High-level Pseudo-code

1. Read the matrices from two files and placed the data into 2D arrays in python

ie Matrix A and Matrix B

1. send the data in matrices to two mappers
2. The mappers will do a row-major and column-major splitting of the matrices and label them with ranks like in MPI, Then information labelled information will be sent to reducers to do the multiplication.
3. Then the results will be yielded to the system and pleased in a Matrix C

Above it the high-level overview of the way implement the mapReduce multiplication using Mrs MapReduce.

|  |  |  |  |
| --- | --- | --- | --- |
| Size N | Execution time with 1 task | Execution time with 64 tasks | Execution time with 256 tasks |
| 16 | 0.05s | 0.15s |  |
| 80 | 0.2s | 0.25s | 0.31s |
| 160 | 0.35s | 0.69s | 0.888s |
| 320 | 9.515s | 3.6s | 4.41s |
| 640 | 76s | 26s | 30s |
| 1280 | 720s | 240s | 240s |

After analysing these results, I got to know that MapReduce was only efficient if utilised well.

Instances where the matrices size is small, there is no point in trying to use MapReduce to solve it. Also the number of task allocation is a key factory in determining the efficiency of the multiplication.