2024 International Conference on Smart Systems for Electrical, Electronics, Communication and Computer Engineering (ICSSEECC) | 979-8-3503-7817-7/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICSSEECC61126.2024.10649542

Performance Analysis of Parallelized PageRank Algorithm using OpenMP, MPI and CUDA

Visali V S
Department of CSE
PSG Institute of Technology and Applied Research
Tamilnadu, India
visali.vs.4903@gmail.com

Manimegalai R
Department of CSE
PSG Institute of Technology and Applied Research
Tamilnadu, India
drrm@psgitech.ac.in

Noor Mahammad S K
Department of CSE
Indian Institute of Information Technology, Design and Manufacturing
Tamilnadu, India
noor@iiitdm.ac.in

Sunitha Nandhini A
Department of CSE
PSG Institute of Technology and Applied Research
Tamilnadu, India
asn@psgitech.ac.in

***Abstract*— Web search engines have developed into a crucial tool for effectively and quickly locating information among the vast web data. The PageRank algorithm is essential for web search as it measures the importance and relevance of web pages based on their incoming links, allowing search engines to rank results by prioritizing high-quality and authoritative content. This ensures that users receive more accurate and valuable information. The PageRank algorithm improves search engine results by assigning a numerical weight to each element in a hyperlinked set of documents, effectively measuring the importance of web pages based on quantity and quality of links pointing to them. With billions of web pages and an extensive network of hyperlinks, the traditional sequential computation of PageRank becomes impractical for timely and efficient processing. Parallelization allows the algorithm to be distributed across multiple processors or computing nodes, enabling simultaneous computation of PageRank scores for different web pages. The main objective of this work is to parallelize the PageRank algorithm using OpenMP, MPI and CUDA and to compare their execution time to find the optimal one. OpenMP simplifies shared-memory parallelism, MPI facilitates communication between distributed processes and CUDA harnesses GPU power for high-performance parallel processing in diverse parallel computing environments. The experimental results demonstrate notable performance enhancements through parallelization using different technologies: OpenMP improves the algorithm's performance by 49.7%, MPI by 62.4%, and CUDA by 84.3%. Hence, optimal results are found when the PageRank algorithm is parallelized using CUDA.**

Keywords—PageRank Algorithm, Hyperlinks, Parallelization, OpenMP, MPI, CUDA, GPU.

I. OVERVIEW OF PAGERANK ALGORITHM

A. About the algorithm

Search engines use a link analysis algorithm called PageRank to assign a numerical weight to each element of a hyperlinked set of documents, such as the World Wide Web[6]. The algorithm is named after Larry Page, one of the co-founders of Google, and it was developed by him and Sergey Brin. The basic idea behind PageRank is to measure the importance of web pages based on the structure of the hyperlink graph. In essence, it views a link from page-A to page-B as a vote by page-A for page-B. The page with more hyperlinks receives more votes and has higher page-rank. Mathematically, PageRank can be computed using Equation (1).

$$PR(A) = (1-d) + d\left(\frac{PR(B)}{L(B)} + \frac{PR(C)}{L(C)} + \cdots + \frac{PR(N)}{L(N)}\right) \quad (1)$$

In Equation (1), *PR(A)* is the PageRank of page A, *d* is a damping factor, usually set to 0.85, *PR(B)*, *PR(C)*, ..., *PR(N)* are the PageRanks of pages that link to page A, and *L(B)*, *L(C)*, ..., *L(N)* are the number of outbound links on pages B, C, ..., N.

The page-rank of a page is determined by adding together the pageranks of all the pages that are linked to it, and then dividing that total by the total number of outbound links on all those pages. The pagerank calculation is an iterative process, and it converges as the pageranks stabilize. The algorithm is used by search engines to order search results, with pages having higher pageranks considered more relevant.

B. Advantages and applications of PageRank algorithm

The PageRank algorithm has several advantages that contribute to its widespread application. One key strength lies in its resistance to manipulation and spam, ensuring a more authentic assessment of a page's importance based on genuine link relationships rather than artificial tactics. Moreover, PageRank offers a global perspective by considering the entire link structure of the web. This comprehensive approach provides a more accurate evaluation of a page's significance, moving beyond simplistic counting-based algorithms. Another notable advantage is the algorithm's adaptability. Beyond its original application in web search, PageRank can be seamlessly applied to various network types. This versatility extends its utility to domains such as social networks, citation networks, and recommendation systems, making it a robust choice for a wide range of applications. PageRank's scalability is a crucial feature, especially in the context of the vast datasets encountered on the web. The algorithm efficiently scales with the size of the network, enabling the analysis and ranking of large amounts of information without compromising performance. Turning to its applications, PageRank plays a pivotal role in search engine ranking. By evaluating the relevance and importance of web pages, it significantly enhances the accuracy of search results, thereby improving the overall user experience.

adding more threads leads to extra overhead and doesn't improve efficiency.

MPI: The best performance is achieved with 4 processes, as supported by Manaskasemsak et al. [4], who noted that increasing the number of processes beyond this point adds communication overhead, reducing scalability.

CUDA: The excellent performance of CUDA in our study matches the findings of Kumar et al. [2] and Duong et al. [11], who also found that GPU acceleration significantly benefits graph processing algorithms. This comparison confirms our results and underscores the efficiency of CUDA for large-scale PageRank computations.

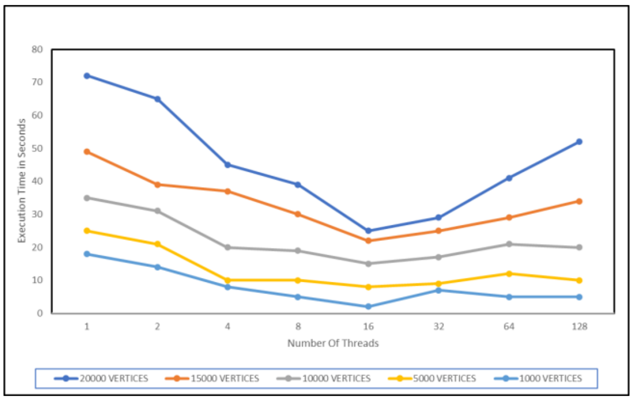

Fig. 4 Execution Time for Varying Number of Threads using OpenMP.

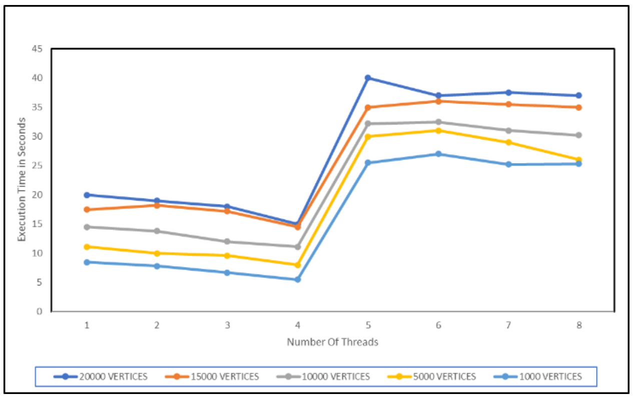

Fig. 5 Execution Time for Varying Number of Threads using MPI.

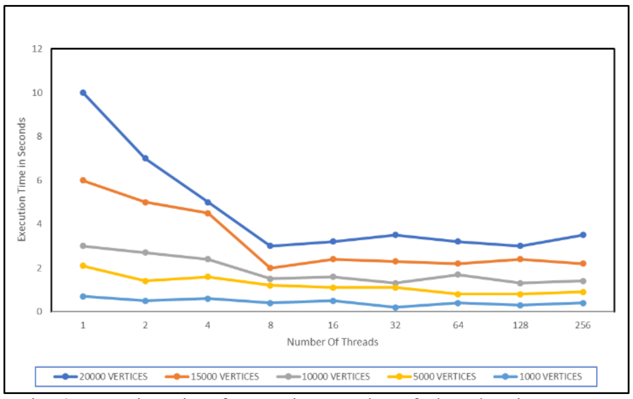

Fig. 6 Execution Time for Varying Number of Threads using CUDA.

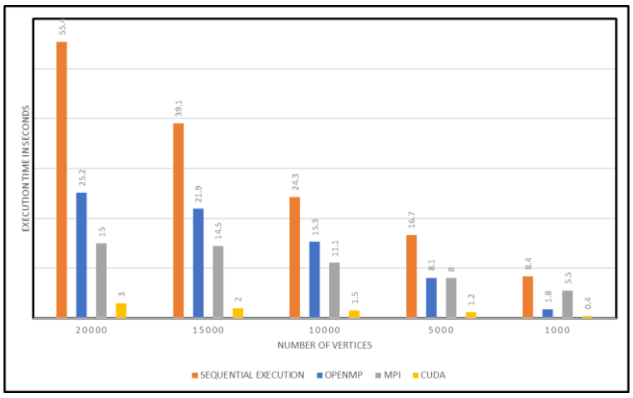

Fig. 7 Comparison of OpenMP, MPI and CUDA Implementations.

V. CONCLUSIONS

The experimental results of parallelizing the PageRank algorithm using OpenMP, MPI, and CUDA are systematically analyzed, and the performance of each parallelization technique is evaluated. The execution times for varying numbers of threads (OpenMP), processes (MPI), and threads per block (CUDA) are measured to understand the scalability and efficiency of each approach. The study demonstrated that CUDA outperformed OpenMP and MPI in terms of execution time, indicating that the GPU-accelerated parallelization using CUDA provided optimal results for the PageRank algorithm. The CUDA implementation took advantage of the parallel processing capabilities of GPUs, resulting in significant speedup and improved efficiency when compared to traditional CPU-based parallelization methods. Profiling using *gprof* highlighted the *pageRank( )* function as a hotspot, emphasizing the need for optimizing this portion of the algorithm. The experimental results further supported the observation, as the parallelization techniques influenced the execution time of this critical function. The blocksize parameter in CUDA implementation showed a noticeable impact on performance, demonstrating the importance of tuning GPU-specific parameters for optimal results.

In summary, the comprehensive experimental results and analysis presented in this work support the conclusion that CUDA provides the most efficient parallelization approach for the PageRank algorithm among other techniques such as OpenMP and MPI.

REFERENCES

[1] Kohlschütter, Christian, Paul-Alexandru Chirita, and Wolfgang Nejdl. "Efficient Parallel Computation of PageRank." In the 28th European Conference on IR Research, London, UK, pp. 241-252, 2006.
[2] Kumar, Tarun, Parikshit Sondhi, and Ankush Mittal. "Parallelization of PageRank on Multicore Processors." In the 8th International Conference, BICDCIT, Bhubaneswar, India, pp. 129-140, 2012.
[3] Cevahir, Ali, et al. "Site-Based Partitioning and Repartitioning Techniques for Parallel PageRank Computation." In the IEEE Transactions on Parallel and Distributed Systems, pp. 786-802, 2006.
[4] Manaskasemsak, Bundit, Putchong Uthayopas, and Arnon Rungsawang. "A Mixed MPI-Thread Approach for Parallel Page Ranking Computation." In the OTM Confederated International Conferences, CoopIS, DOA, GADA, and ODBASE, pp. 1223-1233, 2006.
[5] Desikan, Prasanna Kumar, et al. "Divide and Conquer Approach for Efficient PageRank Computation." In Proceedings of the 6th International Conference on Web Engineering, pp. 233-240, 2006.
[6] Duhan, Neelam, A. K. Sharma, and Komal Kumar Bhatia. "Page Ranking Algorithms: A Survey." In IEEE International Advance Computing Conference, pp. 1530-1537, 2009.

[7] Anakath, A. S., et al. "Optimization of PageRank Algorithm Using Parallelization Method." In AIP Conference Proceedings, vol. 2755, No. 1, 2023.
[8] Yang, Chao-Tung, Chih-Lin Huang, and Cheng-Fang Lin. "Hybrid CUDA, OpenMP, and MPI Parallel Programming on Multicore GPU Clusters." In Computer Physics Communications, vol. 182, no. 1, pp. 266-269, 2011.
[9] Mohamed, Khaled Salah, and Khaled Salah Mohamed. "Parallel Computing: OpenMP, MPI, and CUDA." In Neuromorphic Computing and Beyond: Parallel, Approximation, Near Memory, and Quantum, pp. 63-93, 2020.
[10] Rastogi, Shubhangi, and Hira Zaheer. "Significance of Parallel Computation Over Serial Computation Using OpenMP, MPI, and CUDA." In Quality, IT and Business Operations: Modeling and Optimization, pp. 359-367, 2018.
[11] Duong, Nhat Tan, et al. "Parallel PageRank Computation Using GPUs." In Proceedings of the 3rd Symposium on Information and Communication Technology, pp. 223-230, 2012.
[12] Zhou, Shijie, et al. "Design and Implementation of Parallel PageRank on Multicore Platforms." In IEEE High Performance Extreme Computing Conference, pp. 1-6, 2017.
[13] Yang, Chao-Tung, et al. "Performance Evaluation of OpenMP and CUDA on Multicore Systems." In International Conference on Algorithms and Architectures for Parallel Processing, pp. 235-244, 2012.
[14] Noaje, Gabriel, Michael Krajecki, and Christophe Jaillet. "MultiGPU Computing Using MPI or OpenMP." In Proceedings of the IEEE 6th International Conference on Intelligent Computer Communication and Processing, pp. 347-354, 2010.
[15] Xia, Kun. "The Implementation of Parallel Computation on CPU and GPU." University of Delaware, 2017.
[16] Graham, Susan L., Peter B. Kessler, and Marshall K. McKusick. "Gprof: A Call Graph Execution Profiler." In ACM Sigplan Notices, vol. 17, no. 6, pp. 120-126, 1982.