Para matrizes de dimensão 500:

- Menor tempo sequencial: 0.503571 segundos
- Menor tempo concorrente utilizando 1 thread: 0.645929 segundos
- Menor tempo concorrente utilizando 2 threads: 0.354304 segundos
- TSequencial/TConcorrente(1 thread) = 0.779607
- TSequencial/TConcorrente(2 threads) = 1.421296

Para matrizes de dimensão 1000:

- Menor tempo sequencial: 4.886779 segundos
- Menor tempo concorrente utilizando 1 thread: 5.963380 segundos
- Menor tempo concorrente utilizando 2 threads: 3.072196 segundos
- TSequencial/TConcorrente(1 thread) = 0.819465
- TSequencial/TConcorrente(2 threads) = 1.590647

Para matrizes de dimensão 2000:

- Menor tempo sequencial: 66.993126 segundos
- Menor tempo concorrente utilizando 1 thread: 73.046873 segundos
- Menor tempo concorrente utilizando 2 threads: 40.229269 segundos
- TSequencial/TConcorrente(1 thread) = 0.917125
- TSequencial/TConcorrente(2 threads) = 1.665283

Como a minha solução é O(dim^3), acho que é esperado o tempo aumentar bastante com o aumento da dimensão das matrizes, mas o aumento foi um pouco mais que cúbico (como dobramos a entrada, deveria multiplicar mais ou menos por 8 o tempo de execução, e foi um pouco mais, principalmente de 1000 para 2000).

Meu computador tem 4 núcleos e 4 processadores lógicos.