https://doi.org/10.5392/IJoC.2018.14.1.034

Comparison of Distributed and Parallel NGS Data Analysis Methods based on Cloud Computing

Hyungil Kang
Dept. of Semiconductor Electronics Engineering
Chungbuk Health & Science University, Cheongju, Chungbuk, 28150, Republic of Korea

Sangsoo Kim
Dept. of Course-based Qualification Exam Team2
Human Resources Development Service of Korea, Ulsan, 44538, Republic of Korea

ABSTRACT

With the rapid growth of genomic data, new requirements have emerged that are difficult to handle with big data storage and analysis techniques. Regardless of the size of an organization performing genomic data analysis, it is becoming increasingly difficult for an institution to build a computing environment for storing and analyzing genomic data. Recently, cloud computing has emerged as a computing environment that meets these new requirements. In this paper, we analyze and compare existing distributed and parallel NGS (Next Generation Sequencing) analysis based on cloud computing environment for future research.

***Key words**: DNA, Analysis, NGS, Cloud.*

1. INTRODUCTION

The DNA sequencing method developed by the British biochemist Frederick Sanger in 1977 has been widely used for about 40 years. Sanger sequencing is a technique for analyzing a single strand of DNA strands in a single tube. This method has a limited DNA sequencing throughput, which is very costly and time-consuming to analyze the genome [1].

Next Generation Sequencing (NGS) is a next-generation DNA sequencing method that solves the technical bottleneck of sanger sequencing. NGS was first commercialized in 2006 and has since revolutionized genome research. Fig. 1 shows the cost of genome analysis from 2001 to 2017. In this figure, the graph from 2001 to mid-2007 is the cost of genome analysis using sanger sequencing. The graphs since mid-2007 show the cost of genome analysis using NGS. As shown in the figure, NGS has greatly reduced the cost of genome analysis every year.

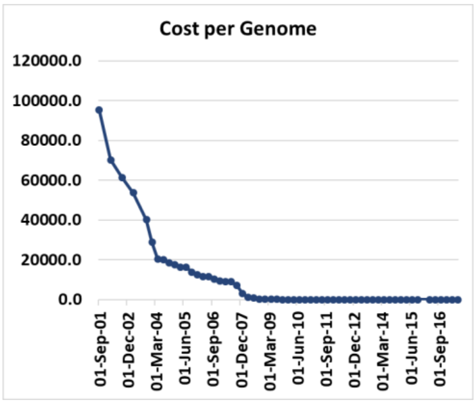

Fig. 1. Cost of DNA analysis

NGS analysis techniques with high throughput have evolved rapidly and are changing the scope of genome research and development of new drugs [3], [4]. With the development of NGS technology, the cost of DNA sequencing has drastically decreased. As a result, NGS analysis has spread to various fields, and genome data has increased exponentially.

With the rapid growth of genomic data, new requirements have emerged that are difficult to handle with big data storage and analysis techniques. Regardless of the size of an organization performing genomic data analysis, it is becoming increasingly difficult for an institution to build a computing environment for storing and analyzing genomic data.

Recently, cloud computing has emerged as a computing environment that meets these new requirements [5]. In a cloud

** Corresponding author, Email: khi69@chsu.ac.kr*

Manuscript received Jan. 03, 2018; revised Apr. 02, 2018; accepted Apr. 02, 2018

computing environment, the computing resources (CPU, memory, storage, etc.) required by the user can be easily and quickly provided in the form of a virtual machine. Cloud computing has spread very quickly due to the characteristics described above. In addition, it is recognized as a means to quickly allocate computing resources required for large-scale NGS analysis and to perform storage and analysis in distributed parallel form.

In this paper, we analyze and compare NGS analysis technology based on cloud. In particular, we focus on cloud-based distributed parallel NGS analysis techniques to speed up large-scale NGS analysis. The composition of this paper is as follows. Section 2 describes NGS in more detail. In Section 3, we analyze cloud-based NGS analysis techniques, and in Section 4, we compare those techniques. Finally conclude in Section 5.

2. NEXT GENERATION SEQUENCING (NGS)

Next Generation Sequencing (NGS) was first commercialized in 2006, bringing about a revolution in genome research for less than a decade. NGS first divides a single genome into many pieces and reads each piece at the same time. Then, ICT is applied to combine the fragments to quickly analyze vast amounts of genomic information [6].

The biggest difference between Sanger Sequencing and NGS is as the following. Sanger Sequencing takes a long time because it uses a very long base sequence for analysis. On the other hand, NGS can greatly reduce the analysis time because it can process the fragments belonging to different regions simultaneously after dividing the nucleotide sequence into numerous pieces.

The NGS analysis process proposed in GATK can be simplified as follows. The core of this process is the generation of raw sequencing reads, reads alignment, reads deduplication, and the detection of variants in the reads (variant calling). In the raw sequencing reads generation step, the genome data is input to the computer from the genome analysis equipment. The sequence alignment step aligns the input DNA reads to a reference genome. In the deduplication step, redundant sequence reads that result from sequencing two or more copies of the exact same DNA fragment introduced during PCR amplification are removed. The variant calling step detects the mutation with deduplicated DNA reads.

The detected mutations are single nucleotide polymorphism (SNP) or short Indels. Then, the SNP information is compared with an existing database (dbSNP) to judge whether the mutation has already been revealed or newly discovered. Annotation also predicts whether the mutation will cause changes to the amino acid and what effect it will have on the protein structure. For extracted SNPs and Indels, further work can be done to improve the quality of the information [9].

There are several software tools used for each step of the NGS analysis as shown in Table 1. Analysts who create the NGS analysis pipeline can choose the appropriate tools in stages. Recommended best practices by GATK are BWA, Picard and GATK [20]. proposes a pipeline consisting of BWA, Picard, and GATK [21]. proposes a pipeline consisting of GATK, BWA, MuTect and MutSig.

Table 1. Tools for each NGS steps

Steps	Tools	GATK best practices
alignment	BWA [10], GATK [11], Bowtie [12]	BWA-MEM
mark duplicates	Picard [13], Samblaster [14]	Picard
variant calling	GATK, MuTect [15], MutSig [16], Freebayes [17], ANNOVAR [18], VEP [19]	GATK

3. DISTRIBUTED AND PARALLEL NGS ANALYSIS METHODS

In a cloud computing environment, the computing resources required for a task to be processed by a user can be allocated in a virtual machine form. That is, appropriate virtual machines can be allocated and used as needed for one NGS data analysis process. For example, in the cloud computing environment, computing resources can be allocated according to the load of tasks such as 8 virtual machines, 2 for deduplication, 8 for mutation detection, etc. in order to quickly sort the reference genome. There is an opportunity to accelerate.

However, the existing analysis tools in Table 1 are designed and developed without consideration of the cloud computing environment. There is a problem that the existing tools must be completely redesigned in order to reduce the execution time by using the computing resources provided by the cloud computing environment in a parallel distributed manner.

To solve this problem, several methods such as [24]-[29] using Apache Hadoop MapReduce [22] or Apache Spark [23] are proposed in the past several years. These methods build a cluster consisting of many computer nodes and accelerate NGS data analysis using a distributed parallel processing framework such as MapReduce or Spark.

BigBWA [25] and SparkBWA [26] allow distributed parallel processing of BWA, the sorting tool for the reference genome, based on MapReduce and Spark, respectively. These methods used an existing BWA approach without modification. They take the existing BWA in Spark or MapReduce to perform several processes at the same time with native code invocation method like JNI. [27] proposed a method to perform de-duplication by executing Samblaster in distributed parallel manner by inputting in stream form without saving the output from BigBWA.

[24], [28], [29] also take an approach similar to SparkBWA and BigBWA, but run the entire NGS analysis process. We will analyze them in more detail in the following sections.

3.1 Cluster-Based Apache Spark Implementation of the GATK DNA Analysis Pipeline [28]

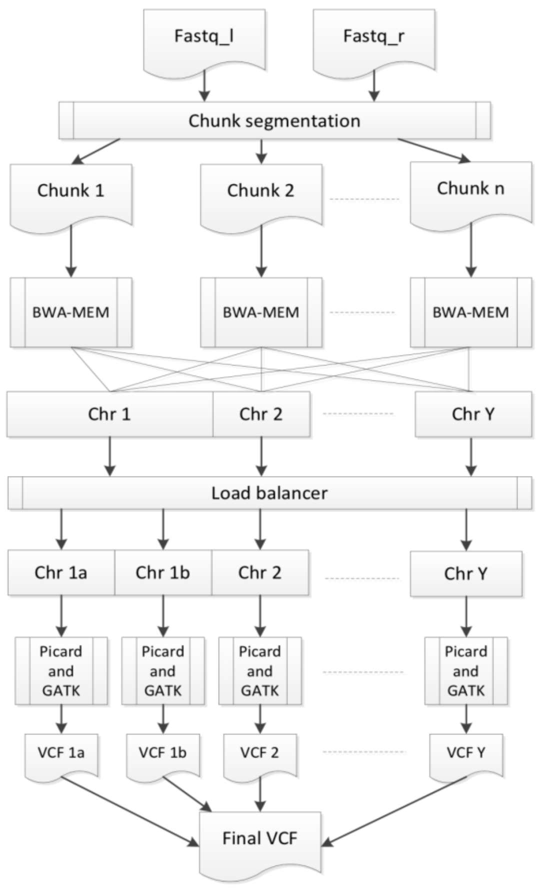

Fig. 2. NGS analysis process of [28]

In this paper, authors presented a framework to parallelize the steps of the GATK best practices using Apache Spark. The framework segments an input data to achieve scalability. It proposed a load balancing algorithm that split chromosomal regions according to the number of reads to each chromosome rather than the static length of the chromosomes. To reduce costs for the split of chromosome regions, it processes the split in memory.

The BWA-MEM of the GATK best practices is able to be scalable on a multicore system using multi-threading. However, for other tools in the GATK best practices it is difficult to run in parallel. The framework is designed to provide a generic method to ensure scalability to the various genomics analysis pipelines by using the Apache Spark.

Fig. 2 shows the overall architecture and the analysis process of the proposed framework in [28]. In this figure there are two FASTQ (2 ends of a pair of sequences) files are generated. The two FASTQ files are split into a number of chunks and uploaded into HDFS. All nodes in a cluster are able to access the chunks in HDFS. BWA-MEM processes in the cluster nodes perform DNA alignment of the short reads against a reference genome chunks in HDFS in parallel.

This step produces a SAM file which includes a list of read alignments (SAM records). SAM records are read into <key, value> pairs in the memory and then divides them into sub-chromosome regions using the load balancing algorithm described above to ensure a better distribution of the subsequent tasks. The read <key, value> SAM records are sorted according to the position field of records. Then mark duplicates step and variant calling step are performed in parallel on each sub-chromosomal region separately. Finally, multiple VCF files are produced and they are merged into one VCF file.

3.2 Halvade [24]

Halvade is a framework that enables parallel NGS analysis on multi-core compute infrastructure as well as multi-node cluster infrastructure. Halvade is designed based on the observation that read alignment and variant calling is able to run in parallel by read and chromosomal region. The alignment of one read is independent of the alignment of another read so the read alignment step can be processed in parallel. Variant calling step, also, is parallel by chromosomal region. That is, variant calling in one chromosomal region is independent of variant calling in a different chromosomal region.

Halvade is based on Apache Hadoop MapReduce while [28] is based on Apache Spark. The map phase of Halvade is the read alignment step and the reduce phase is the variant calling step. The output of the map phase (read alignment step) is sorted in parallel according to the aligned position before processed by the reduce phase (variant calling step). Halvade is designed to achieve a good load balance, maximize data locality and minimize disk I/O by avoiding file format conversions.

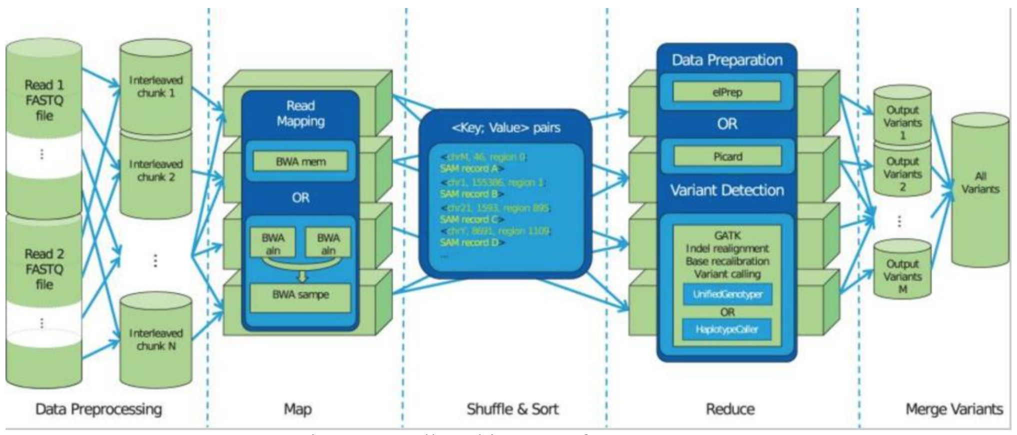

Fig. 3. Overall Architecture of Halvade [24]

Overview of the Halvade framework is shown in Fig. 3. Like Fig. 3, there are two input FASTQ files (2 ends of a pair of sequences). Two FASTQ files are segmented into a number of interleaved chunks. Map tasks for read alignment step are executed in parallel. Each map task process a single input chunk to align the reads in the chunk to a reference genome using BWA. The map tasks produce <key, value> pairs where the key contains position field of a SAM record. The SAM records are grouped into chromosomal regions. The reduce phase for variant calling step process chromosomal regions in parallel. The reduce phase includes data preparation and variant detection. GATK is used in this step. Each reduce task outputs the multiple VCF files, and the files are optionally merged into a single VCF file.

3.3 SparkGA [29]

SparkGA is an upgrade version of [28]. Since [28] performs its entire load balancing step in memory, it results in out of memory errors for large input. SparkGA is proposed to solve the memory problems. In addition, since the memory and computational requirements for different steps vary, SparkGA runs the pipeline in three different application programs such as DNA mapping and load balancing, marking of duplicates and variant calling. SparkGA allows users to tune memory and cores of executors for all those application programs.

The work flow of SparkGA is shown in Fig. 4. There are two FASTQ files generated as input. These two input FASTQ files are divided into interleaved chunks and uploaded into HDFS. Each chunk is then processed by a read alignment task that performs read alignment using BWA-MEM. The output of this step is a SAM file which consists of a list of read alignments (SAM records). Since the length of each chromosome is given, it is possible to perform an approximate load balancing by reading the output of the SAM files produced by each BWA mem task. In addition, SparkGA perform dynamic load balancing to distribute reads evenly according to actual reads available at runtime. Each created chromosomal region based on actual reads achieves a better performance.

Subsequently, for each chromosomal region, the aligned reads are sorted using the position fields of SAM records. Then, the rest of the analysis steps are performed in parallel like [28]. Finally, multiple VCF files for each chromosomal region are produced as output and they are merged into a single VCF file.

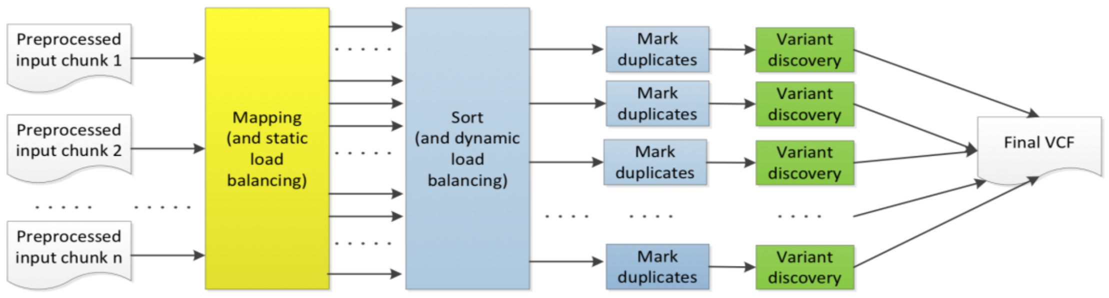
Preprocessed input chunk 1
Preprocessed input chunk 2
Preprocessed input chunk n
Mapping (and static load balancing)
Sort (and dynamic load balancing)
Mark duplicates
Variant discovery
Final VCF

Fig. 4. Workflow of SparkGA

4. COMPARISON

We compare the three existing methods such as [28], Halvade and SparkGA. Halvade is based on Apache Hadoop MapReduce while others are based on Apache Spark. Using Apache Spark has several advantages. First, the code written is more simple as compared to Hadoop MapReduce. Instead of using a map followed by a reduce step, Spark allows our code to contain a few cascaded map calls [28]. Second, Apache Spark tries to perform operations on reads in memory of the nodes, so as to reduce the number of disk IOs. Since Apache Spark has some advantages, it is expected that [28] and SparkGA may outperform Halvade.

[28] has some memory problems while performing load balancing step while SparkGA can process load balancing with small memory. In [29] authors claim that even on a single node with just 16 GB of RAM large data can be processed. Also, in [28], the entire NGS analysis process corresponds to a single Apache Spark application. However, SparkGA performs the NGS analysis in three applications for the three steps such as reads alignment and static load balancing, sorting and dynamic load balancing, and mark duplicates and variant discovery. Through this approach, each application can be executed with different optimized Spark execution parameters

5. CONCLUSION

This paper discusses the basic concepts and analysis process of NGS analysis and describes what software tools are used at each stage of the analysis process. We also looked at the use of cloud computing as a means of allocating the computing resources needed for large-scale NGS data analysis. Finally, we describe the methods of using the distributed parallel characteristics of the cloud computing environment for large-scale NGS data analysis. Most of these methods are distributed and parallelized based on the distributed parallel processing framework.

This approach seems to have the advantage of not modifying core algorithms of widely used software tools. However, there are limitations in using all of the advantages of distributed parallel frameworks when performing existing code as is. In particular, the input / output of each software tool repeatedly occurs on the local hard disk, which limits the performance improvement. In future research, it will be necessary to study the improvement of performance by controlling input / output without changing core algorithm of existing analysis software.

REFERENCES

[1] M. Choi, "Development Trends of Medical Genomics Using Next Generation Sequencing Techniques," Molecular Cell Biology Newsletter, Apr. 2014.
[2] https://www.genome.gov/sequencingcostsdata/
[3] M. C. Schatz, B. Langmead, and S. L. Salzberg, "Cloud Computing and the DNA Data Race," Nature Biotechnology, vol. 28, no. 7, 2010, pp. 691-693.
[4] M. Baker, "Next-generation Sequencing: Adjusting to Data Overload," Nature Methods, vol. 7, no. 7, 2010, pp. 495-499.
[5] B. Calabrese and M. Cannataro, "Bioinformatics and Microarray Data Analysis on the Cloud," Methods in Molecular Biology, vol. 1375, 2016, pp. 25-39.
[6] http://ngenebio.com/
[7] C. Lee, *Bioinformatics Analysis of Next-Generation Sequence Data*, BRIC View Trend Report, 2016
[8] A. Geraldine, V. Auwera, M. O. Carneiro, C. Hartl, R. Poplin, G. Angel, A. Levy-Moonshine, T. Jordan, K. Shakir, D. Roazen, J. Thibault, E. Banks, K. V. Garimella, D. Altshuler, S. Gabriel, and M. A. DePristo, "From FastQ Data to High Confidence Variant Calls: the Genome Analysis Toolkit Best Practices Pipeline," Current Protocols in Bioinformatics, 2013, pp. 11-10.
[9] https://www.bioin.or.kr/board.do?cmd=view&bid=tech&num=216321
[10] BWA, https://github.com/lh3/bwa
[11] GATK, https://software.broadinstitute.org/gatk/
[12] B. Langmead, C. Trapnell, M. Pop, and S. Salzberg, "Ultrafast and Memory-efficient Alignment of Short DNA Sequences to the Human Genome," Genome biology, vol. 10, no. 3, 2009.
[13] http://broadinstitute.github.io/picard/
[14] https://github.com/GregoryFaust/samblaster
[15] https://github.com/broadinstitute/mutect
[16] https://hpc.nih.gov/apps/MutSig.html
[17] https://github.com/ekg/freebayes
[18] https://github.com/WGLab/doc-ANNOVAR/
[19] https://www.ensembl.org/vep
[20] https://gencore.bio.nyu.edu/variant-calling-pipeline/
[21] https://wikis.utexas.edu/display/bioiteam/DNAseq+Variant+Calling+Pipeline
[22] https://hadoop.apache.org/
[23] https://spark.apache.org/
[24] D. Decap, J. Reumers, C. Herzeel, P. Costanza, and J. Fostier, "Halvade: Scalable Sequence Analysis with MapReduce," Bioinformatics, vol. 31, no. 15, 2015, pp. 2482-2488.
[25] https://github.com/citiususc/BigBWA
[26] https://github.com/citiususc/SparkBWA
[27] J. Lee, H. Lee, J. Moon, H. Kang, S. Song, and S. Yu, "Parallel and Distributed PCR Duplication Marking Algorithm Integrated with Genome Sequence Alignment by Using Streaming Technology," Proceedings of TBC 2017, 2017.
[28] H. Mushtaq and Z. Al-Ars, "Cluster-based Apache Spark Implementation of the GATK DNA Analysis Pipeline," In Proceedings of IEEE International Conference on Bioinformatics and Biomedicine (BIBM), 2015, pp. 1471-1477.
[29] H. Mushtaq, F. Liu, C. Costa, G. Liu, P. Hofstee, and Z. Al-Ars, "Sparkga: A Spark Framework for Cost Effective, Fast and Accurate DNA Analysis at Scale," In Proceedings of the 8th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics, 2017, pp. 148-157.

Hyungil Kang
He received the BS, MS degrees in Mokpo University in 1996 and 1998 respectively. He received PhD degree in Computer and Communication Department from Chungbuk National University in 2002. He is an Associate Professor of Department of Semiconductor Electronics Engineering, Chungbuk Health & Science University, Republic of Korea. His research interests are database systems, XML database, bigdata.

Sangsoo Kim
He received the BS in Mokpo University in 1995 and MS in Cheonbuk University in 2006. He is a researcher of Course-based Qualification Exam Team2, Human Resources Development Service of Korea, Republic of Korea. His research interests are bioinformatics and bigdata.