Illustrate how did you implement each of the following phases in your program?

1. Partitioning scheme of tasks and data required to compute 𝐶

Po tasks: initialize matrix 1 and answer

Fill the matrixes

Broadcast matrix2, m,n,k

Scatter matrix 1 to each process

Gather answer from each process

Print the result

All processes tasks: initialize matrix 2

Multiply two matrixes based on local variables

Find the total time taken by program

C requires x + 3 matrixes of memory where x is the number of processes

1. Communication required to partition A and B

4 broadcasts for variables matrix2, m, n, k

Scatter matrix1 to local\_matrix1

Gather local\_answer to answer

1. Aggregation of tasks into larger tasks

My program segments the problem into x blocks of identical parallel performance where x is the number of processes

1. Mapping of larger tasks to processes

My program is top heavy as process 0 does most of the work to set things up and to ensure the other programs have what they need to matrix multiply

Table 1-Run Time

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Matrix Size | | | |
| Comm\_sz | 32x32 | 64x64 | 128x128 | 256x256 |
| 1 | 0.000160635 | 0.00123898 | 0.0133671 | 0.123064 |
| 2 | 0.000132699 | 0.000712076 | 0.00613004 | 0.0622268 |
| 4 | 0.000107805 | 0.00044037 | 0.00278072 | 0.031538 |
| 8 | 0.000223966 | 0.000399419 | 0.00163229 | 0.0312468 |

Table 2-Speedup

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Matrix Size | | | |
| Comm\_sz | 32x32 | 64x64 | 128x128 | 256x256 |
| 1 | 1 | 1 | 1 | 1 |
| 2 | 1.21052156 | 1.73995 | 2.18059 | 1.97767 |
| 4 | 1.49005148 | 2.8135 | 4.80706 | 3.90209 |
| 8 | 0.7172294 | 3.10196 | 8.18917 | 3.93845 |

Table 3-Efficiency

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Matrix Size | | | |
| Comm\_sz | 32x32 | 64x64 | 128x128 | 256x256 |
| 1 | 100 | 100 | 100 | 100 |
| 2 | 60.5260778 | 86.9977 | 109.029 | 98.8834 |
| 4 | 37.251287 | 70.3374 | 120.177 | 97.5522 |
| 8 | 8.96536751 | 38.7744 | 102.365 | 49.2306 |

2) A discussion of how you parallelized the problem according to PCAM methodology

Partitioning: a scatter statement divides the work into equal chucks where each process

does the same thing

Communication: 4 broadcasts of data before partitioning the labour

Agglomeration: my program broadcasts only one matrix, the minimum I need for matrix

multiplication, the rest of the data is split into locals which takes the same space

as the original matrixes, my functions also contain no globals

Mapping: My program has a poor mapping as most of the load is on process 0 but

MPI\_Bcast, MPI\_Scatter and MPI\_Gather all use tree structures to reduce computations by log(n)

3) Interpretation of results (speed up and efficiency)

I noticed that all tasks sped up with more processors. The speed up was more noticeable with the larger matrixes.

Efficiency wise with the exception of 128x128 (must have been a fluke) efficiency went down as processors and matrix sizes went up.

To conclude as we add more processes we should find more speed up but each process is less and less efficient individually while working on the program.