Problem Set 6 - NBODY

Tomasz Michalik

MPCS 51087

`*Swirl.mp4 file showing modded galaxy collision for simulation`

*link to excerpt of final prod simulation:

https://drive.google.com/open?id=1bfBdaQNxGVdhyiYrq5K8gScpf5nqK8xk

———————————————————————————————————————-

Sections

File output/Correctness:

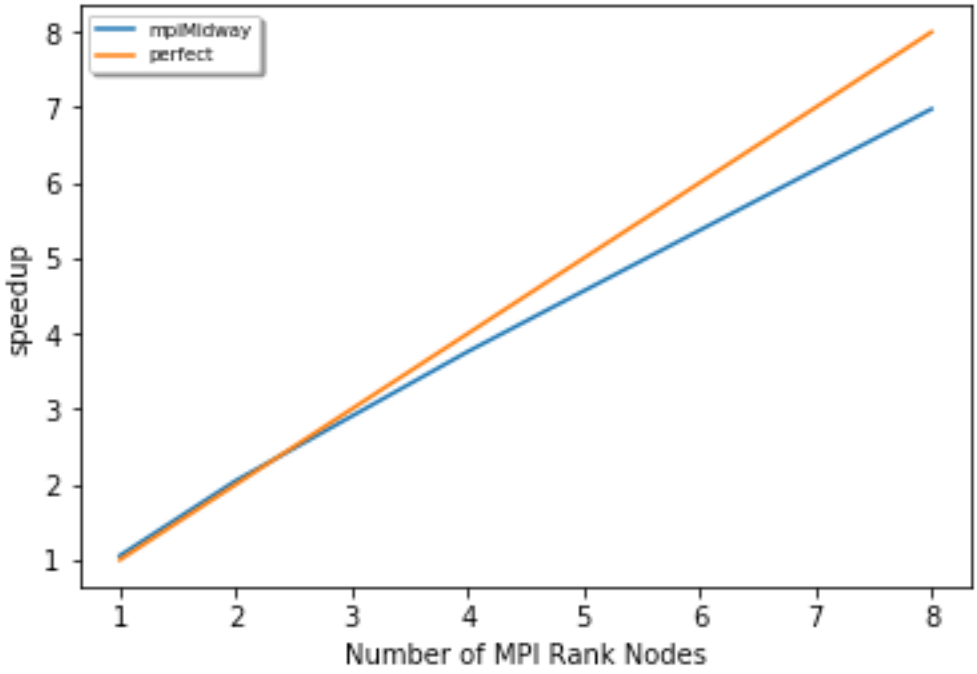
The animations shown in my parallel vs serial and openMP only were very similar. The diff function did generate many differences though i assume this is part of the randomness and the way i adjusted my particles' positions and velocities. The animations are similar and noted in README (SerCorrect vs ParCorrect mp4s, data with same name). Due to my implementation of master handing each rank a piece of data after randomizing it may have had an effect on precision.

Note : i submitted both hybrid and MPI solutions, but due to issues did not run my openMP included code as to not waste hours on machines. I tried few different techniques but it seemed to segfault at a certain point when running with cpu threads. The following is for MPI Full

***Code located in zip file - see README for exact file names

Midway Testing:

Time	179.1 sec	92.2 sec	50.25 sec	27.08 sec
Nodes	1	2	4	8

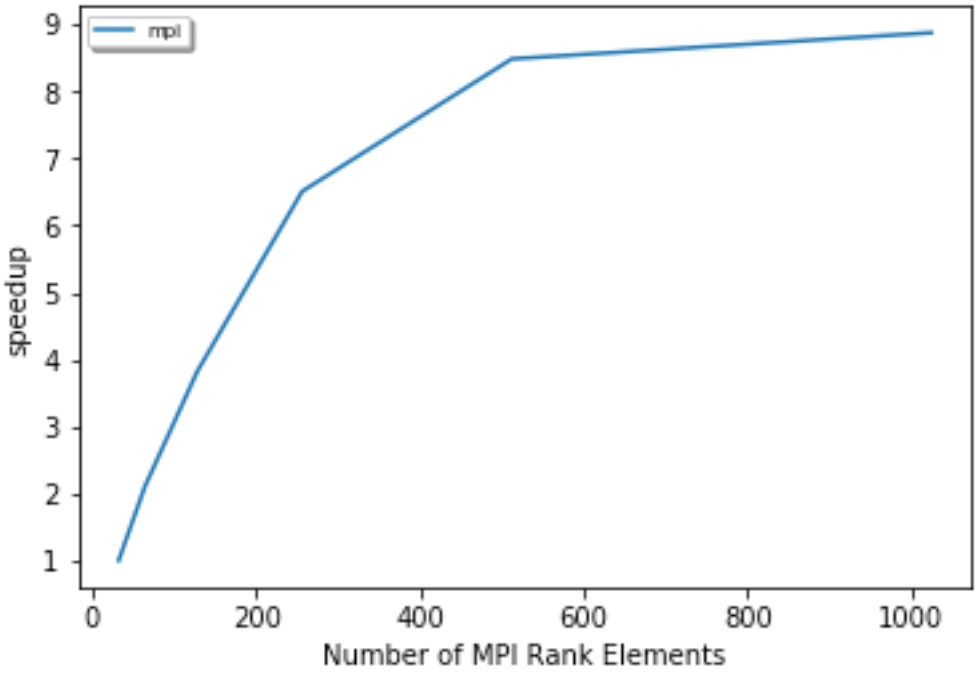


The problem seems to scale pretty well. My runtimes for Full MPI should be fairly accurate. I was able to run OpenMP without MPI, but not compare both.

As seen on the graph below, we experience an almost linear improvement in performance up to 8 NODES (not ranks). All nodes are 16 mpi ranks. So we maintain scalability by now adding nodes vs ranks. If we were scaling by ranks this would also trail off in speedup as we have seen in previous assignments.

BGQ Testing:

Printing: When removing the distributed print routine, the process only took .5 seconds less and around 1% of total time. I am not sure if this is a result of my implementation or if other barriers caused it to be less performant than usual when removing the burden of File IO, but i think in this example the distributed file IO is quite efficient when taking into account the frequency of its use compared to other problems.



*Performance Test: Running full MPI resulted in 29.64 sec. Slightly higher than Midway but when we take into account the perf/watt, the Midway cluster uses more than 4x as much power for a 9% increase in runtime.BGQ has a much higher interactions per second at around double the interactions per second at 7.4 *10^9 vs 3.4 * 10^9. When adding that 5 time more wattage, its almost 10x as performance (or efficient).*

*Strong Scaling: MPI FULL**

Time	789.4	375.3	206.5	121.3	93.04	88.95
Nodes	32	64	128	256	512	1024

Summary: I am not sure if these are higher than expected run times, since i am 5x the problem, but i am not using both threads/ranks. I am running on full MPI for the test with -c64 flag for using all ranks with 1 thread. The performance scales pretty well until 1024 nodes where it hits a wall in terms of perf/node added. This may be the limit of what the problem should be run on for maximum efficiency or reached a problem size limit to the efficiency of increasing # of processors.

Due to not being able to use hybrid version, this may have been faster had i been able to run this version

Summary: Showing linear performance was inefficient as could not see the speedup. Getting near 10x performance boost as more nodes are added. This shows how we can

continue to scale up the problem with both hardware and increasing the number of nodes on a larger scale machine.