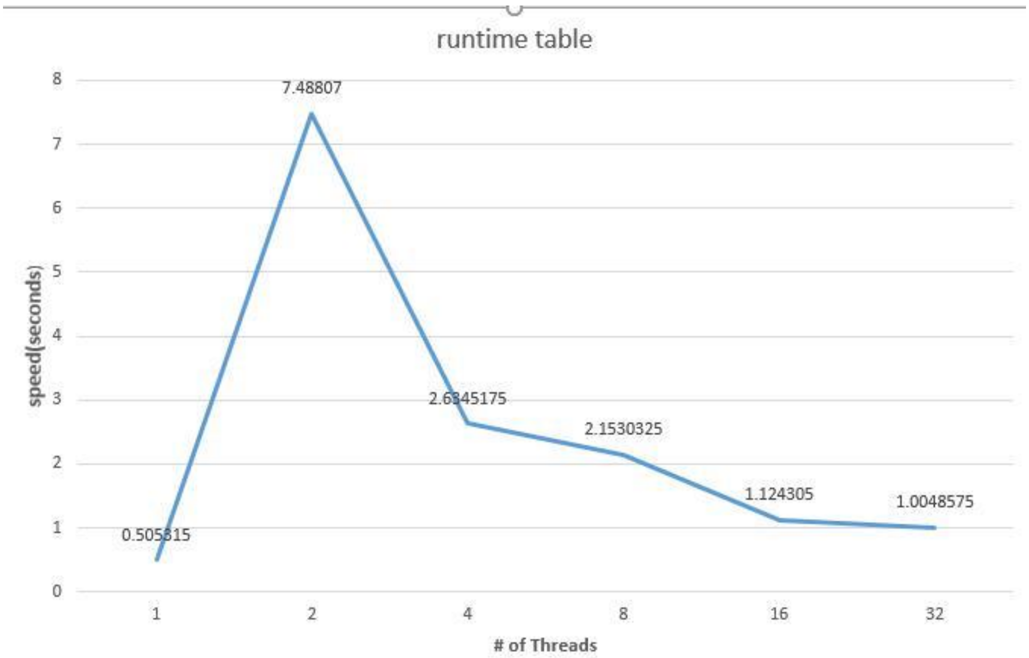
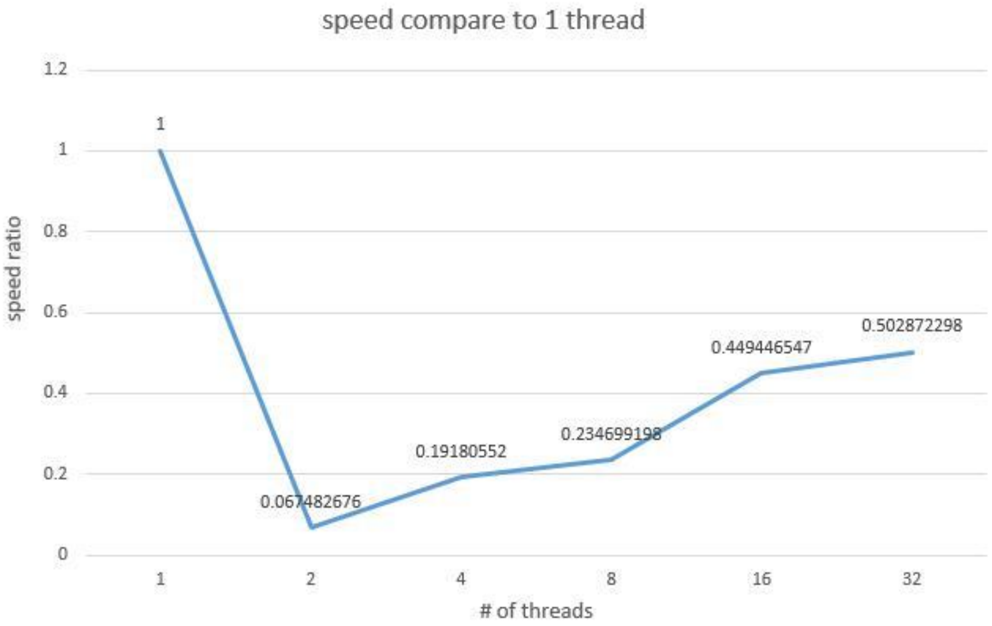
1. A) Run time (n = 10000000)



First of all, I do feel my data looks awkward. The run time data shows there is a speed up for increasing threads for 2 threads and above, but the 1 thread run case still do a faster job than any other run cases. I am running the code on my own VM, but I have tried run the code on Pleiades; the run time number was different, but the trend is still the same, 1 thread beats any other multi-threads cases.

The data showed a serial code beating the multi-threads code result, I think it is something goes wrong with my implementation, but I really have no idea where goes wrong.

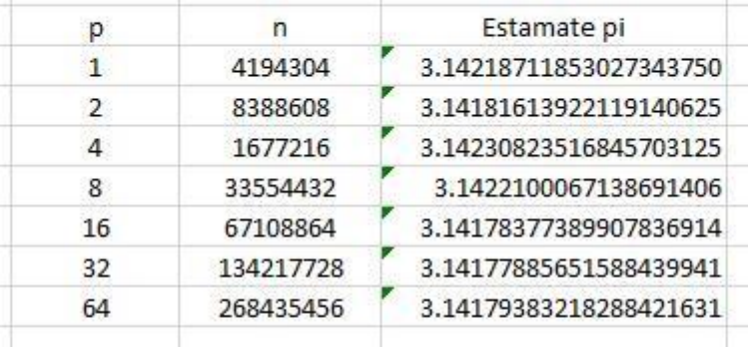
B) speed up



So, here I still try do the examine of speed up by ignore 1 thread run case.

Even through the runtime for multi thread seems unstable, but the trend looks much more reasonable from 2 to 32 threads: from 2-4 threads, there is a roughly 3 times speed up (speed ratio: 0.06748 – 0.191805), but till 16-32 threads there is only little bit speed up (speed ratio: 0.449446 – 0.502872).

2) precision



p	n	Estamate pi
1	4194304	3.14218711853027343750
2	8388608	3.14181613922119140625
4	1677216	3.14230823516845703125
8	33554432	3.1422100067138691406
16	67108864	3.14178377389907836914
32	134217728	3.14177885651588439941
64	268435456	3.14179383218288421631

The data of precision also not really looking nice from my code. at 4 threads to 32 threads cases, the estimate π value is becoming more and more closer to the real π value, but after 32 threads, the precision does not get better.

For some reason, the 2 threads run case have a really good precision compare to its neighbors. I have tried run multiple time of each case, the 2 threads run case have a stable ~3.1418 pi value on my computer.

After all, the result of speed and precision test on my 4 - 64 threads case are some how falls in an expected and reasonable test range. 1 and 2 thread case get weird result. That data shows serial code for this problem is a much better solution than multi-threading, but I think this is caused by a wrong implementation somewhere in my code.