Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IMPACT FACTOR: 5.258

IJCSMC, Vol. 5, Issue. 5, May 2016, pg.122 – 128

A QUALITATIVE APPROACH OF ACCELERATED OPERATION ON GPU USING CUDA FOR LARGE DATABASE OPERATIONS

Vaibhavi Chavda[1], Mehul Patel[2], Hitul Patel[3]

[1]M.E in C.E, Swaminarayan College of Engineering &Technology (S.C.E.T., Kalol), India

[2]Asst. Prof. in Swaminarayan College of Engineering & Technology (S.C.E.T., Kalol), India

[3]HOD of ME(CE) in Swaminarayan College of Engineering & Technology (S.C.E.T., Kalol), India

vaibhavi.chavdaddit@gmail.com; mehul.patel@gmail.com; hitulce@gmail.com

ABSTRACT - GPU provides a vast number of simple, data-parallel, deeply multithreaded cores and high memory bandwidths, GPU Architecture are becomes more approachable due to its fast computing speed, the reason behind fast computational speed and improved time complexity is its multi-core Architecture. In today's time we heard about dual core or quadrate core computers which have two or more core. What is core? A core is usually the basic computation unit of the CPU - it can run a single program context maintaining the correct program state, registers, and correct execution order, and performing the operations through ALUs.So we can imagine if we have hundreds of core in our computer then we can achieve great computational speed, we can able to transfer large amount of data in a less time compare to CPU. CUDA technology leverages the massively parallel processing power of NVIDIA GPUs. The CUDA architecture is a revolutionary parallel computing architecture that delivers the performance of NVIDIA's world-renowned graphics processor technology to general purpose GPU Computing. I will use NVIDIA's C-like CUDA language to explore the effectiveness of GPUs for a variety of application types, and research some algorithms that improve their performance on the GPU. I will also find advantages and inefficiencies of the CUDA programming model by implementing Smith Waterman Algorithm in CUDA.

Keywords— CUDA, GPU, Smith-wateman algorithm, Cost matrix, Gap penalty, Affine gotoh gap panelty

I. INTRODUCTION

This paper focuses on smith waterman algorithm but with some improved change. Smith water man algorithm takes input query and search it into databases and files, previous work and papers shows that traditional smith waterman algorithm wont process large size file for query searching. With adding Affine gap penalty with gotoh function which is very efficient at assigning cost matrix and finds best optimal solution of searching query using heuristic approach. More important with using advance speed of GPU and Using CUDA programming Model we can advance the speed further by combining the approach algorithm and GPU(CUDA).

II. **Introduction to GPU**

In recent years, the computation speed of graphics processing unit (GPU) has increased rapidly. We only take the floating-point operation as an example, and the computation speed of GPU is several times faster than CPU's. The Flops of NVIDIA Ge80 series has gotten 520G in late 2008. whereas Intel 64-bit dual-core CPU has only 32 Glops. Moreover, the mainstream GPU's scale has exceeded significantly than the CPU's now. The transistor number of NVIDIA Quadra FX 5600 has been more than 0.7 billion. From above, we can see the robust computational capability of the GPU. Moreover, as the programmability and parallel processing emerge GPU be used only by the professional people familiar with graphics API, and brings many inconveniences to the common users. The term GPU was popularized by NVidia in 1999, who marketed the GeForce 256 as "first GPU(Graphics Processing Unit)" a single-chip processor within targeted transform, lighting, triangle setup/clipping, and rendering engines that are capable of processing a minimum of 10 million polygons per Rival ATI Technologies coined the term Visual Processing Unit(VPU) with the release of the Radeon 9700 in 2002. Furthermore, GPU-based high performance computers are starting to play a significant role in large-scale modeling. Three of the 10 most powerful supercomputers in the world take advantage of GPU acceleration. The amount of cores that GPUs have depends on the manufacturer. NVidia graphics solutions tend to pack more power into fewer chips, while AMD solutions pack in more cores to increase processing power. Typical high-end graphics cards have 68 cores if it's nVidia, and ~1500 cores if it's AMD. Architecturally, the CPU is composed of a only few cores with lots of cache memory that can handle a few software threads at a time. In contrast, a GPU is composed of hundreds of cores that can handle thousands of threads simultaneously. The ability of a GPU with 100+ cores to process thousands of threads can accelerate some software by 100x over a CPU alone.

III. **Introduction to CUDA**

CUDA is an extension to C based on a few easily-learned abstractions for parallel programming and a few corresponding additions to C syntax. CUDA represents the coprocessor as a device that can run a large number of threads. The threads are managed by representing parallel tasks as kernels mapped over a domain. Data is prepared for processing on the GPU by copying it to the graphics board's memory. Data transfer is performed using DMA and can take place concurrently with kernel processing. CUDA gives program developers direct access to the virtual introduction set and memory of the parallel computational elements in CUDA GPUs.

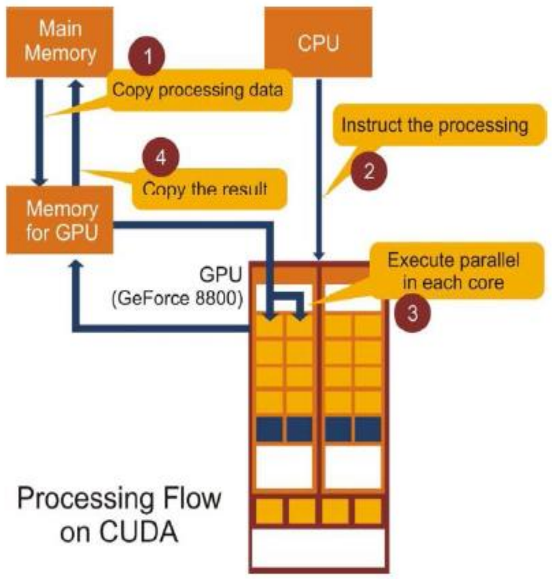

Fig. 1 Flow on CUDA with CPU and GPU

IV. **Introduction to Smith-Waterman**

Performing sequence alignment in CPU can be very time consuming and it may not gave you the expected result, Using GPU we can get the high speed up result by comparing interesting sequence with huge sequence database or local data in GPU. We need filtration also for eliminating law compared sequence by aligning threshold value which would frequency distance. Due to dynamic programming technology, the time complexity of these algorithms is **O**(*n*2). Comparing an interesting sequence with a sequence database including more than thousands of sequences, the total computing time can be very high. With the rapid growth of the biotechnology, the number of sequences is increasing at breakneck speed in the database. GPGPU programming has been successfully utilized in the scientific computing domains that involve a high level of numeric computation. Different to the multi-core CPU, GPGPU has large number of cores. In 2006, NVIDIA gave a programming architecture, Compute Unified Device Architecture (CUDA), to increased GPU computing power. Due to the computational power of GPU, the SW algorithm has implemented by using CUDA . there are two different GPU-based SW algorithms, intertask parallelization and intra-task parallelization, have been defined to compare a query sequence with a number of database sequences. Their experimental results presented that GPU can dramatically improve the alignment performance over CPU. Inter-task parallelization is that a thread performs a task. Therefore the number of threads is equal to the number of the tasks. Intratask parallelization is that the threads of a black perform a task in parallel. Hence the number of blacks is equal to the number of tasks. In general, the performance of inter-task parallelization is better than the intra-task parallelization.so, most of previous works accept inter-task parallelization to design SW algorithm on GPU. But inter-task parallelization need more device memories. Intra-task parallelization is suitable for comparing long sequences in a real application.

V. **Method**

Step 1 : Fill in the dynamic programming matrix
Step 2 : Find the maximal value (score) and trace back the path that leads to the maximal score find the optimal local alignment.

a. Based-pair sequence matching function
   i. F(0, j) = F(i, 0) = 0
   ii. F(i, j) = max { F(i – 1, j) – d F(i, j – 1) – d F(i – 1, j – 1) + s(x , yj) }
b. Insertion function
c. Deletion function
d. Calculate_ fine function
   i. Implement gotoh function for gap panelty
      1. Ak, l : g(k + l) _ g(k) + g(l):
      2. g(k) = a+ Bk
   ii. y(n): y(n) for all n, y(n + 1) - y(n) <= y(n) - y(n – 1)
a. Find maximum_gap among all.
b. O(N 3) time, O(N 2) space
e. Optimal score function.
   i. F(i, j): score of alignment x1…x to y i 1…yj if xi aligns to yj
   ii. G(i, j): score if x , or yj i , aligns to a gap

VI. **Related Work and Algorithm**

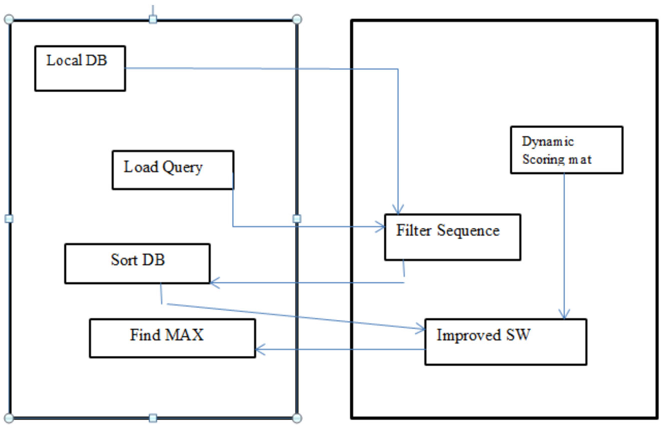

Fig 2 Improved SW Algorithm

- A regular gap extension method would assign a fixed cost per gap, so it is not the optimal solution because if matching length is small or long the gap remains same while filing gap matrix
- Affine gap penalties provide incentive for the alignment algorithm to keep sequence together where possible rather than inserting millions of small gaps. We have three possibilities
  a. *No gap* **b**. *Starting new gap* **c**. *Alongate an existing gap*

Saving time because:

- SW – test with all possible gap lengths
- Gotoh affine :just add β if a gap become Longer
- Gotoh function : $\alpha + \beta k$
  α- cost required to open a gap of any length
  β-the cost to extend the length of an existing gap
  k- the length of the gap

So we can say that it gives optimal solution rather than using regular gap penalty. Here I explain how maximum speed can achieved analytically Now we will discuss how I have include large database and how we get our output.
First of all let's focus on why we need FASTA for our proposed work. As I have explained earlier what is FASTA in Background theory, I will just explain its basic method and approach with large database, Because for finding large database files and process it into GPU we need FASTA.FASTA is First fast sequence searching algorithm for comparing a query sequence against a database. Derived from logic of the dot plot compute best diagonals from all frames of alignment .The method looks for exact matches between words in query and test sequence, After all diagonals are found, tries to join diagonals by adding gaps. Computes alignments in regions of best diagonals. FASTA finds exact local matches and then extends them to get a global alignment. FASTA may be better for less similar sequences. BLAST is a fast, heuristic search tool for sequence databases. BLAST searches involve finding ungapped, locally optimal sequence alignments. BLAST compare an amino acid query sequence against a protein sequence database or a nucleotide query sequence against a nucleotide sequence database, as well as other combinations of protein and nucleic acid comparisons. Both FASTA and BLAST programs universally used to approximate local alignment and local similarity The Algorithm can also be implemented with various forms of parallelization in software running on more common hardware. As Smith waterman is based on FASTA and BLAST. The optimal local alignment score of two sequences can be computed using dynamic Programming approach. I will use smith waterman algorithm with the modification of gotoh for affine penalty functions. The affine gap cost model penalizes insertions and deletions using a linear function in which one term is length independent, and the other is length dependent. A regular gap extension method would assign a fixed cost per gap. An affine gap penalty encourages the extension of gaps rather than the introduction of new gaps. n number of database will be all simultaneously compared to the same query residual of n numbers. The operation will be carried out using vectors of N number independent bytes. The N residues are fed into N independent channels. It loaded one after another into channels. It means when the first of these N database sequence ends, the first residues of the next database sequence is loaded into the channel. The database sequences are read in the order they are found in the original database file. But the database is not sorted by sequence length. Which produces 4 containing 16 sequence block each score profile,64 database sequence. This is how temporary score profile is generated for 64 database sequence residual. Here Acceleration is achieved by parallelizing the SW local alignment algorithm using CUDA.Memory accesses are highly distributed. Database first converted to FASTA format by a script and then formatted to NCBI format database version into the NCBI BLAST binary database format.

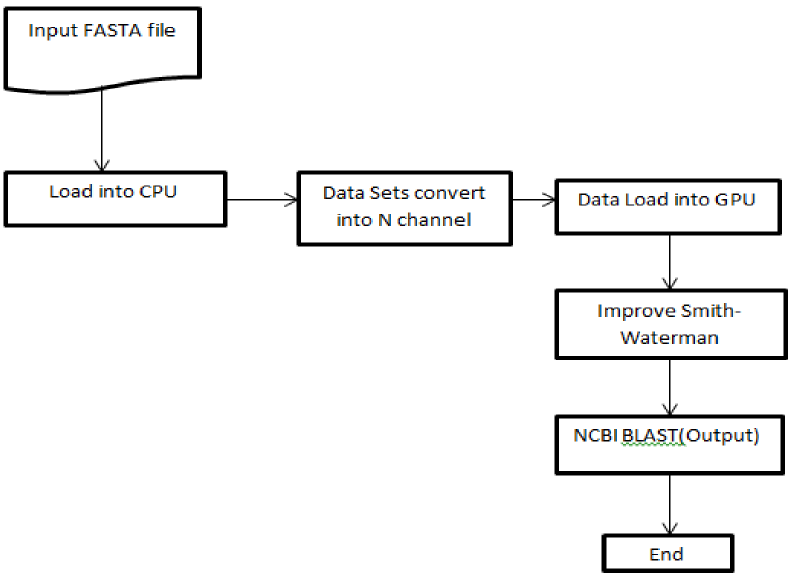

Fig 3. Process flaw of proposed work

C/C++ platform extended by NVidia's CUDA to implement the project. Some External libraries like THRUST, Accelereyes or CUDAfy can be used depending time constraints and learning curve**.** Netbeans or the Eclipse IDE will be used primarily for development in Linux distributions, or Visual Studio 2012 with CUDA SDK & Nsight(GPU code Debugger and Profiler) installed. I used uniprot database while is Swiss-Prot reviewed sequence data sets. Downloaded from uniport.org. Residues are retrieved from the above said database.

framework is highly scalable. To increase capacity of our system all that needs to be done is to add new nodes to the Hadoop cluster. They have proposed a schema to store RDF data in plain text files, an algorithm to determine the best processing plan to answer a SPARQL query and a cost model to be used by the algorithm.

Result

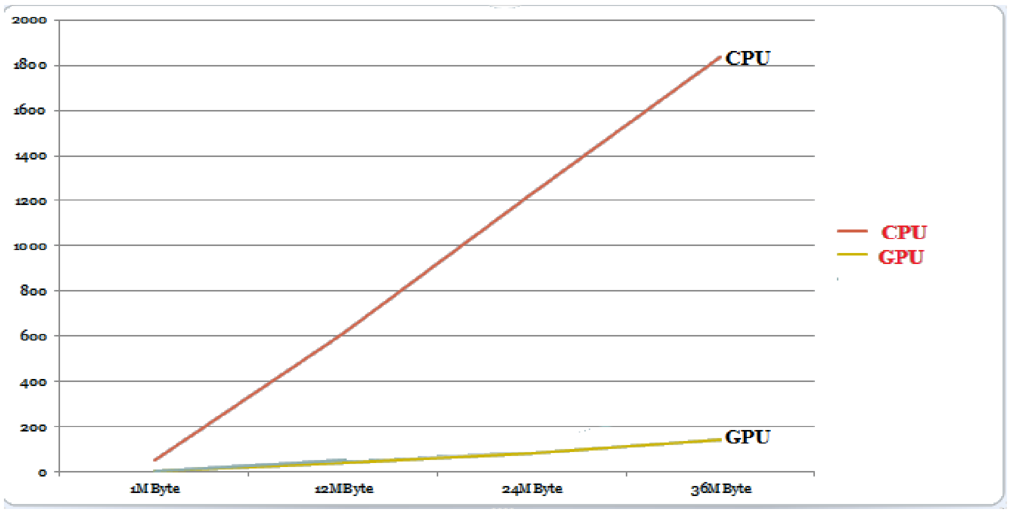

Fig 4.Execution time of CPU and GPU

Here I have attaché The result I get by performing the algorithm, In which Horizontal axis mention size of file given the algorithm and vertical axis mentioned ms(milliseconds)

CONCLUSION

Improved Smith-Waterman Algorithm is basically Smith-Waterman Algorithm but due to its drawback I ave madde some Improvement in the Algorithm to make it very approachable and Improved version on CUDA.Furthur it will implement on CUDA and GPU so that we can use multi-core functionality of GPU to achieve great computational speed. We all know GPU have lots of Multicore and thread based Architecture which is famous for great computational speed. I have combine the algorithm and GPU to make Huge data alignment and with a query sequence by using frequency distance filtration mechanism. The filtration mechanism is performed on CPU, and the SW alignment is performed on GPU. By combining these two parts, the performance can be enhanced significantly The experimental results showed that the highest speedup ratio is about 80 to 90 times over CPU-based SW algorithm implemented than Improved SW on CUDA and test the result But in theoretically it shows great speedup ration and improved time complexity while dealing with huge database.

ACKNOWLEDGEMENT

The authors would like to thank the reviewers for their precious comments and suggestions that contributed to the expansion of this work.

REFERENCES

[1] Efficient GPU-Based Algorithm for Aligning Huge Sequence Database 2013 IEEE International Conference on High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing.
[2] GPU Acceleration Using CUDA Framework International Journal of Innovative Research in Computer and Communication Engineering
[3] An Efficient GPU Implementation of Ant Colony Optimization for the Traveling Salesman Problem 2012 Third International Conference on Networking and Computing
[4] A GPU-BASED HARMONY K-MEANS ALGORITHM FOR DOCUMENT CLUSTERING Zhanchun Gao, Enxing Li, Yanjun Jiang Beijing Key Lab of Intelligent Telecommunication Software and MultimediaBeijing University of Posts and Telecommunicationsgaozc@bupt.edu.cn, enxing.li.bupt@gmail.com
[5] GPU-based Parallel R-tree Construction and Querying, 2015 IEEE International Parallel and Distributed Processing Symposium Workshop
[6] M. Kunjir and A. Manthramurthy, "Using Graphics Processing in Spatial Indexing Algorithms," *Research report, Indian Institute of Science, Database Systems Lab*, 2009.
[7] K. Zhou, Q. Hou, R. Wang, and B. Guo, "Real-time kd-tree construction on graphics hardware," in *ACM Transactions on Graphics (TOG)*, vol. 27, no. 5. ACM, 2008, p. 126.
[8] Q. Hou, X. Sun, K. Zhou, C. Lauterbach, and D. Manocha, "Memoryscalable gpu spatial hierarchy construction," *IEEE Transactions on Visualization and Computer Graphics*, 2010.
[9] S. Popov, J. G¨unther, H. Seidel, and P. Slusallek, "Stackless kd-tree traversal for high performance gpu ray tracing," in *Computer Graphics Forum*, vol. 26, no. 3. Wiley Online Library, 2007, pp. 415–424.
[10] D. Benson and J. Davis, "Octree textures," in *ACM Transactions on Graphics (TOG)*, vol. 21, no. 3. ACM, 2002, pp. 785–790.
[11] G. Ziegler, R. Dimitrov, C. Theobalt, and H. Seidel, "Real-time quadtree analysis using HistoPyramids," *Real-Time Image Processing 2007*, vol. 6496, 2007.
[12] NVIDIA Corp., "CUDA ZONE."
[13] D. Man, K. Uda, H. Ueyama, Y. Ito, and K. Nakano, "Implementations of parallel computation of Euclidean distance map in multicore processors and GPUs," in *Proceedings of International Conference on Networking and Computing*, 2010, pp. 120–127.
[14] K. Ogawa, Y. Ito, and K. Nakano, "Efficient Canny edge detection using a GPU," in *Proceedings of International Workshop on Advances in Networking and Computing*, 2010, pp. 279–280.
[15] M. Dorigo, "Optimization, learning and natural algorithms," Ph.D. dissertation,
[16] M. Dorigo, V. Maniezzo, and A. Colorni, "The ant system: Optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics–Part B*, vol. 26, no.1, pp. 29–41, 1996.
[17] T. St¨utzle and H. H. Hoos, "MAX–MIN ant system," *Future Generation Computer Systems*, vol. 16, no. 8, pp. 889–914, 2000.
[18] M. Dorigo and L. M. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53–66, April 1997.

[19] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. The MIT Press, 2001.
[20] M. Manfrin, M. Birattari, T. St¨utzle, and M. Dorigo, "Parallel ant colony optimization for the traveling salesman problem," in *Proc. Of 5th International Workshop on Ant Colony Optimization and Swarm Intelligence*, vol. LNCS 4150. Springer-Verlag, 2006, pp. 224–234.
[21] P. Delisle, M. Krahecki, M. Gravel, and C. Gagn´e, "Parrallel implementation of an ant colony optimization metaheuristic with openmp," in *Proc. of the 3rd European Workshop on OpenMP*, 2001.
[22] GPU Acceleration Using CUDA Framework Pratul P Nambiar1, V.Saveetha2, S.Sophia3, V.Anusha Sowbarnika 1 UG Student, Department of IT, Info Institute of Engineering, India. 2, 4 Assistant Professor, Department of IT, Info Institute of Engineering, India. 3 Professor, Department of ECE, Sri Krishna College of Engineering & Technology, India.
[23] N. Bandi, C. Sun, D. Agrawal, and A. El Abbadi. Hardware acceleration in commercial databases: a casestudy of spatial operations. In VLDB '04: Proceedings of the Thirtieth international conference on Very largedata bases, pages 1021{1032. VLDB Endowment, 2004.
[24] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Shea_er, and K. Skadron. A performance study of general-purpose applications on graphics processors using cuda. J. Parallel Distrib. Comput., 68(10):1370{1380, 2008.
[25] J. Dean and S. Ghemawat. Mapreduce: simpli_ed data processing on large clusters. Commun. ACM, 51(1):107{113, 2008.}
[26] A. di Blas and T. Kaldeway. Data monster: Why graphics processors will transform database processing. IEEE Spectrum, September 2009.
[27] S. Ding, J. He, H. Yan, and T. Suel. Using graphics processors for high performance IR query processing.