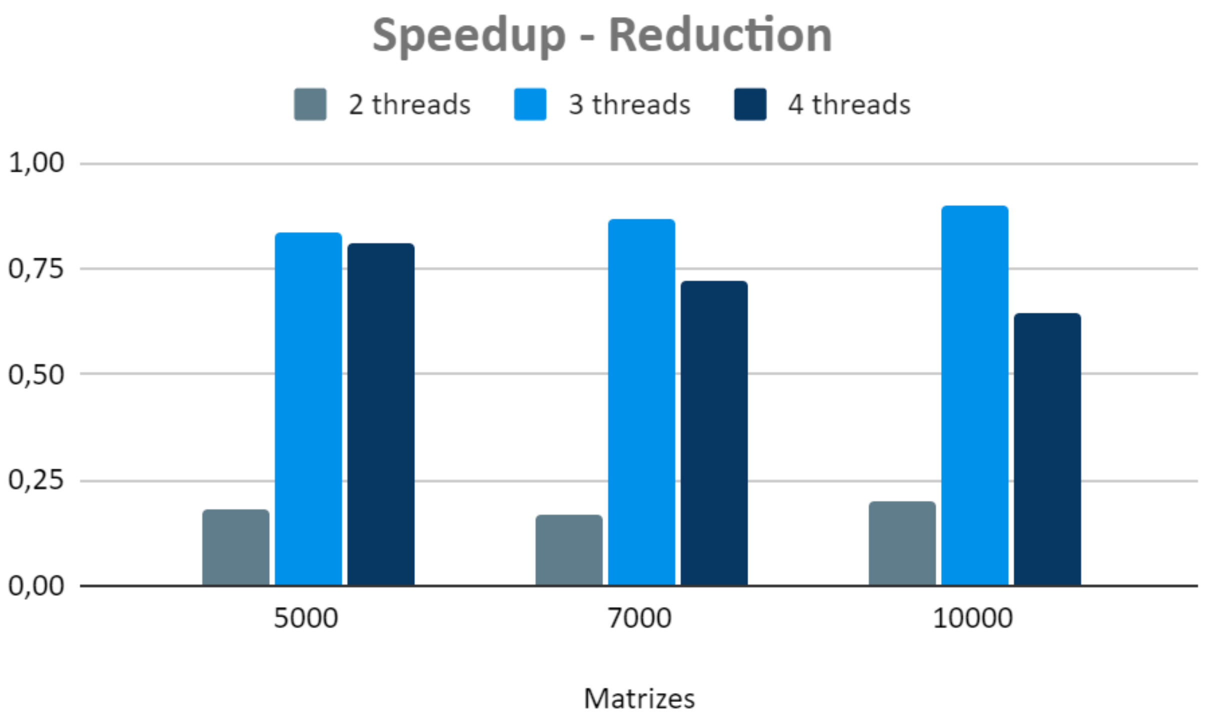
Média realizada com a remoção dos outliers


Reduction - Diagonal
sequencial
2 threads
3 threads
4 threads
1,5
1,0
0,5
0,0
Tempo de execução (s)
5000
7000
10000
Tamanho da Matriz

Speedup – Reduction (Diagonal)

Tamanho	2 Threads (s)	3 Threads (s)	4 Threads (s)
5000	0,18	0,84	0,81
7000	0,17	0,86	0,72
10000	0,20	0,96	0,65

Média realizada com a remoção dos outliers


Speedup - Reduction
2 threads
3 threads
4 threads
1,00
0,75
0,50
0,25
0,00
5000
7000
10000
Matrizes

3. Utilizando Schedule

Usando Schedule

```
void cholesky(double** A, int n) {
  double s = 0;
  int diagonal = 0;
  int i, j, k;
  for (i = 0; i < n; i++) { //coluna
      s = 0;
      if (diagonal == 0) {
          diagonal = 1;
          for (k = 0; k < i; k++)  s += A[i][k] * A[i][k];
          A[i][i] = sqrt(A[i][i] - s);
      }
    #pragma omp parallel for shared(A, i, n) private(j, k, s)
    for (j = i + 1; j <  n; j++) { //linha
        for (k = 0; k < i; k++)  s += A[j][k] * A[i][k];
        A[j][i] = (1.0 / A[i][i] * (A[j][i] - s));
        A[i][j] = A[j][i];
    }
      diagonal = 0;
  }
}
```

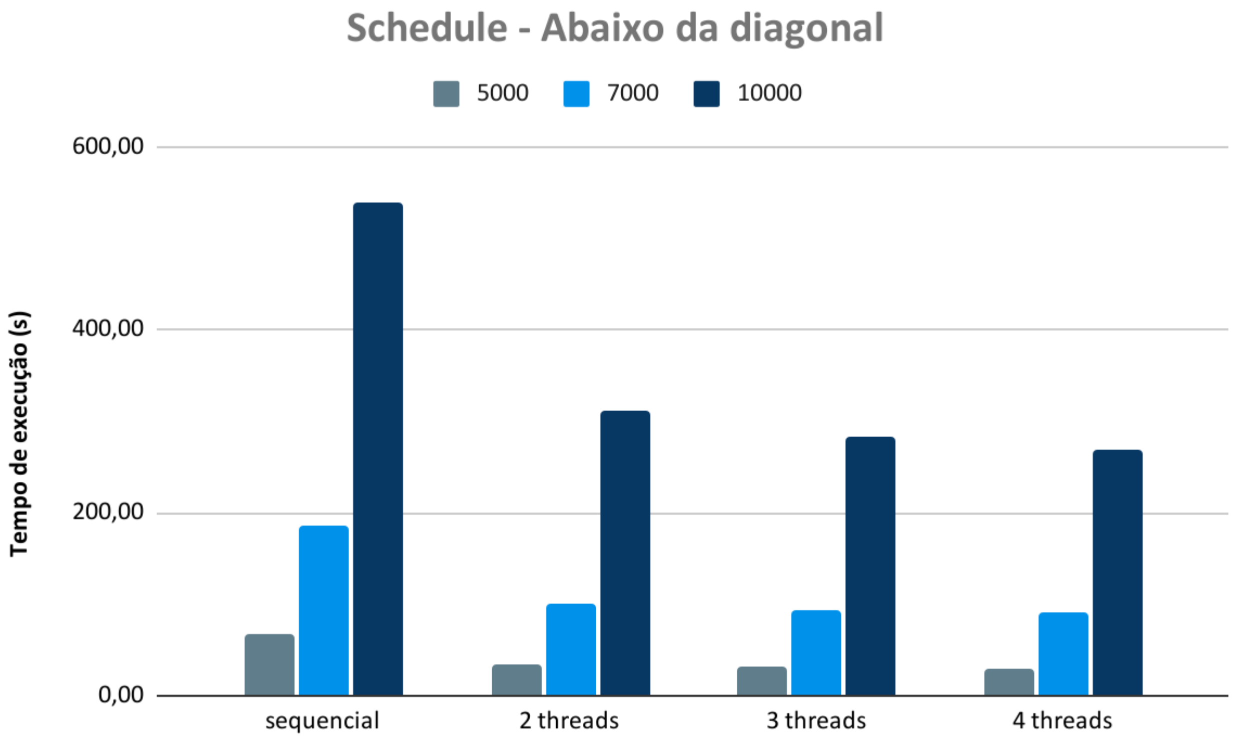
Schedule

```
gettimeofday(&tstart, NULL);
#pragma omp parallel for shared(A, i, n)
private(j,k,s) schedule(dynamic)
    for (j = i + 1; j <  n; j++) { //linha
        for (k = 0; k < i; k++)  s += A[j][k] * A[i][k];
        A[j][i] = (1.0 / A[i][i] * (A[j][i] - s));
        A[i][j] = A[j][i];
    }
gettimeofday(&tend, NULL);
```

Tempo médio (aplicando Schedule) – Abaixo da Diagonal

Tamanho	Sequencial (s)	2 Threads (s)	3 Threads (s)	4 Threads (s)
5000	68,09	35,25	32,47	30,50
7000	185,84	99,81	94,00	92,30
10000	540,02	312,17	284,11	269,12

Média realizada com a remoção dos outliers


Schedule - Abaixo da diagonal
5000
7000
10000
600,00
400,00
200,00
0,00
Tempo de execução (s)
sequencial
2 threads
3 threads
4 threads

Speedup – Comparação (abaixo da diagonal)

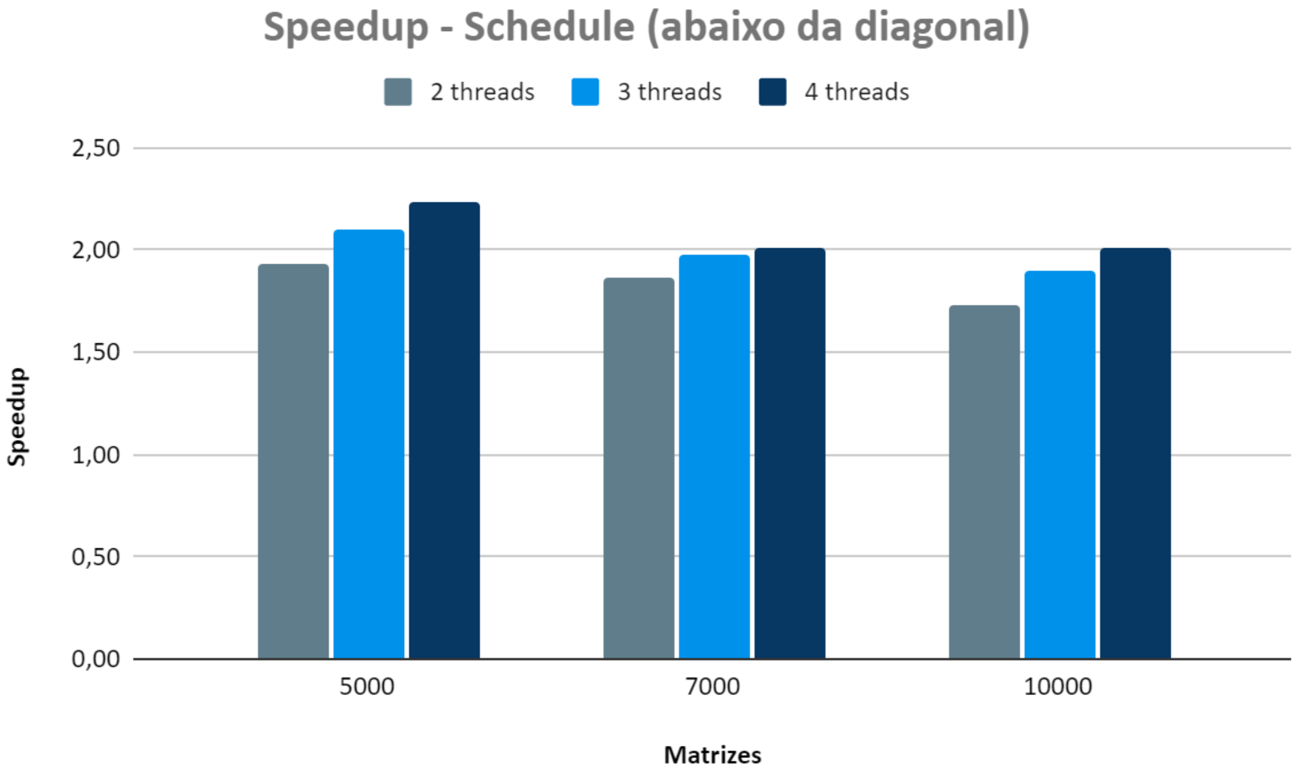
Antigo

Tamanho	2 Threads	3 Threads	4 Threads
5000	1,74	1,59	1,91
7000	1,72	1,53	1,86
10000	1,64	1,56	1,84

Com Schedule

Tamanho	2 Threads	3 Threads	4 Threads
5000	1,93	2,10	2,23
7000	1,86	1,96	2,01
10000	1,73	1,90	2,01

Média realizada com a remoção dos outliers


Speedup - Schedule (abaixo da diagonal)
2 threads
3 threads
4 threads
2,50
2,00
1,50
1,00
0,50
0,00
Speedup
5000
7000
10000
Matrizes

4. Resultado Final

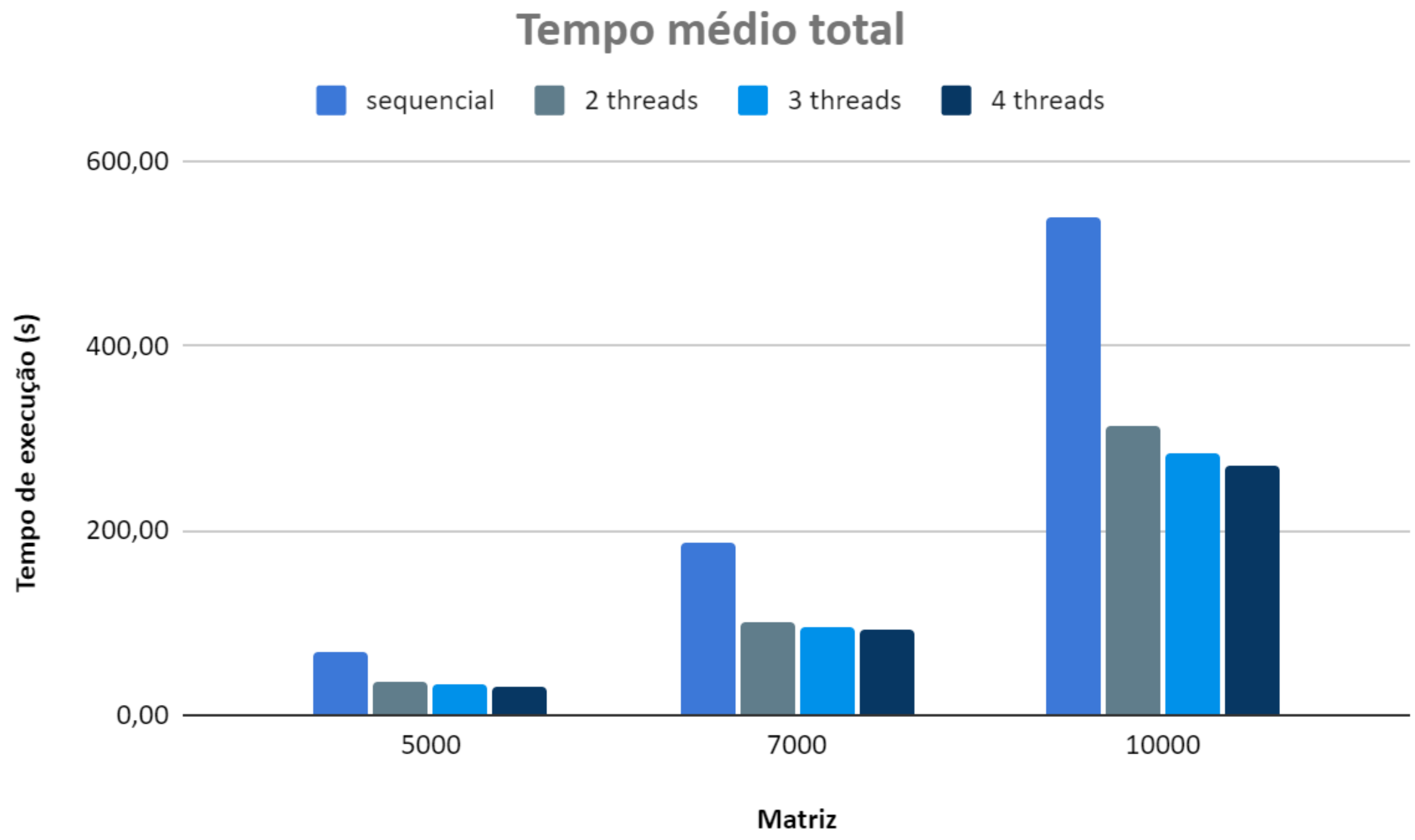
Resultado Final

- Não será utilizado o Reduction
- Será utilizado o Schedule

Tempo Médio Total (novo)

Tamanho	Sequencial (s)	2 Threads (s)	3 Threads (s)	4 Threads (s)
5000	60,14	35,29	32,54	30,29
7000	185,92	99,90	94,12	92,43
10000	540,20	312,39	284,29	269,37

Média realizada com a remoção dos outliers


Tempo médio total
sequencial
2 threads
3 threads
4 threads
600,00
400,00
200,00
0,00
Tempo de execução (s)
5000
7000
10000
Matriz

Speedup – Comparação (Total)

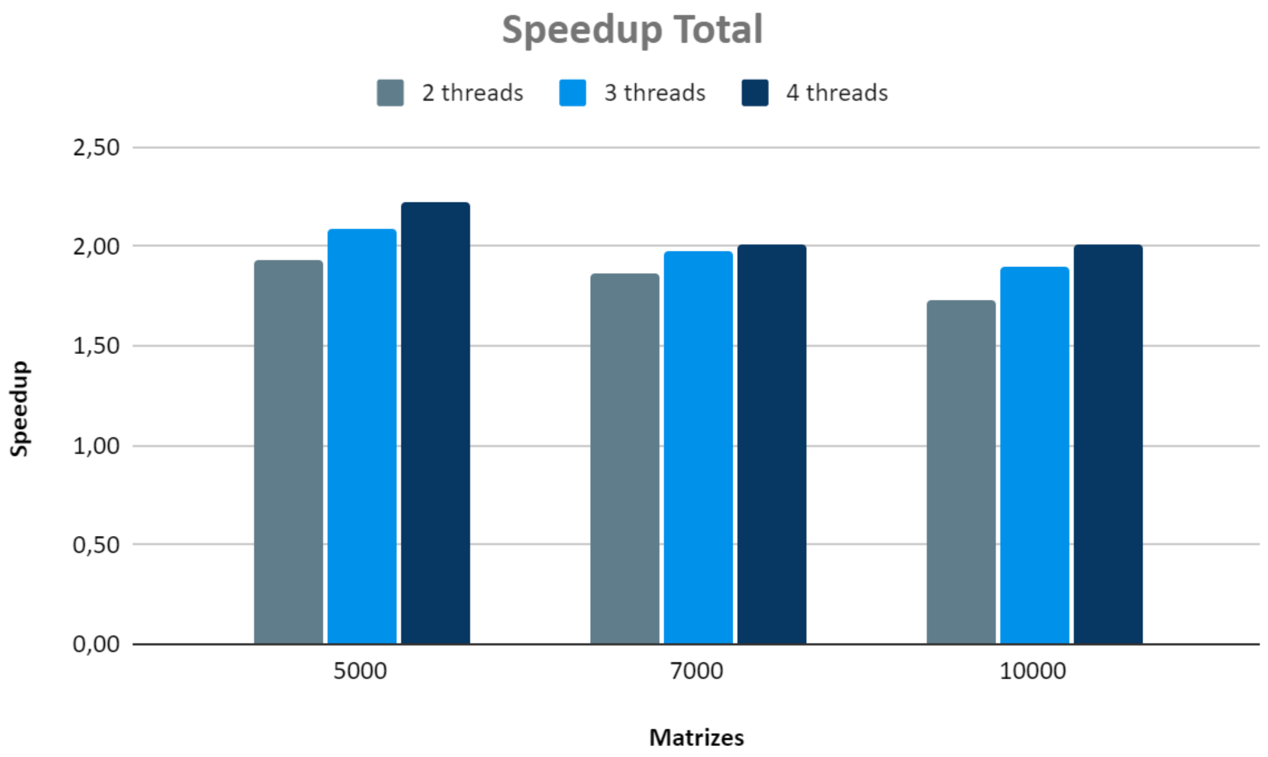
Antigo

Tamanho	2 Threads	3 Threads	4 Threads
5000	1,70	1,53	1,90
7000	1,69	1,63	1,89
10000	1,62	1,60	1,91

Novo

Tamanho	2 Threads	3 Threads	4 Threads
5000	1,93	2,09	2,23
7000	1,86	1,98	2,01
10000	1,73	1,90	2,01

Média realizada com a remoção dos outliers


Speedup Total
2 threads
3 threads
4 threads
2,50
2,00
1,50
1,00
0,50
0,00
Speedup
5000
7000
10000
Matrizes

Obrigada!

lala.trindade.2008@gmail.com

https://github.com/LarissaTrin/CholeskyDecomposition