Large Matrix Multiplication

Sequential and Parallel Programming

Group 8

Atul Singh	(216100191)
Onkar Jadhav	(216100299)
Ranjith Arahatholalu Nandish	(216100180)
Sudhanva Kusuma Chandrashekhara	(216100181)
Ujjwal Verma	(216100297)

Objective

- To multiply large random matrices.
- To compare the performance of sequential and Parallel programs in wall clock time.
- To generate Heatmap for the generated large random matrix.

Matrix Multiplication Algorithm

- A Matrix of n×m order multiplied with B matrix of m×p results in C matrix of n×p order.[1]

- For i from 1 to n:
  - For j from 1 to m:
    - Let C ij = 0
    - For k from 1 to m:
      - Set C= C + A ik * B kj
      - Set C = sum
- Return C

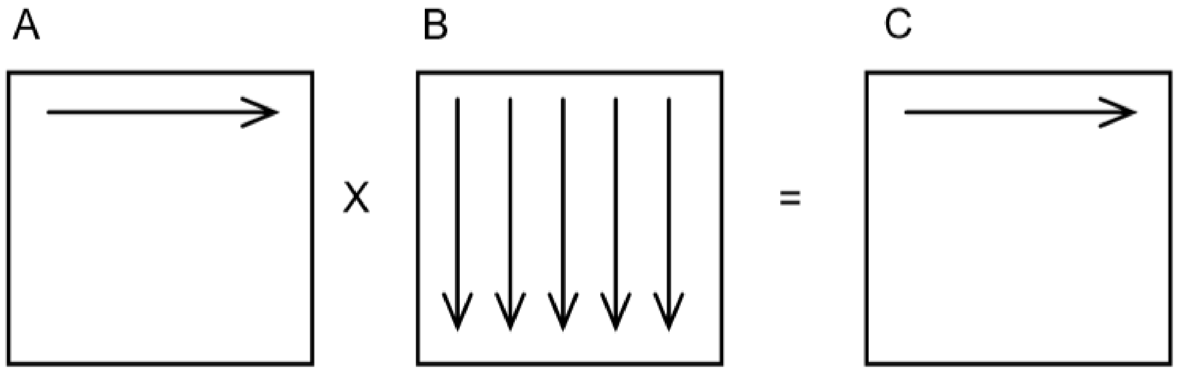


Fig 1. Matrix Multiplication Algorithm [*]

[*]"Parallel Programming Application: Matrix Multiplication", 2009, Available: http://training.uhem.itu.edu.tr/docs/20haziranmpi/matmul1.pdf.

Parallelization for Matrix multiplication

- Parallel Computing is, in general achieved by leveraging the usage of multiple processing units available by sharing the work as required between them.

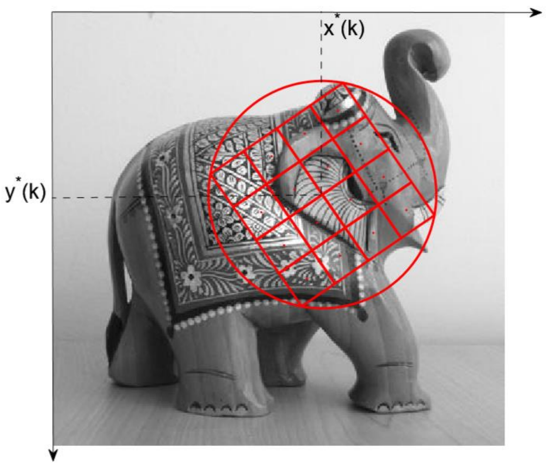


Fig 2. Image Processing[*]

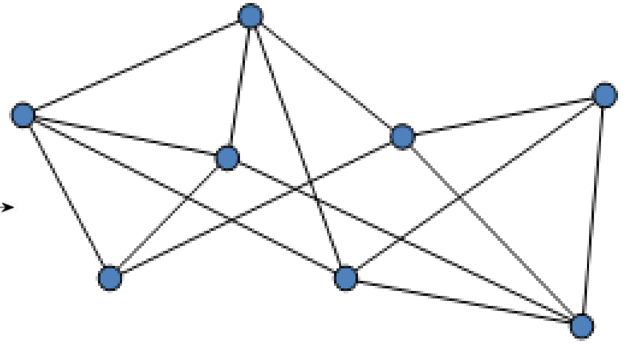


Fig 3. Graph Theory[*]

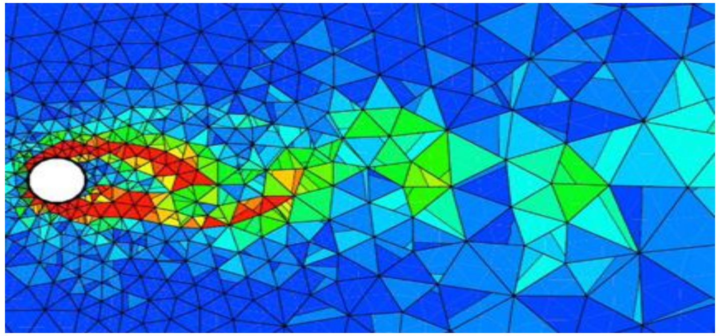


Fig 4. CFD and FEM [^]

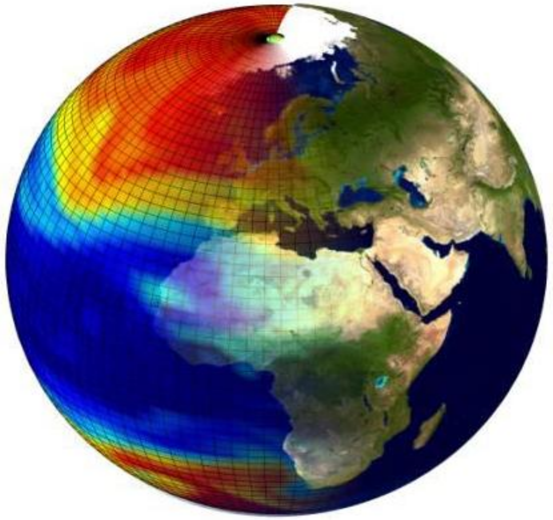


Fig 5. Atmospheric Modeling[#]

Sequential vs Parallel Programming

Sequential programming	Parallel programming
Single core data processing.	Multicore data processing.
Large data execution is more time consuming.[2]	For Large data applications, time consumption is less.[2]
Not feasible for complex application.[3]	Well suited for complex applications.[3]

OpenMP

- OpenMP provides specific API's for writing shared memory parallel programs.
- Incremental approach helps programmer to write the same existing program with parallel directive extensions within them.[4]
- OpenMP uses a fork-join model of parallel execution.[5]

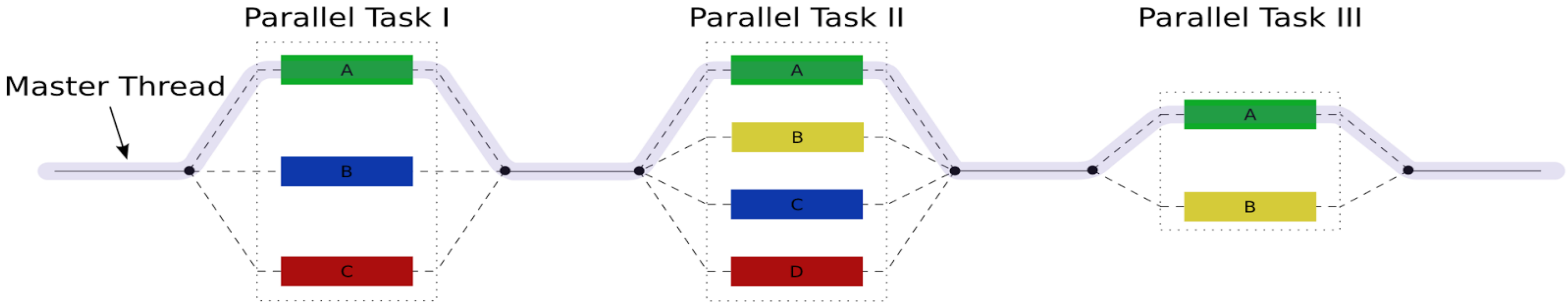


Fig 6. Fork-Join Model [*]

[*]"Fork–join model", *en.wikipedia.org*. [Online]. Available: https://en.wikipedia.org/wiki/Fork%E2%80%93join_model#/media/File:Fork_join.svg.

OpenMP

Advantages	**Disadvantages**
Utilize power of multiple processors to solve problems quickly.	Risk of introducing difficult to debug synchronization bugs and race conditions.[5]
OpenMP provides a portable standard parallel API specifically for programming shared memory multiprocessors.[6]	High chance of false sharing (Multiple shared variable lying in the cache line affecting the performance of code).[6]

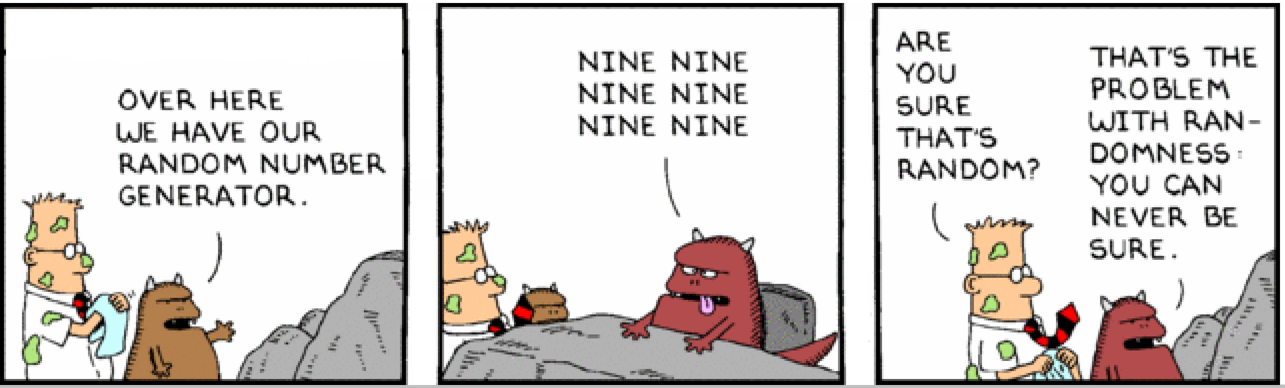
Random Numbers

A random number is a number generated by a process, whose outcome is unpredictable, and which cannot be subsequently reproduced.[7]

- Properties:
  - The numbers are equally probable and unique.[8]
- Applications:
  - In Aerospace and Electronic industries as LED for Radars.
  - Cryptography.
  - In Numerical analysis to describe computation errors.[9]
  - Nuclear Physics.[10]

PRNG - Algorithm

- PRNG :- Generates an uniformly distributed probabilistic random number sequence.
- Seed: This sets a particular value to the random function every time when the seed is called.[11]



A Random number (rand( )) divided by a inbuilt RAND_MAX number

```
double randomgenerator( )
{           double x;
            x=rand( )/double(RAND_MAX);
            return x;
}
double randomgenerator(double a, double b)
{           double t;
            t=(b-a)*randomgenerator( ) + a;
            return t;
}
```

Setting a Seed

Overloaded function to generate random number within a range

```
int main ( )
{           srand(101);
            double x = randomgenerator(1.0,0.0);
}
```

Heat Maps

Graphically represents the individual values of a matrix as different colors taken in a hierarchy to form a pixelated image, to give a better sense of density of the contents of a matrix.[12]

Applications:

- Data Mining purposes, for instance displaying areas of more accessed pages.
- Weather Forecasts and surveys.
- Statistical Observations.

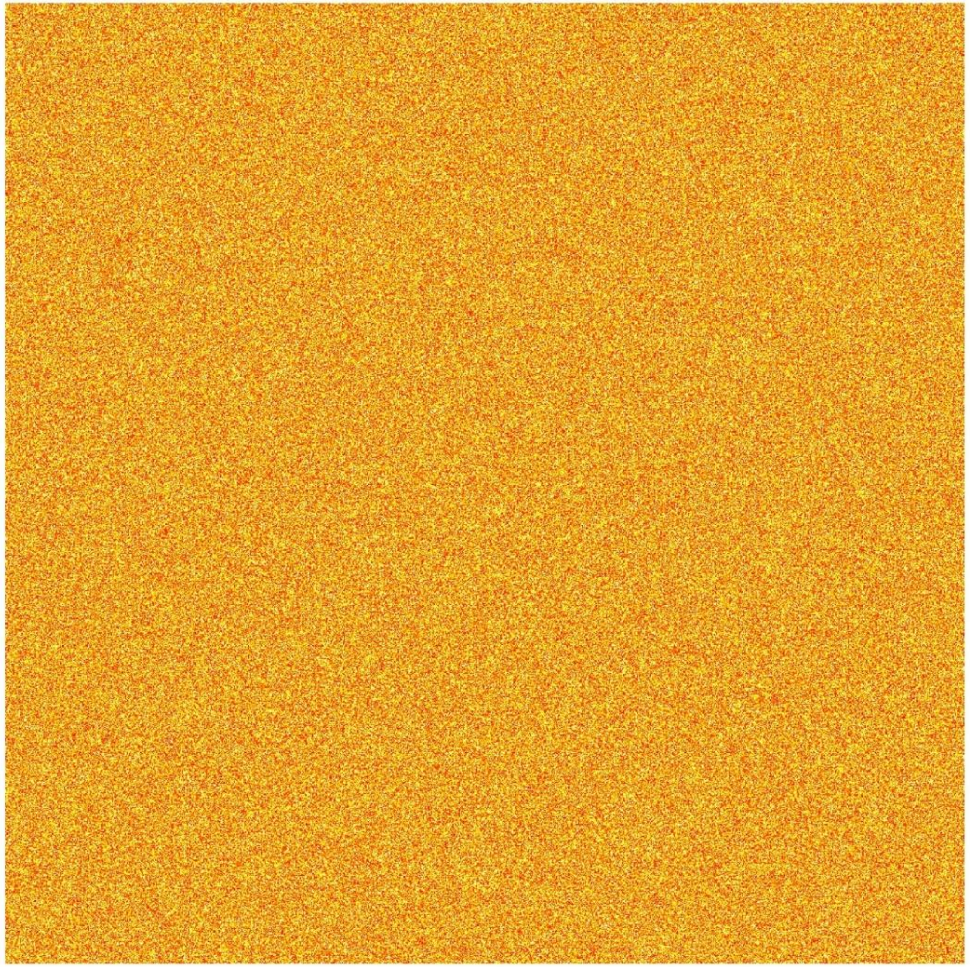


Fig 7. Heat Map generated in R

Parallel Matrix Multiplication

#pragma omp parallel starts a parallel region. All the threads execute the code.

a and b are shared among all threads

Each thread allocates a private copy of j and k from storage

```
# pragma omp parallel shared (a,b) private (j,k)
{
        # pragma omp for schedule(dynamic)
        for(i=0;i<n;i++)
        {
            for(j=0;j<n;j++)
            {
                multi[i][j]=0;
                for(k=0;k<n;k++)
                {
                    multi[i][j]=multi[i][j]+a[i][k]*b[k][j];
                }
            }
        }
```

Iterations are divided according to chunk (1 by default). So each thread has one iteration.[4]

Initializing the Multiplication matrix to zero

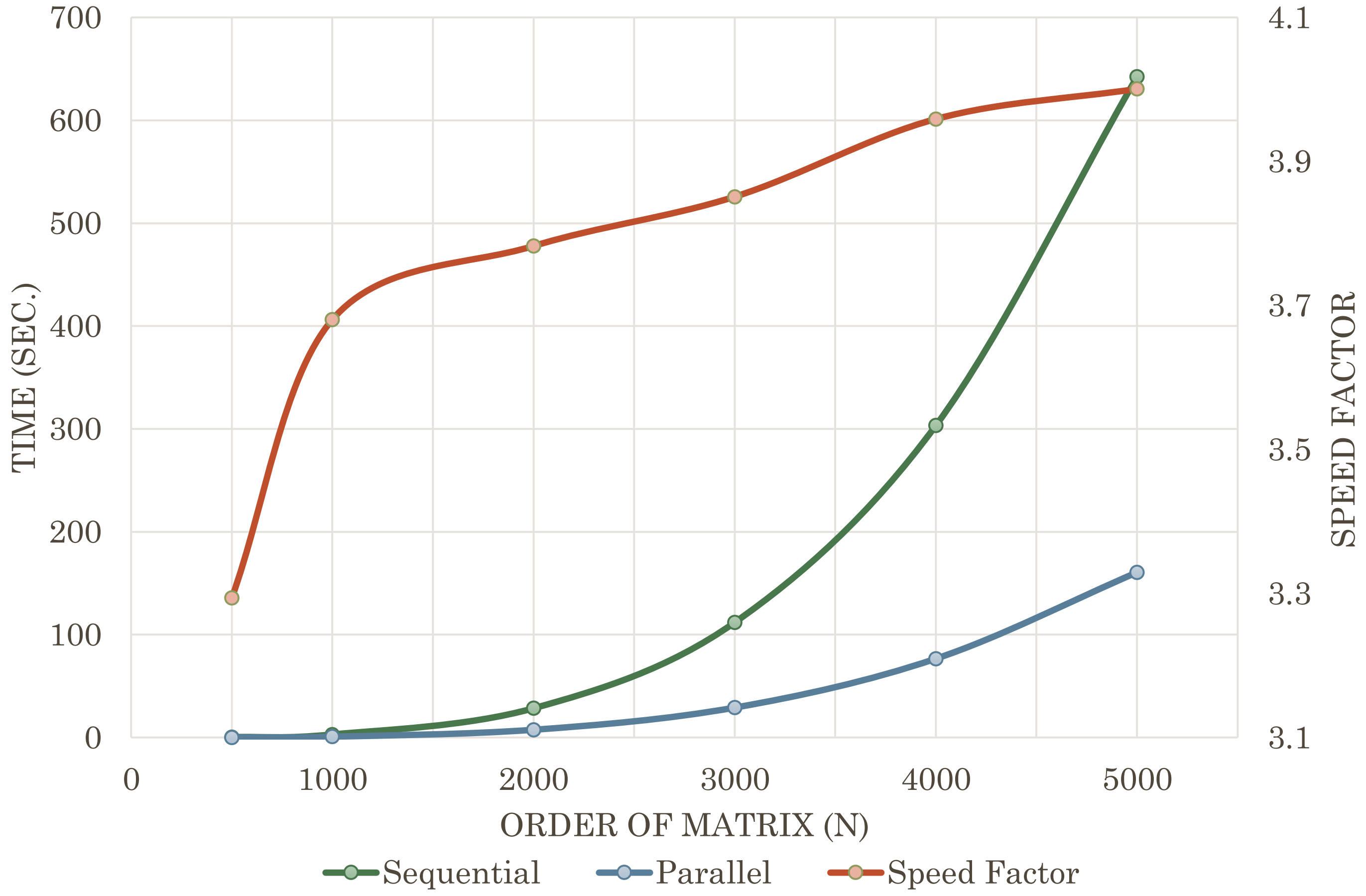
Multiplying matrix A and B and storing it in matrix multi

Random numbers are initialized in a and b matrices

Sequential and Parallel – Performance Comparison

Matrix Order	Sequential Time (s)	Parallel Time (s)	Speed Factor
500	0.3631	0.1102	3.2937
1000	2.9420	0.7993	3.6804
2000	28.386	7.5044	3.7826
3000	111.82	29.037	3.8510
4000	303.27	76.6048	3.9589
5000	642.27	160.537	4.0008

Table 1: Performance data



Graph 1: Sequential vs Parallel programming

Conclusion

- Clearly, from Graph: 1 when working with large order matrices the computation time of sequential increases significantly as compared to the parallel code.
- If the order of the matrix is small, the overhead associated with parallel computation negates the performance gained through multithread work-sharing.

References

[1] "Matrix multiplication algorithm", en.wikipedia.org. [Online]. Available: https://en.wikipedia.org/wiki/Matrix_multiplication_algorithm

[2] "Difference between Sequential and Parallel Programming", Kato Mivule's Tech. [Online]. Available: https://mivuletech.wordpress.com/2011/01/12/difference-between-sequential-and-parallel-programming/

[3] "Introduction to Parallel Computing", Computing.llnl.gov. [Online]. Available: https://computing.llnl.gov/tutorials/parallel_comp/#WhyUse

[4] Rohit Chandra, 2001, ISBN 1-55860-671-8, Parallel Programming in OpenMP, 2001 by Academic Press

[5] "32 OpenMP traps for C++ developers | Intel® Software", Software.intel.com. [Online]. Available: https://software.intel.com/en-us/articles/32-openmp-traps-for-c-developers

[6] "OpenMP", En.wikipedia.org. [Online]. Available: https://en.wikipedia.org/wiki/OpenMP

[7] "Random Numbers Info", Randomnumbers.info. [Online]. Available: http://www.randomnumbers.info/content/Random.htm

[8] "Properties of Random Numbers", eg.bucknell.edu. [Online]. Available: https://www.eg.bucknell.edu/~xmeng/Course/CS6337/Note/master/node37.html

[9] Von Neumann, J.; Goldstine, H.H. (1947). "Numerical inverting of matrices of high order". Bull. Amer. Math. Soc. 53 (11): 1021–1099. doi:10.1090/S0002-9904-1947-08909-6.

[10] Wigner, E. (1955). "Characteristic vectors of bordered matrices with infinite dimensions". Annals of Mathematics. 62 (3): 548–564. doi:10.2307/1970079

[11] D. Roberts, "Random Number Generation Funcitons", Mathbits.com. [Online]. Available: https://mathbits.com/MathBits/CompSci/LibraryFunc/rand.htm

[12] Wilkinson and M. Friendly, "The History of the Cluster Heat Map", The American Statistician, vol. 63, no. 2, pp. 179-184, 2009

[13] “OpenMP Clauses", Msdn.microsoft.com. [Online]. Available: https://msdn.microsoft.com/en-us/library/2kwb957d.aspx

Acknowledgement

We sincerely thank Prof. Dr. Peter Luksch & Markus Wolfien for their continued guidance and encouragement in carrying out this project work.