1.

```
sit315-001@sit315-001-VirtualBox:~$ time ./sequential_matrix_multiply
Result matrix (sample):
328350 323400 318450 313500 308550
333300 328250 323200 318150 313100
338250 333100 327950 322800 317650
343200 337950 332700 327450 322200
348150 342800 337450 332100 326750

real    0m0.007s
user    0m0.005s
sys     0m0.002s
```

```
sit315-001@sit315-001-VirtualBox:~$ time mpirun -np 4 --hostfile ~/hostfile ./mpi_matrix_multiply
Authorization required, but no authorization protocol specified

Authorization required, but no authorization protocol specified

Authorization required, but no authorization protocol specified

Authorization required, but no authorization protocol specified

Authorization required, but no authorization protocol specified

Authorization required, but no authorization protocol specified

Authorization required, but no authorization protocol specified

Authorization required, but no authorization protocol specified

Result matrix (sample):
328350 323400 318450 313500 308550
333300 328250 323200 318150 313100
338250 333100 327950 322800 317650
343200 337950 332700 327450 322200
348150 342800 337450 332100 326750

real    0m0.825s
user    0m0.098s
sys     0m0.141s
```

Time

1. Sequential Version (sequential_matrix_multiply):
real: 0m0.007s
user: 0m0.005s
sys: 0m0.002s

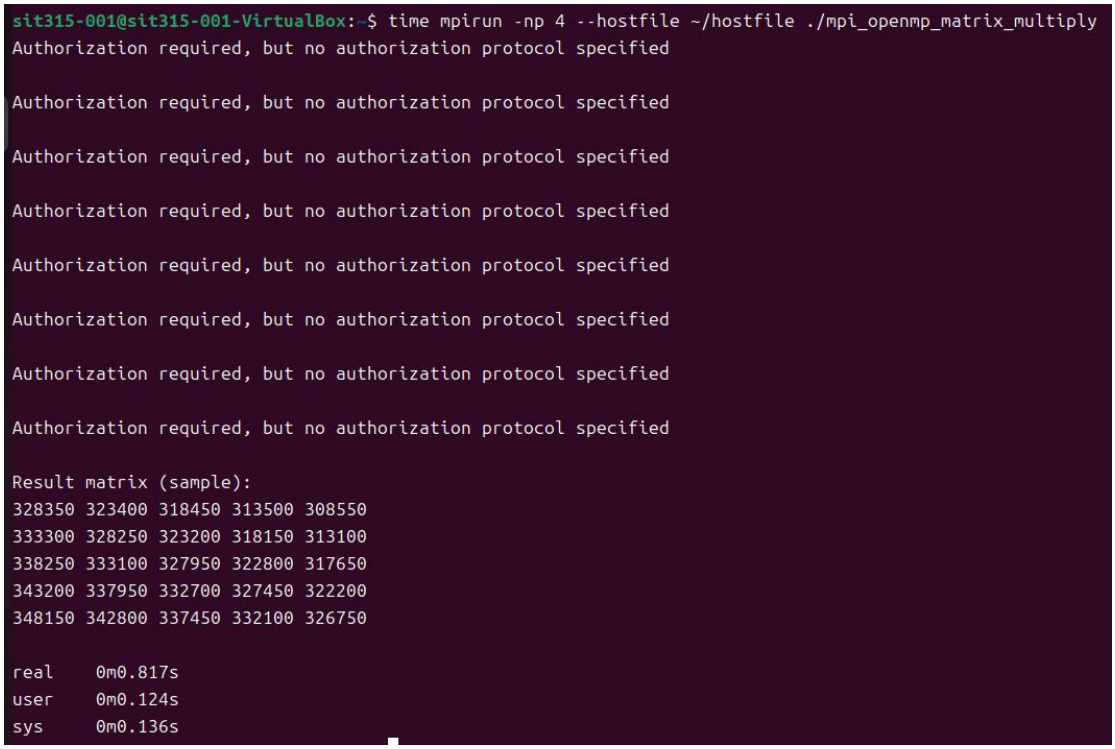
2. MPI Version (mpi_matrix_multiply):
real: 0m0.825s
user: 0m0.098s
sys: 0m0.141s

Conclusion

Sequential Program: The sequential method is fast for small-scale operations due to minimal overhead. However, this method only uses a single CPU core and will not scale efficiently for larger matrices or more complex operations. Its real-time execution is the shortest, but it's limited to small data sizes and lacks scalability.

MPI Programs: MPI introduces parallelization by distributing tasks across multiple processes, but this comes at the cost of inter-process communication overhead. While it makes sense for distributed systems with large data sizes, the observed real-time execution is much higher than the sequential approach due to this overhead.

2.



Time

3. MPI + OpenMP Version (mpi_openmp_matrix_multiply):

real: 0m0.817s

user: 0m0.124s

sys: 0m0.136s

Conclusion

MPI + OpenMP Program: Adds multi-threading but provides minimal additional benefit for small-scale tasks. The overhead from both MPI communication and thread management limits its effectiveness at small sizes, making it no better than MPI alone for smaller problems.

3.

```
sit315-001@sit315-001-VirtualBox:~$ g++ opencl_matrix_multiply.cpp -o opencl_matrix_multiply -lOpenCL
sit315-001@sit315-001-VirtualBox:~$ time ./opencl_matrix_multiply
Result matrix (sample):
32835000.000000 32839940.000000 32844900.000000 32849860.000000 32854800.000000
82334992.000000 82349928.000000 82364928.000000 82379864.000000 82394800.000000
131834944.000000 131859928.000000 131884936.000000 131909880.000000 131934800.000000
181334928.000000 181369936.000000 181404944.000000 181439888.000000 181474800.000000
230834912.000000 230879904.000000 230924944.000000 230969872.000000 231014800.000000

real    0m0.101s
user    0m0.051s
sys     0m0.046s
```

Time

real: 0m0.101s
user: 0m0.051s
sys: 0m0.046s

Conclusion

OpenCL Program: OpenCL shows a dramatic reduction in execution time compared to the other methods. Leveraging GPU acceleration for parallel computation of matrix elements results in a significant performance gain. The real-time execution is the lowest of all methods, highlighting OpenCL's suitability for matrix operations where the workload can be parallelized efficiently across a large number of GPU threads.