An Efficient Bulk Synchronous Parallelized Scheduler for Bioinformatics Application on Public Cloud

Siddu P. Algur, PhD
Rani Chennamma University,
Belgaum

Leena I. Sakri
SDMCET,
Dharwad

ABSTRACT

Genomic sequence alignment of varied species is one of the most sort of applications in bioinformatics. In future bioinformatics technologies are expected to produce genomic data of terabyte. Bioinformatics computation require super computer for sequence alignment computation which involves huge cost. Parallelization technique is a way forward in computing sequence alignment with limited cost and time. Cloud computing and MapReduce framework play an important role in bioinformatics intensive application to achieve parallelization since it provides a consistent performance over time and it also provides good fault tolerant mechanism. The existing gene sequencing methodologies are designed based on Hadoop-MapReduce framework which adopts a serial execution strategy which is an area of concern. This work introduces a Smith-Waterman Alignment on the Bulk synchronous Parallel Map Reduce (SW-BSPMR) cloud platform for bioinformatics gene sequence alignment. This work adopts a widely accepted and accurate SW algorithm for sequence alignment and parallel synchronous scheduler methodology of map and reduce framework process is considered. A customized MapReduce based on Microsoft Azure cloud platform is developed to overcome the issue in Hadoop-MapReduce framework. The experimental study presented in this work proves that the SW-BSPMR can accurately and effectively align bioinformatics genomic sequences of various read length.

Keywords

Bioinformatics, genomic sequence, Scheduler, parallelization, hadoop, Microsoft Azure

1. INTRODUCTION

Bioinformatics is a field that comprises computer science, statistics, mathematics, and biology. One of its main goals is to analyse bioinformatics genome sequence data content in order to obtain the structure of the DNA sequences as well as evolutionary information. Once a new biological sequence is identified and generated, its structural characteristics must be recognized. The first step to achieve this process is to compare the newly obtained genomic sequence with the existing genomic sequences that compose reference database (genomic databases), in search of similarities. **Genome Sequence** is a technique or process that allows researchers to read and decode the genetic information found in the genomic data of anything from animal to plants to bacteria. Once a tiresome, painstaking process, today, thanks to advancement of techniques and the availability of super computers, it has been speed up a multiple fold. Gene sequencing involves defining the order of bases, the nucleotide subunits (Adenine, Guanine, Thymine, and Cytosine, referred to by the letters A, G, T and C) that are present in a DNA. (Humans have around 3 billion base pairs (bp).)

The bioinformatics genomic sequencing data is acquired from Next Generation Sequencing (NGS) technologies (e.g Pacific Bioscience, Illumina etc.). Most of the current genomic data available consists of millions of bioinformatics gene sequences between 32 to 100 base pairs (bp). With the recent development in bioinformatics sequencing technologies millions of bioinformatics sequences with greater than 100bp are being produced. Based on the genomic data, the different sequencing tools are developed. Examples of existing aligners include FASTA [4], BLAST [5], BLAT [6] and SOAP [7].Bioinformatics has considered with "Big Data" as the huge amounts of demonic data that is generated and used for various analysis. A sum of 3 billion US dollars and one decade of time was taken to produce the initial human reference genome containing about 3.5 billion bp. The recent developments in bioinformatics technologies available produce huge amount of genomic data in terms of gigabytes to terabyte per run [8]. Storing and analyzing such data is a problem that exists and needed to be handled. To cater these high computational needs, grid, GPU or cluster computing platforms were provided to the analyst [9]. The grid or cluster computation platforms were not adequate and were constrained by the hardware capacity of the cluster and the increasing number of concurrent access by a number of users. The ever growing gap between the computing capabilities and the sequencing performance is presented in [10]. To overcome this, using cloud computing platforms one can solve the computing and storage issue that bio sequencing brings along. Cloud computing has emerged as the future of data intensive scientific application computing paradigm [26] [27]. Cloud computing environments provide ubiquitous on demand access to shared, scalable and configurable computing resources with minimal management efforts and affordable costs [28][29]. The use of cloud computing platforms to run data intensive application or high performance computations is a widely preferred solution when compared to maintaining independent private computing clusters [30][31]. Features like resource control, resource customization, virtual platforms and elasticity in cloud environments enable easy migration of data intensive applications to the cloud.

Cloud computing platforms provide flexibility and on-demand access to cloud resources. The usage cloud computing technologies remove the seed cost that are required to maintain the high computing clusters and physical storage. Though cloud computing provides a scalable and flexible infrastructure, parallel computing models for the cloud are required to accomplish the desired aim of analyzing gene sequences. One such model for the cloud, called the MapReduce model [11] was presented by Google. The University of California at Berkeley presented the Spark platform [12] for cloud computing. A survey of all the available platforms for analysis of big data on the cloud is available in [13]. Of all the platforms that are available, Hadoop- MapReduce is by far a better and popular choice due to its open source nature and its acceptability by various industry and research academic organizations. In the recent years to effectively utilize the full potential of the cloud, public cloud service providers such as Amazon, Google, and Microsoft Azure etc. offer virtualized computing resources for both hardware and software [14].The virtualization enables user specific customization, flexibility, application execution environments, cost efficiency and reduce power consumption. [15] [16].

The Hadoop framework [17] adopts the MapReduce framework for computing a user specific application on the cloud platform. In the MapReduce model a dual phase execution approach is adopted. In the initial phase the input data to be processed is split into chunks. Each chunk is associated with a mapper or a map worker. The map worker provides the ⟨Key│Value⟩ pairs as outputs that are sorted on the basis of the Key values. The sorted values are provided to the reduce workers i.e. ⟨Key│SortedList(Value) ⟩. The reduce workers store the results in the Hadoop distributed file system. The Map and Reduce workers are generally Virtual Machines (VM's) in the public cloud environments. The Next Generation Sequencing (NGS) tools like CloudAligner [18] and Cloud Burst [19] adopt the Hadoop map reduce framework as the computing platform. The major drawback of these alignment tools is that they are work well for small base pair when single-gap or un-gapped alignment is to be computed. When working with bigger sequence exhibit performance degradation and effect accuracy proved by the results presented in [24]. From the section one, it is evident that all the existing sequence aligners for cloud environments consider the Hadoop framework. Genome alignment is an iterative process and Hadoop suffers when iterative applications are hosted on the framework [20] [21]. The performance of Hadoop suffers when multiway joins are considered [22]. The Hadoop framework considers sequential processing of the map and then the reduce stages that effects performance [23].

In this paper a Smith-Waterman Alignment on the Parallel Azure Map Reduce (SW-BSPMR) to perform genomic sequence alignments on a cloud platform is presented. The SW-BSPMR uses the SW presented in [1] to perform sequence alignments of genomic data. A MapReduce based framework is considered for the execution on the public cloud infrastructure environment. The Smith-Waterman (SW) algorithm [2] [3] in the SW is optimized using MapReduce parallel computation technique using public cloud infrastructure which is discussed in the latter section of the paper. To overcome the issues and limitation of the Hadoop-MapReduce a parallel execution methodology of the Map and Reduce workers is considered. The Map and Reduce task executions on the VM based cloud worker node is parallelized to reduce the task completion times. The bioinformatics sequence aligner presented in [24] bears the accurate similarity of to the SW-BSPMR proposed and is considered for the performance comparisons.

The paper organization is as follows: The research carried out so far is shown in section two and the SW algorithm and its optimization considered are presented in Section three. The results and the experimental study are presented in the section four. The concluding remark is discussed in the last section.

2. LITERATURE SURVEY

These days, each and every scientific research laboratory is able to generate terabytes or petabyte of data (or even more), which is not a surprises to new sequencing technologies in current genomic research. With the improvement in gene sequencing technology/expertise, the scientific/genomic data generated by the sequencer is becoming much cheaper and better. As a result, more genomic data is increasingly being generated or processed which leads to serious issues or concern in storing (handling) and processing. HPC environments keep improving which helps in processing large-scale data at low cost. Combining Cloud infrastructure and MapReduce (MR) technology together is evolving as one of the finest solution to handle this. However, the current tools (technique) of this trend are lacking the efficiency and common features that are found in other well-known tool making them unappealing to the users. Here the analysis is done on some of the existing mechanism and its pros and cons.

D Dahiphale et al. [32] described the issues of the conventional MapReduce frameworks as a) The MapReduce framework uses a sequential processing of the Map and Reduce stages. b) The scalability of the Map Reduce is limited. c) The MapReduce framework provides no support for elastic pricing options. d). The MapReduce model provides no provision for computing streaming data. To overcome these drawbacks a pipelined model is adopted to parallelize the execution of the map and reduce phase. The MapReduce model proposed in [32] is realized on the Amazon public EC2 cloud. The spot instance offering of the Amazon cloud enables flexible pricing. The experimental study presented considering the word count application proves the efficiency of the pipelining based MapReduce model when compared to the conventional MapReduce model. The major drawback of the model proposed in [32] is that the locality optimization is not considered and hosting of additional data dependent applications like Smith Waterman etc. cannot be executed in a pipelined fashion.

The computation of certain data intensive applications like graph applications and iterative applications on MapReduce frameworks exhibit high computation times and costs. To support such applications Google introduced a proprietary cloud computing framework named Pregel [33] that adopts the BSP computing model. In Pregel the graph computations is achieved using a set of super-steps. A super step is used to execute the user defined application or function in a parallel fashion using the data item from the database. Each data item from the database behaves as an agent. The Pregel system adopts vertex-centric execution strategy. The computation of each data item has a graph like representation in BSP. The vertexes in the Pregel deactivate post the computation operation and are reactivated only if additional data items are presented to them. Once all the vertices are deactivated the computation is said to be complete. The local storage of the data items in the nodes executing the computation pose a problem. In the case the data item is large then a spilling-to-disk techniques needs to be in place [34]. The applicability of Pregel for additional applications like imprecise applications, non-graph based external applications, biomedical applications etc. has not been discussed so far .

Hyungro Lee [35] here they analyzed some of the existing bioinformatics application by using cloud and MapReduce technology such as CloudBurst, CloudBLAST etc… there survey shows that map reduce framework is an efficient way of handling sequence data. CloudBLAST used hadoop distributed MapReduce framework to compute similarity between the genes. They also analyzed scientific workflow system such as Galaxy which offers simple web-based workflow toolkits and scalable scientific computing environments and challenges involved in transferring sequence data.

Michael C. Schatz [36] CloudBurst employs (uses) alignment strategy modelled based on RMAP to map with any number of differences or mismatches. With the support of HDFS, CloudBurst genomic alignment scales linearly well as number of reads increases and speeds up linearly as the size of the cluster increases. However, CloudBurst have not enough enrich features as traditional map/align tools has. The most serious and important obstacle is that CloudBurst doesn't support pair-end reads and fastq format input. CloudBurst is a MapReduce (MR) based read-mapping framework modelled, but it runs in parallel on multiple vm with Hadoop MR Framework. It is optimized for mapping many short reads from subsequent generation genomic sequencing machines to a reference genome by allowing dynamically for a user specified number of differences or mismatches. It is a type of seed-and-extend mechanism that indexes the non-overlapping k-mers in the reads as seed. The seed size s = r / (m+1) is then computed from the min length of the reads (r) and the max number of differences or mismatches (m). Then it try to extend the exact seed to count the number of mismatch in an end-to-end alignments using those seeds, and then reports alignment with at high m mismatches. Finally extend the exact seeds matched into an end-to-end gapped alignment by using a dynamic programming (DP) algorithm.

Tung Nguyen et al. [37] CloudAligner address drawback of the existing tools (CloudBurst) and also to promote a Cloud and MR-based solution for genomic issues. Particularly, CloudAligner is specifically designed to achieve better efficacy, longer reads, performance, and extremely high scalability. Cloud aligner has more important general tasks such as bisulfite and pair-end mapping as well as a user friendly interface, and it supports more input and output formats. Cloud aligner works as follows there are 2 main or required input files for CloudAligner namely the read file and the reference file. The reference files are generally in the fasta format whereas the read files can be in the Fastq or Fasta format. Both are changed into the binary files and copied to the hadoop distributed file system or Amazon S3 cloud. When executing, CloudAligner splits the read or query files into smaller pieces called input splits and distributes them to the different maps. Each map aligns its input split onto the whole reference genome files.

LI Xubin [38] CloudBLAST integrates MapReduce (MR) together with virtual machine (VM) and virtual network technologies. Here CloudBLAST have been evaluated in both non-virtualized and LAN-based implementation. Here they compared between mpiBLAST and CloudBLAST and result shows that the CloudBLAST is more efficient than the mpiBLAST. It is the first time that Hadoop in wide-area network achieves satisfied performance. CloudBLAST improves (achieves) efficiency by splitting both of query sequences and sequence database for alignment. Original method as CloudBLAST deals with query sequences by splitting and sending them to different VM computing nodes, while copying complete gene sequence database (DB) to each VM computing node. Therefore, the dual segmentation method presented here a good performance for generalized large sequence database (DB).

Xiao-liang Yang et al. [39] here they proposed a heuristic method blastn which seed-and-extend to search for high scoring gene sequence alignments between the input genomic query sequence and genomic sequences in the database (DB) by using hadoop map reduce model. It has the following three main steps such as building a word list, scanning for hits, and extending hits. In phase one it builds a word list contains all the contiguous w-mers in the query sequence of nucleotides, then it adopts a new 'two-hit' technique requiring the existence of 2 non-overlapping word pairs within a distance D of one another, to trigger an extension and Finally a gap free extension is invoked in 2 directions to find an alignment called a HSP (high-scoring segment pair). If the generated high-scoring segment pair scores above a threshold T, then a gapped extension is triggered, and the alignment result is stated. An enhanced dynamic programming algorithm, called X-drop, is used in the gapped extension to construct the gapped local alignment. The issue with this model is that it as an overlap of 1Mbp at most following each fragment limits the range of alignment which falls in it.

Based on above survey , A parallelized gene sequence model based on SW by incorporating Microsoft Azure cloud infrastructure platform for computation is developed.

3. PROPOSED SYSTEM

The proposed SW-BSPMR provides a cloud platform to perform sequence alignments considering genomic data obtained from NSG techniques. The SW-BSPMR adopts a cloud computing infrastructure framework for MapReduce computation. To support scalability in SW-BSPMR, the Map and Reduce Azure worker nodes are deployed on a Microsoft Azure cloud cluster of VM's. For genomic sequence alignments the SW algorithm is adopted. The advantages of using the SW algorithm for sequence alignments and its advantages over the existing aligners are found in [1]. The SW-BSPMR considers the genomic alignment in dual phases i.e. Map and Reduce phase. The existing aligners on the cloud platform uses Hadoop framework. In the Hadoop framework based solutions the Reduce phase is initiated when Map phase is completed. To overcome the drawbacks or disadvantage of Hadoop a parallel execution mechanism of the Map and Reduce phases is considered. Optimization of the Smith Waterman algorithm is an additional feature considered in SW-BSPMR. Execution of the Map and Reduce functions are designed to run in parallel and effectively utilize the cores available in the worker VM's.

3.1 Smith Waterman Algorithm

The Smith-Waterman algorithm is a well-known dynamic programming algorithm (DPL) for performing local sequence alignment for determining similar (unique) regions between two protein or DNA sequences. The algorithm was first proposed by T. Smith and M. Waterman in 1981. In recent years, it is still a spine /core algorithm of many bioinformatics applications. The algorithm consists of following two steps:

1. Calculate the similarity matrix score.

2. According to the dynamic programming method, trace back the similarity matrix to search for the optimal alignment. In the algorithm, the first step will consume the largest part of the total processing (calculation) time.

Importance of Smith-Waterman Algorithm

Sequence similarity searches performed using the Smith-Waterman algorithm guarantees the optimal local alignments between query and database (Reference) sequences. Thus, ensuring the best performance in term of accuracy and the most precise results - aspects of significant importance when you cannot afford to miss any information gained from the similarity search as e.g. when searching for remote homology. The Smith-Waterman algorithm is one of being the most sensitive algorithm for detection of sequence similarity.

3.2 Bioinformatc Sequence Alignment

To compare two DNA or RNA sequences, the optimal alignment between sequences, which is to place one DNA sequence above the other making clear the correspondence between similar characters is to be found. In an alignment, spaces can be placed in arbitrary positions along the DNA or RNA sequences. Basically, an alignment can be global, containing all characters of the DNA or RNA sequences; local, containing substrings of the genomic sequences; or semi global, composed of prefixes or suffixes of the genomic sequences, where trailing gaps are not considered. In order to measure the resemblance between two bio-sequences, a score is calculated as follows: given an alignment between genomic sequences $\mathbb{Sq}_0$ and $\mathbb{Sq}_1$, the following scores are assigned, for example, for each column:

$\mathbb{M}_{\mathbb{a}} = +1$,If both characters are same i.e. exact match;

$\mathbb{M}_{\mathbb{i}} = -1$, if the characters are not same i.e. mismatch; and

$\mathbb{Gp}_{\mathbb{s}} = -2$, if one of the characters is a gap.

The score is the cumulative result of all these values. A constant value is assigned to gaps. However, keeping gaps together generates more significant results, in a biological perspective. For this reason, the first gap must have a greater penalty than its extension. The penalty for the initial gap is $\mathbb{Gp}_{first}$ and for each successive gap, the penalty is $\mathbb{Gp}_{extension}$. The difference $\mathbb{Gp}_{first} - \mathbb{Gp}_{extension}$is the gap opening penalty $\mathbb{Gp}_{open}$.

The definition of Smith-Waterman (SW) Algorithm is given below

The algorithm SW is an exact method based on dynamic programming to find the optimal local sequence alignment between two genome sequences in quadratic time and space. The SW algorithm was modified by Gotoh [24] in order to calculate extension gap penalties. It is divided in to following phases: calculate the Dynamic Programming matrices and obtain the optimal local alignment.

3.3 Calculate Dynamic Programming Matrices

The algorithm receives input genome DNA sequences $\mathbb{Sq}_0$ and $\mathbb{Sq}_1$, with sizes a and b, respectively. For genomic DNA sequences $\mathbb{Sq}_0$ and $\mathbb{Sq}_1$, there are $a+1$ and $b+1$ possible prefixes, respectively, that also include the nullgenome sequence. The notation used to represent the b^{th}character of a sequence $\mathbb{S}_q$ is $\mathbb{S}_q[b]$ and, to represent a prefix with b characters, the author use$\mathbb{S}_q[1 \dots b]$. The DNA sequence similarity matrix is denoted $\mathbb{SM}$, where $\mathbb{Sq}_{\mathbb{x},\mathbb{y}}$ contains the DNA sequence similarity score between prefixes $\mathbb{Sq}_0[1 \dots x]$ and$\mathbb{Sq}_1[1 \dots y]$. Initially the first row and column (1, 1) are occupied with zeroes. The remaining elements of $\mathbb{SM}$ are obtained from (1), where $\mathbb{Pm}(x, \mathbb{y}) = \mathbb{M}_{\mathbb{a}}$(match) if $\mathbb{Sq}_0[i] = \mathbb{Sq}_1[j]$and $\mathbb{M}_{\mathbb{i}}$ (mismatch) otherwise. To calculate gaps based on the affine gap model, two additional matrices $\mathbb{ME}(2)$ and $\mathbb{MF}(3)$ are required. By considering this, time complexity remains quadratic. The optimal score between DNA sequences $\mathbb{Sq}_0$ and $\mathbb{Sq}_1$ is the highest value in $\mathbb{SM}$ and the position (x, y) where this value occurs represents the end of alignment

$$\mathbb{SM}_{x,y} = max \begin{cases} 0 \\ \mathbb{ME}_{x,y} \\ \mathbb{MF}_{x,y} \\ \mathbb{SM}_{x-1,y-1} - \mathbb{Pm}(x,y) \end{cases} \quad (1)$$

$$\mathbb{ME}_{x,y} = max \begin{cases} \mathbb{ME}_{x,y-1} - \mathbb{Gp}_{extension} \\ \mathbb{SM}_{x,y-1} - \mathbb{Gp}_{first} \end{cases} \quad (2)$$

$$\mathbb{MF}_{\mathbb{x},y} = max \begin{cases} \mathbb{MF}_{x-1,y} - \mathbb{Gp}_{extension} \\ \mathbb{SM}_{x-1,y} - \mathbb{Gp}_{first} \end{cases} \quad (3)$$

Find the optimal alignment phase: To get the perfect local alignment, the proposed algorithm starts from the cell that has the maximum score and continues until a zero-valued cell is reached $\mathbb{SM}_{x,y}$indicates the alignment of $\mathbb{Sq}_0[x]$with a gap in $\mathbb{Sq}_1$ and the alignment of $\mathbb{Sq}_1[y]$ with a gap in $\mathbb{Sq}_0$. Finally, an arrow on the diagonal indices specifies that $\mathbb{Sq}_0[x]$ is aligned with $\mathbb{Sq}_1[y]$. Below figure is the representation of Smith Waterman model.

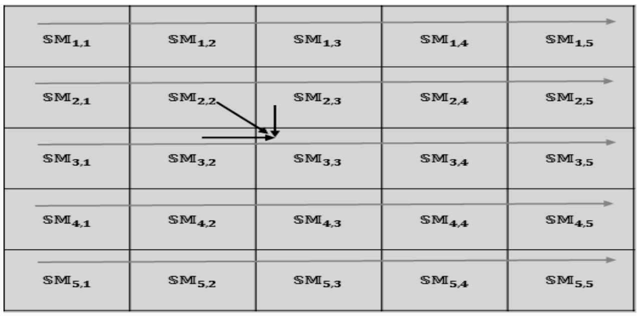
SM1,1 SM1,2 SM1,3 SM1,4 SM1,5
SM2,1 SM2,2 SM2,3 SM2,4 SM2,5
SM3,1 SM3,2 SM3,3 SM3,4 SM3,5
SM4,1 SM4,2 SM4,3 SM4,4 SM4,5
SM5,1 SM5,2 SM5,3 SM5,4 SM5,5

Fig 1: Smith waterman algorithm to compute $\mathbb{SM}$ used in traditional Hadoop- MapReduce framework.

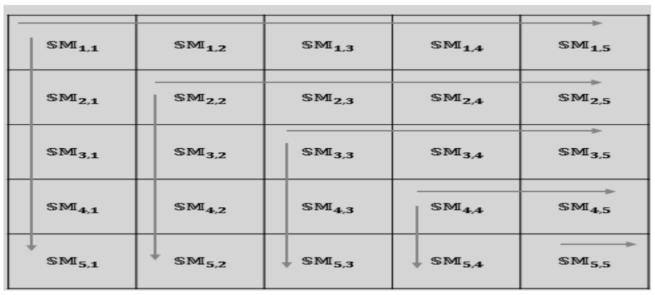
SM1,1 SM1,2 SM1,3 SM1,4 SM1,5
SM2,1 SM2,2 SM2,3 SM2,4 SM2,5
SM3,1 SM3,2 SM3,3 SM3,4 SM3,5
SM4,1 SM4,2 SM4,3 SM4,4 SM4,5
SM5,1 SM5,2 SM5,3 SM5,4 SM5,5

Fig 2: Parallelization technique used to compute $\mathbb{SM}$ in smith waterman algorithm to reduce computation time of VM

3.4 Proposed Smith-Waterman Alignment on the Parallelized Map Reduce model SW-BSPMR

Let $\mathcal{D}$ denote a bioinformatics reference genomic sequence and $\mathcal{Sq}$ the genomic query sequence. The $SW-BSPMR$ is deployed on a public cloud platform namely Azure consist of a master node, map and reduce worker computing nodes. The master computing node of $SW-BSPMR$ initializes cw Map and Reduce worker computing nodes using the virtual computing nodes. Every virtual computing node is presumed to$\mathcal{C}$CPU cores available for task computation. Let $\mathbb{T}_{VM_inz}$denote the time engaged to initialize the virtual platform. The bioinformatics reference sequence $\mathcal{D}$ is split into $\mathcal{D}'$ chunks of sequence with overlapping sections. A reference chunk sequence and the $\mathcal{Sq}$ isthen sent to the map computing nodes as input key, value pairs. The key value pairs considering the reference chunk$\mathcal{D}'$is denoted as (kd, vd), where kd is the key and vd which contains the overlapping offset genome data. The key value pair of genome query sequence$\mathcal{Sq}$ is denoted as(ksq, vsq), where ksq is the key and vq is the genome query sequence. In each of the cw computing map workers, query sequence $\mathcal{Sq}$ is again divided into $\mathcal{Sq}'$ chunks and kept in the available local memory storage. Sequence alignment is done using the smith waterman algorithm considering $\mathcal{D}'$and every single$\mathcal{Sq}'$in a parallel approach using the $\mathcal{C}$cores. The smith waterman algorithmis optimized by a parallelization technique to cut sequence alignment completion times. Let $\mathbb{T}_{\mathbb{MP}}$ depicts the average completion time of the $\mathcal{cw}$ map computing nodes. The map workers post computations provide alignment positions between $\mathcal{Sq}$ and reference chunk$\mathcal{D}'$along with the computed score. The multiple alignment positions and computed score i.e. $\mathcal{vmp}$ along the chunk id of $\mathcal{D}'$i.e. $\mathcal{kmp}$ is stored in the temporary cloud storage memory as $list(\mathcal{kmp}, \mathcal{vmp})$. The map function of the $SW-BSPMR$ can be represented as $map\big((\mathcal{kd}, \mathcal{vd}), (kq\mathcal{ksq}, \mathcal{vsq})\big) \rightarrow list(\mathcal{kmp}, \mathcal{vmp})$.

The $\mathcal{cw}$reduce computing worker nodes get the intermediate genomic data i.e. $list(\mathcal{kmp}, \mathcal{vmp})$execute the shuffle and sort task. In the reduce phase, the collection of all alignment positions i.e. $list(\mathcal{v}gd)$that are non-overlapping and non-redundant is considered. The reduce process can be represented as $reduce\big(\mathcal{kmp}, list(\mathcal{vmp})\big) \rightarrow list(\mathcal{vgd})$. Let $\mathbb{T}_{\mathbb{RP}}$denote the average time taken by the $\mathcal{cw}$ reducecomputing worker nodes to perform the shuffle, sort and reduce process. The total execution time of the $SW-BSPMR$ cloud platform to align the genomic sequence $\mathcal{Sq}$ against $\mathcal{D}$ is computed as,

$$\mathbb{T} = \mathbb{T}_{VM\_inz} + \mathbb{T}_{\mathbb{MP}} + \mathbb{T}_{\mathbb{RP}} \,. \tag{4}$$

The overview of the $SW-BSPMR$ cloud platform is shown in Fig. 3.

Map phase

In the map phase of the $SW-BSPMR$ Azure cloud platform, sequence alignment positions and corresponding sequence scores between query sequence $\mathcal{Sq}$ and reference sequence chunk$\mathcal{D}'$ are computed. The required sequence data i.e. $\mathcal{Sq}$ and chunk$\mathcal{D}'$, is obtained from the cloud storage memory. Let $t_{\mathcal{Gt_Dt_Mt}}$denote the time taken to get the sequence data. To reduce the sequence alignment computation times using parallel computing techniques the sequence $\mathcal{Sq}$ is divided into$\mathcal{Sq}'$chunks. Let $t_{\mathcal{S}I}$denote the time taken to split query sequence $\mathcal{Sq}$ into$\mathcal{Sq}'$ chunks. The sequence alignment considering one chunk of $\mathcal{Sq}'$and$\mathcal{D}'$is done using the smith waterman algorithm. The seed matches of genomic sequences$\mathcal{Sq}'$and $\mathcal{D}'$is computed by using a dynamic programming methodology. The seeds computed are extended to guarantee rule alignment of the genomic sequences using the $smith\ waterman$ algorithm.

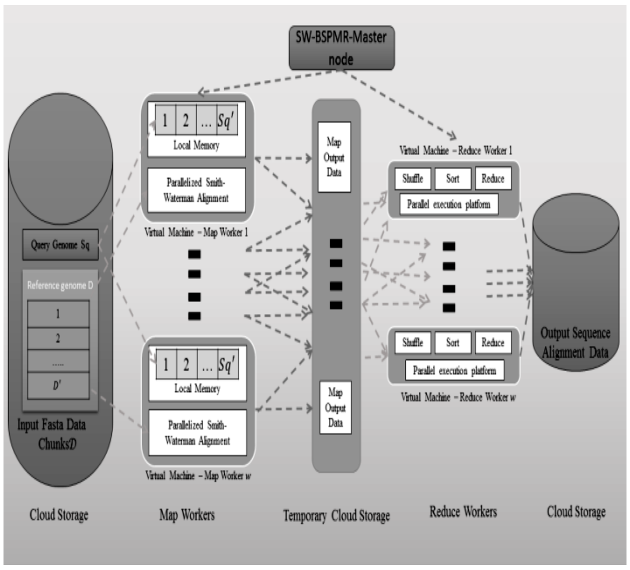
SW-BSPMR-Master node
1 2 ... Sq'
Local Memory
Parallelized Smith-Waterman Alignment
Virtual Machine – Map Worker 1
Query Genome Sq
Reference genome D
1
2
.....
D'
Input Fasta Data Chunks D
1 2 ... Sq'
Local Memory
Parallelized Smith-Waterman Alignment
Virtual Machine – Map Worker w
Map Output Data
Map Output Data
Virtual Machine – Reduce Worker 1
Shuffle
Sort
Reduce
Parallel execution platform
Shuffle
Sort
Reduce
Parallel execution platform
Virtual Machine – Reduce Worker w
Output Sequence Alignment Data
Cloud Storage
Map Workers
Temporary Cloud Storage
Reduce Workers
Cloud Storage

Fig 3: Proposed SW-BSPMR Cloud Model for Bioinformatics sequence alignment

To reduce sequence alignment computation time parallelization of the SW algorithm is considered in$SW-BSPMR$. The parallelization technique to optimize computation is discussed in the latter sub-section.

Lett_{SW}denote the time taken to align $\mathcal{Sq}'^{th}$chunk and$\mathcal{D}'$using the SW algorithm. The complete time taken to align the entire$\mathcal{Sq}'$chunks is therefore$(\mathcal{Sq}' \times t_{SW})$. As $\mathcal{C}$ virtual computing cores are available with each cloud worker node, the parallel computation of $\mathcal{C}$ number of chunks of $\mathcal{Sq}$ is probable. The computation time considering the total$\mathcal{Sq}'$chunks and $\mathcal{C}$ cores is represented as$\left(\frac{\mathcal{Sq}' \times t_{SW}}{\mathcal{C}}\right)$. The alignment positions and alignment scores are stored in the cloud container for reduce computing workers. The time engaged to store this data per map computing worker is denoted as$t_{\mathcal{Int_Str_Dt_M}}$. The total time of the $\mathcal{cw}^{th}$map computing worker node can be represented as

$$\mathbb{T}_{\mathcal{cw\_MP}} = \left(t_{\mathcal{Gt\_Dt\_Mt}} + t_{\mathcal{Sq}I} + \frac{\mathcal{Sq}' \times t_{SW}}{\mathcal{C}} + t_{\mathcal{Int\_Str\_Dt\_M}}\right). \tag{5}$$

The average execution time of the $\mathcal{cw}$ map computing nodes is defined as

$$\mathbb{T}_{\mathcal{MP}} = \frac{\sum_{n=1}^{\mathcal{cw}} \mathbb{T}_{n\_\mathcal{MP}}}{\mathcal{cw}} \tag{6}$$

Reduce Phase

The master node in$SW-BSPMR$initializes $\mathcal{cw}$ reduce computing nodes and $\mathcal{cw}$ map computing nodes concurrently. This parallel initialization methodology allows reducing the total execution time $\mathbb{T}$. In the reduce phase the shuffle task obtains the intermediate data (produced by the computing nodes) from the cloud storage. The sequence alignment positions obtained from the cloud intermediate data are sorted based on the offset data. The time taken by the $\mathcal{cw}^{th}$reduce computing node, to get the intermediate data, shuffle and sort it is represented as$t_{\mathcal{Gt_Dt_Rt}}$. The reduce function in$SW-BSPMR$is used to aggregate the alignment positions. The redundant and overlapping alignments are neglected. Parallelization of the reduce function is considered in $SW-BSPMR$ to utilize the $\mathcal{C}$ virtual computing cores available. Let $\frac{t_{cmp\_\mathcal{R}}}{\mathcal{C}}$ denote the time taken to compute the reduce job utilizing the $\mathcal{C}$ virtual computing cores. The reduce phase provides the sequence alignment results between the genomic sequences $\mathcal{D}$ and $\mathcal{Sq}$ that are written to the cloud container storage for further examination. Let $t_{\mathcal{Int\_Str\_Dt\_R}}$ represent the time taken, by the cw^{th} reduce computing node to write the results onto the cloud container storage. The total time of the cw^{th} reduce computing node can be represented as

$$\mathbb{T}_{cw\_\mathcal{Rp}} = \left(t_{\mathcal{Gt\_Dt\_Rt}} + \frac{t_{\mathcal{Cmp\_R}}}{\mathcal{C}} + t_{\mathcal{Int\_Str\_Dt\_R}}\right). \tag{7}$$

The average execution time of the cw reduce computing nodes i.e. $\mathbb{T}_{\mathcal{RP}}$ is defined as

$$\mathbb{T}_{\mathcal{RP}} = \frac{\sum_{n=1}^{cw} \mathbb{T}_{n\_\mathcal{RP}}}{cw} \tag{8}$$

Using (6), (8) the total execution time of the $SW-BSPMR$ can be defined as

$$\mathbb{T} = \mathbb{T}_{VM\_\mathcal{Inz}} + \frac{\sum_{i=1}^{cw}\left(\mathbb{T}_{n\_\mathcal{MP}} + \mathbb{T}_{n\_\mathcal{RP}}\right)}{cw}. \tag{9}$$

From (9) it can be observed that the computing time of the $SW-BSPMR$ depends on the computation time of the map and reduce computation nodes. The core alignment steps (i.e. based on the SW algorithm) are done by the map computing nodes i.e. $\mathbb{T}_{\mathcal{RP}} \ll \mathbb{T}_{\mathcal{MP}}$. The time taken to start the map and reduce computing $VM's$ i.e. $\mathbb{T}_{VM\_\mathcal{Inz}}$ is dependent on the cloud platform user for deployment and can be ignored. Therefore it can be stated that the total sequence alignment execution time

$$\mathbb{T} \approx \frac{\sum_{n=1}^{cw}\left(\mathbb{T}_{n\_\mathcal{MP}}\right)}{cw}. \tag{10}$$

Optimizing the SW algorithm in SW is a possible solution to reduce the total execution time $\mathbb{T}$

Smith Waterman Algorithm Optimization

In the SW algorithm computation of the similarity matrix score i.e. $\mathbb{SM}$ consumes the maximum time. Adopting the parallelization technique for computation of $\mathbb{SM}$ is considered in $SW-BSPMR$. To parallelize the computation of $\mathbb{SM}$, the adoption of CUDA/GPU based techniques is considered. The availability and the cost of such computing platforms on the public clouds is an issue. To maximize the utilization of resources available with the VM computing node at minimal costs in $SW-BSPMR$, . The parallel computation technique is shown by grey parallel lines in Fig. 2. The computation of all the cells of $\mathbb{SM}$ that fall under the parallel lines can be computed in a parallelized fashion.

4. SIMULTION RESULT AND ANALYSIS

The system environment used is windows 7 enterprises 64-bit, 8 GB ram, i-5 quad core processor operating system. Dot net framework 4.0 and C# 6.0 programming language is used for the proposed work and java programing language is used for the existing Hadoop and conducted experimental study on following parameter for linear speed ups and task completion time and compared the proposed gene sequencing model with existing sequencing model.

To evaluate the performance of SW-BSPMR comparison with the SW-Hadoop is considered. The deployment of SW-Hadoop and SW-BSPMR is considered with one VM computing node. The SW-BSPMR is deployed on the Microsoft Azure cloud platform. The SW-Hadoop is designed using the Hadoop- MapReduce framework. The Apache Hadoop & YARN 2.6.0 version is used in the deployment of the SW-Hadoop. Identical configurations of VM computing nodes are considered in the deployments. The $SWBSPMR$ model is deployed on the Azure cloud considering A3 VM instances. Each A3 VM instance consists of 4virtual computing cores, 7 GB of RAM and 120 GB of local hard drive space. The $SW-BSPMR$ model deployed on the Azure cloud platforms consists of one master node and four worker nodes to perform the map and reduce jobs/tasks. Using Azure HDInsight [40] [41]. HDInsight enables deployment and provisioning of Apache Hadoop clusters on the Azure cloud platform. The Apache Hadoop & YARN version 2.6.0 is considered for performance evaluation. The master node of the Azure cluster runs on the Windows Server 2012 R2 operating system. A cluster of 4 worker nodes of A3 VM instances is considered for the Hadoop deployment. The baker yeast genomic database (Saccharomyces cerevisiae S288c) is considered for evaluation [25]. Experiments are conducted using a constant reference genomic sequence and four query sequences of varied lengths are considered. The experiments conducted with the reference and query genomic sequences are summarized in Table 1. Considering the smith waterman sequence alignment computation on the SW-BSPMR and SW-Hadoop clusters the total time taken to execute the alignment is monitored. The time taken of the Map stage (which is shown in Fig 4&7) and the Reduce stage (which is shown in Fig 5&7) along with the total time taken to complete sequence alignment (which is shown in Fig 6&7) is observed. In Fig 9 the total time taken to align sequence by different map worker with varied query sequence size is shown and in Fig 10 the total time taken by different reduce worker with varied query sequence size is shown. The results obtained prove that the proposed SW-BSPMR aligner deployed on Azure outperforms the SW-Hadoop. In experiment 1 (sequence length of 1K) the speed up achieved for SW-BSPMR is about 12.5. For sequence alignments i.e. in experiment 4 (sequence length of 500k which is shown in Fig 4) the speedup was observed to be 43.5. As it is seen that the query length increases the computation time to align sequence also increase and also the performance of SW-Hadoop severely degrades with increase in query size when compared with the performance of the proposed SW-BSPMR. Fig 8 shows that an average speed up of 23.5 is achieved considering SW-BSPMR when compared to the SW-Hadoop.

Table 1: Experiment information considered to compare the performance of SW – BSPMR with the SW – Hadoop

No	Reference Genome	Length	Query Genome	Length
1	Saccharomyces cerevisiae S288cchromosome XII	1001933 bp	Saccharomyces cerevisiae KillerVirusM1_1996_NC001782	1859 bp
2	Saccharomyces cerevisiae S288c chromosome XII	1001933 bp	Saccharomyces cerevisiae virus L-BC_ NC_001641.1	4478 bp
3	Saccharomyces cerevisiae S288c chromosome XII	1001933 bp	Saccharomyces cerevisiae S288c chromosome_ NC_001224	85984 bp
4	Saccharomyces cerevisiae S288c chromosome XII	1001933 bp	Saccharomyces cerevisiae S288c chromosome VIII	559066 bp

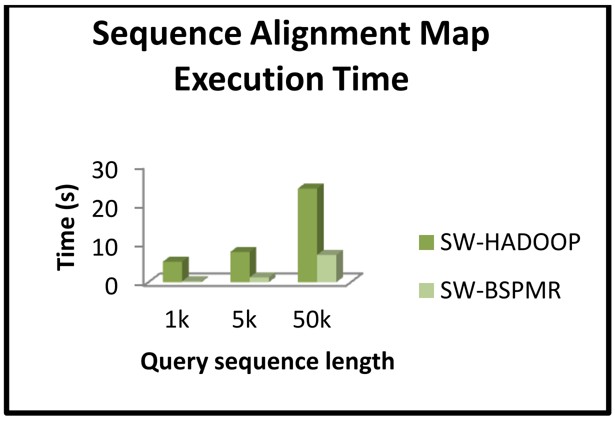

Fig 4: Sequence alignment map execution

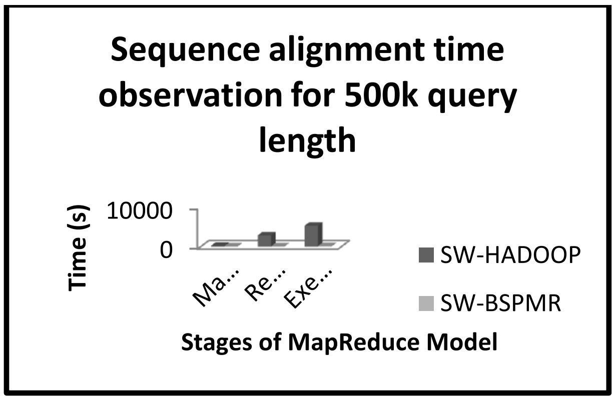

Fig 7: Observation time for 500k query length.

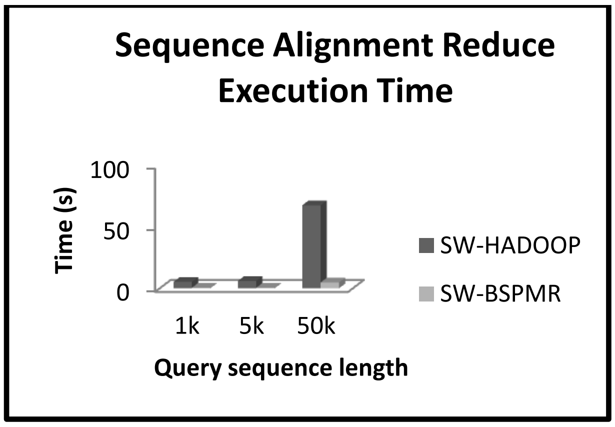

Fig 5: Sequence alignment reduce execution

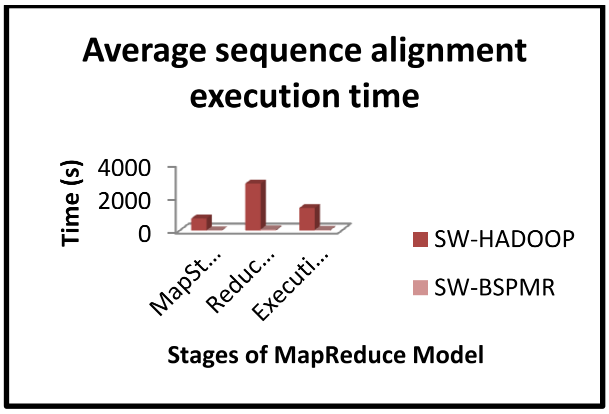

Fig 8: Average sequence alignment execution time

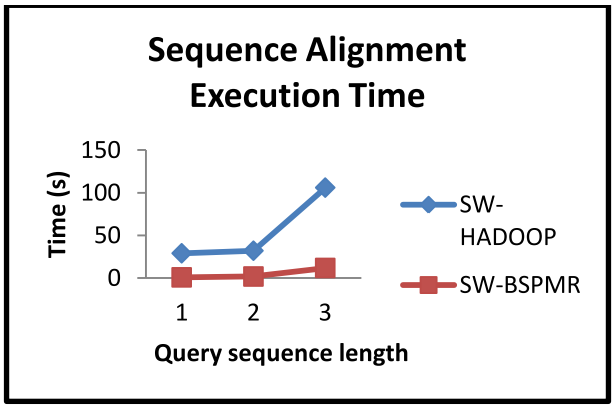

Fig 6: Sequence alignment execution time

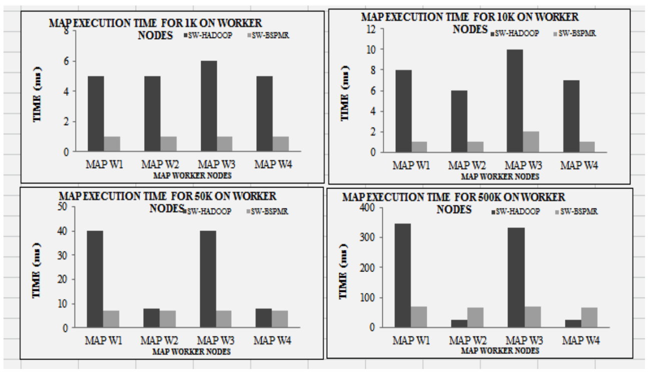

Fig 9: Map execution time for varied sequences.

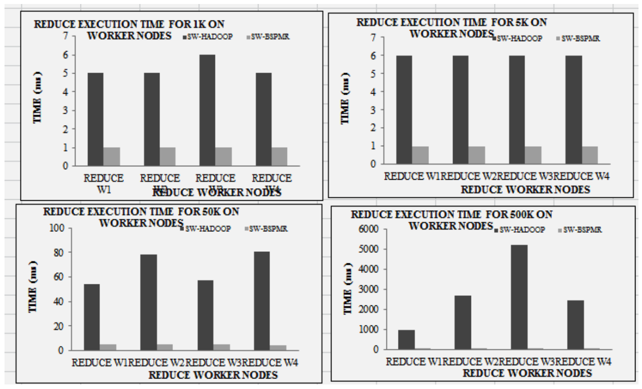

Fig 10: Reduce execution time for varied sequence

5. CONCLUSION

To solve the issues related to data storage and data intensive computation cloud platforms are used. Bio-sequencing analysis of genomic data is a vital application. The enormous growth of the NSG technologies produce huge amount of bioinformatics data. For analysis of genomic data various alignment tools are used. The current sequence aligners exhibit deficiencies in alignment of sequence genomic data that are currently been used. The existing bioinformatics sequence aligners that uses hadoop MapReduce framework for computation suffer from issues that are discussed in this work. In this paper the SW-BSPMR cloud platform is used align sequences is proposed. The smith waterman algorithm is been used for bio-sequence alignment in the SW-BSPMR cloud platform and a parallel Map Reduce execution methodology is used. A parallel execution of the map and reduce framework is utilized to maximize the utilization of the virtual machine based cloud computing platform. The paper also highlights the comparison of the proposed SW-BSPMR with the existing systems sequence alignments. Experiments to prove the efficiency of the optimized SW algorithm is presented. Comparison with the SW-Hadoop for sequence alignment is presented through experimental study. The results obtained shows significant improvement considering the SW-BSPMR when compared to the SW-Hadoop. In the future the authors propose to undertake the efficiency of the proposed scheduler by running varied application on it.

6. REFERENCES

[1] Taylor N. Job and Jin H. Park "Exploiting High Performance on Bioinformatics Applications in a Cloud System", vol. 22, no. 2, pp.22-24, 2014

[2] T.F. Smith and M.S. Waterman, "Identification of Common Molecular Subsequences," J. Molecular Biology, vol. 147, no. 1, pp. 195-197, Mar. 1981

[3] O. Gotoh, "An Improved Algorithm for Matching Biological Sequences," J. Molecular Biology, vol. 162, no. 3, pp. 705-708, Dec. 1982.

[4] W.R. Pearson and D.J. Lipman "Improved Tools for Biological Sequence Comparison" US National Academy of Sciences, vol. 85, pp. 2444-2448, 1988.

[5] S. Altschul, T. Madden, A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. Lipman, "Gapped BLAST and PSI-BLAST: A New Generation of Protein Database Search Programs," Nucleic Acids Research, vol. 25, pp. 3389-3402, 1997.

[6] W. James Kent "BLAT-The BLAST-Like Alignment Tool", Genome Res., vol. 12, no. 4, pp.656 -664 2002

[7] Li R, Li Y, Kristiansen K, Wang J. SOAP: shortoligo nucleotide alignment program. BMC Bioinformatics 24(5):713714, 2008.

[8] T. Nguyen, et al., "CloudAligner: A fast and full-featured MapReduce based tool for sequence mapping," BMC Res Notes, vol. 4, p. 171, 2011.

[9] Bakery M, Buyyaz R. Cluster computing at a glance. In: Buyyaz R, ed. High Performance Cluster Computing: Architectures and System. Upper Saddle River, NJ: Prentice-Hall; 1999:3–47.

[10] Schatz MC, Langmead B, Salzberg SL: Cloud computing and the DNA data race. Nat Biotechnol 2010, 28(7):691–693.

[11] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," in OSDI, 2004, pp. 137–150.

[12] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica "Spark: Cluster Computing with Working Sets," in Proceedings of the 2nd USENIX Conference on Hot topics in Cloud Computing, (Boston, MA), June 2010.

[13] D. Singh and C. K. Reddy, "A survey on platforms for big data analytics," Journal of Big Data, vol. 2, article 8, 2014.

[14] Jianhua Zhang; Wenbo Zhang; Heng Wu; Tao Huang, "VMFDF: A Virtualization-based Multi-Level Fault Detection Framework for High Availability Computing," e-Business Engineering (ICEBE), 2012 IEEE Ninth International Conference on , vol., no., pp.367,373, 9-11 Sept. 2012

[15] Chuliang Weng; Jianfeng Zhan; Yuan Luo, "TSAC: Enforcing Isolation of Virtual Machines in Clouds," Computers, IEEE Transactions on, vol.64, no.5, pp.1470, 1482, May 1 2015

[16] J. E. Smith and R. Nair, Virtual Machines: Versatile Platforms for Systems and Processes. New York, NY, USA: Elsevier, 2005

[17] Hadoop, http://hadoop.apache.org

[18] T. Nguyen, et al., "CloudAligner: A fast and full-featured MapReduce based tool for sequence mapping," BMC Res Notes, vol. 4, p. 171, 2011.

[19] Schatz M: CloudBurst: highly sensitive read mapping with MapReduce. Bioinformatics 2009, 25(11):1363.

[20] G. Malewicz, M. Austern, A. Bik, J. Dehnert, I. Horn, N. Leiser, and G. Czajkowski, "Pregel: a system for largescale graph processing," in Proceedings of the 2010 international conference on Management of data, ser. SIGMOD '10. New York, NY, USA: ACM, 2010, pp. 135–146.

[21] J.Ekanayake, H.Li, B.Zhang et al., "Twister: A Runtime for iterative MapReduce," in Proceedings of the First International Workshop on MapReduce and its Applications of ACM HPDC 2010 conference June 20-25, 2010, Chicago, Illinois, 2010

[22] Jiang, D.; Tung, A.; Gang Chen, "MAP-JOIN-REDUCE: Toward Scalable and Efficient Data Analysis on Large Clusters," Knowledge and Data Engineering, IEEE Transactions on , vol.23, no.9, pp.1299,1311, Sept. 2011

[23] Dahiphale, D.; Karve, R.; Vasilakos, A.V.; Huan Liu; Zhiwei Yu; Chhajer, A.; Jianmin Wang; Chaokun Wang, "An Advanced MapReduce: Cloud MapReduce, Enhancements and Applications," Network and Service Management, IEEE Transactions on , vol.11, no.1, pp.101,115, March 2014

[24] Feng X, Grossman R, and Stein L: PeakRanger: a cloud-enabled peak caller for ChIP-seq data. BMC Bioinformatics 2011, 12:139.

[25] Saccharomyces genome database (SGD). (2015). [Online] Available: http://www.yeastgenome.org/

[26] Marinescu, D.C., "Parallel and Distributed Computing: Memories of Time Past and a Glimpse at the Future," Parallel and Distributed Computing (ISPDC), 2014 IEEE 13th International Symposium on , vol., no., pp.14,15, 24-27 June 2014

[27] Gartner, Inc. Gartner says worldwide cloud services market to surpass $68 billion in 2010. http://www.gartner.com/it/page.jsp?id=1389313,

[28] P. Mell and T. Grance, The NIST Definition of Cloud Computing, US National Institute of Science and Techonology Std., 2011.[Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

[29] Bera, S. Misra, S, Rodrigues J.J.P.C, "Cloud Computing Applications for Smart Grid: A Survey," Parallel and Distributed Systems, IEEE Transactions on, vol.PP, no.99, pp.1, 1.PrePrints.doi: 10.1109/TPDS.2014.2321378

[30] P. Mehrotra, J. Djomehri, S. Heistand, R. Hood, H. Jin, A. Lazanoff,S. Saini, and R. Biswas, "Performance Evaluation of Amazon EC2 for NASA HPC applications," in Proceedings of the 3rd workshop on Scientific Cloud Computing. New York, NY, USA: ACM, 2012

[31] Chun Hui Suen, "Evaluating and Improving the Performance and Scheduling of HPC Applications in Cloud", IEEE Transactions on Cloud Computing, , no. 1, pp. 1, PrePrints , doi:10.1109/TCC.2014.2339858

[32] Dahiphale, D.; Karve, R.; Vasilakos, A.V.; Huan Liu; Zhiwei Yu; Chhajer, A.; Jianmin Wang; Chaokun Wang, "An Advanced MapReduce: Cloud MapReduce, Enhancements and Applications," Network and Service Management, IEEE Transactions on , vol.11, no.1, pp.101,115, March 2014

[33] G. Malewicz, M. Austern, A. Bik, J. Dehnert, I. Horn, N. Leiser, and G. Czajkowski, "Pregel: a system for largescale graph processing," in Proceedings of the 2010 international conference on Management of data, ser. SIGMOD '10. New York, NY, USA: ACM, 2010, pp. 135–146.

[34] Kajdanowicz, T.; Indyk, W.; Kazienko, P.; Kukul, J., "Comparison of the Efficiency of MapReduce and Bulk Synchronous Parallel Approaches to Large Network Processing," Data Mining Workshops (ICDMW), 2012 IEEE 12th International Conference on , vol., no., pp.218,225, 10-10 Dec. 2012

[35] Hyungro Lee: Using Bioinformatics Applications on the Cloud.

[36] Michael C. Schatz: CloudBurst: highly sensitive read mapping with MapReduce.

[37] Tung Nguyen, Weisong Shi and Douglas Ruden: CloudAligner: A fast and full-featured MapReduce based tool for sequence mapping.

[38] LI Xubin, JIANG Wenrui, JIANG Yi and ZOU Quan: Hadoop Applications in Bioinformatics.

[39] Xiao-liang Yang, Yu-long Liu, Chun-feng Yuan, Yi-hua Huang: Parallelization of BLAST with MapReduce for Long Sequence Alignment

[40] Hdinsight (hadoop on Azure)," https://www.hadooponAzure.com/.

[41] Baheti, V.K., "Windows Azure HDInsight: Where big data meets the cloud," IT in Business, Industry and Government (CSIBIG), 2014 Conference on, vol., no., pp.1,2, 8-9 March 2014

[42] Forman, G. 2003. An extensive empirical study of feature selection metrics for text classification. J. Mach. Learn. Res. 3 (Mar. 2003), 1289-1305.

[43] Brown, L. D., Hua, H., and Gao, C. 2003. A widget framework for augmented interaction in SCAPE.

[44] Y.T. Yu, M.F. Lau, "A comparison of MC/DC, MUMCUT and several other coverage criteria for logical decisions", Journal of Systems and Software, 2005, in press.

[45] Spector, A. Z. 1989. Achieving application requirements. In Distributed Systems, S. Mullender