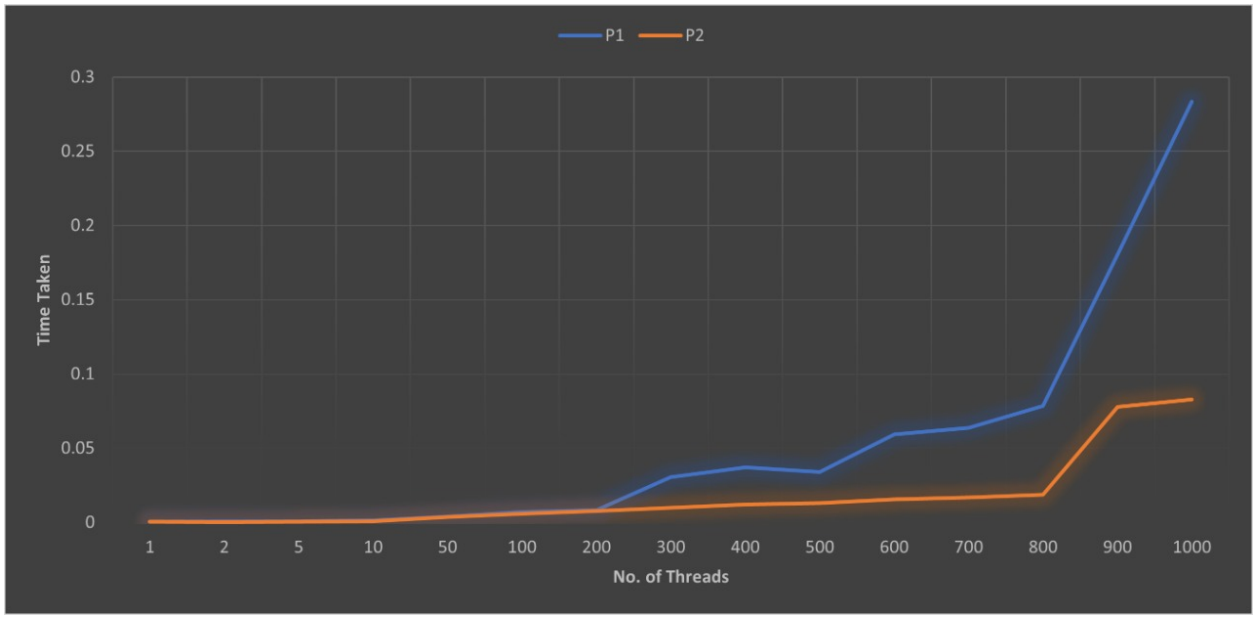
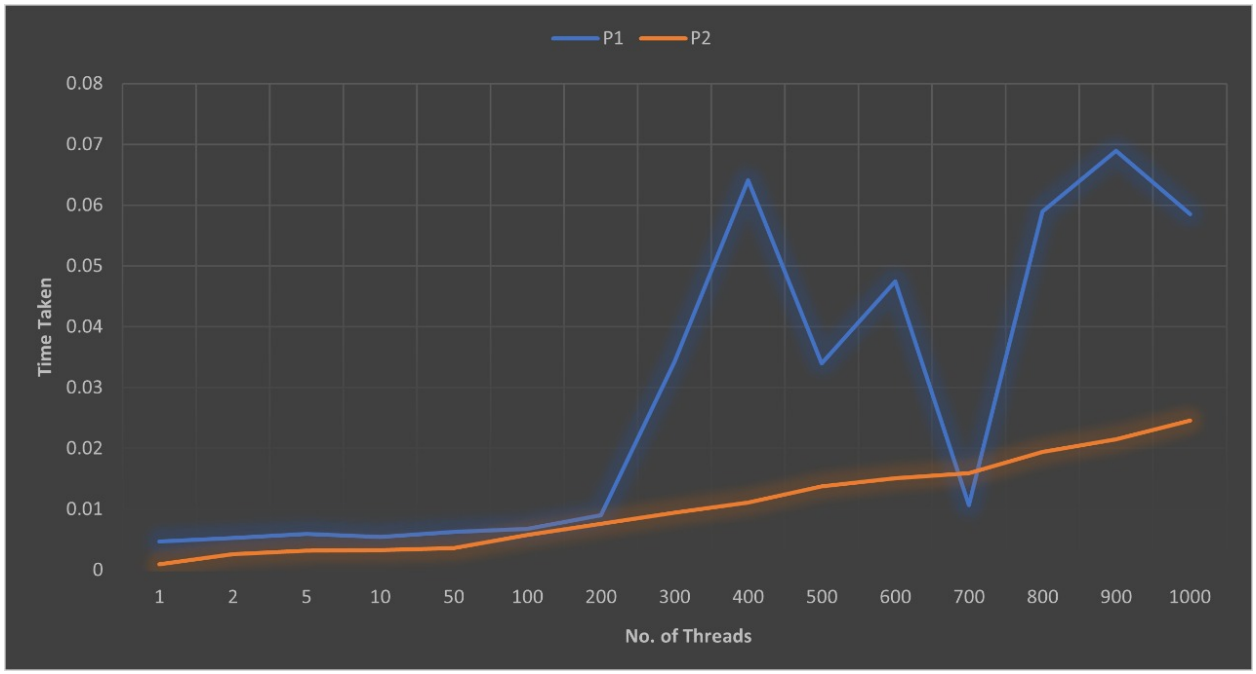
OsamaZameer f20191509@hyderabad.bits-pilani.ac.in
KushaalTummala f20180422@hyderabad.bits-pilani.ac.in
SushanthKunchala f20180411@hyderabad.bits-pilani.ac.in
SaiVamshiKamaraju f20180420@hyderabad.bits-pilani.ac.in
Athul V f20180860@hyderabad.bits-pilani.ac.in
AmanKumar f20180548@hyderabad.bits-pilani.ac.in
SatyamKumar f20180403@hyderabad.bits-pilani.ac.in

Following are the graphs obtained from Process P1 and P2

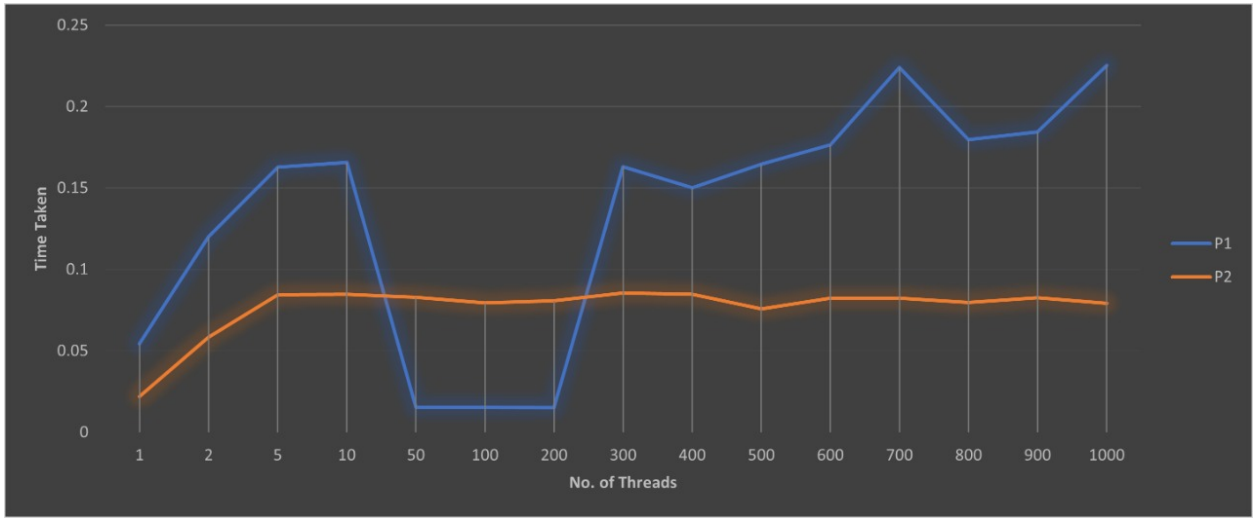
**Hundred.txt**



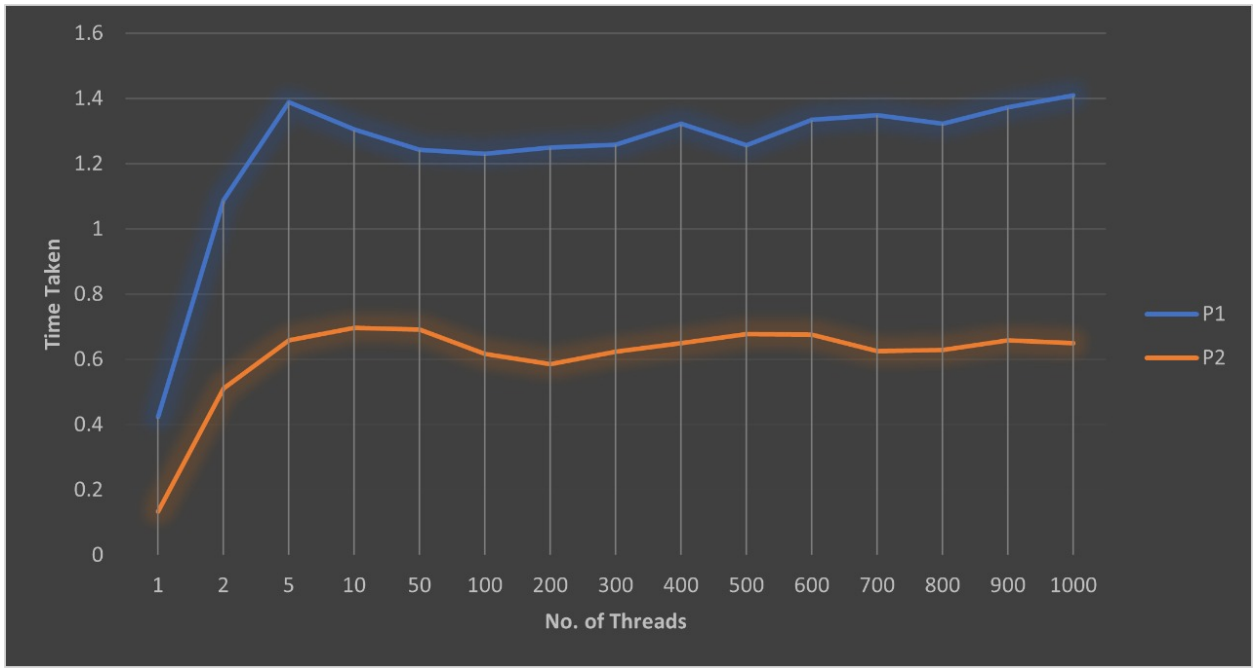
**Thousand.txt**




**OneLakh.txt**




**OneMillion.txt**



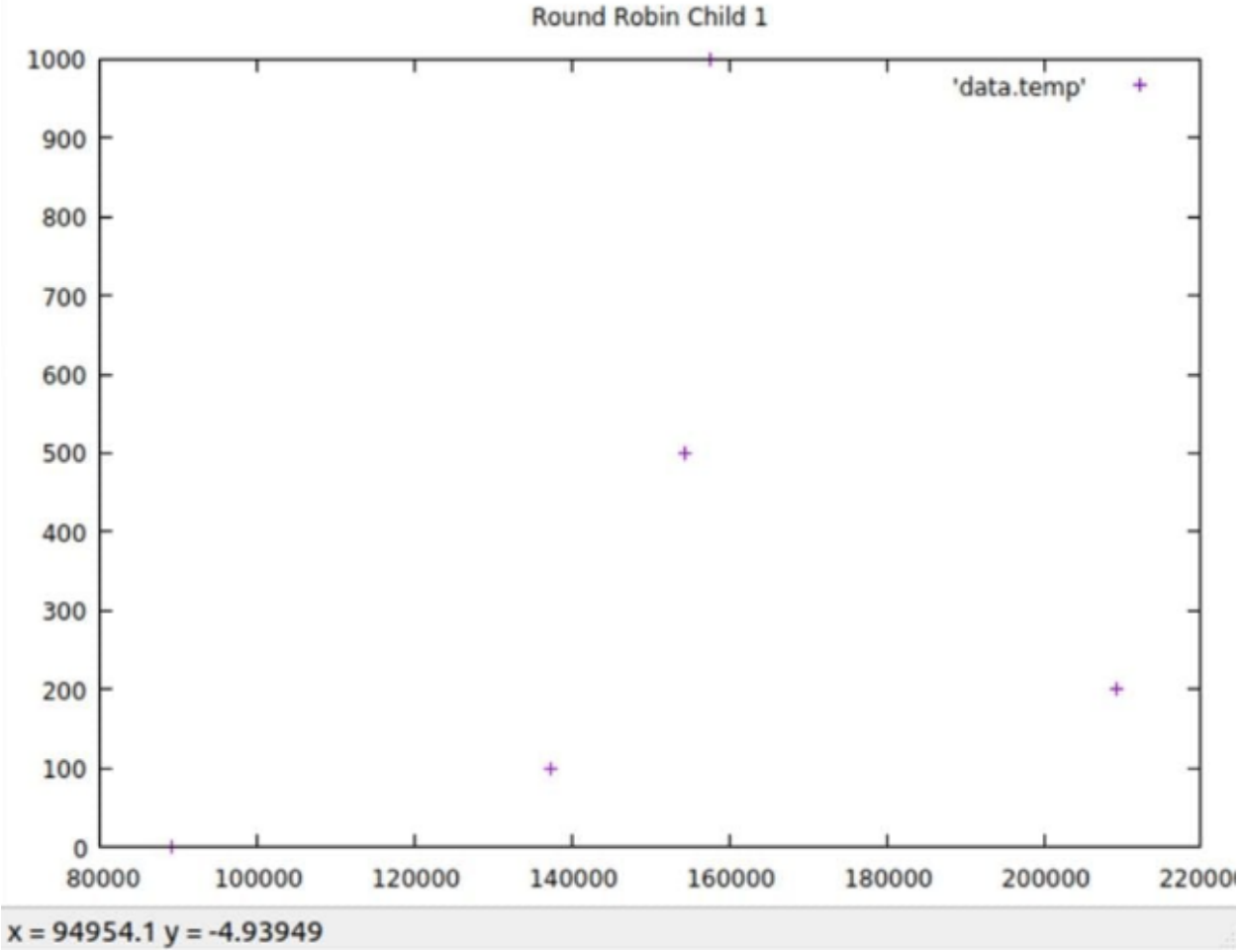

We can identify that the perfect number of threads to use is highly variable, depending on the number of threads used, the amount of data read and processed. As of now, we have not reached any conclusive results on the perfect number of threads to use in a process.And as in most cases(pursuing the txt files), the single threaded solution works, this might be because of File Locking followed by the filesystem or because of us using a HDD to read from.

**“It is never a good idea to read from the same physical (spinning) hard disk from different threads simultaneously, because every switch causes an extra delay of around 10ms to position the read head of the hard disk (would be different on SSD)” ~** Souced(StackOverflow)

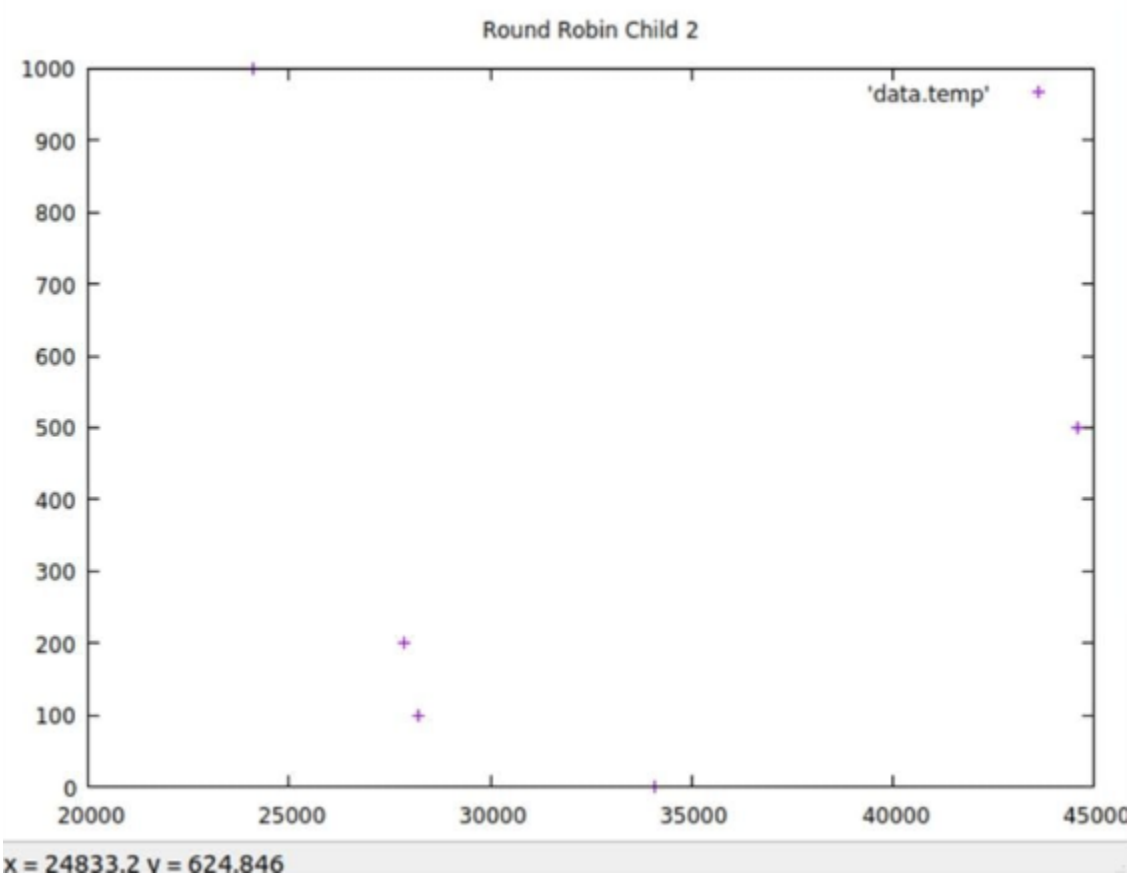
Also you can see a significant difference in speed in bigger files when using threads. As the 10ms delay mentioned above is small comparatively.

But as we look into P2. The process is significantly faster than P1. This is due to the nature of processing the data that is being sent to it and it does not have to go through the hassle of the HDD. Albeit even there a single thread is faster due to use of mutex lock on the sum integer on the multithread for synchronization.

Process P1 - Here X axis is in microseconds, Here Y is no of threads.



Process P2 - Here X axis is in microseconds, Here Y is in threads.




**Analysis**

The pink dots represent various text files. As you can see the P2 completes significantly faster in round robin scheduling compared to P1.

Difference between Preemptive priority scheduling vs Round Robin

In  Preemptive priority The average waiting time and average response time is unknown beforehand.
It is easy to implement and best suited for real time operating systems.
The problem of blocking of a process can be solved using aging.

Round Robin
Round-Robin (RR) executes the processes based upon the time quantum defined i.e. each process is executed for a fixed amount of time.
Round-Robin (RR) is preemptive in nature.
The average waiting time for a given set of processes is quite small and depends on the time quantum.

It is quite easy to implement RR in any system.Each process is executed and every user feels that his work is being done as the CPU gives equal amount of time to each process.