Analysis

MandelbrotSet Analysis:

From the efficiency graph, we can conclude that hiding communication latencies and improving parallel efficiency has contributed only to a small extent in improving efficiency. When compared to Fibonacci tasks, the number of decomposed tasks is less in MandelbrotSet and therefore not many communications takes place between computers and space. However ameliorating communication latencies have improved parallel efficiency by 8 % compared to without any ameliorations.

Fibonacci Analysis:

As mentioned in the class, Fibonacci is the most interesting problem in this experiment. It involves many subtasks and also many compose tasks. Since amelioration is done by executing compose tasks within space and also utilizing multi threading and asynchronous communication between space and computers, I was able to hide communication latency. This could be seen from the efficiency of upto 90%.

TSP Analysis:

TSP problem using tighter lower bound in branch and bound strategy yields very good improvement in performance. By ameliorating communication latencies by utilizing multithreading and also compose tasks are light weight which is executed in space itself. All these factors gives an 10 % improvement in parallel efficiency. When the number of computers increase communication between computers for setting the shared value is extra overhead. This could be seen from the time taken for computation when number of computers increase. Hiding it to an extent gave good improvement in efficiency.

Future Improvement in Infrastructure:

Infrastructure can be improved by implementing strategies like task caching. Incorporating execution of multiple jobs at the same time with the help of powerful language feature Java Reflection will be challenging. Also implementing logging feature for aggregating all the average task execution average times across computers will be helpful in designing the infrastructure dynamically. It will allow how much hosts will be a bottleneck for Space. Thus infrastructure can be made elastic to grow or shrink based on the amount of computations involved.