OS2 CS3523

Name: Swapnil Bag

Roll no: ES22BTECH11034

Aim: To evaluate the efficiency of a parallel matrix squaring algorithm by studying the impact of assigning individual threads to CPU cores.

Low level design

- First the input matrix and output matrix (C_chunk and C_mixed to store answer) is globally defined to have a max size of 4096
- A structure th_param is created that stores the matrix size N, the thread value i (0,1,2….) the chunk size p and the max number of threads K
- In main function value of N, K, C ,BT and matrix is read through command line argument
- Worker thread arrays are created of size K (threads_1 for chunk and threads_2 for mixed method)

- **Experiment 1: program 1**

  Calculate the number of cores needed to bind the Bounded threads, BT/b (b = K/C) . First iterate over the number of cores required and clear the cpu_set using CPU_ZERO() and set the core using CPU_SET(). Then the thread id of the created thread is obtained and bound to the core using pthread_setaffinity_np() to the cpu_set structure. The remaining threads scheduled normally using pthread_create and pthread_join. The total time is measured for both chunk and mixed method

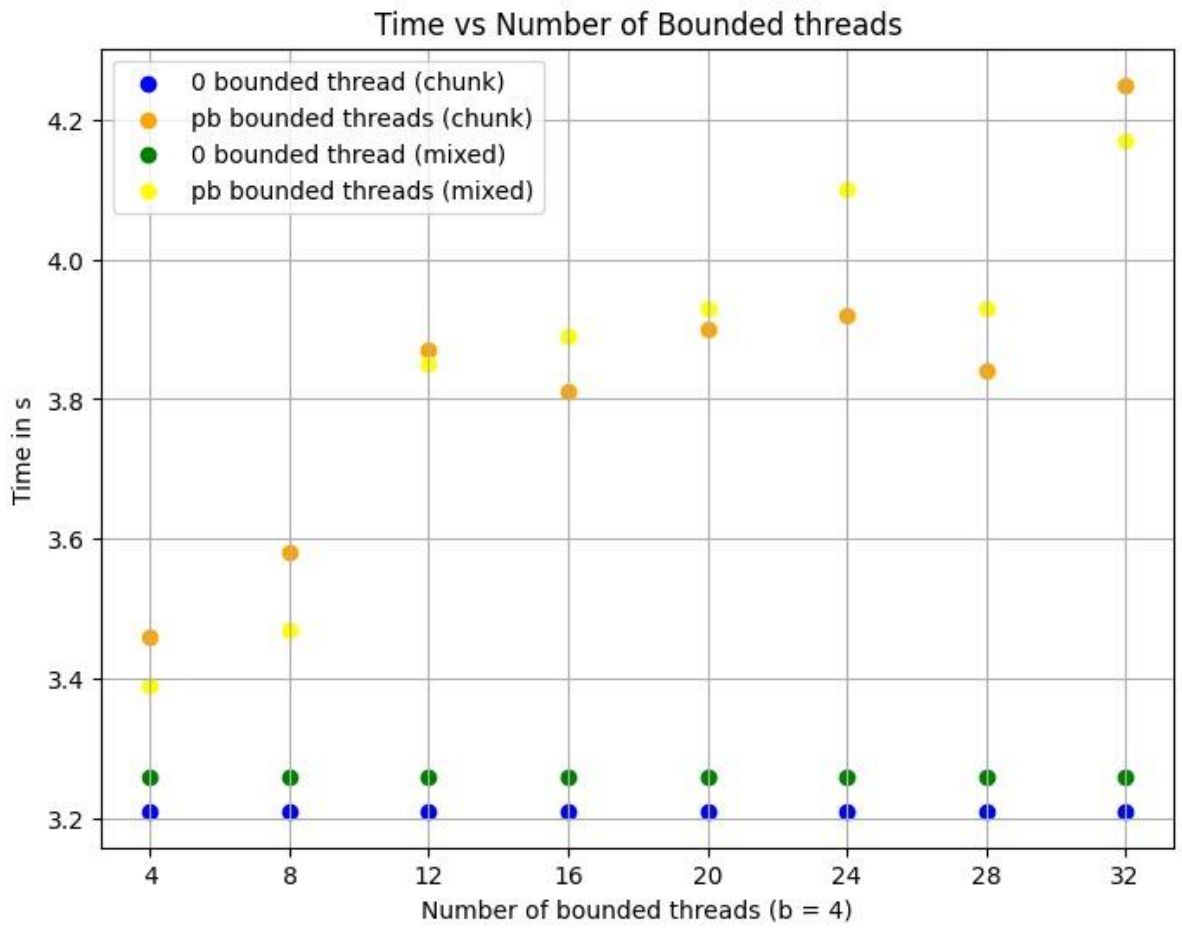
  Experiment 2: program 2

  Here K/2 threads are distributed equally to C/2 cores and the rest K/2 threads are scheduled by the OS. Determine the number of threads given to each core. Time taken by program without binding is measured (assgn 1). First iterate over the number of cores required and clear the cpu_set using CPU_ZERO() and set the core using CPU_SET(). The threads are created by pthread_create() by passing the function name (square_chunk or square_mixed) and data argument. Then the thread id of the created thread is obtained and bound to the core using pthread_setaffinity_np() to the cpu_set structure. If BT < C/2 then the bound threads are assigned to core0. The remaining threads scheduled normally using pthread_create and pthread_join. The total time is measured for both chunk and mixed method

- **Output** : printed to out.txt
  Experiment 1 : time taken for both chunk and mixed method
  Experiment 2: Average time taken by bounded threads, unbounded threads and total time taken for both methods

- **Hardware** : Intel(R) Core(TM) i5-1021U CPU @ 1.60 Hz cores : 4 logical cores : 8 C = 8
  (intel hyperthreading)

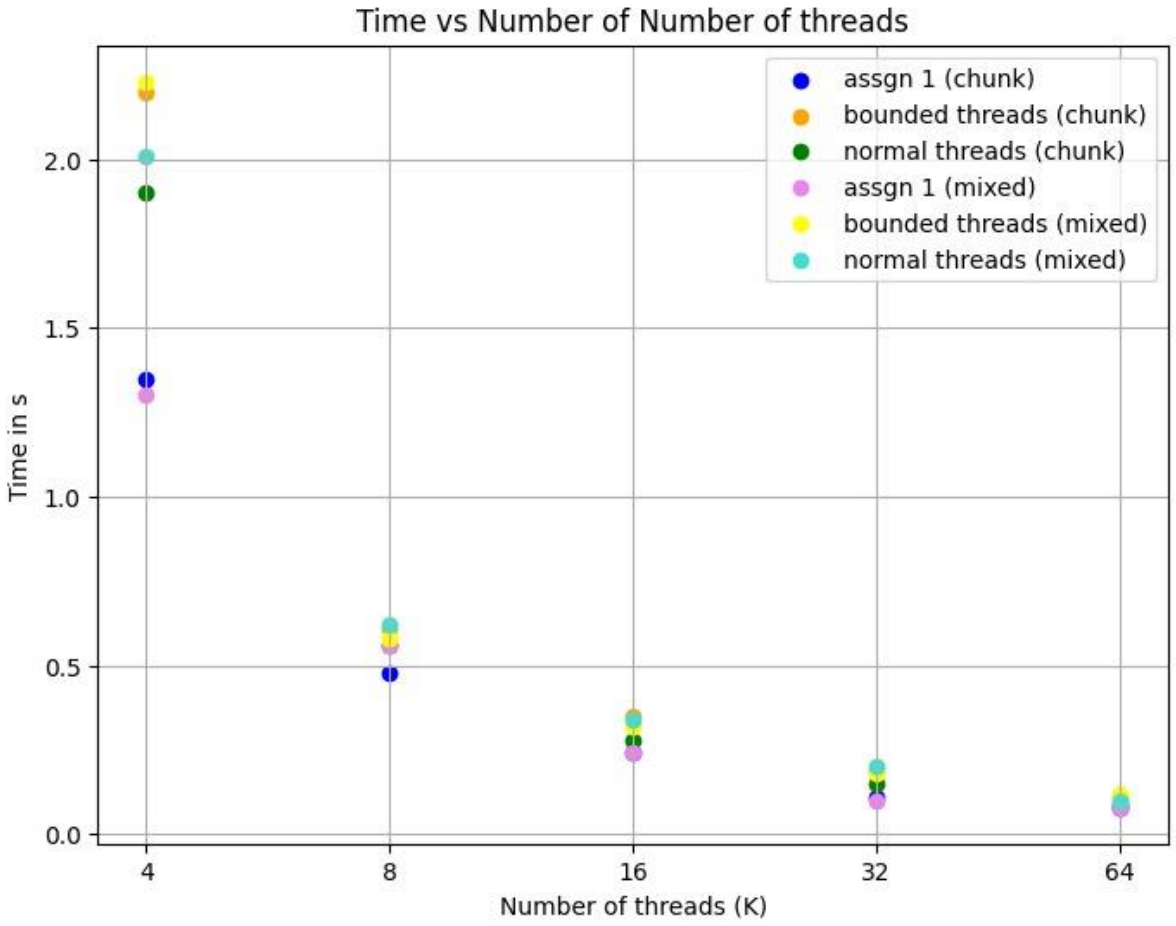
Experiment 1



	Bounded Threads	chunk	mixed
0	0	3.21	3.26
1	4	3.46	3.39
2	8	3.58	3.47
3	12	3.87	3.85
4	16	3.81	3.89
5	20	3.90	3.93
6	24	3.92	4.10
7	28	3.84	3.93
8	32	4.25	4.17

From the graphs its evident that on binding the threads to certain number of cores, the time increases. The possible reasons could be increase in workload on certain cores when it handles b threads bound to it and other threads are also assigned by OS scheduler that are not bound, this might lead to inefficient utilisation of resources and improper load balancing. There are frequent context switches that leads to an overhead. Binding multiple threads to the same cores can increase cache pressure, as threads access data or contend for space in the CPU's caches. There is no evident pattern in performance and is random and increases possibly because of the other background processes or how OS schedules tasks

Experiment 2



	Bounded Threads	chunk assgn1	chunk bound	chunk normal	mixed assgn1	mixed bound	mixed normal
0	4	1.35	2.20	1.90	1.30	2.23	2.01
1	8	0.48	0.61	0.56	0.56	0.58	0.62
2	16	0.24	0.35	0.28	0.24	0.32	0.34
3	32	0.11	0.19	0.15	0.10	0.18	0.20
4	64	0.08	0.11	0.09	0.08	0.12	0.10

The average time for a thread when scheduled without binding reduces with increasing K. But The average time for bounded K/2 threads is more than K/2 normal threads because of workload imbalance on certain cores or cache pressure. But the average time for both reduces as K increases and becomes nearly equal as the workload tends to become equally shared and more parallelism.