Supercomputing for Big Data ET4310 (2016)

Assignment 3

Parag Bhosale(4517415)

November 2016

Introduction

The aim of the assignment is to use Apache Spark in the GATK DNA analysis pipeline. The goal of the Spark in this assignment to implement the pipeline in parallel for independent data chunks in order to get lower execution time. Another goal of the assignment is to use HDFS system to download and upload the data.
In background section, the problem of the DNA analysis is stated and how it can be solved using Apache Spark is stated. The implementation section will highlight the implementation strategy for the solution. In next section, results are discussed. In the last section, the conclusion is made.

Background

The assignment is distributed in 3 parts. The first part is to interleave two FASTAQ files according to records and write them in parallel. To process the partitions of the RDD in parallel, we must use **foreachPartitions**, **mapPartitions** or **mapPartitionswithIndex**. These transformations will process the partitions in parallel. The level of parallelism is defined by the number of the cores(local mode). Consider, if a RDD had 8 partitions and the number of cores is 4, then 4 partitions will run in parallel. Once 4 partitions are done, the remaining partitions will be processed. To have better parallelism, we need to distribute the data in equal partitions so that no core will be idle. Similar concept of mapPartitions is used in GATk pipeline(BWA and variant calling)
Hadoop is used in this assignment as HDFS system. In the third part, input files are downloaded from HFDS to a local directory. Similarly, the final output file is uploaded to HDFS. This has an application when Spark is run in cluster mode.
In this assignment, interleaved raw sequence is mapped against a reference genome. The reference genome has 24 chromosome which indicates the reference genome is human genome. For mapping, Burrows-Wheeler Aligner algorithm(bwa mem) is used. After this, we will find out mapped SAM records. For bwa mem, mapPartitions is used as specified earlier. We must balance the partitions of the output of the BWA-MEM step so that we can exploit the mapPartitions features. Let us suppose each partition takes 5 minutes to complete the BWA-MEM and there are 8 such partitions and program will be run on 8 cores. This means the whole BWA-MEM will be completed in 5 minutes where all partitions will be processed in parallel.

Similarly, variant calling is done in parallel. Variant calling will generate the output file of format**.vcf** and this file logs the variation in gene sequence.
This whole pipeline is used in the field of bioinformatics. DNA analysis is one the main big data applications. One of the main bottlenecks of the DNA analysis is huge data and hence computation time. This assignment will provide the one of the solutions to this problem.

Implementation

Part one

The first part is to interleave the two faster short reads. We have used zipwithindex transformation to achieve the required result. Further foreachPartitions is used to write the **fq.gz** files in parallel.

Part Two

Output chunks written in part one are processed in parallel in this part. External config.xml is used to parametrize the required paths and the parameters. When BWA MEN is done, loadbalancer is used to balance the load. The further variant call is executed in parallel. But before that records are sorted using sortwith and compareSAM functions.

Part Three

In this part, the functionality of part two is extended further. Hadoop is used in this part as HDFS. We have written the library Filemove to upload and download the files to and from the HDFS. For this part, methods(**copyFromLocalFile,copyToLocalFile**) from **org.apache.hadoop** package are used.

Results

The first part of assingment was ran on the Lenovo laptop and remaining two were ran on Kova machine.

Part one

Result of the part was as expected. All chunk files were parallely . As we can see in the table 1,

Runtime(sec) with out compressed	Runtime(sec) with compressed
25	40

Table 1: Average run time for method one

the execution time is increased as files are written directly in .gz from the spark. We can optimize this by zipping the files by coding in **run.sh** scripts.

Part Two

Results of the part is not entirely as expected. The main concerns of the result is difference of the my output file and reference output file. We can see the differences and execution time for both parts 2 and 3 in the table 2. Format of vcf file and log files are created as expected. From timestamp in logs we can conclude that all 4 partitions are processed parallely.

Part	Runtime(min)	Difference
Part 2	18	131
Part 3	18	146

Table 2: Average run time for the method two

Bottleneck

If we observe the vc and bwa logs carefully, we can conclude that a lot of time is taken when bwa mem is completed and variantcall is called. This means loadbalancer is taking more than expected. We would need better loadbalancer and repartitioning.
Another bottleneck would be groupByKey used. groupByKey is less efficient than reduceByKey. But this will not have significant improvement as data is not that huge.

Part Three

The result for this part is expected. All required files are moved to and from the HDFS as expected. The implementation of the Filemove library was successful. The execution time and results are shown in table 2.

Conclusion

In this assignment, Apache Spark is used successfully to parallelize the GATK DNA analysis pipeline using **foreachPartitions and mapPartitions**. Logs are written in parallel tasks to log the timings and details so that they can be used while analyzing the performance of the program. But at the same time, a faster loadbalancer is needed and further analysis is needed to reduce the difference.