Implementing matrix multiplication using pthreads and calculating the absolute column sum norm of the resultant matrix.

All the programs were run in my local machine as i have problems accessing the cluster.

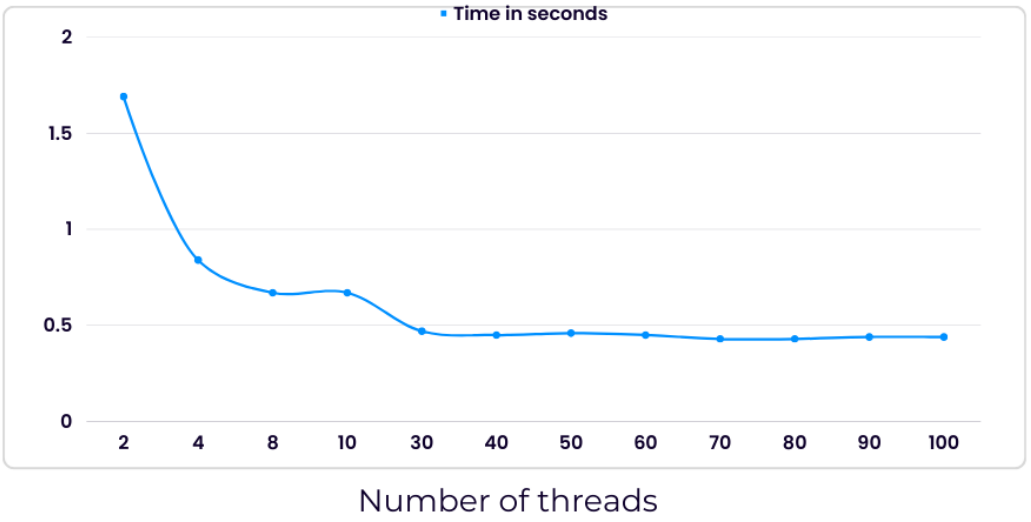
System info

```
Host Name:                 NITRO
OS Name:                   Microsoft Windows 11 Home Single Language
OS Version:                10.0.22621 N/A Build 22621
OS Manufacturer:           Microsoft Corporation
OS Configuration:          Standalone Workstation
OS Build Type:             Multiprocessor Free
Registered Owner:          abhijithsathyendran@gmail.com
Registered Organization:   N/A
Product ID:                00342-42637-92137-AAOEM
Original Install Date:     5/13/2023, 2:50:27 PM
System Boot Time:          11/14/2023, 7:36:52 PM
System Manufacturer:       Acer
System Model:              Nitro AN515-58
System Type:               x64-based PC
Processor(s):              1 Processor(s) Installed.
                           [01]: Intel64 Family 6 Model 154 Stepping 3 GenuineIntel ~3100 Mhz
BIOS Version:              Insyde Corp. V2.03, 2/24/2023
Windows Directory:         C:\Windows
System Directory:          C:\Windows\system32
Boot Device:               \Device\HarddiskVolume1
System Locale:             en-us;English (United States)
Input Locale:              en-us;English (United States)
Time Zone:                 (UTC+05:30) Chennai, Kolkata, Mumbai, New Delhi
Total Physical Memory:     16,088 MB
Available Physical Memory: 6,548 MB
Virtual Memory: Max Size:  23,512 MB
Virtual Memory: Available: 10,953 MB
Virtual Memory: In Use:    12,559 MB
Page File Location(s):     C:\pagefile.sys
Domain:                    WORKGROUP
Logon Server:              \\NITRO
Hotfix(s):                 7 Hotfix(s) Installed.
                           [01]: KB5030651
                           [02]: KB5032007
                           [03]: KB5012170
                           [04]: KB5026039
                           [05]: KB5032190
                           [06]: KB5031592
                           [07]: KB5032393
Network Card(s):           4 NIC(s) Installed.
                           [01]: Bluetooth Device (Personal Area Network)
                                 Connection Name: Bluetooth Network Connection
                                 Status:          Media disconnected
                           [02]: Killer(R) Wi-Fi 6 AX1650i 160MHz Wireless Network Adapter (201NGW)
                                 Connection Name: Wi-Fi
```

Variants received. -1b , 2a, 3a

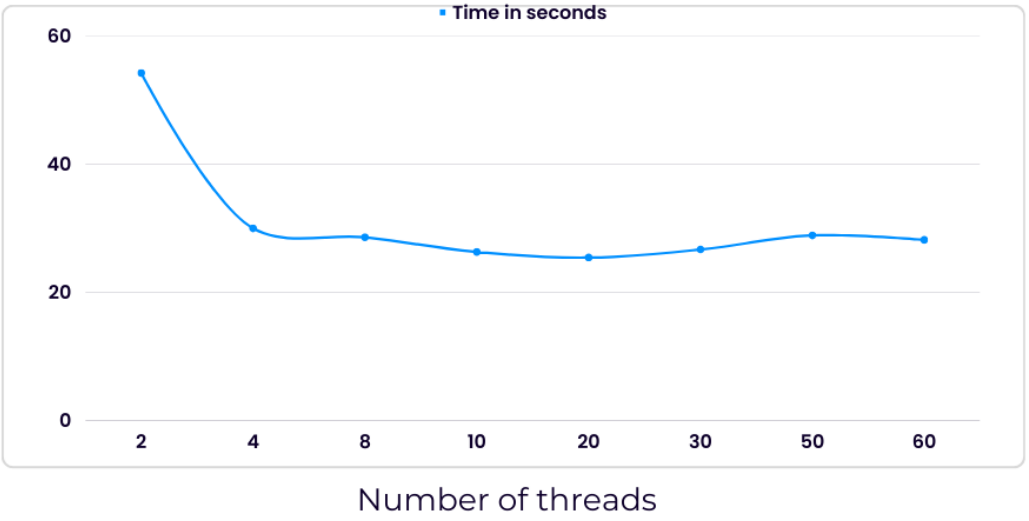
- **MATRIX SIZE N=1024.**
  With a fairly large size as n=1-24 normal naive matrix multiplication takes around. 7.6 seconds. See attached excel for accurate values.
  This time is significantly reduced by sharing the workload among parallel threads. Even with just 2 threads, the execution time is reduced to 1.7 seconds.

It achieves a peak performance of 0.4 seconds from around 30 threads, after which it reaches a saturation point. And no significant increase in performance with more threads is observed.

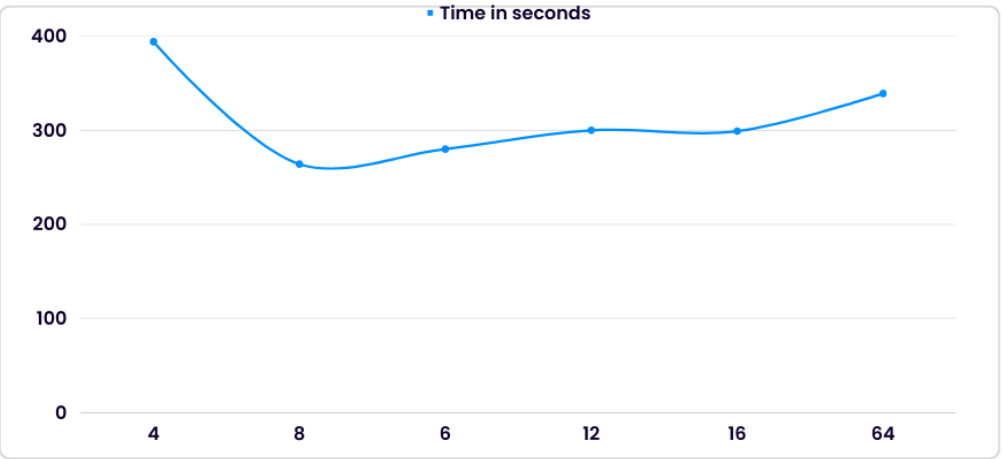


- **MATRIX SIZE N=2048**

  With a matrix 2048, the single threaded implementation takes around 222 seconds. As per time complexity algorithms, a matrix operation is of the order of n^3. When the matrix size is doubled it would increase the execution time 8 folds. In this case the magnitude of increase is much more.
  Introducing parallelism and multithreading reduces the time to as low as 25 seconds with as many as 20 threads. On further increasing the number of threads we do not observe the same level of performance



- **MATRIX SIZE N=4096**
  The execution time for a straightforward matrix multiplication is over 600 seconds. A multithreaded implementation of the same is about to bring it down to as low as 260 seconds.
  With such a large matrix size any number of threads more than 8 reduces the performance.



OBSERVATION

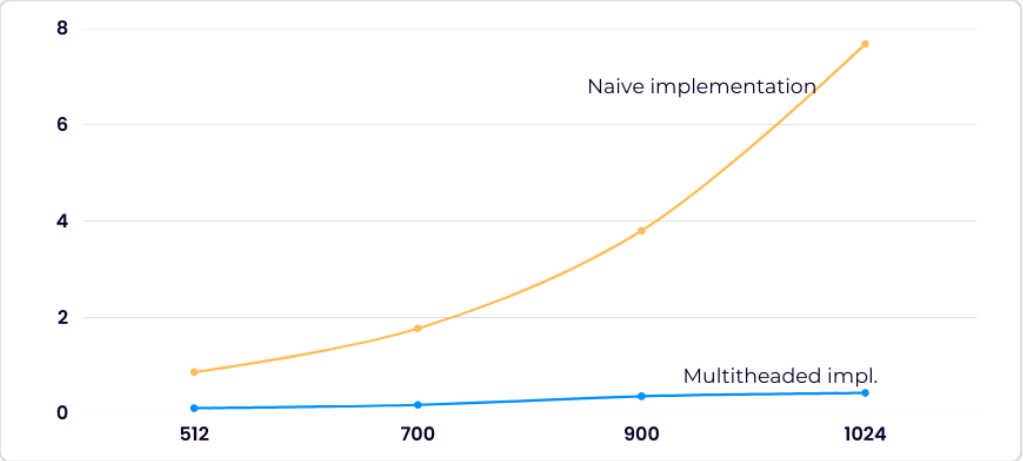
—----------

Overall there is a significant difference in execution time between a single and multi threaded matrix multiplication. As expected with more number of threads to effectively utilise system resources and parallelize task, performance is improved,

In this implementation the Left matrix is partitioned into rows and fed to multiple threads. In this way each thread operates on its own set of rows and calculates the max column sum norm. The result is kept in a global list which is synchronised using mutex.
Following graph depicts how multithreaded applications performed much faster in comparison to their single threaded counterparts.

Almost all of the multithreaded computations were finished in under a second whereas it took almost 8 seconds for matrices with sizes 1024 under single sequential execution.

Infinitely increasing the number of threads however does not guarantee any increase in performance. As observed from the explanations for matrices with size 1024,2048 and 4096, there is a saturation point for the number of matrices, beyond which increasing the number of threads consumes more resources, blocks other threads and slows down the performance.



Matrix size v time in seconds