Erlang - Matrix Multiplication

Dipankar Niranjan (201503003) & Susobhan Ghosh (201503005)

PROJECT GOAL: To design distributed matrix multiplication using Erlang with two granularities.

GRANULARITIES:
Assuming we're multiplying two matrices A and B, we've implemented the following granularities:

1. Row X Column: Sending a row of A and a column of B to a node.
2. Row X Matrix: Sending a row of A, and entire B matrix to a node.

RESULTS: These tests were all carried out on one single machine.
Nodecount = 8

Matrix Size / Granularity	Row X Column (μs)	Row X Matrix (μs)
2x100, 100x4	1926	1001
50x50, 50x50	409151	13334
50x10, 10x5	43429	8416
2x2, 2x1	668	705
3x2, 2x1	937	999
3x5,5x1	801	1041

Nodecount = 4

Matrix Size / Granularity	Row X Column (μs)	Row X Matrix (μs)
2x100, 100x4	1986	1024
50x50, 50x50	518893	21051
50x10, 10x5	75888	9575
2x2, 2x1	722	910
3x2, 2x1	821	834
3x5,5x1	970	1021

CONCLUSION: From our tests, we have inferred the following:

1. Decreasing the nodecount increases the total computation time for both the granularities. This is due to decreasing number of computational resources available to perform the matrix multiplication
2. Row X Matrix granularity seems to perform faster except when the number of [row x column] or [row x matrix] multiplications are less than number of actual nodes available (nodecount), and number of nodes used is same for both because in this case, granularity 1 reduces to granularity 2, and the results become comparable.

   This maybe due to the fact that there is a communication overhead involved in the first granularity, and also, there is a performance boost in the second granularity due to locality of references. This might change if we have an actual distributed system with multiple machines.