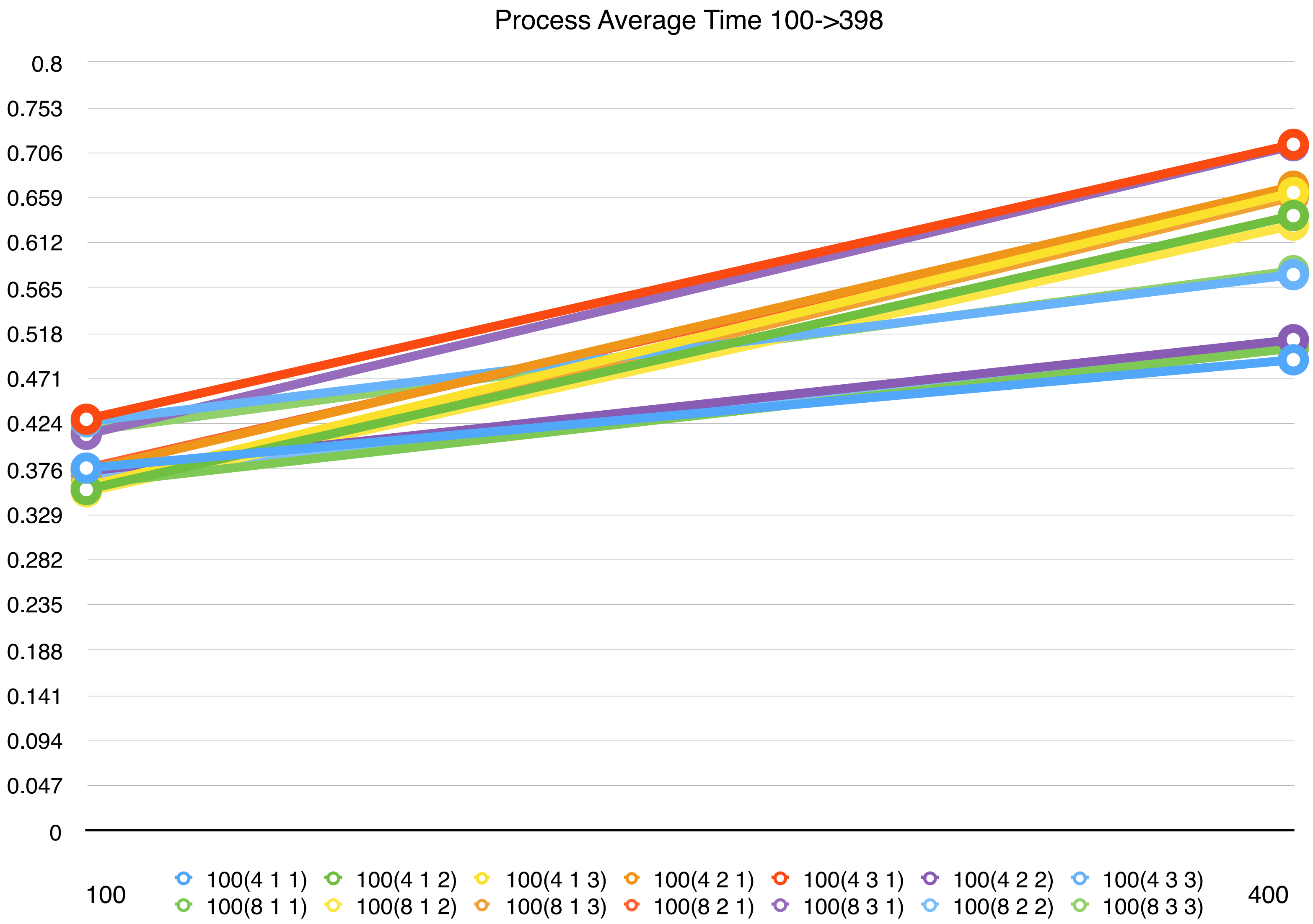
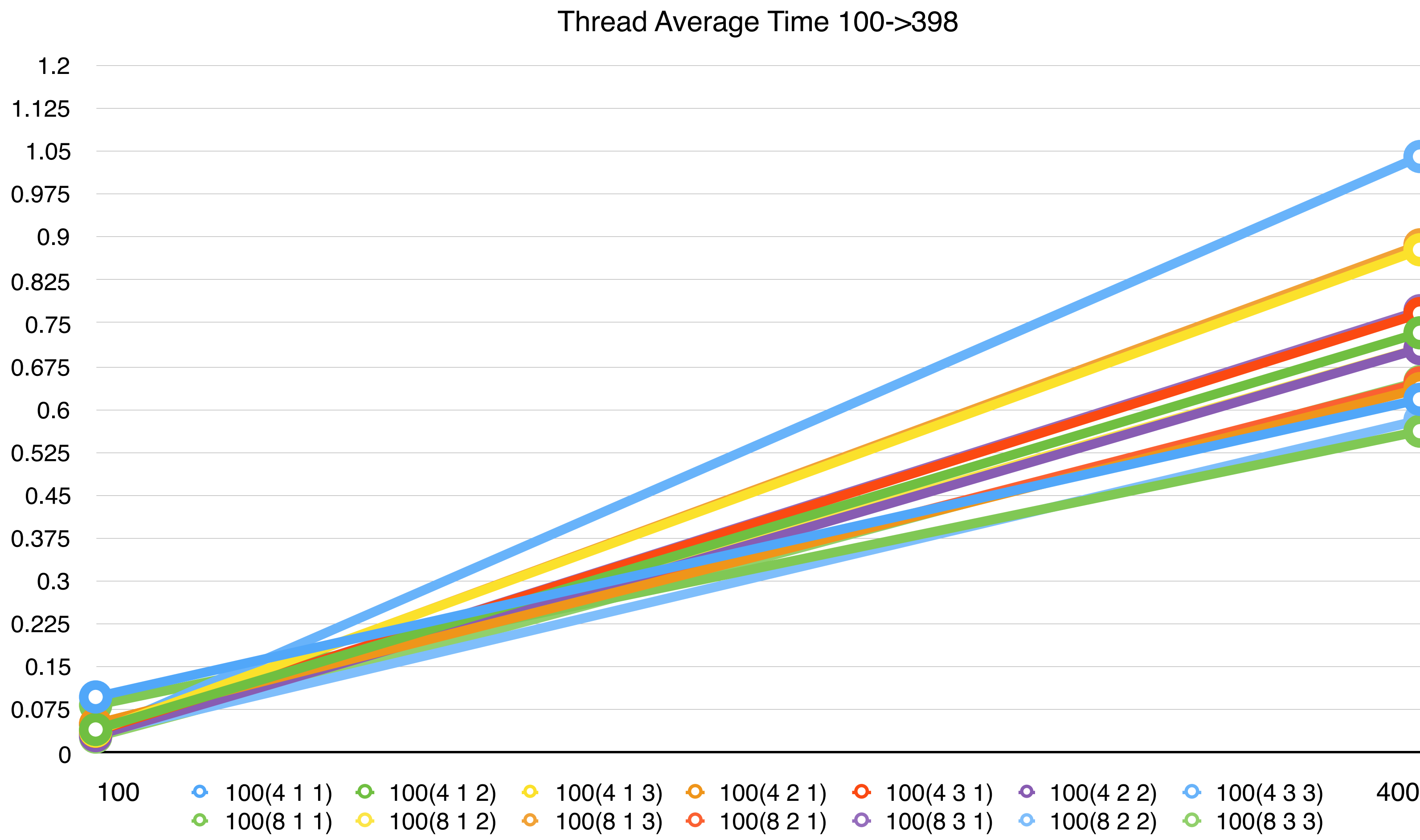
ECE 254 Lab3 Report

Average Timing Measurement Data Chart:

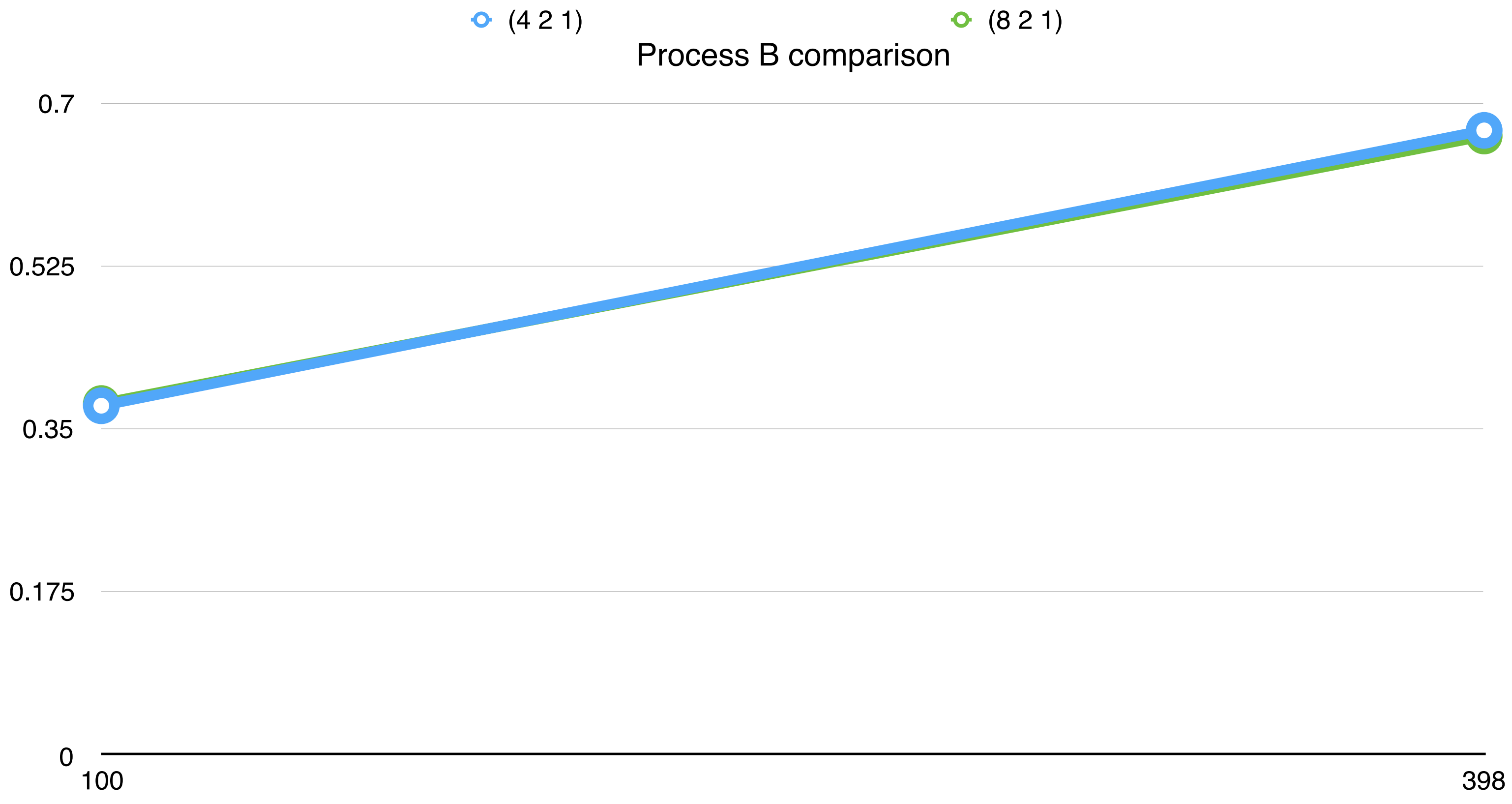
N	B	P	C	Process Average Time (	Process Standard Devia	Thread Average Time (n	Thread Standard Deviat
100(4 1 1)	4	1	1	0.377674	0.0827876544178	0.423778	0.0975475920564
100(4 1 2)	4	1	2	0.355288	0.0391477337275	0.394294	0.0407450557
100(4 1 3)	4	1	3	0.358684	0.022535397578	0.416228	0.0378994461173
100(4 2 1)	4	2	1	0.375188	0.0249052736584	0.35616	0.0505628163773
100(4 3 1)	4	3	1	0.428456	0.0335031948327	0.378618	0.0375599264643
100(4 2 2)	4	2	2	0.374472	0.0355708478392	0.373588	0.0300099692769
100(4 3 3)	4	3	3	0.425764	0.0423322135495	0.461652	0.0344991434097
100(8 1 1)	8	1	1	0.360674	0.076968927003	0.402752	0.0832338782948
100(8 1 2)	8	1	2	0.352924	0.0310793536612	0.380072	0.0403258083118
100(8 1 3)	8	1	3	0.353798	0.0232071798373	0.406036	0.0354033713649
100(8 2 1)	8	2	1	0.377536	0.0305902713947	0.345716	0.0501627286339
100(8 3 1)	8	3	1	0.412714	0.0270131857433	0.359926	0.0367735302086
100(8 2 2)	8	2	2	0.369028	0.0338984839779	0.340264	0.033062521138
100(8 3 3)	8	3	3	0.416014	0.0377190376866	0.360208	0.0274089170892
398(4 1 1)	4	1	1	0.490314	0.0574165777803	0.617146	0.0588397542823
398(4 1 2)	4	1	2	0.640352	0.0408476204448	0.73334	0.059361876655
398(4 1 3)	4	1	3	0.664434	0.0355953598661	0.878302	0.0607302132056
398(4 2 1)	4	2	1	0.670948	0.0293362113437	0.635934	0.0600311722691
398(4 3 1)	4	3	1	0.714526	0.0339272357259	0.766924	0.0506888372721
398(4 2 2)	4	2	2	0.5112	0.0374053472113	0.705746	0.0419602369393
398(4 3 3)	4	3	3	0.57894	0.0538750443155	1.041072	0.051024492315
398(8 1 1)	8	1	1	0.502434	0.0603717288472	0.561592	0.0692556534588
398(8 1 2)	8	1	2	0.630318	0.031246645836	0.706216	0.0730687165892
398(8 1 3)	8	1	3	0.65973	0.0361444477064	0.886946	0.0666802750744
398(8 2 1)	8	2	1	0.66491	0.0247490181623	0.645968	0.0507749049827
398(8 3 1)	8	3	1	0.713002	0.0282554418829	0.771912	0.0584617503672
398(8 2 2)	8	2	2	0.511198	0.0332667220507	0.581012	0.0370006466971
398(8 3 3)	8	3	3	0.583066	0.0551574622694	0.648316	0.044562227772



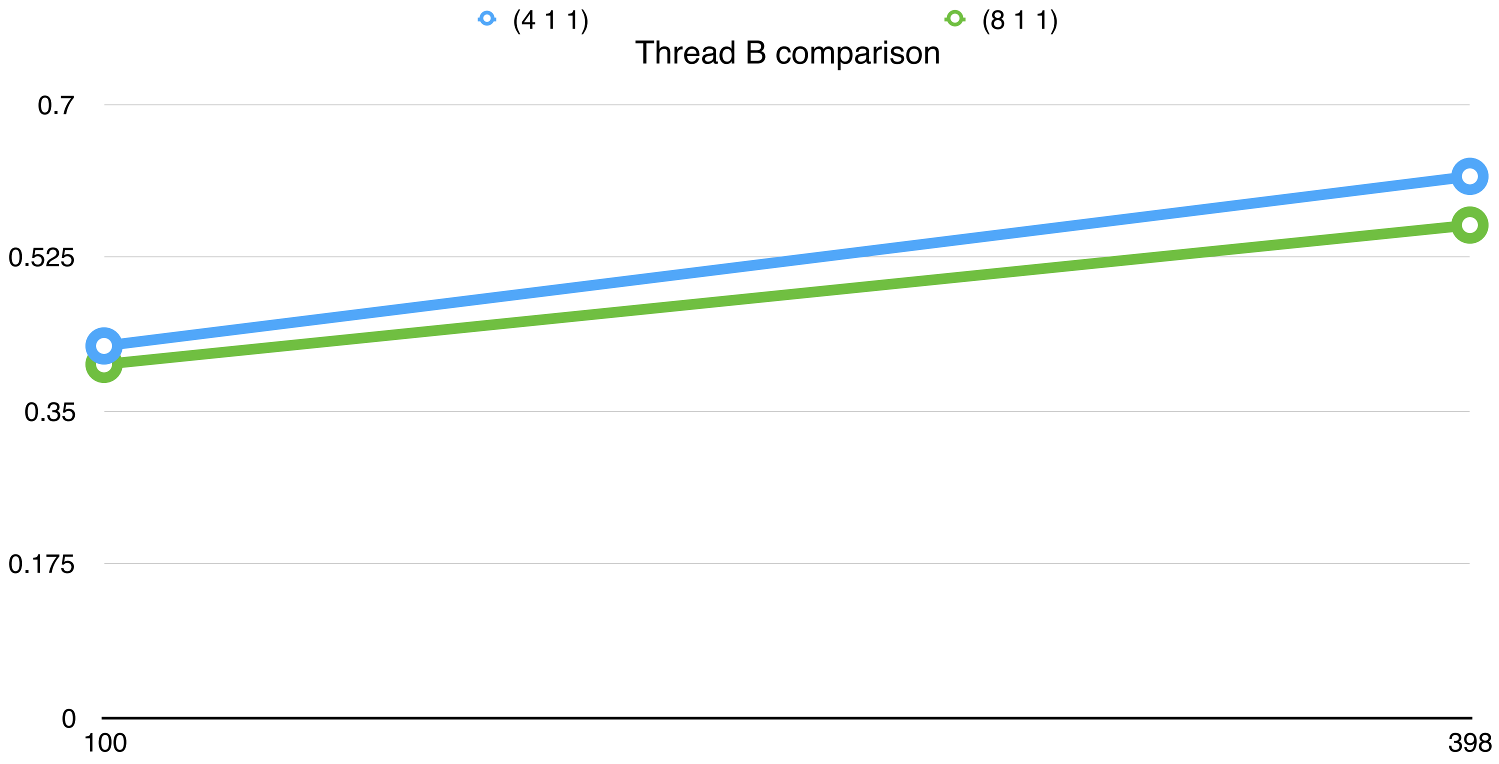
Generally from the graph we can see the average run time of processing 100 numbers will be less than the runtime of 400 numbers. This is because the number of tasks increase, processor takes more time to produce and consume. Also if N=0, the run time will be 0. So every line in this graph will touch (0,0). We can see the slope of some lines are clearly higher than others, the line of 100(4 3 1) for instance.



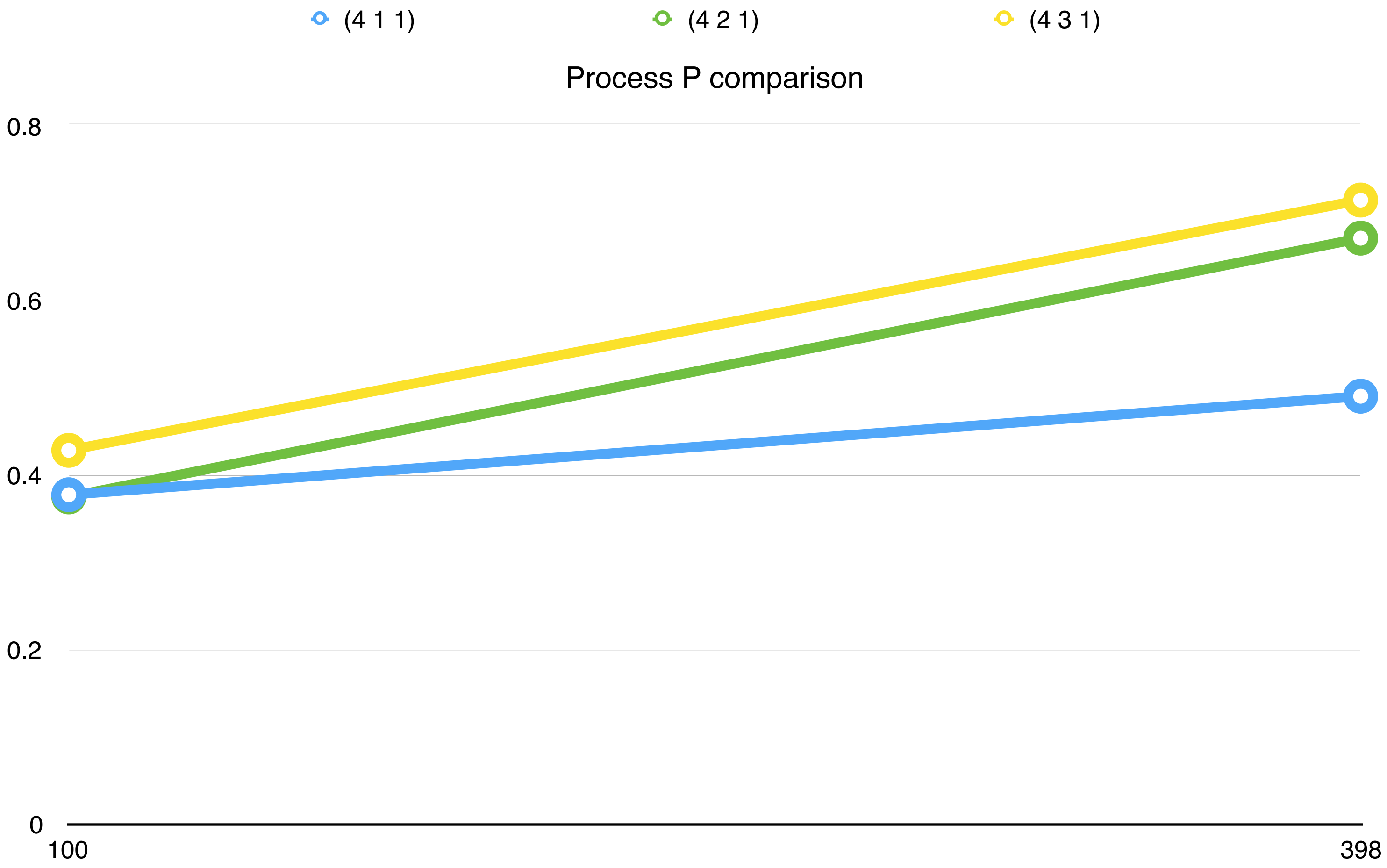
Generally from the graph we can see the average run time of processing 100 numbers will be less than the runtime of 400 numbers. This is because the number of tasks increase, processor takes more time to produce and consume. Also if N=0, the run time will be 0. So every line in this graph will touch (0,0). We can see the slope of some lines are clearly higher than others, the line of 100(4 1 1) for instance.



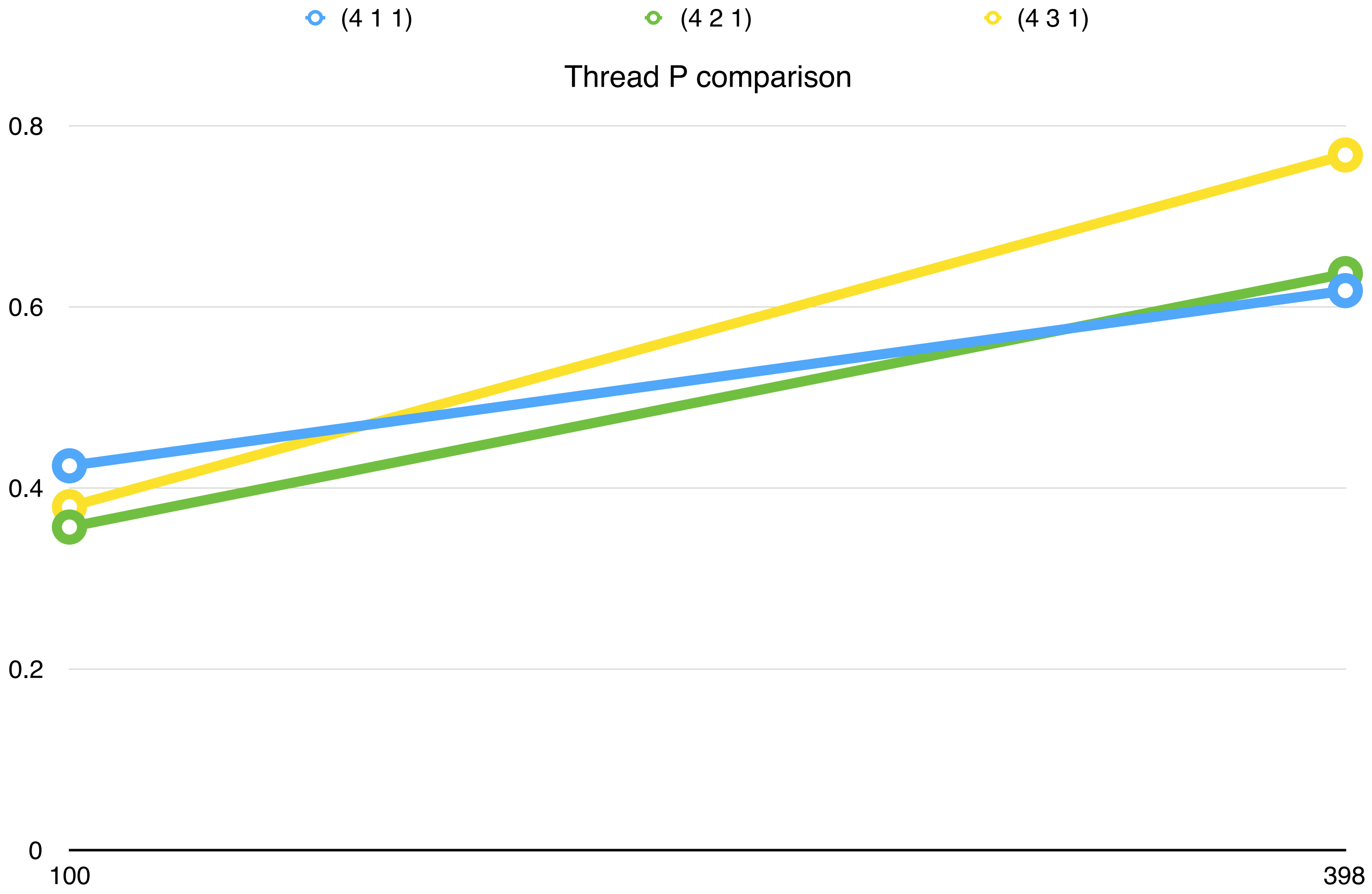
From the graph we can tell when we switch the buffer size used inside process, there is minor difference in runtime. We can see the runtime of 100 numbers with buffer size 4 is greater than buffer size 8, while the runtime of 398 numbers with buffer size 4 is conversely less than buffer size 8. When the buffer size increases, the slope of the graph decreases. As the increase of buffer size, it allows producer to produce faster instead of waiting consumer to empty the queue.



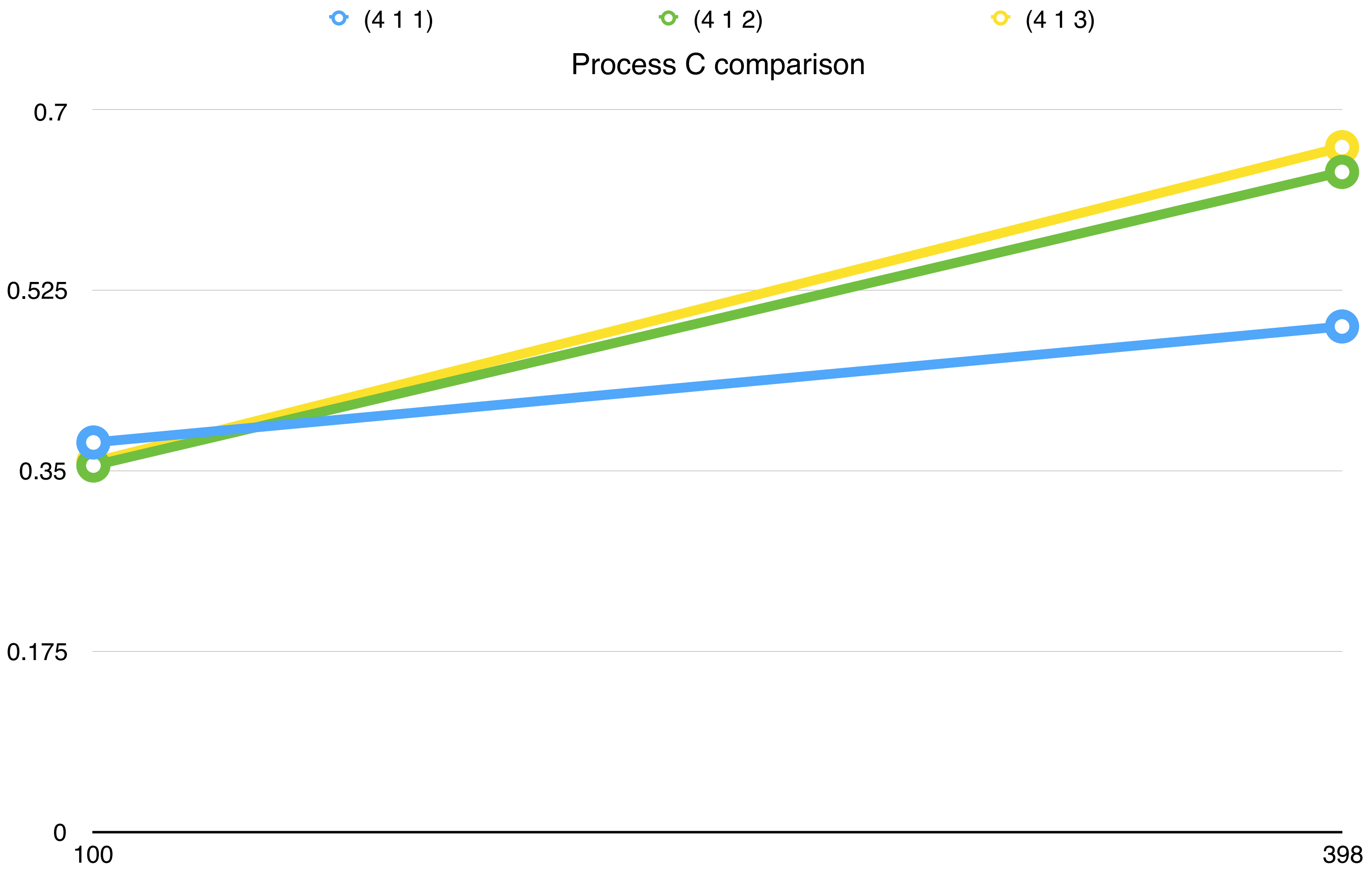
From the graph we can tell when we switch the buffer size used inside thread, there is a seeable difference in runtime. We can see the runtime of 100 numbers with buffer size 8 is greater than buffer size 4, and the runtime of 398 numbers with buffer size 8 is also less than buffer size 4. When the buffer size increases, the slope of the graph decreases. As the increase of buffer size, it allows producer to produce faster instead of waiting consumer to empty the buffer used by thread.



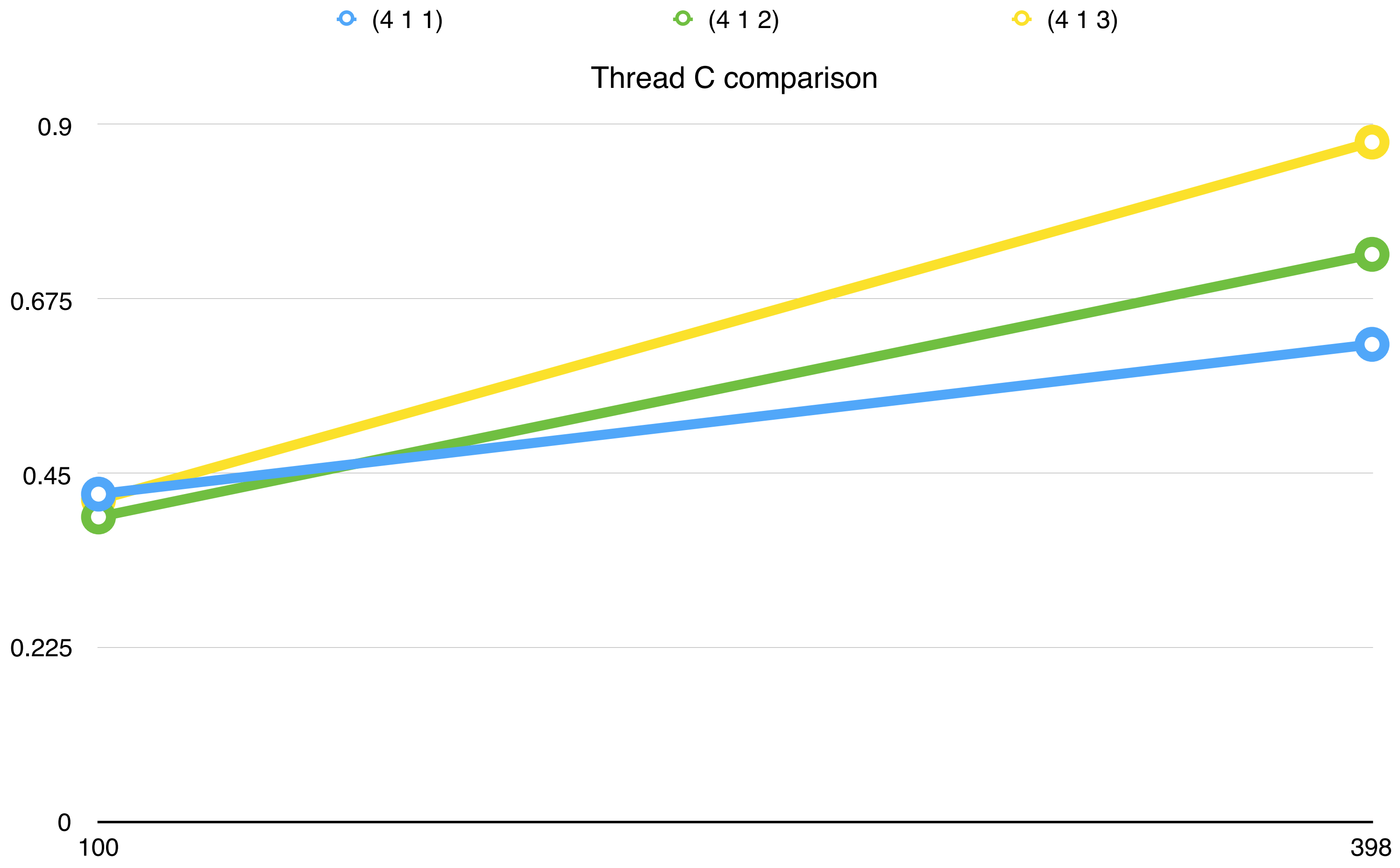
From the graph we can tell when we switch the number of producer used inside process, there is great difference in runtime. By switching the producer number from 1 to 2 to 3, we can see the runtime of both 100 and 398 numbers goes down. When the number of producer goes up, the slope of the graph does not change a lot(emit the (4 1 1) one!). As the increase of only one producer one time may not change the the overall runtime rate.



From the graph we can tell when we switch the number of producer used inside process, there is difference in runtime. By switching the producer number from 1 to 2 to 3, we can see the relation becomes complicated. We can easily find that when we increase the number of producers, the thread with more producers tend to be come slower and slower(when the number reaches 398 it becomes very obvious). This might be a trade-off. Although we increase the producer which seems like we are having more 'labour's, the waiting time among those producers is also increasing. With same amount of consumers, it is not likely to increase efficiency as we think it could be.



From the graph we can tell when we switch the number of consumer used inside process, there is great difference in runtime. By switching the consumer number from 1 to 2 to 3, we can see the runtime of both 100 and 398 numbers goes down. When the number of consumer goes up, the slope of the graph does not change a lot(emit the (4 1 1) one!). As the increase of only one producer one time may not change the the overall runtime rate. For the run time, when we increase the total process number(which is consumer in this case), it will have more chances to be blocked.



This is similar to the case of thread with changes in producer. From the graph we can tell when we switch the number of consumer used inside process, there is difference in runtime. By switching the producer number from 1 to 2 to 3, we can see the relation becomes complicated. We can easily find that when we increase the number of consumers, the thread with more consumers tend to be come slower and slower(when the number reaches 398 it becomes very obvious). This might be a trade-off. Although we increase the consumers number which seems like we are having more ‘labour’s, the waiting time among those consumers is also increasing. With same amount of producers, it is not likely to increase efficiency as we think it should be.

Generally, the runtime of process will be greater than runtime of threads when the number is low. As we can see the average runtime of threads dealing with 100 numbers is approximately 0.075ms, whereas the average runtime of threads dealing with 100 numbers is approximately 0.4ms. However, as the number of tasks increase to 400, the runtime of threads increased to an average 0.75ms, process increased to an average 0.6ms. We can tell the average slope of runtime of threads is higher than the slope of process.

So thread will be faster with small number of tasks, but the runtime will increase faster than process. Therefore for larger numbers process will be better than thread.