LITERATURE REVIEW: Parallel String Matching

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

December 25, 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 DNA Sequencing or search engines or content search in large databases, a large amount of data inevitably takes a lot of time. Modern CPUs tend to have a large number of cores, and it is increasingly common to utilize GPUs for data processing. Parallelization has become an important part of algorithm design. Because of the data structure of String Matching, it can take benefits from Parallel Programming to solve large-scale Matching problems efficiently on large-scale systems.

This project is to explore the parallel implementation of traditional sequence string matching algorithms and implement its parallel version over cilk and Cuda, then compare the benefit of different string matching algorithms in different parallel architectures and the applications and limitations on a distributed architecture. In this project, I implement six different string-matching algorithms in CPU parallel and three in GPU. For algorithms that are difficult to algorithmically parallelize, I partition the index to perform parallel operations. Some of the algorithms take also use bit parallel or SSE instruction to speed up. I analyze The speedup obtained by different algorithms, and the speedup obtained in the worst case.

A more effective algorithm can be selected through pattern length analysis and preprocessing. Our results also discuss this point. The data type also has an impact on efficiency. A gene sequence with four letters has more repetitions than natural language, and this repetition will also greatly affect efficiency. We'll also compare different datasets and explore the differences to determine the best algorithm.

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[1]. There are different computer architectures according to Flynn classification, and for different architectures, there have different methods to

minimize latency and improve performance. Moreover, the CPU and GPU architectures also have different memory hierarchies, which need to be taken into account when designing string-matching algorithms. This study compares the advantages and disadvantages of different architectures in solving string-matching problems and measures the performance of CPU parallelism and GPU parallelism.

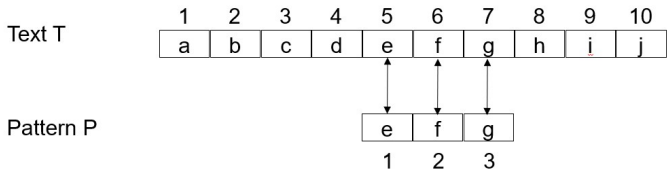

Figure 1: String Matching wants to find the pattern from Text

2.1 Sequential 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[1]. Since 1970, more than 80 String Matching algorithms have been proposed. These algorithms use preprocessing techniques to speed up the search process. In addition, algorithms such as the edit distance algorithm and suffix tree can be used to solve more complex string-matching problems. Based on previous research those algorithm deformations, combinations, and expansion, String Matching has been continuously optimized. Now, using parallelized algorithms to improve performance is a more mainstream method[1], by running the traditional algorithm in parallel, the original algorithm can be accelerated. The specific algorithm is listed in section 4 Implementation.. By exploring the details of these algorithms, the efficiency of parallelized algorithms can be further improved.

2.2 Parallel String Matching

Parallel computing has great potential and 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. Specific to the String Matching algorithm, It can be parallelized from data input, It is also possible to parallelize certain operations. In conclusion, parallel computing has great potential and can be used to speed up the calculation speed of string-matching algorithms. However, depending on the algorithm, the best core utilization can vary and it is important to analyze the speedup achieved in different numbers of threads.

Parallel String Matching base on MISD

Although there exist MISD architectures dedicated to pattern matching, such as Halaas's

recursive architecture [6], which can be used to solve string-matching problems, its performance is not competitive compared to other solutions. It can match up to 127 patterns at a clock frequency of 100 MHz, but this is not much better than a 1GHZ single-core CPU. Moreover, MISD processors 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 units on the GPU. Most Parallel String Matching benefits from SIMD architecture. Such as Chinta Someswararao's Butterfly Model [9], and some string matching algorithms are developed based on the SSE instruction set [1], for example, the SSEF algorithm. which is a SIMD instruction set developed by Intel.

Parallel String Matching base on GPU(CUDA)

Compared with the 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 [2], 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 [3]. 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. There are not many studies in this area. Hitoshi developed an algorithm called PZLAST used for MIMD processor PEZY-SC2 and compared the performance of BLAST+, CLAST, and PZLAST algorithms[11], which are specially optimized for the biological field. However, they didn't explore the percentage of parallel utilization of the algorithm.

Distributed Memory Parallel String Matching

There are few articles discussing the 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[10]. Results at the time showed that shared-memory architectures based on multiprocessors were the theoretical best performance, but there is an upper limit to the amount of parallelism, at 80 threads he starts to get degraded speedup, and beyond 96 threads the speedup becomes marginal. considering cost constraints, GPUs that did not reach the PCI-Express bandwidth limit had the best price/performance ratio. The performance of GPU has developed several times in the past ten years, and theoretically, its performance is far faster than that of CPU now. Although distributed can provide sufficient space and computing resources, limited by the cost of communication, the performance of distributed computing is not satisfactory. Especially for a single String Matching problem. Nonetheless, this can lead to considerable performance gains for multiple patterns search, as shown by Panagiotis' work[8], In a 16-processor cluster, the matching time can be reduced by half of the original.

3 Conclusion

Based on our research, string matching based on parallel algorithm has good performance in theory, especially the GPU-based method can achieve extremely high performance at a considerable cost, and what can multi-thread parallelism and GPU parallelism achieve? Speedup ratio, which is what I want to explore in this project.

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.

[11] H. Ishikawa et al., "PZLAST: an ultra-fast sequence similarity search tool implemented on a MIMD processor," 2021 Ninth International Symposium on Computing and Networking (CANDAR), 2021, pp. 102-107, doi: 10.1109/CANDAR53791.2021.00021.