Report

Cloud Assignment 3

Team Name: Tanmay-Mithilesh-Jayanth-HW3-CC
Students:

Jayanth Chidananda A20517012 (jchidananda@hawk.iit.edu)
Mithilesh Bade A20514157 (mbade@hawk.iit.edu)
Tanmay Anand A20519843 (tanand4@hawk.iit.edu)

ANSWER -

Mode	Type	Size	Threads	Measured Time	Measured Throughput	Theoretical Throughput	Efficiency
flops	single	small	1	4.564499	2.190821	784	0.2794414541
flops	single	small	2	2.331839	4.288461	784	0.5469975765
flops	single	small	4	1.250563	7.996398	784	1.019948724
flops	single	medium	1	44.717271	2.236272	784	0.2852387755
flops	single	medium	2	23.167187	4.316450	784	0.550567602
flops	single	medium	4	12.362081	8.089253	784	1.031792474
flops	single	large	1	443.14270	2.256609	784	0.2878327806
flops	single	large	2	236.29696	4.231963	784	0.539791199
flops	single	large	4	125.20705	7.986770	784	1.018720663
flops	double	small	1	7.313272	1.367377	784	0.1744103316
flops	double	small	2	3.985464	2.509118	784	0.3200405612
flops	double	small	4	2.089803	4.785140	784	0.6103494898
flops	double	medium	1	72.077030	1.387405	784	0.1769649235
flops	double	medium	2	38.072562	2.626563	784	0.3350207908
flops	double	medium	4	21.019904	4.757396	784	0.6068107143
flops	double	large	1	723.10014	1.382934	784	0.1763946429
flops	double	large	2	382.27513	2.615917	784	0.3336628827
flops	double	large	4	206.58064	4.840725	784	0.6174394133

Mode	Type	Size	Threads	Measured Time	Measured Throughput	Theoretical Throughput	Efficiency
matrix	single	small	1	0.886999	1.049970	784	0.1339247449
matrix	single	small	2	0.458442	2.031495	784	0.2591192602
matrix	single	small	4	0.260599	3.573776	784	0.4558387755
matrix	single	medium	1	58.654233	-0.006740	784	-0.0008596938776
matrix	single	medium	2	29.326941	-0.013481	784	-0.001719515306
matrix	single	medium	4	15.619327	-0.025312	784	-0.003228571429
matrix	single	large	1	4,080.8063	-0.000332	784	-0.00004234693878
matrix	single	large	2	1943.2411	-0.000670	784	-0.00008545918367
matrix	single	large	4	1116.3879	-0.001167	784	-0.0001488520408
matrix	double	small	1	0.474647	1.962137	784	0.2502725765
matrix	double	small	2	0.245421	3.794796	784	0.484030102
matrix	double	small	4	0.149522	6.228666	784	0.7944727041
matrix	double	medium	1	32.562636	-0.012141	784	-0.001548596939
matrix	double	medium	2	16.659520	-0.023731	784	-0.003026913265
matrix	double	medium	4	9.147800	-0.043219	784	-0.005512627551
matrix	double	large	1	2,088.6303	-0.0006222	784	-0.0000793622449
matrix	double	large	2	1068.0496	-0.001220	784	-0.0001556122449
matrix	double	large	4	715.4259	-0.001821	784	-0.0002322704082

Mode	**Type**	**Size**	**Threads**	**Measured Time(In seconds)**	**Measured Throughput (Gflops)**	**Theoretical Throughput**	**Efficiency**
shpl	single	1024	4	-	-	-	-
shpl	single	4096	4	-	-	-	-
shpl	single	16386	4	-	-	-	-
xhpl	double	1024	4	0.01	605	784	77.16836735
xhpl	double	4096	4	0.34	133	784	16.96428571
xhpl	double	16386	4	15.36	190	784	24.23469388

Calculation of Theoretical throughput
FLOPS = (sockets) x (cores per socket) x (cycles per second) x (FLOPS per cycle)
FLOPS = 1 x 14 x 3.5 GHz x 16
Theoretical throughput = 784 Gflops

The following techniques were used to achieve parallelisation for matrix:

1. Pthreads: the rows were divided between the pthreads with a starting and ending row, to split the computation needed for the matrix. A divider variable was used to set the number of rows each thread should perform on, by dividing the number of rows by the number of threads. Then each thread was allowed to perform from the end of the previous thread till the number of rows set to the divider.

2. Transposing: the second matrix was transposed to make it faster by performing a row by row read, instead of a column by column. The processor usually stores contiguous rows in cache, effectively utilizing memory and cache.

3. Tiling: accessing a tile of rows and columns at a time significantly increases the performance, as even if we added more for loops , the outer loops were able to loop through the tiles faster. We set the tile size to 5 as we found that size gave the optimal results.

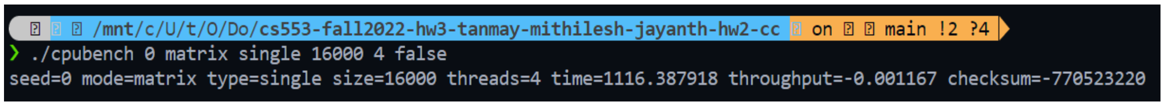
4. Compiler flags: we used the compiler optimization flag “-O3”, to instruct the compiler to optimize the code when compiling to produce a highly optimized executable.

The following techniques were used to achieve parallelisation for flops:

1. Pthreads: we used pthreads to divide the outer for loops, effectively dividing the size between the different threads using the divider.

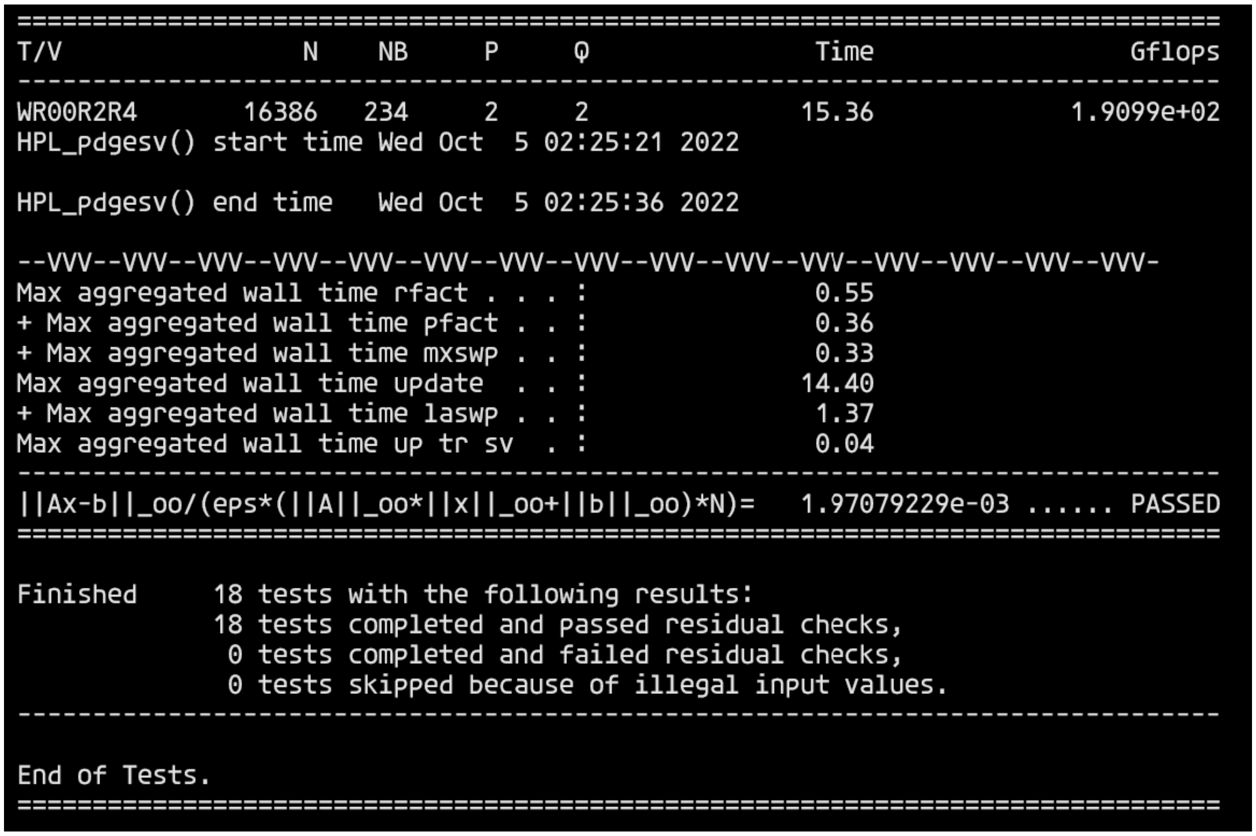
Summary of the experimentations-

Observing the throughput we can infer that there is optimal parallelization with an increased number of threads.The computation work is being effectively divided between the threads.We Started for 2 hours execution time for running 16000 X 16000 single matrix on 4 cores to 18 mins on 4 cores.



```
/mnt/c/U/t/O/Do/cs553-fall2022-hw3-tanmay-mithilesh-jayanth-hw2-cc  on  main !2 ?4
❯ ./cpubench 0 matrix single 16000 4 false
seed=0 mode=matrix type=single size=16000 threads=4 time=1116.387918 throughput=-0.001167 checksum=-770523220
```

For Linpack benchmark-
We tried to run the linpack benchmark by building mhpl 2.3 library.We encountered various errors due to missing intel openAPI.We installed Intel's Base Toolkit and HPC Toolkit for building xhpl.We were not able to build shpl as we were unable to find build files for that online.After installation we tried benchmarking many times by changing .dat file for xhpl to get the most optimized benchmarks.This is the best result that we were able to get from running matrix of size 16386 on 4 cores on an Intel Core i7 processor.



```
================================================================================
T/V                N    NB     P     Q               Time                 Gflops
--------------------------------------------------------------------------------
WR00R2R4       16386   234     2     2              15.36             1.9099e+02
HPL_pdgesv() start time Wed Oct  5 02:25:21 2022

HPL_pdgesv() end time   Wed Oct  5 02:25:36 2022

--VVV--VVV--VVV--VVV--VVV--VVV--VVV--VVV--VVV--VVV--VVV--VVV--VVV--VVV--VVV-
Max aggregated wall time rfact . . . :               0.55
+ Max aggregated wall time pfact . . :               0.36
+ Max aggregated wall time mxswp . . :               0.33
Max aggregated wall time update  . . :              14.40
+ Max aggregated wall time laswp . . :               1.37
Max aggregated wall time up tr sv  . :               0.04
--------------------------------------------------------------------------------
||Ax-b||_oo/(eps*(||A||_oo*||x||_oo+||b||_oo)*N)=   1.97079229e-03 ...... PASSED
================================================================================

Finished     18 tests with the following results:
             18 tests completed and passed residual checks,
              0 tests completed and failed residual checks,
              0 tests skipped because of illegal input values.
--------------------------------------------------------------------------------

End of Tests.
================================================================================
```