LAB-3

1. ALGORITHM:

- The algorithm divides the total input data among the number of processes input from the terminal. The total input size could be 2^26 and the maximum number of processors can be 32.
- Each processor sort its block of elements using quicksort.
- The pivot is chosen as the median of the first process block and broadcast it to all the other processes in the same group.
- Divide each process block in low/high sub-lists if they are lesser/greater than the broadcasted value for each process in the group.
- Exchange the high sub-list for each process in the lower group with the low sub-list for the corresponding process in the high group.
- Divide the group size by 2 and split the group into lower and upper processes. Iteratively call the hyper-sort algorithm for the new groups formed.

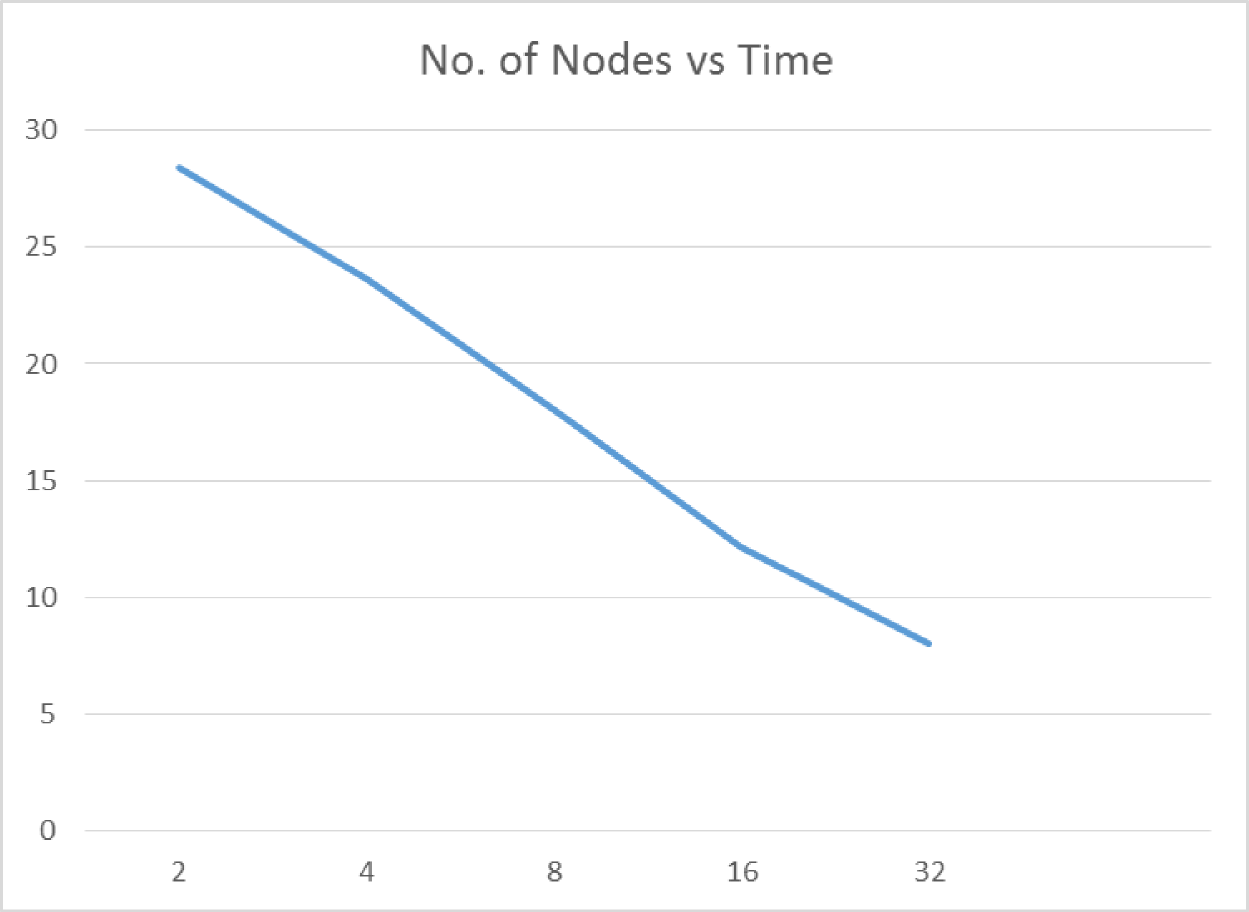
2. Parallelization Strategy:

- All the processes run in parallel and simultaneously.
- For a given group, the first processor provides the pivot for all the other processors in the group.
- After sorting for one phase is complete, for the next phase we split the original group into the two sub-groups by calling MPI_Comm_split. This split is such that processes containing numbers greater than the pivot are in one sub group while the processes containing numbers lesser than the pivot are in another subgroup.
- At the end, all the other processors send their respective sorted lists to the 0th processor, which concatenates this output from the other processors to give the final sorted list.

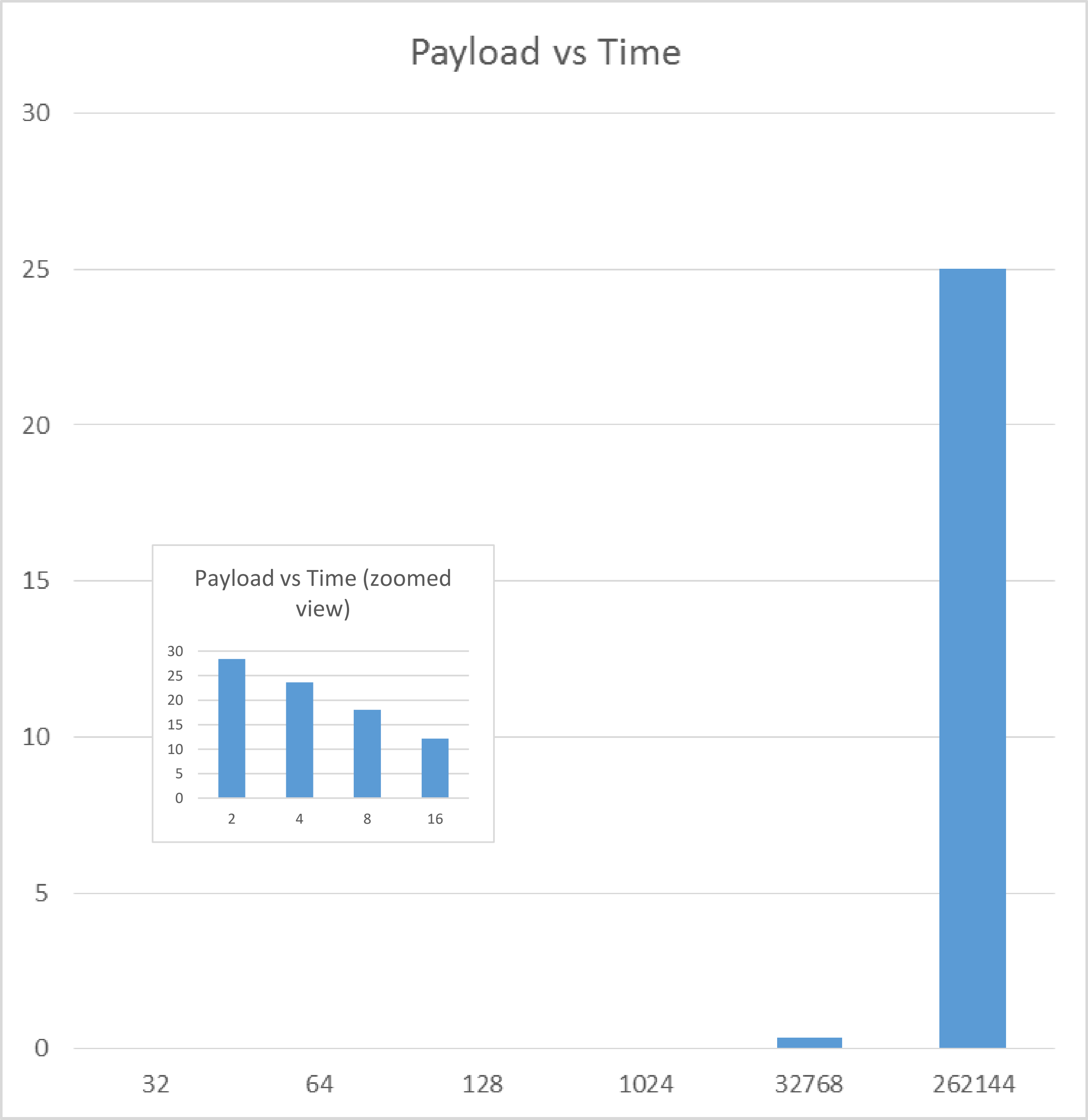
3. Load Balancing Strategy:

- Initially, all the processors load the data and take equal number of elements in their lists according to their process ids.
- For the first iteration, all the processors run on equal number of elements in their list. Processor 0 has an added task of broadcasting it's pivot to all the other processes.
- After first iteration, the lists with corresponding processors is non-uniformly divided. This s because the median in the pivot of the list of only process 1 and not of the list of other processors.
- The iteration steps run for log(P) number of times, where P is the number of processors.
- After the iteration ends, all the processors send their respective sorted lists to processor 0 which receives all of them and combines them to form the final sorted array. Thus, after the while loop, Processor 0 has the more work than the other processors.

Fixed payload of 2^{18}



Fixed no. of nodes of 8


Payload vs Time
Payload vs Time (zoomed view)