|  |  |  |
| --- | --- | --- |
| **Método** | **Valor de PI** | **Tempo** |
| Sequencial | 3.14141 | 4.413 |
| Paralelo OMP Crit | 2.0942 | 36.981 |
| Paralelo Seed per Task | 3.14163 | 3.19182 |

**Relatório 14**

Houve melhora significativa em eficiência entre o método sequencial e paralelo utilizando seeds diferentes por task, porém o método utilizando a mesma seed para todas as tasks apresentou resultados consideravelmente inferiores, tanto em precisão quanto desempenho.

A melhor estimativa de pi foi para o método sequencial, não muito diferente do método paralelo com várias seeds, diferentemente do método paralelo de mesma seed, que foi muito inferior em precisão.

O problema da seeds foi solucionado gerando uma nova seed para cada thread, para que exista efetivamente a aleatoriedede no código.

Para a paralelização do algoritmo existe a necessidade de gerar uma seed em função de alguma varíavel do código, uma vez que não queremos que as diferentes seeds sejam iguais, impossibilitando a existência efetiva de aleatoriedade no código.

O uso de threads trouxe benefícios em tempo de execução, porém não diferenciou muito do método sequencial para a precisão do algoritmo, sendo inclusive inferior.