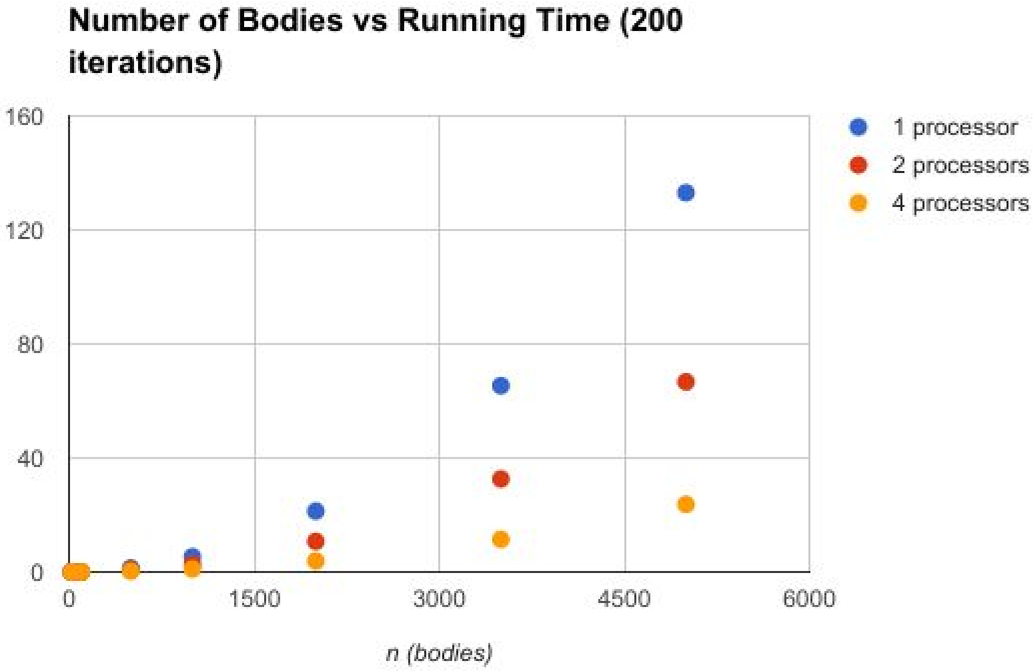
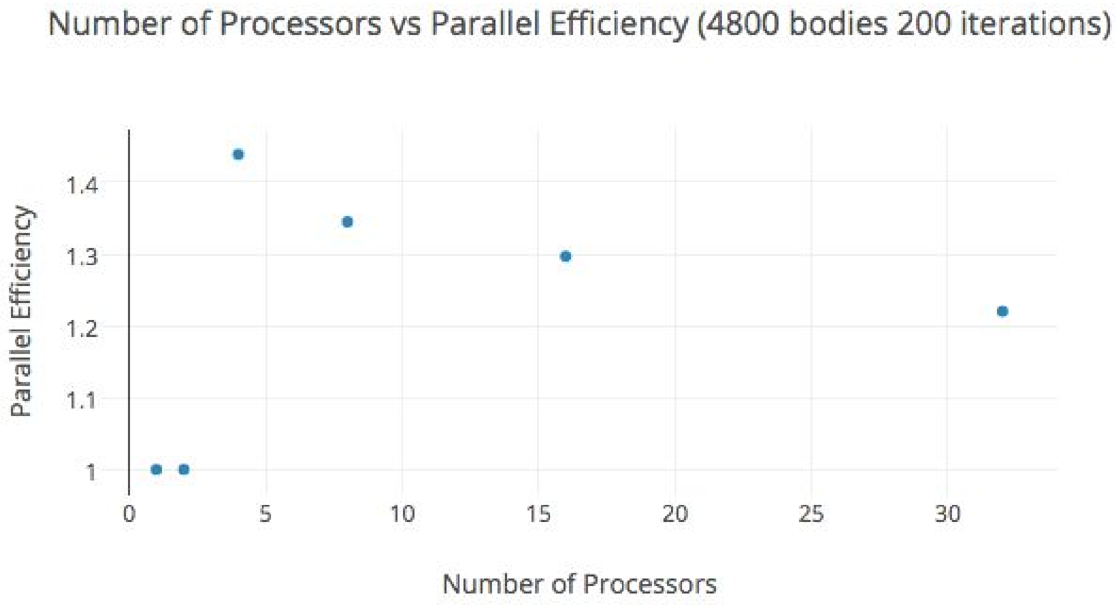
Simon Huynh
Marco Simone



n, iter = 200	1 processor	2 processors	4 processors
20	0.003042	0.001776	0.002533
100	0.057762	0.028018	0.013194
500	1.355426	0.675615	0.288801
1000	5.342938	2.69372	1.012264
2000	21.358761	10.735535	3.847031
3500	65.36173	32.654009	11.436646
5000	133.068096	66.707405	23.742365

Simon Huynh
Marco Simone



Number of Processors	Time (seconds)	Parallel efficiency
1	122.69284	1
2	61.337631	1.000143289
4	21.328204	1.438152505
8	11.407007	1.344489839
16	5.914354	1.296557917
32	3.142793	1.219982115

How to compile: Use default makefile.

Conclusion:

As expected the scaling of the time for the different number of processors followed a 1/p relationship. Every time we doubled the amount of processors the time it takes to complete the calculation roughly halves. Because of this we know that the calculation takes up the majority of the time and that the communication is negligible for large values of n. We were surprised to see that when we doubled the amount of processors from 1 to 2, and 2 to 4 over different values of n, the time was less than half. So, the scaling was better than expected.