HOME WORK 3 – Distributed Memory N-body with MPI

November 30, 2017

Introduction

This report is a basic summary of the analysis of N-body algorithm using OPenMPI for parallelization.

Code Modification

Please click link to go to codes repo

Benchmark Analysis

Table Key

PARTICLES: Number of Particles
PROCESSES: Number of Processes
TIME: Runtime in microseconds
SPEEDUP: Sp = Ts / Tp
Where;
Sp = Speed-up, Ts = Execution time in series Tp = Execution time in Parallel with P cores

EFFICIENCY: Ep = Ts / pT
Where; Ep = Efficiency, p = Number of Threads

Strong Scalability

Strong Scaling: Serial Benchmark

PARTICLES	TIME
1000	25.433723
2000	100.268770
4000	400.147935
8000	1604.038234
16000	6405.5837761

Strong Scaling: Parallel Benchmark for 1000 Particles

PROCESSES	TIME	SPEEDUP	EFFICIENCY
2	13.2275	1.9228	0.9614
4	6.693263	3.7999	0.9500
8	3.358169	7.5737	1.0563
16	1.724769	14.746	0.9216
32	0.921790	27.592	0.8622
64	0.545337	46.639	0.7287
128	0.476963	53.324	0.4166
256	0.421122	60.395	0.2359

Strong Scaling: Parallel Benchmark for 2000 Particles

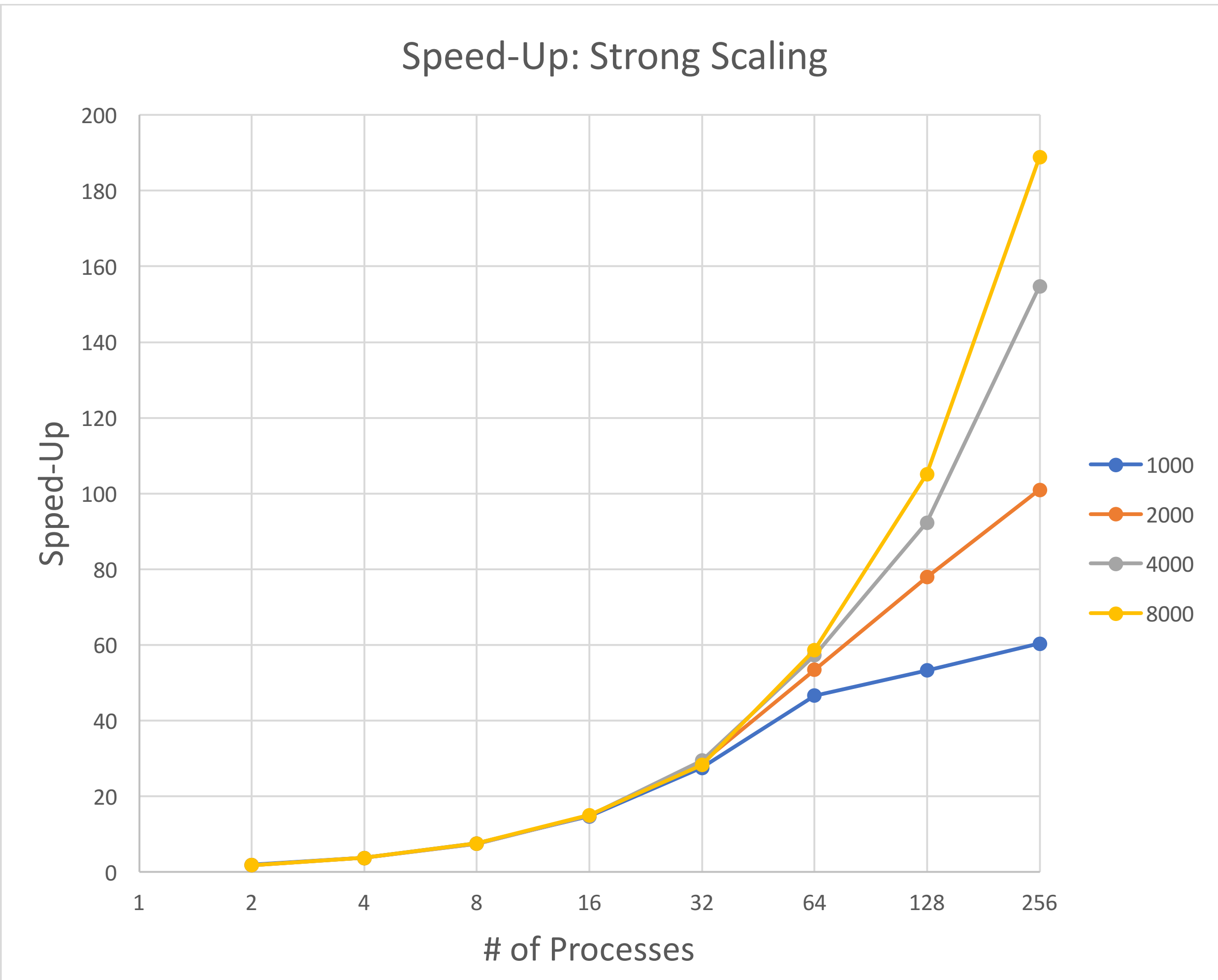
PROCESSES	TIME	SPEEDUP	EFFICIENCY
2	53.061749	1.8897	0.9448
4	26.636245	3.7644	1.0626
8	13.357500	7.5066	0.9383
16	6.729174	14.901	0.9313
32	3.468578	28.908	0.9034
64	1.874830	53.482	0.8339
128	1.286498	77.939	0.6089
256	0.992870	100.99	0.3945

Strong Scaling: Parallel Benchmark for 4000 Particles

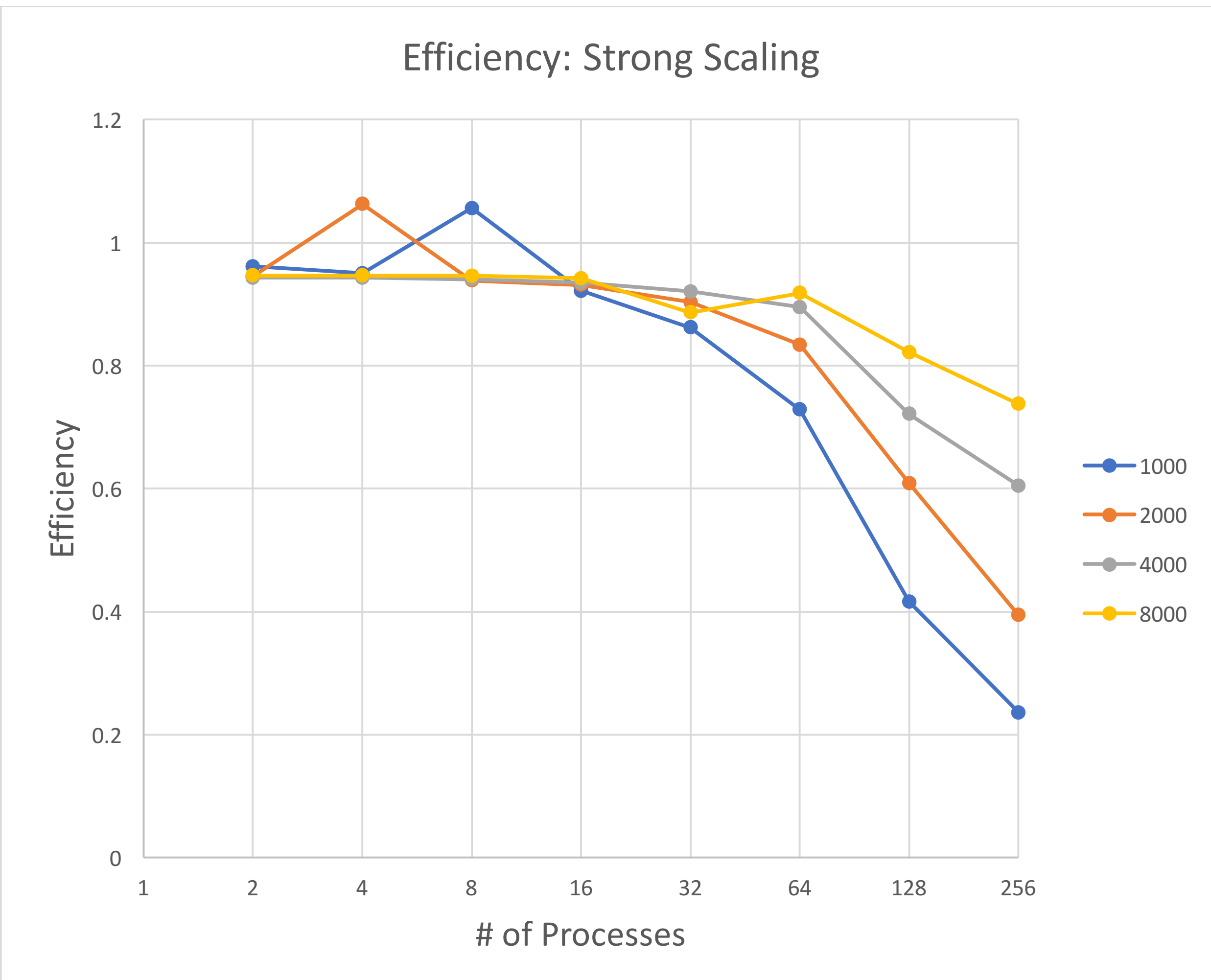
PROCESSES	TIME	SPEEDUP	EFFICIENCY
2	212.146425	1.8862	0.9431
4	106.073760	3.7724	0.9431
8	53.191162	7.5228	0.9404
16	26.747779	14.960	0.9350
32	13.583481	29.458	0.9206
64	6.986296	57.276	0.8949
128	4.334127	92.325	0.7212
256	2.584722	154.81	0.6047

Strong scaling: Parallel Benchmark for 8000 Particles

PROCESSES	TIME	SPEEDUP	EFFICIENCY
2	847.122028	1.8935	0.9468
4	423.612602	3.7866	0.9466
8	211.945913	7.5681	0.9460
16	106.412916	15.074	0.9421
32	56.556234	28.362	0.8863
64	27.303180	58.749	0.9180
128	15.250064	105.18	0.8217
256	8.490988	188.91	0.7379



In the graph above, for 1000 particles, we see an inclination from 2 processes through to just before 64 processes and a linear Speed-up from 64 processes to 256. We also see the same type of scaling for 2000, 4000, and 8000 particles as the Speed-up inclines from 2 processes through to 32, just before 128, and 128 processes respectively and a further Linear Speed-up to 256. We can infer that an increase in the number of process does not readily lead to a decline in the Speed-up even though the amount of work is reduced. Although the Speed-up is not continuously Super Linear all through, we still get a reasonable performance boost by increasing the number of processes.



From the graph above, we can see that for 1000 particles we see a Speed-up of about 95 percent using 2 processes and a very slight decline. Increasing the number of processes to 4 gives use a Linear performance boost and a further increment in the number of processes causes an inverse performance, hence we see an almost perfect Linear decline. A further increment in the number of processes leads to a decline in performance. For 2000 particles, we notice about the same phenomenon with a Linear Speed-up at 2 processes from about 95 percent to about 106 percent. Increasing the number of the processes causes an almost Linear decline. A further increment in the number of processes causes a further decline. For Large enough work, like 4000 and 8000 particles, we don't see an increase, or decrease in performance with an increase in the number of processes until about 32 processes for 4000 particles and 16 processes for 8000 particles. Notice that at a further increment in the number of particles from 16 to 32 in 8000 particles gives a Linear Speed-up, until a further increment to 64 processes causes a reverse in performance, hence the decline.

Weak Scalability: Parallel Benchmark

PARTICLES	PROCESSES	TIME	EFFICIENCY
200	1	1.067476	1
283	2	1.107529	0.96384
400	4	1.106645	0.96461
566	8	1.123555	0.95009
800	16	1.117051	0.95562
1131	32	1.164305	0.91684
1600	64	1.219269	0.87550
2263	128	1.594581	0.66944
3200	256	1.920175	0.55593

Weak Scaling: Parallel Benchmark

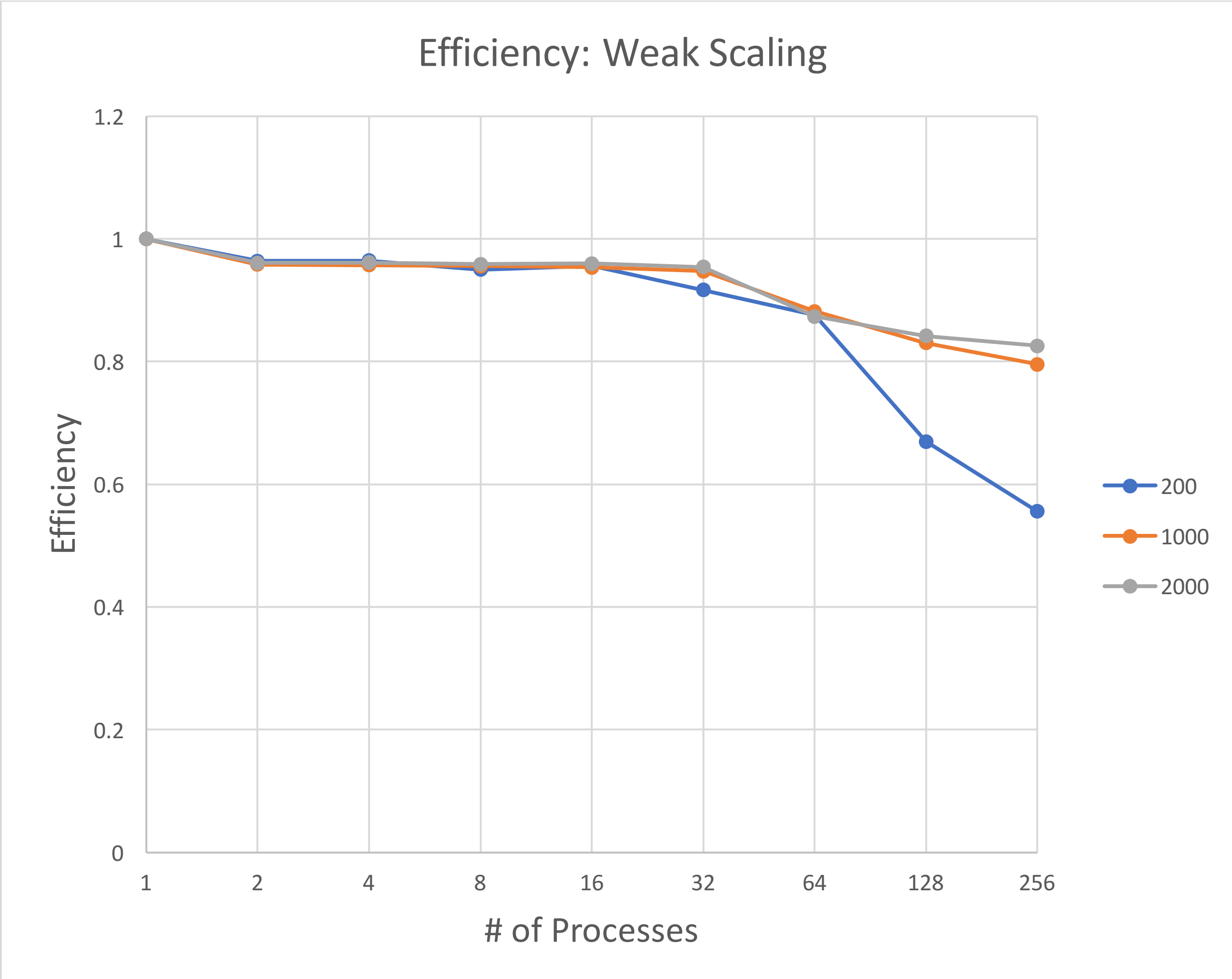
PARTICLES	PROCESSES	TIME	EFFICIENCY
1000	1	25.495007	1
1414	2	26.611413	0.95805
2000	4	26.640297	0.95701
2828	8	26.680346	0.95557
4000	16	26.729546	0.95381
5657	32	26.914289	0.94740
8000	64	28.910461	0.88186
11314	128	30.709361	0.83020
16000	256	32.048332	0.79552

Weak Scaling: Parallel Benchmark

PARTICLES	PROCESSES	TIME	EFFICIENCY
2000	1	102.036297	1
2828	2	106.170586	0.96106
4000	4	106.133483	0.96140
5657	8	106.399173	0.95900
8000	16	106.291287	0.96000
11314	32	106.933127	0.95421
16000	64	116.802235	0.87358
22627	128	121.197669	0.84190
31999	256	123.561676	0.82579

Weak Scaling: Parallel Benchmark

PROCESSES	PARTICLES=200	PARTICLES=1000	PARTICLES=2000
1	1.032528	25.975124	100.26877
2	0.560662	13.2275	53.061749
4	0.317195	6.693263	26.636245
8	0.169107	3.358169	13.357500
16	0.113186	1.724769	6.729174
32	0.093042	0.921790	3.468578
64	0.113931	0.545337	1.874830
128	0.151649	0.476963	1.286498
256	0.175286	0.421122	0.992870



The graph above is an illustration of the combined Weak Scaling Efficiency values of 200, 1000, and 2000 particles when parallelized with 1, 2, 4, 8, 16, 32, 64, 128, and 256 processes respectively.
As seen in the graph above, for all amount of work starting from 200, 1000 and 2000 particles respectively, Efficiency declines. A slight performance boost is gotten from increasing the number of processes in all cases and we observe a relative flat line (plato) through 4 to 8 and 16 processes. A further increment in the number of processes for a small amount of work like 200 particles causes the decline in performance, signifying there is not enough work to be done in parallel and the processes and spending too much time communicating than doing actual computing. The same is seen for larger amounts of work like 1000 and 2000 particles beginning at 32 processes through 256, shows a decline in performance with a relative increase in the number of processes

Conclusion

MPI scalability is relatively better than OpenMP scalability especially in the cases Speed-up and Efficiency for Strong Scaling. However, OpenMP seems to scale better in terms of Weak Scaling as it relates to Efficiency.

Challenges Faced

1. Finding the global(net) results across all ranks by implementing MPI Reduce
2. Implementing MPI_Allgatherv
   Resolution:
   I. I emailed Dr. Stone for assistance and he walked me through it.