ISSN NO. 2320-5407

Journal homepage: http://www.journalijar.com
Journal DOI: 10.21474/IJAR01

INTERNATIONAL JOURNAL OF ADVANCED RESEARCH

RESEARCH ARTICLE

A Survey of Multiple Sequence Alignment Parallel Tools.

***Mohammed W. Aly[1], Naglaa M. Reda[2], Kasim A. Al-salem[1] and Fayed F. M. Ghaleb[2].**

1. Department of Computer Science, University of Cihan/Sulaimanya, Kurdstan, Iraq.
2. Department of Mathematics, Faculty of Science, Ain Shams University, Cairo, Egypt.

Manuscript Info

Manuscript History:

Received: 16 March 2016
Final Accepted: 19 April 2016
Published Online: May 2016

Key words:

****Corresponding Author***

Mohammed W. Aly.

Abstract

Motivation: Multiple sequence alignment is a key problem to most bioinformatics applications, from evolutionary studies to prediction of protein structure, molecular function, intermolecular interactions, gene finding and phylogenitic analysis. The last ten years have witnessed a big improvement to existing multiple alignment tools and the development of new ones. Varieties of parallel architectures have been experimented such as supercomputer, cluster, grid, cloud, and multi-core machine for the purpose of reaching the highest level of accuracy and speed.

Results: This paper surveys most popular tools to clarify how parallelism accelerates the processing of large biological data set and improve alignment accuracy. It also introduces a comparative study that aims at guiding biologists to choose the appropriate software depending on their requirements and their hardware potentials.

Copy Right, IJAR, 2016,. All rights reserved.

Introduction:-

Bioinformatics [1] is the interface between biological and computational sciences. Its focus is on developing and applying computationally intensive techniques to increase the understanding of biological processes. It offers a plethora of challenging problems that require tremendous computational resources to be solved accurately. One of the most important and basic challenges in bioinformatics is the sequence alignment. It is used to extract functional and evolutionary information of genes and proteins. Thus it has become a fundamental tool in many different domains such as molecular function prediction [2], intermolecular interactions, residue selection [3] and phylogenitic analysis [4].

Sequence alignment is the problem of comparing biological sequences by searching for a series of characters (nucleotides for DNA sequences or amino acids for protein sequences) that appear in the same order in the input sequences, possibly introducing gaps into them. When the number of sequences is two then it is referred to pair-wise sequence alignment, otherwise its multiple sequence alignment (MSA). The most important types of sequence alignment problems are global and local. Global alignment is to find the best match between the entire sequences. While local alignment must find the best match between certain regions of the sequences.

The algorithms used for alignment problem can be either dynamic programming based or heuristic-based or a combination of both. Dynamic programming is a general optimization technique that relies on the fact that the solution to a problem consists of the combined solutions of the sub-problems. Furthermore, several of the sub-problems may be the same. Thus, they are solved only once. Dynamic programming based algorithms generate optimal solutions. However, they are computationally intensive which makes them impractical for a large number of sequence alignments. One the other hand, heuristics are approximation algorithms. They generate a near optimal solution but they are more practical. In the case of sequence alignment, these heuristics often use a combination of a restricted form of dynamic programming and other approximations in order to reduce the search space of possible solutions.

Most MSA methods are based on one of the two most public pair-wise alignment algorithms. The first is an optimal algorithm proposed by Needleman and Wunsh (NW) [5] for global alignment. And the second is an improvement to the NW algorithm proposed by Smith and Waterman (SW) [6] to obtain the local alignment. Both algorithms are exact, based on dynamic programming, and composed of three phases: initialization, similarity matrix computation and trace back. Nevertheless, they differ in their applied techniques at each phase.

The main problem of MSA from the computer science point of view is the huge number of residue-to-residue comparisons that are needed when searching for similarities. Not only the logical definition of the problem makes it time consuming but also the fact that in practice, a single genome may contain in the order of billions of residues. Therefore, researchers were directed to High Performance Computing (HPC) due to the tremendous and steadily growing amount of molecular data [7].

Advances in HPC platforms provide high computational capability and uniform high-speed memory access to multi-terabyte data structures. The technique they use for obtaining high performance is to parallelize the task to be run simultaneously by multiple vector execution units with SIMD and by multiple processors with MIMD. Many different hardware architectures have been experimented by various MSA tools such as cluster [8], multi-core machine [9], grid [10], cloud [11], and supercomputer [12].
This paper introduces a comprehensive coverage of the most efficient and popular parallel MSA tools. The objective is to emphasis each tool's requirements, limitations, capabilities, advantages and disadvantages.

MSA methods:-

High quality MSA is an essential task in bioinformatics. It helps in many criteria such as identifying diagnostic patterns or motif to characterize protein families, demonstrating homology between new sequences and existing families of sequences, and predicting the secondary and tertiary structures of the new sequences [7]. Nevertheless, it is NP-complete problem [13] with the computational cost growing exponentially with the number of sequences. Therefore, many algorithms have been developed aiming to reach the most accurate and efficient alignment. Most commonly used algorithms are classified into two categories, progressive and iterative.

Progressive Method:-

Most widely used MSA tools utilize the progressive method that was first introduced in [14]. For aligning N sequences, it first generates all possible N(N-1)/2 pair-wise sequence alignment in order to calculate a distance matrix giving the divergence of each pair of sequences. Second it creates a guide tree constructed from pair-wise sequence distances using a clustering method such as UPGMA [15] or Neighbor-Joining [16]. Third it builds up the final multiple alignments by progressive inclusion of the N sequences profile alignment according to the order given by the guide tree. While progressive sequence alignment is computationally efficient, it doesn't guarantee a global optimal alignment, and suffers from some shortcomings. The main drawback is that any errors happening at a certain alignment step will propagate and possibly affect the final multiple alignment.

Iterative Method:-

The first iterative algorithm dates back to the origin of MSA in 1987 [17]. It was proposed in order to circumvent the inherent errors in progressive alignment method. It refines the alignment by making an initial alignment of groups of sequences and then revising the alignment to achieve a more reasonable result.
Many iterative methods have been developed to refine alignment. They can be deterministic or stochastic, depending on the strategy used to improve the alignment. Deterministic method involves extracting the sequence one by one from multiple alignments and realigning them to the remaining sequences. This procedure is terminated when no improvement can be made. On the other hand, stochastic iterative methods include Hidden Markov Models (HMMs) [18], simulated annealing [19] and evolutionary computation such as genetic algorithms (GAs) [20] and evolution strategies [21]. Their main advantage is to allow for a good separation between the optimization process and evaluation criteria [22].

MSA parallel tools evolution:-

Various parallel tools have been developed for MSA problem. They concentrate on parallelizing the most time consuming task; the computation of pair-wise sequence distance. Parallelism is achieved by considering the two different granularity alternatives: fine–grained and coarse-grained. The performance of each tool depends on the

used alignment method, the proposed strategy to apply parallel computing, and the chosen platform for implementation. The most important MSA tools are summarized below with a brief overview of each tool series.

FASTA series:-

FASTA series [23, 24] was the first widely used program for database similarity searching. FASTP was the most extensively used program for protein. FASTA exploits refinements that result in a significant improvement in sensitivity. It takes a given nucleotide (DNA/RNA) or amino-acid sequence and searches a corresponding sequence database by using local sequence alignment to find matches of similar database sequences. It aligns multiple sequences and thereafter computes pair of match and mismatch between the sequences. It uses a heuristic algorithm which is an efficient implementation of an AltiVec-enabled SW. Its source code features optimizations targeted at modern processors to be faster without losing much sensitivity.

A variety of research work has been done to study the performance of parallel FASTA. On Sun servers [25], the parallel efficiency of FASTA programs was quite high, especially for large searches. It shows a 54-fold speedup when it runs on 62 CPUs of the Sun Enterprise 10000 (64400 MHz CPUs with 8MB L2 cache).

On a parallel cluster of workstations [26], it executes very quickly when a small query or database is chosen. The parallel FASTA ported using Sun-MPI libraries was run on Fast Ethernet across 64 processors and a speedup of 44 fold was observed. Better speedup was observed when searching a longer query sequence against a huge database.

On a grid [27], FASTA has been implemented in the Grid Application Development Software (GrADS) project. The GrADS adapts the master-worker paradigm, scheduling and rescheduling the tasks on an appropriate set of resources, launching and monitoring the execution. Its scheduler makes a static schedule for its application where the whole or a portion of sequence databases are replicated on some or all of the grid nodes. The master informs each worker which portions of database should be loaded into memory, sends the input query sequence and collects the results. This paradigm provides a plausible method for providing large amounts of computing power applicable with the rapid growth of large biology data collections. FASTA have been implemented by many other bioinformatics grid projects [10].

On Cell B.E. [28], the AltiVec APIs are converted to the synergistic processing unit (SPU) APIs, and those that are not available on the SPU have been implemented. Results show that the Cell B.E. is a promising power-efficient platform considering that the total power consumption of the Cell is less than half of a contemporary superscalar processor. Also superior results have been achieved in [8] mainly due to the vector execution on the eight SPU cores. And profiling indicates that the computation dominates the total runtime (up to 99.9% considering a bandwidth of 18 Gbytes/s for the SPU).

On a hybrid cloud environment [11] which consisted of services provided by the public Amazon cloud (S3 and SQS), with computing nodes residing on the Intel cluster running Eucalyptus software, the SW similarity search program from the FASTA package were implemented. A component-based approach to computational science applications on cloud infrastructures was used. It considers virtual machines running on the cloud as components and the cloud as a large-scale distributed container for hosting these components. The experiments demonstrated that this approach can be effective and overhead incurred by the component startup time as well as the performance penalty caused by virtualization are reasonable.

BLAST series:-

BLAST [29, 30] is most noted heuristic-based tool for protein and DNA sequence homology search because of its impressive speed. It works on the principle of hashing small matching sequences and then extending the hash matches to create high scoring segment pairs until attend the highest score possible (HSP). Several variations of the original BLAST algorithm were developed to accommodate different types of sequence alignments [31]. The original BLAST uses the two-hit alignment method to increases speed. Gapped BLAST adds the ability to generate gapped alignments. PHI-BLAST combines pattern search with the search for statistically significant sequence similarity to detect subtle similarities. PSI-BLAST executes the BLAST algorithm iteratively and updates the used position-specific scoring matrix for detecting weak sequence similarities. MEGABLAST employs the XDrop alignment greedy algorithm to particularly align DNA sequences that are highly similar. NCBI-BLAST performs an accurate gapped alignment using two-hit alignment method with far fewer HSPs statistics that leads to much increased sensitivity and speed. GPU_BLAST [32] is the most recent accelerated version of the popular NCBI-

BLAST using a general-purpose graphics processing unit (GPU). The speedups achieved range mostly between 3 and 4.

There have been a number of efforts to parallelize BLAST for achieving higher performance. TurboBLAST [33] is an accelerated, parallel deployment of NCBI BLAST which delivers high performance on networked clusters of heterogeneous PCs, workstations, or Macintosh computers. A speedup was 10.8 obtained by using a distributed Java "harness" that splits BLAST jobs into multiple small pieces, processes the pieces in parallel, and integrates the results into a unified output.

Soap-HT-BLAST [34] is a high throughput web-based system that runs NCBI BLAST/PSI-BLAST in a distributed, parallel environment through the Internet. Its hardware architecture includes one head node and three compute nodes; each has four Ultraspac III 900MHz CPUs and 8GB memory. It is implemented using Perl and its module SOAP::Lite for Web services. At one time, the CPU loads of all working compute nodes are limited to 85% and the maximum number of submissions is six.

PARACEL-BLAST [35] is the most advanced BLAST software written specifically for large-scale cluster systems. It is an integrated native-parallel application of NCBI BLAST. It develops a costumer scheduler and makes database splitting dynamically which improve performance, efficiency, and scalability. It provides optimizations, enhancements, and features not available in any other BLAST system.

mpiBLAST [36] uses MPI to parallelize BLAST by querying segments of the target database simultaneously on a computing cluster. It preprocesses the target database by dividing it into unique segments for use by individual nodes in the cluster. The segment is copied to the cluster node for reducing communication with shared storage during the computation. mpiBLAST achieved super-linear speedup on up to 128 nodes of a 240 node Linux Beowulf cluster. But efficiency decreased as the number of nodes increased.

pioBLAST [37] was produced as an improvement to mpiBLAST through an optimization of the I/O and communication. It features online dynamic partitioning of the database, parallel I/O through the use of an MPI-IO, and efficient merging of results. It gains additional performance during sorting of results by overlapping the writing of sets of results as computational nodes report back to the master node. It has been tested on the IBM Blade Cluster using up to 32 nodes offering a total of 128 Intel Xeon 2.8-3.0 GHz Processors; each is equipped with 4 GB memory and 40 GB disk place. Results show time decrease to 64% which is considerably milder than that of mpiBLAST (14%).

ScalaBLAST [38] is a parallel implementation of the original NCBI BLAST for high-throughput calculations in a cluster or supercomputer. It accommodates very large databases and scales linearly to as many as thousands of processors on both distributed memory and shared memory architectures. It relies on distributing the target database over available memory, multilevel parallelism to exploit concurrency, parallel I/O, latency hiding through data prefetching combined with effective task scheduling to achieve high-performance and scalability. It has been used to perform multiple genome BLAST calculations against a variety of databases on commodity clusters or high-performance architectures. When using 1500 processors, it was up to 1500 times faster on real datasets compared to BLAST and makes distributed memory perform like true shared memory.

G-BLAST [39] was developed during designing and implementing a BioGrid framework. It performs genomic sequence alignments using Grid computing environments and accessible mpiBLAST applications. G-BLAST is also suitable for cluster computing environments with a server node and several client nodes. It is able to select the most appropriate work nodes, dynamically fragment genomic databases, and self-adjust according to performance data. It's more efficient than mpiBLAST due to its GUI friendly interface, flexibility and speedup. A number of other grid/cloud-enabled applications have been developed for BLAST [10, 40].

Clustal series:-

The Clustal series [41] are the most widely used programs for global multiple sequence alignment. The first Clustal program [42] combined the progressive alignment strategy with dynamic programming using a guided tree. Then ClustalV featured the ability to produce phylogenetic trees from alignments, using the neighbor-joining method and bootstrap confidence measures. The third generation, ClustalW incorporated a number of improvements to the alignment algorithm, including sequence weighting, position-specific gap penalties and the automatic choice of a

suitable residue comparison matrix at each stage in the multiple alignment. It was also developed to ClustalX with a more user-friendly graphical interface.

MULTICLUSTAL [43] was introduced as a multithreaded optimization version of the ClustalW. Its alignment gives a domain structure, which is more consistent with the 3D structures of proteins. It searches for the best combination of ClustalW input parameters. Its performance gives speedups range from 1.5 to 3.0 when compared with the original one.

SGI parallel Clustal [44] was the first attempt to accelerate ClustalW by parallelizing all three stages on a shared memory SGI Origin machine using OpenMP. It shows speedup of up to 10 folds when running ClustalW on 16 CPUs. So as HT ClustalW the other parallel programming approach that adopts high-throughput (HT) automation. It obtains similar time on smaller number of, therefore freeing additional computer resources without performance degradation.

ClustalW-MPI [45] uses MPI with dynamic scheduling proposes and runs on distributed workstation clusters. The calculations of pairwise distances scale up to 15.8 using 16 processors. And speedup of 4.3 can be achieved with its mixed fine and coarse grained approach using 16 processors for the essentially not parallelizable progressive alignment.

pCLUSTAL [46] is another parallel version of ClustalW using MPI. In contrast to the commercial SGI parallel Clustal version, which requires an expensive SGI multiprocessor system, pCLUSTAL can be run on a range of distributed and shared memory parallel machines, from high-end parallel multiprocessors to PC clusters, to simple networks of workstations and achieve relative speedup. Another acceleration of ClustalW have been developed in [47] on multiprocessor SMP cluster using a hybrid MPI/OpenMP method with mixed fine and coarse grained parallelization approach in the third stage. A speedup of 80 and 9.2 was obtained for the first and third stages respectively, and an overall speedup of 35 using 40 nodes and 80 processors.

MT-ClustalW [48] presented a fully multithreading optimized version of ClustalW which utilize the machine resources and achieve higher throughput on multicore computers. It achieves over 2 times faster than the sequential ClustalW with 8 threads.

GPU-ClustalW [49] was the first to publish MSA acceleration on GPUs. A single GPU card (GeForce 7800 GTX) was programmed with OpenGL Shading Language (GLSL). It achieved a stage 1 speedup of 11.7 compared with a 3.0 GHz Pentium 4 processor. Stages 2 and 3 were executed sequentially for an overall speedup of 7.2. Then MSA-CUDA [50] parallelizes all three stages of the ClustalW processing pipeline by the GPU using CUDA. It demonstrates average speedups of 36.91 for long protein sequences on a GeForce GTX 280 GPU compared to the sequential ClustalW running on a Pentium 4 3.0 GHz processor.

While on Cell BE [51], acceleration has been achieved in an overall speedup by 9.1. It speeds up the pairwise alignment phase of ClustalW with a factor of 51.2 by making extensive use of vectorization and by scheduling the application across all cores. Also the progressive alignment phase is sped up by a factor of 5.7 when applying loop unrolling and loop skewing optimizations. On the other hand, [52] achieved a peak speedup of 9.03 for 1000 protein sequences with an average length of 446. It accelerates the distance matrix computation by using Playstation3 powered by Cell BE.

Then [53] demonstrate a speedup of 24.4 times when using 16 synergistic processor units on a QS21 Cell Blade compared to single-thread execution on the power processing unit. Its highly optimized implementation is just 3.8 times faster than a 3-thread version running on an Intel Core2 Duo.

ClustalXeed [54] has incremental improvements over previous versions. It can compute a large volume of biological sequence data sets, which were not tractable before. It uses both physical RAM and a distributed file-allocation system for distance matrix construction and pair-align computation to solve the conventional memory-dependency problem. It markedly improves the computation efficiency of disk-storage system by implementing INSTA load-balancing algorithm. Performance tests on data set comprised of 52,750 protein sequences from the cytochrome P450 superfamily show that the average speed-up for 50 nodes was about 19.6 with INSTA and 14.8 without INSTA on 100×CPU AMD Opteron 244 (1.8 GHz) 64-bit cluster system with Linux (Fedora 4 Core). The master

node consists of a 10-terabyte HDD and 8 GB DRAM. Each node is a dual-core Opteron system with 4 GB DRAM and a 1-terabyte HDD.

Clustal Omega [55] is the latest version. It can align virtually any number of protein sequences quickly and that delivers accurate alignments. It uses the OpenMP library to enable multithreaded computation of pairwise distances and alignment match states. The accuracy of the package on smaller test cases is similar to that of the high-quality aligners. On larger data sets, Clustal Omega outperforms other packages in terms of execution time and quality.

In [56] a new multithreaded algorithm was proposed to eliminate the synchronization delays bottlenecks by using threads and different scheduling approach. It was incorporated in ClustalW-MPI and enables a better use of CPU cycles and also a better memory usage. Tests involved cases with 500 sequences with sizes of 1000, 2000, 3000, 4000 and 5000 residues, and shows speedup improvement.

In [57] a new parallel implementation of distance matrix computations of ClustalW is based on MPI and OpenMP mechanism. It achieves speedup of about 9 on 50 sequences of average length of 161.000 nucleotide, with 32 nodes each nodes content two Intel Quad core Xeon 2.83 GHz of processor, 64 bit technology.

A parallel optimized algorithm of second stage of ClutalW by [58]. This paper introduces a new algorithm based on converting used matrices in NJ into vectors and eliminating redundant computations, while preserving the accuracy. The results show reductions of the time and space complexities while the accuracy is preserved. It implemented on a 2.0 GHz core i7 Intel CPU in C++ and tested on various real DNA and RNA sequences. Results show how the proposed vectorization approach greatly improves the performance and achieves more than 2.5-fold speedup when aligning 8000 sequences compared to ClustalW- MPI.

T-COFEE series:-

T-Coffee [59] (Tree-based Consistency Objective Function For alignment Evaluation) was the first MSA software that uses a consistency-based objective function optimized using progressive alignment. It tries to maximize the score between the final multiple alignment and a library of pair-wise residue-by-residue scores derived from a mixture of local and global pair-wise alignments. Then 3DCoffee [60] used a mixture of pair-wise sequence alignments and pair-wise structure comparison methods to generate multiple sequence alignments. It combines TCoffee with the threading program Fugue that improves the accuracy by four percentage points when using one structure only per dataset. Nevertheless M-Coffee (meta-method for assembling MSA) [61] extends T-Coffee by using consistency to estimate a consensus alignment. It was twice as likely to deliver the best alignment as any individual method. Also R-Coffee [62] was designed to align RNA sequences while exploiting secondary structure information. It used an alignment-scoring scheme that incorporates secondary structure information within the alignment.

Parallel T-Coffee (PTC) [63] was the first parallel implementation of T-Coffee. It is based on MPI and RMA mechanisms. It realized a speedup of about 3 with 80 processors on Cluster consisting of dual Intel Xeon 3GHz nodes. Most of the speedup comes from parallelizing and distributing pair-wise alignment tasks dynamic scheduling for a near linear speedup during library generation. Since most of the potential users of this tool are not familiar with the use of grids or supercomputers, a web portal was created with Rapid in [64] that deploy PTC on different HPC environments. It allows users to upload a large number of sequences and provides a user-friendly solution.

Cloud-Coffee [65] is another parallel implementation of T-Coffee. Its approach is different; it is based on shared-memory architectures, like multi-core or multi-processors. It was benchmarked on the Amazon Elastic Cloud (EC2) and showed that the parallelization procedure is reasonably effective. As for a web server with moderate usage (10K hits/month) the cloud provides a cost-effective alternative to in-house deployment. For instance, the medium instance (5 EC2 Computation Units, ECU) runs 3.7 times faster than its small counterpart (1 ECU).

MAFFT series:-

MAFFT [66] is another popular MSA program. It included two novel techniques that reduce the CPU time. It rapidly identified homologous regions by the fast Fourier transform FFT. It was modified to improve the accuracy at [67]. Then it was updated in [68] with two new techniques. The PartTree algorithm improves the scalability of progressive alignment and the Four-way consistency objective function improves the accuracy of ncRNA alignment.

All stages of MAFEET have been parallelized in [69] using the POSIX Threads library with the best-first and simple hill-climbing parallelization strategies. This approach achieved a peak speedup of 10 times with different random numbers on a 16 core PC (4×Quad-Core AMD Opteron Processor 8378).

Muscle series:-
MUSCLE [70] (Multiple Sequence Comparison by Log-Expectation) is a widely used tool in MSA. It has achieved a highest rank in accuracy and the fastest speed compared to others. Because it includes fast distance estimation using kmer counting, progressive alignment uses a new profile function, and refinement using tree-dependent restricted partitioning.

MUSCLE-SMP [71] was the first parallel attempt of MUSCLE on shared memory system. It achieves an overall speedup of 15.2 on a 16 processors SMP system using OperMP for 50-150 proteins of average length 330.

MUSCLE-based multiscale simulations [72] have been presented in the two types of infrastructures: local HPC cluster and Amazon AWS cloud solutions. And presented solution has been integrated with Grid Space virtual laboratory that enables users to develop and execute virtual experiments on the underlying computational and storage resources through its website based interface.

Also the multithreaded algorithm in [56] was incorporated in Muscle-SMP and achieved superior results. The absolute time needed for each execution is lower than the time needed by the original version, sometimes using less than half.

DIALIGN series:-
DIALIGN was proposed in [73]. Its algorithm compares every pair of sequences, generating a set of ungapped fragments with high score. These fragments are used to, incrementally and iteratively, generate the final alignment. DIALIGN 2.0 [74] introduced an optimization that was able to generate less fragments of longer sizes. DIALIGN-TX [75] used greedy and progressive approaches for segment-based MSA. It incorporated optimizations such as anchors and a guide tree to generate more accurate alignments.

DIALIGN-P [76] the parallel version of DIALIGN 2.0 was proposed for homogenous clusters. It achieved speedup of 19.32, 10.87 and 4.15 on (20, 55 and 100 seqs.) respectively by using 64 processors.

DIALIGN-TX-MPI [77] is the parallel version of DIALIGN-TX. It used an iterative heuristic method for MSA that is based on DP and generates alignments by concatenating ungapped regions with high similarity. It was implemented using both OpenMP and MPI on a heterogeneous multi-core cluster composed by 3 nodes with 4, 8 and 16 cores, respectively. The best speedup (3.13) was obtained when comparing set of (324) seq. with an average length of (1029) with HWF allocation policy.

Others:-
Sample-Align-D [78] another parallel algorithm was proposed. It was based on partitioning the set of sequences into smaller subset using k-mer count based similarity index (k-mer rank). Then each subset is independently aligned in parallel. The algorithm has been implemented on a cluster of workstation on 16 node using MPI library. It was able also to align 2000 randomly selected sequences from the Methanosarcina acetivorans genome in less than 10 minutes, compared to over 23 hours on sequential MUSCLE [70] system running on a single cluster node. Also it was able to align 20000 sequences in just around 25 seconds.

ProbCons [79] is a practical tool for progressive protein MSA based on probabilistic consistency. It introduces a pair-HMM progressive alignment algorithm that uses maximum expected accuracy rather than Viterbi highest probability alignment, and the probabilistic consistency transformation to incorporate multiple sequence conservation information during pair-wise alignment. It achieves statistically significant improvement over other methods, containing an average of 7% more correctly aligned columns than those of T-Coffee, 11% more than CLUSTAL W, and 14% more than DIALIGN.

Probalign [80] adopts a very similar strategy to ProbCons, but employs a partition function to calculate posterior probabilities instead of using a pair-HMM. Results indicate that Probalign alignments are generally more accurate than Probcons, MAFFT and MUSCLE.

MSAProbs [81] is a new and practical multiple protein sequence alignment algorithm designed by combining a pair-HMM and a partition function to calculate posterior probabilities. It also investigates two critical bioinformatics techniques, namely weighted probabilistic consistency transformation and weighted profile-profile alignment, to achieve high alignment accuracy. In addition, it is optimized for modern multi-core CPUs by employing a multi-threaded design in order to reduce execution time. It statistically demonstrates dramatic accuracy improvements over several top performing aligners: ClustalW 2.0.12, MAFFT 6.717, MUSCLE 3.8.31, ProbCons 1.12, and Probalign 1.3

ParaAT [82] is a recent parallel tool that is capable of constructing multiple protein-coding DNA alignments for a large number of homologs. It is well suited for large-scale data analysis in the high-throughput era. It assigns each homolog to one of the slave threads, customizes by user one of multiple sequence aligners (including ClustalW, Mafft, Muscle, T-Coffee), consolidates the results from all slave threads, then parallel back-translates multiple protein sequence alignments into the corresponding DNA alignments. Tests were performed on a 64 bit x86 Intel_ Xeon_ machine with 24 cores (X5650) and provides good scalability and exhibits high parallel efficiency.

BLASR, in [83] advocate high-level programming methodology for next generation sequencers (NGS) alignment tools for both productivity and absolute performance. The paper analyse the problem of parallel alignment and review the parallelisation strategies of the most popular alignment tools, which can all be abstracted to a single parallel paradigm. By using a SIMD/SSE architecture to obtain an overall speedup of about 12 on a system with an Intel Sandy Bridge has two 8-core sockets (2 HyperThreads) @2.2 GHz, 20 MB L3 cache with Linux x86_64.

Discussion:-

Despite the frequent use of the MSA tools by biologists and scientists, the decision about which tool to use is a difficult problem. Lately some researches [84, 85, 86] have been produced for surveying, comparing and evaluating sequential tools to address this critical issue and highlight a number of specific strengths and weaknesses of them.

But with parallel tools the situation is much difficult. There are a lot of them as surveyed above. The assessment and the choice of the most convenient tool are subjected to variant metrics. Most important metrics that affect the usability and popularity of the aligner were selected and summarized below. Also a comprehensive comparison of the available most popular and efficient parallel MSA software tools against these metrics emphasizing their advantages and disadvantages is presented in table 1.

- *Availability*: The ease to obtain and use the software. The program needs to be publicly available and user-friendly, so that anyone can apply it and compare its results with others.
- *Portability*: The ability to run the program with different operating systems (OS). This is very significance as most scientists intend to run it on their PCs rather than web interfaces.
- *Hardware architecture*: The parallel platform needed to operate the tool such as supercomputers, clusters, grids, clouds, and multi-cores. Of Course its cost, availability and accessibility must be considered.
- *Database size*: the allowed limit of the sequences number and length. Since the maximum number of sequences (MaxNo) and the sequence maximum length (MaxL) are deeply important to biologists. It is limited by the storage available by the used platform.
- *Speedup:* The acceleration gained by using the parallel tool compared to the sequential one. It is an essential metric in measuring the performance. It clarifies the tool's high computational capability and high-speed memory access.

Conclusion:-

The aim of this paper is to direct both biologists and scientists to choosing the most appropriate MSA tool for their specific needs, thus enabling more efficient research. The general and central considerations of the MSA methods have been reviewed, to clarify their main differences. The most popular and widely used tools have been surveyed with a study of their evolution from the first appearance till now. A detailed discussion of existing parallel tools has been introduced supported by a comparative study according to variant metrics, emphasizing their main characteristics.

References:-

1. Albert, Y., Zomaya, (2006). Parallel computing for bioinformatics and computational biology: models, enabling technologies, and case studies. *John Wiley & Sons, Inc., Hoboken, New Jersey*.
2. Pierri, C.L., Parisi G., Porcelli, V. (2010). Computational approaches for protein function prediction: a combined strategy from multiple sequence alignment to molecular docking-based virtual screening. *Biochim Biophys Acta,* 1804(9):1695-712.
3. Alfonso Benítez-Páez, Sonia Cárdenas-Brito, and Andrés J. Gutiérrez (2011). A practical guide for the computational selection of residues to be experimentally characterized in protein families. *Brief Bioinform first published online*.
4. Liu, K., Linder, C. R. and Warnow, T. (2010). Multiple sequence alignment: a major challenge to large-scale phylogenetics. *PLoS Currents: Tree of Life*.
5. Needleman, S.B. and Wunsch, C.D. (1970). A general method applicable to the search for similarity in the amino acid sequence of two proteins. *J. Mol. Biol.*, 48:443–453.
6. Smith, T. F. and Waterman, M. S. (1981). Identification of common molecular subsequences. *Journal of Molecular Biology* 147: 195–197.
7. Bertil Schmidt (2011). High Performance Parallel Computer Architectures. *CRC Press, 2011.*
8. Bina K. Pandey, Sanja K. Pandey and Divijay Pandey (2011). A Survey of Bioinformatics Applications on Parallel Architectures. *International Journal of Computer Applications*, **23**, 0975 – 8887 .
9. Sebastian Isaza Ram´ırez (2011) Multicore Architectures for Bioinformatics Applications. Ph.D. Thesis, *TU Delft University*, Delft, Switzerland.
10. Manjula K. A. and Raju G. (2010). A Study on Applications of Grid Computing in Bioinformatics (2010). *IJCA Special Issue on CASCT*.
11. Maciej Malawski, Jan Meizner, Marian Bubak, et al., (2011). Component Approach to Computational Applications on Clouds. *Procedia CS* 4: 432-441.
12. Plamenka B, Gancheva, V. , Markov, S. et al., (2011). Parallel performance evaluation and profiling of multiple sequence nucleotide alignment on the supercomputer. *BlueGene/P*, (IDAACS), 2011 IEEE 6th International Conference.
13. Wang, L. and Jiang T. (1994). On the complexity of multiple sequence alignment. *Journal of computational biology*, 1:337–348.
14. Feng, D. and Doolittle, R. (1987). Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *J Mol Evol*, 25:351-360.
15. Sneath, P.H.A. and Sokal, R.R. (1973). Numerical Taxonomy. *(Freeman, San Francisco, CA)*, 230-234.
16. Saitou, N. and Nei, M. (1987). The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Mol Biol Evol*, 4:406-425.
17. Barton, G.J. and Sternberg, M.J.E. (1987). A strategy for the rapid multiple alignment of protein sequences: confidence levels from tertiary structure comparisons. *J. Mol. Biol.* 198:327-337.
18. Krogh, Brown, M., Mian, I., Sjlander, K. et al., (1994). Hidden Markov Models in Computational Biology: Applications to Protein Modeling. *Journal of Molecular Biology, 235: 1501–1531*.
19. Kirkpatrick, S., Gelatt, C. D. and Vecchi, M. P. (1983). Optimization by simulated annealing, *Science*, 220 (4598): 671–680.
20. Holland, J. (1973). Genetic algorithms and the optimal allocations of trials, *SIAM J. Comput.*, 2: 88–105.
21. Rechenberg, I. (1973). Bioinik, evolution und optimierung, *Naturwissenschaftiche Rundschau*, 26: 465–472.
22. Choudry, R. (1999). Application of Evolutionary Algorithms for Multiple Sequence Alignment. *Stanford University*.
23. Pearson, W.R. and Lipman, D.J. (1988). Improved tools for biological sequence comparison. *Proceedings of the National Academy of Sciences of the United States of America*, 85:2444-2448.
24. Pearson, W. R. (1990). "Rapid and sensitive sequence comparison with FASTP and FASTA", *Methods Enzymol.* 183: 63-98.
25. Ilya Sharapov (2001). "Computational Applications for Life Sciences on Sun Platforms: Performance Overview", *Whitepaper, 2001*.
26. Chintalapati Janaki and Rajendra R. Joshi (2003). "Accelerating comparative genomics using parallel computing", *Silicon Biology 3, 0036, Bioinformation Systems, 2003*.
27. YarKhan, A. and Dongarra, J. J. (2005). "Biological Sequence Alignment on the Computational Grid Using the GrADS Framework", *Future Generation Computer Systems*, 21:980 – 986.
28. Vipin Sachdeva, Michael Kistler, Evan Speight, et al., (2008). Exploring the viability of the cell broadband engine for bioinformatics applications. *Parallel Computing*, 34(11):616–626.

29. Altschul, S. F., Gish, W. Miller, W. et al., (1990). Basic local alignment search tool. *J Mol Biol* 215:403–410.
30. Lenwood, S. (2011). Heath and Naren Ramakrishnan Problem Solving Handbook in Computational Biology and Bioinformatics. *(Book) © Springer Science + Business Media, 2011*.
31. Altschul, S. F., Thomas L. Madden, Alejandro A. Schäffer, et al., (1997). "Gapped BLAST and PSI-BLAST: a new generation of protein database search programs", *Nucleic Acids Res.*, 25(17): 3389-3402.
32. Panagiotis D. Vouzis and Nikolaos V. Sahinidis (2011). GPU-BLAST: using graphics processors to accelerate protein sequence alignment. *Bioinformatics*: 182-188.
33. Bjornson, R.D., Sherman, A.H., Weston, S.B. et. al. (2002). "TurboBLAST: A Parallel Implementation of BLAST Built on the TurboHub", International Parallel and Distributed Processing Symposium: *IPDPS Workshops*, p.0183.
34. Jiren Wang and Qing Mu (2003). Soap-HT-BLAST : high throughput BLAST based on Web services, *BIOINFORMATICS APPLICATIONS NOTE*, 19(14):1863-1864.
35. Marc A. Rieffel, Tristan G. Gill, and William R. White (2004). "Bioinformatics Clusters in Action", *Paracel Inc*.
36. Darling, A. E., Carey, L. and Feng, W. (2003). The design, implementation, and evaluation of mpiBLAST, 4th Int. Conf. on Linux Clusters: *The HPC Revolution 2003 in conjunction with Cluster World Conference & Expo, June 2003*.
37. Lin, H., Ma, X., Chandramohan, P. et al., (2005). Efficient data access for parallel BLAST, presented at the 19th IEEE Int. *Parallel Distrib. Process. Symp., Apr. 2005.*
38. Oehmen, C. and Nieplocha, J. (2006). ScalaBLAST: A scalable implementation of BLAST for high-performance data-intensive bioinformatics analysis, *IEEE Transactions on Parallel & Distributed Systems*, 17 (8): 740-749.
39. Yang, C. T., Han, T. F. and Kan, H. C. (2009). "G-BLAST: a Grid-Based Solution for mpiBLAST on Computational Grids", *Concurrency and Computation: Practice and Experience*, Vol. 21, No. 2, pp. 225-255.
40. Andréa Matsunaga, Maurício Tsugawa and José Fortes (2008). CloudBLAST: Combining MapReduce and Virtualization on Distributed Resources for Bioinformatics Applications, *Fourth IEEE International Conference on eScience*:222-229.
41. Thompson J. D. (2005). The Clustal Series of Programs for Multiple Sequence Alignment, The Proteomics Protocols Handbook Edited by: *J. M. Walker © Humana Press Inc., Totowa, NJ, 2005.*
42. Thompson, J.D., Higgins, D.G. and Gib-Son, T.J. (1994). "CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, positions-specific gap penalties and weight matrix choice," *Nucleic Acids Research*, 22:4673–4680.
43. Yuan, J., Amend, A., Borkowski, J., et al., (1999). MULTICLUSTAL: a systematic method for surveying Clustal W alignment parameters, *Bioinformatics*, 15 (10), 862.
44. Dmitri Mikhailov, Haruna Cofer, and Roberto Gomperts (2001). Performance optimization of Clustal W: Parallel Clustal W, HT Clustal, and Multiclustal. *SGI ChemBio, Silicon Graphics, Inc.*
45. Kuo-Bin Li (2003). ClustalW-MPI: ClustalW analysis using distributed and parallel computing. *BIOINFORMATICS APPLICATIONS NOTE.*
46. Cheetham, J., Dehne, F., Pitre, S., et al., (2003). "Parallel Clustal W for PC Clusters," *Proceedings of International Conference on Computational Science and Its Applications (ICCSA)*, vol. 2668, pp. 300–309.
47. Guangming Tan, Shengzhong Feng, and Ninghui Sun (2005). Parallel multiple sequences alignment in SMP cluster. *In Proceedings of the Eighth International Conference on High-Performance Computing in Asia-Pacic Region (HPCASIA'05), July 2005.*
48. Kridsadakorn Chaichoompu and Surin Kittitornkun (2006). Multithreaded ClustalW with improved optimization for Intel multi-core processor. *In International Symposium on Communications and Information Technologies, ISCIT* '06, pages 590-594, 18-20.
49. Weiguo Liu, Bertil Schmidt, Gerrit Voss, et al., (2006). GPU-ClustalW: Using graphics hardware to accelerate multiple sequence alignment. *In High Performance Computing, HiPC 2006*, volume LNCS 4297, pages 363-374. Springer Berlin /Heidelberg.
50. Yongchao Liu, Bertil Schmidt, and Douglas L. Maskell (2009). MSA-CUDA: Multiple sequence alignment on graphics processing units with CUDA. *In 20th IEEE International Conference on Application-specic Systems, Architectures and Processors (ASAP'09),* pages 121-128. IEEE Computer Society.
51. Hans Vandierendonck, Sean Rul, Michiel Questier et al., (2008). Experiences with parallelizing a bio-informatics program on the cell BE. *In High Performance Embedded Architectures and Compilers (HiPEAC'08),* volume 4917/2008, pages 161–175. Springer, Berlin.

52. Adrianto Wirawan, Bertil Schmidt, and Chee Keong Kwoh (2009). Pairwise distance matrix computation for multiple sequence alignment on the Cell Broadband Engine. *In Computational Science - ICCS 2009*, volume LNCS 5544, pages 954-963. Springer-Verlag Berlin Heidelberg.
53. Hans Vandierendonck, Sean Rul, and Koen De Bosschere (2010). Accelerating multiple sequence alignment with the Cell BE processor. *The Computer Journal*, 53(6):814–826.
54. Taeho Kim and Hyun Joo, (2010). ClustalXeed: a GUI-based grid computation version for high performance and terabyte size multiple sequence alignment. *BMC Bioinformatics, 11:467.*
55. Fabian Sievers, Andreas Wilm, David Dineen, et al., (2011). Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega.
56. Geraldo F.D. Zafalon, (2012). Using Threads to Overcome Synchronization Delays in Parallel Multiple Progressive Alignment Algorithms, *American Journal of Bioinformatics* 1 (1): 50-63.
57. Mohammed W. Al-Neama, NaglaaM. Reda, and Fayed F.M. Ghaleb, (2014). An Improved Distance Matrix Computation Algorithm for Multicore Clusters. BioMed Research International. Article ID 406178, http://dx.doi.org/10.1155/2014/406178.
58. Mohammed W. Al-Neama, NaglaaM. Reda, and Fayed F.M. Ghaleb, (2014). Accelerated Guide Trees Construction for Multiple Sequence Alignment. International Journal of Advanced Research. Volume 2, Issue 4, 14-22.
59. Evandro A. Marucci Notredame C, Higgins DG, et al., (2000). T-Coffee a novel method for fast and accurate multiple sequence alignment. J Mol Biol; 302:205–17.
60. O'Sullivan, O., Suhre, K., Abergel, C. et al., (2004). 3DCoffee: combining protein sequences and structures within multiple sequence alignments. *J. Mol. Biol.*, 340,385–395.
61. Iain M. Wallace, Orla O'Sullivan, Desmond G. Higgins et al., (2006). M-Coffee: combining multiple sequence alignment methods with T-Coffee. *Nucleic Acids Research*, Vol. 34, No. 6, 2006.
62. Wilm, A., Higgins, D.G. and Notredame, C. (2008). R-Coffee: a method for multiple alignment of non-coding RNA. *Nucleic Acids Res*., 36, e52.
63. Jaroslaw Zola, Xiao Yang, Adrian Rospondek, et al., (2007). Parallel T-Coffee: A parallel multiple sequence aligner. *In Proceedings of the ISCA 20th International Conference on Parallel and Distributed Computing Systems*, pages 248253, September 24-26 2007.
64. Josep Rius, Fernando Cores, Francesc Solsona, et al., (2011). A user-friendly web portal for T-Coffee on supercomputers. *BMC Bioinformatics, Vol. 12, No. 1, 150.*
65. Di Tommaso P., Orobitg M., Guirado F., et al., (2010). Cloud-Coffee: implementation of a parallel consistency-based multiple alignment algorithm in the T-Coffee package and its benchmarking on the Amazon Elastic-Cloud. *Bioinformatics*, 26(15):1903-1904.
66. Katho, K., Misawa, K., Kuma, K.-i., et al., (2002). MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform. *Nucl. Acids Res*., 30(14), 3059-3066.
67. Katoh,K. and Toh,H. (2005) MAFFT version 5: improvement in accuracy of multiple sequence alignment. Nucleic Acids Res, 33, 511-518.
68. Katoh, Toh. (2007). PartTree: an algorithm to build an approximate tree from a large number of unaligned sequences. *Bioinformatics 23:372-374.*
69. Katoh, K. and Toh, H. (2010). Parallelization of the MAFFT multiple sequence alignment program. *Bioinformatics*, Vol. 26 no. 15 2010, Pp 1899-1900.
70. Edgar, R. C. (2004). "MUSCLE: multiple sequence alignment with high accuracy and high throughput", *Nucleic Acids Res*, vol. 32, pp. 1792-1797.
71. Deng, X., Li, E., Shan, J. et al., (2006). Parallel implementation and performance characterization of muscle. *Proceedings of the 20th International Parallel and Distributed Processing Symposium Apr. 25-29, IEEE Xplore Press*, pp: 1-7.
72. Katarzyna Rycerz, Marcin Nowak, Pawel Pierzchala, et al., (2011): Comparison of Cloud and Local HPC Approach for MUSCLE-based Multiscale Simulations, *IEEE Seventh International Conference on e-Science Workshops*: 81-88, Dec. 2011.
73. Morgenstern, B., Frech, K., Dress A. et al., (1998). "DIALIGN: Finding Local Similarities by Multiple Sequence Alignment", *Bioinformatics*, vol. 14, pp.290-294.
74. Morgenstern, B. (1999). "DIALIGN 2: Improvement on the Segment-to-Segment Approach to Multiple Sequence Alignment of DNA and Protein Sequences", *Bioinformatics*, vol. 15, pp.211-218.
75. Amarendran R Subramanian, Michael Kaufmann and Burkhard Morgenstern (2008). "Dialign-TX: Greedy and Progressive Approaches for Segment-Based Multiple Sequence Alignments, Algorithms for Molecular Biology, vol. 3.

76. Schmollinger, M., Nieselt, K., Kauffman, M. et al., (2004). "DIALIGN-P: Fast Pair-wise and Multiple Sequence Alignment using Parallel Processors", *BMC Bioinformatics*, vol. 5.
77. Emerson de Araujo Macedo, and Azzedine Boukerche (2011). Hybrid MPI/OpenMP Strategy for Biological Multiple Sequence Alignment with DIALIGN-TX in Heterogeneous Multicore Clusters. *2011 IEEE International Parallel & Distributed Processing Symposium.*
78. Saeed, Fahad and Khokhar Ashrafq, (2009). Sample-Align-D: A High Performance Multiple Sequence Alignment System using Phylogenetic Sampling and Domain Decomposition. *The 7th IEEE International Workshop on High Performance Computational Biology*.
79. Do, C.B., Mahabhashyam, M.S., Brudno, M. et al., (2005). ProbCons: probabilistic consistency-based multiple sequence alignment. *Genome Res.*, 15, 330–340.
80. Roshan, U. and Livesay, D.R. (2006). Probalign: multiple sequence alignment using partition function posterior probabilities. *Bioinformatics*, 22, 2715-2721.
81. Yongchao Liu, Bertil Schmidt and Douglas L. (2010). Maskell: "MSAProbs: multiple sequence alignment based on pair hidden Markov models and partition function posterior probabilities". *Bioinformatics*, 26(16): 1958 -196.
82. Zhang, Z., Xiao, J., Wu, J., et al., (2012). ParaAT: A parallel tool for constructing multiple protein-coding DNA alignments, *Biochem Biophys Res Commun*, 419(4):779-781.
83. Claudia Misale, Giulio Ferrero, Massimo Torquati, , et al., (2014). Sequence Alignment Tools: One Parallel Pattern to Rule Them All?. BioMed Research International. Article ID 539410. http://dx.doi.org/10.1155/2014/539410.
84. Asieh Sedaghatinia, Rodziah Binti Atan, Khairinatajul Arifin, et al., (2009). Comparison and Evaluation of Multiple Sequence Alignment Tools In Bininformatics, *Journal of Computer Science*, Volume: 9, Issue: 7, Pages: 51-56.
85. Mohamed Radhouene Aniba, Olivier Poch and Julie D. Thompson (2010). Survey and Summary Issues in bioinformatics benchmarking: the case study of multiple sequence alignment, *Nucleic Acids Research Advance Access.*
86. Thompson, J.D., Linard, B., Lecompte, O., et al., (2011). A Comprehensive Benchmark Study of Multiple Sequence Alignment Methods: Current Challenges and Future Perspectives. gg6:e18093.