Report

Operating Systems–2: Spring 2024
Programming Assignment 2 : Thread Affinity
Submission Deadline: 23rd February 2024, 9:00 pm

Name: P.Yasaswini
Roll No : CS22BTECH11046

Program Low-Level Design:

- **1. Variables declared globally:**
  - First declaring out_put_matrix 2-D Vector.
  - globally,"times" vector to store running time of each thread.
  - Declared respective variables n,k,c,BT and p(n/k),rem(n%k).
  - Used struct to store thread data i.e input_matrix, start_row,end_row.
- **2.Chunk_multiplication:**
  - As a thread enters the function, time will start for that thread and the stored data in the specific thread will be given to the temporary variable "thread" to access that safely we will cast it.
  - First every thread gets p(n/k) threads equally but if k is not multiple of n then some thread will be left. So they are assigned to threads starting from the first thread(i=0,1,2..n%k-1) and we will do multiplication and store it to the output_matrix.
  - After that we will end the clock , we will store the duration time in the "time_spent" variable using chrono:: high_resolution_clock.
  - Vector "times" will store the "time_spent" according to the thread number.
  - We will delete allocated memory for"thread_data".

- **3.Mixed Multiplication:**
  - Here thread data will store only index and input matrix.
  - We will assign rows using indexes as we did in assignment 1.
  - Same as chunk but assigning rows to threads is different,Here remaining rows are distributed as each row is given to first l(BT/b) threads.
- **4.Main Function:**
  - A local variable ℓ is defined for the maximum number of cores that can be given “b” threads equally i.e BT/b. But if k<c the ℓ = BT i.e 1 thread is given l cores.
  - Now we read the input file.
  - We will assign values to p,b,rem(rows left after dividing p rows to k threads) using n,k,c,BT variables which are declared globally.
  - We will declare the input_matrix to store the data in input_file ,after reading the input file we will close it.
  - Creating Cpu sets using cpu_set_t, we will take an array of cpu_sets[l], initially zero out all of them, and set them.
    - CPU_SET(i, &cpu_sets[i])
    - i is CPU index
    - &cpu_sets[i] is pointer to the CPU set
  - We will start time for overall performance of threads when some threads are being bound.
  - We created pthreads using vectors.
  - As BT is always less than K then ℓ will be less than C.
    - ℓ= BT/b=(BT/k)*c.
    - If BT < k then BT/k <1
    - Implies ℓ will be less than C.

- We will assign b threads to ℓ cores, here for assigning I used pthread_setaffinity_np and assert will help to debug we can safely distribute threads to cores.
- And the remaining threads are distributed by OS.
- At last all threads will join and terminate.
- We will end time for overall performance of threads when some are being bound.
- Duration is stored in “time_taken”.
- For experiment 1:
  - 1.We will give input BT=0 and time_taken will give the output
  - 2. Variable “Time_taken” will be the output.
- For experiment 2:
  - 1. BT=0, sum all the times stored in times[ ] array and divide / number of threads(k).
  - 2. Here sum times, stored in times[ ] array from index=0 to index = BT-1, and divide the sum by BT when BT is not equal to 0.
  - 3. For normal threads sum times in times[ ] array from index = BT to index = K-1 and divide the sum by (K- BT) when K is not equal to BT.
- Finally Write all results to the output_file.

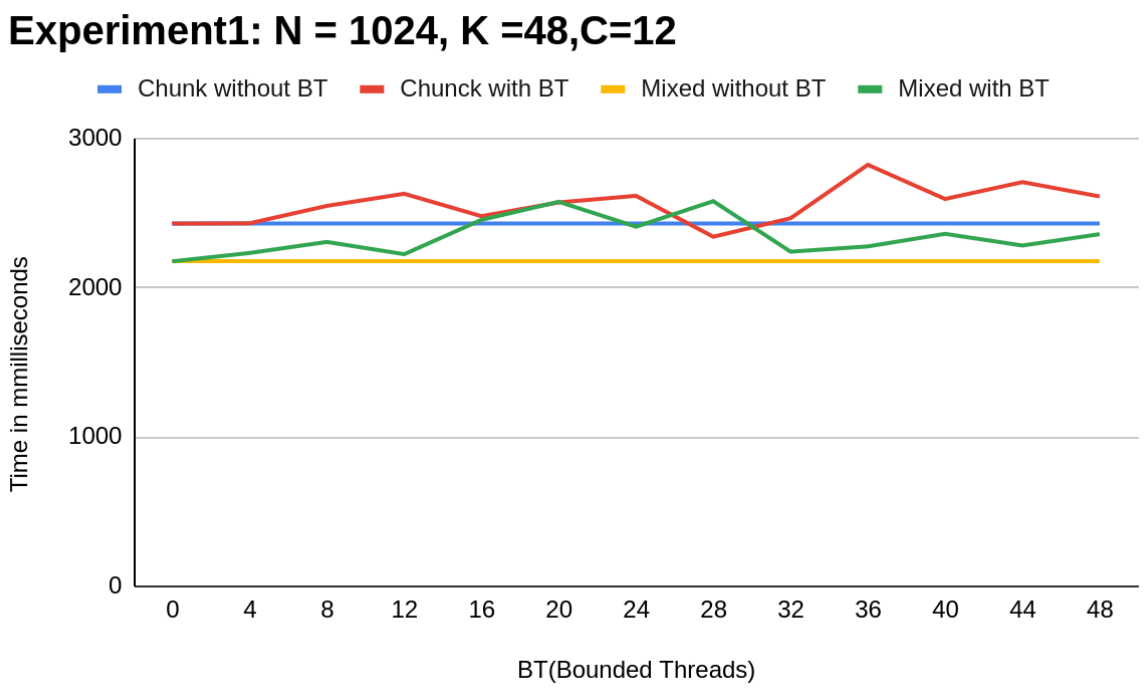
Experiment1:

N = 1024, K = 48, C = 12 are fixed

	Chunk without BT	Chunck with BT	Mixed without BT	Mixed with BT
0	2425	2425	2173	2173
4	2425	2427	2173	2229
8	2425	2544	2173	2303
12	2425	2625	2173	2221
16	2425	2475	2173	2451
20	2425	2567	2173	2572
24	2425	2611	2173	2405

28	2425	2337	2173	2576
32	2425	2461	2173	2238
36	2425	2819	2173	2273
40	2425	2590	2173	2357
44	2425	2703	2173	2279
48	2425	2607	2173	2355

Graph-plot:



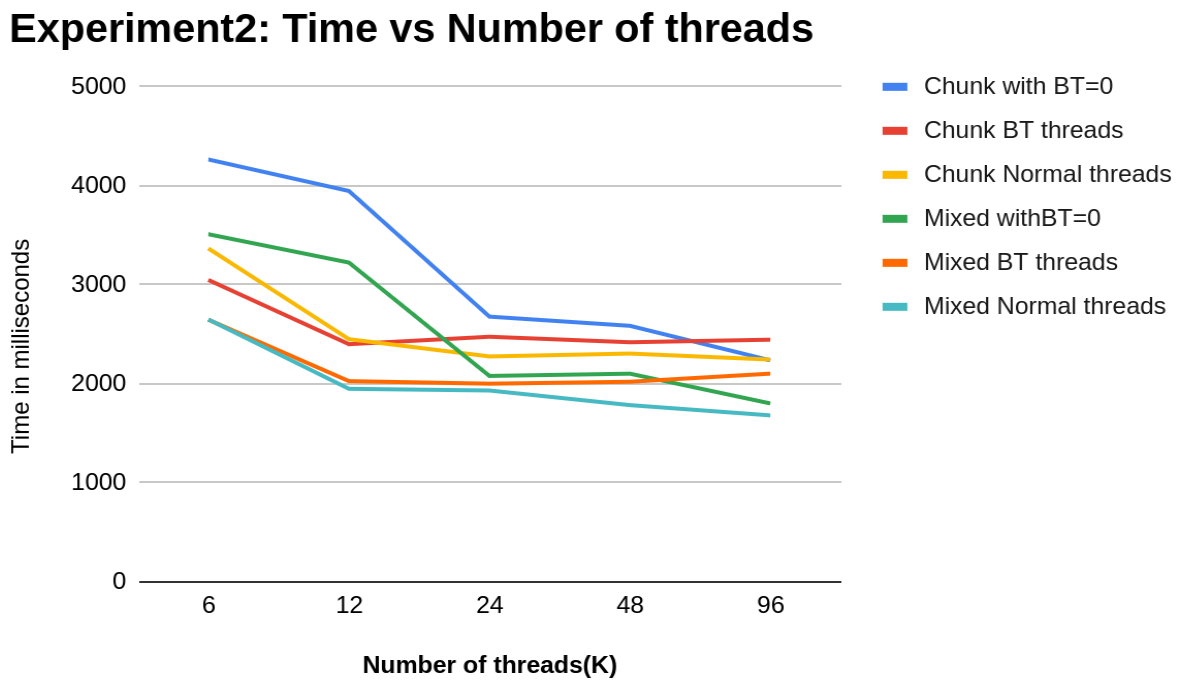
Observations:

- Without Bounded threads both performance of chunk and mixed almost the same.
- With Bounded threads mixed shows better performance than chunk while increasing bounded threads.

Experiment2:

	Chunk with BT=0	Chunk BT threads	Chunk Normal threads	Mixed withBT=0	Mixed BT threads	Mixed Normal threads
6	4261	3043	3363	3507	2643	2645
12	3943	2395	2447	3221	2023	1945
24	2674	2470	2273	2076	1997	1928
48	2581	2414	2302	2099	2017	1780
96	2232	2440	2241	1798	2099	1677

Graph-plot:
N= 1024, C=12;



Observation:

- Here on increasing the number of threads both chunk and mixed with/without BT performance increases because we are distributing work to more threads implies workload per thread decreases.
- As same as experiment 1 with bounded threads mixed shows better performance.

- For both chunk and mixed, BT threads take less time than normal threads because we assign threads to cores such that threads will run in parallel and lead to take less time.
- Conclusion: On bounding some threads increases performance for squaring matrix for both chunk and mixed.