National University of Computer & Emerging Sciences Karachi Campus

Project Report

COURSE: Parallel & Distributed Computing

Instructor: Muhammad Danish Khan

Section: 5B

Group Members:

Maryam Hussain 19K-0286
Mohammad Usama 20K-0190
Fabiha Atique 20K-0369

Project Title:

SERIAL VS PARALLEL COMPUTING OF MATRIX MULTIPLICATION (IN OPENMP AND MPI)

Project Objective:

To identify the matrices that are multiplicities in fewest time complexity and less computations.

Introduction and brief description:

The project will identify the matrices that multiplicities in fewest time complexity and less computations. The computational complexity of matrix multiplication determines how rapidly it may be executed. Matrix multiplication algorithms are a key subroutine in theoretical and numerical algorithms for numerical linear algebra and optimization, therefore determining the optimal time to do them is crucial.

Technologies Used:

- ➔ C++ language
- ➔ OpenMP
- ➔ MPI
- ➔ P Threads
- ➔ filling
- ➔ Ubuntu
- ➔ Google Colab notebook for graph representation https://colab.research.google.com/drive/1UY6nFl2cOwkoUuTpal1fh6yBmlLbev0T?usp=sharing

Methodology:

Number of square matrices will be generated by the system using the randomSquareMatrix function defined in code and stored in a single matrix variable. For sequential the traditional method will be followed i.e (step by step multiplying rows and columns), both the matrix and the final result will be saved in the file. Parallel for loop will be using in OpenMP and within #pragma the local results with respective thread ID will be saved in separate file “ParallelMultiplyTestLocalResult”, plus both the matrix and the final outcome will be saved in “ParallelMultiplyTest”. In Optimized Parallel Multiplication first we convert the 2 dimensional matrix into 1 dimensional matrix and then multiply both, the local results and results will be saved as done in Parallel Multiplication. In the MPI Matrix Multiplication code, the master task distributes a matrix multiply operation to numtasks-1 worker tasks.

Matrices to be multiplied have random values. Computation time of the multiplication is ten stored in a file.

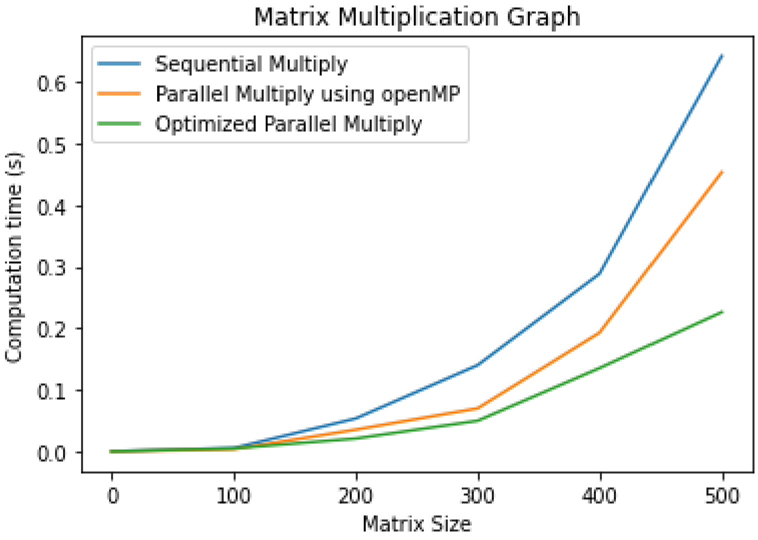
Optimized Parallel vs. Parallel vs. Sequential execution with different matrix sizes:

Execution time analysis of **optimized** parallel matrix multiplication program & **parallel** matrix multiplication program in OMP and **sequential** matrix multiplication program without OMP for different matrix sizes was observed.

During project output matrix size was varied with values of matrix size 100*100, 200*200, 300*300, 400*400 and 500*500. In the parallel program the number of threads was kept constant.

Optimized parallel program took less of execution time than parallel program and sequential program. Sequential program took significantly larger execution time. It was observed that the parallel version in OMP is beneficial if the problem size is significantly large, otherwise the benefit of parallel processing cannot be achieved.

Output matrix size	Execution time (seconds) with OMP		Execution time (seconds) without OMP
	Optimized Parallel Multiply	Parallel Multiply	Sequential Multiply
100*100	0.004846	0.003437	0.005421
200*200	0.020878	0.035357	0.053466
300*300	0.049719	0.069972	0.140142
400*400	0.135349	0.192909	0.288580
500*500	0.226305	0.453744	0.642640

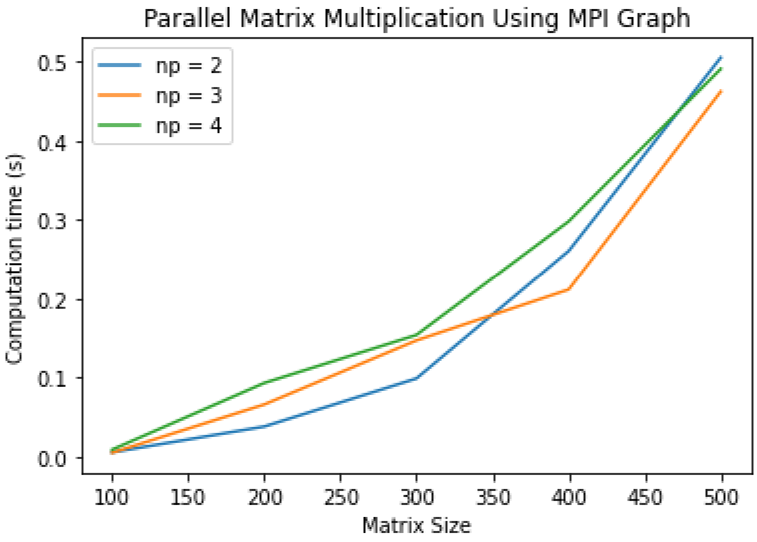


Matrix Multiplication in MPI with different matrix sizes and processors

Execution time analysis of matrix multiplication in MPI for different matrix sizes and number of processors was observed. During project output matrix size was varied with values of output matrix size 100*100, 200*200,300*300,400*400 and 500*500.Number of processors (np) were kept as 2, 3 and 4. Increasing the number of processors results in increased execution time.

The program was run on a computer with a processor: Intel(R) Core(TM) i3-1005G1 CPU @ 1.20GHz 1.19 MHz, Memory 12GB RAM, Operating System –Ubuntu 18.04, 64-bit.

Output matrix size	Execution time (seconds) in MPI		
	np=2	**np=3**	**np=4**
100*100	0.005448	0.004940	0.008853
200*200	0.038085	0.065992	0.093184
300*300	0.098921	0.147049	0.153959
400*400	0.260584	0.211529	0.298154
500*500	0.505458	0.462535	0.491139



What would be the Source Data?

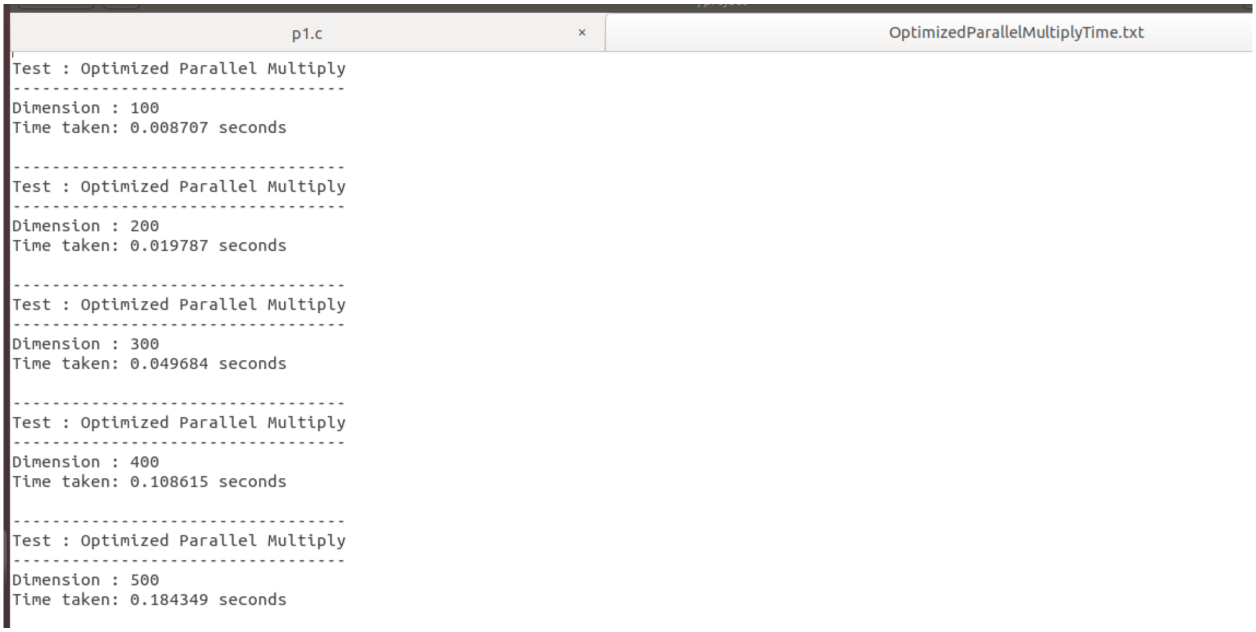
Matrix data values will be assigned randomly.

Applications of Serial & Parallel Matrix Multiplication:

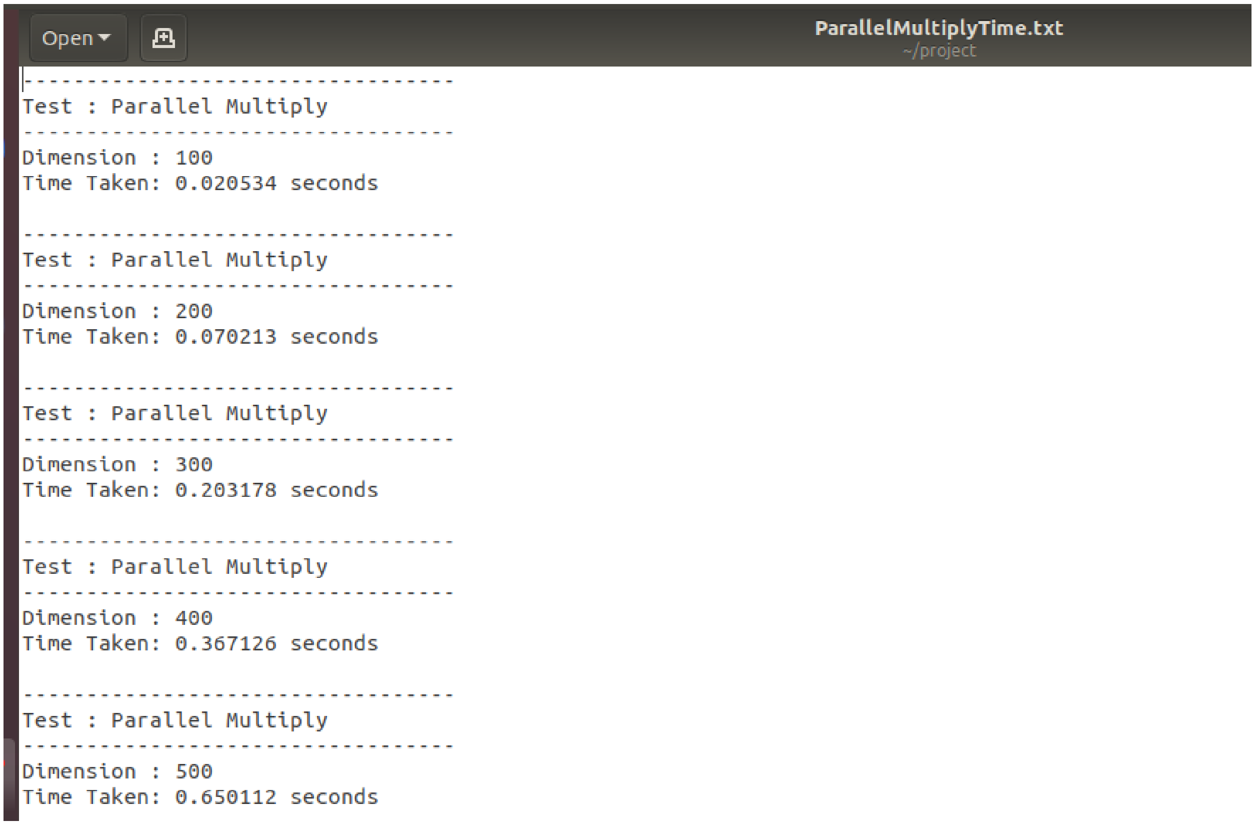
- It is used in optic science to account for refraction and reflection.
- It helps in solving linear equations
- It is used widely in such areas as network theory, solution of linear systems of equations, transformation of coordinate systems, and population modeling.
- Graph Theory
- Numerical Algorithms
- Signal Processing
- Digital Control

Conclusion

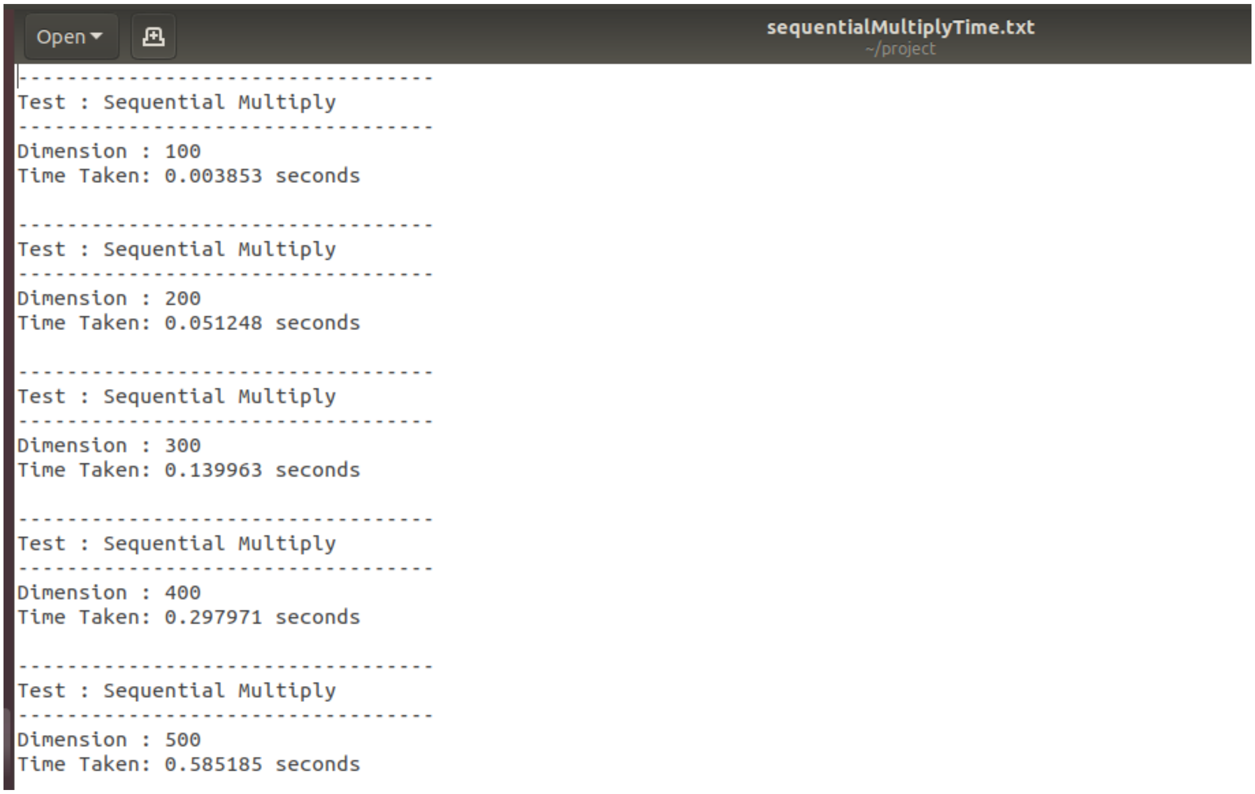
The parallel programming in OMP is beneficial only when the input problem size is significantly larger. For smaller size problems, it is better to go with sequential programming. OMP encourages the parallel execution of the program and efficiently utilizes the multi-core processors in the present generation CPUs. The work discussed in the project does not observe fully the pattern of execution times with high values of matrix size. It is due to the unavailability of the high configuration machines. We would like to extend the analysis with various multiprocessor machines, such as having 8-cores and 16-cores on voluminous data.



```
Test : Optimized Parallel Multiply
----------------------------------
Dimension : 100
Time taken: 0.008707 seconds

----------------------------------
Test : Optimized Parallel Multiply
----------------------------------
Dimension : 200
Time taken: 0.019787 seconds

----------------------------------
Test : Optimized Parallel Multiply
----------------------------------
Dimension : 300
Time taken: 0.049684 seconds

----------------------------------
Test : Optimized Parallel Multiply
----------------------------------
Dimension : 400
Time taken: 0.108615 seconds

----------------------------------
Test : Optimized Parallel Multiply
----------------------------------
Dimension : 500
Time taken: 0.184349 seconds
```

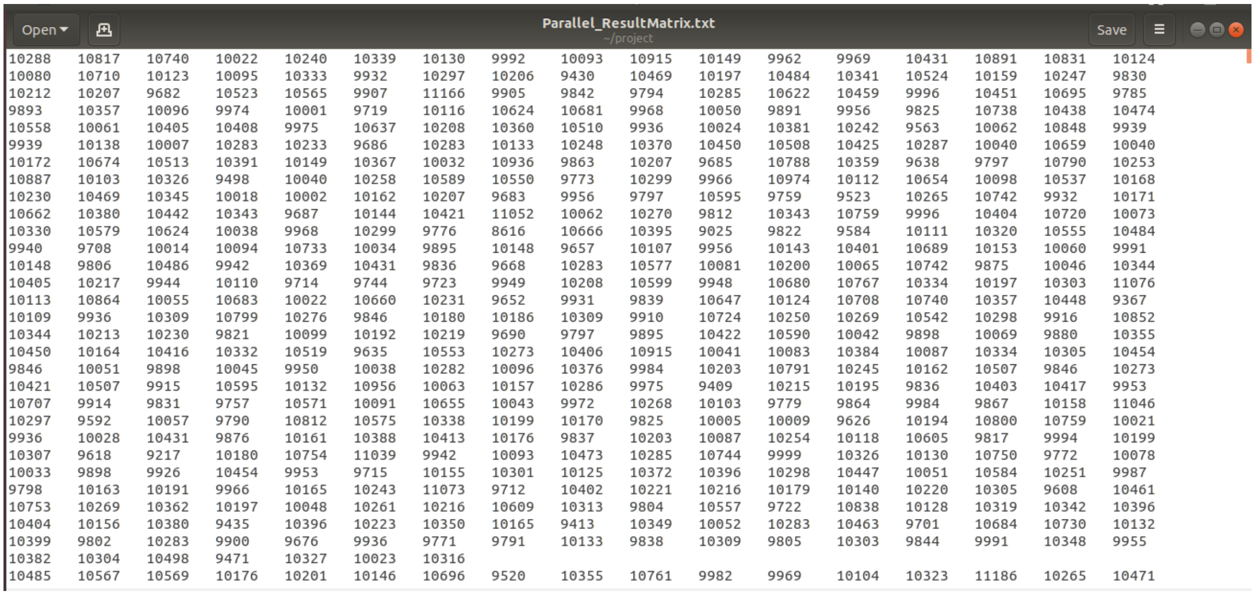


```
----------------------------------
Test : Parallel Multiply
----------------------------------
Dimension : 100
Time Taken: 0.020534 seconds

----------------------------------
Test : Parallel Multiply
----------------------------------
Dimension : 200
Time Taken: 0.070213 seconds

----------------------------------
Test : Parallel Multiply
----------------------------------
Dimension : 300
Time Taken: 0.203178 seconds

----------------------------------
Test : Parallel Multiply
----------------------------------
Dimension : 400
Time Taken: 0.367126 seconds

----------------------------------
Test : Parallel Multiply
----------------------------------
Dimension : 500
Time Taken: 0.650112 seconds
```

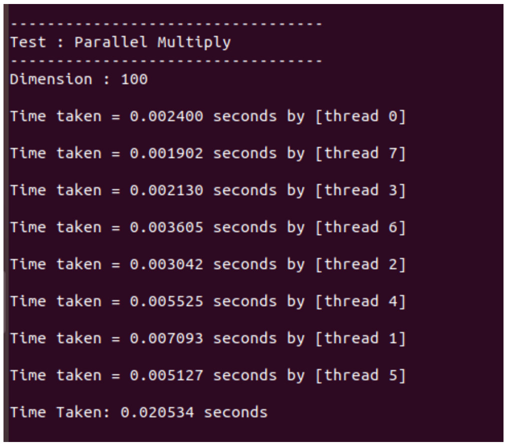


```
----------------------------------
Test : Sequential Multiply
----------------------------------
Dimension : 100
Time Taken: 0.003853 seconds

----------------------------------
Test : Sequential Multiply
----------------------------------
Dimension : 200
Time Taken: 0.051248 seconds

----------------------------------
Test : Sequential Multiply
----------------------------------
Dimension : 300
Time Taken: 0.139963 seconds

----------------------------------
Test : Sequential Multiply
----------------------------------
Dimension : 400
Time Taken: 0.297971 seconds

----------------------------------
Test : Sequential Multiply
----------------------------------
Dimension : 500
Time Taken: 0.585185 seconds
```

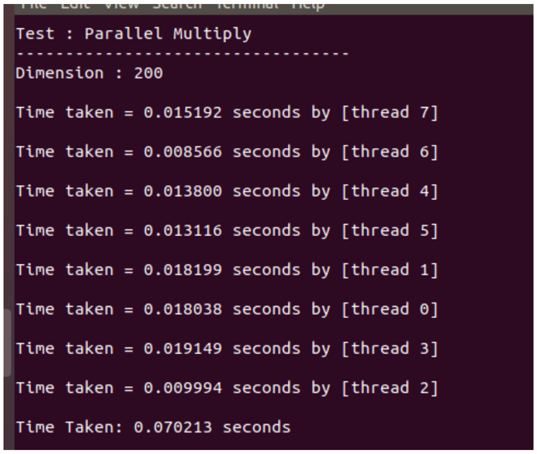
Resultant Matrix Using Parallel Multiplication using OpenMP:



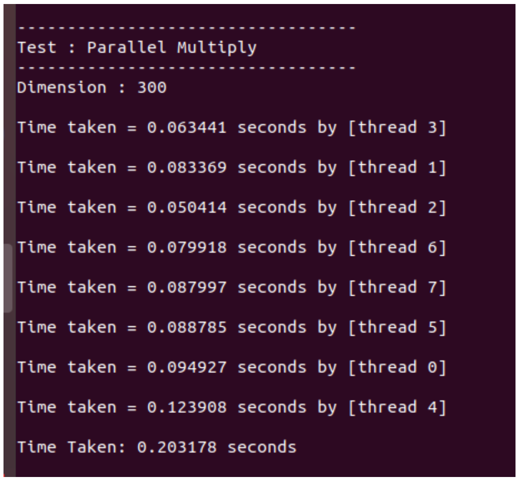
```
10288   10817   10740   10022   10240   10339   10130   9992    10093   10915   10149   9962    9969    10431   10891   10831   10124
10080   10710   10123   10095   10333   9932    10297   10206   9430    10469   10197   10484   10341   10524   10159   10247   9830
10212   10207   9682    10523   10565   9907    11166   9905    9842    9794    10285   10622   10459   9996    10451   10695   9785
9893    10357   10096   9974    10001   9719    10116   10624   10681   9968    10050   9891    9956    9825    10738   10438   10474
10558   10061   10405   10408   9975    10637   10208   10360   10510   9936    10024   10381   10242   9563    10062   10848   9939
9939    10138   10007   10283   10233   9686    10283   10133   10248   10370   10450   10508   10425   10287   10040   10659   10040
10172   10674   10513   10391   10149   10367   10032   10936   9863    10207   9685    10788   10359   9638    9797    10790   10253
10887   10103   10326   9498    10040   10258   10589   10550   9773    10299   9966    10974   10112   10654   10098   10537   10168
10230   10469   10345   10018   10002   10162   10207   9683    9956    9797    10595   9759    9523    10265   10742   9932    10171
10662   10380   10442   10343   9687    10144   10421   11052   10062   10270   9812    10343   10759   9996    10404   10720   10073
10330   10579   10624   10038   9968    10299   9776    8616    10666   10395   9025    9822    9584    10111   10320   10555   10484
9940    9708    10014   10094   10733   10034   9895    10148   9657    10107   9956    10143   10401   10689   10153   10060   9991
10148   9806    10486   9942    10369   10431   9836    9668    10283   10577   10081   10200   10065   10742   9875    10046   10344
10405   10217   9944    10110   9714    9744    9723    9949    10208   10599   9948    10680   10767   10334   10197   10303   11076
10113   10864   10055   10683   10022   10660   10231   9652    9931    9839    10647   10124   10708   10740   10357   10448   9367
10109   9936    10309   10799   10276   9846    10180   10186   10309   9910    10724   10250   10269   10542   10298   9916    10852
10344   10213   10230   9821    10099   10192   10219   9690    9797    9895    10422   10590   10042   9898    10069   9880    10355
10450   10164   10416   10332   10519   9635    10553   10273   10406   10915   10041   10083   10384   10087   10334   10305   10454
9846    10051   9898    10045   9950    10038   10282   10096   10376   9984    10203   10791   10245   10162   10507   9846    10273
10421   10507   9915    10595   10132   10956   10063   10157   10286   9975    9409    10215   10195   9836    10403   10417   9953
10707   9914    9831    9757    10571   10091   10655   10043   9972    10268   10103   9779    9864    9984    9867    10158   11046
10297   9592    10057   9790    10812   10575   10338   10199   10170   9825    10005   10009   9626    10194   10800   10759   10021
9936    10028   10431   9876    10161   10388   10413   10176   9837    10203   10087   10254   10118   10605   9817    9994    10199
10307   9618    9217    10180   10754   11039   9942    10093   10473   10285   10744   9999    10326   10130   10750   9772    10078
10033   9898    9926    10454   9953    9715    10155   10301   10125   10372   10396   10298   10447   10051   10584   10251   9987
9798    10163   10191   9966    10165   10243   11073   9712    10402   10221   10216   10179   10140   10220   10305   9608    10461
10753   10269   10362   10197   10048   10261   10216   10609   10313   9804    10557   9722    10838   10128   10319   10342   10396
10404   10156   10380   9435    10396   10223   10350   10165   9413    10349   10052   10283   10463   9701    10684   10730   10132
10399   9802    10283   9900    9676    9936    9771    9791    10133   9838    10309   9805    10303   9844    9991    10348   9955
10382   10304   10498   9471    10327   10023   10316
10485   10567   10569   10176   10201   10146   10696   9520    10355   10761   9982    9969    10104   10323   11186   10265   10471
```



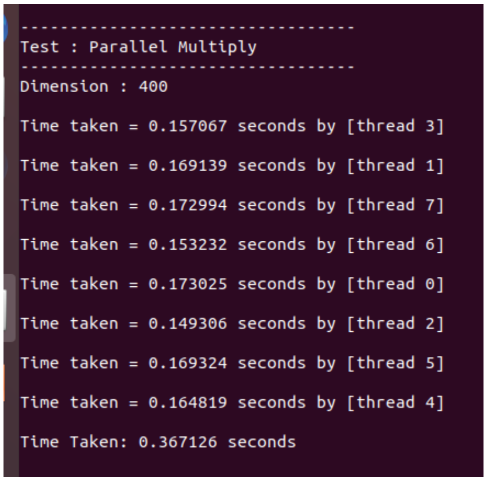
```
---------------------------------
Test : Parallel Multiply
---------------------------------
Dimension : 100

Time taken = 0.002400 seconds by [thread 0]

Time taken = 0.001902 seconds by [thread 7]

Time taken = 0.002130 seconds by [thread 3]

Time taken = 0.003605 seconds by [thread 6]

Time taken = 0.003042 seconds by [thread 2]

Time taken = 0.005525 seconds by [thread 4]

Time taken = 0.007093 seconds by [thread 1]

Time taken = 0.005127 seconds by [thread 5]

Time Taken: 0.020534 seconds
```



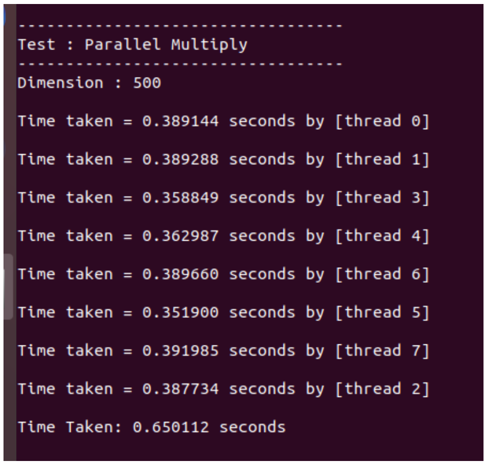
```
Test : Parallel Multiply
---------------------------------
Dimension : 200

Time taken = 0.015192 seconds by [thread 7]

Time taken = 0.008566 seconds by [thread 6]

Time taken = 0.013800 seconds by [thread 4]

Time taken = 0.013116 seconds by [thread 5]

Time taken = 0.018199 seconds by [thread 1]

Time taken = 0.018038 seconds by [thread 0]

Time taken = 0.019149 seconds by [thread 3]

Time taken = 0.009994 seconds by [thread 2]

Time Taken: 0.070213 seconds
```



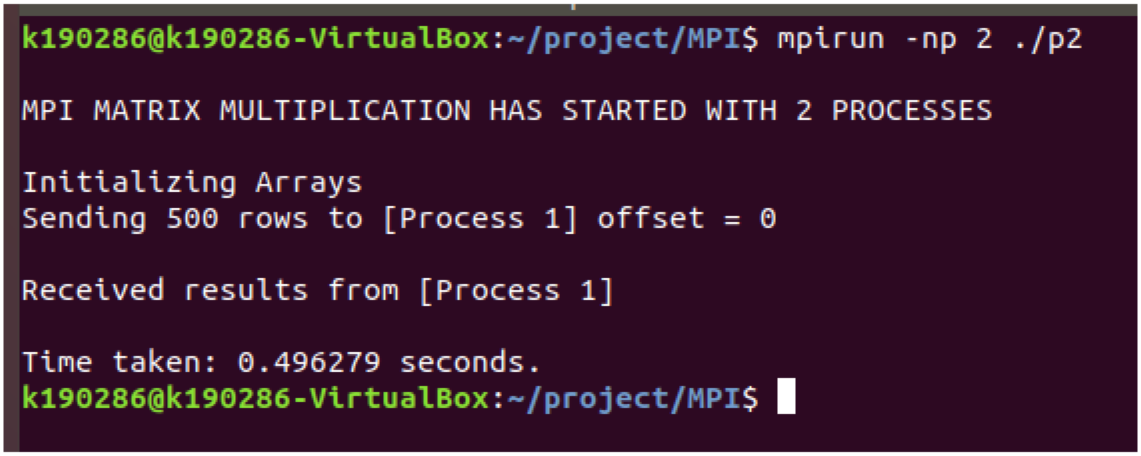
```
---------------------------------
Test : Parallel Multiply
---------------------------------
Dimension : 300

Time taken = 0.063441 seconds by [thread 3]

Time taken = 0.083369 seconds by [thread 1]

Time taken = 0.050414 seconds by [thread 2]

Time taken = 0.079918 seconds by [thread 6]

Time taken = 0.087997 seconds by [thread 7]

Time taken = 0.088785 seconds by [thread 5]

Time taken = 0.094927 seconds by [thread 0]

Time taken = 0.123908 seconds by [thread 4]

Time Taken: 0.203178 seconds
```



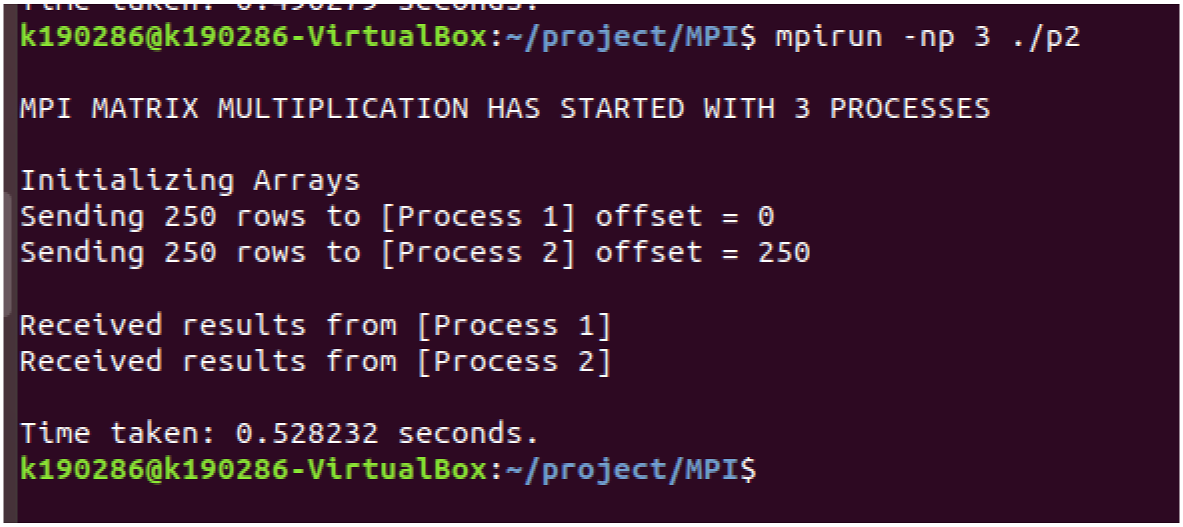
```
---------------------------------
Test : Parallel Multiply
---------------------------------
Dimension : 400

Time taken = 0.157067 seconds by [thread 3]

Time taken = 0.169139 seconds by [thread 1]

Time taken = 0.172994 seconds by [thread 7]

Time taken = 0.153232 seconds by [thread 6]

Time taken = 0.173025 seconds by [thread 0]

Time taken = 0.149306 seconds by [thread 2]

Time taken = 0.169324 seconds by [thread 5]

Time taken = 0.164819 seconds by [thread 4]

Time Taken: 0.367126 seconds
```



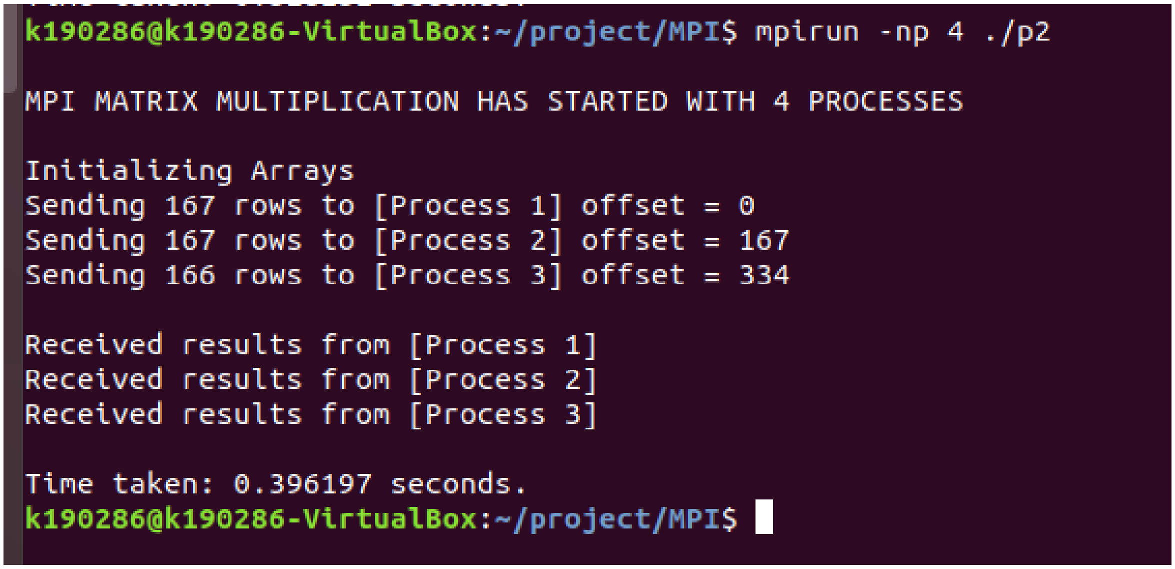
```
---------------------------------
Test : Parallel Multiply
---------------------------------
Dimension : 500

Time taken = 0.389144 seconds by [thread 0]

Time taken = 0.389288 seconds by [thread 1]

Time taken = 0.358849 seconds by [thread 3]

Time taken = 0.362987 seconds by [thread 4]

Time taken = 0.389660 seconds by [thread 6]

Time taken = 0.351900 seconds by [thread 5]

Time taken = 0.391985 seconds by [thread 7]

Time taken = 0.387734 seconds by [thread 2]

Time Taken: 0.650112 seconds
```



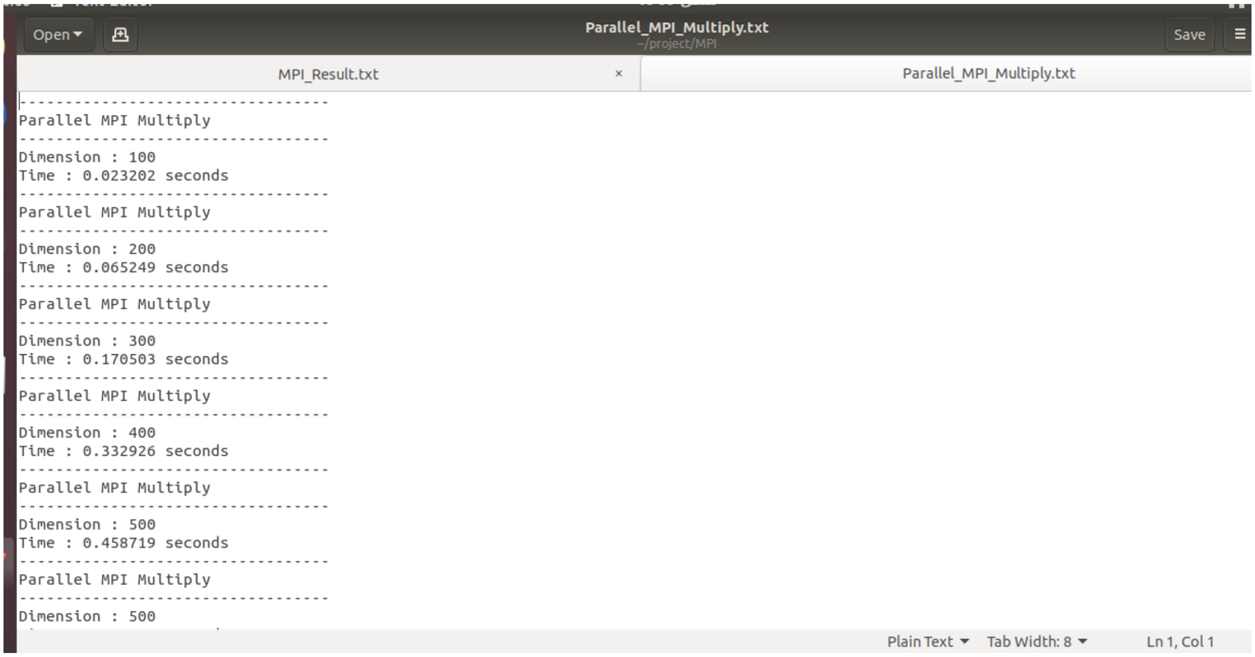
```
k190286@k190286-VirtualBox:~/project/MPI$ mpirun -np 2 ./p2

MPI MATRIX MULTIPLICATION HAS STARTED WITH 2 PROCESSES

Initializing Arrays
Sending 500 rows to [Process 1] offset = 0

Received results from [Process 1]

Time taken: 0.496279 seconds.
k190286@k190286-VirtualBox:~/project/MPI$
```



```
Time taken: 0.496279 seconds.
k190286@k190286-VirtualBox:~/project/MPI$ mpirun -np 3 ./p2

MPI MATRIX MULTIPLICATION HAS STARTED WITH 3 PROCESSES

Initializing Arrays
Sending 250 rows to [Process 1] offset = 0
Sending 250 rows to [Process 2] offset = 250

Received results from [Process 1]
Received results from [Process 2]

Time taken: 0.528232 seconds.
k190286@k190286-VirtualBox:~/project/MPI$
```

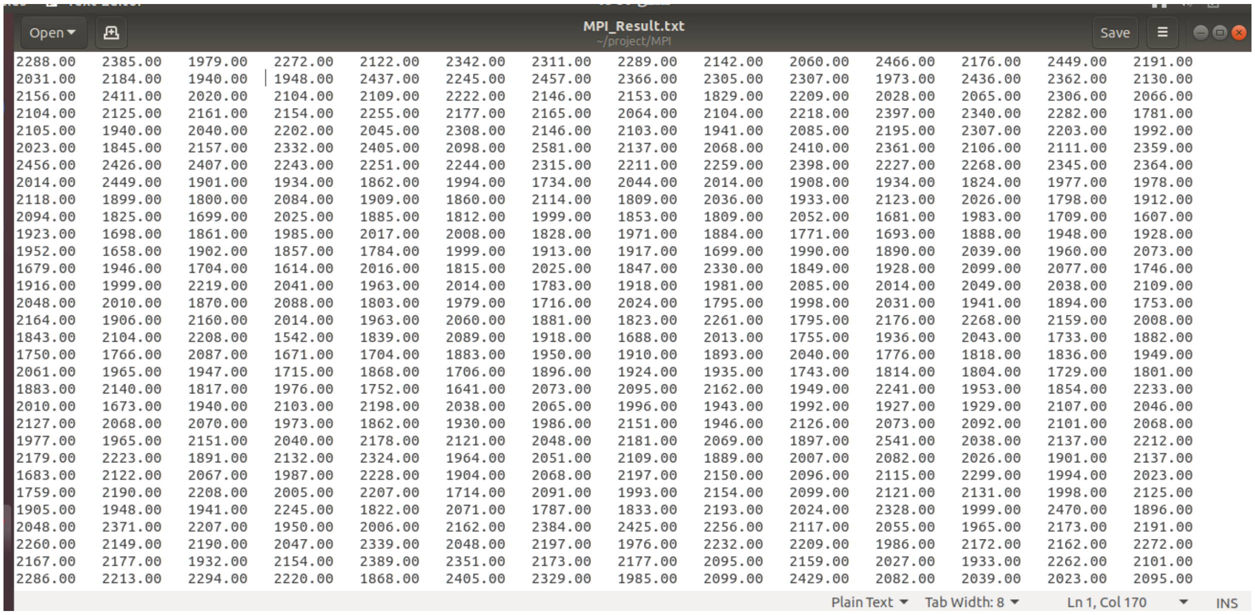


```
Time taken: 0.528232 seconds.
k190286@k190286-VirtualBox:~/project/MPI$ mpirun -np 4 ./p2

MPI MATRIX MULTIPLICATION HAS STARTED WITH 4 PROCESSES

Initializing Arrays
Sending 167 rows to [Process 1] offset = 0
Sending 167 rows to [Process 2] offset = 167
Sending 166 rows to [Process 3] offset = 334

Received results from [Process 1]
Received results from [Process 2]
Received results from [Process 3]

Time taken: 0.396197 seconds.
k190286@k190286-VirtualBox:~/project/MPI$
```



```
---------------------------------
Parallel MPI Multiply
---------------------------------
Dimension : 100
Time : 0.023202 seconds
---------------------------------
Parallel MPI Multiply
---------------------------------
Dimension : 200
Time : 0.065249 seconds
---------------------------------
Parallel MPI Multiply
---------------------------------
Dimension : 300
Time : 0.170503 seconds
---------------------------------
Parallel MPI Multiply
---------------------------------
Dimension : 400
Time : 0.332926 seconds
---------------------------------
Parallel MPI Multiply
---------------------------------
Dimension : 500
Time : 0.458719 seconds
---------------------------------
Parallel MPI Multiply
---------------------------------
Dimension : 500
```

Resultant Matrix Using Parallel Multiplication using MPI



```
2288.00    2385.00    1979.00    2272.00    2122.00    2342.00    2311.00    2289.00    2142.00    2060.00    2466.00    2176.00    2449.00    2191.00
2031.00    2184.00    1940.00    1948.00    2437.00    2245.00    2457.00    2366.00    2305.00    2307.00    1973.00    2436.00    2362.00    2130.00
2156.00    2411.00    2020.00    2104.00    2109.00    2222.00    2146.00    2153.00    1829.00    2209.00    2028.00    2065.00    2306.00    2066.00
2104.00    2125.00    2161.00    2154.00    2255.00    2177.00    2165.00    2064.00    2104.00    2218.00    2397.00    2340.00    2282.00    1781.00
2105.00    1940.00    2040.00    2202.00    2045.00    2308.00    2146.00    2103.00    1941.00    2085.00    2195.00    2307.00    2203.00    1992.00
2023.00    1845.00    2157.00    2332.00    2405.00    2098.00    2581.00    2137.00    2068.00    2410.00    2361.00    2106.00    2111.00    2359.00
2456.00    2426.00    2407.00    2243.00    2251.00    2244.00    2315.00    2211.00    2259.00    2398.00    2227.00    2268.00    2345.00    2364.00
2014.00    2449.00    1901.00    1934.00    1862.00    1994.00    1734.00    2044.00    2014.00    1908.00    1934.00    1824.00    1977.00    1978.00
2118.00    1899.00    1800.00    2084.00    1909.00    1860.00    2114.00    1809.00    2036.00    1933.00    2123.00    2026.00    1798.00    1912.00
2094.00    1825.00    1699.00    2025.00    1885.00    1812.00    1999.00    1853.00    1809.00    2052.00    1681.00    1983.00    1709.00    1607.00
1923.00    1698.00    1861.00    1985.00    2017.00    2008.00    1828.00    1971.00    1884.00    1771.00    1693.00    1888.00    1948.00    1928.00
1952.00    1658.00    1902.00    1857.00    1784.00    1999.00    1913.00    1917.00    1699.00    1990.00    1890.00    2039.00    1960.00    2073.00
1679.00    1946.00    1704.00    1614.00    2016.00    1815.00    2025.00    1847.00    2330.00    1849.00    1928.00    2099.00    2077.00    1746.00
1916.00    1999.00    2219.00    2041.00    1963.00    2014.00    1783.00    1918.00    1981.00    2085.00    2014.00    2049.00    2038.00    2109.00
2048.00    2010.00    1870.00    2088.00    1803.00    1979.00    1716.00    2024.00    1795.00    1998.00    2031.00    1941.00    1894.00    1753.00
2164.00    1906.00    2160.00    2014.00    1963.00    2060.00    1881.00    1823.00    2261.00    1795.00    2176.00    2268.00    2159.00    2008.00
1843.00    2104.00    2208.00    1542.00    1839.00    2089.00    1918.00    1688.00    2013.00    1755.00    1936.00    2043.00    1733.00    1882.00
1750.00    1766.00    2087.00    1671.00    1704.00    1883.00    1950.00    1910.00    1893.00    2040.00    1776.00    1818.00    1836.00    1949.00
2061.00    1965.00    1947.00    1715.00    1868.00    1706.00    1896.00    1924.00    1935.00    1743.00    1814.00    1804.00    1729.00    1801.00
1883.00    2140.00    1817.00    1976.00    1752.00    1641.00    2073.00    2095.00    2162.00    1949.00    2241.00    1953.00    1854.00    2233.00
2010.00    1673.00    1940.00    2103.00    2198.00    2038.00    2065.00    1996.00    1943.00    1992.00    1927.00    1929.00    2107.00    2046.00
2127.00    2068.00    2070.00    1973.00    1862.00    1930.00    1986.00    2151.00    1946.00    2126.00    2073.00    2092.00    2101.00    2068.00
1977.00    1965.00    2151.00    2040.00    2178.00    2121.00    2048.00    2181.00    2069.00    1897.00    2541.00    2038.00    2137.00    2212.00
2179.00    2223.00    1891.00    2132.00    2324.00    1964.00    2051.00    2109.00    1889.00    2007.00    2082.00    2026.00    1901.00    2137.00
1683.00    2122.00    2067.00    1987.00    2228.00    1904.00    2068.00    2197.00    2150.00    2096.00    2115.00    2299.00    1994.00    2023.00
1759.00    2190.00    2208.00    2005.00    2207.00    1714.00    2091.00    1993.00    2154.00    2099.00    2121.00    2131.00    1998.00    2125.00
1905.00    1948.00    1941.00    2245.00    1822.00    2071.00    1787.00    1833.00    2193.00    2024.00    2328.00    1999.00    2470.00    1896.00
2048.00    2371.00    2207.00    1950.00    2006.00    2162.00    2384.00    2425.00    2256.00    2117.00    2055.00    1965.00    2173.00    2191.00
2260.00    2149.00    2190.00    2047.00    2339.00    2048.00    2197.00    1976.00    2232.00    2209.00    1986.00    2172.00    2162.00    2272.00
2167.00    2177.00    1932.00    2154.00    2389.00    2351.00    2173.00    2177.00    2095.00    2159.00    2027.00    1933.00    2262.00    2101.00
2286.00    2213.00    2294.00    2220.00    1868.00    2405.00    2329.00    1985.00    2099.00    2429.00    2082.00    2039.00    2023.00    2095.00
```