Assignment 03
Parallel Programming

06-02-01-10-51-14-1-11051
Meesala Raviteja

Minimum Degree Reordering:

Implementation:
The total operations performed in a minimum degree reordering can be summarised as follows:

- Find minimum degree Node
- Break- tie using a heuristic. (implemented by choosing the minimum current vertex number)
- Delete Node from the graph, while making sure all the neighbours are connected to each other.
- Number the deleted nodes sequentially.

A graph data type was used for the implementation, with the above operations defined as routines. The algorithm is non parallelizable. The only scope could be to parallelize the node search. However, this doesnt improve the overall order of complexity involved.

The total time taken : 5.342 seconds
Total fills resulted : 31035

Nested Dissection (using Kernighan – Lin Algorithm):

Nested dissection is a divide and conquer heuristic. The idea here is to partition the graph using the Kernighan Lin algorithm, which tries to separate a given sparse graph with least number of separators possible.

Implementation (key ideas):

- Divide graph to two equal parts, having same number of nodes.
- Run through the KL algorithm, to improve the efficiency of the division, by reducing the separator nodes.
- Generate A,B and V the two subgraphs and a partition set.
- Push A and B into a process queue, with other information useful for ordering.
- Select a partition list from B and remove it from B. Number the partition set from last.

Parallelism:
Initially, the communication pattern is established based on the number of partitions required.

Key ideas:

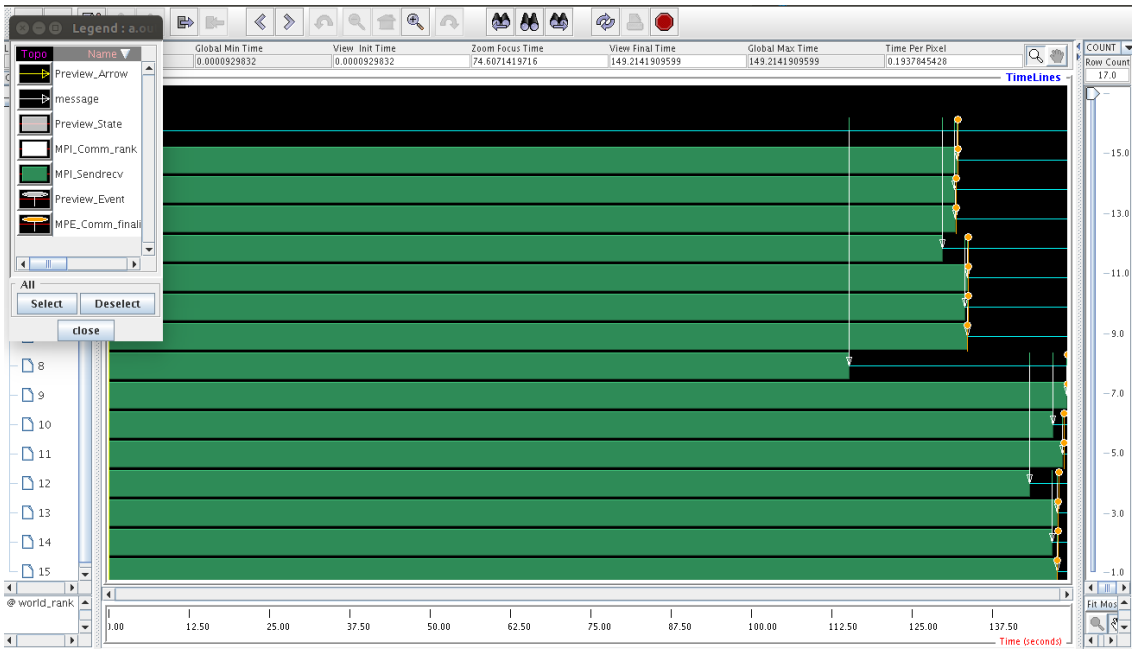
- Used binary tree to distribute work among processes.
- Initial partition done at process '0'.
- Resulting graph, B sent to 1 other process while A is pushed into its own queue.
- The number of processes involved double at every step until all processes have been involved, after which the queue is used to process the graphs.

Total time taken : 152 seconds for 16 processes
140 seconds for 32 processes
Total fill-ins : 10560

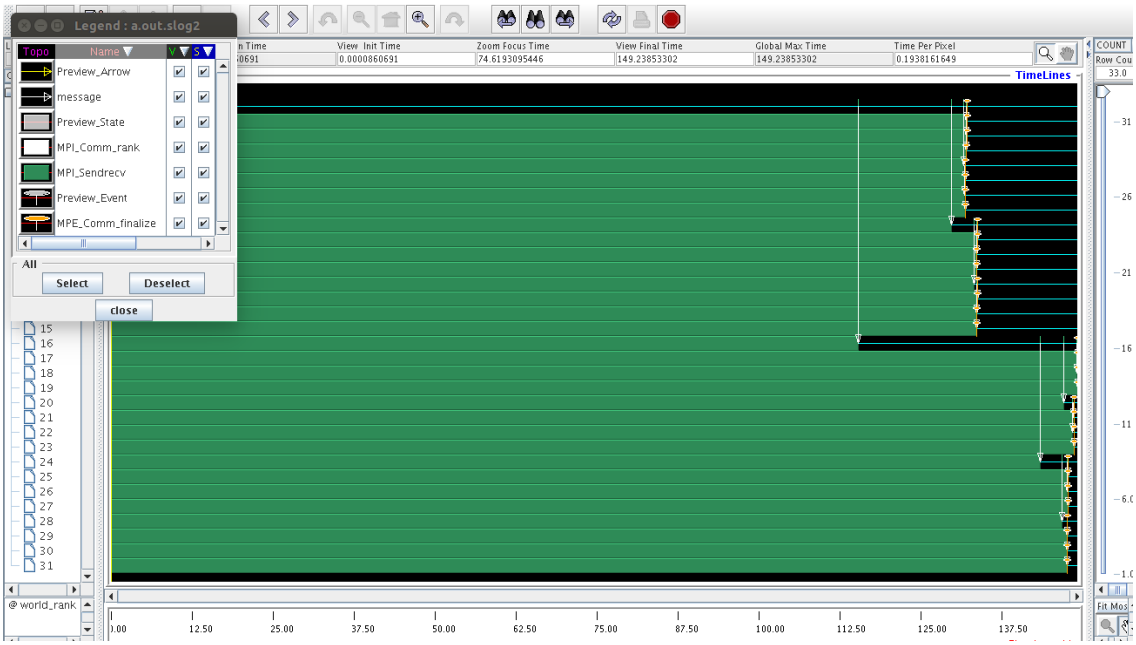
Speedup: for 32 processes: 5.3/140 = 0.04
for 16 processes: 0.035

Profiling:

For 16 processes:



For 32 processes:



As can be seen, process '0' involves in heavy computation in partitioning the initial graph. The partition algorithm involves O(N^2) given N nodes in the graph. However, this operation can be parallelized, by invoking all the free processes that can be identified from the pre-processed binary tree for process distribution. Other effective implementation could be using a hybrid CUDA implementation to parallelize the N^2 gain computations (is a SIMD compuation) involved.
It would be hard to narrow down on the best strategy, in order to reduce the number of fills, as both methods are heuristic. However for the matrix considered nested dissection proved effective in reducing fills. Hence one could sacrifice speedup to gain time in future steps involved in the traingular solve by reducing more fill-ins.

Other improvements:

ND-KL algorithm doesnt provide a heuristic for the final ordering for the A,B graphs resulting after 128 partitions. Hence these were ordered randomly. A mix of ND-KL with minimum ordering can be applied here as the size of the graph has massively reduced.
