International Journal of Internet, Broadcasting and Communication Vol.16 No.2 119-126 (2024)
http://dx.doi.org/10.7236/IJIBC.2024.16.2.119

IJIBC 24-2-15

Design of Distributed Cloud System for Managing large-scale Genomic Data

Seine Jang*, Seok-Jae Moon**

** The master's course, Graduate School of Smart Convergence, Kwangwoon University, Seoul, Korea*
*** Professor, Graduate School of Smart Convergence, Kwangwoon University, Seoul, Korea*
E-mail : {seine1025, msj8086}@kw.ac.kr

Abstract

The volume of genomic data is constantly increasing in various modern industries and research fields. This growth presents new challenges and opportunities in terms of the quantity and diversity of genetic data. In this paper, we propose a distributed cloud system for integrating and managing large-scale gene databases. By introducing a distributed data storage and processing system based on the Hadoop Distributed File System (HDFS), various formats and sizes of genomic data can be efficiently integrated. Furthermore, by leveraging Spark on YARN, efficient management of distributed cloud computing tasks and optimal resource allocation are achieved. This establishes a foundation for the rapid processing and analysis of large-scale genomic data. Additionally, by utilizing BigQuery ML, machine learning models are developed to support genetic search and prediction, enabling researchers to more effectively utilize data. It is expected that this will contribute to driving innovative advancements in genetic research and applications.

Keywords: *HDFS, Spark on YARN, Big data, Distributed Cloud System, BigQuery ML*

1. INTRODUCTION

The scale of data generated in diverse industries and research domains has consistently exceeded petabyte levels and continues to grow. Notably, with the global execution of large-scale human genomic analysis projects and the evolution of gene sequencing technologies, the volume of genomic data is witnessing continuous expansion. Consequently, there is ongoing research aimed at effectively integrating and analyzing this large-scale genomic data. To effectively integrate and manage rapidly generated big data in various formats, high-cost hardware and software are required [1]. Distributed cloud computing, capable of processing the demands of data analysis in real-time, emerges as the most suitable technology for this purpose. In 2017, the DNA Data Bank of Japan (DDBJ) Center started a cloud service called DDBJ Group Cloud (DGC) utilizing the NIG supercomputer [2]. In addition, in 2021, the UK Biobank has been striving to enhance global accessibility by introducing a cloud-based research analysis platform for researchers [3]. However, existing gene integration systems are typically built on relational database platforms, analyzing and processing data using structured normalized queries or non-structured queries. Furthermore, they execute machine learning through high-level programming languages like Python. Google Cloud Platform's BigQuery ML is a cloud-based machine learning service that enables the construction and deployment of machine learning models using SQL queries [4]. BigQuery ML offers the advantage of creating operational-level machine learning models

Manuscript Received: March. 28, 2024 / Revised: April. 6, 2024 / Accepted: April. 12, 2024
Corresponding Author: msj8086@kw.ac.kr
Tel: 02-940-8283, Fax: 02-940-5443
Author's affiliation: Professor, Graduate School of Smart Convergence, Kwangwoon University, Seoul, Korea

with minimal coding using SQL tools and techniques, without requiring knowledge of statistics, algorithms, data preprocessing, or high-level programming languages.

This paper proposes a distributed cloud system for the management of large-scale genomic data. The proposed system is built upon HDFS, effectively integrating various gene databases for storing and managing data [5]. It ensures data interoperability in a secure environment with data integrity. Additionally, the combination of Spark on YARN manages optimal resource allocation for distributed cloud tasks and constructs modules capable of processing large-scale genomic data sets rapidly [6]. Furthermore, in this proposed system, for genomic data analysis, SQL's INNER JOIN combines the user input data with the tables from the gene databases to create new result tables. These newly generated result tables for each database are then utilized by BigQuery ML to not only perform gene search but also to create machine learning models, enabling analysis and training functionalities within the system.

The structure of this paper is as follows: Chapter 2 describes the components, operational structure, and algorithms of the proposed system. Chapter 3 provides a comparative analysis between the existing system and the proposed system. Finally, Chapter 4 concludes the paper.

2. PROPOSED SYSTEM

2.1 System Components

The proposed system in this paper enables the integration and interoperability of distributed databases. Additionally, it provides modules for the creation and analysis of machine learning models for gene search and prediction using BigQuery ML. Fig 1 overviews the proposed system, comprising primarily of Hadoop's HDFS, YARN, Apache Spark, and BigQuery ML.

The main roles of the system components are as follows:

- BigQuery ML: A cloud-based machine learning service provided by Google Cloud Platform. It allows for data preprocessing, machine learning model creation, and training using simple SQL syntax, without requiring knowledge of complex tools or high-level programming languages. Utilizing the familiar SQL query syntax, BigQuery ML combines tables from various databases using INNER JOIN to generate new result tables for machine learning analysis.

- Spark: Apache Spark provides fast processing and flexibility by employing immutable data structures for parallel and distributed processing of big data.

- YARN: YARN manages resource allocation and job scheduling in Hadoop clusters, providing an optimal environment for large-scale data processing.

- HDFS: HDFS ensures data reliability and availability by distributing storage across multiple nodes. Its scalability allows for storing various types and sizes of data effectively.

- Raw Data DB: A database archiving human genomic datasets submitted by researchers worldwide across various fields. These databases are distributed across multiple countries and research institutes.

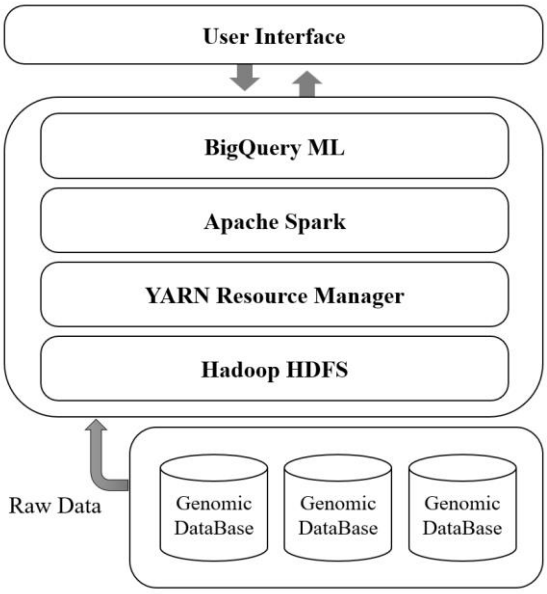

Figure 1. Architecture overview of the proposed system

As presented in Fig 2, the user inputs the genomic data table they want to analyze. Upon receiving the input from user, Spark accesses the raw databases managed in HDFS. SQL queries are executed to parse the data tables stored in each database and the input data table from the user. Using INNER JOIN, Spark combines tables containing data that matches the specified range provided by user to create a new table. Subsequently, the new result table is passed to the BigQuery ML execution module. At this stage, the user writes SQL queries to create machine learning models. BigQuery ML module then executes machine learning based on the received result tables to perform gene search and prediction. The resulting new data table is then returned to user.

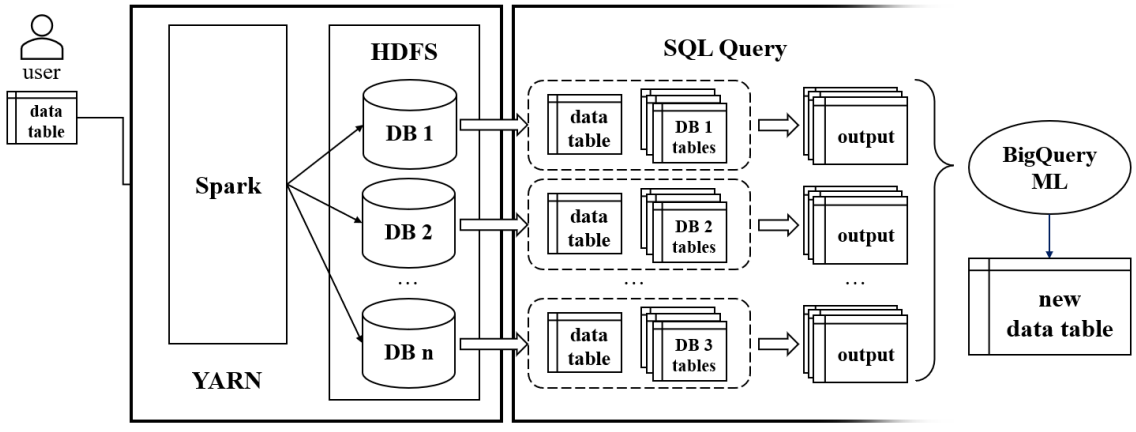

Figure 2. Interaction among user, distributed cloud system and BigQuery ML

2.2. BigQuery ML

Short Tandem Repeats (STRs) are repetitive DNA sequences occurring at specific loci within the human DNA, characterized by varying numbers of repeat units among individuals [7]. Fig 3 shows the anticipated

structure of the data tables based on a sample composed of D8S1179, D21S11, D7S820, CSF1PO, D3S1358, among over a million STR sequences, stored in each respective database.

Locus	ANOM
D8S1179	9, 14
D21S11	28, 29
D7S820	8, 9
CSF1PO	10, 12
D3S1358	15, 18
…	…

Figure 3. Structure of the data tables stored in each raw data DB

The following Table 1 shows the algorithm for extracting data tables to execute BigQuery ML using SQL queries. The algorithm allows user to specify a tolerance range for the values of the input genomic data. This range is utilized to extract data tables similar to the input data, enabling user to specify and combine tables based on their genomic data values. The `input_data` represents the genomic data table input by user, consisting of the locus of each gene and the corresponding two allele values observed at that locus. By defining the `input_data` table in this manner and performing an INNER JOIN with another table stored in the database (`other_data`), a new table (`joined_data`) is created. The condition for the INNER JOIN is that the LOCUS value in `input_data` matches the LOCUS value in `other_table`, and the two Value values fall within a certain range. The resulting `joined_data` table, generated by joining rows meeting these conditions, is then extracted. Following the process outlined in Table 1, BigQuery ML is executed based on the data tables combined for each database.

Table 1. SQL query algorithm

Algorithm 1: SQL query algorithm

```
Input: input_data
WITH input_data AS (
    SELECT 'D8S1179' AS LOCUS, '9, 14' AS Value
    UNION ALL SELECT 'D21S11', '28, 29'
    UNION ALL SELECT 'D7S820', '8, 9'
    UNION ALL SELECT 'CSF1PO', '10, 12'
    UNION ALL SELECT 'D3S1358', '15, 18'
),
joined_data AS (
    SELECT input_data.*, other_data.*
    FROM input_data
    INNER JOIN other_table AS other_data
    ON input_data.LOCUS = other_data.LOCUS
    AND (
        ABS(SPLIT(input_data.Value, ',')[OFFSET(1)]-other_data.Value) <= 1
        OR ABS(SPLIT(input_data.Value, ',')[OFFSET(2)-other_data.Value] < =1
    )
)
SELECT * FROM joined_data;
```

Table 2 shows the algorithm for creating, evaluating, and performing predictions on the data using BigQuery ML to predict the race of the `input_data`. The logistic regression model is created using the CREATE MODEL statement. The necessary input variables for model training are retrieved from `joined_data`, and the labels to be predicted are fetched from `input_data`. These values are combined based on the LOCUS value using INNER JOIN. The performance of the generated model is evaluated using the ML.EVALUATE function, which includes performance metrics such as accuracy, precision, and recall in the evaluation results. Subsequently, the ML.PREDICT function is employed to predict the race of the corresponding gene (`input_data`) and return the associated probability values. The choice of machine learning algorithm may vary depending on the nature of the actual data, and considering more features is essential for improving the performance of the model.

Table 2. BigQuery ML algorithm

Algorithm 2: BigQuery ML algorithm

```
--CREATE ML MODEL
CREATE MODEL 'your_project.your_datset.race_prediction_model'
OPTION (model_type=;logistic_reg') AS
SELECT
    other_data.*;
    input_data.race AS label
FROM
    Joined_data AS other_data
INNER JOIN
    input_data
ON
    Other_data.LOCUS = input_data.LOCUS;

--EVALUATE ML MODEL
SELECT
    *
FROM
    ML.EVALUATE(MODEL 'your_project.your_dataset.race_prediction_model');

--PREDICT THE RESULT
SELECT
    predicted_label,
    predicted_probabilities
FROM
    ML.PREDICT(MODEL 'your_project.your_dataset.race_prediction_model',
    (
        SELECT
            *
        FROM
            Input_data
        )
    );
```

2.3 Operational Procedure

This chapter describes the sequential process of the proposed system.

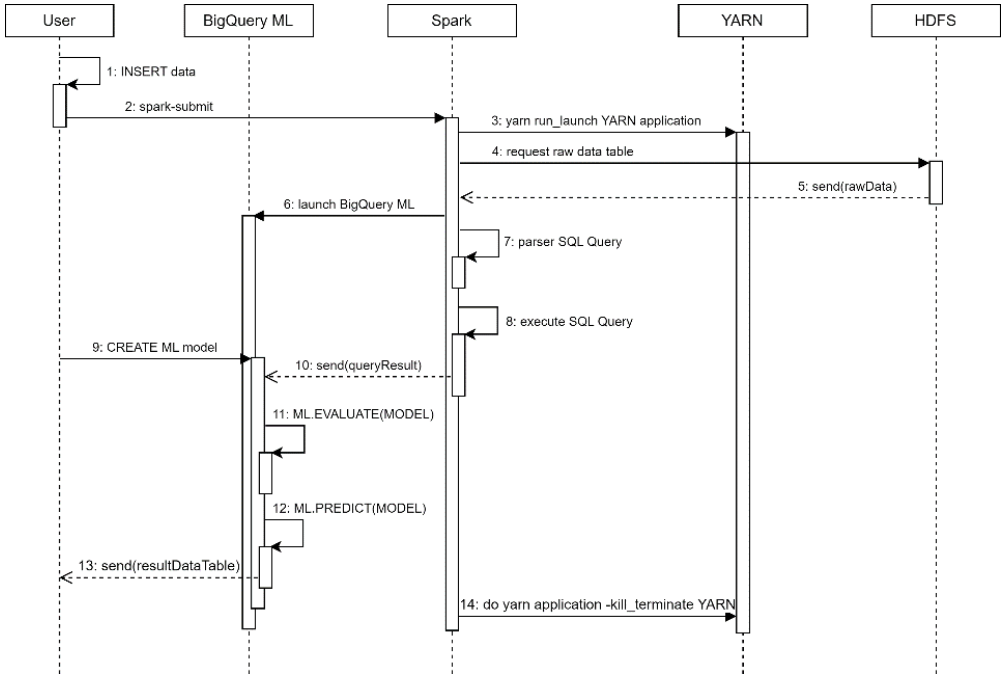

Figure 5. Sequence diagram of the proposed system

As presented in Fig. 5, when user inputs a data table, Spark initiates YARN for distributed cloud computing. The YARN resource manager allocates and manages cluster resources to maintain an optimal environment throughout the system's operation. Additionally, Spark accesses databases managed in HDFS, loading data and performing distributed processing. During this process, SQL queries execute syntactical analysis of the data tables input by user and each database, utilizing INNER JOIN.

Next, Spark operates the BigQuery ML module and executes SQL queries to process the combined tables. The user writes BigQuery ML SQL syntax to create a machine learning model. Based on the syntax provided by user, BigQuery ML analyzes the data tables and performs predictions. The final result, a newly generated data table from the machine learning process, is then transmitted to user. If no further distributed cloud computing is required, Spark terminates YARN.

3. COMPARSION OF SYSTEMS

Table 3 summarizes the comparison results of software used for research, including DNA analysis in fields such as forensics and biology, namely DNAxs, Galaxy, and Geneious [8-10]. In this paper, comparisons were made regarding server scalability, whether machine learning support is provided, along with the programming language used by each supported software and the engine for database search.

Table 3. Comparison of systems

	DNAxs [8]	Galaxy [9]	Geneious Prime [10]	Proposed System
Server Scalability	Runs on both locally and on a Windows or Linux server that support Java 17 or higher, The statistical library in DNAxs supports parallel computing	Easily expand a Galaxy server available compute capacity by sending user jobs to cloud resources	Runs on the user's local computer and offers a cloud-based data storage option	Supports distributed cloud computing based on Hadoop
Machine Learning	Using 'NOC' tools	Using a toolkit 'Galaxy-ML'	Unavailable	Using 'BigQuery ML'
Language Support for ML	Python, Java	Python	Unavailable	SQL
Database Search Engine	Using 'SmartRank'	Using 'Galaxy GUI'	Using 'BLAST'	Using 'Apache Spark'

The first involves the server scalability. DNAxs can run on the user's local computer and on servers supporting Java 17 or higher on Windows and Linux. However, it does not support cloud-based data storage, and its integrated 'DNAStatistX' module supports parallel computing. Galaxy, through 'GalaxyCloudRunner,' facilitates easy scalability of Galaxy servers by transferring user tasks to cloud resources, with resources dynamically allocated from AWS, Azure, GCP, or OpenStack in an automated manner. Geneious Prime offers a cloud-based data storage option called 'Geneious Cloud' and executes data analysis on the user's local computer. However, it does not support data sharing, and once data is deleted from the cloud, it is permanently removed without the option for recovery. Additionally, automatic backup with the user's local database is not supported. On the other hand, the proposed system provides a Hadoop-based distributed cloud environment, enabling database sharing across all operating system environments. HDFS distributes data across multiple nodes for replication, thus preventing data loss and facilitating easy recovery. From the machine learning perspective, Galaxy utilizes the 'Galaxy-ML' toolkit to create a complete machine learning pipeline, including normalization, feature selection, model definition, hyperparameter optimization, and cross-fold validation through a web-based user interface. Galaxy-ML is accessible only through web browsers and is capable of handling large datasets. It is written in Python, allowing users to utilize the Galaxy-ML API with Python. Additionally, DNAxs includes an estimation NOC tool based on machine learning approaches, automatically providing an estimation NOC. However, Geneious Prime does not support machine learning. Regarding database search capabilities, DNAxs can perform DNA database searches using 'SmartRank' internally, allowing for more complex mixed DNA profile searches. Galaxy, through its web-based graphical user interface (Galaxy GUI), enables searches in public databases. Additionally, Geneious Prime can perform seven types of searches on databases stored in NCBI or local databases using 'BLAST (Basic Local Alignment Search Tool)'. The proposed system can conduct SQL searches by accessing integrated raw databases in HDFS via Apache Spark.

4. CONCLISION

In this paper, a distributed cloud system effectively combining existing technologies is proposed for the management and analysis of large-scale genetic data. Through the integration of Hadoop HDFS, YARN, Spark, and BigQuery ML, the process from data processing to machine learning model generation is streamlined and optimized. This system efficiently integrates large-scale databases, providing users with cost-effective computing resources and eliminating the maintenance requirements of centralized systems. Moreover, users can benefit from performing data processing, machine learning model creation, evaluation, and prediction using only SQL, without the need for expertise in advanced programming languages like Python or Java. This marks a pivotal turning point towards innovative advancements in the field of genetic data analysis. Additionally, we expect to open up new possibilities for efficient data management and analysis in research and application domains by proposed system.

ACKNOWLEDGMENT

※ This paper was supported by the KwangWoon University Research Grant of 2024.

References

[1] "Big Data Analysis using BigQuery on Cloud Computing Platform," Australian Journal of Engineering and Innovative Technology. Universe Publishing Group - UniversePG, pp. 1–9, Jan. 27, 2021, DOI: https://doi.org/10.34104/ajeit.021.0109.
[2] https://www.ddbj.nig.ac.jp/services/ddbj-group-cloud-e.html
[3] N. E. Allen et al., "Prospective study design and data analysis in UK Biobank," Science Translational Medicine, vol. 16, no. 729. American Association for the Advancement of Science (AAAS), Jan. 10, 2024, DOI: https://doi.org/10.1126/scitranslmed.adf4428.
[4] https://cloud.google.com/bigquery/docs/bqml-introduction
[5] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The Hadoop Distributed File System," 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST). IEEE, May 2010, DOI: https://doi.org/10.1109/msst.2010.5496972.
[6] H. Zhang, H. Huang, and L. Wang, "Meteor: Optimizing spark-on-yarn for short applications," Future Generation Computer Systems, vol. 101. Elsevier BV, pp. 262–271, Dec. 2019, DOI: https://doi.org/10.1016/j.future.2019.05.077.
[7] P. N. Lopez Gonzalez, J. E. Bautista-Gonzalez, M. J. Lopez-Gonzalez, J. E. Sosa-Escalante, and L. Gonzalez-Herrera, "The most frequent autosomal STRs involved in exclusion of paternity cases in a population from southeast, Mexico," Forensic Science International: Genetics Supplement Series, vol. 7, no. 1. Elsevier BV, pp. 465–467, Dec. 2019, DOI: https://doi.org/10.1016/j.fsigss.2019.10.053.
[8] https://www.forensicinstitute.nl/products-and-services/forensic-products/dnaxs
[9] N. Goonasekera, A. Mahmoud, J. Chilton, and E. Afgan, "GalaxyCloudRunner: enhancing scalable computing for Galaxy," Bioinformatics, vol. 37, no. 12. Oxford University Press (OUP), pp. 1763–1765, Oct. 26, 2020, DOI: https://doi.org/10.1093/bioinformatics/btaa860.
[10] https://www.geneious.com/features/cloud/