The block diagram below shows the workflow of my NORMAL.py program. The initial execution is same as that in SCRUB.py, which includes logging, file division into blocks, and adjusting the blocks to start and end with a complete tick.

Since my SCRUB program generates noise.txt with the index of noise ticks in ascending order, it becomes easy to divide the noise.txt for each block. Because of this each block has to navigate only the part of noise file that includes noises in that block.

The main function “get_stats()” does all the major work of reading the block in chunks, reading noise, calculate returns and other parameters required to finally compute normal distribution statistics for the data. The approach to calculate running stats for normal distribution is taken from the article on
https://www.johndcook.com/blog/skewness_kurtosis/

For each block, there will be one price for which we will not know the last price as that will be a part of some other block, hence, if there are 3 processors (data file is divided into 3 blocks), each block has 20 signal ticks, then each block will have only 19 returns. This means that we will loose as many returns as the number of processors, which is not significant compared to the amount of data. This is a tradeoff my Normal program makes.

Once each block computes its parameters, they are sent to root node where further processing is done to combine them using reduce functionality and a defined function. Overall, it takes about 90 seconds to run for small data file and about 3.5 minutes for large data file. If run on Penzias the times may reduce as the systems will have more computing power.