Exploiting High Performance on Bioinformatics Applications in a Cloud System

Taylor N. Job and Jin H. Park

Abstract—**Bioinformatics is the field of manipulating ever increasing biological data and thus, needs high performance processing. As a convenient high performance system, cloud computing systems are more and more popularly used in this field. We develop and examine the models of high performance bioinformatics computations on a cloud system. In our practice, sequence alignment algorithms, Smith-Waterman algorithm and CloudBurst algorithm, are used and combinations of currently used technologies are evaluated for performances on the Amazon Elastic Compute Cloud (EC2) system. Among the various computation models, our focus is on the models with SHadoop and Fair Scheduler, as we are looking for new performance improvements from them. In our practice, using SHadoop for both Smith-Waterman and CloudBurst algorithms showed the best performance, i.e., speedup of 1.86x and 1.19x with Smith-Waterman and CloudBurst, respectively, over baseline models with Hadoop.**

Index Terms—**bioinformatics, cloud computing, Hadoop, SHadoop, sequence alignment**

I. Introduction

THE concept of Big Data and it's use is now everywhere in today's world. From business to academia, various organizations are now utilizing current technologies to access and manipulate information in the sea of ever-changing data. Bioinformatics is a closely related field of manipulating Big Data, which encompasses a wide variety of biological data such as DNA and RNA data from a wide variety of species. Most of the biological data are available in the form of very large text files, e.g., FASTA and FASTQ, and could take hours to days to process depending on the application and the scope of the operation.

Some significant ideas about analyzing large amounts of raw data come from Google's MapReduce [1] concept, which splits up the job and executes the parts on a cluster of computing nodes in a parallel manner. Apache Hadoop [2], the open source version of MapReduce, is a popular software tool used to implement this concept and is what we use in our practice described in this paper. Besides, cloud computing environment is a convenient way of accessing service, data and storage through internet, and more and more widely used in bioinformatics computing to achieve high performance. In the cloud system environment, users are able to scale the number of nodes needed for the job using a tool like Hadoop. Amazon's Elastic Compute Cloud (EC2) [3] is one of the most widely used commercial cloud computing services and has a good user interface which makes it easier to setup, strip down, or add more computing nodes to the cluster. In our practice, we use EC2 for the computing cluster.

While the usage of Hadoop on a cloud computing system does speed up the processing of bioinformatics operations, there have appeared some approaches of improving performance. Since Hadoop's default scheduler is based on a FIFO mechanism, the resulting throughput is low when many jobs are submitted. Apache Fair Scheduler [4] is a pluggable MapReduce scheduler and helps multiple users or multiple job submissions by supporting the mechanism of executing shorter jobs without delay and thus, increases the system throughput. SHadoop [5] is a fully compatible, optimized version of Hadoop, and its primary goal is reducing the execution time of MapReduce jobs, especially short jobs. This is done by optimizing the setup and cleanup operations of a job and by introducing an instant message system between the job-tracker and the task-trackers that speeds up the communication of scheduling information between them. Reducing the execution time of MapReduce is definitely beneficial when running an exhaustive bioinformatics sequence alignment application. CloudBurst [6] is a parallel read-mapping algorithm used to map DNA-sequence data to the reference genome. It is implemented on a Hadoop based cluster and aims to optimize the parallel execution. CloudBurst's creators claim that as the number of processors increases on a user's cluster, the speedup is near linear [6].

In this paper, we build hybrid computation models from the aforementioned technologies to exploit high performance on bioinformatics computing and our practice is limited to sequence alignment algorithm, Smith-Waterman [7] and CloudBurst algorithms. In our practice, all experimentations are implemented with a Hadoop based cluster on Amazon EC2 cloud service. In particular, our experimentations include running the Smith-Waterman sequence alignment algorithm written in Python in MapReduce format by itself, with the Fair Scheduler added, with SHadoop added, and with a combination of the Fair Scheduler and SHadoop added to the cluster. We also run the CloudBurst algorithm by itself, with the Fair Scheduler, and with SHadoop. Our main focus is on the combination of SHadoop with Fair Scheduler and the combination of SHadoop with CloudBurst as we are looking for new performance improvements from them.

The rest of this paper is organized as follows. In Section II, a brief review of the related work is presented. In Section III, the concept and implementation of the proposed computation models are described. In Section IV, Experimental results and performance analysis are presented, and Section V concludes the paper.

Manuscript received July 12, 2014; revised Aug. 08, 2014.

Taylor N. Job is with the Computer Science Department, California State University, Fresno, CA 93740 USA (e-mail: taylorjob@mail.fresnostate.edu).

Jin H. Park is with the Computer Science Department, California State University, Fresno, CA 93740 USA (e-mail: jpark@csufresno.edu).

II. Related Work

In this section, we briefly review some recent approaches on building and running bioinformatics applications on cloud computing systems.

As the field of bioinformatics expands, so has the number of different ideas and approaches that use current data-driven technologies, such as a variety of cloud services as well as parallel frameworks and applications, to aid research and development. Research work in [8] examines performance from using Hadoop on a cloud computing system. The authors describe how to obtain an increase in performance by utilizing Hadoop on a cloud computing service. They explore different alignment tools and applications that perform sequence alignment including CloudBurst. A similar work is described in [9] in which the idea of using a parallel platform for executing the bioinformatics tool, dotplot, in a cloud environment is presented. In this work, Microsoft's Azure software is used to parallelize the execution of the tasks, instead of using Hadoop. In the research described in [10], Google App Engine computing platform is used as the computational resource. The authors introduce the method of building the computer generated protein models used in the protein structure prediction. The proposed protein model comparator is their solution to the problem of large-scale model comparison and can be scaled for different data sizes.

III. Proposed Scheme

A. Components of Computation Models

Amazon Elastic Compute Cloud (EC2) and Simple Storage Service (S3)

Our series of computational experiments are implemented on a 6-node cluster (5 DataNodes and 1 NameNode) provided by Amazon's EC2 cloud service. We also utilize Amazon's Simple Storage Service (S3) to store and easily access data used in the computation. Figure 1 shows the structure of the system.

SHadoop

SHadoop provides computational efficiency based on a coupe of ideas. The first idea is that it optimizes the given tasks that initialize and complete a MapReduce job, which in turn reduces the time taken to execute it. The second idea is that it uses an instant message communication system to relay all messages from JobTracker to TaskTrackers. This mechanism accelerates the performance-sensitive task scheduling as well as execution. It is reported that SHadoop can reach stable performance improvement of ~25% in average from testing benchmarks including WordCount, Sort, Grep, and Kmeans [5]. The performance improvement with SHadoop on standard exhaustive searching operations gave us the motivation of using it to improve the performance of data intensive computations in bioinformatics.

Fair Scheduler

Apache Fair Scheduler is a pluggable MapReduce scheduler, which creates an interface for multiple job submissions to a single computing cluster to share CPU resources evenly. A single job, which is currently running, is allocated with the entire cluster, but when other jobs are

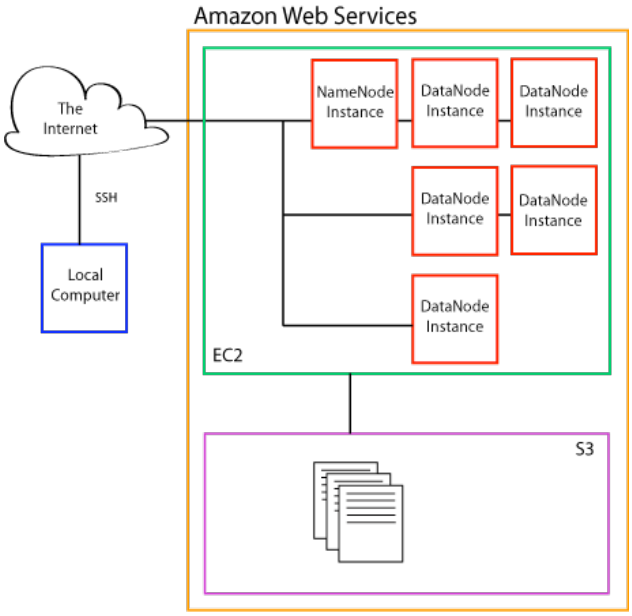


Figure 1. Computing cloud platform

submitted to the same cluster idle task slots are assigned to new jobs. This allows idle tasks to help parallelizing the execution. The Fair Scheduler organizes jobs into pools and allocates available resources equally among them. This optimization in scheduling improves the system throughput since smaller jobs can finish earlier without waiting for the completion of heavy jobs. The default scheduler used in Hadoop is based on the FIFO mechanism and thus, is inefficient comparing to the Fair Scheduler when multiple jobs are submitted to a single cluster. In the Fair Scheduler, there are certain number of parameters, which can be configured to allow even more customization, including preemption of jobs in other pools, limiting the number of jobs in a pool, etc.

CloudBurst

CloudBurst is a seed-and-extend based algorithm, which maps short query sequences (reads) to the reference genome, and uses Hadoop to exploit parallelism. Compared with RMAP, which is an early day's short read mapping tool, it is reported that CloudBurst achieves speedup of up to 30 times

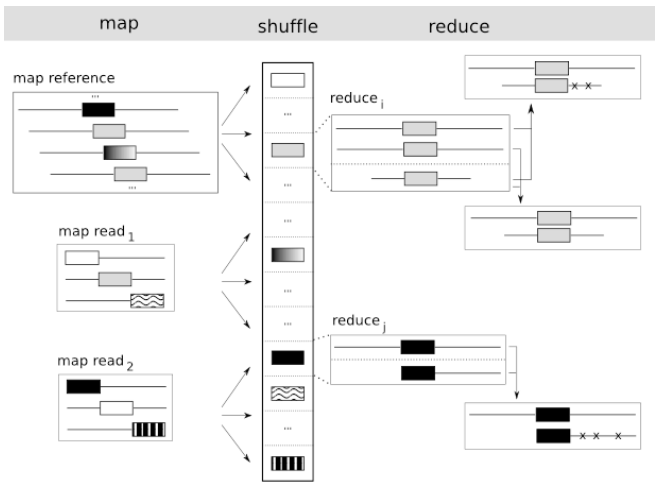


Figure 2. Overview of CloudBurst algorithm

ISBN: 978-988-19253-7-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

faster and it reduces the execution time of a job from hours to minutes in a large cluster with 96 cores [6]. It is also reported that the near linear speedup is achieved as the number of processors increases in the system [6]. Since CloudBurst is available as open-source it can be utilized in a variety of bioinformatics MapReduce applications. Figure 2 shows the operational overview of the CloudBurst algorithm.

B. *Proposed Computation Models*

In our practice, we implemented seven computation models for executing Smith-Waterman (4 models) and CloudBurst (3 models) sequence alignment algorithms on the Amazon EC2 cloud system for performance comparison. Figure 3 shows the computation models, which we built and tested. In the figure, top four models are for Smith-Waterman algorithm and bottom three models are for CloudBurst algorithm. As described earlier, our main focus is on the models with SHadoop and Fair Scheduler. The models with Hadoop (without Fair Scheduler) are implemented for the purpose of comparison, i.e., baseline of the comparison. In the models for Smith-Waterman algorithm, the algorithm is implemented in Python with the format of MapReduce. For the sake of simplicity, we implemented the Smith-Waterman algorithm with the linear gap scoring scheme. We skip describing the detailed implementation of Smith-Waterman algorithm in each model in this paper.

IV. Experimental Results

For the implementation of the computation models that we proposed, we use 6 Large Instances (5 DataNodes (slaves) and 1 NameNode (master)) in the EC2 cloud system. A Large Instance (m3.large) in EC2 has 2 virtual cores and 7.5 GB of storage. Software installation/setup are done successfully with the following packages, which we accessed from Internet sources: Hadoop, SHadoop, Fair Scheduler and CloudBurst. To configure the master and slaves during Hadoop cluster setup we used the following operations, and we skip describing detailed installation/setup steps of the software in this paper.

Configure Master and Slaves:
On NameNode:
$ sudo vi masters
Copy in Public DNS from NameNode
$ sudo vi slaves
Copy in Public DNS from every DataNode
On each DataNode:
$ sudo vi masters
Keep blank
$ sudo vi slaves
Copy in Public DNS from current DataNode

To measure the performance of the computation models (refer Figure 3) we ran 3 jobs, one started after another, with 3 different data sizes, i.e., prokaryotic genomes EColi.fa (5.6MB), Salmonella.fa (4.9MB) and Streptococcus_suis.fa (1.9MB). In our practice, we use the block size of 102,400 bytes for each mapper. To observe the performance differences among different models, we ran the 3 jobs in the order of largest to smallest and ran the jobs from smallest to largest, and computed average of the two trials. Graphs in Figure 4 illustrate the results.

Figure 3. Computation models

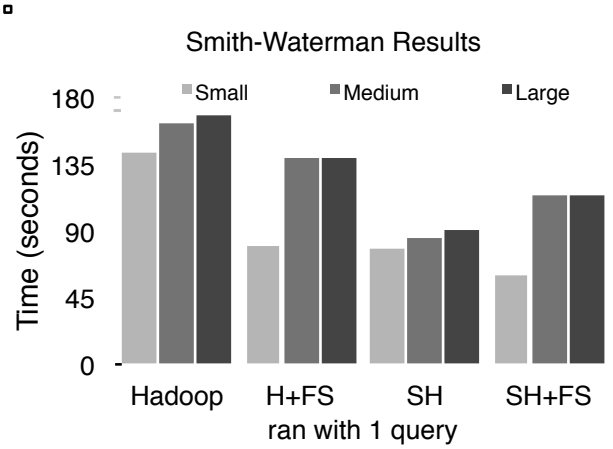
In Figure 4(a), Hadoop represents the baseline model (Hadoop with default FIFO scheduler) and H+FS, SH and SH+FS represent Hadoop with Fair Scheduler, SHadoop, and SHadoop with Fair Scheduler, respectively. As shown in Figure 4(a), for the Smith-Waterman algorithm, using Fair Scheduler with Hadoop achieves reasonable performance gain over simply using Hadoop, but it does not guarantee the gain with SHadoop. In fact, using SHadoop itself shows the best performance overall.

Figure 4(b) shows performances with CloudBurst algorithm. In the figure, CB_H represents CloudBurst with Hadoop (baseline model) and CB_H+FS and CB_SH represent CloudBurst (with Hadoop) with Fair Scheduler and CloudBurst with SHadoop, respectively. The best

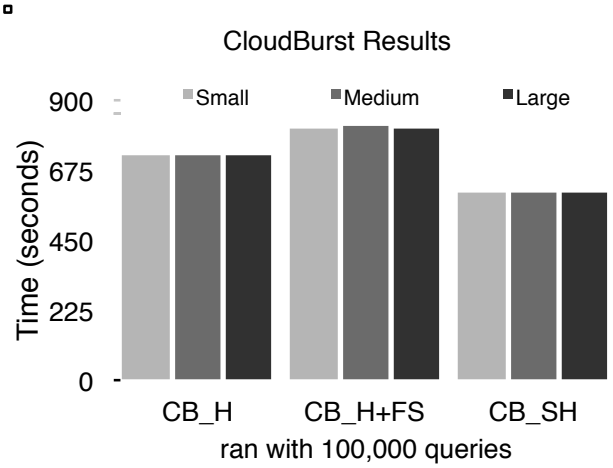
ISBN: 978-988-19253-7-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

performance gain is observed with the model of CloudBurst with SHadoop (refer Figure 4(b)). In this application, using Fair Scheduler shows the worst performance.

In conclusion, using SHadoop achieves the best performance with both Smith-Waterman (1.86x speedup over baseline model) and CloudBurst (1.19x speedup over baseline model) algorithms in our practice though our practice is limited with relatively small sized reference data.



(a) Smith-Waterman results



(b) CloudBurst results

Figure 4. Performances of computation models

V. Conclusion and Discussion

In this research, we built seven different computation models on a cloud computing system to accelerate bioinformatics applications, and tested for their performances. Although our practice is limited with a couple of sequence alignment algorithms and relatively small reference genomes we observed that using SHadoop shows the best performance with both Smith-Waterman algorithm and CloudBurst algorithm. Different from our initial guess, adding Fair Scheduler to SHadoop did not show the best performance, except the smallest job case. We plan to test more diversified bioinformatics applications with the proposed computation models with larger reference database to yield more comprehensive results.

References

[1] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters", Comm. of the ACM, Vol. 51, No. 1, 2008, pp. 107-113.
[2] Apache Hadoop, http://hadoop.apache.org., accessed on March 2014.
[3] Amazon Elastic Compute Cloud, http://www.amazon.com/ec2/, accessed on March 2014.
[4] Apache Fair Scheduler, http://hadoop.apache.org/docs/r2.3.0/hadoop-yarn/hadoop-yarn-site/FairScheduler.html, accessed on March 2014.
[5] R. Gu, Y. Huang, Y. Sun, et al., "SHadoop: Improving MapReduce performance by optimizing job execution mechanism in Hadoop clusters," Journal of Parallel and Distributed Computing, Vol. 74, No. 3, 2014, pp. 2166-2179.
[6] M. Schatz, "CloudBurst: highly sensitive read mapping with MapReduce," Bioinformatics, Vol. 25, No. 11, 2009, pp. 1363-1369.
[7] T.F. Smith and M.S. Waterman, "Identification of Common Molecular Subsequences," J. of Molecular Biology, Vol. 147, 1981, pp. 195-197.
[8] Z. Quan, J. Wen-rui, L. Xu-bin, and J. Yi, "Hadoop Applications in Bioinformatics," Proceedings of the 7th Open Cirrus Summit, 2012, pp. 48-52.
[9] M. Cano, J. Karlsson, G. Klambauer, et al., "Enabling Large-Scale Bioinformatics Data Analysis with Cloud Computing," Proc. of the 10th IEEE International Symposium on Parallel and Distributed Processing with Application, 2012, pp. 640-645.
[10] P. Widera and N. Krasnogor, "Protein models comparator: scalable bioinformatics computing on the Google App Engine platform." arXiv preprint arXiv:1102.4293, 2011.

ISBN: 978-988-19253-7-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)