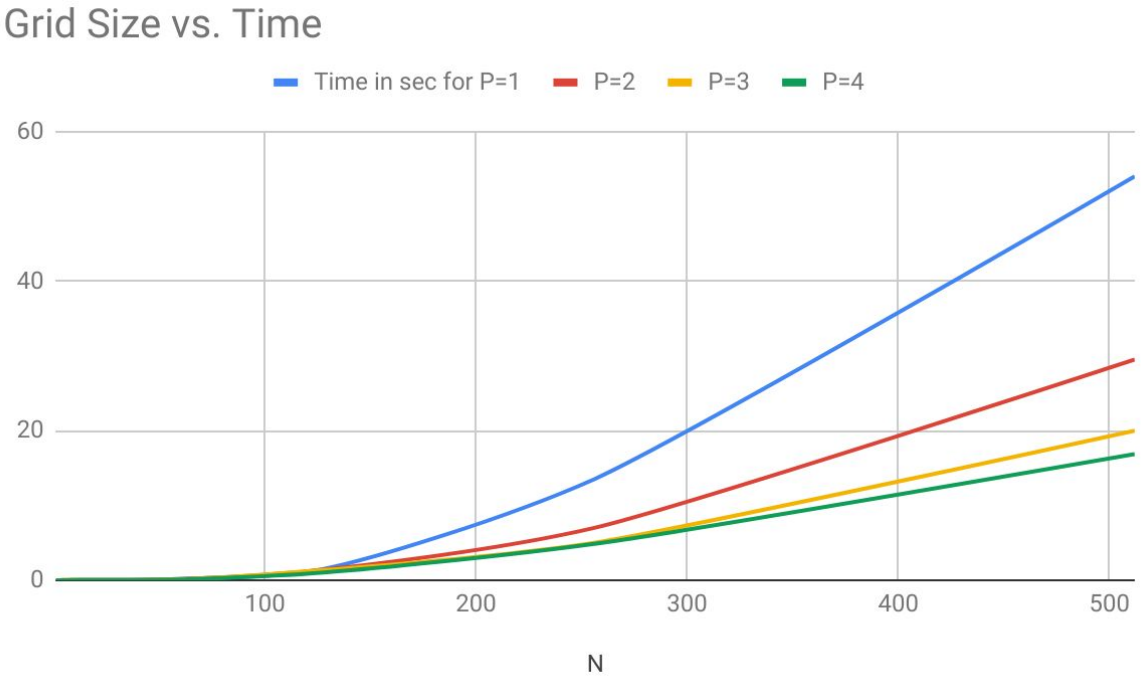
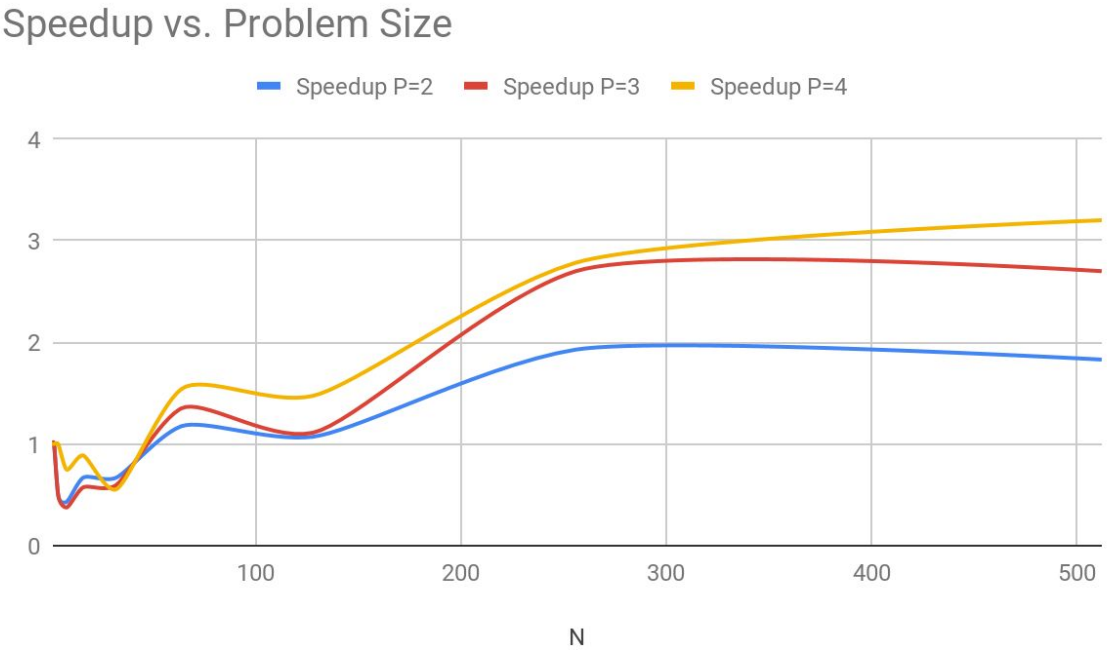
COL331 A2 Report
Shashwat Shivam
2016CS10328

Jacobian Algorithm

Plots for the time taken by programs with different number of processes vs. the grid size of Jacobian computation is given below:-



The corresponding speedups for the results are as follows:-



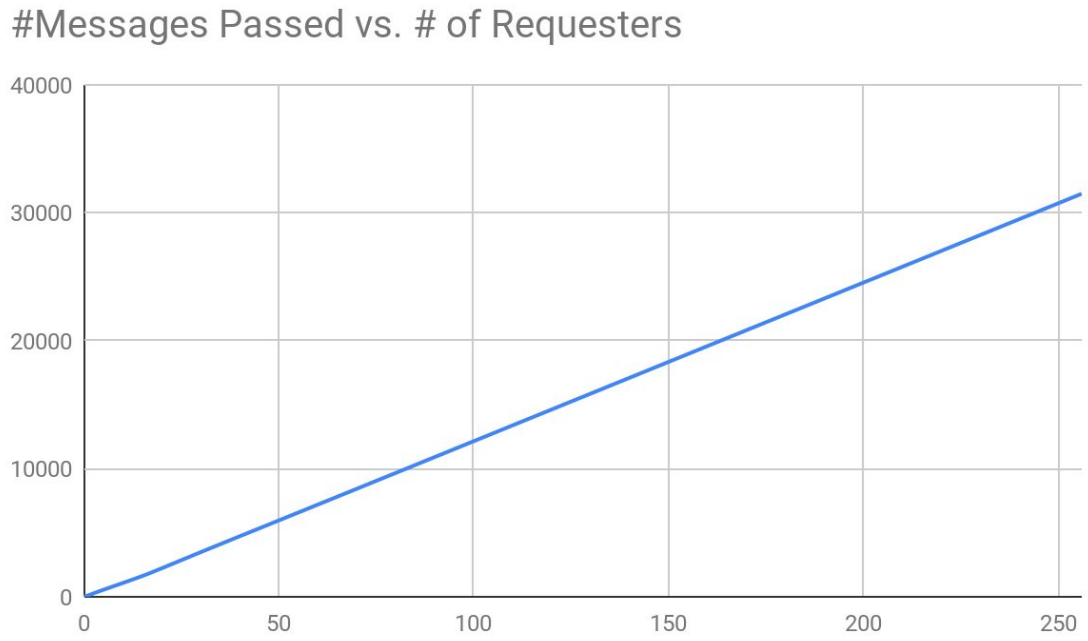
We observe that the time taken for small sizes is similar for the different number of processes. This is due to the fact that the speedup is offset by the overhead of forking threads and synchronization between threads in each iteration. For larger problem sizes we see the speedup when the overhead becomes less when compared to the decrease in computation time. For a larger number of threads on larger problem sizes, we get a better speedup. For P number of threads, we can ideally get up to P times speedup. This is observed in the speedup graph. For P=4 we observe that the speedup does not reach 4 as speedups near 4 will be seen for even larger problem sizes.

For correctness check the result of the sequential algorithm was compared to the parallel version with the same number of iterations. If the two outputs matched then this shows the correctness of the parallel implementation.

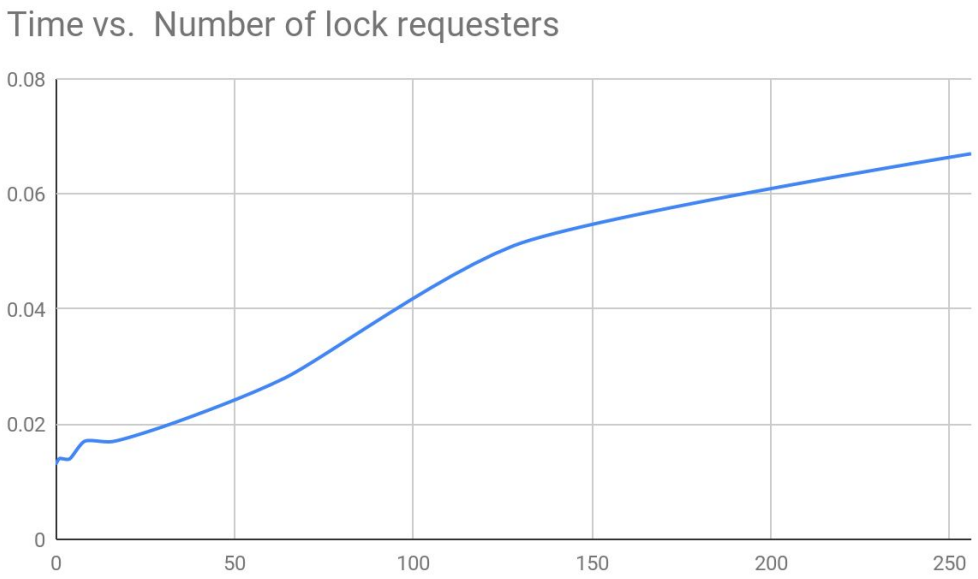
Maekawa Algorithm

All processes have been set to type P3 to simulate the maximum number of messages and activity during running the algorithm. Process type P1 does not request the lock and thus reduces the number of messages in system. Process type P2 sleeps after acquiring the lock thus allowing other nodes to stabilize.

Plot for the number of messages passed with a varying number of requesters for a fixed process number is as follows:-

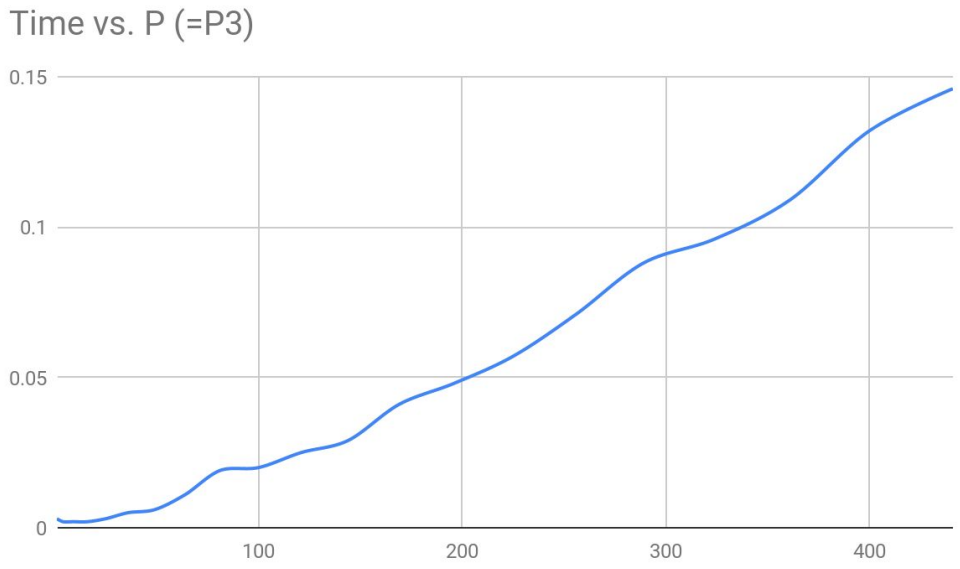


The plot for time vs. the number of lock requesters is as follows:-



We can see that the time does not increase very rapidly even when a large number of processes request the lock (for a fixed P).

The plot below shows the time taken to complete critical sections of all requesting processes in comparison with the total number of processes. (Note:- All processes are of type P3).



We can see a linear increase trend in time to complete all critical sections which is very efficient as if we see per process lock acquiring time, it comes out to be nearly constant. This is due to the fact that for acquiring 1 critical section for the highest priority process, a constant number of messages are needed.

For correctness check, the order of output of lock acquires was checked. This must be in the order of priority of requests and the number of acquires = the number of releases = P2 + P3. Also, a release for the same process should follow it’s acquire.