Universität Rostock Traditio et Innovatio

Computational Science and Engineering

Introduction to High Performance Computing

Comparison between Sequential and Parallel Programming for Large Matrix Multiplication

Supervised by
Markus Wolfien

Submitted by
Group 8

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

1. Matrix Multiplication

Matrix multiplication is a binary operation that produces a matrix from two matrices. If **A** is a $n \times m$ order matrix and **B** is a $m \times p$ order matrix,
The **matrix product AB** (denoted without multiplication signs or dots) is defined to be the $n \times p$ order matrix. [1]

A= $\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{pmatrix}$ B= $\begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1p} \\ b_{21} & b_{22} & \cdots & b_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \cdots & b_{mp} \end{pmatrix}$ AB= $\begin{pmatrix} ab_{11} & ab_{12} & \cdots & ab_{1p} \\ ab_{21} & ab_{22} & \cdots & ab_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ ab_{n1} & ab_{n2} & \cdots & ab_{np} \end{pmatrix}$

Where each *i, j* entry is given by multiplying the entries A_{ik} (across row *i* of **A**) by the entries B_{kj} (down column *j* of **B**), for $k = 1, 2, ..., m$ and summing the results over *k*:

$$AB = \sum_{k=1}^{m} A_{ik}\, B_{kj}$$

2. Algorithm

A simple iterative algorithm can be constructed which loops over the indices *i* from 1 through *n* and *j* from 1 through *p*, computing the above using a nested loop:

- Input: matrices **A**: $n \times m$ order matrix and **B**: $m \times p$ order matrix.
- Let *Multi* be a new matrix formed of the order n x p. [2]

```
For $i$ from 1 to $n$:                      (matrix A is accessed row-by-row)
    For $j$ from 1 to $p$:                  (matrix B is accessed column-by-column)
        Let $Multi_{ij} = 0$
        For $k$ from 1 to $m$:
                Set $Multi \leftarrow Multi + A_{ik} \times B_{kj}$
        Set $Multi_{ij} \leftarrow$ sum
Return $Multi$
```

3. Parallel Implementation and choosing OpenMP

Considering our two available matrices A and B to be multiplied, The OpenMP-enabled parallel code exploits coarse grain parallelism, and provides a standard parallel API specifically for programming shared memory multiprocessors.
We have parallelized the first loop which drives the access to the resultant multiplied matrix in the first dimension. Here, work-sharing among the threads is such that each thread will calculate different rows of the multiplied matrix. Each thread writes different part of the result in the resultant array, to prevent any problems during the parallel execution. Access to matrices *A, B* and *C* are read-only and avoid any problems related to memory access.

3.2 On Parallizing code

Code has been parallized in order to utilize all available hardware resources and divide a large task such as a 1000×1000 matrix multiplication into smaller tasks so as to minimize processing time.

3.3 OMP Pragma Clauses used

Private: Each thread has its own copy of private variable, which is invisible to other threads. This is done because every thread needs to do its own iterations of the loops.
Shared: All threads can see the same copy as well as read and write on the shared variables.
Dynamic Scheduling: It executes the scheduled pragma block and the size is controlled by the value of chunk which is 1 by default. When the thread finishes, it will be assigned the next iteration that hasn't been executed yet.
Barrier: Here, each thread waits for all the other threads to arrive at the barrier. Only when all threads have arrived do they continue execution past the barrier directive. It also synchronizes execution of multiple threads in the parallel region. [3]

4. Applications

4.1 Applications of Matrix multiplication

Matrix multiplication finds its use in various mathematical disciplines and branches of science, such as,

i. A system of linear equations can be solved by solving the matrix equation of the form, **Ax=b.** Where A is a coefficient matrix of order $m \times n$, x is a column vector of n entries, and b is column vector of m entries. [2]
ii. Matrix multiplication is also used for Image processing. Any number of color transformations can be composed using standard matrix multiplication.
iii. It is also used for processing digital video and digital sound. Techniques for filtering or compressing digital audio signals rely on matrix multiplication.[4]
iv. In solving graph theory problems.

4.2 Applications of Random Matrix

i. In Aerospace and Electronic industries as LED for Radars.
ii. In Numerical analysis to describe computation errors.[5]
iii. In Nuclear Physics to model the nuclei of heavy atoms to resemble spacing between eigenvalues of random numbers.[6]

4.3 Applications of Heat maps

A heat map is an ingenious display that simultaneously reveals row and column hierarchical structure in a data matrix. It consists of a rectangular tiling, with each tile shaded on a colour scale to represent the value of the corresponding element of the data matrix.[7]
i. Web heat maps have been used for displaying areas of a Web page most frequently scanned by visitors.
ii. Biological heat maps are typically used in molecular biology to represent the level of expression of many genes across a number of comparable samples.[8]
iii. A density function can be visualized using heat maps for representing the density of dots in a map. It enables to observe density of points independently of the zoom factor.[9]

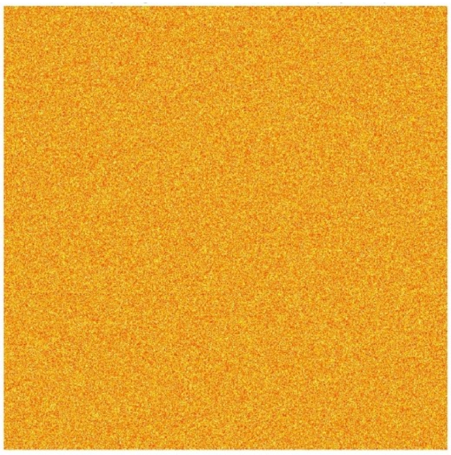
Figure 1. Heat Map

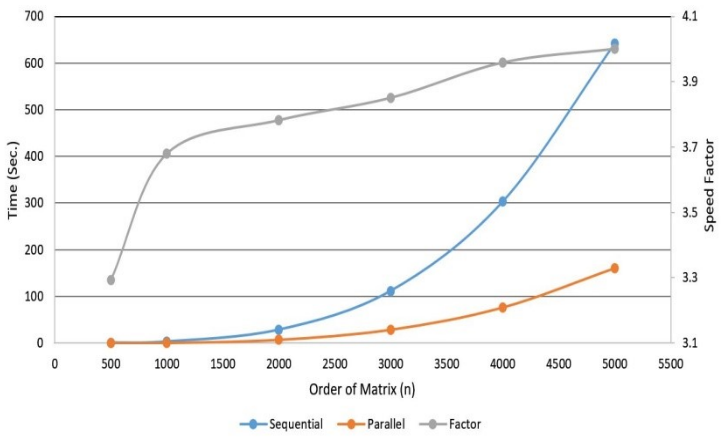

Figure 2. Sequential vs Parallel programming

Conclusion

Clearly, from Figure 2. 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] https://en.wikipedia.org/wiki/Matrix_multiplication
[2] https://en.wikipedia.org/wiki/Matrix_multiplication_algorithm
[3] Rohit Chandra, 2001, ISBN 1-55860-671-8, Parallel Programming in OpenMP, 2001 by Academic Press.
[4] http://graficaobscura.com/matrix/index.html
[5] 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.
[6] Wigner, E. (1955). "Characteristic vectors of bordered matrices with infinite dimensions". *Annals of Mathematics*. **62** (3): 548–564. doi:10.2307/1970079.
[7] Wilkinson and M. Friendly, "The History of the Cluster Heat Map", *The American Statistician*, vol. 63, no. 2, pp. 179-184, 2009.
[8] https://en.wikipedia.org/wiki/Heat_map
[9] *Perrot, A.; Bourqui, R.; Hanusse, N.; Lalanne, F.; Auber, D (2015). "Large interactive visualization of density functions on big data infrastructure". IEEE 5th Symposium on Large Data Analysis and Visualization (LDAV), 2015: 99–106. doi:10.1109/LDAV.2015.7348077.*