Anderson Nguyen
EECS 120 HW2

This is for image size **5000 x 5000** for all these test cases. It was hard to test other cases due to the fact that we needed to share the resources with other students.

Susie

Cores	Ts	Tp	Speedup
1	13.4506	13.87560	0.96937
2	13.4506	7.319775	1.83757
4	13.4506	4.531293	2.96838
8	13.4506	3.957009	3.399183
16	13.4506	3.888333	3.45922
32	13.4506	3.766947	3.57069
64	13.4506	3.636496	3.69878

Joe

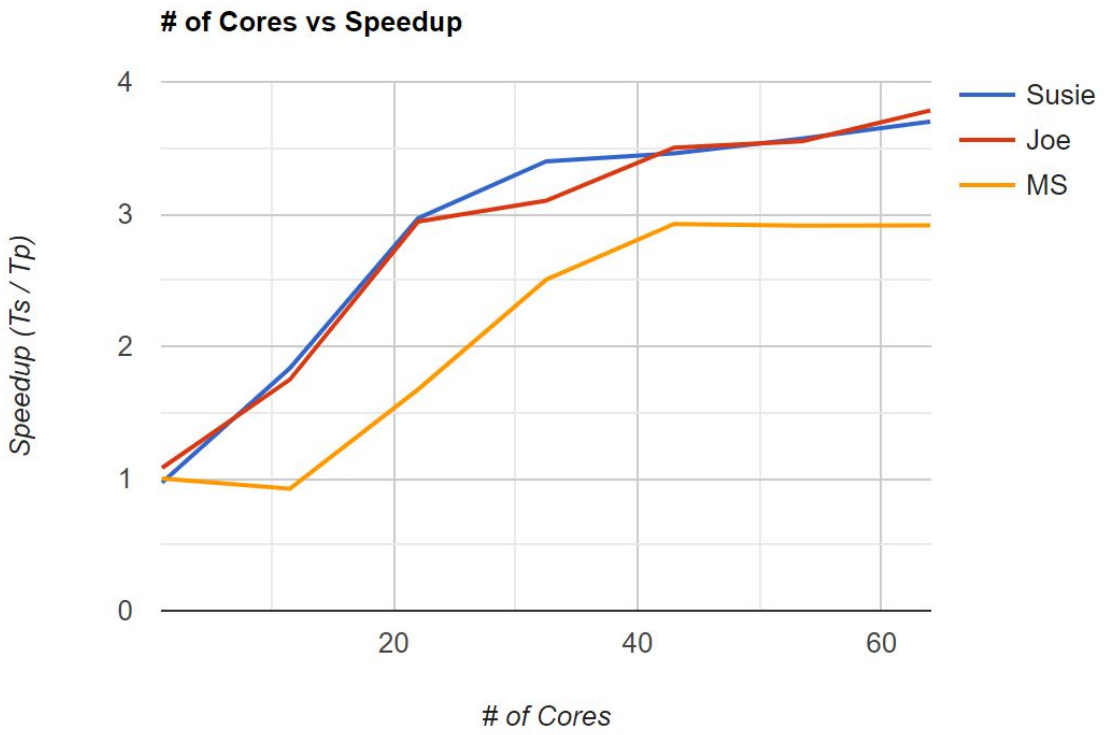
Cores	Ts	Tp	Speedup
1	13.4506	12.44245	1.081025
2	13.4506	7.689203	1.749284
4	13.4506	4.569561	2.943521
8	13.4506	4.337020	3.101346
16	13.4506	3.839813	3.502931
32	13.4506	3.787465	3.551346
64	13.4506	3.554673	3.783920

Master/Slave

Cores	Ts	Tp	Speedup
1	13.4506	13.3506	1
2	13.4506	14.90678	0.902314
4	13.4506	8.0353228	1.673934
8	13.4506	5.371778	2.503938

16	13.4506	4.598729	2.924851
32	13.4506	4.619107	2.911948
64	13.4506	4.622199	2.913734

Plot



Load balancing strategies

Which do you think is better? Why? Which intern do you offer a full-time job?

From my testing, I think both interns algorithms are pretty close in terms of speed. I would choose Suzie because she was faster in most of the test cases. Joe's implementation divides up the rows into blocks that aren't necessarily divided equally whereas Susie's is more evenly divided, which in the long term, will produce better results. The company should hire Susie based on the numbers but either interns are a good pick.

Compare the master/slave strategy with Susie/Joe's implementation. Which do you think will scale to very large image sizes? Why?

The master/slave strategy should be faster than the intern's implementation if it scales to a larger image size because the slave processors will keep working until the job is done. Master/slave avoids the equal division of work and will just keep working until a slave requests. This handles load balancing well because it would prevent one node from being completely overworked in the case that there requires high computation or an infinite loop.