1. First MPI implementation (DijkstraSPv1.cpp) – It uses MPI\_Send() to update the overall minimum (J and Dist[J]) by sending the minimum values in each thread to thread 0 which is the coordinator thread, as well as a worker thread. The root thread (0) then uses MPI\_Recv() to receive those values, compare them and store the overall value. Again the root thread uses MPI\_Send() and MPI\_Recv() to transmit the overall min to each thread (there are no shared variables among all threads). Finally the minimum distances are computed and assembled in thread 0 also with the help of MPI\_Send() and Recv().
2. Second MPI implementation (DijkstraSPv2.cpp) – uses collective communications: **broadcast, reduce and gather**, to replace some of the previous send and receive functions. MPI\_Bcast() is used now to update the overallmin (J and Dist[J]). Node 0 sends 2 objects of type MPI INT to each node (including itself). The source of the data will be located at address overallmin at node 0, and the other nodes will receive the data at a location of that name. This call will improve performance over the for clause and MPI\_Send() because MPI is tailored to the platform on which we are running. MPI\_Reduce() means that at this point all nodes in this group participate in a „reduce" operation. The type of reduce operation is MPI MINLOC, which means that the minimum value among the nodes will be computed, and the index attaining that minimum will be recorded as well. Each node contributes a value to be checked, and an associated index, from a location mymin in their programs. The overall min value/index will be computed by combining all of these values at node 0, where they will be placed at a location overallmin. For updating the final **Dist[]** array the partial arrays from each are gathered to thread 0 with the help of MPI\_Gather(). Each node (including Node 0) contributes chunk number of MPI integers, from a location mind+startv in that node's program. Node 0 then receives chunk items sent from each node, stringing everything together in node order and depositing it all at mind in the program running at Node 0.