We implemented Sava using a swim failure detector our simple distributed system from MP3. Machine 1 is our client which executes the swim protocol and by round-robin pinging, asks the master and worker machines to ping back. If failures are detected the client disseminates the updated member list. Our master machine is 2 and our backup master is machine 3. Machines 4 to 10 are our worker machines. The client takes in job commands which we pass in as a string of the class name ‘page rank vertex’ or shortest path vertex’ to run the page rank and shortest path algorithms respectively. The master assigns each worker a number of partitions where each partition contains certain nodes and the also assigns each worker a chunk of the input graph. Each worker parses the input: if it loads a vertex not belonging to it, it sends a message to the owner so that machine can add that vertex to its vertex list. While iteration is less than some super step number or computation halts, meaning all the vertices are inactive and no messages are being, the master call the workers to launch their nodes. Each worker loops through all its active nodes and calls the job’s compute method on each. The page rank and shortest path algorithms were implemented according the code given in the Pregel paper. While calling compute, if the compute call requests that messages be sent, these messages are also asynchronously sent. The worker launches a thread for each partition for each segment of nodes and a barrier waits for all threads to finish. For each super step, the master also has a barrier waiting for all workers to finish. If a master fails, the client automatically joins machines 3 and this machine re-partitions the graph and assignments and starts the workers over. Similarly, with worker failures, the iteration returns to zero. When the computation halts, the workers write to their output files using put. The master then calls get and reduces the worker files into the final result file.