**Members:**

**18BCE0912-MALLA JYOTSNA**

**18BCE0854-PREETHAM LEKKALA**

**Parallelization and Detection of randomly generated Nucleotide Sequences using Multiple Threads**

# Abstract

The future of technology is the development and enhancements in parallel and distributed computing. All products and their fundamental concepts have shifted to parallel computing models. Everyone agrees that serial computing is easy to use and implement, but not efficient enough for industry-level purposes. Thus a higher number of industries are providing and using cloud solutions that work based on parallel and distributed computing. For instance, Amazon’s AWS or Google’s Google Cloud platform is becoming the center for development, may it be in the field of web development, or the field of data analytics. This study is focused on being familiarized with the domain owing to the fast-improvement in technology. Gene sequencing can aid researchers in optimum processing and efficiency regarding optimized system models. Computational science and bioinformatics has become the most recent multidisciplinary field which clarifies numerous parts fields of software engineering, while computational science outfits its methodology, advances for reacting to organic inquiries helpfully.

**1. Introduction**

Genome is a developing field, continually introducing numerous new difficulties to scientists in both organic and computational parts of utilization. Succession examination is an extremely fundamental and significant activity. They identify similar parts between two successions known as the inquiry arrangement and the reference grouping[1]. The worldwide and close by courses of action are the most prevalent sorts of collection game plan. In the overall course of action, we find the prevalent accomplice between parts of the progressions. On the other hand, close by course of action counts endeavor to facilitate bits of progressions and not the entire of them. The local arrangement is quicker than worldwide arrangement, because of the absence of a need to adjust the whole successions[2]. In our venture, we would execute the Smith-Waterman Algorithm in a sequential and equal way for examination and investigation. As sound judgment recommends, equal usage ought to execute and give a similar outcome as the sequential usage however in a lesser measure of time. The future demands more and more orientation towards Parallel and Distributed Computing. Many of the tasks that were once carried out sequentially are now being carried out in parallel to use resources more efficiently and get faster results[3]. Work is being done in biology with the application Smith-Waterman algorithm it is possible to process and understand nucleic acid/ protein sequences[4]. Thus, in this project, we use this inspiration to create something on a smaller scale, but with a large scope. An exceptionally productive and basic execution of the Smith-Waterman calculation for a nearby arrangement of groupings. In the study, it would chiefly how to distinguish groupings in proteins and nucleic acids and how to recognize them utilizing equal processing utilizing different strings. It would also learn how to apply parallel processing into a real-world problem – something we don’t have the opportunity to do in our university curriculum. There would be several limitations – we would only be creating a basic model and not a full-fledged model that will be 100% accurate[5]. At the end of the project term, we should be able to deliver a working model that should be able to provide accurate detection of sequences present. the future demands more and more orientation towards Parallel computing. Many of the tasks that were once carried out sequentially are now being carried out in parallel to use resources more efficiently and get faster results. Thus, in this project, we use this inspiration to create something on a smaller scale, but with a large scope[6].

An exceptionally productive and basic execution of the Smith-Waterman calculation for a nearby arrangement of groupings. In the project, it would chiefly how to distinguish groupings in proteins and nucleic acids and how to recognize them utilizing equal processing utilizing different strings. It would also learn how to apply parallel processing into a real-world problem – something we don’t have the opportunity to do in our university curriculum. There would be several limitations – we would only be creating a basic model and not a full-fledged model that will be 100% accurate. At the end of the project term, we should be able to deliver a working model that should be able to provide accurate detection of sequences present[7].

**2. Literature Survey**

Analysts use tools based on heuristic-based based sequence alignment tools due to limitations existing in the SW algorithm which results in sensitivity loss. Blast, RPAlign withSW can be used for sequences that are distantly related and remain anonymous. The proposed method will align the mega base-scale sequences reasonably fast with high sensitivity where SW becomes intractable computationally. It has been located that the proposed method can grant an alignment fantastic related to that of the SW algorithm while requiring extensively less time[8]. BLAST has increasingly low sensitivity for the sequences which are distantly related and therefore it does not reflect the exact biological evidence. However, it has decreased the runtime when compared to high-quality SW implementation[9]. The local alignment pairwise can be calculated between a subject and a query sequence using the SW algorithm which is an important algorithm in Bioinformatics. Many GPU algorithms depend on parallelism among multiple pair alignments and scale-up. The SW algorithm parallelizes each segment. This algorithm allocates the iterative computations elements of a matrix uniquely to recompute easily the data required instead of waiting for other processors[10]. Resulting in an algorithm for pairwise sequence alignment of two large sequences using a parallel version of the Smith-Waterman algorithm on GPU. Their way also either decreases or fully terminates the inter-block synchronization on the GPU and makes better use of its memory resources. The heuristic algorithms such as BLAST and FASTA which were developed for speed are not a total replacement for the slower but more rigorous algorithms such as SmithWaterman. Filtering dissimilar sequences can be done by a heuristic from large databases such as GenBank and the Smith-Waterman algorithm performs a much more detailed, in-depth alignment in a way that cannot be handled by the other heuristic methods[11]. To improve the speed and efficiency, a developed associative algorithm with further parallelization reduces the running time for rigorous pairwise local sequence alignment. To implement the Smith-Waterman algorithm, the huge computational power of NVIDIA’s GPUs is used to achieve high solutions for local sequence alignment[12]. NVIDIA CUBA GPU is implemented using parallelized SmithWaterman along with the wavefront algorithm. the sequence alignment algorithm on the multiple GPUs achieves better performance than the Smith-Waterman algorithm on the CPU and the single GPU[13]. MSA are multiple GPUs that have become highly efficient and have become an integral workload because of the growth of computational and bioinformatics industries. There is less understanding of the characteristics and performance and MSA despite the enormous attention given to the MSA design. We analyze the performance and characteristics of MSA software from the perspective of multicore machines using several MSA programs that use a wide variety of alignment approaches. Parallel programming approaches using threads and MPI are used to map parallelism in multicore machines[

Parallelism in multicore machines is mapped using parallel programming approaches incorporating threads and MPI[14].

Researchers have not had much success in tackling the problem of gene sequencing, which is one of the most predominant issues faced by them. They haven’t been able to make an optimized system model that could facilitate optimum processing and efficiency without implementing overhead. This problem could be tackled by developing such kinds of a system by using a dynamic programming approach called the enhanced form of the Smith-Waterman algorithm along with other supporting and optimized techniques[15]. A highly robust and efficient parallel computing development of a system is possible with the SW algorithm. Given two sequences it calculates the local alignment of sequences used to identify similar DNA, RNA, proteins. It gives a deeper understanding of the approach for gene sequencing and alignment using SW. SW uses a dynamic programming approach to detect the local alignments of biological gene pairs.

For the alignment of multiple biological sequences, a new iterative parallel algorithm has been proposed. The best MSA (multiple sequence alignment) of a set of sequences can be detected using a simultaneously running independent process. Longest Common Subsequence (LCS) can be used to generate the first MSA. Iteration can be used to improve MSA by applying a number of operators so as to produce alignments more accurately[16].

**3. Methodology**

The current issue was handled with a measured methodology. Eight capacities were built, every one of which would be clarified as follows: element – This capacity is utilized to compute the number of components that have been found by the Smith-Waterman Algorithm. Three conditions are given: One of which is to see whether the number of components in the corner to corner is expanding, diminishing, or stable. calcFirstDiagElement – This capacity is utilized to ascertain the situation of the greatest scored incentive in the lattice. This worth should be found because the algorithm recommends that the backtracking to discover the way ought to be begun from this specific point. similarity score – This capacity is utilized to discover the ideal request of execution dependent on three conditions, which are utilized to compute the new estimations of left, upper, and corner components. Every cycle, the estimations of the greatest component is refreshed and embeddings into the similitude and ancestor frameworks. matchMismatchScore – This capacity is utilized to ascertain a closeness work or the letters for a match or bungle. If the estimation of the two components is equivalent, it is a match, in any case, it's a confusion. Backtrack – The motivation behind this capacity is to alter the grid that should be printed and encourages us to distinguish the way to be taken to get the most ideal arrangement. printMatrix – It's a circled cycle usage to show the framework. printPredecessorMatrix - It is in this capacity where we print the bolts delineating the way of nearby arrangement. Generate – This capacity produces the two successions An and B which would be privately lined up with one another. An irregular seed is utilized to guarantee the reproducible idea of yield.

## *3.1 Tools Used*

The C programming language gives numerous standard library capacities to record information and yield. These capacities make up the greater part of the C standard library stdlib.h is the header of the universally useful standard library of C programming language which incorporates capacities including memory designation, process control, changes, and others. It is perfect with C++ and is known as cstdlib in C++. The name "stdlib" means "standard library" The math.h header characterizes different scientific capacities and one full scale. All the capacities accessible in this library accept to serve as contention and return serve as the outcome. It is a library that permits memory multiprocessing programming in C.In C programming language time.h (utilized as ctime in C++) is a header record characterized in the C Standard Library that contains time and date work presentations to give normalized access to time/date control and design.

## *3.1 Smith-Waterman Algorithm*

Smith-Waterman calculation figures the nearby arrangement of two groupings. It certifications to discover the most ideal nearby arrangement considering the predetermined scoring framework. This incorporates a replacement lattice and a hole scoring technique. Scores think about match, jumble, and replacement. To quantify the correlation between two groupings, a score be determined as follows: Given an alignment between sequences S0 and S1, the following values must be assigned, for each column.

**3.2 WORKING OF SMITH WATERMAN ALGORITHM**

Initialization of the framework and consider two successions An and B. • Matrix loading up with the reasonable scores. The two arrangements are set in a framework structure by methods for A+1 sections and B+1 columns. The incentive in the mainline and first segment is set to zero. The second and basic advance of the calculation is filling to the whole framework. To fill every single cell it is imperative to know the corner to corner esteems. . • Trace back the arrangement for a proper arrangement is follow backing; before that the most extreme score got in the whole framework must be identified for the nearby arrangement of the successions. • It is probably going to those most extreme scores that can be available in at least one than one cell, in such case there might be choice of at least two arrangements, and as well as can be expected to be acquired by scoring it. • Tracing back starts from the position which has the most noteworthy worth, pointing back with the pointers, subsequently discover the conceivable forerunner, at that point go to the next antecedent and proceed until it arrives at the score 0.

**3.3 WORKING OF SMITH WATERMAN ALGORITHM**

The current issue was handled with a measured methodology. Eight capacities were built, every one of which would be clarified as follows:

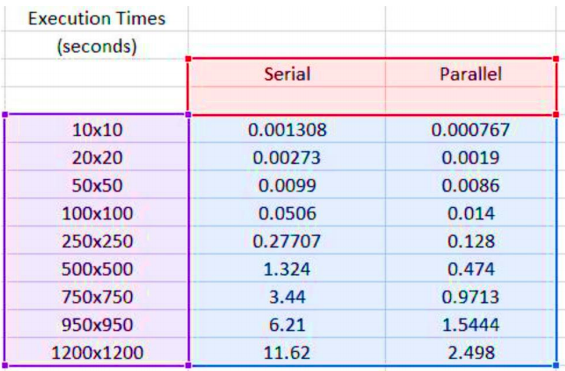
element – This capacity is utilized to compute the number of components that have been found by the Smith-Waterman Algorithm. Three conditions are given: One of which is to see whether the number of components in the corner to corner is expanding, diminishing, or stable. calcFirstDiagElement – This capacity is utilized to ascertain the situation of the greatest scored incentive in the lattice. This worth should be found because the algorithm recommends that the backtracking to discover the way ought to be begun from this specific point.

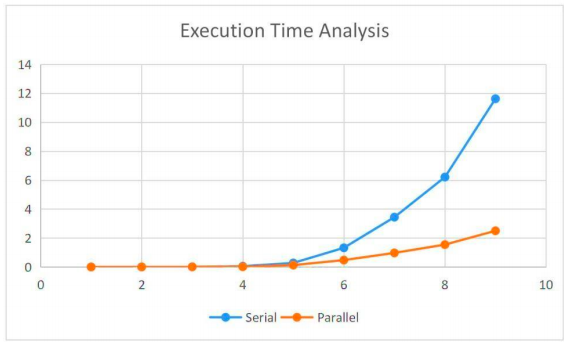
similarity score – This capacity is utilized to discover the ideal request of execution dependent on three conditions, which are utilized to compute the new estimations of left, upper, and the corner to corner components.

Every cycle, the estimations of the greatest component is refreshed and embeddings into the similitude and ancestor frameworks. matchMismatchScore – This capacity is utilized to ascertain a closeness work or the letters for a match or bungle. If the estimation of the two components is equivalent, it is a match, in any case, it's a confusion. Backtrack – The motivation behind this capacity is to alter the grid that should be printed and encourages us to distinguish the way to be taken to get the most ideal arrangement. printMatrix – It's a circled cycle usage to show the framework. printPredecessorMatrix - It is in this capacity where we print the bolts delineating the way of nearby arrangement. Generate – This capacity produces the two successions An and B which would be privately lined up with one another. An irregular seed is utilized to guarantee the reproducible idea of yield.

**4. Results and Discussion**

**Table 1:** Execution times

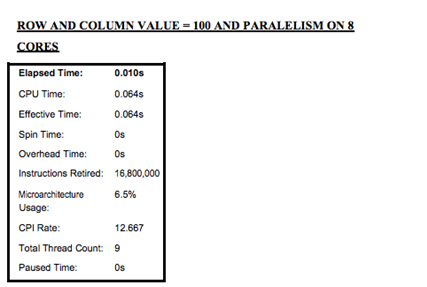
****

****

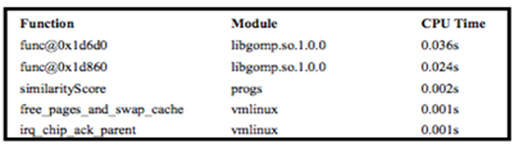
**Figure 2:** Execution Time Analysis

As we can see in the above graph, for small lengths of the sequence, serial and parallel programs tend to give the output in almost the same amount of time. However, as the length increases, the execution time for serial implementation also increases exponentially, hence forming a steep graph. However, the parallel implementation remains stable with not a high rise in the execution time because of the parallel execution of the task with two threads, making the process faster than its serial counterparts.

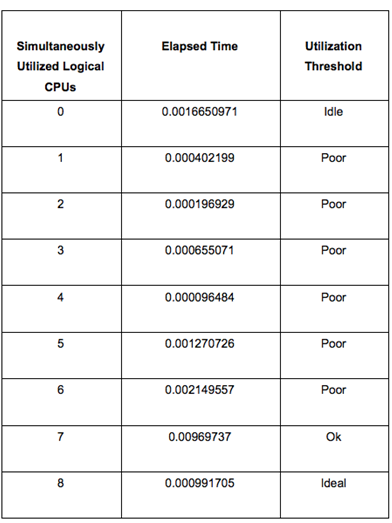
The other test cased will be explained In the following tables.



**Figure 3:** Elapsed time for various CPU processes



**Figure 4:** Top Hotspots for consuming CPU time



**Figure 5:** Utilization threshold for different number of utilized CPU

**5. Conclusion**

As we can see in the above graph, for small lengths of the sequence, serial and parallel programs tend to give the output in almost the same amount of time. However, as the length increases, the execution time for serial implementation also increases exponentially, hence forming a steep graph. However, the parallel implementation remains stable with not a high rise in the execution time because of the parallel execution of the task with two threads, making the process faster than its serial counterparts.

**References**

[1] Deepa. B. C and Nagaveni. V, “Parallel Smith-Waterman Algorithm for Gene Sequencing”, International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169 Volume: 3 Issue: 5, May 2015

[2] Sanghamitra Bandyopadhyay (Senior Member, IEEE) and Ramkrishna Mitra, “A Parallel Pairwise Local Sequence Alignment Algorithm”, IEEE, Volume: 5, June 2014

[3] Ananth Prabhu G and Dr. Ganesh Aithal, “Automatic Parallelization for Parallel Architectures Using Smith Waterman Algorithm”, IEEE, Issue: 3, May 2017

[4] Nagaveni. V, “Parallel Smith-Waterman Algorithm for Gene Sequencing”, International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321- 8169 Volume: 3, May 2015

[5] Chirag Jain and Subodh Kumar (Department of Computer Science and Engineering IIT Delhi), “Fine-grained GPU parallelization of Pairwise Local Sequence Alignment”, ARPN, Volume: 2, Jan 2017

[6] Deepa B.C(Department of CSE, AIT), “Parallel Smith-Waterman Algorithm for Gene Sequencing”, International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169 Volume: 3, May 2015

[7] Umeda, Takayuki, and Shuhei Oya. "Performance comparison of open-source parallel sorting with OpenMP." 2015 Third International Symposium on Computing and Networking (CANDAR). IEEE, 2015.

[8] Alyasseri, Zaid Abdi Alkareem, Kadhim Al-Attar, and Mazin Nasser. "Parallelize Bubble Sort Algorithm Using OpenMP." arXiv preprint arXiv: 1407.6603 (2014).

[9] A. C. Dusseau, D. E. Culler, K. E. Schauser, R. P. Martin, Fast Parallel Sorting under Log P : Experience with the CM-5, IEEE Trans. Parallel Distributed Systems, 7, 791–805 (1996).

[10] S. J. Lee, M. Jeon, D. Kim, A. Sohn, Partitioned parallel radix sort, J. Parallel Distr. Comput., 62, 656–668 (2002).

[11] K. Batcher, Sorting networks and their applications, Proc.AFIPS Spring Joint Computer Conference, 32, pp.307–314 (1968).

[12] J. S. Huang, Y. C. Chow, Parallel Sorting and Data Partitioning by Sampling, Proc. 7th Computer Software and Applications Conference, pp.627–631 (November 1983).

[13] M. Jeon, D. Kim, Parallel merge sort with load balancing, Inter. J. Parallel Prog., 31, 21–33 (2003).

[14] Knessl, Charles, and Wojciech Szpankowski. "Quicksort Algorithm Again Revisited." Discrete Mathematics & Theoretical Computer Science 3.2, 43-64 (1999).

[15] Aumüller, Martin, and Martin Dietzfelbinger. "Optimal Partitioning for Dual-Pivot Quicksort." ACM Transactions on Algorithms (TALG) 12.2, 18 (2015)

[16] Nicolae Goga, Oludele Awodele,”Improved GROMACS Algorithms using the MPI Parallelization”,The 5th IEEE International Conference on E-Health and Bioengineering – EHB, 2015.Desalegn Getachew Melese , Huagang Xiong ,Qiang Gao ,”Consumed Energy as factor for Cluster Head Selection”,IEEE Conference,pp.1-4,2010.