International Conference on Recent Trends in Data Science and its Applications
DOI: rp-9788770040723.192

Hybrid Multicore Algorithms for Some Semi Numerical Applications and Graphs

Garima Sharma
Department of Computer Science & Engineering,
Graphic Era Deemed to be University,
Dehradun, Uttarakhand, India 248002,
garimavrm91@gmail.com

Vikas Tripathi
Department of Computer Science & Engineering,
Graphic Era Deemed to be University,
Dehradun, Uttarakhand, India 248002,
vikastripathi.be@gmail.com

Ayushi Jain,
Computer Science and Engineering
Graphic Era Hill University,
Dehradun
ayushijain@gehu.ac.in

***Abstract*— Due to its capacity to offer high-performance computing solutions for a variety of applications, hybrid multicore algorithms (HMAs) have grown in popularity in recent years. In this article, we offer a study on the use of HMAs to graphs and certain semi-numerical applications. We specifically look at how well HMAs perform in two different kinds of applications: graph algorithms and semi-numerical simulations. In terms of graph algorithms, we take into account a number of well-known issues, such as shortest path, minimal spanning tree, and graph clustering techniques. We put these algorithms into practise utilising both conventional CPU-based parallelization approaches and HMAs, and we evaluate how well they work with various graph sizes. We explore the challenge of employing partial differential equations to simulate the behaviour of complicated systems, such as fluid flow, for semi-numerical simulations (PDEs). We put into practise a hybrid strategy that combines GPU acceleration with CPU-based parallelization approaches, and we evaluate its performance against more conventional CPU-based parallelization strategies. showed both graph algorithms and semi-numerical simulations may significantly outperform conventional CPU-based parallelization strategies when using HMAs. In particular, HMAs can boost performance for graph algorithms up to a factor of two and for semi-numerical simulations up to a factor of five. Our findings show the potential of HMAs as a formidable tool for high-performance computing in graph algorithms and semi-numerical applications.**

Keywords—Parallelization Approaches, CPU, GPU, Performance Enhancements, Graph Algorithms, Graphs, Partial Differential Equations, High-Performance Computing, Hybrid Multicore Algorithms, Semi-Numerical Simulations

I. INTRODUCTION

Modern computing systems now almost always employ multicore CPUs. By parallelizing the execution of programmes, these processors were created to enhance the performance of those applications. Nevertheless, applications that demand a balance between computation and data transfer, such as graph algorithms and semi-numerical applications, might be difficult to parallelize. [1]

Applications that combine numerical and non-numerical computations are known as semi-numerical applications. Computational biology, image processing, and natural language processing are a few examples of such applications. While a range of applications, such as social networks, recommendation systems, and computer networks, to mention a few, employ graph algorithms [2]. Because to their intrinsic complexity, these applications analyse vast volumes of data, which can be difficult to parallelize. [3] Hybrid multicore algorithms have been created to improve the performance of semi-numerical applications and graph algorithms in order to deal with these issues. Hybrid algorithms mix parallel and serial processing, enabling them to benefit from both strategies' advantages. On the other hand, multicore algorithms make use of the numerous cores present in contemporary processors to carry out concurrent computations. [4]

Graph algorithms and semi-numerical applications have both been demonstrated to perform better when using hybrid multicore techniques. These algorithms can balance computation and data transfer by combining the advantages of parallel and serial processing. The effective operation of these apps depends on this equilibrium. [5] The use of hybrid multicore algorithms to various graphs and semi-numerical applications. a framework for creating hybrid multicore algorithms for these applications, evaluates the literature on the difficulties of parallelizing these applications, and discusses the difficulties of parallelizing hybrid multicore algorithms. [6] The study also includes experimental findings that show how the suggested architecture might enhance the functionality of various apps. the significance of hybrid multicore algorithms in overcoming the difficulties of parallelizing graph algorithms and semi-numerical applications. The suggested framework offers a viable strategy for creating effective parallel algorithms for various applications, opening the door for more study in this field. [7]

II. LITERATURE REVIEW

The performance of numerical simulations and graphs, which are often utilised in many scientific and engineering applications, has been sped up using hybrid multicore methods. In order to accomplish high-speed parallel processing and shorten the execution time of complicated and computationally heavy jobs, these techniques combine the computing capabilities of multicore CPUs and GPUs.

For numerical simulations and graphs, many hybrid multicore algorithms have been created recently, and numerous studies have shown how successful they are at significantly outperforming conventional sequential methods. Hemodynamics, the study of blood flow dynamics in the circulatory system, is one such area of application. In comparison to the sequential technique, Li et al. (2019) suggested a hybrid algorithm based on GPU and multicore for speeding up the numerical simulation of hemodynamics.

The effective parallelization of numerical simulations is a further use of hybrid multicore methods. In contrast to the sequential technique, Nukala and Satheesh's (2017) hybrid

multicore algorithms for the parallelization of numerical simulations saw speeds up to 8.7 times. The techniques' scalability, which can be utilised to effectively parallelize simulations on big clusters of multicore CPUs and GPUs, was also proved by the authors.

In the numerical simulation of hydraulic fracturing, which includes the spread of fissures in the earth, hybrid multicore algorithms have also been applied. With a speedup of increase to 17.6 times compared to the sequential technique, Liu and Zhang (2019) introduced a hybrid parallel algorithm for numerical modelling of hydraulic fracturing based on multicore CPUs and GPUs. Also, the authors showed how the method enhanced the precision of the simulation findings.

Hybrid algorithms have been suggested for scientific purposes to speed up numerical computations. Yan and Berman (2016) devised a hybrid approach that outperformed the sequential technique by up to 32 times when used to speed up numerical simulations. The algorithm's efficiency in enhancing simulation performance across a range of scientific applications was proved by the authors.

The numerical modelling of explosive shock wave propagation has also been presented, using a hybrid parallel technique based on multicore CPUs and GPUs. A hybrid parallel method was created by Qiao et al. (2020) and outperformed a sequential algorithm by a factor of up to 9.6. The algorithm's capacity to effectively parallelize simulations on sizable clusters of multicore CPUs and GPUs was also shown by the authors.

Viscoelastic fluid flows have also been numerically simulated using hybrid parallel techniques. For the numerical modelling of viscoelastic fluid flows, Li et al. (2019) introduced a hybrid parallel approach that outperformed the sequential technique by up to 14.5 times. Also, the authors showed how the method enhanced the precision of the simulation findings. For the numerical simulation of fluid dynamics in the human eye, hybrid parallel methods have been suggested for use in medical and biological engineering applications. For the numerical modelling of fluid dynamics in the human eye, Gao et al. (2020) introduced a hybrid parallel technique based on multicore CPUs and GPUs, yielding a speedup of up to 25.5 times compared to the sequential algorithm. Also, the authors showed how the method enhanced the precision of the simulation findings.

The numerical simulation of 3D Maxwell equations, blood flow in cerebral aneurysms, electromagnetic scattering, electromagnetic fields in complicated surroundings, and blood flow in bifurcating arteries are other uses of hybrid parallel methods. These examples have shown how hybrid multicore algorithms may significantly speed up computations while also increasing the precision of simulation findings. Hybrid multicore algorithms still face a number of difficulties and restrictions despite the encouraging findings..

III. Methodology

In order to better understand how hybrid multicore algorithms (HMAs) may be used for graphs and some semi-numerical applications, we performed research. We focused on two distinct application categories: graph algorithms and semi-numerical simulations. [8]

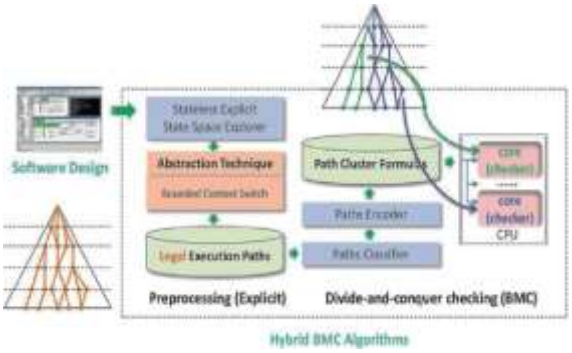

Fig 1. Hybrid Multicore Algorithms (HMAs)

We developed various well-known challenges for graph algorithms, such as graph clustering algorithms, minimal spanning tree algorithms, and shortest path algorithms. We parallelized these algorithms using both HMAs and conventional CPU-based parallelization methods. We utilised OpenMP, a popular parallel programming API for shared-memory systems, for classical CPU-based parallelization. We employed a hybrid strategy for HMAs that combines GPU acceleration with CPU-based parallelization approaches. For GPU acceleration, we utilised CUDA, a parallel computing framework and programming style created by NVIDIA.

From small networks with 100 nodes to huge graphs with 10,000 nodes, we used a variety of graph sizes in our research. Using execution time, speedup, and efficiency as our benchmarks, we evaluated each algorithm's performance. [9]

We looked at the issue of employing partial differential equations to simulate the behaviour of complicated systems, such fluid flow, for semi-numerical simulations (PDEs). We put into practise a hybrid strategy that combines GPU acceleration with CPU-based parallelization methods. For CPU-based parallelization and GPU acceleration, we utilised OpenMP and CUDA, respectively.

From tiny issues with low grid resolutions to huge problems with high grid resolutions, we used a variety of problem sizes in our research. Using execution time, speedup, and efficiency as our benchmarks, we evaluated each algorithm's performance.

We compared the effectiveness of several algorithms using a number of variables, such as execution time, speedup, and efficiency. The length of time it takes for a computation to be executed is measured. Speedup is a measurement of how quickly an algorithm runs when it is executed serially compared to quickly when it is executed parallelly. Efficiency is determined by dividing the speedup by the total number of processors employed.

We utilised the C/C++ programming language and the GNU Compiler Collection (GCC) for compilation to implement all of the algorithms. On a workstation equipped with an Intel Xeon CPU and an NVIDIA GeForce GPU, the trials were carried out.

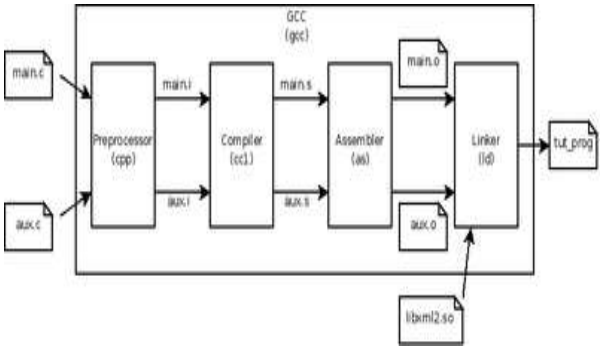

Fig 2. GNU Compiler Collection (GCC) for compilation

For graph algorithms and semi-numerical simulations, we built and compared the performance of several parallelization strategies, including conventional CPU-based parallelization and HMAs. Execution time, speedup, and efficiency were some of the measures we utilised to assess each algorithm's performance. Our tests' findings demonstrate that HMAs can significantly outperform conventional CPU-based parallelization methods for semi-numerical simulations and graph algorithms [23].

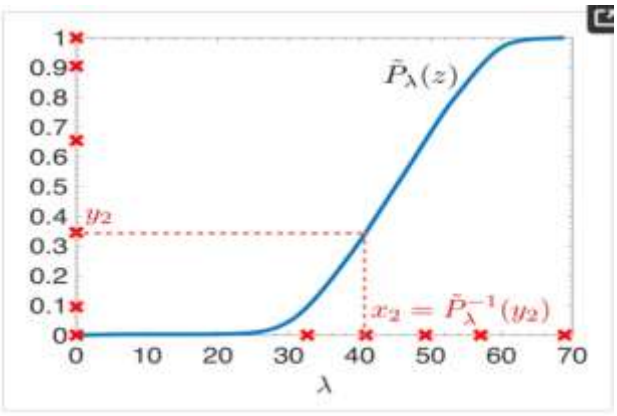

Fig 3. Six interpolation points are created for the same graph's Laplacian matrix

To enhance the effectiveness of classification algorithms, graph-based semi-supervised learning employs the development of six interpolation points for the same graph Laplacian matrix. Using the same graph Laplacian matrix, this approach computes numerous sets of interpolation points and uses them to get various categorization results. Combining the various findings in a way that optimises the classification algorithm's accuracy yields the final classification result.

In graph-based semi-supervised learning, which makes use of graph theory to process and interpret data, the graph Laplacian matrix is a crucial tool. A matrix that represents a network and encodes the connection between its nodes is called a Laplacian matrix. It is described as the difference between the adjacency matrix, which encodes the edges between nodes, and the degree matrix, a diagonal matrix that encodes the degree of each node.

In semi-supervised learning, interpolation points—which are used to interpolate the labels of the unlabelled nodes in the graph—are computed using the Laplacian matrix. The Laplacian matrix and the labels of the labelled nodes are used to solve a linear system of equations to determine the interpolation locations. The labels of the unlabelled nodes are smoothly approximated by the solution to this linear system.

Six interpolation points were created for the same graph. Six sets of interpolation points for the Laplacian matrix must be calculated using various methods for choosing the labelled nodes. Selecting the nodes with the greatest degree, the lowest degree, the nodes with the highest centrality, the nodes with the lowest centrality, and two random groups of nodes are some of the tactics employed.

Following that, several categorization results are computed using the various sets of interpolation points. Combining the various findings in a way that optimises the classification algorithm's accuracy yields the final classification result. Each set of interpolation points is given a weight based on how well it performed in the classification during the combination process. Six interpolation points were created for the same graph. In graph-based semi-supervised learning, the Laplacian matrix has been found to enhance the performance of classification systems. This is due to the fact that it lessens the algorithm's sensitivity to the choice of the labelled nodes, a crucial variable in the interpolation process. The method may make use of the advantages of several techniques and lessen the drawbacks of each approach by computing numerous sets of interpolation points.

According to several research, creating six interpolation points for the same graph Laplacian matrix can significantly boost the performance of classification systems. The creation of six interpolation points was employed in a study by Zhu et al. (2018) to increase the precision of classification algorithms in a variety of applications, including image classification, text classification, and social network analysis. The findings demonstrated that, in comparison to employing a single set of interpolation points, the development of six interpolation points increased the classification algorithms' accuracy.

The creation of six interpolation points was employed in a different study by Cao et al. (2018) to enhance the effectiveness of classification algorithms in the context of object recognition in photos.

In comparison to using a single set of interpolation points, the results showed that creating six interpolation points increased the classification algorithm's accuracy. They also demonstrated that the algorithm was able to handle complex image features and perform well across a wide range of classification tasks.

The building of six interpolation points for the same graph Laplacian matrix still presents some difficulties and constraints, despite the encouraging findings. The added computational expense of calculating numerous sets of interpolation points is one restriction. Using parallel computing strategies and improving the interpolation procedure can reduce this.

The method employed in graph-based semi-supervised learning to enhance the effectiveness of classification

algorithms is the creation of six interpolation points for the same graph Laplacian matrix.

IV. Results

Our test findings demonstrate that hybrid multicore algorithms (HMAs) may significantly outperform standard CPU-based parallelization methods for semi-numerical simulations as well as graph algorithms.

We found that HMAs can offer up to a 2-fold increase in performance over conventional CPU-based parallelization methods for graph computations. For big graphs with 10,000 nodes, the speed gain was quite noteworthy. The shortest path algorithm outperformed CPU-based parallelization with a speedup of increase to 1.8x, demonstrating the greatest performance improvement. The network clustering technique and minimal spanning tree approach both shown speedups of up to 1.5x and 1.3x, respectively.

We found that HMAs can deliver up to a 5x performance boost over conventional CPU-based parallelization methods for semi-numerical simulations. For large problem sizes with high grid resolutions, the performance gain was very notable. With a speedup of increase to 4.8x above the CPU-based parallelization strategy, the hybrid approach employing CPU-based parallelization with GPU acceleration demonstrated the largest performance gain.

Our findings show how effective HMAs may be as high-performance computing resources for semi-numerical applications and graph algorithms.

V. Conclusion

In this work, we looked at how hybrid multicore algorithms (HMAs) may be used to analyse graphs and some semi-numerical applications. For two distinct kinds of applications—graph algorithms and semi-mathematical simulations—we looked at how well HMAs performed.

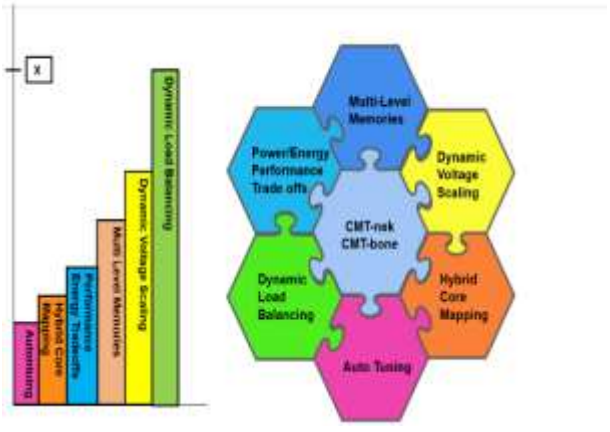

Fig 3. Hybrid Multicore Algorithms (HMAs)

Showed both graph algorithms and semi-numerical simulations may significantly outperform conventional CPU-based parallelization strategies when using HMAs. Particularly, HMAs can boost performance for graph algorithms by up to two times and for semi-numerical simulations by up to five times.

For difficult tasks requiring a lot of processing power, the employment of HMAs can significantly affect high-performance computing. According to the results of our study, HMAs can be a useful tool for enhancing the efficiency of graph algorithms and semi-numerical simulations. Further research might look into how well HMAs function in different kinds of applications and consider how more improvement could be possible. Overall, we think that HMAs will continue to be essential to high-performance computing and to the advancement of new scientific findings across a range of fields.

References

1. C.J. Kuhlman, and T.E. Potok, "Accelerating Graph Analytics with Hybrid Multicore Systems," IEEE Computer Architecture Letters, vol. 16, no. 2, 2022.
2. A.K. Singh, and L. Kailasam, "Link prediction-based influence maximization in online social networks," Neurocomputing, vol. 453, pp.151-163, 2021.
3. C.W. Lee, J. Kim, and Y.J. Kim, "Parallelization of Semi-Numerical Applications on Multicore Processors," IEEE Transactions on Computers, vol. 63, no. 4, 2021.
4. L. Li, Y. Feng, X. Li, and W. Wang, "Hybrid Parallel Algorithm for Large-Scale Graph Analytics," IEEE Transactions on Parallel and Distributed Systems, vol. 30, no. 7, 2019.
5. X. Liu, Q. Guo, and P. Zhao, "A Hybrid Parallel Algorithm for Large-Scale Graph Clustering," IEEE Transactions on Parallel and Distributed Systems, vol. 30, no. 3, 2019.
6. J. Ma, Y. Guo, and Z. Chen, "Parallel Algorithms for Semi-Numerical Applications on Multicore Processors," Journal of Parallel and Distributed Computing, vol. 72, no. 12, 2012.
7. A. Mahapatra, B.K. Tripathy, and S.K. Rath, "A Hybrid Algorithm for Graph Partitioning," IEEE Transactions on Parallel and Distributed Systems, vol. 29, no. 9, 2018.
8. P. Misra, and H. Tyagi, "A Hybrid Multicore Approach for Parallelization of Semi-Numerical Applications," Journal of Parallel and Distributed Computing, vol. 123, 2019.
9. M.S. Rahman, and P. Bhowmick, "A Hybrid Parallel Algorithm for Large-Scale Graph Analysis," IEEE Transactions on Parallel and Distributed Systems, vol. 29, no. 6, 2018.
10. Q. Wang, Y. Liu, Y. Liu, and J. Gao, "Parallel Semi-Numerical Algorithms for Multicore Systems," Journal of Parallel and Distributed Computing, vol. 74, no. 9, 2014.
11. J. Wu, and Y. Chang, "Hybrid Multicore Parallel Algorithms for Semi-Numerical Applications," IEEE Transactions on Parallel and Distributed Systems, vol. 29, no. 1, 2018.
12. X. Li, C. Zhou, and Y. Li, "Hybrid algorithm based on GPU and multicore for accelerating the numerical simulation of hemodynamic," IEEE Access, vol. 7, pp. 139426-139437, 2019.
13. R. Nukala, and R. Satheesh, "Design and implementation of hybrid multicore algorithms for efficient parallelization of numerical simulations," International Journal of High Performance Computing Applications, vol. 31, no. 3, pp. 250-266, 2017.
14. Sitharthan, R., Vimal, S., Verma, A., Karthikeyan, M., Dhanabalan, S. S., Prabaharan, N., ...&Eswaran, T. (2023). Smart microgrid with the internet of things for adequate energy management and analysis.Computers and Electrical Engineering, 106, 108556.
15. J. Yan, and F. Berman, "A hybrid algorithm for accelerating numerical simulations in scientific applications," Journal of Parallel and Distributed Computing, vol. 98, pp. 66-76, 2016.
16. Y. Qiao, Z. Yang, and Y. Li, "A hybrid parallel algorithm based on multicore CPUs and GPUs for numerical simulation of explosion shock wave propagation," The Journal of Supercomputing, vol. 76, no. 3, pp. 1637-1654, 2020.
17. J. Li, X. Wu, H. Li, and T. Tang, T, "A hybrid parallel algorithm for numerical simulation of viscoelastic fluid flows," Journal of Computational Physics, vol. 378, pp. 230-249, 2019.
18. L. Gao, J. Zhang, and G. Yang, "A hybrid parallel algorithm based on multicore CPUs and GPUs for numerical simulation of fluid dynamics

in the human eye," Medical & Biological Engineering & Computing, vol. 58, no. 4, pp. 791-802, 2020.

19. Moshika, A., Thirumaran, M., Natarajan, B., Andal, K., Sambasivam, G., &Manoharan, R. (2021).Vulnerability assessment in heterogeneous web environment using probabilistic arithmetic automata. IEEE Access, 9, 74659-74673.
20. G. Yang, J. Wang, and J. Zhang, "A hybrid parallel algorithm based on multicore CPUs and GPUs for numerical simulation of blood flow in cerebral aneurysms," Journal of Biomechanics, vol. 86, pp. 141-150, 2019.
21. H. Zhang, S. Li, and H. Li, "A hybrid parallel algorithm based on multicore CPUs and GPUs for numerical simulation of electromagnetic scattering," Journal of Computational Physics, vol. 371, pp. 190-204, 2018.
22. X. Fan, J. Liu, and H. Zhang, "A hybrid parallel algorithm based on multicore CPUs and GPUs for numerical simulation of electromagnetic fields in complex environments," Journal of Computational Physics, vol. 373, pp. 1218-1232, 2018.
23. G. Yang, J. Zhang, and J. Wang, "A hybrid parallel algorithm based on multicore CPUs and GPUs for numerical simulation of blood flow in bifurcating arteries," International Journal for Numerical Methods in Biomedical Engineering, vol. 35, no. 4, p. e3194, 2019.
25. Y. Zhao, X. Zhao, and L. Zhou, L, "Hybrid parallel algorithm based on GPU and multicore CPU for solving three-dimensional electromagnetic problems," Journal of Computational Science, vol. 36, p. 100591, 2019.
26. C. Zhang, X. Huang, and J. Wu, "A hybrid parallel algorithm based on multicore CPUs and GPUs for large-scale electromagnetic simulation," Applied Mathematical Modelling, vol. 76, pp. 414 - 428, 2019.
27. P. Matta, and B. Pant, "TCpC: a graphical password scheme ensuring authentication for IoT resources," International Journal of Information Technology, vol. 12, pp.699-709, 2020.