




Function Six

Patrick Ferraz e Jefson Matos



O problema

- Calcular a média dos valores das linhas de uma matriz $l \times c$
 - Sequencial
 - Paralelo
- Identificar a maior diferença entre as médias encontrada com ambos resultados
- Calcular o speedup e eficiência para matrizes de tamanho variados com 'n' processamentos



O problema - Sequencial

```
/**
 * Laço para percorrer linhas da matriz
 * somar os valores e calcular a média.
 * lsize = linha; csize = coluna
 */
for (size_t i = 0; i < lsize; i++)
{
    media[i] = 0.0;
    for (size_t j = 0; j < csize; j++)
        media[i] += A[i*csize + j];
    media[i] = media[i]/(double)csize;
}
```



O problema - Paralelo

```
// Setando o número de threads que serão utilizadas
omp_set_num_threads(num_threads);
/**
 * Laço paralelo para percorrer linhas da matriz
 * somar os valores e calcular a média.
 * lsize = linha; csize = coluna
 */
#pragma omp parallel for schedule(dynamic)
for (size_t i = 0; i < lsize; i++)
{
    media[i] = 0.0;
    for (size_t j = 0; j < csize; j++)
        media[i] += A[i*csize + j];
    media[i] = media[i]/(double)csize;
}
```

static



Metodologia de teste

- Comprovação do desempenho:
 - Processamento sequencial - paralelo 2, 4 e 8
 - Matrizes
 - 2.048 x 4.096 (x2 até) 16.384 x 32.768
 - 5.120 x 5.120 (x2 até) 40.960 x 40.960
 - Quantidade de processamento: 50, 25, 12, 6 respectivamente
 - Geração da maior diferença encontrada entre as médias
- Comprovação de tempo:
 - `omp_get_wtime()` para cada chamada e registro do menor e maior
- Script para processar e gerar saída com todos resultados

Metodologia de teste

```
typedef struct registro
{
    double diff;
    double maior_tempo;
    double menor_tempo;
    size_t num_threads;
} tRecord;
```

```
#!/bin/bash
#SBATCH -J MatrixLine
#SBATCH -p long
#SBATCH -N 1
#SBATCH -n 1
#SBATCH -c 8
```

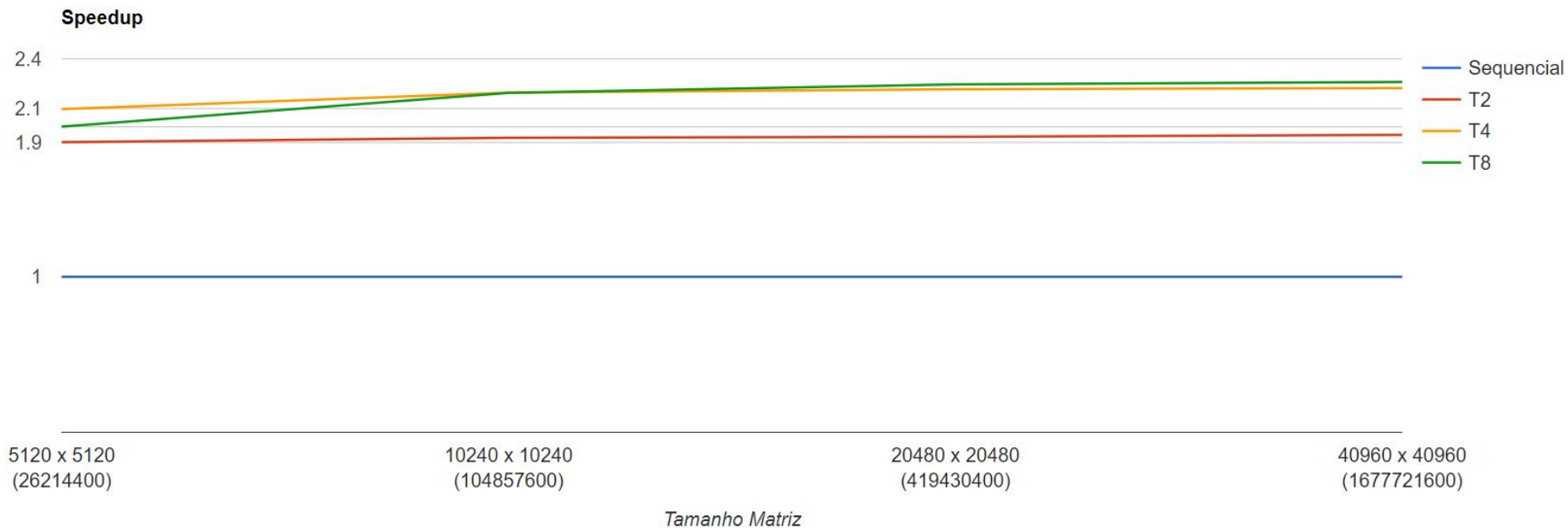
```
srun ./run 5120 5120
```

```
for (size_t t = 1, nt = 2; nt <= numThreads; nt *= 2, t++)
{
    r[t].num_threads = nt;
    r[t].maior_tempo = 0.0;
    r[t].menor_tempo = umax;
    for (size_t j = qtProcess; j > 0; j--)
    {
        r[t].diff = processMatrix(A, lsize, csize, nt, &ptime);
        if (r[0].maior_tempo < ptime.tempo_sq) r[0].maior_tempo = ptime.tempo_sq;
        if (r[0].menor_tempo > ptime.tempo_sq) r[0].menor_tempo = ptime.tempo_sq;

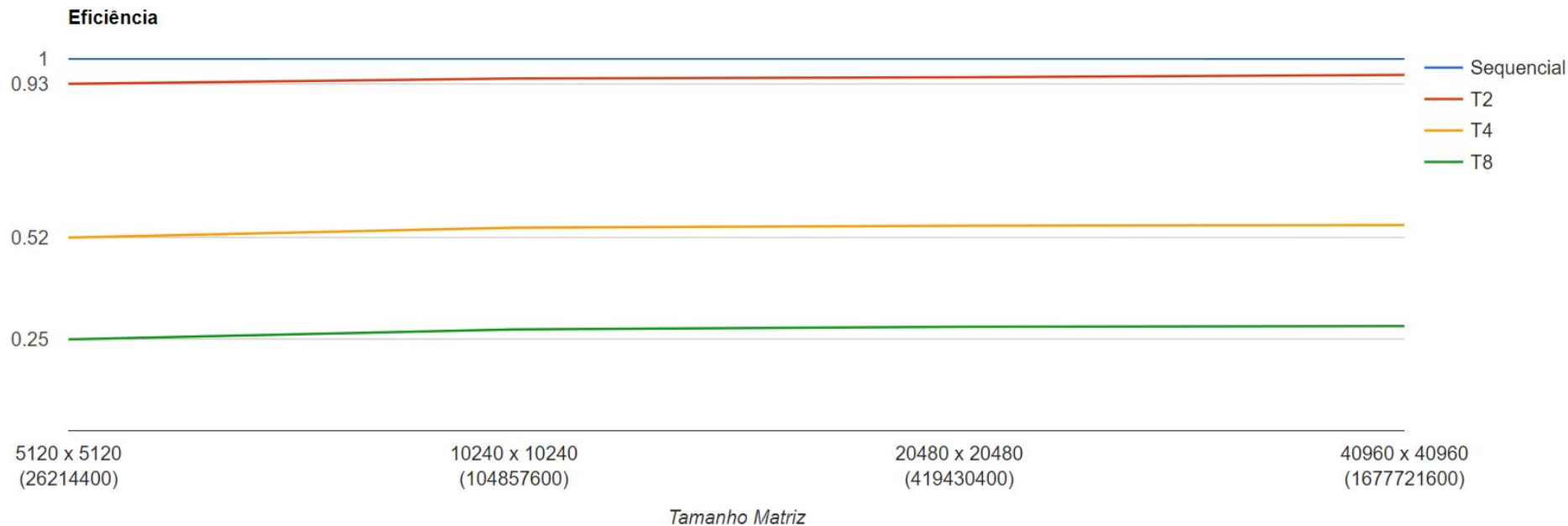
        if (r[t].maior_tempo < ptime.tempo_pp) r[t].maior_tempo = ptime.tempo_pp;
        if (r[t].menor_tempo > ptime.tempo_pp) r[t].menor_tempo = ptime.tempo_pp;
    }
}

recordGenerator(r, qtRecord, lsize, csize, qtProcess, f_out);
lsize <=> growMatrix; // *2
csize <=> growMatrix; // *2
if (qtProcess >= 6) qtProcess >=> growProcess; // /2
```

Resultados - Speedup



Resultados - Eficiência





Resultados - Dynamic

Tamanho matriz: 10240 x 10240 (104857600)

Quantidade de processamento: 25

N Thread(s)	Tempo		Maior Diff	Speedup	Eficiencia
	min	max			
1	0.295095	0.296529	0.000000	1.000000	1.000000
2	0.273215	0.274017	0.000000	1.080081	0.540041
4	0.143590	0.146351	0.000000	2.055123	0.513781
8	0.146231	0.146774	0.000000	2.018009	0.252251

Tamanho matriz: 40960 x 40960 (1677721600)

Quantidade de processamento: 6

N Thread(s)	Tempo		Maior Diff	Speedup	Eficiencia
	min	max			
1	4.735877	4.746447	0.000000	1.000000	1.000000
2	2.628494	2.632325	0.000000	1.801746	0.900873
4	2.271081	2.277200	0.000000	2.085296	0.521324
8	2.283173	2.285442	0.000000	2.074253	0.259282



Discussão e conclusões

- Tamanho não é documento