Incremental DNA Sequence Analysis in the Cloud

Romeo Kienzler[1], Rémy Bruggmann[2], Anand Ranganathan[3], Nesime Tatbul[1]

[1] Department of Computer Science, ETH Zurich, Switzerland
romeok@student.ethz.ch, tatbul@inf.ethz.ch
[2] Bioinformatics, Department of Biology, University of Bern, Switzerland
remy.bruggmann@biology.unibe.ch
[3] IBM T.J. Watson Research Center, NY, USA
arangana@us.ibm.com

Abstract. In this paper, we propose to demonstrate a "stream-as-you-go" approach that minimizes the data transfer time of data- and compute-intensive scientific applications deployed in the cloud, by making them incrementally processable. We describe a system that implements this approach based on the IBM InfoSphere Streams computing platform deployed over Amazon EC2. The functionality, performance, and usability of the system will be demonstrated through two DNA sequence analysis applications.

1 Introduction

In many areas of science, huge amounts of data is being generated at rates that outrun the ability of researchers to store, transmit, and analyze it. For example, in DNA sequence analysis, complex workflows need to be efficiently executed over digital DNA fragments that are now being generated much faster and cheaper owing to the recently invented Next Generation Sequencing (NGS) methods [17]. For such data- and compute-intensive scientific applications, researchers are increasingly turning to cloud computing as a scalable and cost-effective solution. In this case, raw input data that is generated by special scientific devices (e.g., NGS machines) outside the cloud must first be shipped into the cloud. However, due to limited bandwidth between the client and the cloud, transferring large data sets into the cloud can introduce significant latencies and may even become a bottleneck that hinders the scalability advantage of the cloud.

In our recent work, we have proposed an incremental data access and processing approach for data- and compute-intensive cloud applications that can hide data transfer latencies while maintaining linear scalability [7], [8]. In our approach, data is accessed in a "stream-as-you-go" fashion instead of in whole batches, making a stream-based data management architecture a suitable base for implementation. In this demonstration, we propose to show the functionality, performance, and usability of our approach in action through two practical applications of DNA sequence analysis:

1. **Read alignment:** This application involves a very basic and common process in DNA sequence analysis workflows: aligning digital DNA fragments, called *reads*, against a reference genome. In the demo, we will show how a well-known read aligner package (SHRiMP [13]) as part of a more complex workflow can be transparently replaced with our stream-as-you-go version to incrementally run in the

cloud, both producing early results as well as significantly reducing the total processing time of the whole workflow.

2. **SNP detection:** This application additionally involves detecting SNPs (Single Nucleotide Polymorphisms [3]) as reads are being aligned against a reference genome. Different from the first demo, we will show how a complete workflow can be replaced with our stream-as-you-go version and can be pushed into the cloud through an easy-to-use client interface. In this demo, we will use Bowtie [10] (instead of SHRiMP) as the read alignment package and SOAPsnp [12] as the SNP detection package, making our approach directly comparable to the state of the art performance-wise (i.e., the MapReduce-based approach of Crossbow [9]). Thus, we will also report on our performance improvements.

In the rest of this paper, we describe in more detail, our stream-as-you-go approach and the applications that will be used to demonstrate its key features.

2 The Stream-as-you-go Approach

Our key idea to address the data upload latency of scientific applications deployed in the cloud is to enable useful data processing as soon as the first piece of the data set hits the cloud rather than waiting until the arrival of the whole data set. This way, the data transfer latency can be hidden by overlapping it with data processing time. To realize this idea, we propose to use a stream-based data management platform. Our main motivation to do so is to exploit the incremental and in-memory data processing model of Stream Processing Engines (SPEs) (in our specific implementation, the IBM InfoSphere Streams engine [2]). More specifically, we bring (parts of) existing scientific workflows (algorithms/software) into the cloud in a way that they can work with their input data in an incremental fashion. One generic way of realizing this is to use command line tools provided by most of these software. They commonly read and write to standard Unix pipes, which we can exploit by building custom streaming operators that wrap the relevant Unix processes. Then the SPE essentially acts as the middleware to handle all system-level requirements such as inter-process communication, data partitioning and dissemination, operator distribution, and dynamic scaling.

Figure 1 illustrates our general approach. As seen, the main goal is to provide partitions of the source data to the analysis processes running in parallel on different slave cloud nodes in a streaming fashion. This way, data transfer time can be hidden and early results can be generated. Furthermore, incremental processing of streaming data also allows in-memory processing, eliminating the latency of disk access.

3 The DNA Sequence Analysis Use Case

Determining the order of the nucleotide bases in DNA molecules and analyzing the resulting sequences have become very essential in biological research and applications. With the invention of the NGS methods in 2004 [17], higher amounts of genetic data can be read in much less time and at lower cost [7], which has led to the generation of very large datasets to be efficiently analyzed. The output of NGS machines are random

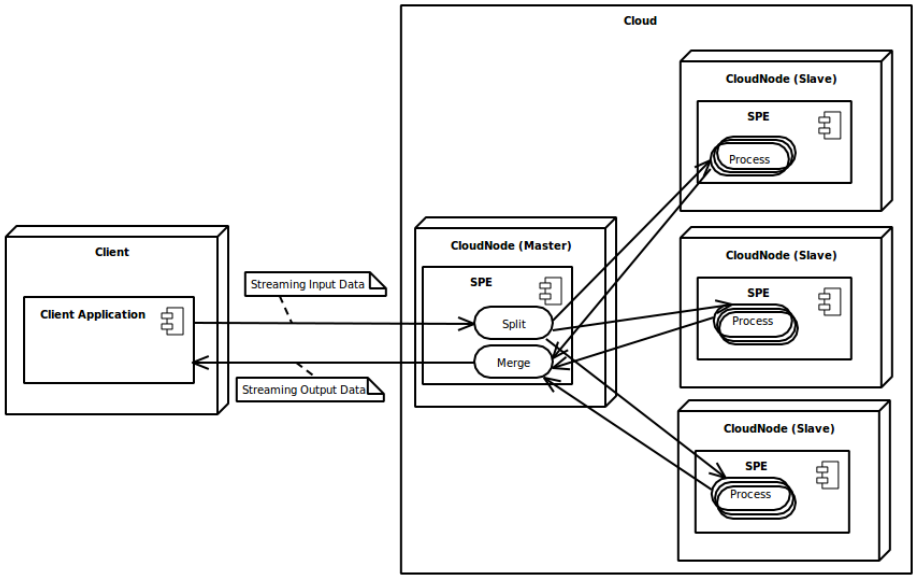

Fig. 1. The stream-as-you-go approach

DNA fragments (reads) of short length. Therefore, they must first be aligned into a complete sequence by mapping them back to a reference genome [11]. The alignment can also highlight the differences against the reference. Such a difference is called a polymorphism. The polymorphism of a single DNA letter is called *Single Nucleotide Polymorphism (SNP)*. SNPs are important to identify, since they are recognized as the main cause of human genetic variability [5]. As such, read alignment and SNP detection are two common, computationally-intensive applications in this domain.

Researchers have recently started using cloud infrastructures for various DNA sequence analysis applications. The current state of the art in massively parallel analysis of large genomic data sets is mainly based on using MapReduce [6] or other similar frameworks [15]. Prominent examples include CloudBurst [14] and CloudAligner [16] for read alignment, and Crossbow [9] for the complete SNP detection process. Despite providing basic scalability, all these solutions suffer from the data transfer latency, since the cloud frameworks that they are based on are primarily designed for batch processing of data stored in a distributed file system in the cloud. In the following, we describe how read alignment and SNP detection can be modeled and implemented using our stream-as-you-go approach, which overcomes this bottleneck.

3.1 Read Alignment a la Stream-as-you-go

Figure 2 shows our stream-as-you-go implementation of the read alignment application. The complete workflow consists of an input data format conversion process, the SHRiMP read aligner process, and an output data format conversion process. Only the SHRiMP part of the workflow is replaced with a stream-as-you-go version deployed

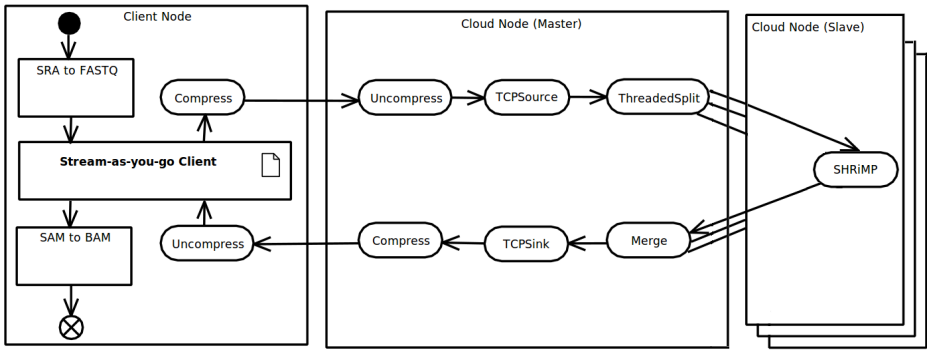

Fig. 2. Stream-as-you-go implementation of read alignment

in the cloud, while the original data conversion processes continue to run at the client node. The client application sends compressed read data to the cloud. After being uncompressed, data gets submitted to a Streams application which first splits it across the available cluster nodes. Split reads are then aligned in parallel using the SHRiMP read aligner package. The output from each SHRiMP instance which are incrementally generated on each processing node in parallel are finally merged, compressed, and sent back to client, where they are uncompressed before the final format conversion. Further details about this implementation can be found in our earlier publication [7].

3.2 SNP Detection a la Stream-as-you-go

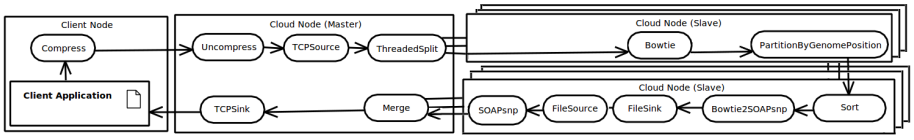

Fig. 3. Stream-as-you-go implementation of SNP detection

Figure 3 shows our stream-as-you-go implementation of the SNP detection application. The client application sends compressed read data to the cloud. After being uncompressed, data gets submitted to a Streams application which first splits it across the available cluster nodes. Split reads are then aligned in parallel using the Bowtie read aligner package. The output from each Bowtie instance is further partitioned by genome position to be then sorted using a distributed in-memory insertion sort algorithm. After some data conversion steps, the sorted data is fed into the SOAPsnp SNP detection package for SNP calling. Results which are incrementally generated on each processing node in parallel are finally merged and sent back to the client over a TCP connection. Further details about this implementation can be found in our earlier publication [8], where we also show almost an order of magnitude reduction in total processing time compared to the state of the art (MapReduce-based Crossbow [9]).

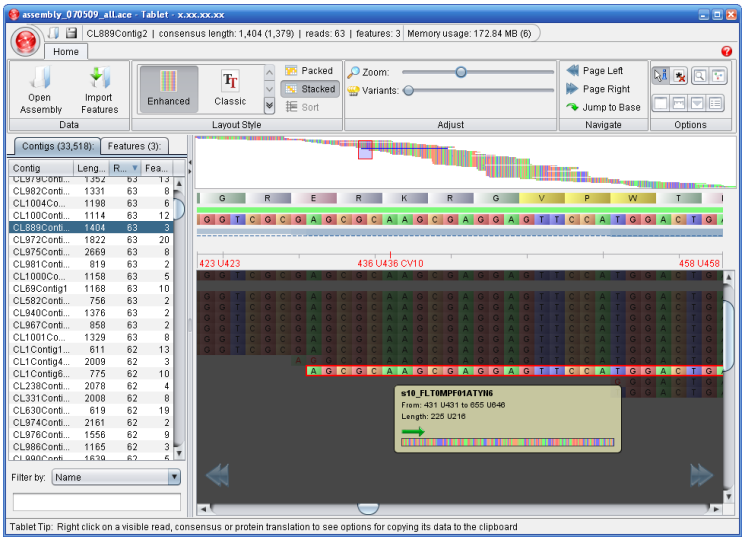

Fig. 4. Tablet-based visualization of aligned reads in comparison to the reference genome

4 Demonstration Details

In order to demonstrate the key features of our stream-as-you-go approach, we will use the two application scenarios whose implementations are described in the previous section.

In the read alignment demo, a partial workflow (i.e., the SHRiMP part) will be converted into an incremental version and will be transparently pushed to the cloud. We will contrast the speed of a local run of the whole workflow against its cloud-enabled counterpart. We will show that, besides a significant speedup, nothing else changes. The process as well as the results stay the same.

For illustration purposes, we will sniff the traffic between the client and the cloud. On the outgoing link, we will see the raw read data, whereas on the incoming link, already processed results will be seen. Data sets will be visualized and explained using the Tablet assembly viewer software [4]. For example, Figure 4 displays the results of an experiment, in which 30000 reads of Streptococcus suis (an important pathogen of pigs) have been aligned against its reference genome. This read data set is taken from the CloudBurst project [14].

In the SNP detection demo, a complete workflow will be converted into a Streams-based incremental version to be deployed in the cloud. We will show an easy-to-use graphical user interface, which allows researchers to run a complete SNP calling process on cloud resources without worrying about the details (Figure 5). The interface allows to select the source and the target data files (for reads and the reference genome) as well as the predefined data analysis process to be used. They can also configure their cloud cluster by selecting the number of nodes to be used based on a corresponding time and price estimation. Once the analysis completes, we will display the detected SNPs. We are planning to use the “E. Coli Small Example” dataset provided at the

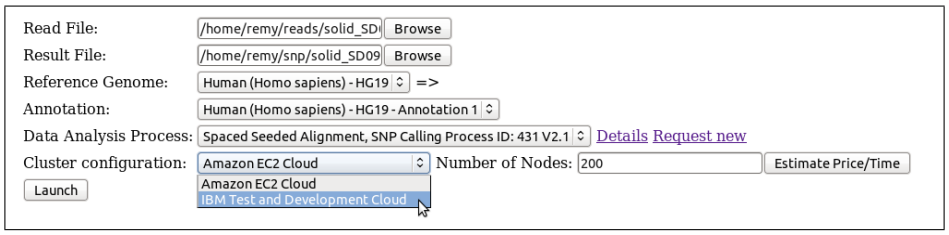

Fig. 5. Easy-to-use cloud deployment interface

Crossbow website [1]. The read file in this dataset is taken from an E. Coli experiment and contains 8922730 reads with a total size of 1.4 GB. The process aligns these reads against the E. Coli reference genome (NC_008253.1) containing 5594158 base pairs with a total size of 5.4 MB.

Acknowledgements. This work has been supported in part by an IBM faculty award.

References

1. Crossbow, http://bowtie-bio.sourceforge.net/crossbow/
2. IBM InfoSphere Streams, http://www.ibm.com/software/data/infosphere/streams/
3. SNP, http://en.wikipedia.org/wiki/Single-nucleotide_polymorphism
4. Tablet Assembly Viewer, http://bioinf.scri.ac.uk/tablet
5. Collins, F.S., Guyer, M., Chakravarti, A.: Variations on a Theme: Cataloging Human DNA Sequence Variation. Science 278(5343) (1997)
6. Dean, J., Ghemawat, S.: MapReduce: Simplified Data Processing on Large Clusters. In: OSDI Conference (2004)
7. Kienzler, R., Bruggmann, R., Ranganathan, A., Tatbul, N.: Large-scale DNA Sequence Analysis in the Cloud: A Stream-based Approach. In: Euro-Par VHPC Workshop (2011)
8. Kienzler, R., Bruggmann, R., Ranganathan, A., Tatbul, N.: Stream As You Go: The Case for Incremental Data Access and Processing in the Cloud. In: ICDE DMC Workshop (2012)
9. Langmead, B., Schatz, M.C., Lin, J., Pop, M., Salzberg, S.L.: Searching for SNPs with Cloud Computing. Genome Biology 10(11) (2009)
10. Langmead, B., Trapnell, C., Pop, M., Salzberg, S.: Ultrafast and Memory-efficient Alignment of Short DNA Sequences to the Human Genome. Genome Biology 10(3) (2009)
11. Li, H., Homer, N.: A Survey of Sequence Alignment Algorithms for Next Generation Sequencing. Briefings in Bioinformatics 11(5) (2010)
12. Li, R., Li, Y., Fang, X., Yang, H., Wang, J., Kristiansen, K., Wang, J.: SNP Detection for Massively Parallel Whole-Genome Resequencing. Genome Research 19(6) (2009)
13. Rumble, S.M., Lacroute, P., Dalca, A.V., Fiume, M., Sidow, A., Brudno, M.: SHRiMP: Accurate Mapping of Short Color-space Reads. PLoS Computational Biology 5(5) (2009)
14. Schatz, M.C.: CloudBurst: Highly Sensitive Read Mapping with MapReduce. Bioinformatics 25(11) (2009)
15. Taylor, R.: An Overview of the Hadoop/MapReduce/HBase Framework and its Current Applications in Bioinformatics. BMC Bioinformatics 11(Suppl 12) (2010)
16. Tung, N., Weisong, S., Douglas, R.: CloudAligner: A Fast and Full-featured MapReduce-based Tool for Sequence Mapping. BMC Research Notes 4 (2011)
17. Voelkerding, K.V., Dames, S.A., Durtschi, J.D.: Next Generation Sequencing: From Basic Research to Diagnostics. Clinical Chemistry 55(4) (2009)