2012 4th International Conference on Bioinformatics and Biomedical Technology
IPCBEE vol.29 (2012) © (2012) IACSIT Press, Singapore

Cloud-enabling Sequence Alignment with Hadoop MapReduce: A Performance Analysis

Krithika Arumugam [1+], Yu Shyang Tan [1], Bu Sung Lee [1] and Rajaraman Kanagasabai [2]

[1] School of Computer Engineering, Nanyang Technological University, Singapore
[2] Institute for Infocomm Research, Singapore

Abstract. The Bio Sequence alignment involves arranging DNA, RNA or protein sequences in order to find similarity between the aligned sequences, which helps us to find the structural, functional and evolutionary relationships among organisms. Next generation sequencing has led to the generation of billions of sequence data, making it increasingly infeasible for sequence alignment to be performed on standalone machines. Hence algorithms capable of running in multi-node clusters and deployable on cloud have been investigated. The performance improvement offered by these algorithms is not usually predictable since the cloud implementation has many overheads which may sometimes negatively affect the performance. In this paper, we investigate this issue and provide a detailed study of the performance of a canonical MapReduce implementation of sequence alignment algorithm, CloudBurst: Highly sensitive short read mapping with MapReduce. We consider a mapping task involving several million reads to a reference genome, and observe that the cloud-enabled sequence alignment can lead to many-fold speedup and scales up as the number of nodes increases. It was noted that the performance of sequence alignment algorithms depended upon the configuration of the cluster in which it is executed. Running the algorithm with a large input in a cloud computing environment was more efficient than running in a single node.

Keywords: Sequence Alignment, Read Mapping, Cloud computing, Hadoop, MapReduce.

1. Introduction

Next generation sequencing machines are generating billions of short sequences of DNA called reads which are mapped against a reference genome to find SNP's, conserved regions, construct phylogenetic trees to find evolutionary significance and for use in various other biological analyses [2]. Gaps or insertions are made during the alignment of sequences to produce a better alignment. Several algorithms and tools have been developed to solve this problem. BLAST [15] is one of the commonly used algorithm for sequence alignment. BLAST algorithm which uses the seed-and-extend technique is designed to run in a single node, hence it is difficult to manage the huge data generated by the next generation sequencing machines [19].

Recently a new style of computing, called Cloud computing [12], that provides opportunities to implement large scale data-intensive computing infrastructures [9] in cost-effective manner, has emerged. Rather than investing in large expensive infrastructure setup, the cloud paradigm enables elastic provisioning of computing resources in an on-demand basis. Cloud computing based approach is promising to address the large-scale biological data analytics challenges. Driven by the large scale nature of bio resources [8], the cloud paradigm is being increasingly adopted in Bioinformatics to propose highly scalable algorithms [14]. Some examples include: Crossbow, [7] for whole genome resequencing analysis and SNP genotyping, Myrna, [6] for calculating differential gene expression from large RNA-seq data sets, Contrail, [11] for de novo assembly from short sequencing reads.

+ Corresponding author. Tel.: + 65-82466895;
E-mail address: krit0002@e.ntu.edu.sg.

Cloudburst [10] is a famous algorithm for mapping next-generation short read sequencing data to a reference genome for SNP discovery and genotyping. It is capable of performing sequence alignment in multi-node clusters and is deployable on cloud. However, the performance improvement offered is not usually predictable since the cloud implementation has many overheads which may sometimes negatively affect the performance. In this paper, we investigate this issue and provide a detailed performance analysis. We consider a mapping task involving several million reads to a reference genome, and observe that the cloud-ready sequence alignment can lead to many-fold speedup and scales up as the number of nodes increases.

2. Method

2.1. Blast

Basic Local Alignment Search Tool (BLAST) [15] was developed to fill the gaps which the other algorithms lacked, in finding the alignment scores based on mutations, with the constraint of reducing the time for alignment. It heuristically calculates the local similarity scores between sequences, with distinct alignments in the conserved regions. First the similar residues are found and a score is calculated based on the identities and replacements in the segments. The sequence segment can have any length such that the similarity score of the segment is the sum of the score of aligned residues found [15]. From such scoring method we can find Maximum Segment Pairs which has the highest score and the segment is said maximal when the score does not change any further either by reducing or by extending the length of the pair [16]. The time consumed for such a scoring method is the product of the length of the sequences under consideration [15].

The scores are evaluated against a cut-off score from which a sequence is identified as a significant segment or an insignificant segment. Although the segments which are in the transition region very near to the cut-off value reveals some interesting relationship in the biological context. The main aim of BLAST is to reduce the time consumption, for which an even smaller threshold value is considered for a certain fixed word length. The sequence pairs are scanned for any subsequence of length equal to the word length with at least a score equalling the threshold. The found segment is extended to see if a cut-off value is achieved. The value of the threshold plays an important factor in the execution time as it increases the time if the threshold value is smaller, but tends to give better probability of segments with score of at least the cut-off value.

2.2. CloudBurst

A CloudBurst [10] is a parallel read mapping algorithm implemented in Hadoop MapReduce [1]. It uses the seed and extend algorithm to map the reads against the reference genome.

MapReduce [1] is a popular cloud computing programming model or a framework for distributed computing used especially where large data sets are involved and analysed. In this model, a *map* function and a *reduce* function is specified by the user [5]. Hadoop is an open source project, implementing MapReduce, supporting distributed computing and also providing scalability [4]. Hadoop in short provides distributed storage and analysis system [17]. Storage is provided by the Hadoop Distributed File System (HDFS) [3] and the analysis system is the MapReduce programming model or the computational model.

CloudBurst functions as follows. The input sequence files are converted from ASCII format to binary format and are fed to the sequence alignment program. In the mapping phase it generates key-value pairs of seeds which are substrings of the reads as well as the reference genome. In the shuffling phase the seeds are grouped based on the keys, which collects together similar keys between the reads and the reference sequence. In the reduce phase an exact alignment with the allowed mismatches is found by extending the similar keys, performing an end-to-end alignment, via these processes the seed and extend technique is implemented.

2.2.1 Mapping Phase

The reference sequence is fed to the mapper which generates key value pairs. The key is a seed of length 's' from the sequence and the value is a tuple of MerInfo. A Mer is a fixed length substring of the larger sequence. It emits pairs as shown in Table 1. Seed is the fixed length substring of the sequence,

referred to as k-mer and the MerInfo gives the identification number of the seed, offset position in the reference sequence. The isRef property identifies the key as the reference sequence; else it is considered as a read sequence. The isRC property identifies the key as a normal sequence or a reverse compliment of all non overlapping seeds in the reference sequence under consideration. The left_flank and right_flank are the flanking sequences of length (m – s + k) bp, where 'm' is the minimum read length. The minimum read length is given as output information during the conversion of the read sequence. The value 's' is the seed size and 'k' is the user-defined number of mismatches allowed.

Table 1: The output format of the map phase.

Output of Mapper: (Key, value)
Key : seed
Value: MerInfo [id, position, isRef, isRC, left_flank, right_flank]

When an allowed mismatch is defined by 'k', a read sequence of length 'm' bp is divided in to $\left(\frac{m}{k+1}\right)$ smaller reads, and is assumed that $\left(\frac{m}{k+1}\right)$ at least one part should match exactly for having at the most 'k' number of mismatches. In the case of the read sequence, the mapper emits key value pairs with non overlapping k-mers with its position as value, with the isRef property set to '0'.

The low complexity seeds (containing only a single base character) take much longer time to get aligned as compared to the normal seeds [10]. CloudBurst handles this time inefficiency by making redundant copies of those seeds. The reads if found to match with any of the low complexity seed, is assigned to any one of the corresponding redundant seed. By parallelizing this across the reducers the time taken to align the low complexity seeds is reduced and thereby the load balance across the nodes improves [10].

2.2.2 Shuffle Phase

In the shuffle phase the default property of the Hadoop framework's sorting facility is used in the algorithm. The shuffle phase groups the key value pairs generated by all the mappers in to a single compiled list with similar key values. The values of the corresponding keys are grouped together either if it's a read-key or a seed-key (reference sequence). This creates a hash table of matching seeds and reads, and its position.

2.2.3 Reduce Phase

In the reduce phase the exact matching seeds are extended to find an inexact alignment within the allowed mismatches. From the output of the mapper the reducer segregates the seed and read keys in to two sets with respect to the MerInfo. After segregation, the two sets undergo a Cartesian product function; in the intersection space the flanking bases are compared [10]. If a matching sequence with k mismatches is found it is checked to determine if it is a duplicate seed, as the same alignment could have multiple seeds with exact alignments. The output of the reduce phase contains the reads with all its alignments limited to a maximum mismatch of 'k' bases.

2.2.4 Alignment Filtration

After the MapRed job for aligning the reference sequence and the reads are done, the output alignments are filtered with another MapRed job, so that an unambiguous best alignment for each read can be given as output. The map phase of the filtering job emits reads as keys and alignment information as the value. The shuffle phase of the job groups alignments for similar reads, after which only the top two alignments are emitted out of the reducer as output.

3. Results

We evaluate how sequence alignment algorithms perform in a cloud by executing CloudBurst algorithm in a variety of configurations. Preliminary testing was done in a cluster of 4 nodes with single core processor, of which 3 nodes were used as the tasktrackers where the actual computations in Hadoop take place. We used reference genome of 1.2 billion bp and 3000 probes from the NCBI [13]. The reads were 32 bp long. The input was chosen so as to utilise the cluster's full capacity. The nodes contained 90GB of hard disk

space and 2GB of physical memory and 6GB of virtual memory. Apache Hadoop-0.20.2 was setup with 2 map tasks per node. The data was uploaded to the HDFS before the jobs were started.

The web user interface of HADOOP framework contains the data related to the job that ran in the framework. It contains the time taken for the different sub states of the MapRed job. The raw data of time consumed can be visualised as the graph shown in figure 1. Figure 1 shows the pattern of execution of the MapRed job of the CloudBurst job, which contains the CloudBurst job, itself and the filter alignment job. The CloudBurst job aligns the sequences and the filter alignment job filters the aligned sequences according to the output requirement. The shuffle tasks starts with a smaller offset from the start of the map task. Since the map task emits only the seeds, which reduces the map load, it reaches the threshold fairly earlier. But in the case of the reduce task the time taken is high, because the reduce task extends the hits and aligns a longer match.

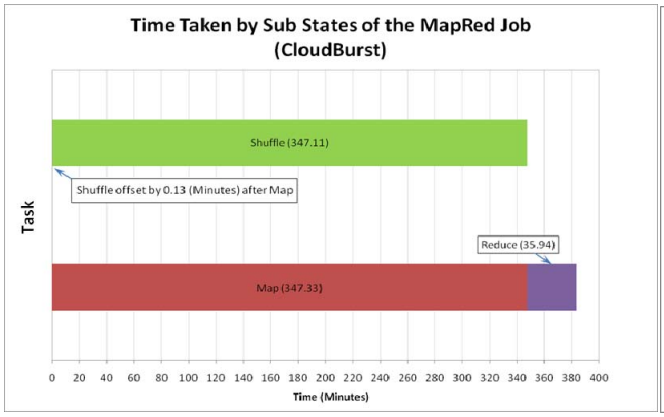

Fig. 1: Time consumed for CloudBurst algorithm's sub FilterAlignment job in CloudBurst algorithm.

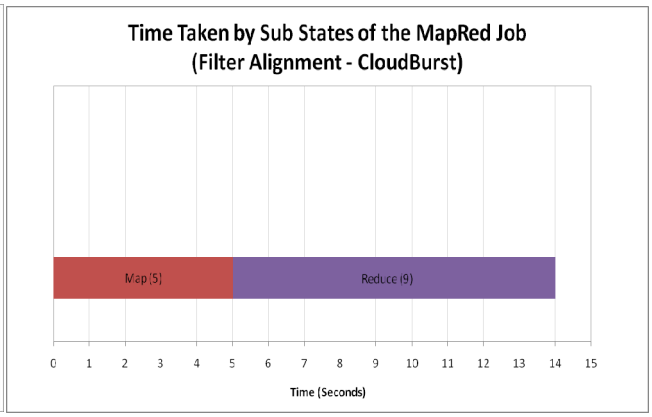

Fig. 2: Time consumed for the sub states of the states of MapRed job.

Figure 2 shows the time for the filter alignment job of the CloudBurst algorithm. It filters the aligned sequences, hence the time taken is very less when compared to the CloudBurst. The shuffle phase is absent in this job since the overall load to the MapRed framework is less in this filter alignment job.

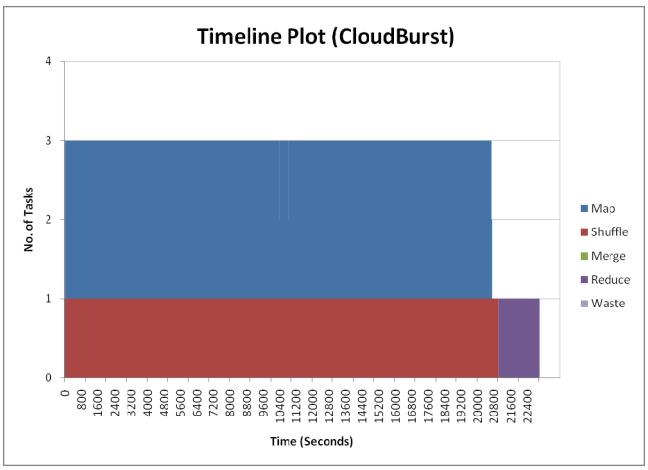

Fig 3: Timeline plot for the CloudBurst job

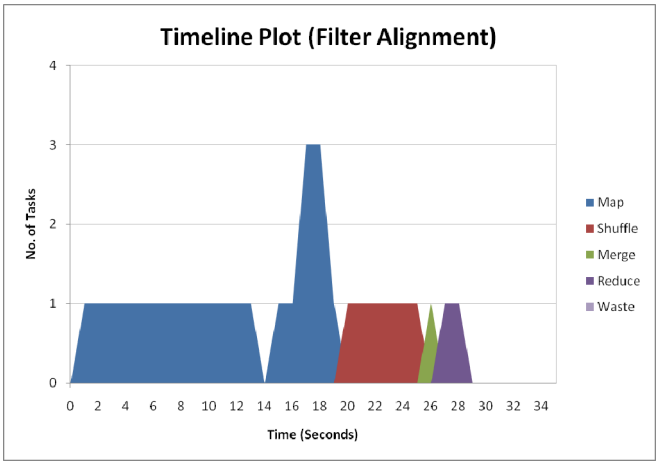

Fig. 4: Timeline plot for the Filter Alignment.

Timeline analysis plot was introduced by Yahoo [18] to visualize the performance of the Hadoop job. The plot shows all the phases of the Hadoop job, and thus gives a complete overview of job. The 'x' coordinate represents the time elapsed or the heart beat time. The 'y' coordinate shows the total number of tasks that are running at that instant of time. The tasks are split into map and reduce, further the reduce task is split in to shuffle, merge and finally reduce. The shuffle phase copies the output from different mapper's to the reducer, and the merge, merges intermittently the output collected by the shuffle phase and sends the

merged data to the reducer. The reducer finally writes it as output. Thus at any point of time many of the mentioned phases run in the framework. The need to track these running tasks arises with the increase in the number of such tasks, which is dependent on the job complexity. The Timeline plot besides showing the number of tasks running at every instant also shows which of those tasks are running. This enables the user to visualize the phase in which the lag is present and other important information like the start-up time required to launch all the tasks in that phase.

The figure 3 shows the time line plot for the CloudBurst job which depicts that most of the time the map tasks are running without any interruption. The loads are balanced across the tasks. The shuffle starts immediately. The filter alignment job has a timeline plot as shown in figure 4. The map tasks are distinctly depicted to have started one after the other and at about 17 seconds later the full cluster capacity is reached. The shuffle tasks starts nearly at the end of the map task. The merge sub state is visible here and has lasted for nearly for 2 seconds.

Next, CloudBurst was executed in a bigger cluster. We used reference genome and reads of 2.6 billion bp and 20 million bp respectively from the NCBI [13]. The reads were 36 bp long. The cluster setup had 12 nodes of which 10 were used as the tasktrackers where the actual computations in Hadoop take place. The nodes were running on Ubuntu version 10.10. Each node is a dual core with 2800MHz 'AMD Athlon(tm) II X2 240 Processor' with 255GB of hard disk space in each node. Cloudera hadoop-0.20.2-cdh3u0 was setup and 2 map tasks per node was used. The data was uploaded to the HDFS before the jobs were started.

The first test in the bigger cluster examined how the CloudBurst algorithm scales with higher speed up as the number of nodes increased. The reads were aligned against the reference genome allowing one mismatch. The results show that the speedup of CloudBurst algorithm increases as the number of nodes increase. Figure 5 shows that the time taken to complete the task reduces as the number of nodes increases. Time taken to convert and upload the data to HDFS is not included.

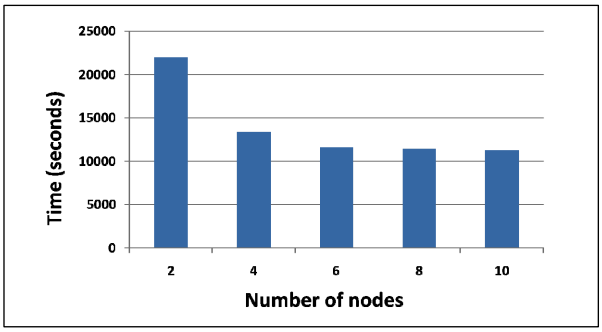

Fig. 5: CloudBurst Speedup Vs Number of nodes

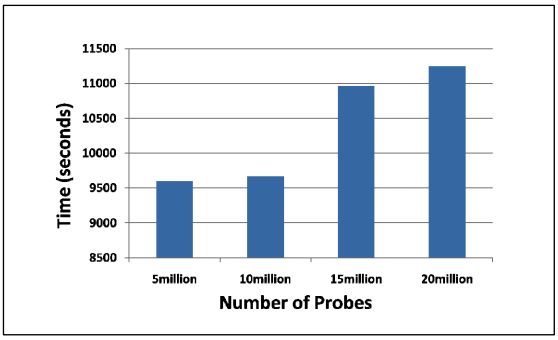

Fig. 6: CloudBurst Speedup Vs Number of Probes

Next, NCBI BLAST (version 2.2.25) was run on a single node with a quad core processor with the same input. The BLAST was continuing to run for more than a day when compared to CloudBurst executed in a multinode cluster where the sequence aligment was completed in a few hours.

The final test explored the time taken for the CloudBurst algorithm to complete by varying number of probe sequences to be aligned against the reference genome. Figure 6 shows the result of the test. Figure 6 shows that as the number of probe sequences to be aligned against the reference genome increases, time taken by the CloudBurst algorithm to complete the job also increases.

The performance of the sequencing algorithms on a cluster depends on many variables such as the number of nodes, processing capability of the cluster, memory availability in the nodes, the disk space availability, the sequencing programs' architecture to generate lesser intermediate results and the ability to handle large data set. The configuration of the cluster and the efficient usage of the available resources directly affect the performance of the sequencing algorithm.

The CloudBurst algorithm hashes the whole genome sequence in the mapping phase of the job and hence needs more number of cores and space for efficient hashing with respect to time. CloudBurst algorithm gives overlapping alignments for a fixed length of the probe sequence. The time and resource utilised by the

CloudBurst algorithm increases with increase in the number of probes and decreases with the increase in number of nodes.

4. Discussion

Next generation sequencing has led to the generation of billions of sequence data, making it increasingly infeasible for sequence alignment to be performed on standalone machines. We considered the cloud computing approach for sequence alignment that promises the ability of running in multi-node clusters and providing elastic scalability. We investigated the performance of a canonical cloud-ready algorithm, and provided a detailed study of the performance of a canonical MapReduce-enabled sequence alignment algorithm on a mapping task involving several million reads to a reference genome. It was noted that the performance of sequence alignment algorithms depended upon the configuration of the cluster in which it is executed. Running the algorithm with a large input in a cloud was more efficient than running in a single node. CloudBurst algorithm executed in a cloud seems to be faster than the NCBI BLAST which was run in a single node. The Hadoop framework plays an important role since it is capable of handling large data sets and also the general aspects of distributed computing is automatically handled by hadoop. Thus, we conclude that Hadoop's ability to run sequence alignment algorithm in a cloud with massively large data will help the researchers in ground breaking discoveries by sequence analysis, which would not have been possible by using a single node.

5. References

[1] Apache Software Foundation, Hadoop, 2007, http://hadoop.apache.org/ .

[2] Cole Trapnell, Steven L Salzberg, How to map billions of short reads onto genomes, *Nature Biotechnology*, Volume 27, Number 5, May 2009.

[3] Dhruba Borthakur, "The Hadoop Distributed File System: Architecture and Design", Apache Software Foundation, 2007, Retrieved from http://hadoop.apache.org/hdfs/docs/current/hdfs_design.html .

[4] Jason Venner, Pro Hadoop, Springer, 2009.

[5] J.Dean and S.Ghemawat, MapReduce: Simplified data processing on large clusters, in USENIX Symposium on *Operating Systems Design and Implementation*, San Francisco, CA, Dec 2004, pp.137-150.

[6] Langmead B, Hansen KD, Leek JT, Cloud-scale RNA-sequencing differential expression analysis with Myrna, *Genome Biol* 2010, 11:R83.

[7] Langmead B, Schatz MC, Lin J, Pop M, Salzberg SL, Searching for SNPs with cloud computing, *Genome Biol* 2009, 10:R134.

[8] Massimo Gaggero, Simone Leo, Simone Manca, Federico Santoni, Omar Schiaratura, Gianluigi Zanetti, Parallelizing bioinformatics applications with MapReduce, CSR4, Edificio 1, Sardegna Ricerche, Pula, Italy.

[9] Michael D. Kane and John A. Springer, Integrating Bioinformatics, Distributed Data Management, and Distributed Computing for Applied Training in High Performance Computing. In Proceedings of the 8th *ACM SIGITE conference on Information technology education (SIGITE '07). ACM*, New York, NY, USA, 33-36.

[10] M.C. Schatz: CloudBurst: Highly sensitive read mapping with MapReduce, *Bioinformatics* 2009, 25:1363-1369.

[11] M.C. Schatz, Dan Sommer, David Kelley, and Mihai Pop, Contrail: Assembly of Large Genomes using Cloud Computing, Retrieved from http://sourceforge.net/apps/mediawiki/contrail-bio/index.php?title=Contrail .

[12] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia, A view of Cloud Computing, *Commun. ACM* 53, 4 April 2010, 50-58. DOI: 10.1145/1722654.1722672 .

[13] NCBI, http://www.ncbi.nlm.nih.gov/ .

[14] Ronald C Taylor, An overview of the Hadoop/MapReduce/HBase framework and its current applications in bioinformatics, *BMC Bioinformatics,* 2010, 11(Suppl 12):S1.

[15] S.F.Altschul et al, Basic Local Alignment Search Tool, *Journal of Molecular Biology*, 1990, vol.225, pp.403-410.

[16] Tom Madden, The NCBI Handbook - The Blast Sequence Analysis Tool, 2002, Retrieved from http://www.ncbi.nlm.nih.gov/bookshelf/br.fcgi?book=handbook&part=ch16.

[17] Tom White, Hadoop – The definitive Guide, 2009, O'Reilly Yahoo Press.

[18] YAHOO! Developer Network: Hadoop Sorts a Petabyte in 16.25 Hours and a Terabyte in 62 Seconds", 2009 Retrieved from http://developer.yahoo.com/blogs/hadoop/posts/2009/05/hadoop_sorts_a_petabyte_in_162/.

[19] Yutao Qi and Feng Lin, Parallelisation of the Blast Algorithm, *Cellular & Molecular Biology Letters*, Volume 10, pp. 281 – 285, 2005.