Wesley Chen

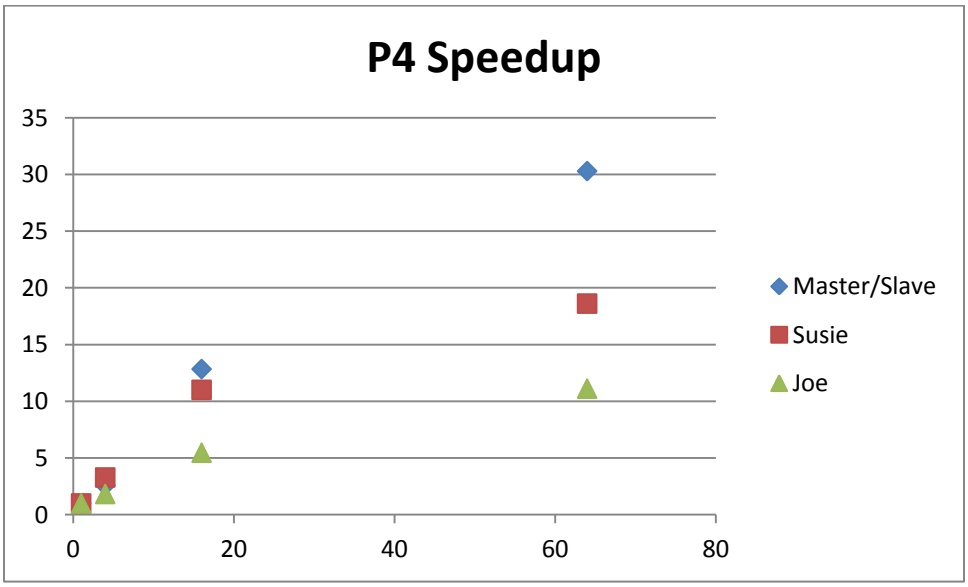
CS205 HW 2 P4

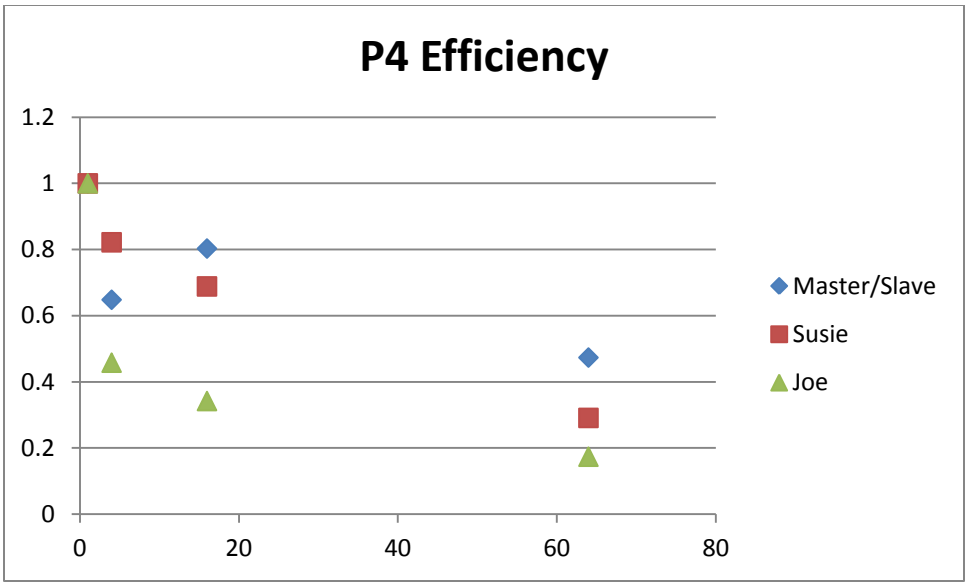
The way Intern Susie has it split up is that each process p does a striping across the entire final image. This means that ever p^{th} row will belong to processor p. Intern Joe breaks up the image into p chunks and gives each chunk to a specific processor. For this case, Susie should be promoted because the work is not even throughout each chunk of the image. As seen by the time it takes to run the serial version as well as the final image output – the first rows have very few calculations (are are black) and most of the "details" come from the middle rows (which can also be observed to take the longest with the serial version). Susie's striping method will give each processor a more balanced amount of work.

As we will see in the table below, we see that Master-Slave load balancing is generally the fastest and most efficient past a certain number of processors as it gives everyone immediately something to do. Ideally, we could make it even better, by starting each processor off with the most expensive rows, which are the middle ones in our case, but in a specific problem, we may not know this heuristic.

I noticed while running that some of the times had a bit of variation to them – I tried running multiple times to take averages. Also of note is that the Master/Slave is not very efficient relative to Susie's method at 4 cores which surprised me but could be due to some overhead in the implementation as well as the fact that Susie's implementation is not too bad for the Mandelbrot setup. Else, everything is as expected, Joe's is the worst due to the worst load balancing – with a 0.17 efficiency with 64 cores – which makes sense as some rows just take much longer to compute and not time can be saved in this way. Then comes Susie, and finally the best being Master/Slave.

Below is the tabulated data and plots of the speedup and efficiency for 1,4,16,64 cores:





Data:

Master/Slave	1	4	16	64
Time	47.10307	18.16818	3.666377	1.554377
Speedup	1	2.592614	12.84731	30.30351
Efficiency	1	0.648154	0.802957	0.473492
Susie	1	4	16	64
Time	47.10307	14.3288	4.276041	2.529631
Speedup	1	3.287302	11.01558	18.62053
Efficiency	1	0.821825	0.688474	0.290946
Joe	1	4	16	64
Time	47.10307	25.71284	8.607227	4.23464
Speedup	1	1.831889	5.472503	11.12328
Efficiency	1	0.457972	0.342031	0.173801