Sequence Data Indexing Method Exploiting the Parallel Processing Resources of GPGPU

Taehwan Kim[1], Seokil Song[1], Dojin Choi[1], Yunsik Kwak[1], Bongen Goo[1]

[1] Dept. of Computer Engineering, Korea National University of Transportation, Chungbuk, Korea
rlaxoghks05@gmail.com, sisong@ut.ac.kr, mycdj91@naver.com, yskwak@ut.ac.kr
bggoo@ut.ac.kr

Abstract. In this paper, we propose a new sequence retrieval method that utilizes the parallel processing power of General-Purpose Computation on Graphics Processing Unit (GPGPU). GPGPU has high parallel computing power, due to its characteristic (SIMD) we need to carefully design a method to attain the parallel computing power. Existing CPU based sequence retrieval method does not consider the feature of GPGPU, so even though it is running based on GPGPU it is not able to fully use the parallelism of GPGPU. In this paper, we propose a sequence retrieval method with considering the features of GPGPU, and implement our proposed method, existing method and some variants of the proposed method. Finally, we compare them through experiments with real data.

Keywords: GPGPU, sequence data, indexing

1 Introduction

Sequence data has great significance in the commercial, scientific, and engineering sectors, and it plays an important role in decision-making and policy formulation. Real-world applications that deal with sequence data are very diverse and include multimedia searching, stock market data analysis, and sensor-based monitoring [1, 2]. Sequence data is generally analyzed through a similarity search for finding similar patterns. DTW (Dynamic Time Warping) is a representative method used for similarity searches within sequence data. The Euclidean distance represents each of the components constituting the sequence data in an independent manner. Thus, it is not suitable for measuring the distance between sequence data sets of different lengths or of different sampling rates.

DTW is a distance measurement technique that overcomes this problem. It is able to find the optimum distance between data sets by making it possible to expand or collapse the sequence data relative to the time axis. However, the complexity of DTW restricts its use in high-volume time series databases. In order to resolve this problem, a variety of indexing techniques have been investigated [1, 2]. They filter a sequence database without incurring a false negative to find a candidate set, and use DTW to find the final result.

However, most of existing methods are not proper for very large amount of sequence database that consists of very long sequence data. For example, we developed a sequence database system which is called HBase (Humming Base) with KETI (Korea Electronics Technology Institute). HBase manages very long sequence data that is extracted from music data. The average length of a sequence is about 5000. To enhance the query processing performance, HBase uses the dynamic sequence abstraction method that is the variation of [2]. HBase processes a query within a few seconds even when it is running on a high performance server computer with 16 cores.

In this paper, we propose a sequence data indexing method that utilizes the parallel processing power available in GPGPU. GPGPUs have two attractive features motivating the use for sequence data indexing. First, they are widely available. Generally, they can be used in desktop PCs and workstations. Second, a typical GPGPU have over an order of magnitude more processing power than a typical CPU. Importantly, technology trends support an accelerated growth in computational power for future GPGPUs indicating that GPUs will become an even more powerful and important resource to be utilized for general purpose tasks [3].

GSPRING [4] presented a parallelized DTW method based on NVIDIA's CUDA environment. To overcome the data dependency problem, it calculates the cells and fill the table along the diagonal direction. The degree of parallelism can be maximized to m, the length of query pattern, in optimal situations. Further parallelism is achieved by applying the computing model of CUDA. A two-dimensional array of blocks is declared and each block handles one single pair of stream comparison. Based on that fact that hundreds to thousands of comparisons can be processed simultaneously.

In this paper, we apply the GSPRING to HBase and optimize the GSPRING which reduces some branch statements. Then, we evaluate HBase with GSPRING and optimized GSPRING with real data.

2 Optimized GSPRING Method

The main ideal of GSPRING is to carefully consider the data-dependent nature of the DTW table, intuitive parallel programming methods are not easily applied. For example, in Figure 1(a), d(4, 8) depends on d(5, 4), d(5, 2) and d(1, 4), i.e., d(4, 8) can be calculated after d(5, 4), d(5, 2) and d(1, 4) are calculated. Therefore, a parallel DTW approach of GSPRING is achieved by implementing the computation flow proceeding in the diagonal direction

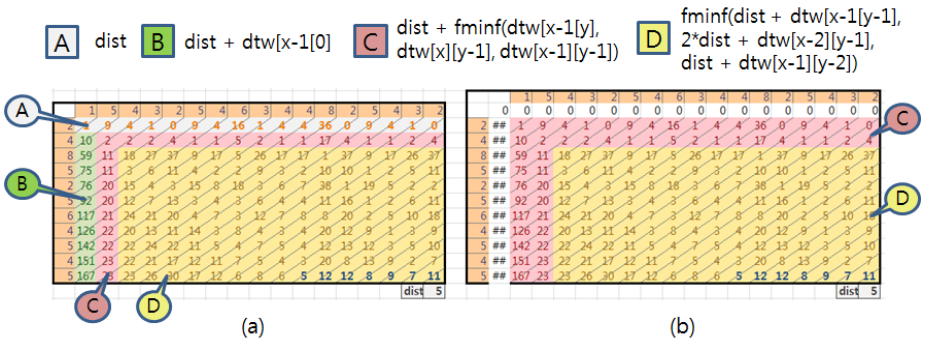


Fig. 1. gSPRING and optimized gSPRING

We apply the GSPRING to our HBase to enhance DTW calculation time. Also, we optimized the GSPRING as shown in Figure 1(b). As shown in the figure, GSPRING calculates A, B, C, D differently. It means that each thread may perform different commands in certain time, and it may increase operating time. So we fill initially zero into the DTW table's first row and column as shown in Figure 1(b). It eliminates A and B cases.

3 Performance Evaluation

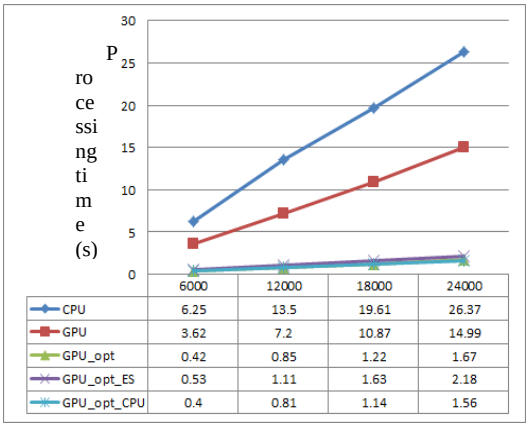
To evaluate the performance of subsequence matching using GPU, we perform several experiments. First we implement the HBase with GSPRING and optimized GSPRING using C language and CUDA on the CPU and NVIDIA GPU. The platform we use is shown in Table 1.

Table 1. Experimental environments.

OS	CentOS6.3 64bit, 2.6.32
CUDA Version	4.2
CPU	AMD Phenom(TM) II X6 1055T 800MHz (8 cores)
RAM	8GB
GPU	NVIDIA GTX670, 980MHz 1344 core, 6008MHz 2GB RAM

We implement 5 versions of HBase such as original HBase (CPU), HBase with GPU (GPU), HBase with optimized GSPRING (GPU_opt), HBase with optimized GSPRING and CPU (GPU_opt_CPU), HBase with optimized GPU and early stopping (GPU_opt_ES). CPU means that existing HBase that only use CPU, and GPU means that existing DTW algorithm of HBase is performed on GPU without considering parallelism, GPU_opt means that HBase with the proposed optimized GSPRING. GPU_opt_CPU uses GPU and CPU in parallel, and GPU_opt_ES employs early stopping algorithm [1] to GPU_opt.

We perform experiments with real data set provided by KETI (Korea Electronics Technology Institute). The data size is 24,000 and the number of queries is 20. We perform each query 5 times and measure average running time. Figure 2 shows the results of our experiments. As shown in the figure, GPU_opt_CPU shows the best performance. However, the difference between GPU_opt and GPU_opt_CPU is ignorable. The performance of GPU_opt_ES is not best even though the overall calculation is lower than others because the branch statements are more.



	6000	12000	18000	24000
CPU	6.25	13.5	19.61	26.37
GPU	3.62	7.2	10.87	14.99
GPU_opt	0.42	0.85	1.22	1.67
GPU_opt_ES	0.53	1.11	1.63	2.18
GPU_opt_CPU	0.4	0.81	1.14	1.56

Fig. 2. Processing time vs. data size

4 Conclusion

In this paper, we present the optimized GSPRING method and implemented the method based on HBase. Also we implemented various versions of HBase to evaluate the optimized GSPRING method. The performance evaluation showed that the optimized GSPRING is superior to others.

Acknowledgments. This research was supported by Technology Development Program for ('Agriculture and Forestry' or 'Food' or 'Fisheries'), Ministry for Food, Agriculture, Forestry and Fisheries, Republic of Korea.

References

1. Assent, I., Wichterich, M., Krieger, R., Kremer, H., Seidl, T.: Anticipatory DTW for Efficieint Similarity Search in Time Series Databases. In: Proceedings of the VLDB Endowment, pp. 826—837 (2009).
2. Sakurai, Y., Yoshikawa, M., Faloutsos, C.: FTW : Fast Similiarity Search under the Time Warping Distance. In: Proceedings of ACM PODS, pp. 326—337 (2005)

3. Govindaraju, N., Gray, J., Kumar, R., Manocha, D.: GPUTeraSort:high performance graphics co-processor sorting for large database management. In: Proceedings of SIGMOD, pp. 325—336 (2006)
4. Chang, J., Yeh, M.: Monitoring Multiple Streams with Dynamic Time Warping using Graphic Processors. In: Workshop Notes of the International Workshop on Parallel Data Mining in conjuction with SIAM DM (2011)