International Journal of Scientific Engineering and Science
Volume 4, Issue 5, pp. 32-38, 2020. ISSN (Online): 2456-7361

Common Distributed Data Storage for Higher Education Management Information System in Afghanistan

Mohammad Zia Sana
Faculty of Computer Science, Kabul Polytechnic University, Kabul, Afghanistan

Abstract— *In today's organizations, a management information system (MIS) plays significant roles; it supports the decision-making process and controls management which impacts the organization's functions, performance, and productivity. Therefore, one of the primary goals of the National Higher Education Strategic Plan (NHESP) of the Ministry of Higher Education of Afghanistan was the development of the Higher Education Management Information System (HEMIS). This system is active now, but its implemented partially and not used by all universities. This paper aims to analyze the HEMIS and suggest appropriate solutions to solve its main issue(s).*

One of the main issues is the centralized structure of the system which causes a single point of failure. To this end, we propose the distributed database solution which provides a common database backend for the HEMIS and distributes it over the network to all educational institutions as well as MoHE. In this solution, some constraints of the current situation such as budget limitations and internet connection constraints are also considered. The university users can use the system locally and synchronize their changes at the central location (MoHE) once the connection is available. Besides, the distributed database eliminates the process bottleneck by having many replicas and more than one system can process the user's request. Hence, the performance of the system will be higher, can handle many requests, and store more data.

Keywords— *Distributed Database System, Higher Education Management Information System, Data Synchronization, Distributed Database Storage, Data Replication.*

I. INTRODUCTION

The use of information technology has improved the educational management operations due to its efficiency and effectiveness and most of the universities now use a computer system (MIS) for efficient administration and proper management.

As the higher education sector grew in Afghanistan and the number of students increased in educational institutions; the government realized the importance of accurate data for decision making. So, the Ministry of Higher Education of Afghanistan (MoHE) decided to develop a HEMIS for itself and other educational institutions inside the country. Hence, in 2008 the MoHE developed the NHESP that developing a HEMIS was one of its significant parts.

In 2012, this system was implemented to ensure accurate data collection and also automatic management of all information processes inside the MoHE and universities. Besides collecting data, providing reports in a meaningful manner and within a reasonable timeframe for all stakeholders is a major objective of this system. This system was designed and implemented in a centralized approach, and it is located in MoHE and university users access the system through the Internet. Therefore, it acts as the campus management system for Afghan universities as well.

The system has not been fully implemented in universities and so did not have the expected impacts on universities as well as the MoHE operations and system automation. Due to the implementation of the current structure, the HEMIS is facing a single point of failure issue, and bottlenecks can occur as a result of high traffic which suffers the system's performance. Hence, the centralized structure of the system is a major cause of the mentioned problems since all data is stored in one place.

II. LITERATURE REVIEW

The distributed database concept is an alternative design that can solve problems exist with the centralized database system like a single point of failure and performance bottleneck. A distributed database is a group of numerous databases that have a logical relationship between and deployed over a network [1]. After 1994, the usage of the distributed database system is increasing and most organizations move toward distributed database management for managing and administrating the system [2]. Research accomplished on distributed databases in 1991, predicted a massive shift from traditional structures to distributed databases due to organizational requirements to manage huge amounts of data [3]. Many applications will be dispersed and therefore the database will also be distributed [4]. Hence, implementing a distributed database has become common in enterprises especially when it is getting larger. Different business conditions encourage the utilization of the distributed database, for instance, distribution and autonomy of business unit, information sharing, and availability, data communication cost and reliability, database fast recovery, organization growth and high performance [5]. Therefore, different options can be used for distributing the database across the network like data replication, horizontal partitioning, vertical partitioning and combination of theses.

Database replication is the procedure of establishing and maintaining numerous instances of the same database and the process for sharing data or changes in databases among different locations while not having to copy the whole database [5]. It enables us to copy a database or even a file

Volume 4, Issue 5, pp. 32-38, 2020. ISSN (Online): 2456-7361

from the master (primary node) database management system to its accurate slave node. A node can either replicate a portion or all the data. Replication may permit an organization to move a database of a centralized mainframe onto less high expensive, departmental, or near to end-users [6]. Moreover, it is used to ensure the consistency among the nodes, improves the accessibility, increased availability, reliability, fault tolerance as well as the performance [7].

However, currently, NoSQL databases can operate as a distributed database and it was designed to scale out horizontally and continue running on various servers that work together [8]. On the other hand, HEMIS was developed using relational databases so the relational database system should be distributed and replication is used to distribute databases over the universities. For the implementation of a heterogeneous distributed database system in the educational sector, hospitals, and organizations replication, partitioning of database synchronization techniques is used [9 & 15].

Some free and open-source tools can be used to manipulate replication and database synchronization across the database systems such as SymmetricDS and Galera Cluster. SymmetricDS is a web-based, open-source, database-independent, data synchronization software that uses asynchronous data replication for both file and database synchronization with the support for multi-master replication, filtered synchronization, and transformation across the network in a heterogeneous environment [16].

A conclusion section is not required. Although a conclusion may review the main points of the paper, do not replicate the abstract as the conclusion. A conclusion might elaborate on the importance of the work or suggest applications and extensions.

III. METHODOLOGY

For conducting this research and implementing the system we use the waterfall methodology. First of all, we gather the requirements, then design the system, implement the system, and finally, we will test and train the users.

Different research methods will be used to collect information about the data that each university will share with MoHE, including interviews and meetings with the university's academic and student affairs as well as IT department staff and HEMIS administrators. We would also analyze the existing documents to find and classify the shared and unique data of each university.

IV. HEMIS STURCTURAL CHALLANGES

HEMIS is a centralized web-based system and consists of a database system which is hosted on a server in MoHE. This is so that all stakeholders (Kabul and provincial universities and MoHE users) access the system using the Internet connection. In this case, all data like student files, academic affairs, and administrative employees are stored in one central location. To use the system, each user must have a credential to access the system through the Internet which is provided by MoHE for all universities.

At a later date, when the data of a huge number of universities and MoHE itself are stored in the system, there will be a large amount of data in the system. What happens if all data is deleted by the user's fault or a natural disaster, then a significant number of data will be lost from all the higher education institutions including MoHE. Additionally, the ICT infrastructure is limited in the current situation of our country that if an institution does not have an Internet connection for a long time, then it is unable to use the system and even can't access their data. No one can access if the system fails. The problems outlined above will occur in the current centralized structure of the system.

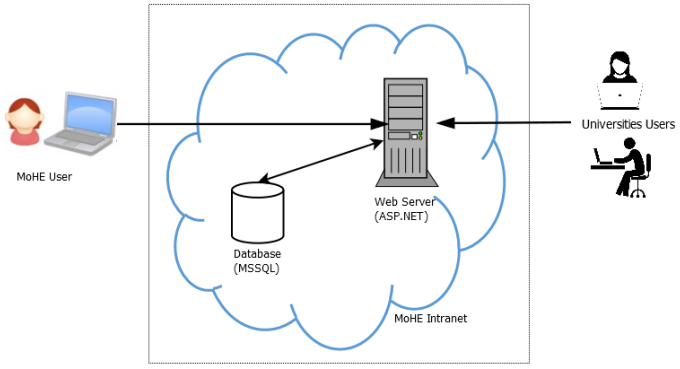

Figure 1. Centralized Structure of HEMIS

Figure 1 illustrates the system structure, where it is deployed in MoHE, and all data is stored in one central location. The MoHE users can access the system through the MoHE local network, while the university users should access the system using the Internet. The following outlines the major challenges of the system regarding its centralized structure:

The first problem is breaking the MoHE rules and regulations: implementing the system in a centralized method breaks the MoHE rules where all university data stores just in a single system and each client access its related data.

The MoHE's rules state that some data like student's information (profile, study records, and marks) should have three copies and each copy retain in three different locations faculty, university, and MoHE for some security reasons. If these data are located in one place, it will be easy to obtain and change, but this is a violation of data security. To guarantee data security, data should be retained in three different locations. Therefore, if someone changes data, it is easy to distinguish the incorrect data and avoid using false information and it is hard to change data simultaneously in all three locations.

The current structure of the HEMIS provides an opportunity for destructive people to alter data because it is stored just in one central location. Despite this, the current system has a user trace mechanism and logs every transaction. However, in a computer system, it is also possible to remove the logs then it will be difficult to track the data changes.

The second problem of the centralized system is the lack of ICT infrastructure in the higher education institutions: most institutions of higher education have basic ICT resources, such as computers, campus networking, and Internet connection.

Therefore, they cannot use the system for their daily work in the offices unless they have a stable Internet connection alongside other IT resources mentioned shortly. However, some institutions equipped with the necessary ICT infrastructure such as, networking and computer labs, yet they have a very limited and unstable Internet connection, which is not sufficient for a system that needs to be frequently accessed all the time by many users. In a nutshell, the system usage is dependent on the Internet connection and is not used locally within a university.

The third problem is the processing overhead of the system: because one single system is carrying out all user's requests, if a user requests a query it passes a long way to get the result. For example, when a user from Herat University attempts to access information from Kabul University, it requires a round trip to Kabul every time he/she requests, so the system will be comparatively slow. Accessing such a system with a low bandwidth Internet makes the overall performance extremely poor. Currently, HEMIS might not face this problem, as only administration staff access it. When some student-related services are integrated with the system and access given to the students too, the number of users will reach hundreds or thousands and at that time the system will face significant problems.

The significant challenge is the single point of failure of the system: an entire centralized system depends on a single CPU or a single machine. When the CPU fails, the whole system is inoperable. As mentioned above, HEMIS is a centralized system that has a single server and a single database along with a single switch that is connected to the Internet, where all these devices are installed in the same room, if the webserver or the database server machine fails, the whole system will be down. Considering the current situation in Afghanistan, the chances of hardware failure are high due to unstable electricity voltage, insufficient cooling systems, insecurity, and insurgents' blasting. In the case of fire or any other natural disaster, the whole system and data will be lost because everything is in the same location including the backup. Therefore, with the current situation when the system is down, no one has access, even the universities.

V. DISTRIBUTED DATABASE FOR HEMIS

Organizations will usually opt for one of the following methods to design and implement their system accordingly, centralized, or distributed. Choosing between these methods depends on different factors like the organization size, locations, scalability, and infrastructures.

A distributed database is a single database that is shared physically over computers in different locations that are connected by a data communication network [5]. The centralized structure of the HEMIS has many problems, such as processing overhead, single point of failure, performance bottleneck, breaking the rules of MoHE, and requiring users always to have an Internet connection. The distribution mechanism will solve most of the current challenges of the system and attempts to use HEMIS as a campus management system for all educational institutions.

The distributed database systems are perfectly adapted to the current situation in Afghanistan while the Internet connection becomes a challenge for most of the universities. According to our research and USWDP report, the lack of Internet or having a slow Internet connection is one of the reasons which HEMIS is not fully implemented, so an alternative for educational institutions is to have the system and utilize it locally. Therefore, by distributing HEMIS to the universities; a copy of the system will be installed on the university's local network, and it is accessible inside the university without having an Internet connection, aside from when transferring the changes to the central system.

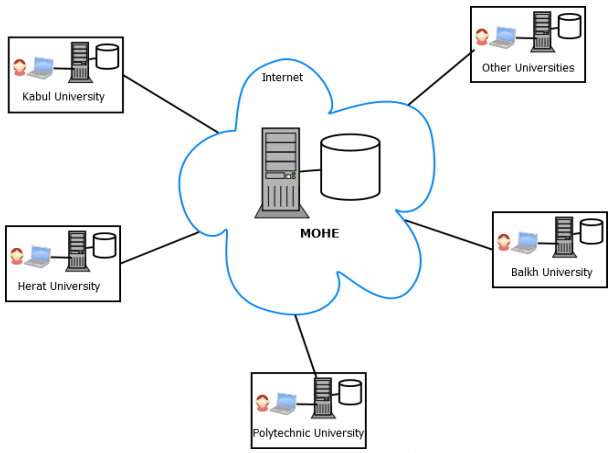

Figure 2. Distributed Structure of HEMIS

Figure 2 shows the distributed structure of the system whereas every educational institute obtains the system and uses it locally. Also, it has a connection with MoHE to synchronize data. The distributed method does not have a single point of failure issue since every node has access to its data and a node failure does not affect the other parts of the system. For instance, when the MoHE system fails, universities use their system locally, but data changes will not transfer unless the MoHE system is up again.

The centralized system has low performance due to processing overhead, whereas the distributed system minimizes the processing overhead [2]. Each university's user functions on a local basis, and a university's user requests will not transmit to a remote server.

Therefore, the processing overhead is reduced and the single point of failure will no more exist because each university has its system, and failure in MoHE does not affect data loss in universities. We also ensure high data availability and removes the process bottleneck for both MoHE and universities by introducing many replicas that are copying data on multiple servers. With having more replicas more than one system can process the user's request, hence the system performance increased and can handle many requests at a time.

A. *Data Synchronization Architecture*

According to the implementation plan, the system should be used by both MoHE and university users. So in a distributed database mechanism the database dispersed over

multiple nodes and these nodes connected through the network. There will be many nodes including MoHE and educational institutions. Each node has a copy of the system based on its functionalities. For example, MoHE needs to have a student module, human resource module, scholarship module, but university nodes do not require the scholarship module because it is not related to them.

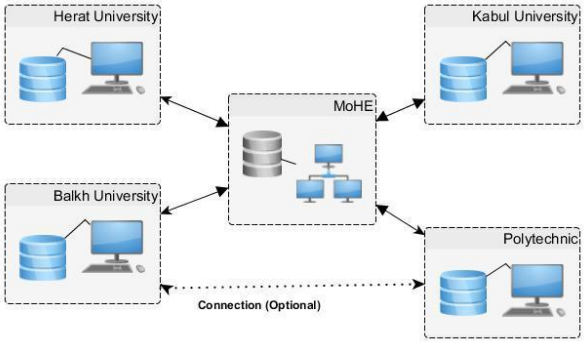

Figure 3. Nodes relationship

To synchronize data, the relationship between the nodes should be considered. Figure 3, illustrates data synchronization architecture designed for the system. There is one central node (MoHE), and the relationship between the node is the same as the parent-child. Every university has a relation with MoHE, but MoHE may or may not have a connection with the universities. The flow of data can be accomplished based on the real system that indicates what data universities transfer to MoHE and vice versa. According to the proposed architecture, data can also flow between universities when it is required, but it is optional. For instance, a student wants to study his master's degree at Kabul University while he has studied his bachelor's degree at Herat University.

There are three approaches we propose for transferring the data between universities. Firstly, universities can establish relationships with each other like a mesh topology in the network. However, universities are all connected, but this is a complicated procedure at the same time the system administrator will become abstruse and there may be many links that have been formed but not used.

The second approach is to establish the data synchronization links for a node in the time it requires. This method is better than the first one, but requires complex administration and also depends on the system implementation. If a central system administration controls and manages the system then the links can be built simply but if the administration of the nodes is local then an agreement between both nodes is necessary before setting up the link, and both should protocol with each other in the early stages.

In the third approach, we can transfer data between universities through the central node. As it is evident from figure 3 each node is connected with the central one and the links between them are in two directions. Thereby, it is easy to use the central node as an intermediate node for transferring data between universities. Therefore, the MoHE node acts as a router in the network that manages and routes the packages.

As far as the implementation and management of centralized data distribution between nodes require less effort and it is not necessary to create additional links, instead, the links which have already been established are utilized, so this approach is more optimal.

B. Database Synchronization using SymmetricDS

As far as each university has its system then they need to transfer data to the MoHE and vice versa if required. For data synchronization between these systems, several options like MySQL, Galera Cluster which support MySQL and MariaDB and SymmetricDS are there [17]. Among these tools, with considering the Internet connection limitation, we propose the SymmetricDS application.

SymmetricDS is open-source software designed to withstand network outage, work across low-bandwidth connections, and to be scalable to a large number of databases [18]. It is a data synchronization software that uses asynchronous data replication for both file and database synchronization with the support for multi-master replication.

SymmetricDS has been utilized in various methods such as snapshot, log, trigger, and time stamp method for capturing data changes [19]. For this scenario, we opt triggered based and incremental update. Every time data is updated, the system conducts the synchronization with the central server and the last change is transferred instead of transferring the whole database.

It is well adapted to current versions of HEMIS where different database technologies are used, and Internet connection is acknowledged as a problem for most of the universities. Where users can access the system locally, and data will sync when an Internet connection is available. Additionally, the administration of the system can be set up as centralized in MoHE which provides easy management and control with the data flow among the nodes. Since this technology can be installed on the top of the database without a huge change in the system, it is easy to adapt to the existing system.

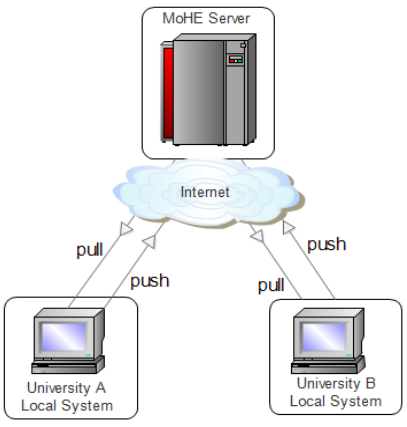

Figure 4. HEMIS Data Synchronization

Figure 4 illustrates the synchronization process between the systems where each university has a local system that can pull and push data to and from the MoHE. In a distributed database system replication is used for copying data over the

nodes so, for achieving data synchronization between universities and MoHE replication mechanisms are considered.

C. Data Replication

Replication is a popular option for distributing data to store a separate copy of the database at multiple sites of a network, and it allows an organization to move a centralized database to location specific and close to users [5]. Furthermore, replication is used to improve local database performance and protect the availability of system because alternate data access options exist. There are three options for replicating data, full replication, partial, and no replication, each of these replication methods is adapted and applied to the different environments [5].

In our solution, we suggest using partial replication because it is suited better for copying some fragments of the database from universities to the MoHE and vice versa. In a partial method, the database is fragmented and multiple copies of the same fragment are stored and maintained in various places [5]. The reliability of this approach is high because new copies of fragments would be still available. All universities would have the same copies of HEMIS with a similar database schema, but the system in MoHE has different modules and functions. There are two types of modules, the unique and shared modules. The unique modules are implemented in one of the two nodes either in MoHE or university like the scholarship module which just belongs to MoHE. The shared modules are used by both the university and MoHE; for instance, the student management system is a shared module, in which both nodes consist and utilize it. The common modules are the only modules that need to share and synchronize data. So we do not need to replicate the entire database, only some parts of the databases are copied among the nodes.

For data synchronization, it is also important to decide when the data changes should apply to a node. There are two options periodic and near real-time replications. Periodic replication usually sends data after a given moment or on regular bases like weekly, monthly, or schedule bases [5]. An application that does not require current data like data warehousing, data mining, or decision support, uses periodic replication, and nodes can perform without updated data for some time. Snapshot is one type of periodic replication. Near real-time replication transmits data at the time data changes occur. This kind of replication is useful in scenarios such the online banking and airline management systems. This methodology utilizes a message broadcasted across the network for each completed transaction, and one way is to use a trigger for generating these messages. When data is updated, triggers stored at each local site, capture, and update the commands on the remote database [5].

In our solution, we use the periodic replication mechanism. Since MoHE uses the data of universities mostly for statistical and decision-making purposes, the periodic approach is preferable whereas the update is conducted at a regular time. Moreover, the Internet connection is not available at some universities, and also they suffer from low bandwidth connections. These are the reasons to use this type of replication since the MoHE node can last for some time without an update from the universities.

In summary, by applying the proposed solution, every educational institute will have the HEMIS locally without always requiring an Internet connection. Furthermore, by helping the SymmetricDS data synchronization will be accomplished automatically.

D. Replication Topologies

The connection between universities and MoHE can be defined based on how they want to contribute. There are different types of replication topologies like master-slaves, multi-master, and multi-source [20]. In our case study, we use multi-source replication for data synchronization, as data is transferred from many sources (universities) to a central location (MoHE).

The university system performs locally and when data changes occur; the node should synchronize the changes with the MoHE. Similarly, the MoHE users use the system when a change is required to transfer to any other node; it runs the synchronization process so that the university will obtain the data.

The multi-source topology can be used while the MoHE acts as a slave and the universities as masters. Data comes together from different sources, and one node is the focal point to combine all the data. Slave node does not indicate that it is just read-only, but users can write in the local nodes and then changes are propagated to the central node

Furthermore, it is important to consider how data should propagate, two different approaches exist for data propagation over the network, synchronous, and asynchronous. As far as, asynchronous tolerate some delay, it works better in our case study where the Internet connection is the concern, and the strength of the Internet connection is still an issue for some universities. The latency does not matter, although the MoHE needs data for decision making and static. For a resolution to the problems with the Internet, they can provide an Internet connection for data synchronization at the time it is necessary or based on a schedule.

E. More Replica: High Availability and Performance

After HEMIS has been distributed to the universities, the data synchronization can be manipulated to transfer data between the nodes. Therefore, the processing overhead will be reduced, and the issue of a single point of failure can be decreased but not eliminated.

Moreover, the performance bottleneck can occur to any node yet, while the number of records is getting larger and larger every day, in particular when a more educational institution or new facilities join the system, more data need to be processed, stored, and maintained. So the bottlenecks can occur as a result of high traffic, and the performance can suffer because more users will utilize the system. To process the user's request with high speed and lower latency, the machine should be upgraded to produce an efficient result. The easy option is to scale the machine vertically, but it is not cost-effective. Instead, we can scale out horizontally.

International Journal of Scientific Engineering and Science
Volume 4, Issue 5, pp. 32-38, 2020. ISSN (Online): 2456-7361

Consequently, a distributed database has the replication feature, and by replicating data to multiple servers, we can achieve greater availability (no point of failure) and higher performance [6]. Each node is a server and has the computing power, so by adding more nodes we extremely have a robust system that can handle more requests. By copying data of a particular node into the multiple nodes, the single point of failure will no more exist and the system's performance will be increased by managing to route the user's requests to specific systems. If one node crashes, another node can operate with no disruption.

For assuring high data availability, the master/slave topology is suitable both for universities and MoHE sides to store a copy of data on multiple servers. The user requests are distributed to a specific node regardless of the request type, where a master node can perform read and write but the slave node is read-only. As far as both MoHE and universities have more write than reading operations and on the other hand configuring all nodes as the master is complex and it causes data synchronization to become more complicated, so we use master-slave to send read requests to the slave and write requests to the master node. Here, we list how to design data distribution on both sides.

- Full replication in the MoHE side to have the same copy of data in master and slave nodes.
- The university systems can be partitioned horizontally by departments or can use full replication.

Since the MoHE system can use full replication, multiple sites have the same copy of the database; so the same database is available at various locations. The reliability is high; failure of any site does not make trouble in accessing data because there are multiple copies of the database, and still data can be obtained from a site that is functioning [5].

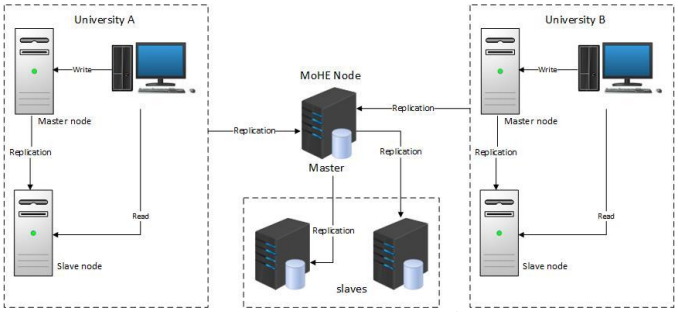

Figure 5. Data Replication Architecture for each Node

Figure 5 illustrates the replication servers on each node in which the slave node replicate from the master node. The master is the primary database system and asynchronously duplicate to the slave node. For assuring data consistency in each local system, data synchronization should run locally as well. When the master node is down the slave node is becoming a master to provide data. The slave node answers the read operations instead of only one primary node for answering every request, so the performance is scaled.

For achieving high data availability and removing the performance bottleneck in our scenario, more slaves' nodes are added on each side thus the central node (MoHE) slave node is configured as master as shown in Figure 5. Therefore, data synchronization happens between master nodes which are called multi-master. As stated previously, concerning the alternative technologies for our solution like MySQL Cluster and Galera Cluster, SymmetricDS is a better choice to achieve asynchronous data changes in a multi-master replication topology.

Even though the master/slave has many advantages, it would also cause the data inconsistency problems when they cannot read the latest write. As a result, we can obtain both data availability and high performance.

F. MySQL Router Load Balancing

However, two copies of data exist on both sides (MoHE and universities), but currently, all load is going to the master server. Now it is a question how to distribute the workload to both servers? MySQL router is a middleware that intelligently routes MySQL connection and transactions to increase performance (load balancing) and uptime (failover) [21]. Therefore, it balances the workload and handles failover, if the primary server fails then it will find the secondary server and establishes the connection to it. The router has both read-only and read-write mode.

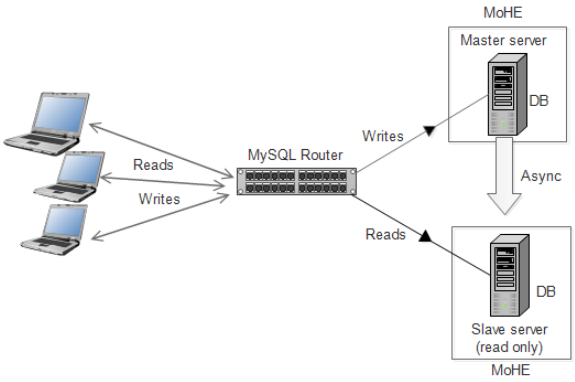

Figure 6. MySql Router Implementation

Figure 6 illustrates the implementation of MySQL router in MoHE. As mentioned above, we suggest a master/slave topology so there is a master node that should handle reads and writes and slave node for read requests. By configuring the router, the workload will be distributed among both servers. Hence, we achieve both high performance and high data availability.

VI. RESULT

After implementing the proposed solution, the system will be locally accessible for each university and the single point of failure will not exist more. Moreover, the performance is enhanced and the bottleneck will not occur. The users will not worry about sharing data between universities and MoHE because the system will take care of it automatically. Furthermore, in this solution, some constraints of the current situation are also considered such as budget limitations and Internet connection constraints. Finally, the HEMIS will have the expected output, and university users, as well as MoHE, will have their system.

VII. CONCLUSION

We analyzed the current status of HEMIS which is implemented partially in some universities. To this aim, we have conducted many interviews with the involved persons and participated in conferences and presentations and also observed the system and examined related documents. The main challenges that system face is the lack of infrastructure, unreliable Internet connection, untrained employee, and the centralized structure of the system. We discovered that the central architecture is the main reason why the system usage in universities has been unsuccessful. To address these challenges, we introduced an alternative approach that uses distributed databases. Using this approach, we will be able to distribute the system over the universities. We introduced a commonly distributed database backend for HEMIS which all universities will have the system locally and synchronizes the database with a central point in MoHE. The users do not need to be connected to the Internet and can use the system offline. We have improved the system availability using the replication mechanism, and the system's scalability and performance enhanced as well. The system can sustain better in disasters, as well as it can scale easily by adding a few configuration files. Similarly, it is extensible and supports the addition of new features. This solution is not just limited to HEMIS, any ministries or organizations that have similar use cases can easily adapt it to its system. Mainly this concept fits well when data sharing between the departments and branches is required especially in a disconnected area where the Internet is limited. Despite the distributed solution has many advantages, implementing the system to educational institutes, requires some resources including the trained individuals, software, and hardware such as a local server, a secure room, system administrator, power and so on. The system uses SymmetricDS to support distributed database architecture and manipulate data synchronization. SymmetricDS is an open-source and database independent application. Besides, it uses asynchronous replication for synchronizing data between nodes and supports areas with a slow Internet connection, and frequent network outage. Furthermore, the MySQL router is proposed for routing the user's requests to the particular server which enhances the performance of the system and provides high availability. Finally, we have shown that we can easily adapt HEMIS with this application as far as it installed on the top of the database system. We also developed a prototype that considered different scenarios for implementing the HEMIS in a distributed way.

REFERENCES

[1] M. T. Ozsu and P. Valduriez, "Distributed database systems: where are we now?," Computer, vol. 24, no. 8, pp. 68–78, Aug. 1991, doi: 10.1109/2.84879.

[2] M. T. Ozsu and P. Valduriez, "Distributed data management: unsolved problems and new issues," Read. Distrib. Comput. Syst., pp. 512–544, 1994.

[3] B. Johnsirani and M. Natarajan, "An Overview of Distributed Database Management System," Int. J. Trend Res. Dev. IJTRD, vol. 2, no. 5, Oct. 2015, Accessed: Aug. 03, 2016. [Online]. Available: http://www.ijtrd.com/papers/IJTRD106.pdf.

[4] İ. Köse, "Distributed Database Security," Data Netw. Secur.-Spring, 2002, Accessed: Aug. 03, 2016. [Online]. Available: http://repository.binus.ac.id/content/M0184/M018456716.pdf.

[5] J. A. Hoffer, R. Venkataraman, and H. Topi, Modern Database Management, 12 edition. Boston: Pearson, 2015.

[6] A. Runceanu, M. Popescu, and M. Runceanu, "Techniques For Data Replication On Distributed Databases," Nov. 2008, Accessed: Sep. 13, 2016. [Online]. Available: http://www.academia.edu/download/30904010/52_Adrian_Runceanu.pdf.

[7] S. Goel and R. Buyya, "Data replication strategies in wide area distributed systems," Enterp. Serv. Comput. Concept Deploy., vol. 17, 2006, Accessed: Nov. 13, 2016. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.445.9055&rep=rep1&type=pdf.

[8] M. Allen, "Relational Databases Are Not Designed For Scale," MarkLogic, Nov. 09, 2015. http://www.marklogic.com/blog/relational-databases-scale/ (accessed Jun. 30, 2016).

[9] M. Faiz and U. Shanker, "Data synchronization in distributed client-server applications," in Engineering and Technology (ICETECH), 2016 IEEE International Conference on, 2016, pp. 611–616, Accessed: Nov. 14, 2016. [Online]. Available: http://ieeexplore.ieee.org/abstract/document/7569323/.

[10] W. E. Hammond, M. J. Straube, and W. W. Stead, "The synchronization of distributed databases," in Proceedings of the Annual Symposium on Computer Application in Medical Care, 1990, p. 345, Accessed: Nov. 14, 2016. [Online]. Available: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2245569/.

[11] M. Holmgren, "Multi-Master Database Replication and e-Learning–Theoretical and Practical Evaluation," 2015, Accessed: Nov. 14, 2016. [Online]. Available: http://www.diva-portal.org/smash/record.jsf?pid=diva2:892052.

[12] D. Song and S. Jing, "Data synchronization solution in cement enterprise," in Control Conference (CCC), 2016 35th Chinese, 2016, pp. 9543–9546, Accessed: Nov. 14, 2016. [Online]. Available: http://ieeexplore.ieee.org/abstract/document/7554873/.

[13] J. Wang and D.-S. Zhang, "Research and Design of Distributed Database Synchronization System Based on Middleware," in 2015 8th International Conference on Intelligent Computation Technology and Automation (ICICTA), 2015, pp. 685–688, Accessed: Nov. 14, 2016. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=7473390.

[14] Y. Wang, J. Wen, W. Fang, and X. Rao, "Research on Incremental Heterogeneous Database Synchronization Update Based on Web Service," in Computational Intelligence and Communication Networks (CICN), 2015 International Conference on, 2015, pp. 1415–1419, Accessed: Sep. 27, 2016. [Online]. Available: http://ieeexplore.ieee.org/abstract/document/7546331/.

[15] Z. Zhenyou, L. Bo, and L. Shu, "Research synchronization mechanism for distributed heterogeneous database," in 2012 Fourth International Conference on Computational and Information Sciences, 2012, Accessed: Nov. 14, 2016. [Online]. Available: https://www.infona.pl/resource/bwmeta1.element.ieee-art-000006300791.

[16] Y. Awe, "Java Trunk : Bi-directional data and file synchronization with SymmetricDS," Jun. 08, 2014. http://javatrunk.blogspot.com/2014/06/bi-directional-data-and-file.html (accessed Jul. 04, 2016).

[17] "The Codership Documentation — Galera Cluster Documentation." https://galeracluster.com/documentation-webpages/documentation/index.html (accessed Apr. 29, 2020).

[18] "SymmetricDS 3.11 User Guide." https://www.symmetricds.org/doc/3.11/html/user-guide.html (accessed May 05, 2020).

[19] Y. Wang, J. Wen, W. Fang, and X. Rao, "Research on Incremental Heterogeneous Database Synchronization Update Based on Web Service," Dec. 2015, pp. 1415–1419, doi: 10.1109/CICN.2015.273.

[20] A. Davies, High Availability MySQL Cookbook. Packt Publishing Ltd, 2010.

[21] "MySQL :: MySQL Router." https://www.mysql.com/products/enterprise/router.html (accessed May 05, 2020).