System Info:

Operating system is a virtual machine in virtualbox:

Linux ubuntu 6.2.0-39-generic #40~22.04.1-Ubuntu SMP PREEMPT_DYNAMIC Thu Nov 16 10:53:04 UTC 2 x86_64 x86_64 x86_64 GNU/Linux

CPU:

CPU(s) (core): 10
Model name: 12th Gen Intel(R) Core(TM) i5-12500H
CPU op-mode(s): 32-bit, 64-bit
Address sizes: 39 bits physical, 48 bits virtual
Caches (sum of all):
L1d: 480 KiB (10 instances)
L1i: 320 KiB (10 instances)
L2: 12,5 MiB (10 instances)
L3: 180 MiB (10 instances)

Memory:

MemTotal: 9810576 kB

Manufacturer	PartNumber	Speed
Crucial Technology	CT16G48C40S5.M8A1	4800
Samsung	M425R1GB4BB0-CQKOL	4800

Compiler:

gcc (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0

Comparison of mpi and pthread:

MPI (Message Passing Interface) and Pthreads (POSIX Threads) are both parallel programming models, each designed for specific parallel computing scenarios. MPI operates in a distributed memory model, where processes communicate through explicit message passing, making it well-suited for large-scale parallelism across multiple nodes in a cluster or supercomputer. In contrast, Pthreads is tailored for shared memory systems, allowing multiple threads within a single process to communicate more seamlessly through shared variables. While MPI provides explicit control over data distribution and inter-process communication, resulting in more complex programs, Pthreads offers implicit parallelism with easier communication among threads. MPI excels in scalability for high-performance computing, while Pthreads are more suitable for parallelism within a single machine or node, particularly in multi-core environments.

Tests and Results:

To see the results better we'll use relatively big number of toss which is 1B (1000000000)

Single threaded c program test results:

```
hsnsvn@ubuntu:~/Desktop$ ./q2 1000000000
Estimated value of Pi with 1000000000 tosses: 3.141539
Elapsed time: 24.552119
```

MPI Program Results with different num of processess

```
hsnsvn@ubuntu:~/Desktop$ mpirun -np 1 ./q3 1000000000
Elapsed time = 9.491656 seconds
Estimated Pi is approximately 3.1415903959999998
```

```
hsnsvn@ubuntu:~/Desktop$ mpirun -np 2 ./q3 1000000000
Elapsed time = 4.746100 seconds
Estimated Pi is approximately 3.1415799600000001
```

```
hsnsvn@ubuntu:~/Desktop$ mpirun -np 4 ./q3 1000000000
Elapsed time = 2.486294 seconds
Estimated Pi is approximately 3.1415523040000002
```

```
hsnsvn@ubuntu:~/Desktop$ mpirun -np 8 ./q3 1000000000
Elapsed time = 1.544183 seconds
Estimated Pi is approximately 3.1415157439999999
```

MultiThread Program Results with different num of threads

```
hsnsvn@ubuntu:~/Desktop$ ./q4 1 1000000000
-> # Thread: 1
-> # Tosses: 1000000000

-> pi: 3.141591
-> Elapsed Time: 2.757938 sec
-------------------------------------------------------
```

```
hsnsvn@ubuntu:~/Desktop$ ./q4 2 1000000000
-> # Thread: 2
-> # Tosses: 1000000000

-> pi: 3.141562
-> Elapsed Time: 1.816263 sec
-------------------------------------------------------
```

```
hsnsvn@ubuntu:~/Desktop$ ./q4 4 1000000000
-> # Thread: 4
-> # Tosses: 1000000000

-> pi: 3.141591
-> Elapsed Time: 1.525798 sec
-------------------------------------------------------
```

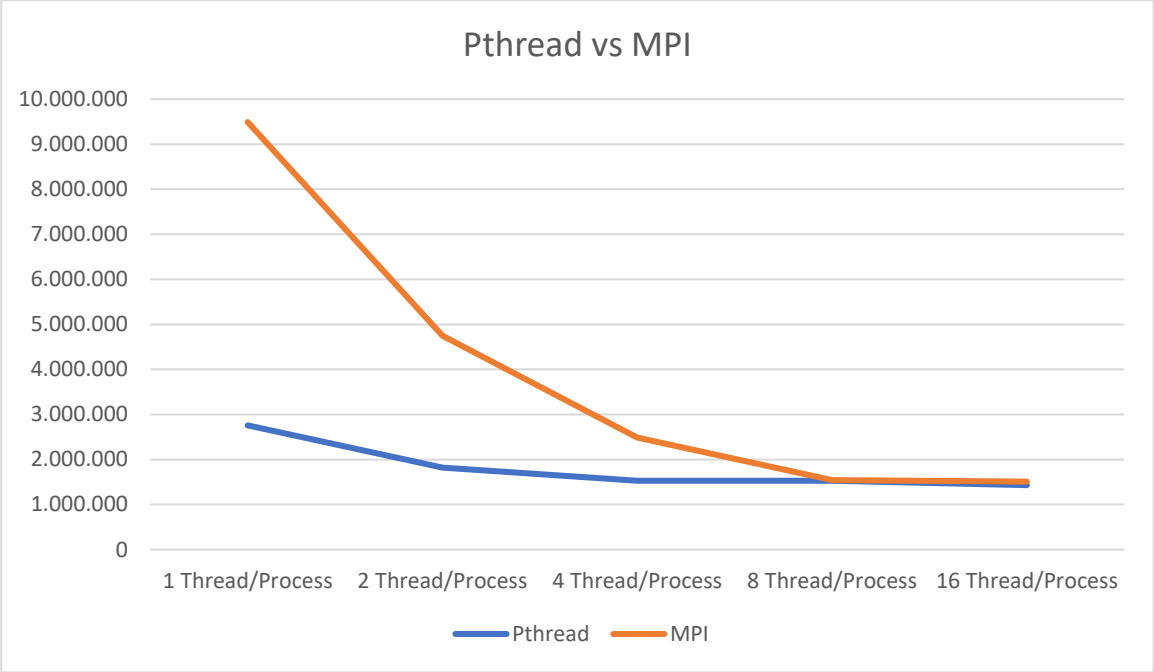
```
hsnsvn@ubuntu:~/Desktop$ ./q4 8 1000000000
-> # Thread: 8
-> # Tosses: 1000000000

-> pi: 3.141410
-> Elapsed Time: 1.525160 sec
-------------------------------------------------------
```

```
hsnsvn@ubuntu:~/Desktop$ ./q4 16 1000000000
-> # Thread: 16
-> # Tosses: 1000000000

-> pi: 3.141734
-> Elapsed Time: 1.430086 sec
-------------------------------------------------------
```

Graphs:



We can calculate efficiency with this formula:

Efficiency = Speedup / Number of Threads

Efficiency for MPI Implementation (Q3):

Efficiency = Speedup / Number of Processes
Efficiency for different process counts:
1 process: 9.491656 / 1 = 9.491656
2 processes: 4.746100 / 2 = 2.373050
4 processes: 2.486294 / 4 = 0.621574
8 processes: 1.544183 / 8 = 0.193022

Efficiency for Pthread Implementation (Q4):

Efficiency = Speedup / Number of Threads
Efficiency for different thread counts:
1 thread: 2.757938 / 1 = 2.757938
2 threads: 1.816263 / 2 = 0.908132
4 threads: 1.525798 / 4 = 0.381450
8 threads: 1.525160 / 8 = 0.190645
16 threads: 1.430086 / 16 = 0.089380
32 threads: 1.460155 / 32 = 0.045630
64 threads: 1.448853 / 64 = 0.022642

When we look at the efficiency of them, We can clierly see that the both parallel implementations are gives better results than q2. also we can
see that pthread is more efficient for the same task than mpi. This difference is because of the architecture of the mpi. While MPI is created for
distrubuted and more complex systems, pthread is created for single machines and simpler systems (as we discassed above).

Hacı Hasan Savan
1901042704