Accelerating Biological Sequence Alignment Algorithm on GPU with CUDA

Fang Zheng[1 2]
1 School of Computer
Wuhan University
Wuhan,China
2 School of Science,
Huazhong Agricultural
University,Wuhan,China
zf22_00@163.com

Xianbin Xu
School of Computer
Wuhan University
Wuhan,China
xbxu@whu.edu.cn

Yuanhua Yang[1 2]
1 School of Computer
Wuhan University
Whuhan,China
2 Jianghan Art
Vocational College
Qianjiang,hubei Province
P.R,China
yangyuanhua123@163.com

Shuibing He
1 School of Computer
Wuhan University
Wuhan,China
2 Laboratory of High
Confidence Software
Technologies (Peking
University), Ministry
of Education,Beijing，
China
hesbingxq@163.com

Yuping Zhang
School of Computer
Wuhan University
Wuhan,China

Abstract—In this paper, we have used Compute Unified Device Architecture (CUDA) GPU to accelerate pairwise sequence alignment using the Smith-Waterman (SW) algorithm. Smith-Waterman(SW) is by far the best algorithm for its accuracy in similarity scoring.But the executing time of this algorithm is too long in sequence alignment.So we describe a multi-threaded parallel design and implementation of the Smith-Waterman (SW) on CUDA to reduce execution time.And according the architecure of CUDA,we have divided the computation of a whole pairwise sequence alignment scoring matrix into multiple sub-matrices,using 32 threads to process on sub-matrice,more over we optimized memory distribution scheme, and used reduction to find the maximum element of the alignment scoring matrix.We experiment the algorimthm on GeForce 9600 GT,connet to Windows xp 64-bit system.The results show this mplementation achieves more better performance than the other parallel implementation on the Graphics Processing Unit.

Keywords- Sequence Alignment, Global Alignment, Local Alignment, Dynamic Programming, GPU, CUDA

I. INTRODUCTION

Sequence analysis finds similarity in biological sequences that is an import and widely used operation in bioinformatics. The purpose is to find the best possible alignment of a set of sequences. However, biological sequence alignment is also a computationally expensive application as the growth of the volume of sequence data, the sequence alignment algorithms are still very costly in performance .

Graphics Processor Units (GPUs) have been proposed recently as a high performance and relatively low cost acceleration platform for biological sequence alignment [1].As modern GPUs have become increasingly powerful, inexpensive and relatively easier to program through high level API functions, they are increasingly being used for non-graphic or general purpose applications (called GPGPU computing). In this paper, we will present how Compute Unified Device Architecture (CUDA) GPUs can be used to accelerate pairwise sequence alignment using the Smith-Waterman (SW) algorithm [2][3]. The SW algorithm uses a dynamic programming method to find the best local alignment between two query and database sequences, The time complexity of SW is O(mn)[4], where m and n are the lengths of the two sequences aligned. Due to this computational requirement ,many heuristic algorithms used for sequence alignment problem Heuristic methods may be faster than SW, however they loss accuracy in finding the best possible local alignments.

Smith-Waterman algorithm is widely used in many bioinformatical applications, either as the last stage of sequence similarity search performed with approximate algorithms [5, 6], or within more advanced algorithms,such as profile-profile methods [7]. Algorithms for sequence matching are also used in the so-called next generation sequencing methods [8, 9], which are used for rapid sequencing of whole genomes. These methods generate millions of relatively short sequences, which then need to be assembled. It has been shown that application of the exact SW algorithm as a part of the assembly algorithm significantly improves the quality of assembled sequence data [10].

In this paper we present an improved implementation of the SW algorithm on GPU with CUDA. The differences between the two implementations are discussed and suggestions for development of efficient codes are proposed.

The rest of this paper is organized as follows: section 2 describes the GPU architecture and Smith-Waterman algorithm. Our implementation of the algorithm on the GPU architecture is detailed in section 3. Section 4 contains our results and evaluation.Conclusion and future work is shown in section 5.

978-0-7695-4501-1/11 $26.00 © 2011 IEEE
DOI 10.1109/ICCIS.2011.61

II. BACKGROUND AND RELATED WORK

A. GPU Programming with CUDA

CUDA (Compute Unified Device Architecture) is a parallel computing architecture developed by NVIDIA Corporation [11].and allows to write and run general-purpose applications on the NVIDIA GPU's.CUDA uses threads for parallel exection,and GPU allows thousands of threads for parallel exection at the same time.

The GPU used in our implementation is NVIDIA's Geforce 9600GT,and the compute capability of it is 1.1.which has 8 Stream Multiprocessors. On the GPU, there is a hierarchy of memory architecture to program on it . As introduced in the CUDA programming guide, we present the memories in our implementation:

- Registers: Read-Write per-thread, each SP has its own registers (1024).
- Shared Memory: Read-write per-block, and its size is 16KB for each multiprocessor.
- Global Memory: Read-write per-grid, has a large space (about 768M) offers global access but it is a slower storage.
- Constant Memory: Read-only per-grid, its size is 8KB.
- Texture Memory: Read-only per-grid depends on the global memory.

In the memory architecture,the fastest memories are the shared memories and registers. They are on chip shared by all threads in a mutiprocessor,they can be read and written to by eachthread directly. but the size is limited. The other memories are all located on the GPUs main RAM. The constant memory is favorable when multiple processor cores load the same value from cache. Texture cache has higher latency but it has a better acceleration ratio for accessing large amount of data and non-aligned accessing.The memory architecture of GPU is described in Figure 1.To gain better performance, we must manage the shared memory,registers,and global memory usage.

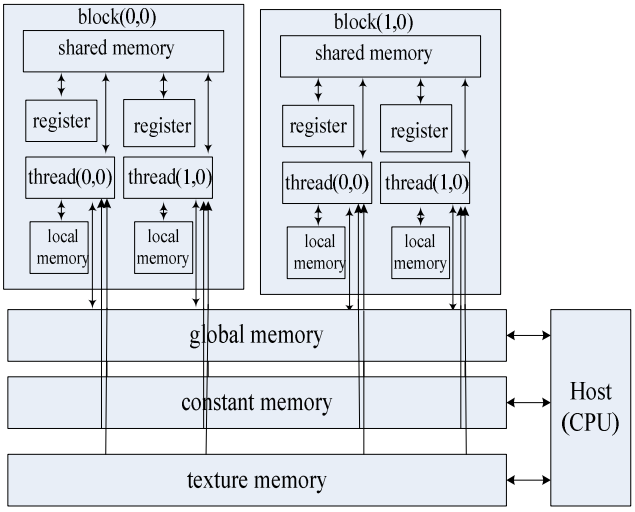

Figure 1.Memory architecture of NVIDIA's Geforce 9600 GT

B. Smith-Waterman Algorithm

The Smith-Waterman algorithm[3] is a dynamic programming method to compare between two alignment sequences (query and database sequence) to find the best local alignment. Smith Waterman Algorithm is one of a widely used algorithms in bioinformatics.The algorithm is achieved in two stages.First, using the initial conditions and equations of the two sequences to calculate the alignment score.Second,using the backtracing algorithm to get the alignment result. More specifically,let Q denotes a query sequence with a length of n:

$Q=q_0q_1q_2q_3\ldots\ldots q_{n-1}$

Let D denote the a database sequence with a length of m:

$D=d_0d_1d_2d_3\ldots\ldots d_{m-1}$

The equations for computing the alignment scores are as follows:

$$E(i,j)=max\{E(i,j-1)-G,H(i,j-1)-G\} \quad (1)$$

$$F(i,j)=max\{F(i-1,j)-G,H(i-1,j)-G\} \quad (2)$$

$$H(i,j)=max\{0,E(i,j),F(i,j),H(i-1,j-1)-W(qi,dj)\} \quad (3)$$

The algorithm is as following:

Let W(qi,dj) denotes the substitution matrix(be given) which gives a score describing the likelihood of substitution between qi and dj. G is the penalty for a mismatch.From these equations,we observe that the value of H(i,j) depends on the values of its upper neighbour H(i-1,j-1),left neighbour H(i-1,j) and left upper neighbour H(i,j-1),we discribe in Figure 2.

```
for i=0 to n do
    for j=0 to m do
        calculate H(i,j) use H(i-1,j-1) H(i-1,j) H(i,j-1)
end
```

Q \ D	A	C	T	…	…	T	G
G	H0,0	H0,1	H0,2				
A	H1,0	H1,1	H1,2				
…							
…							
T	Hn-2,0	Hn-2,1					
C	Hn-1,0	Hn-1,1					

Figure 2.alignment matrix of Smith-Waterman

Since GPU has the ability of allocating thousands of parallel threads to compute a task,so it is very suitable to accelerate SW algorithm. The GPU' computation is focused on the alignment scoring matric.From the equation(1)(2)(3)and Figure 2, we can see that the anti-diagonal element is independce from each other,so we use different threads to compute them parallely.

III. OUR IMPLEMENTATION OF SMITH-WATERMAN

As already analysised in section II, we can see that the anti-diagonal element is independent from each other,so we can use different threads to compute them parallely.There are many Smith-Waterman algorithm implementations on various parallel platforms [12] [13] [14] [15] [16],In GPU platform,some implementation is that each thread processes one alignment matrix between a database sequence and a query sequence. Or it uses a thread block to processes one alignment matrix. In our implementation,we divide the computation of a whole pairwise sequence alignment matrix into multiple sub-matrices .and we use 32 threads to process one alignment sub-matrix parallely. We will discuss it with more detail in this section.

A. Partion and Parallelization

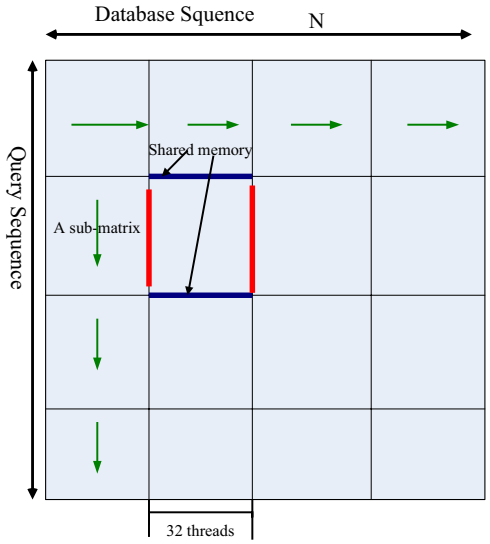

Figure 3. The partition and parallelization Method of aignment matrix

The alignment matrix is size D×Q (see Figure 3), D and Q are lengths of a database sequence and a query sequence. As D and Q could be very large,so we devided the matrix into many sub-matrices that the transfer of sub-matrix and the computation of the sub-matrix computation can be parallelized.

In our implementation ,we use 32 threads to compute sub-matrix once time in a multiprocess.As the 32 threads in one warp execute the same instruction,so threads in one warp are always synchronized.Using 32 threads(one warp) to process one sub-matrice can reduce the synchronization statement call to improve the efficiency.

From Figure 2, we can see that the anti-diagonal elements parallely could be computed parallely.So constant cache is used to store substitution matrix because that the substitutin matrix is used frequently and the latency of constant cache is lower than global memory.The database sequence is stored in the global memory as the huge size. As illustrated in Figure 2,the computation of the i-th row depends on the (i-1)-th row and the i-th row ,and the i-th row stored in the shared memory should be updated when the elements in the matrix are updated.In addition,the intermediate results of computing on the row are stored in the shared memory for the next sub-matrix's computation.Each thread stored the maximum of the row(i-th row) in the sub-matrice which it has computed in its register and transfered them to the global memory when the sub-matrice's computation is done.After calculating all sub-matrices,we can trace back to get the maximum element of the alignment matrix and return the result to the host.

The following is the procedure:

- Before the computing,tranfering the query sequence , the database sequence and substitution matrix to the shared memory and constant memory.
- The block of threads computes the sub-matrice with the value in the register and the shared memory ,storing the result of the last row and the last column in the shared memory for the next sub-matrice's computation while transferring the next sub-sequence from the global memory to the shared memory for the next computation.Each thread storingthe max of the row which it has computed in its register,and the max is updated in the computing process.
- Tracing back to get the maximum matching value of the whole alignment matrix and transfering the result to the host.

In our implementation,we use NVIDIA GeForce 9600GT which has 8 SMs and each SM has 8 Stream Processors(SPs). There are 8192 registers,16KB shared memory , 64KB constant memory in one SM and Geforce 9600 GT has 768M global memory.We haved discussed before that 32 threads have been used to process one sub-matrice,and there are $32^{\times}8=256$(each SM has 8 SPs) threads running parallely.So there will be 8 sequence alignments that can be processed at the same time.As there are 256 threads to be executed parallely in kernel,if the query squence is longer than 256,the kernel function has to be executed more times.

B. Reduction of Results

As the alignment matrix is computed by many sub-parallelogram,we use the optimized reduction to get the maximum element of the alignment matrix. Moreover we used loop unrolling in reduction,and this method can improve the efficiency.the following is the detailed discription:

- Each thread stored the maximum element of the sub-matrice which it has computed in the shared memory.
- As the computation of the sub-matrice is completed,using reduction method to get the maximum element of the whole alignment matrix.

C. Other optimization techniques

In our method,the computation of the alignment matrix has two nested loops.The outer loop allocates the thread block to compute whole the alignment matrix.The inner loop calculates the sub-alignment matrix of the allocated thread block .We use unrolling loop(a common means used to

enhance efficiency) to unroll inner loop avoiding divergengce judgement.This method improved the register usage.And more,we used parallel computation and data transmission,this method alos achieved speed-up of the algorithm.

IV. RESULT AND EVALUATION

We have tested our implementation on Intel(R) Core(TM) 920 with 64-bit Windows Xp OS.The computer aslo has NVIDA Geforce 9600 GT GPU.

We used the Swiss-Port protein sequence database (release 15.12,2009) to evaluate our implementation.And more we selected query sequence with the length ranging from 64 to 2048 amino acids to test the performance.We also used BLOSUM 45 as substitution matrix and the gap-penalty is -10.

First,we compared our experimental results with Liu's method[17], The evaluation of our implementation shows that average speed-up is about 1.3x,the results are presented in the Table I. However,the two implementations used different GPUs,So we compared the performance of our implementation with a CPU implementation of Smith-Waterman algorithm,Table IIpresents the comparative results.This shows that our GPU implementation is up to 19x compared with CPU implementation.

TABLE I. COMPARE OUR APPROACH WITH LIU'S IMPLEMENTATION

query length	comparing with Liu's execution time(sec)		
	our implementation	*Liu's*	*speedup*
64	5.42	19.5	1.3
128	10.37	25	
256	21.16	36.3	
512	45.58	59.2	
1024	90.21	105.1	
2048	183.67	197.9	

TABLE II. COMPARE OUR APPROACH WITH CPU'S IMPLEMENTATION

Query length	comparing with cpu's execution time(sec)		
	our implementation	*cpu*	*speedpu*
64	5.42	121.27	19
128	10.37	234.2	
256	21.16	483.9	
512	45.58	975.4	
1024	90.21	1675.6	
2048	183.67	3368.3	

V. CONCLUSION AND FUTURE WORK

Smith-Waterman is one of the most widely used sequence alignment algorithms.And as the rapidly increasing size of sequence databases,the requirement of the computer's computing power also is increased.This paper presented a better implementation of the Smith-Waterman algorithm on GPU with CUDA.We divided a alignment matrix calculation to sub-matrice calculation.and used thread block to calculate sub-matrix.We used 32 threads to calculate the sub-matrice and used the global memory and shared memory avoiding bank conflicts.And moreover,some optimization methods were used in our implementation such as the reduction and the loop-unrolling(avoiding divergengce judgement) ,these means improved efficiency obviously.

Future work will use CUDA to accelerate other biological sequence with GPUs,such as multiple sequence alignment.

VI. ACKNOWLEDGMENTS

This work is supported by upported by Fundamental Research Funds for the Central Universities (Grant No.3101012),and by Key Laboratory of High Confidence Software Technologies Program (Grant No.HCST201104).

REFERENCES

[1] M. Charalambous, P, Transcoso and A. Stamatakis. "Intial experiences porting a bioinformatics application to a graphics processor". In Processings of 10th Panhellenic Conference on Informatics, 2005.

[2] O. Gotoh, "An improved algorithm for matching biologicalsequences, " J Mol Biol, vol. 162, pp. 707−708, 1982.

[3] T. F. Smith and M. S. Watermann. Identification of common molecular subsequence. Journal of Molecular Biology, 147:196-197, 1981.

[4] K. Elissa, "Title of paper if known," unpublished.Sanchéz, F., Salamí E., Ramierez, A., Valero, M.: Performance Analysis of Sequence Alignment Applications. In: Proceedings of the IEEE International Symposium on Workload Characterization.

[5] Pearson W. and D. Lipman, Improved tools for biological sequence comparison. Proc Natl Acad Sci USA, 1988. 85: p.

[6] Rognes, T., PARALIGN: rapid and sensitive sequence similarity searches powered by parallel computing technology.Nucleic Acids Research, 2001. 29: p. 1647-1652.2444 - 2448.

[7] Rychlewski, L., et al., Comparison of sequence profiles.Strategies for structural predictions using sequenceinformation. Protein Science, 2000. 9: p. 232−241.

[8] Margulies, M., et al., Genome sequencing in microfabricated high-density picolitre reactors. Nature, 2005. 437(7057): p. 376-380.

[9] Mardis, E.R., The impact of next-generation sequencing technology on genetics. Trends in Genetics, 2008. 24: p. 133-141.

[10] Blazewicz, J., et al., A new algorithm for genome assembly from short reads in 1st-International-Conference-on-Information-Technology-IT-2008. 2008, IEEE: Gdańsk,Poland.

[11] "NVIDI A CUDA programming guide 2.3." [Online].Available: http://developer.download.nvidia.com/compute/cuda/23/toolkit/docs/NVIDIA CUDA Programming Guide 2.3.pdf

[12] M. Farrar, "Striped smith-waterman speeds database searches six times over other simd implementations," Bioinformatics,vol. 23, no. 2, pp. 156−161, 2007.

[13] A. Szalkowski, C. Ledergerber, P. Krahenbuhl, and C. Dessimoz, " SWPS3 - fast multi-threaded vectorized Smith-Waterman for IBM Cell/B.E. and x86/SSE2," BMC Research Notes, vol. 1, p. 107, 2008..

[14] M. Farrar, "Optimizing smith-waterman for the cell broadband engine. " [Online]. Available: http://farrar.michael googlepages.com/ SW-CellBE.pdf.

[15] S. Manavski and G. Valle, "CUDA compatible GPU cards as efficient hardware accelerators for Smith-Waterman sequence alignment," BMC Bioinformatics, vol. 9, no. Suppl 2, p. S10, 2008.

[16] K. Benkrid, Y. Liu, and A. Benkrid, "A highly parameterized and efficient fpga-based skeleton for pairwise biological sequence alignment," Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, vol. 17, no. 4, pp. 561−570,2009. [Online]. Available: http://dx.doi.org/10.1109/TVLSI.2008.2005314

[17] W. Liu, B. Schmidt, G. Voss, A. Schroder and W. Muller-Wittig. " Bio-Sequence Database Scanning on GPU". Inproceeding of 20th IEEE International parallel &distributed processing symposium: 2006 (IPDSP 2006)HICOMB workshop Rhode Island, Greece. 2006.