LITERATURE REVIEW: — Parallel String Matching —

Dengyu Liang
University of Ottawa
dengyuliang@cmail.carleton.ca

October 7, 2022

1   Introduction

String Matching is one of the most common problems in computer science, and although Single-pattern algorithms work very well on this problem, the algorithm speed is affected by the linear increase in the size of the matching string, especially when solving Bioinformatics and DNA Sequencing or Search engines or content search in large databases, a large amount of data inevitably takes a lot of time. And because of the search method of String Matching, it can take full advantage of Parallel Programming to solve large-scale Matching problems efficiently on large-scale systems.

This project is to explore the parallel implementation of traditional string matching algorithms and implement some efficient parallel string matching algorithms, as well as compare the optimization methods of string matching algorithms for different parallel architectures. Through this project, I will implement one or more string matching algorithms, and at least one efficient version, and analyze the focus of parallel algorithms under different architectures.

2   Literature Review

String Matching is an important problem in computer science, and many people continue to study it. The most recent comparison was Philip Pfaffe, who discussed 7 parallelized string matching algorithms based on SIMD. There are different architectures for computers, and for these architectures there are different methods to minimize latency and improve performance. In this study, the advantages and disadvantages of different architectures will be compared and the performance of applicable algorithms will be compared.

2.1   Simple String Matching

String Matching is one of the common fundamental problems, and many algorithms have been proposed to solve it. Faster solutions were proposed as early as 1977, the kmp algorithm and the Boyer–Moore algorithm[**?**]. Since 1970, more than 80 String Matching algorithms have been proposed. Through deformation, combination and expansion, String Matching has been continuously optimized. Now, using parallelized algorithms to improve

performance is now a more mainstream method[**?**] , nonetheless, can benefit from the ideas of these optimization algorithms. By exploring the details of these algorithms, the efficiency of parallelized algorithms can be further improved.

2.2 Multiple core(threads) Parallel String Matching

Parallel computing has great potential and has extremely high scalability. Making good use of parallel computing can greatly speed up the calculation speed. Although not all algorithms can benefit from parallel computing, especially some dynamic programming algorithms. Specific to the String Matching algorithm, this requires us to study, compared to the complex single-threaded optimization, to what extent does parallelism improve the operating efficiency?

Parallel String Matching base on MISD

Although there exist MISD architectures dedicated to pattern matching that can be used to solve string match [**?**], there is not much research in this area and it has not shown better performance. MISD processors also lack practical applications, and research in this area has been shelved.

Parallel String Matching base on SIMD

SIMD uses one instruction stream to process multiple data streams. A typical example is the unit on the GPU. Most Parallel String Matching benefits from SIMD architecture. Such as Chinta Someswararao's Butterfly Model [**?**], and many algorithms are developed based on the SSE instruction set [**?**].

Parallel String Matching base on GPU(CUDA)

Compared with CPU, modern GPU has huge advantages in parallel computing. A single GPU integrates thousands of computing units, so thousands of parallel computing can be realized on a single GPU. There are also some ways to use CUDA programming provided by nVidia gpu for String Matching. Giorgos Vasiliadis implements string search and regular expression matching operations for real-time inspection of network packets through the pattern matching library [**?**], And in 2011 it reached a data volume close to 30 Gbit/s. Efficient GPU algorithms can be 30 times faster than a single CPU core, and Approximate String Matching with k Mismatches reports an 80 times speedup [**?**]. This makes gpu computing have certain advantages in cost and speed, but the research in this area is not as much as the traditional string algorithm.

Parallel String Matching base on MIMD

MIMD machines can execute multiple instruction streams at the same time. Many modern CPUs belong to this type. In order to fully mobilize multiple instruction streams, targeted parallel algorithms are inevitable.

Distributed Memory Parallel String Matching

There are few articles discussing String Matching algorithm on Shared and Distributed-Memory Parallel Architectures, except Antonino Tumeo's Aho-Corasick algorithm in 2012 compared and analyzed the distributed memory architecture and shared memory architecture[**?**]. Results at the time showed that shared-memory architectures based on multiprocessors were

the theoretical best performance, considering cost constraints, GPUs that did not reach the PCI-Express bandwidth limit had the best price/performance ratio. Although distributed can provide sufficient space and computing resources, limited by the cost of communication, the performance of distributed computing is not satisfactory. Nonetheless, this can lead to considerable performance gains, as shown by Panagiotis' work[**?**]

References

[1] Pfaffe, P., Tillmann, M., Lutteropp, S., Scheirle, B., Zerr, K. (2017). Parallel String Matching. In: , et al. Euro-Par 2016: Parallel Processing Workshops. Euro-Par 2016. Lecture Notes in Computer Science(), vol 10104. Springer, Cham. https://doi.org/10.1007/978-3-319-58943-5 15

[2] C. -L. Hung, T. -H. Hsu, H. -H. Wang and C. -Y. Lin, "A GPU-based Bit-Parallel Multiple Pattern Matching Algorithm," 2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), 2018, pp. 1219-1222, doi: 10.1109/HPCC/SmartCity/DSS.2018.00205.

[3] G. Vasiliadis, M. Polychronakis and S. Ioannidis, "Parallelization and characterization of pattern matching using GPUs," 2011 IEEE International Symposium on Workload Characterization (IISWC), 2011, pp. 216-225, doi: 10.1109/IISWC.2011.6114181.

[4] Yuan Zu, Ming Yang, Zhonghu Xu, Lin Wang, Xin Tian, Kunyang Peng, and Qunfeng Dong. 2012. GPU-based NFA implementation for memory efficient high speed regular expression matching. SIGPLAN Not. 47, 8 (August 2012), 129–140. https://doi.org/10.1145/2370036.2145833

[5] Y. Liu, L. Guo, J. Li, M. Ren and K. Li, "Parallel Algorithms for Approximate String Matching with k Mismatches on CUDA," 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops PhD Forum, 2012, pp. 2414-2422, doi: 10.1109/IPDPSW.2012.298.

[6] A. Halaas, B. Svingen, M. Nedland, P. Saetrom, O. Snove and O. R. Birkeland, "A recursive MISD architecture for pattern matching," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 12, no. 7, pp. 727-734, July 2004, doi: 10.1109/TVLSI.2004.830918.

[7] Knuth, D. E., Morris, Jr, J. H., Pratt, V. R. (1977). Fast pattern matching in strings. SIAM journal on computing, 6(2), 323-350.

[8] P. D. Michailidis and K. G. Margaritis, "Performance Evaluation of Multiple Approximate String Matching Algorithms Implemented with MPI Paradigm in an Experimental Cluster Environment," 2008 Panhellenic Conference on Informatics, 2008, pp. 168-172, doi: 10.1109/PCI.2008.13.

[9] Someswararao, Chinta. (2012). Parallel Algorithms for String Matching Problem based on Butterfly Model. International Journal of Computer Science and Technology.

[10] A. Tumeo, O. Villa and D. G. Chavarria-Miranda, "Aho-Corasick String Matching on Shared and Distributed-Memory Parallel Architectures," in IEEE Transactions on Parallel and Distributed Systems, vol. 23, no. 3, pp. 436-443, March 2012, doi: 10.1109/TPDS.2011.181.