Implementing Hadoop and Moodle Integrated Student Support System

Myint Myat Aung
Data Distribution Management Lab
University of Information Technology
Yangon, Myanmar
myintmyataung@uit.edu.mm

Shin Min Htet
Data Distribution Management Lab
University of Information Technology
Yangon, Myanmar
shinminhtet@uit.edu.mm

Hein Myo Set
Data Distribution Management Lab,
University of Information Technology
Yangon, Myanmar
heinmyoset@uit.edu.mm

Aung Hpone Thant
Data Distribution Management Lab
University of Information Technology
Yangon, Myanmar
aunghponethant@uit.edu.mm

Sithu Htut
Data Distribution Management Lab
University of Information Technology
Yangon, Myanmar
sithuhtut@uit.edu.mm

Kyaw Zin Htet
Data Distribution Management Lab
University of Information Technology
Yangon, Myanmar
kyawzinhtet@uit.edu.mm

Nwe Nwe Myint Thein
Data Distribution Management Lab
Faculty of Computer Science
University of Information Technology
Yangon, Myanmar
nwenwemyintthein@uit.edu.mm

2024 5th International Conference on Advanced Information Technologies (ICAIT) | 979-8-3315-1004-6/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICAIT65209.2024.10754929

***Abstract*—Learning management systems (LMS) generate large amounts of real-time data which has the potential to provide valuable insights. Hadoop MapReduce is a scalable framework that is designed to handle such data. This paper presents a system architecture that integrates Hadoop MapReduce into Moodle, a popular LMS, for analyzing academic data. The implementation of the architecture allows users to generate a variety of reports from student attendance logs through a Moodle plugin which utilizes the Hadoop cluster for processing data. Users such as lecturers can then utilize these reports to identify which students do and do not engage in which courses as well as monitor their eligibility for exams. The use of Hadoop MapReduce provides high performance computing (HPC) capabilities that offer distributed parallel processing and economical scalability by means of adding commodity nodes to the cluster.**

Keywords—Learning management system, LMS, Moodle, Hadoop, MapReduce, Big data, High performance computing, HPC, Attendance

I. Introduction

In this modern age, the integration of the efficient data processing frameworks is becoming more critical because most of the schools and universities use more and more technology to handle huge amounts of data. A popular open-source framework, Hadoop, enables the distributed processing and storage of large datasets across multiple computers. Hadoop is scalable from a single to thousands of machines, and this factor ensures the great reliability and availability of the system. MapReduce, a programming model of Hadoop, which divides the big datasets into smaller, manageable tasks, is one of the most essential elements of Hadoop. This model is excellent for handling vast volumes of data quickly and effectively since it allows for parallel processing.

Moodle is one of the most widely used open-source Learning Management Systems (LMS) and is being used by many educational institutions to manage classes, track how students are doing, and keep attendance records. It is highly popular and widespread among many schools and universities. But, due to the expanding amount of data generated from student activities and interactions, traditional methods of managing data in Moodle are not fast or efficient enough to meet performance and scalability requirements. In this situation, the use of Hadoop and MapReduce gives significant advantages for processing and analyzing vast amounts of student data.

The proposed implementation is a student attendance system that is developed by integrating Hadoop's MapReduce framework to manage the massive datasets produced from Moodle's attendance records. The system exports Moodle SQL data onto the Hadoop Distributed File System (HDFS) and uses Hadoop MapReduce to analyze the student attendance. Student IDs and attendance data are mapped using MapReduce scripts, which is then reduced to generate reports with context. These reports include attendance summaries and the highest maximum percentage of attendance possible. This system enhances the scalability and efficiency of the attendance management process and allows the schools and universities to handle large data sets with the least or without performance bottlenecks and delays.

The paper structure is outlined as follows: The upcoming Section 2 covers previous related studies. Next, Section 3 presents the background theory of Hadoop, MapReduce, and Moodle. Section 4 provides the implementation of the system while section 5 focuses on report generation and data analysis of the system, and then the last Section 6 concludes the system.

II. Related Works

Nikolovska et al. [1] describes an architecture for performing analysis of Moodle data using Hadoop in the cloud. The authors explain how data is collected from the Moodle database and data logs, containing both structured and unstructured data. The data is then sent onto the HDFS which is spread across multiple cloud nodes. The system uses Hadoop Hive for querying the data. Hive then uses the Hive Execution Engine to break down the HiveQL queries into MapReduce tasks. The paper emphasizes using "Hadoop in the cloud". In this paradigm, all Hadoop processes run on infrastructure managed by a cloud provider offering benefits of "flexibility, speed of change, lower risk, worldwide

979-8-3315-1004-6/24/$31.00 ©2024 IEEE

availability, cloud provider features, capacity, efficiency and reduced cost" [1, p. 5]. The system presented in this paper, in contrast, consists of a ready-to-use interface that allows running queries on a live Moodle site and database (albeit a small delay due to periodic export). The Hadoop cluster is also able to run in the cloud.

The 2023 conference paper studied by Irena Valova at University of Ruse [2] also explores the use of Hadoop for enhancing the educational process while emphasizing the use of Hadoop with HUE (Hadoop User Experience) user interface to provide students with hands-on experience in big data processing and analysis, particularly for enhancing their practical skills. The paper highlighted the importance of Hadoop's scalability and distributed processing capabilities in managing large datasets within educational environments. It showcased how Hadoop can not only improve the educational infrastructure but also prepare students for future careers in data-driven industries.

The author [3] presented the architecture of Apache Hadoop and broke down its components bit by bit, highlighting their importance in handling large volumes of data within a distributed system. The Hadoop Distributed File System (HDFS) facilitated the storage of massive data blocks across a distributed network. The Hadoop framework handled fsImage and edit files, ensuring that data remains available and intact. This paper also presented examples of Hadoop applications, including those in weather monitoring and bioinformatics analysis.

A researcher from the University of Mumbai, Rohan Gupta [4] developed an Attendance Management System which uses Hadoop to manage the large-scale attendance data within and across colleges. The system benefited from Hadoop's MapReduce algorithm for efficient data processing and storage in the Hadoop Distributed File System (HDFS). This method settled the challenges of handling Big Data in educational institutions, especially with the increasing amount and volume of attendance records.

In research brought out in 2019, the authors [5] developed and implemented a barcode-based student attendance system. It was designed to enhance the accuracy and efficiency of attendance tracking by using barcodes to record student attendance in real-time, trying to replace traditional paper-based methods. They built the system by using Unified Modeling Language (UML), Microsoft Access 2007, and ASP.NET. The system can also identify students who frequently miss classes and send them automated email warning messages.

In 2021, William Villegas-Ch and colleagues [6] proposed a Hadoop-based framework to analyze student learning because of the change from traditional learning to remote learning due to the COVID – 19 pandemics. The framework mainly used Hadoop's MapReduce for processing large datasets and Apache Impala for data analysis purposes. It mainly focused on comparing student performance in face-to-face versus remote learning environments. This study emphasized how changes in the education environment affect academic outcomes.

III. Methodology

A. Hadoop

Hadoop is a Java-based open-source framework built to store and process large datasets across clusters of computers. It works by decomposing large tasks into smaller ones that can be processed in parallel on different machines. Hadoop is designed to scale from a single computer to thousands of connected machines, with each offering local storage and computation.

The core architecture of Hadoop consists of a central NameNode and multiple DataNodes. The NameNode manages the cluster's metadata, like file names, paths, data block locations, and replication details. It assigns tasks to the DataNodes but doesn't store any actual data itself. DataNodes store the actual data and perform operations like creating, replicating, and deleting data blocks. If a DataNode fails, the system continues to function because the data is replicated across other nodes. However, the failure of the NameNode can disrupt the entire cluster.

Hadoop consists of 4 main components, Hadoop Distributed File System (HDFS), Yet Another Resource Negotiator (YARN), MapReduce, and Hadoop Common. In addition to these core components, the broader Hadoop ecosystem includes tools like Apache Pig, a scripting platform for writing MapReduce programs, Apache Hive for querying large datasets using SQL-like syntax, Apache HBase for NoSQL storage, Apache Spark for faster in-memory processing, Presto for distributed SQL querying, and Apache Zeppelin for interactive data analysis. These tools extend Hadoop's capabilities for big data storage, analysis, and processing.

B. MapReduce

MapReduce is a Java-based distributed framework in the Apache Hadoop ecosystem that simplifies distributed programming. It involves two main steps: Map and Reduce. In the Map step, data is divided into smaller pieces for parallel processing, where transformation logic is applied to each part. In the Reduce step, the results from the Map phase are aggregated [7]. Typically, MapReduce uses Hadoop's Distributed File System (HDFS) for input and output, but technologies like Sqoop can be used to connect to relational databases.

MapReduce was first developed at Google in 2004 by Jeffery Dean and Sanjay Ghemawat [8]. Their paper, "MapReduce: Simplified Data Processing on Large Clusters," was inspired by the map and reduce functions in functional programming. Google originally used it with the Google File System (GFS). However, by 2014, Google had transitioned away from MapReduce as its main data processing model. In the Hadoop ecosystem, newer systems like Hive and Pig now offer SQL-like querying alongside jobs written in the MapReduce model.

A MapReduce system has 3 steps [9]:

- **Map:** Input data is divided into smaller blocks. Hadoop determines how many mappers to use based on the data size and system resources. Each block is processed by a mapper, which applies the map function and temporarily stores the output. The master node ensures that duplicated data is processed only once.

- **Shuffle, combine, and partition:** Mapper output is reorganized by key, ensuring data with the same key is sent to the same node. An optional combiner can reduce the data before shuffling. The partition phase, which is mandatory, assigns the data to specific reducers.
- **Reduce:** Reducers cannot start until all mappers have finished. The reducers process <key, value> pairs, aggregating the values for each key. While the map function is required for sorting and filtering, the reduce function is optional.

MapReduce consists of two main tasks, divided into phases. The first phase uses the Map function, while the second phase utilizes the Reduce function. Figure 1 illustrates the MapReduce architecture [10].

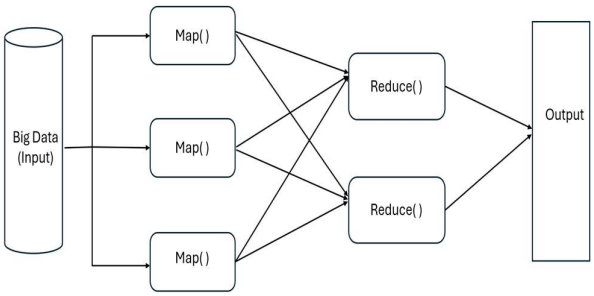


Fig. 1. MapReduce Architecture

MapReduce has several advantages. Its primary strength is **scalability**. It divides complex problems into smaller tasks, processes them across multiple nodes, and then combines the results, allowing for efficient parallel processing of large datasets.

MapReduce is also **fault-tolerant**. Input data is replicated across nodes, so if one node fails, other nodes can process the replicated data without interrupting the workflow.

Lastly, MapReduce is **cost-effective**, as it makes efficient use of system resources through parallel processing. Its scalability further enhances its cost-effectiveness, allowing organizations to process growing data volumes without needing major infrastructure investments [8].

C. Moodle

Nowadays, the number of internet users and the significance of online tools in education are rapidly growing. Many students use the Internet as an essential tool for their studies, including accessing online libraries and participating in online conferences. The Internet is being used as the primary technical tool in order to search, collect, use and summarize the necessary information necessary for knowledge acquisition and to provide efficient knowledge transfer between teachers and students at any university. The application of E-learning education methods proves to be highly beneficial as one of the solutions in higher education levels.

The word "E-learning" refers to the use of digital platforms and tools to assist learning outside of the traditional classroom environment and the most important feature of E-learning is that it takes place in the network environment. That means that the client's (student's) computer is in continuous communication with the main server. Also, the study materials of the class are accessible via an Internet browser on the computer or laptop. This approach offers adaptability, flexibility, and the capability to meet various learning preferences and requirements, making it increasingly popular among the educational centers (universities, institutes, colleges, and schools).

Learning Management System (LMS) serves as the foundation of E-learning, offering a systematic platform where educators can create, distribute, and assess learning materials, while students can access these resources and participate in various learning activities such as submitting assignments, answering quizzes and checking for attendance. One of the most commonly used platforms in E-learning is Moodle (Modular Object-Oriented Dynamic Learning Environment). It is an open-source LMS that offers educators and learners a powerful, robust and engaging online learning environment. It has proven to be an immensely popular and important tool in distance education. Moodle also supports a wide range of features, such as course management, content distribution, and assessment tools, all of which can be personalized to meet the unique requirements of different educational environments. Some evaluations have also indicated that Moodle is one of the top-rated programs when compared to other open-source course management systems [11].

IV. System Implementation

The system architecture is implemented as follows:

A. Components

The system consists of the following components, as shown in Figure 2:

1) Moodle web server: The Moodle Web Server is a typical Moodle installation with the Moodle Attendance plugin added.

2) Report generator Moodle plugin: This is a Moodle Block plugin that adds a block to the dashboard of the Moodle user, e.g. teacher. The block consists of a user interface with drop down menus and forms that allows users to select the kind of reports or graphs they would like to generate. The user specifies fields such as the type of graph, the label for the graph, the time period for the report, etc. This block also includes a list of previously generated reports for the user to view. It also allows the user to check the status of currently running report generation jobs. Clicking on a report directs the user to the generated report file.

3) Hadoop cluster: The Hadoop cluster is a typical Hadoop installation across multiple machines consisting of the HDFS, YARN and MapReduce services. YARN is responsible for receiving job requests, scheduling resources and executing the jobs.

4) Hadoop Streaming: Mapper script: This script performs the Map operation of the MapReduce paradigm in Hadoop. It is a Python script run with the Hadoop Streaming framework. It maps the attendance data into keys, such as student ids or course ids, and values, such as attendance marks.

5) Hadoop Streaming: Reducer script: This script performs the Reduce operation of the MapReduce paradigm

in Hadoop. It is a Python script run with the Hadoop Streaming framework. It reduces previously mapped values by summarizing them, e.g. calculating maximum possible attendance by summing attendance marks.

6) Generated Reports for Analysis: This is the report file that is generated with the Report Generator Plugin. The file consists of one or more graphs and information depending on the type of report selected.

7) Periodic data export script: This script periodically exports relevant data from Moodle's SQL Database to the Hadoop Distributed File System (HDFS) for processing. Details about this process are explained below.

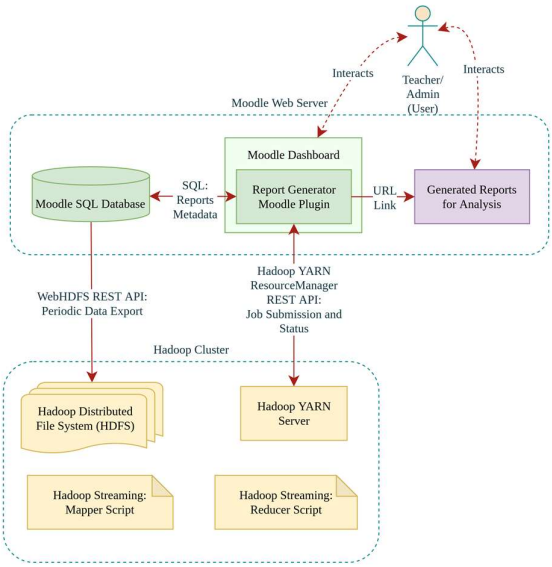


Fig. 2. Proposed System Architecture

B. *Exporting data to the HDFS*

Moodle stores all its data in a local SQL database. To take advantage of HDFS, the data from the SQL database needs to be transferred to HDFS first. The following steps are taken to perform this.

An SQL query is first executed that creates a table from the related attendance data, namely the fields of Student ID, Student Name, Course ID, Course Name, Class Session Date, and Attendance Status. This data is acquired from joining and filtering the Moodle built-in tables: mdl_attendance_log, mdl_attendance_sessions, mdl_attendance_statuses, mdl_attendance, mdl_course, and mdl_user. This table is then exported to a CSV file to a temporary directory, resulting in a final log file containing every attendance taken.

Finally, the CSV file is copied to the HDFS by using the WebHDFS REST API. This is done using the 'curl' linux command to send a PUT request to the API with the data from the CSV file. At this point, the attendance data is on the HDFS and is ready to be processed by Hadoop.

All the above is automated using a Linux Bash script. This script is run periodically to keep the Moodle data ready on the HDFS for processing and analysis.

C. *Usage Workflow*

The user wanting to perform analysis of students' engagement, e.g. teacher or course administrator, first login to their Moodle account. On the dashboard, the Report generator plugin displays a block with the UI for generating reports. The teacher can select the type of report they want depending on what they would like to analyze, such as the summary of maximum possible attendances for a course or attendances for courses that a particular student attends. Students to be included in the report can be selected, as well as the courses of interest. The time period of sessions to be used in calculation also needs to be specified, e.g. one semester's duration. The teacher can then generate the report, which is illustrated in Figure 3.

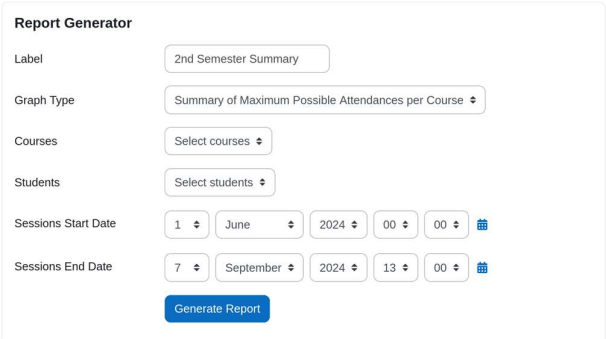


Fig. 3. Report generator plugin UI: Generate new report

Once the report is generated, the teacher can view the report along with all other previously generated reports in the Generated Reports section, as shown in Figure 4. Each report generation submission (Hadoop job) is associated with a separate, unique link. The status column indicates the Hadoop processing status of the report. Once generation is complete, the teacher visits this link to view the report, at which point they can easily perform analysis.

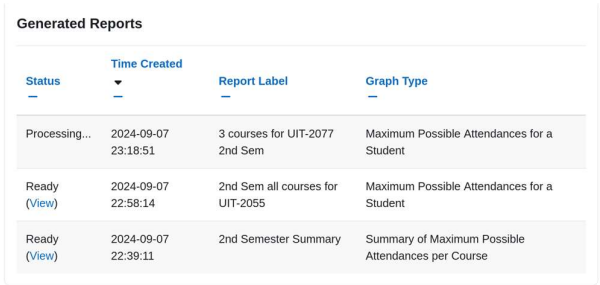


Fig. 4. Report generator plugin UI: List generated reports

V. Report Generation and Analysis

To demonstrate report generation and data analysis, we will use two datasets. The first dataset contains actual student attendance data from the Moodle system of the University of Information Technology (UIT) was utilized. Attendance logs for one semester from a class was requested. The class consists of 6 courses (subjects) and 39 students. The dataset comprises ~5000 entries. The second, larger, dataset is a generated dataset that simulates the attendance of students at UIT from its establishment in 2012 to 2024, given that a batch of 200 students are admitted every year. The dataset comprises ~3.6 million entries.

The system can generate a variety of report formats depending on how they can assist teachers in gaining insight about students' engagement and performance.

One type of report supported in this implementation is a pie chart summarizing the maximum possible attendances of the students of one or more courses.

The maximum possible attendance for a student at a point in time is the theoretical highest attendance percentage that a student can get given that they attend all other remaining sessions without fail. In other words, a student cannot get an attendance percentage higher than their current maximum possible attendance, however diligent they may be in the future. This is a common measure used in courses to determine whether a student has adequate class attendance and can sit for an exam. It is therefore crucial for teachers to monitor the maximum possible attendances of students in the courses they teach.

The system can assist teachers in this matter by generating a pie chart with each "pie" being the number of students within a certain range of attendance of a course, as shown in Figure 5, generated from the first dataset. One pie might show the number of students within 95% to 100% attendance, another pie from 90% to 95% and so on. The graph is colored such that the color progresses from green to red as the attendance rate falls. This allows the teacher to immediately notice which courses have a low attendance rate.

This kind of report grants the teacher the ability to easily know how many students are at or nearing the attendance limit and take appropriate action such as issuing warnings and discussing their situation. Useful insights can be obtained from the graphs. For example, CST-4305 has a relatively high attendance rate. This can mean that students show more interest in learning the subject, or students are more motivated to attend the course regularly.

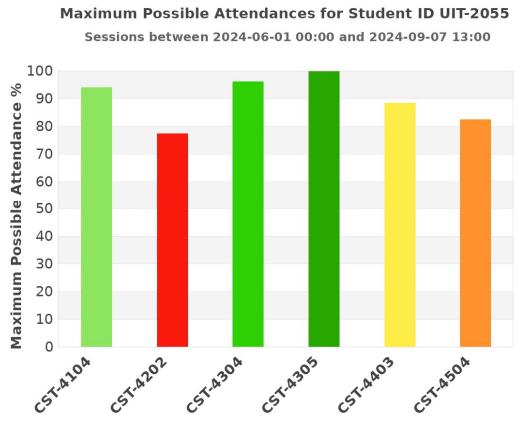


Fig. 5. Bar graph showing maximum possible attendances for the 6 courses that one student is enrolled in

Another type of report supported by the implementation is a bar graph showing maximum possible attendances for courses that a particular student is enrolled in. A bar represents the maximum possible attendance percentage of a course for the student. The report can be configured to display bars for multiple courses of interest. The same color progression from green to red is used for visual aid, starting from red for less than 80% attendance to dark green for 100% attendance, as shown in Figure 6, also generated from the first dataset.

This type of report allows analysis of a particular student's engagement. This can be used in conjunction with other observed information about the student, such as test grades and assignment marks, to evaluate the student's academic performance. The teacher can immediately see which courses have low attendance percentages and can issue a timely warning to or discuss the situation with the student.

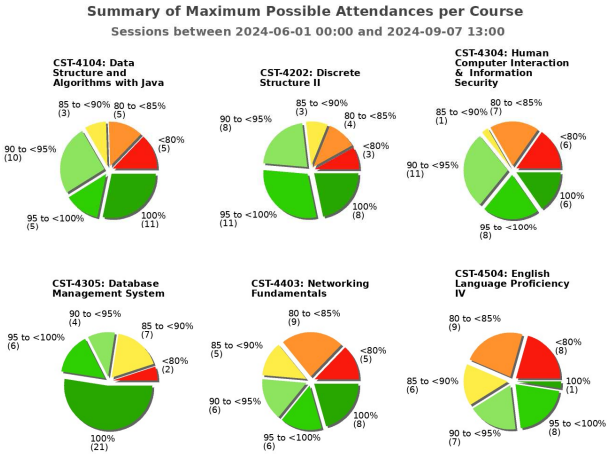


Fig. 6. Pie chart summarizing the maximum possible attendances of all students for 6 courses

A summary of Maximum Possible Attendances report can also be generated for a large aggregation of students, such as for all the classes of the semester. Figure 7 showcases Maximum Possible Attendances for all 200 students of Semester 4. The report below was created using the second dataset, which contains a larger dataset of generated attendance logs.

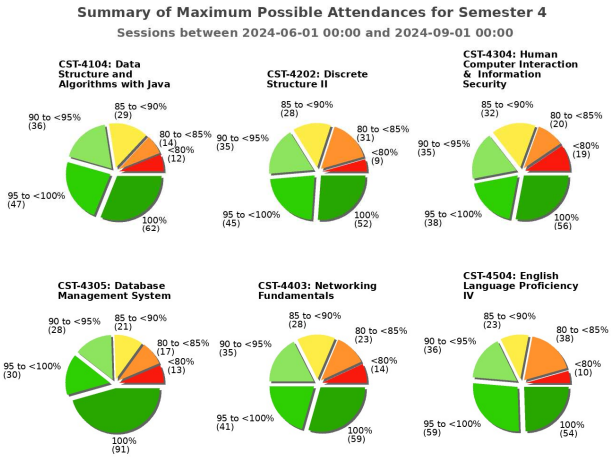


Fig. 7. Pie chart summarizing the maximum possible attendances of all classes and courses of a semester

Given that there are:

- 2 semesters per year

- 15 weeks per semester
- 12 sessions per week (6 courses 2 sessions each)

There are 2 x 15 x 12 = 360 attendance log entries for 1 student for a year. Approximately 200 students get admitted to a university such as UIT every year, for a total 360 x 200 = 72,000 entries. For a 12-year time frame, this attendance data accumulates with every batch admitted and if the attendance history of graduated students is to be kept for historical analysis purposes, this data can reach a substantial amount of 3,600,000 entries.

If each entry takes 100 bytes of disk space, the final attendance log data will be around 343.322 MB. If we consider the attendance of future students along the life of the university, and the growth of the university itself, the data can double or even triple, stepping into the GBs range.

In such a situation with large quantities of data, the distributed nature of Hadoop MapReduce allows for efficient processing across multiple nodes. Furthermore, as the data grows, Hadoop can scale horizontally by adding more machines.

The execution times to generate the 3 reports (Figure 5, 6, 7) are shown in Figure 8. We can see that while the last report uses a dataset that is much larger (~5000 vs ~3.6 million log entries), the execution time increases only by around twice and thus showcases the scalability of MapReduce with increasing amounts of data.

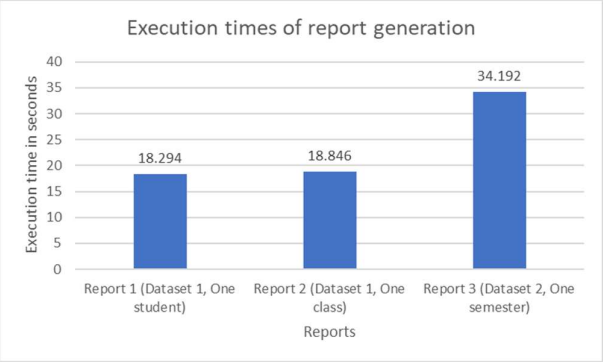


Fig. 8. Execution times of generating the 3 reports using the Hadoop MapReduce System

VI. Conclusion

The integration of Hadoop MapReduce into Moodle demonstrates a powerful approach to handling large-scale academic data, particularly student attendance. By utilizing Hadoop's distributed processing, the system addresses the limitations of traditional data management in Moodle, offering enhanced scalability and faster processing times. The resulting attendance reports provide valuable insights for educators, allowing them to track student engagement with greater precision and verify their exam eligibility. This work not only improves data handling within learning management systems but also paves the way for future advancements in educational data analytics, such as real-time data synchronization and cloud-based deployment, could extend the system's capabilities, making it an even more valuable tool for modern educational institutions.

Future developments could enhance the current system by integrating additional types of student data, such as assignment submissions, exam results, and participation in online activities enabling more comprehensive learner analytics and deeper insights into student performance and behavior patterns. Analyzing logs of student activity from Moodle could provide a detailed view of student interactions, helping to identify learning trends or engagement issues early on.

Another area for improvement is enhancing real-time data processing by employing tools like Apache Sqoop for seamless data synchronization between SQL databases and Hadoop's HDFS, eliminating the need for periodic data exports. Expanding the architecture to include cloud-based Hadoop clusters could improve scalability and availability, making the system more adaptable to growing data volumes and institutional needs. Finally, creating an interactive and user-friendly dashboard for educators to customize and visualize various data insights would significantly enhance the system's practical value.

References

[1] A. Nikolovska, A. Velinov, S. Spasov, and Z. Zdravev, "Framework for Big Data analytics of Moodle data using Hadoop in the cloud," 2018.

[2] I. Valova, "Using Big Data and Hadoop in the Student Learning Process - Enhancing the Educational Process through Real Experience," International Conference Automatics and Informatics 2023, Varna, Bulgaria, Oct. 5-7, 2023.

[3] P. R. Giri and G. Sharma, "Apache Hadoop Architecture, Applications, and Hadoop Distributed File System," Semiconductor Science and Information Devices, vol. 4, issue 1, April 2022.

[4] R. Gupta, "Implementation of attendance management system using Hadoop," 2019.

[5] R. Al Sheikh et al, "Developing and implementing a barcode based student attendance system," International Research Journal of Engineering and Technology (IRJET), vol. 6, January 2019.

[6] W. Villegas-Ch, M. Roman-Cañizares, S. Sánchez-Viteri, J. García-Ortiz, and W. Gaibor-Naranjo, "Analysis of the state of learning in university students with the use of a Hadoop framework," Future Internet, vol. 13, pp. 1-25, 2021.

[7] A. Kumar, "Introduction to MapReduce in Hadoop", Medium, https://medium.com/@ashwin_kumar_/introduction-to-mapreduce-in-hadoop-f8d72a345619/ (accessed Sep. 11, 2024).

[8] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," 6th Symp. Oper. Syst. Des. Impl., San Francisco, CA, USA, Dec. 6-8, 2004.

[9] "Understanding MapReduce," Databricks. https://www.databricks.com/glossary/mapreduce/ (accessed Sep. 11, 2024).

[10] "Map Reduce in Hadoop," Geeks for Geeks, https://www.geeksforgeeks.org/map-reduce-in-hadoop/ (accessed Sep. 11, 2024).

[11] N. Cavus and T. Zabadi, "A comparison of open source learning management systems," Procedia - Social and Behavioral Sciences, vol. 143, pp. 521-526, August 2014.