**Documentation**

**Tests:**

Matrix of 10x10

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Language/Size | 10x10 | 50x50 | 100x100 | 1000x1000 |
| Java :Tasks | Total execution time: 0  Sum statistics:  Fastest time: 0millis  Number of threads: 2 | Total execution time: 0  Sum statistics:  Fastest time: 0millis  Number of threads: 3 | Total execution time: 0  Sum statistics:  Fastest time: 6millis  Number of threads: 9 | Total execution time: 16  Sum statistics:  Fastest time: 1240millis  Number of threads: 3 |
| C++ : thread pool | Multiplication: 0 threads  Sum : 2 threads | Multiplication : 0 threads  Sum: 7 threads | Multiplication: 0 threads  Sum:8 threads | Fastest sum 800millis  Fastest product:more than 5 minutes  Multiplication:??  Sum: 9 threads |
| C++: futures | Mutiplication: 0 threads  Sum:2 threads | Multiplication:8 threads  Sum:9 threads | Multiplication: 8 threads  Sum:8 threads | Fastest sum:721 millis  Fastest product:  800 millis  Multiplication : 9 threads  Sum:9 threads |
| Java:Futures | Total execution time: 1  Product statistics:  Fastest time: 2milis  Number of threads: 14 | Total execution time: 1  Product statistics:  Fastest time: 32milis  Number of threads: 7 | Total execution time: 1  Product statistics:  Fastest time: 100milis  Number of threads: 7 | Total execution time: 101  Product statistics:  Fastest time: 10024milis  Number of threads: 2 |

**Execution:**

The execution determines the optimal number of threads for each implementation and size of the matrix

**Time:**

As a conclusion the best configuration on my computer is the C++ implementation with futures and 9 threads. I think that as the input data increases the best number of threads is similar with the number of threads of my processor and C++ tends to be faster being the lowest level high level programming language.

**Hardware**

