

DEVELOPMENT BVAG REPLICATION
PROTOTYPE IN DISTRIBUTED DATABASE
ENVIRONMENT

MUHAMMAD FATHUL AMIN BIN
SULAIMAN

Bachelor of Computer Science
(Software Engineering) with Honours

UNIVERSITI MALAYSIA PAHANG

UNIVERSITI MALAYSIA PAHANG

DECLARATION OF THESIS AND COPYRIGHT

Author's Full Name : MUHAMMAD FATHUL AMIN BIN SULAIMAN

Date of Birth

Title : DEVELOPMENT BVAG REPLICATION PROTOTYPE IN
DISTRIBUTED DATABASE ENVIRONMENT

Academic Session : 2019/2020

I declare that this thesis is classified as:

- CONFIDENTIAL (Contains confidential information under the Official Secret Act 1997)*
- RESTRICTED (Contains restricted information as specified by the organization where research was done)*
- OPEN ACCESS I agree that my thesis to be published as online open access (Full Text)

I acknowledge that Universiti Malaysia Pahang reserves the following rights:

1. The Thesis is the Property of Universiti Malaysia Pahang
2. The Library of Universiti Malaysia Pahang has the right to make copies of the thesis for the purpose of research only.
3. The Library has the right to make copies of the thesis for academic exchange.

Certified by:

(Student's Signature)

New IC/Passport Number
Date: 27/1/2023

(Supervisor's Signature)

Assoc. Prof. Ts. Dr. Noraziah
Binti Ahmad

Name of Supervisor
Date: 21/2/2023

NOTE: * If the thesis is CONFIDENTIAL or RESTRICTED, please attach a thesis declaration letter.

THESIS DECLARATION LETTER (OPTIONAL)

Librarian,
Perpustakaan Universiti Malaysia Pahang,
Universiti Malaysia Pahang,
Lebuhraya Tun Razak,
26300, Gambang, Kuantan.

Dear Sir,

CLASSIFICATION OF THESIS AS RESTRICTED

Please be informed that the following thesis is classified as RESTRICTED for a period of three (3) years from the date of this letter. The reasons for this classification are as listed below.

Author's Name
Thesis Title

Reasons	(i)
	(ii)
	(iii)

Thank you.

Yours faithfully,

(Supervisor's Signature)

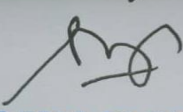
Date:

Stamp:

Note: This letter should be written by the supervisor, addressed to the Librarian, *Perpustakaan Universiti Malaysia Pahang* with its copy attached to the thesis.

SUPERVISOR'S DECLARATION

I hereby declare that I have checked this thesis and in my opinion, this thesis is adequate in terms of scope and quality for the award of the degree of Bachelor of Computer Science in Software Engineering.



ASSOC. PROF. DR. NORAZIAH BINTI AHMAD
ASSOCIATE PROFESSOR
FACULTY OF COMPUTING
COLLEGE OF COMPUTING & APPLIED SCIENCES
UNIVERSITI MALAYSIA PAHANG
26600 PEKAN, PAHANG DARUL MAKMUR
TEL : 09-424 4700 FAX : 09-424 4666

(Supervisor's Signature)

Full Name : Assoc. Prof. Ts. Dr. Noraziah Binti Ahmad
Position : Associate Professor
Date : 21/-2/2023

(Co-supervisor's Signature)

Full Name :
Position :
Date :



STUDENT'S DECLARATION

I hereby declare that the work in this thesis is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at Universiti Malaysia Pahang or any other institutions.

Fathul Amin

(Student's Signature)

Full Name : MUHAMMAD FATHUL AMIN BIN SULAIMAN

ID Number : CB19070

Date : 27 JANUARY 2023

DEVELOPMENT BVAG REPLICATION PROTOTYPE
IN DISTRIBUTED DATABASE ENVIRONMENT

MUHAMMAD FATHUL AMIN BIN SULAIMAN

Thesis submitted in fulfillment of the requirements
for the award of the degree of
Bachelor of Computer Science (Software Engineering) with Honours

Faculty of Computing
UNIVERSITI MALAYSIA PAHANG

JANUARY 2023

ACKNOWLEDGEMENTS

Foremost, I would like to express my sincere gratitude to my supervisor Assoc. Prof. Ts. Dr. Noraziah Ahmad for the continuous support of my degree study, for her patience, motivation, enthusiasm, and immense knowledge. Her guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for degree study.

Special gratitude also to my family, especially to my father, Sulaiman bin Mohd Yuneh; my mother, Zetty Azikin binti Ismail; for their patience and moral support throughout my life.

Finally, I thank my fellow mates in Faculty of Computing; Dhiyaurahman Danial, Afiq Danial, Ain Mutaqorrobin and Badri Hilmi for the stimulating discussions, for the sleepless nights we were working together and for all the fun we have had in these few years.

ABSTRAK

Replikasi ialah teknik yang berguna untuk sistem pangkalan data teragih. Melalui teknik ini, sesuatu data boleh diakses dari pelbagai lokasi. Oleh itu, ia meningkatkan ketersediaan data dan kebolehcapaian kepada pengguna. Apabila satu tapak gagal, pengguna masih boleh mengakses data yang sama di tapak lain. Teknik seperti Baca-Satu-Tulis-Semua (ROWA), Skim Replikasi Hierarki (HRS) dan Skim Replikasi Cawangan (BRS) adalah teknik popular yang digunakan untuk replikasi dan pengurusan data. Walau bagaimanapun, teknik ini mempunyai kelemahan dari segi kos komunikasi. Akibatnya, ROWA, HRS dan BRS mengambil masa pelaksanaan yang lama untuk transaksi kerana teknik ini perlu mereplikasi datanya ke semua pelayan. Dalam penyelidikan ini, skim beberapa-data-ke-beberapa-tapak yang dipanggil Peruntukan Undi Perduaan pada Grid (BVAG) dicadangkan. Ia berfungsi dengan mempertimbangkan penetapan undi perduaan jiran kepada struktur grid logiknya pada salinan data berpecah-belah untuk mengurus urus niaga dalam sistem. Untuk memudahkan, jiran ditugaskan dengan undi satu atau sifar. Tugasannya menyediakan kos komunikasi minimum kerana bilangan minimum saiz kuorum yang diperlukan. Selain itu, ia meminimumkan kapasiti storan yang diperlukan kerana kami menyimpan pangkalan data yang telah berpecah-belah. Pembangunan prototaip untuk teknik replikasi BVAG telah dijalankan menggunakan HTML, CSS, JavaScript dan PHP. Prototaip ini dibangunkan untuk menghasilkan aplikasi berasaskan web yang menggunakan teknik replikasi BVAG. Daripada pembangunan itu, BVAG berasaskan web telah dibangunkan dan diuji dengan input data asas untuk mereplikasi data. Daripada keputusan, ia menunjukkan prototaip BVAG berasaskan web berfungsi dan mampu mereplikasi data ke tapak jiran.

ABSTRACT

Replication is a useful technique for distributed database systems. Through this technique, a data can be accessed from multiple locations. Thus, it increases data availability and accessibility to users. When one site fails, user still can access the same data at another site. Techniques such as Read-One-Write-All (ROWA), Hierarchical Replication Scheme (HRS) and Branch Replication Scheme (BRS) are the popular techniques being used for replication and data management. However, these techniques have its weaknesses in terms of communication costs. Consequently, ROWA, HRS and BRS take long executing time for a transaction since these techniques have to replicate its data to all servers. In this research, the some-data-to-some-sites scheme called Binary Vote Assignment on Grid (BVAG) is proposed. It works by considering neighbors binary vote assignment to its logical grid structure on fragmented data copies in order to manage transactions in the systems. For simplicity, the neighbours are assigned with vote one or zero. The assignment provides minimum communication cost due to the minimum number of quorum size required. In addition, it minimizes the storage capacity needed since we store database that has been fragmented. The development of prototype for the BVAG replication techniques were carried out using HTML, CSS, JavaScript and PHP. The prototype was developed in order to produce a web-based application that utilize BVAG replication techniques. From the development, the web based BVAG were developed and tested with basic data input to replicate the data. From the results, it shows the web based BVAG prototype is working and able to replicate data to the neighbour's site.

TABLE OF CONTENT

DECLARATION	
TITLE PAGE	
ACKNOWLEDGEMENTS	ii
ABSTRAK	iii
ABSTRACT	iv
TABLE OF CONTENT	v
LIST OF TABLES	viii
LIST OF FIGURES	ix
LIST OF SYMBOLS	xi
LIST OF ABBREVIATIONS	xii
CHAPTER 1 INTRODUCTION	1
1.1 Introduction	1
1.2 Problem Statement	3
1.3 Objective	3
1.4 Scope	4
1.5 Significance of Project	4
1.6 Report Organization	4
CHAPTER 2 LITERATURE REVIEW	6
2.1 Introduction	6
2.2 Data Grid	6
2.3 Binary Vote Assignment Grid (BVAG) Replication Techniques	7

2.4	Read-One-Write-All Monitoring Synchronization Transaction System (ROWA-MSTS) Replication Techniques	8
2.5	Hierarchical Replication Scheme (HRS) Replication Techniques	9
2.6	Database Replication	10
2.7	Review of Existing Systems	10
2.7.1	Apache Ignite	10
2.7.2	MongoDB	12
2.7.3	Apache Cassandra	13
2.8	Comparative Analysis	15
2.9	Summary	16
2.10	Proposed Application & System	16
CHAPTER 3 METHODOLOGY		17
3.1	Introduction	17
3.2	Project Management Framework	17
3.2.1	Agile Methodology	17
3.3	Project Requirement	21
3.3.1	Functional Requirement	21
3.3.2	Non-Functional Requirement	21
3.4	Proposed Design	22
3.4.1	Flowchart	22
3.4.2	Context Diagram	23
3.4.3	Framework Architecture	24
3.5	Design Prototype (Framework Architecture)	25
3.6	Testing Plan	26
3.7	Potential Use of Proposed Solution	27

3.8	Gantt Chart	27
CHAPTER 4 RESULTS AND DISCUSSION		28
4.1	Introduction	28
4.2	Hardware and Software Components	28
4.3	BVAG PROTOTYPE IMPLEMENTATION	29
4.3.1	Prerequisites	29
4.3.2	Creating Hosted Ubuntu Server Virtual Machine	30
4.3.3	Initial Server Setup with Ubuntu 20.04	34
4.3.4	Install MySQL on Ubuntu 20.04	38
4.3.5	Configure MySQL BVAG Replication on Ubuntu 20.04	42
4.4	BVAG TESTING	52
4.5	RESULT AND CONCLUSION	52
CHAPTER 5 CONCLUSION		53
5.1	Introduction	53
5.2	Limitation and Constraint	53
5.3	Future Work	54
REFERENCES		55
APPENDIX A SAMPLE APPENDIX 1		57
APPENDIX B SAMPLE APPENDIX 2		58

LIST OF TABLES

Table 2.1: Comparison of existing DBMS	15
Table 3.1: Functional Requirements	21
Table 3.2: Non-Functional Requirements	22

LIST OF FIGURES

Figure 2.3.1: 9 sites organized in 3 x 3 grid structure	7
Figure 2.4.1: The framework of ROWA-MSTS	9
Figure 2.5.1: All replicas in HRS update data	9
Figure 2.7.1: Apache Ignite Cache Replication Techniques	11
Figure 2.7.2 : MongoDB Master/Slave Replication	12
Figure 2.7.3 : MongoDB Replica Set	13
Figure 2.7.4 : Apache Cassandra Features	13
Figure 2.7.5 : Apache Cassandra Replication Techniques	14
Figure 3.2.1: Agile Methodology Phases	18
Figure 3.4.1: Flowchart of BVAG without concurrent transaction	22
Figure 3.4.2: Context Diagram of BVAG	23
Figure 3.4.3: Use Case Diagram BVAG	24
Figure 3.5.1: Design Prototype of BVAG	25
Figure 3.8.1: Gantt Chart	27
Figure 4.3.1: Create Droplets - Choose Region	30
Figure 4.3.2: Create Droplets - Choose OS	31
Figure 4.3.3: Create Droplets - Choose Hardware Specifications	32
Figure 4.3.4: Create Droplets - Set Server Password	32
Figure 4.3.5: Create Droplets	32
Figure 4.3.6: 4 Ubuntu Server Created	33
Figure 4.3.7: Logged in as root	34
Figure 4.3.8: Add New User	35
Figure 4.3.9: Granting Administrative Privileges	36
Figure 4.3.10: UFW List	36
Figure 4.3.11: UFW Allow OpenSSH	37
Figure 4.3.12: sudo apt update	38
Figure 4.3.13: sudo apt install mysql-server	38
Figure 4.3.14: sudo systemctl start mysql.service	38
Figure 4.3.15: sudo mysql_secure_installation	39
Figure 4.3.16: Set Password MySQL	40
Figure 4.3.17: Accepting Security Questions	40
Figure 4.3.18: MySQL Service Status Check	41
Figure 4.3.19: Generate UUID member3 nodes	42

Figure 4.3.20: Nano Text Editor	43
Figure 4.3.21: BVAG Replication Script Configuration	44
Figure 4.3.22: Shared Replication Group Config	45
Figure 4.3.23: sudo mysql	46
Figure 4.3.24: CREATE USER Syntax	47
Figure 4.3.25: Grant Replication Slave	47
Figure 4.3.26: Output	48
Figure 4.3.27: Group Replication Member 1 Status	49
Figure 4.3.28: Query from Table	50
Figure 4.3.29: Check Membership Replication	51
Figure 4.3.30: Successful Replicate Data using BVAG	51
Figure 4.4.1: Data replicated on member2	52

LIST OF SYMBOLS

LIST OF ABBREVIATIONS

BVAG	Binary Vote Assignment Grid
ROWA	Read-One-Write-All
HRS	Hierarchical Replication Scheme
BRS	Branch Replication Scheme
DDS	Distributed Database System

CHAPTER 1

INTRODUCTION

1.1 Introduction

In a distributed environment, organisations must supply current data to geographically distant users and manage a high volume of requests for data distributed across numerous sites (Azila et al., 2021). Therefore, the storage, availability, and consistency of data are critical challenges that must be addressed in order for distributed users to access data quickly and safely from many websites. (Azila et al., 2021). Replication is one method of making such data more widely available. A distributed database system's replication procedure involves duplicating and maintaining database items across several databases (Tarun et al., 2019)

One of the main problems in distributed systems is synchronous data replication in the database environment. It is safe to do so for a system such as school library management systems. However, for a system such as an online banking system, which requires tight semantic data, it is not a wise solution to use this approach. For instance, even in the modern day, internet inter-banking transactions still make use of asynchronous updates in banking systems. When consumers conduct an online inter-banking transaction, they have to wait for the recipient account to be updated, which may take anywhere from one day to several days. Another example might be that a cheque must first be validated and approved before it can be released to the recipient.

Database technology has emerged as an essential component in the majority of modern businesses. Because of developments in telecommunication services, Distributed Database Systems (DDS) have become more possible and accessible to be implemented. Often, a DDS is made up of a number of distinct but interconnected databases that are housed in various locations throughout the world and are able to interact with one another

over a network (Avram, n.d.). Usually, the system is managed by a Distributed Database Management System (DDBMS). Every site of DDS has its own hardware; hence, it is capable of independent operation.

Replication is referred to as the act of transferring information across redundant resources, such as software or hardware components, in order to guarantee their consistency. The data's reliability, fault-tolerance, and accessibility are all improved as a result of this procedure (Ahmad et al., 2007). Because other data access methods exist, replication offers users with rapid, local access to shared data while protecting application availability.

Data can be replicated in either a synchronous or an asynchronous method. These two modes are known as synchronous and asynchronous replication, respectively. In most cases, asynchronous replication will transmit data in a not consistent way rather than in a continuous flow. Asynchronous replication also produced issues with the receiver receiving data from the sender. It is appropriate for single object updates. It fails, however, when numerous objects are involved in a single update since it will only update when there is a request (Sathya & Seshu, 2008).

The use of synchronous replication is an option for resolving issues that arise from asynchronous replication. The fact that synchronous replication relies on quorum to carry out its actions means that data consistency can always be relied upon.(Budiarto et al., 2002). In addition, synchronous replication ensures that data stores maintain a "strict consistency" throughout the process. Within a single transaction, any copy that has been updated will immediately have those updates applied to all other copies that are part of that transaction. This guarantees that all of the copies across all of the sites are identical and consistent with one another. In a distributed system setting, having a copy that is consistent across all websites is beneficial to the business because it ensures that data is always up to date and can be accessed at any time and from any location. Synchronous replication, on the other hand, calls for an enormous amount of storage capacity. This is because it requires multiple copies of replicated data to be stored across a number of sites, in addition to an expensive synchronisation mechanism, in order to keep the data consistent after changes are made. As a consequence of this, a suitable approach is required to manage the replicated data in a distributed system environment (Deris et al., 2009).

1.2 Problem Statement

A wide variety of industries, including banking and insurance, as well as many businesses, have made use of data replication in order to safeguard their information in the event of unanticipated system failures. The process of data replication, also known as data duplication, generates a backup duplicate of the data that is stored on each of the multiple servers. The approaches that are currently utilised in the design and implementation of database systems do not have high availability and reliability. This is since if the database site goes down, the entire system is rendered inoperable. The currently available replication strategies that have been suggested make use of several replicas sites, for instance, Read-One-Write-All (ROWA) techniques (Ahmad, 2010). If one site is updated, the modifications that users have made on other sites will be updated when that site is updated. Binary Vote Assignment Grid (BVAG) (Ubaidillah et al., 2021) can be used to handle database replication in a cluster server for synchronous updating, with the constraint that it will only consider scenario in which there is no failure. Because the data was already fragmented before being allocated to locations, this technique can reduce the amount of money spent on communication while simultaneously reducing the amount of storage space that is required.

1.3 Objective

Based on problem statements, the objectives of the project are:

- i. To develop the BVAG replication techniques prototype in a distributed database environment.
- ii. To test the prototype to make sure it can replicate committed data.
- iii. To ensure the prototype is highly available and fault tolerant.

1.4 Scope

User Scope:

- i) General Small/Large Institution

System Scope:

- i) To develop a distributed database replication technique
- ii) To develop it in a distributed database environment

Development Scope:

- i) This prototype is using 4 Ubuntu Server Virtual Machine hosted on DigitalOcean Cloud Hosting
- ii) The BVAG replication technique is using MySQL Replication.
- iii) Each server is assigned unique IP address.

1.5 Significance of Project

The focus of this project which is data replication have been elaborated. When there are advantages, there are also disadvantages. The significance of this project is to develop and test a replication technique which is the BVAG in a distributed database environment since the working prototype is not available yet and hence it needs to be developed and to make sure the prototype is web based since it is more suitable and easier to use.

1.6 Report Organization

This report consists of three chapters. Chapter 1 explains about the overview of the project including the Introduction, Problem Statements, Objectives of the project,

Scope and Report Organization. Chapter 2 which is Literature Review will focus on explaining about data grid, data replication in the grid, database replication as well as existing data replication techniques. Chapter 3 will focus on the methodologies of the project including the Project Management Framework used, project requirements which includes functional and non-functional requirements of the BVAG replication prototype. Chapter 4 addresses the implementation of BVAG web application prototype. The conclusions of the present research are summarized and presented in Chapter 5. Suggestion and recommendations for the future work are also present in this chapter.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This chapter reviews about data grid, data replication in the grid, existing system, and summary about the reviews.

2.2 Data Grid

Grid computing, it refers to the evolving computational and networking infrastructure that is meant to enable widespread and reliable access to data and computational resources over a wide area network across the organisational domain (Foster et al., 2001; Francine Berman et al., 2003). Connecting computers and storage resources around the globe, the data grid lets users to exchange data and resources. As of now, the Data Grid's storage capacity is at about Terabytes. Network and Grid designers face a major issue in ensuring effective access to such large and geographically scattered data. It is possible that a big amount of bandwidth will be utilised during the process of transferring a file from the server to the client whenever a user makes a request for a certain file. In addition, the amount of latency that is involved may be considerable depending on the size of the files that are involved. (Sathya et al., 2006).

The use of a data grid is an excellent method for handling the massive amounts of information generated by scientific investigations and computer simulations. Using Grid, we can scale, reduce, and adapt our infrastructure in response to our changing demands while still maintaining a secure and flexible architecture (Linesch & Marketing, 2007). It's getting more and more critical to address a problem with grid data management. Managing a data grid isn't a simple task, and there are a number of connected issues that must be taken into account. In terms of data management, the grid

makes it possible to store a large number of replicas of data objects, each of which may have a unique version or degree of freshness. This facilitates a high level of availability, reliability, and performance, which in turn allows it to cater to the requirements of users and applications in the most effective manner (Voicu et al., 2009). In addition, the amount of data that is being managed by the data grid is always expanding (Pérez et al., 2010).

2.3 Binary Vote Assignment Grid (BVAG) Replication Techniques

In BVAG, all sites are logically organized in the form of two-dimensional grid structure. That means, if a BVAG involves of twenty-five sites, it will logically organize in the form of 5×5 grids as shown in Figure 2.3.1 . Every site has a primary data file. A site is either operational or failed and that state of either it is operational or failed of each site is statistically independent to the others. When a site is operational. the copy at the site is available; otherwise it is unavailable.

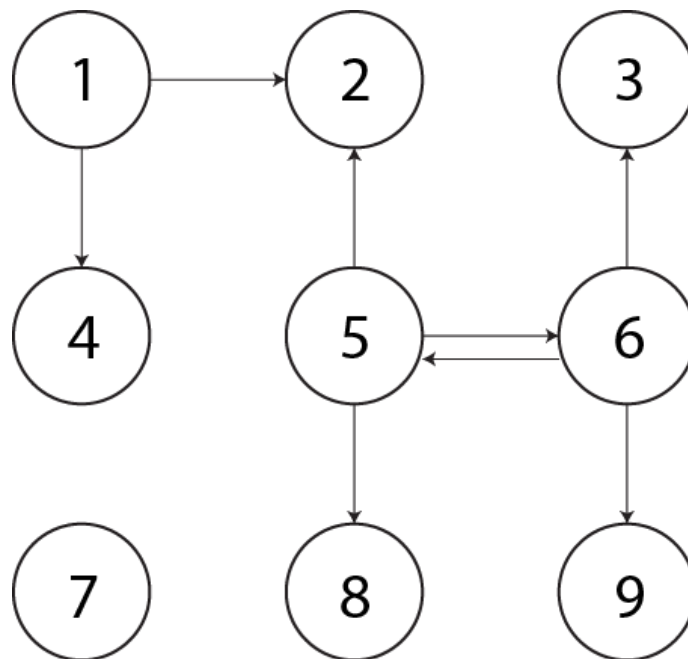


Figure 2.3.1: 9 sites organized in 3 x 3 grid structure

A site X is a neighbour to site Y, if X is logically adjacent to Y. A data from its primary site will replicate to the neighbouring sites. The number of sites that data will be

replicated, d , are smaller or equal to 5. From Figure 2.3.1, data from site 1 will be replicated to site 2 and site 4 which are its neighbour. Site 6 will replicate its data to 3 servers because it has 3 neighbours. Its neighbours are site 3, 5 and 9. Site 5 has 4 neighbours which are site 2, 4, 6 and 8. Hence, the total maximum number of data replication are 5.

2.4 Read-One-Write-All Monitoring Synchronization Transaction System (ROWA-MSTS) Replication Techniques

Read-One-Write-All Monitoring Synchronization Transaction Systems (ROWA-MSTS) have been developed based on ROWA technique. The ROWA-MSTS techniques handle each site either it is operational or down and to communicate each other. The researcher used VSFTPD (GPL licensed FTP server for UNIX systems) as an agent communication between replicated servers. In ROWA-MSTS techniques, replicas consistencies are guaranteed by the consistency of execution on one replica, but the client replicas are only updated and cannot provide accurate responses to queries. Synchronous replicated methods guarantee that all replicas are maintained consistent at all times by executing each transaction locally only after all replicas have agreed on the execution order. Through this, a very strict level of consistency is maintained. Figure 2.4.1 shows the framework of ROWA-MSTS in distributed environment.

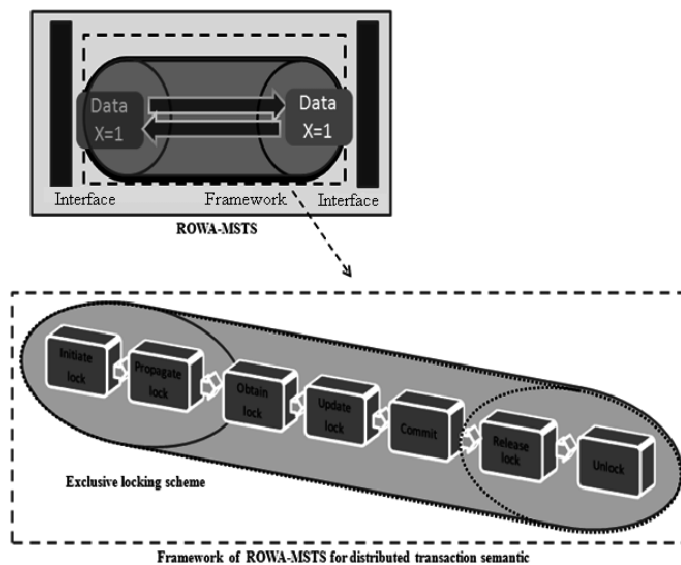


Figure 2.4.1: The framework of ROWA-MSTS

2.5 Hierarchical Replication Scheme (HRS) Replication Techniques

Hierarchical Replication Scheme (HRS) consists of a root database server and one or more database servers organized into a hierarchical topology. Figure 2.5.1 shows all replicas in HRS update data.

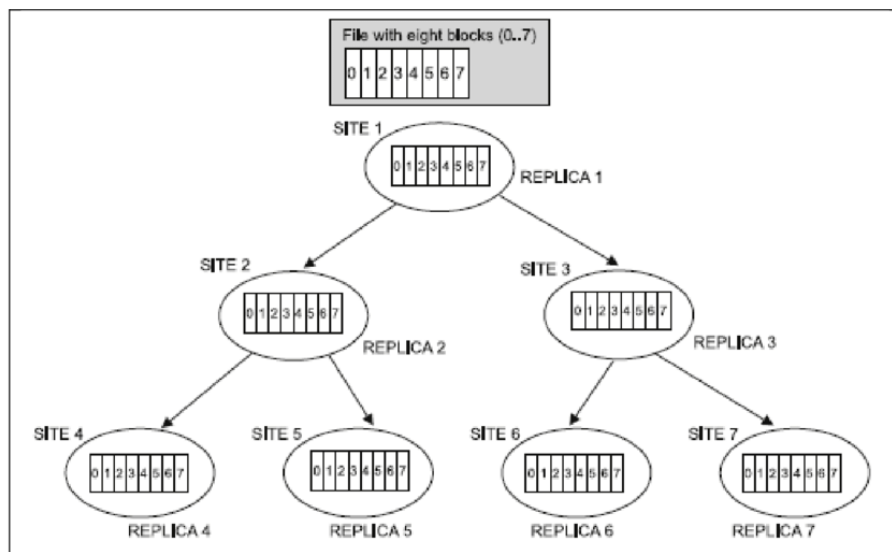


Figure 2.5.1: All replicas in HRS update data

The architecture of HRS is shown in Figure 2.5.1. Based on Figure 2.5.1, replication process starts when a transaction initiates at any block at site 1. In HRS, all update operations are conducted on a master replica, and then the modifications are propagated to all replicas. Once the changes have been made, all the data will be replicated into all sites. At last, all sites will have all the same data.

The drawback in HRS it is requires many replica servers. Consequently, it will take more executing time to compare to BVAG because BVAG only requires minimum 3 servers. HRS also will replicate whole file to its replica servers while in BVAG, the file will be fragmented before it is replicated to replica servers.

2.6 Database Replication

Replication frequently referred as parallel and distributed computing technique widely used in order to achieve high performance, scalability, fault-tolerance and high availability of computer systems. Most of commercial and research databases are based on the asynchronous replication model where changes will be updated after a transaction has committed. As a result, inconsistencies of data among the copies may arise. In order to eliminate the inconsistencies, synchronous replication models can be used. In practice, many database designers do not regard synchronous replication as a practical option due to limitations of traditional data replication techniques such as deadlock. Most of the work done synchronous replication protocols are based on one-copy-serializability which mean the effect of transaction performed by clients on replicated objects should be the same as if they had been performed one at a time on a single set of objects.

2.7 Review of Existing Systems

This section explains about the review of three existing Database Management System (DBMS) and its properties.

2.7.1 Apache Ignite

Apache Ignite is an open-source distributed database server software that enables developers to deal with big size data sets in real time and with other features of in-memory computing. Its default storage and processing tier is RAM. It is written in Java. It is built on Spring and supports Java 7, Java 8, .Net, C++, and PHP. Apache Ignite is a Key-Value data model implementation.

Ignite is totally peer-to-peer in nature. Each node in the Ignite cluster may accept read and write requests regardless of the location of the data being written. The Ignite design indicates that the whole system is naturally scalable and highly available. Internode communication in Ignite enables all nodes to get updates fast and without the need for a master coordinator. Nodes may be added or withdrawn without causing any disruptions in order to improve the available RAM. Ignite data fabrics are completely

robust, enabling automatic detection and recovery of a single or several servers without causing any downtime.

According to Figure 2.7.1, cache data is replicated to all cluster members. Because the data is duplicated to each cluster node, it is immediately accessible for usage. This gives the fastest possible read access since each member uses its own memory to access the data. The disadvantage is that regular writing is quite costly. To update a replicated cache, the new version must be sent to all other cluster members. This will hinder scalability if the update frequency is high.

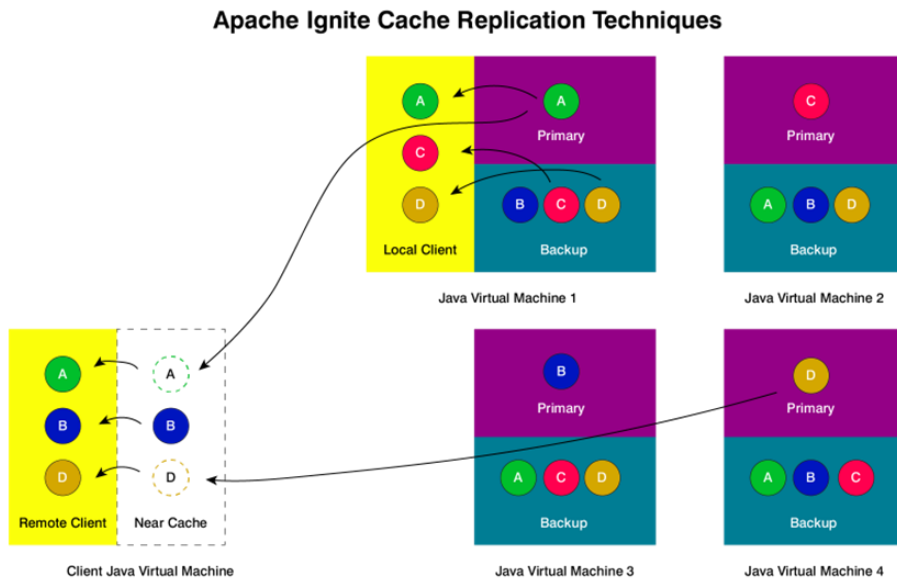


Figure 2.7.1: Apache Ignite Cache Replication Techniques

The data is replicated across all cluster nodes in the figure above. The size of a replicated cache is restricted by the amount of memory available on each node with the least amount of RAM. This mode is optimal for instances in which cache reads outnumber cache writes and data collections are modest. Replication's scalability is inversely related to the number of members, the frequency of member updates, and the size of the updates.

2.7.2 MongoDB

MongoDB is a general purpose, document-based, distributed database. It was built in C/C++ programming language. MongoDB's replication technique is Master/Slave and Replica Set. While the servers in a Master/Slave replication have distinct functions, there is only one Master. As seen in Figure 2.7.2, one is a Master server, while the rest are slaves. Write operation implement on Master, Slaves will send the synchronize data command asynchronously to Master to update its data. Read operations are only implemented on the Master to ensure strong consistency, whereas read operations are implemented on the Slave to ensure eventual consistency. Because Master/Slave replication does not support automatic failover, if the Master fails, the Slave must shut down and restart in order to change to the Master role.

Master/Slave Replication

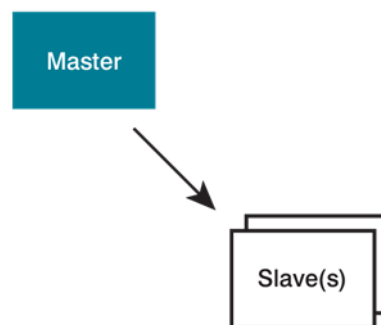


Figure 2.7.2 : MongoDB Master/Slave Replication

A replica set is a collection of MongoDB servers that keep a copy of the same data with automatic failover and recovery of member nodes. Figure 2.7.3 illustrates the replica set model. In a replica set, there are three server states: primary, secondary, and recovering. At any moment in time, only one server is main in a replica set. The primary server supports write and read with strong consistency, whereas the secondary server supports read with eventual consistency. A recovering server is one that has regained sync prior to entering secondary mode. A write is genuinely committed when it has been duplicated to a majority of the set's members. Prior to the genuine cluster wide commit,

writes committed at the primary of the set may be visible. Thus, we have "READ UNCOMMITTED" semantics for reading.

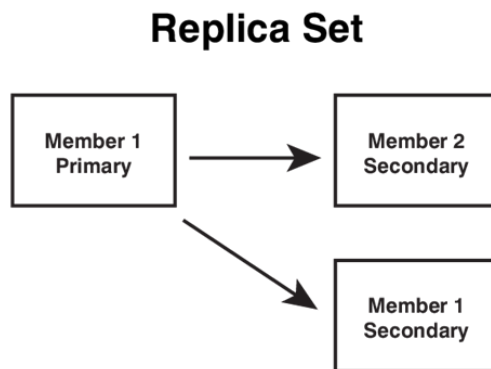


Figure 2.7.3 : MongoDB Replica Set

2.7.3 Apache Cassandra

An open source distributed database server software, Apache Cassandra is designed to store massive volumes of data on low-cost servers while ensuring high availability. It was built in C/C++ programming language. Cassandra can run clusters across many data centres with no issues. Key-value and wide column data models are used to form a hybrid model for this structure's data storage. In a Cassandra system, two things are critical: the data partition and the data model.

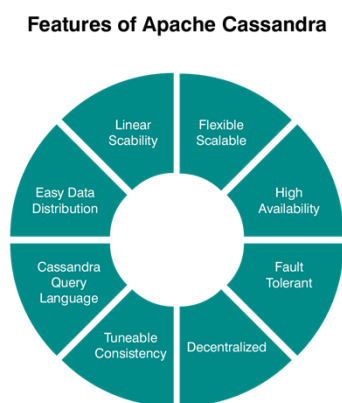


Figure 2.7.4 : Apache Cassandra Features

Since data is kept on numerous workstations for data security reasons, syncing data between the machines is critical. Cassandra clusters may be constructed using many physical computers or multiple networked virtual machines. Due to the presence of other computers on the network, cluster members will be able to determine which node is a part of the cluster based on the information included in the configuration files. The cluster members interact through the Gossip protocol. Additionally, it employs a new protocol to configure the cluster: this is the Snitch, which enables specifying which node belongs to the data centre and therefore creates the cluster.

Once the cluster has been configured using the Gossip and Snitch protocols, it is required to describe the placement of the data for the system to be fault-tolerant and have high-availability. This requires determining which node or nodes to store data in. To provide high availability, data must be stored on several nodes. To provide fault tolerance and high availability, we duplicate the data, often three times. This implies that we have computed a token for each partition key, which indicates which node will retain that partition, but in the cluster ring, the following (clockwise) two nodes (assuming the replication factor is 3) will also store this partition. The client sends data to the data centre in Figure 2.7.5, and since the B node is responsible for the data, the two nodes next to it will also store it. Note that the client programme talks with the F node, which is now the coordinator, but it may also communicate with any other node; in this case, the node becomes the coordinator since all nodes are equal.

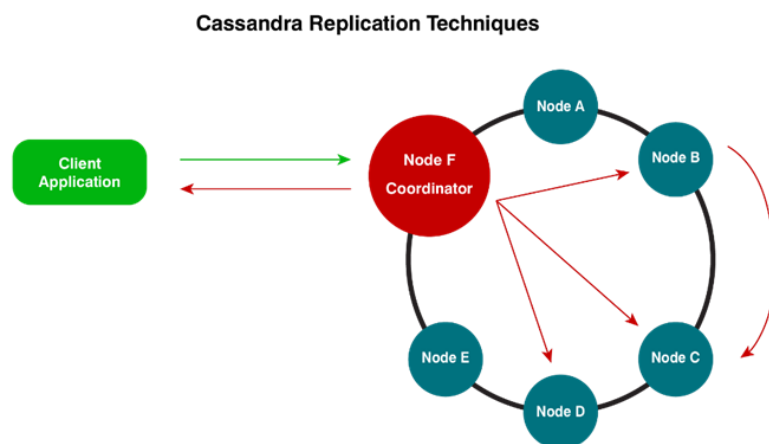


Figure 2.7.5 : Apache Cassandra Replication Techniques

2.8 Comparative Analysis

Based on the review done earlier, Table 2.1 shows the comparison between the three existing DBMS.

Table 2.8.1: Comparison of existing DBMS

Specifications	Apache Ignite	MongoDB	Apache Cassandra
Programming Language	C#, C++, Java, PHP, Python	C#, C++, Java, C, Go, PHP, Python	C#, C++, Java, Javascript, PHP, Python
Operating Systems	Linux, OS X, Solaris, Windows	Linux, OS X, Solaris, Windows	Linux, OS X, Windows
Database Model	Relational	Key-value store	Key-value store
Partitioning Methods	Sharding	Sharding	Sharding
Advantages	Faster in read-intensive applications	MongoDB's schema is not predefined. It means that it has a dynamic schematic architecture that works with non-structured data and storage.	Offers superior write performance, massive and linear scalability.
Disadvantages	Keeping data only in memory has some serious drawbacks if there is a catastrophic cluster failure.	Requires a high amount of storage due to the lack of joins functionalities which lead to the duplication of data. There is an increase in data redundancy which takes up unnecessary space in the memory.	It is disk-based, which ultimately limits the speed of some operations because data needs to be written to and read from disks.

2.9 Summary

This chapter reviews a study on data grid as well as reviews existing DBMS such as Apache Ignite, MongoDB and Apache Cassandra. This chapter also shows comparison between the three (3) existing DBMS as well as the advantages and disadvantages.

2.10 Proposed Application & System

The prototype that this project shall propose is a custom-built prototype using Microsoft Visual Studio and integrate with Apache Ignite DBMS. It will include the features such as Update Data, View Log and Replicate Data correspond to BVAG data replication techniques.

CHAPTER 3

METHODOLOGY

3.1 Introduction

This chapter will focus on the software development approach that will be used on this project. The waterfall model will be utilised as the technique for this. The waterfall model was one of the first concepts used in software development. The waterfall model, sometimes known as the liner-sequential life cycle model, is another name for the liner-sequential life cycle model. In this chapter, we will also go over each step of the waterfall model in depth, as well as how we approach each phase. The chapter will also examine the system's functional and non-functional aspects, as well as the proposed system's limits and constraints. Finally, there will be an early idea of the suggested system's design. Finally, the Gantt Chart will be displayed to represent the time period of the project's development till conclusion.

3.2 Project Management Framework

3.2.1 Agile Methodology

Based on the numerous types of software development cycles, after extensive research, the best selected development cycle to be used in development for this proposed project will be the AGILE life cycle model. The main reason for selecting this model as our proposed system will be to be developed on a database thus this model can perfectly fit this project. The reason for picking the AGILE life cycle model is that we are able to determine whether the system requirements are all meet. It helps the process of development to be more effective to ensure that the standard of the system is meet with what the customer demands. With AGILE development we are able to deploy a version

at an early stage to get a first impression to understand better on the customer feedback and make improvements to the system.

Figure 3.2.1 below shows all the phases in Agile methodology which includes planning, design, development, testing and deployment and feedback review.

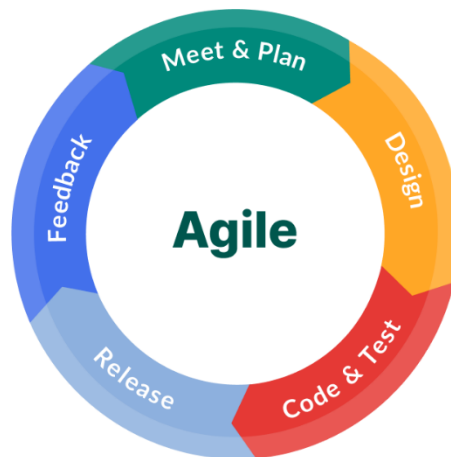


Figure 3.2.1: Agile Methodology Phases

3.2.1.1 Planning Phase

The state of the needs phase will be the initial phase of the AGILE life cycle. Obtaining the needs that are required for their businesses during the planning process. This phase is critical since it aids in narrowing the scope of the distributed database system's purpose and giving developers an idea of where the opportunity for improvement may be found. This phase involves getting to know the parties involved as well as getting feedback on previous reports. This will be extremely helpful in determining the system's scope and bounds. The initial step in creating a distributed database prototype utilising the BVAG approach was to figure out what the system's goal was. From there, the requirements are needed to ensure to fulfil the scope and constraints that will be faced in the system. Other than that, what will also be done is to view the existing system to have a better understanding of the system by going through reports and papers to get more information. For our proposed system we will be first collecting

information from the reports to determine what is the best approach to develop the system. We see how data is easily breach or tampered and how a singular database causes problem for the companies or government agency as they are more easily attack and can cause a collapse to the organization. Once all the requirements and relevant information have been collected then the development life cycle of this system can move to the next phase which is the design phase.

3.2.1.2 Design Phase

The design phase will be the second phase of this AGILE development cycle. In this phase, once we have understood the system requirements, we are able to narrow down, we can create a use case diagram and the flow of the system on how the implementation of using the system. In this phase for this project, once we have understood the requirements of the system, we are able to determine on the properties of the system required for the system such as what the system requirement and the implementation of BVAG replication technique into the system. We will design how the flow of the system be from the user and how data inserted, and the data is replicated once committed. Once this phase completed, we will move to the construction phase.

3.2.1.3 Development Phase

This phase is the third phase of AGILE development life cycle which is the development phase. In this phase, will be the process of the developing the system. It will begin with ensuring the tools required to develop the system and the process of developing the module for the system. For this project, we will take into consideration of the tools require to develop the system. We will need to consider the tools such as software to implement the distributed database system, how to implement BVAG techniques when user commit data and many other more. Other factors that also need to be considered are also on how to communicate with the database any many other more. Once the factors of using the tools to implement the system have been taken into consideration, we will begin with the development process where we will break into parts to ensure that the project development can be completed within the time frame. The breakdown of the project development will be from implementing the distributed database and then to developing the insert of data of the user which the database will

commit and do replication. Once the development phase is complete, we will begin the next phase which will be deployment.

3.2.1.4 Testing and Deployment Phase

In this phase, we will release the final product of the system to know better of how well the system is and to ensure the requirements of the system is meet and the stakeholders can use it in their own work environment to ensure that is satisfy their work condition environment. For this project, once the development of the project is completed, we will try to recreate in a work scenario of the system to ensure that it meet with the requirements of the system and ensure the system functionality works as is intended. We will also allow the stakeholders to try out the system to ensure that the system satisfy with the stakeholders work environment. We will also be conducting testing to the system so that it meets with the requirements of the system and also that it meets the standards of what the stakeholders want. Multiple technique will be used on the testing phase to ensure that they are able to detect bugs as well. For this project, we will use several testing techniques such as black box testing and white box testing to ensure we are able to detect the bugs that are in the system and the traffic limit of the system to ensure the availability of the system. We will also ask the stakeholders to test the system to ensure that they can understand the system and use the system to determine whether it met their standards. Once the testing phase is complete, we will move to the final phase which is the feedback phase.

3.2.1.5 Feedback Phase

In this phase which is the feedback phase, we will use this phase as a way to understand the system better to determine whether the system require modification from the testing conducted on the system and to gain feedback from the stakeholders to ensure whether the stakeholders and the system achieve the intention of what the stakeholders want. Discussion will be done from all the feedback of the testing and stakeholders. Then the next step we will deciding on the next part of the system. For this project, based on the testing result and the stakeholders opinion on the system, we will analysis the result to determine the system best route to take on whether to improve the system more to ensure it meets the requirements of the system or the system has reach the satisfaction level of the desired opinion of the stakeholders. If the stakeholder's opinion and test result

prove that improvement needs to be done on the system, we will return back to phase and continue the proses again until the system has reach the satisfaction level. If the stakeholders and test result shows a satisfaction level of the system, they system is then ready to be used in the stakeholders work environment.

3.3 Project Requirement

3.3.1 Functional Requirement

Below shows Table 3.3.1 which consists of functional requirements for the proposed prototype.

No	Requirement ID	Requirement
1	BVAG-FR01	The system should allow the databases to connect with another database within the same network.
2	BVAG-FR02	The system should replicate data stored in the database using BVAG approach.
3	BVAG-FR03	The system should provide log history for every activity executed.

Table 3.3.1: Functional Requirements

3.3.2 Non-Functional Requirement

Below shows Table 3.3.2 which consists of non-functional requirements for the proposed prototype.

No	Requirement ID	Requirement
1	BVAG-NFR01	The system databases can always be added to the network to increase with the demand from the organization

2	BVAG-NFR02	The system shall divert the network to other databases that are still online if one of the other databases went offline or went into an error state.
3	BVAG-FR03	The system should be able to run 24 hours per day at any geographical location to ensure the operation of that organization is running smoothly.

Table 3.3.2: Non-Functional Requirements

3.4 Proposed Design

3.4.1 Flowchart

Figure 3.4.1 below shows the flowchart diagram of the Binary Vote Assignment Grid (BVAG) without concurrent transaction replication techniques prototype. The figure shows the input processes and expected output of the prototype.

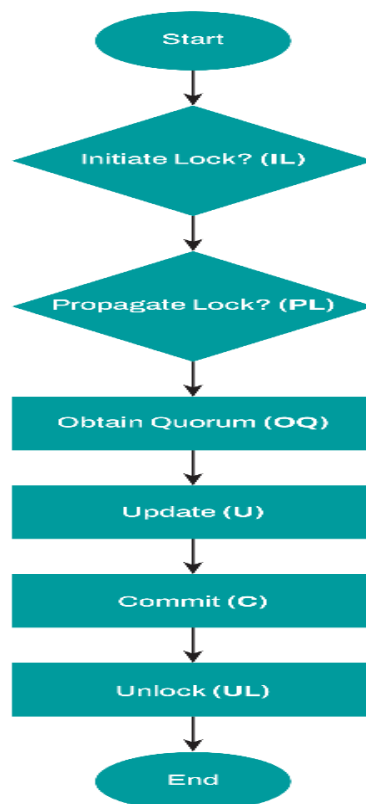


Figure 3.4.1: Flowchart of BVAG without concurrent transactions

There are six main phases involve in BVATM; Initiate Lock (IL), Propagate Lock (PL), Obtain Quorum (OQ), Update(U), Commit (C), Unlock (UL). IL phase involves locking the primary site if the primary site is in available (0) status. After the primary site has been locked, the PL phase determines the status of each neighbour's site. Then, OQ phase declares that the quorum obtained is enough for the transaction to be continued. Next, the primary data will be updated in the U phase. Afterward, the updated primary data which is also called as new primary data is replicated to the neighbours' sites in C phase. Last but not least, the transaction will unlock (UL) all the sites that are involved in the transaction.

3.4.2 Context Diagram

Figure 3.4.2 below shows the context diagram of the Binary Vote Assignment Grid (BVAG) replication techniques prototype.

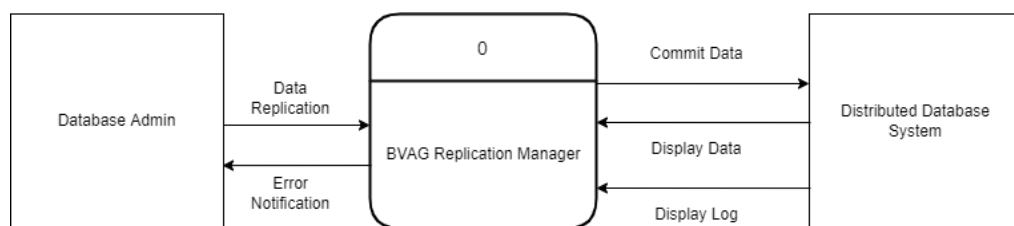


Figure 3.4.2: Context Diagram of BVAG

This BVAG Replication Manager has two entities which are Database Admin and the Distributed Database System. The DDS is responsible for displaying data and displaying activities log requested by BVAG. Meanwhile the Database Admin will execute the Data Replication and will receive error notification.

3.4.3 Framework Architecture

Figure 3.4.3 below shows the use case diagram of the Binary Vote Assignment Grid (BVAG) replication techniques prototype.

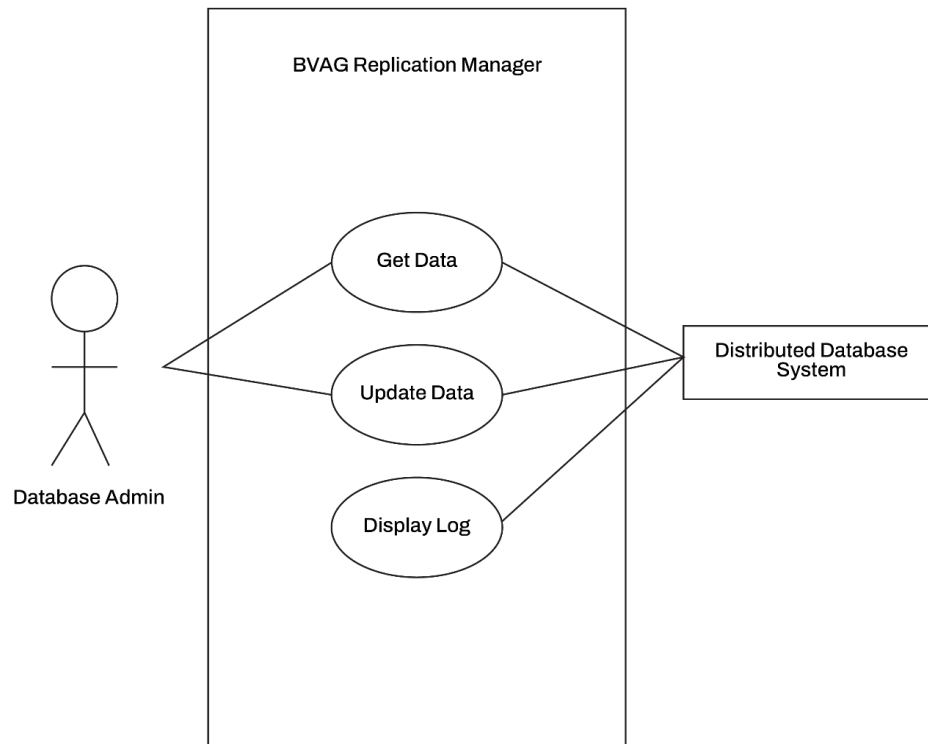


Figure 3.4.3: Use Case Diagram BVAG

Based on the use case diagram, there are a total of 3 use cases or modules in this system. Among them are Get Data module, Update Data module and Display Log module. The actors for this system are Database Admin and Distributed Database System. The database admin will be able to access the Get Data module and Update Data module while the DDS can access all module.

3.5 Design Prototype (Framework Architecture)

Figure 3.5.1 below shows the design prototype of the Binary Vote Assignment Grid (BVAG) replication techniques prototype.

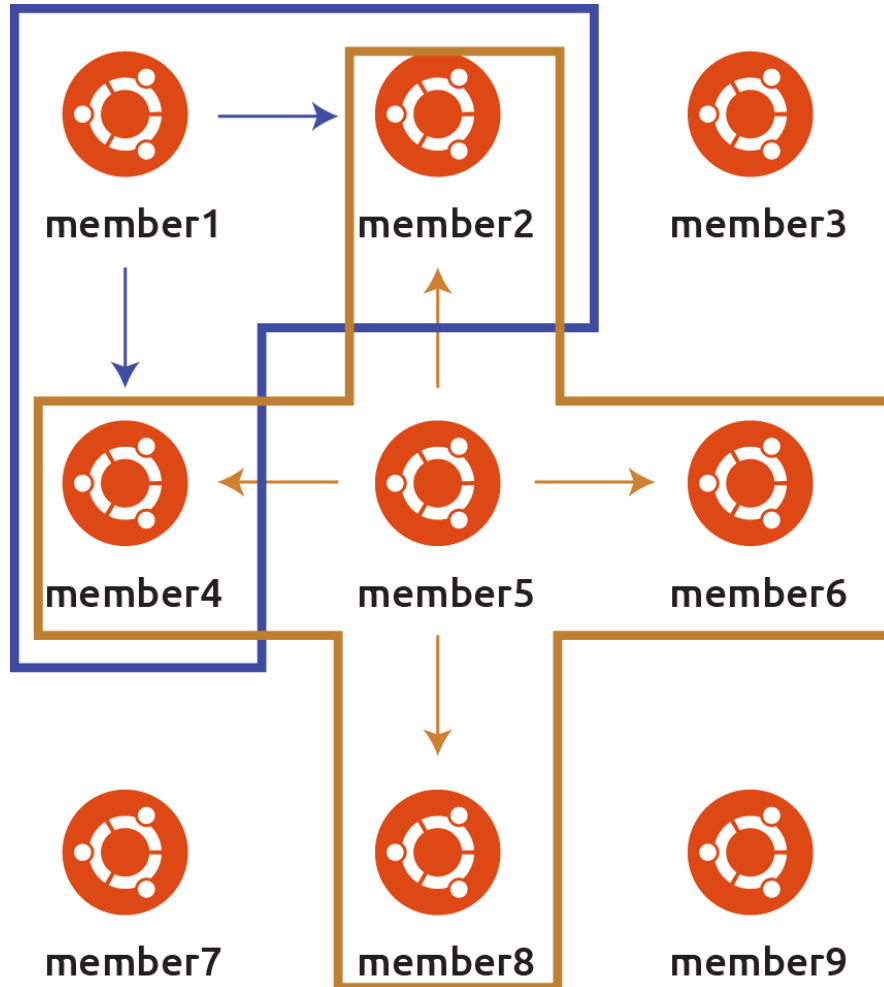


Figure 3.5.1: Design Prototype of BVAG

The prototype above shows the 9 x 9 grid for Ubuntu 20.04 LTS Linux server that will be configured to talk to their respective neighbours in order to do data replication. For this project, the server will be deployed on DigitalOcean Cloud Hosting with MySQL replication. As we can see in the Figure 3.5.1, each server is named member1 to member9 to represent the BVAG distributed database server grid.

3.6 Testing Plan

A test case is prepared in order to test the functionality of the BVAG replication techniques prototype. The reason behind having to conduct these test cases is to ensure that the system can run smoothly, and the end-user doesn't experience problem when using the system. Table 3.6.1 are the test cases that will be conducted on the system.

No.	Module	Activities	Status		Comments
1.	Get Data	System generate MySQL syntax	Yes <input type="checkbox"/>	No <input type="checkbox"/>	
		Display Data	Yes <input type="checkbox"/>	No <input type="checkbox"/>	
2.	Update Data	System initiate lock	Yes <input type="checkbox"/>	No <input type="checkbox"/>	
		System initiates propagate lock	Yes <input type="checkbox"/>	No <input type="checkbox"/>	
3.	Display Log	System display lock phases activities	Yes <input type="checkbox"/>	No <input type="checkbox"/>	
		System show time taken for each activity	Yes <input type="checkbox"/>	No <input type="checkbox"/>	

Table 3.6.1: User Acceptance Table BVAG Prototype

This test has been performed by:

Name : _____

Signature : _____

Date : _____

3.7 Potential Use of Proposed Solution

The potential use of the proposed solution is any organization or company can use the BVAG replication techniques prototype to implement it in a distributed database environment. This is to provide convenient approach to achieve data consistency for a distributed database. Furthermore, handling database update operations with less computational time is crucial for synchronous replication.

This prototype however does not consider failure cases so in the future and with more time spent, a fault tolerance approach can be developed and integrated for this replication techniques. This prototype also shows only a few fields to update data into different replication server. To make this more user friendly, various forms of input can be created. Therefore, many input type fields can be added in future. This proposed solution also can make a significant improvement on communication cost and faster transaction in a distributed database environment. The proposed solution ensures if one site goes down, other sites is still available.

3.8 Gantt Chart

The Figure 3.8.1 below is the Gantt Chart for the BVAG prototype, which is based on the project methodology, Agile Methodology.

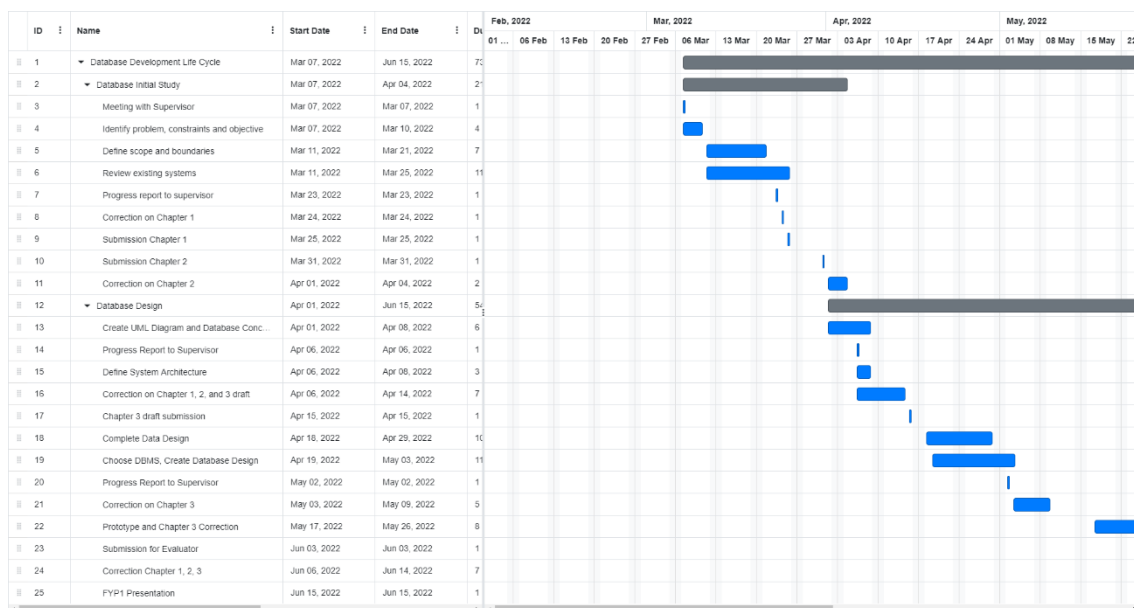


Figure 3.8.1: Gantt Chart

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Introduction

This chapter discuss about the development, implementation, and testing of this project. It includes the steps used to complete the system, method utilized, user interfaces, as well as result and discussions.

4.2 Hardware and Software Components

From the user's specification, the functionality offered by BVAG application is for database replication. BVAG prototype is deployed and tested on Digital Ocean Cloud Service Provider for this implementation. The implementation of BVAG requires some minimum hardware and software specifications. The hardware specifications are shown in Table 4.2.1 was used in each replication server for implementation. The software specifications are shown in Table 4.2.2 was used in each replication server.

Table 4.2.1: Server Hardware Specifications

Hardware	Apache Ignite
Processor	Intel vCPU
Memory	1 Gigabytes RAM
Solid State Disk	25 Gigabytes
Operating Systems	Ubuntu 20.04 (LTS) Linux Server x64

Table 4.2.2: Server Software Specifications

Software	Apache Ignite
Database Management System (DBMS)	MySQL
Visual Studio Code	Version 2022

4.3 BVAG PROTOTYPE IMPLEMENTATION

The implementation process is to record all the steps in developing Binary Vote Assignment Grid (BVAG) replication techniques prototype.

4.3.1 Prerequisites

To setup the BVAG replication techniques using MySQL, four Ubuntu 20.04 (LTS) Linux Server x64 will be used. Note that three is the minimum number of MySQL instances you need to deploy BVAG replication in MySQL, while nine is the maximum. All the virtual machine server will be deployed and hosted on Digital Ocean Cloud Service Provider, the distributed database can also be setup and deployed on localhost using at least 3 PC's using a network switch. Picture below shows the PCs in the faculty server cluster lab deployed using 3 PCs.



The Ubuntu virtual machine server will also each should have a non-root administrative user with **sudo** privileges, and a firewall configured with UFW. MySQL is also required to be installed to each server and using the latest version.

Each virtual machine server will also have their own unique IPv4 and IPv6 address with both public and private address which will be setup later.

4.3.2 Creating Hosted Ubuntu Server Virtual Machine

After registering with DigitalOcean Cloud Service, we need to setup the virtual machine server which is called Droplet in DigitalOcean. shows all the region that can be chosen for the server. Generally, we need to pick the closest region to our location, so we have a low latency server and fast speed. Hence, Singapore data center is chosen. Figure 4.3.1: Create Droplets - Choose Region shows the chose region page.

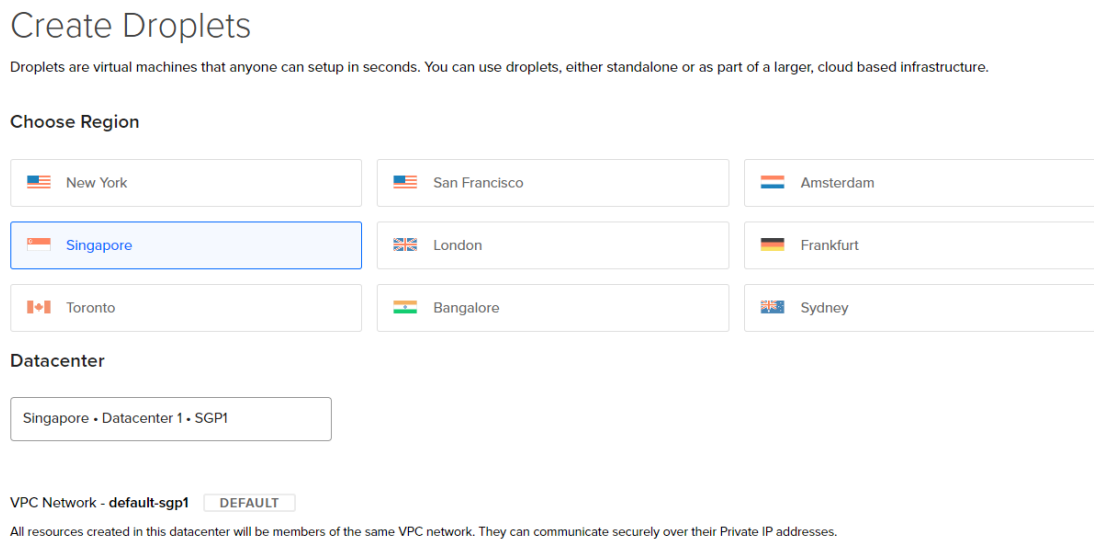


Figure 4.3.1: Create Droplets - Choose Region

After that, we need to choose the operating system in which we will run the MySQL server. Ubuntu 20.04 LTS x64 is chosen. Figure 4.3.2 shows the choose OS page.

Choose an image

OS Marketplace Custom images

The screenshot shows the 'Choose an image' interface. At the top, there are two tabs: 'OS Marketplace' (selected) and 'Custom images'. Below the tabs, there are six OS icons in a row: Ubuntu (highlighted with a blue border), Fedora, Debian, CentOS, AlmaLinux, and Rocky Linux. Below the icons, there is a 'Version' dropdown menu currently set to '20.04 (LTS) x64'.

Figure 4.3.2: Create Droplets - Choose OS

Next, the hardware specifications for the Ubuntu Server is chosen. Figure 4.3.3 shows the choose hardware specifications page. Since, this BVAG won't require too much resources for basic task, we will chose only Basic Intel vCPU, 1 GB RAM and 25GB SSD.

The screenshot shows the hardware specifications page. It is divided into two main sections: 'Droplet Type' and 'CPU options'.

Droplet Type: This section is divided into 'SHARED CPU' and 'DEDICATED CPU'. Under 'SHARED CPU', the 'Basic (Currently selected)' option is highlighted. Under 'DEDICATED CPU', there are four options: 'General Purpose', 'CPU-Optimized', 'Memory-Optimized', and 'Storage-Optimized'.

CPU options: This section shows three radio button options: 'Regular' (selected, Disk type: SSD), 'Premium Intel' (Disk: NVMe SSD, marked 'NEW'), and 'Premium AMD' (Disk: NVMe SSD, marked 'NEW').

Below the CPU options, there are six pricing plans in a row, each with a price per month and hour, and a list of specifications:

Price	Specifications
\$4/mo \$0.006/hour	512 MB / 1 CPU 10 GB SSD Disk 500 GB transfer
\$6/mo \$0.009/hour	1 GB / 1 CPU 25 GB SSD Disk 1000 GB transfer
\$12/mo \$0.018/hour	2 GB / 1 CPU 50 GB SSD Disk 2 TB transfer
\$18/mo \$0.027/hour	2 GB / 2 CPUs 60 GB SSD Disk 3 TB transfer
\$24/mo \$0.036/hour	4 GB / 2 CPUs 80 GB SSD Disk 4 TB transfer
\$48/mo \$0.071/hour	8 GB / 4 CPUs 160 GB SSD Disk 5 TB transfer

At the bottom right of the pricing plans, there is a 'Show all plans' link.

Figure 4.3.3: Create Droplets - Choose Hardware Specifications

After that, the password for the Ubuntu server is set. Figure 4.3.4 shows the set server password page.

Choose Authentication Method ?

SSH Key
Connect to your Droplet with an SSH key pair

Password
Connect to your Droplet as the "root" user via password

Create root password *

Type your password...

PASSWORD REQUIREMENTS

- Must be at least 8 characters long
- Must contain 1 uppercase letter (cannot be first or last character)
- Must contain 1 number
- Cannot end in a number or special character

⚠ Please store your password securely. You will not be sent an email containing the Droplet's details or password.

Figure 4.3.4: Create Droplets - Set Server Password

Finally, we can create the server. Figure 4.3.5 shows the Create Droplet button and the option to name the droplet.

Finalize Details

Quantity
Deploy multiple Droplets with the same configuration.

Droplet

Hostname
Give your Droplets an identifying name you will remember them by.

Tags

Project

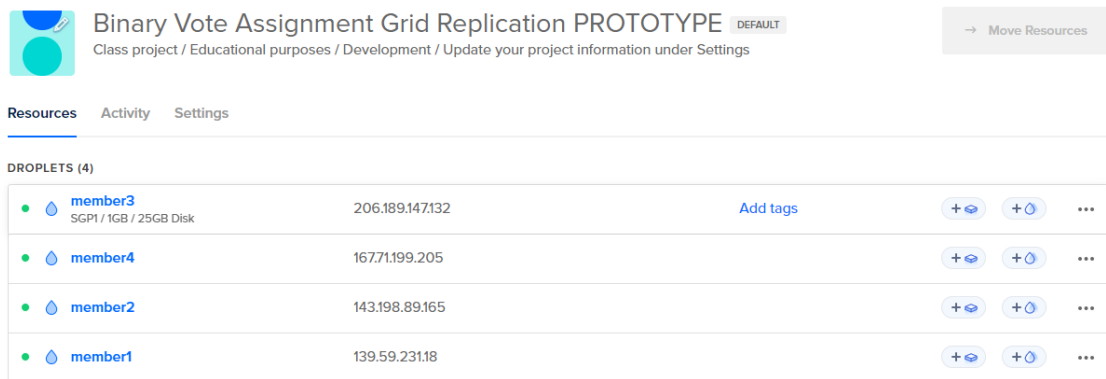
\$6.00/month
\$0.009/hour

[CREATE VIA COMMAND LINE](#)

Figure 4.3.5: Create Droplets

This process is repeated until at least 4 Ubuntu Virtual Machine Server created. After all the process repeated, we can see overall server that has been created with all the

public and private IP address assigned to each server. Figure 4.3.6 shows all the server that has been created.



The screenshot shows a cloud management interface for a project named "Binary Vote Assignment Grid Replication PROTOTYPE". The interface includes a header with the project name, a "Move Resources" button, and navigation tabs for "Resources", "Activity", and "Settings". Below the tabs, a section titled "DROPLETS (4)" displays a table of four servers. Each server row includes a status indicator (green dot), a name (member3, member4, member2, member1), a resource specification (SGP1 / 1GB / 25GB Disk), a public IP address, an "Add tags" link, and control icons for power, refresh, and a menu.

Member	Public IP
member3	206.189.147.132
member4	167.71.199.205
member2	143.198.89.165
member1	139.59.231.18

Figure 4.3.6: 4 Ubuntu Server Created

4.3.3 Initial Server Setup with Ubuntu 20.04

4.3.3.1 Logging in as root

To log into newly created server, we need to know the server's public IP address and the password. The server console is opened and logged in as root using the command, Figure 4.3.7 shows the command in the console typed.

ssh root@server_ip_address

```
root@member3:~# ssh root@10.104.0.5
The authenticity of host '10.104.0.5 (10.104.0.5)' can't be established.
ECDSA key fingerprint is SHA256:SSSxv+Q/VsRgT7DYYx9zG40gJcIsrLy+PoJAPwgaETk.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.104.0.5' (ECDSA) to the list of known hosts.
root@10.104.0.5's password:
Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 5.4.0-122-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Mon Jan 30 07:25:55 UTC 2023

System load:          0.0
Usage of /:           7.6% of 24.05GB
Memory usage:        26%
Swap usage:          0%
Processes:           108
Users logged in:     1
IPv4 address for eth0: 206.189.147.132
IPv4 address for eth0: 10.15.0.8
IPv6 address for eth0: 2400:6180:0:d0::c4:3001
IPv4 address for eth1: 10.104.0.5

32 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

New release '22.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

*** System restart required ***
Last login: Mon Jan 30 07:21:45 2023 from 162.243.190.66
root@member3:~#
```

Figure 4.3.7: Logged in as root

4.3.3.2 Creating a New User

Once logged in as root, we'll be able to add a new user account. In the next part, we will log in with this new account instead of root. For now, we will create a new user called "fathul", "muhammad", "amin" for each server. We need to use the below command to create the user. Figure 4.3.8 shows the command in the console typed. We also need to set the password for the user account as well as some user informations.

adduser frodo

```
root@member3:~# adduser frodo
Adding user `frodo' ...
Adding new group `frodo' (1000) ...
Adding new user `frodo' (1000) with group `frodo' ...
Creating home directory `/home/frodo' ...
Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for frodo
Enter the new value, or press ENTER for the default
  Full Name []: Frodo Baggins
  Room Number []:
  Work Phone []: 0987654321
  Home Phone []: 0987654321
  Other []:
Is the information correct? [Y/n] Y
root@member3:~#
```

Figure 4.3.8: Add New User

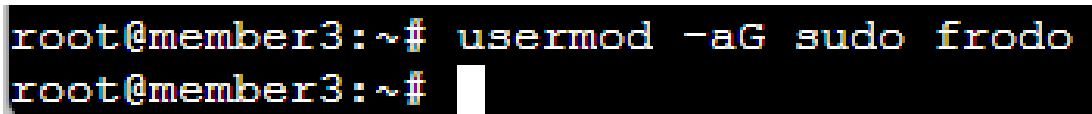
4.3.3.3 Granting Administrative Privileges

Now we have a new user account with regular account privileges. However, we may sometimes need to do administrative tasks. To avoid having to log out of our normal user and log back in as the root account, we can set up what is known as superuser or root privileges for our normal account. This will allow our normal user to run commands with administrative privileges by putting the word `sudo` before the command.

To add these privileges to our new user, we need to add the user to the sudo group. By default, on Ubuntu 20.04, users who are members of the sudo group are allowed to use the sudo command. Figure 4.3.9 shows the command ran in the console.

As root, we need to run this command to add the new user to the sudo group:

```
usermod -aG sudo frodo
```



```
root@member3:~# usermod -aG sudo frodo
root@member3:~#
```

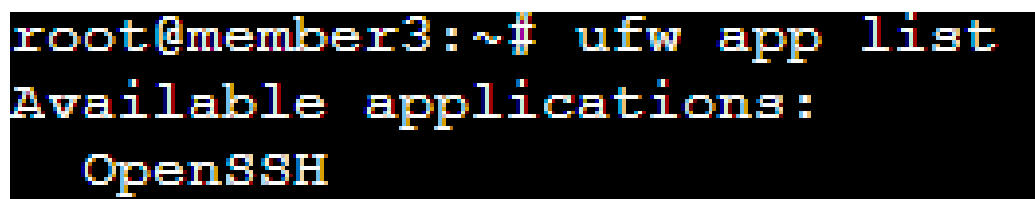
Figure 4.3.9: Granting Administrative Privileges

4.3.3.4 Setting Up a Basic Firewall

Ubuntu 20.04 servers can use the UFW firewall to make sure only connections to certain services are allowed. We can set up a basic firewall using this application. Applications can register their profiles with UFW upon installation. These profiles allow UFW to manage these applications by name. OpenSSH, the service allowing us to connect to our server now, has a profile registered with UFW.

We can see this by typing the below command, Figure 4.3.10 shows the output:

```
ufw app list
```



```
root@member3:~# ufw app list
Available applications:
  OpenSSH
```

Figure 4.3.10: UFW List

We need to make sure that the firewall allows SSH connections so that we can log back in next time. We can allow these connections by typing below commands. Figure 4.3.11 shows the output.

```
ufw allow OpenSSH  
ufw enable  
ufw status
```

```
root@member3:~# ufw allow OpenSSH  
Rules updated  
Rules updated (v6)  
root@member3:~# ufw enable  
Command may disrupt existing ssh connections. Proceed with operation (y|n)? y  
Firewall is active and enabled on system startup  
root@member3:~# ufw status  
Status: active
```

To	Action	From
--	-----	----
OpenSSH	ALLOW	Anywhere
OpenSSH (v6)	ALLOW	Anywhere (v6)

Figure 4.3.11: UFW Allow OpenSSH

4.3.4 Install MySQL on Ubuntu 20.04

4.3.4.1 Installing MySQL

On Ubuntu 20.04, we can install MySQL using the APT package repository. To install it, we need to update the package index on our server, after that we need to install the `mysql-server` package and lastly ensuring the MySQL server is running. We can do all this by running the below command and Figure 4.3.12, Figure 4.3.13 and Figure 4.3.14 shows the output.

```
sudo apt update  
sudo apt install mysql-server  
sudo systemctl start mysql.service
```

```
frodo@member3:~$ sudo apt update  
[sudo] password for frodo:  
Hit:1 http://mirrors.digitalocean.com/ubuntu focal InRelease  
Hit:2 http://mirrors.digitalocean.com/ubuntu focal-updates InRelease  
Hit:3 http://mirrors.digitalocean.com/ubuntu focal-backports InRelease  
Hit:4 https://repos-droplet.digitalocean.com/apt/droplet-agent main InRelease  
Hit:5 http://security.ubuntu.com/ubuntu focal-security InRelease  
Hit:6 https://repos.insights.digitalocean.com/apt/do-agent main InRelease  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
39 packages can be upgraded. Run 'apt list --upgradable' to see them.
```

Figure 4.3.12: `sudo apt update`

```
Renaming removed key_buffer and myisam-recover options (if present)  
mysql will log errors to /var/log/mysql/error.log  
mysql is running as pid 33506  
Created symlink /etc/systemd/system/multi-user.target.wants/mysql.service.  
Setting up libcgi-pm-perl (4.46-1) ...  
Setting up libhtml-template-perl (2.97-1)
```

Figure 4.3.13: `sudo apt install mysql-server`

```
frodo@member3:~$ sudo systemctl start mysql.service  
frodo@member3:~$ █
```

Figure 4.3.14: `sudo systemctl start mysql.service`

4.3.4.2 Configuring MySQL

For fresh installations of MySQL, we will run the DBMS's included security script. This script changes some of the less secure default options for things like remote root logins and sample users.

To run the security script, the below command will be used, Figure 4.3.15 shows the output.

```
sudo mysql_secure_installation
```

```
frodo@member3:~$ sudo mysql_secure_installation
Securing the MySQL server deployment.
Enter password for user root:
VALIDATE PASSWORD COMPONENT can be used to test passwords
and improve security. It checks the strength of password
and allows the users to set only those passwords which are
secure enough. Would you like to setup VALIDATE PASSWORD component?
Press y|Y for Yes, any other key for No: |
```

Figure 4.3.15: sudo mysql_secure_installation

This will take us through a series of prompts where we can make some changes to our MySQL installation's security options. The first prompt will ask whether we like to set up the Validate Password Plugin, which can be used to test the password strength of new MySQL users before deeming them valid.

If we elect to set up the Validate Password Plugin, any MySQL user we create that authenticates with a password will be required to have a password that satisfies the policy we select. The strongest policy level — which we can select by entering 2 — will require passwords to be at least eight characters long and include a mix of uppercase, lowercase, numeric, and special characters, for this project, we will just use No. Then proceed with set the password. Figure 4.3.16 shows the password has been set which is “1234”.


```
Press y|Y for Yes, any other key for No: N
Using existing password for root.
Change the password for root ? ((Press y|Y for Yes, any other key for No) : y
New password:
Re-enter new password:
```

Figure 4.3.16: Set Password MySQL

From there, we can press Y and then ENTER to accept the defaults for all the subsequent questions. This will remove some anonymous users and the test database, disable remote root logins, and load these new rules so that MySQL immediately respects the changes we have made. Once the script completes, our MySQL installation will be secured. We can now move on to creating a dedicated database user with the MySQL client.

```
testing, and to make the installation go a bit smoother.
You should remove them before moving into a production
environment.

Remove anonymous users? (Press y|Y for Yes, any other key for No) : y
Success.

Normally, root should only be allowed to connect from
'localhost'. This ensures that someone cannot guess at
the root password from the network.

Disallow root login remotely? (Press y|Y for Yes, any other key for No) : y
Success.

By default, MySQL comes with a database named 'test' that
anyone can access. This is also intended only for testing,
and should be removed before moving into a production
environment.

Remove test database and access to it? (Press y|Y for Yes, any other key for No) : y
- Dropping test database...
Success.

- Removing privileges on test database...
Success.

Reloading the privilege tables will ensure that all changes
made so far will take effect immediately.

Reload privilege tables now? (Press y|Y for Yes, any other key for No) : y
Success.

All done!
```

Figure 4.3.17: Accepting Security Questions

4.3.4.3 Testing MySQL

After all the steps done so far, MySQL should have started running automatically. To test this, we will check its status using the below command. Figure 4.3.18 shows the output.

systemctl status mysql.service

```
Frodo@member3:~$ systemctl status mysql.service
● mysql.service - MySQL Community Server
   Loaded: loaded (/lib/systemd/system/mysql.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2023-01-30 08:17:17 UTC; 24min ago
     Main PID: 33743 (mysqld)
    Status: "Server is operational"
     Tasks: 39 (limit: 1131)
    Memory: 363.3M
    CGroup: /system.slice/mysql.service
           └─33743 /usr/sbin/mysqld
```

Figure 4.3.18: MySQL Service Status Check

This means MySQL is up and running. We now have a basic MySQL setup installed on our server.

4.3.5 Configure MySQL BVAG Replication on Ubuntu 20.04

4.3.5.1 Generating a UUID to Identify the MySQL BVAG Nodes

Before opening the MySQL configuration file to configure the server nodes replication settings, we need to generate a UUID that we can use to identify the MySQL nodes we will be creating. We will use the below command to generate a valid UUID for the nodes. Figure 4.3.19 shows the output if we ran it on member3 nodes.

uuidgen

```
frodo@member3:~$ uuidgen  
1ad8c23d-4f10-45ea-b7ce-e2f1982325b9
```

Figure 4.3.19: Generate UUID member3 nodes

We need to generate it on member1 as we are assuming member1 are the primary server nodes. After running the command on member 1, we get the UUID

f69494fe-5505-4d54-b209-312c9c1e9e18

4.3.5.2 Setting Up BVAG Replication in the MySQL Configuration File

Now we are ready to modify MySQL's configuration file. The main MySQL configuration file on each MySQL server using the preferred text editor. Here, we will be using nano text editor provided by Ubuntu. We will run the below command. Figure 4.3.20 shows the output.

sudo nano /etc/mysql/my.cnf

```
GNU nano 4.8 /etc/mysql/my.cnf
#
# The MySQL database server configuration file.
#
# You can copy this to one of:
# - "/etc/mysql/my.cnf" to set global options,
# - "~/.my.cnf" to set user-specific options.
#
# One can use all long options that the program supports.
# Run program with --help to get a list of available options and with
# --print-defaults to see which it would actually understand and use.
#
# For explanations see
# http://dev.mysql.com/doc/mysql/en/server-system-variables.html
#
# * IMPORTANT: Additional settings that can override those from this file!
#   The files must end with '.cnf', otherwise they'll be ignored.
#

!includedir /etc/mysql/conf.d/
!includedir /etc/mysql/mysql.conf.d/

[ Read 21 lines ]
^G Get Help      ^C Write Out    ^W Where Is    ^R Cut Text    ^J Justify     ^C Cur Pos     M-U Undo
^X Exit          ^O Read File    ^N Replace     ^U Paste Text  ^G To Spell   ^L Go To Line  M-R Redo
```

Figure 4.3.20: Nano Text Editor

On Ubuntu, MySQL comes installed with a number of different files we can use to define various configuration changes. By default, the my.cnf file is only used to source additional files from subdirectories. We will have to add our own configuration beneath the !includedir lines. This will allow we to override any settings from the included files.

To begin, we start a new section by including a [mysqld] header and then add the settings we need to enable group replication, as highlighted in the following example. Note that these settings are modified from the minimum settings required for replication outlined in the official MySQL documentation. The loose- prefix allows MySQL to handle options it does not recognize gracefully and without failure. We will need to fill in and customize some of these settings.

The below script is inserted into the MySQL configuration file to setup the BVAG replication. Figure 4.3.21 shows the BVAG script.

```
[mysqld]
# General replication settings
disabled_storage_engines="MyISAM, BLACKHOLE, FEDERATED, ARCHIVE, MEMORY"
gtid_mode = ON
enforce_gtid_consistency = ON
master_info_repository = TABLE
relay_log_info_repository = TABLE
binlog_checksum = NONE
log_slave_updates = ON
log_bin = binlog
binlog_format = ROW
transaction_write_set_extraction = XXHASH64
loose-group_replication_bootstrap_group = OFF
loose-group_replication_start_on_boot = OFF
loose-group_replication_ssl_mode = REQUIRED
loose-group_replication_recovery_use_ssl = 1

# Shared replication group configuration
loose-group_replication_group_name = "f69494fe-5505-4d54-b209-312c9c1e9e18"
loose-group_replication_ip_whitelist = "10.104.0.2,10.104.0.3,10.104.0.4"
loose-group_replication_group_seeds = "10.104.0.2:33061,10.104.0.3:33061,10.104.0.4:33061"

# Single or Multi-primary mode? Uncomment these two lines
# for multi-primary mode, where any host can accept writes
loose-group_replication_single_primary_mode = OFF
loose-group_replication_enforce_update_everywhere_checks = ON

# Host specific replication configuration
server_id = 3
bind-address = "10.104.0.4"
report_host = "10.104.0.4"
loose-group_replication_local_address = "10.104.0.4:33061"
```

Figure 4.3.21: BVAG Replication Script Configuration

4.3.5.3 BVAG Script Explanation: Shared Replication Group Config

We need to set the **loose-group_replication_group_name** to the UUID value we generated previously with the `uuidgen` command on member1. Make sure we place the UUID between the empty pair of double quotes.

Next, we need to set **loose-group_replication_ip_whitelist** to a list of all of our MySQL server IP addresses, separated by commas. The **loose-group_replication_group_seeds** setting should be almost the same as the whitelist, but should append a designated group replication port to the end of each member. For the

purposes of this guide, we use the recommended group replication port, 33061. Figure 4.3.22 shows the changed setting on the script.

```
# Shared replication group configuration
loose-group_replication_group_name = "f69494fe-5505-4d54-b209-312c9c1e9e18"
loose-group_replication_ip_whitelist = "10.104.0.2,10.104.0.3,10.104.0.4"
loose-group_replication_group_seeds = "10.104.0.2:33061,10.104.0.3:33061,10.104.0.4:33061"
```

Figure 4.3.22: Shared Replication Group Config

4.3.5.4 Updating Each Server's UFW Rules

On each of our member servers, we need to open up access to both of these ports for the other members in this grid so they can all communicate with one another.

Command for member1:

```
sudo ufw allow from member2_server_ip to any port 3306
sudo ufw allow from member2_server_ip to any port 33061
sudo ufw allow from member3_server_ip to any port 3306
sudo ufw allow from member3_server_ip to any port 33061
```

Command for member2:

```
sudo ufw allow from member1_server_ip to any port 3306
sudo ufw allow from member1_server_ip to any port 33061
sudo ufw allow from member3_server_ip to any port 3306
sudo ufw allow from member3_server_ip to any port 33061
```

Command for member3:

```
sudo ufw allow from member1_server_ip to any port 3306
sudo ufw allow from member1_server_ip to any port 33061
sudo ufw allow from member2_server_ip to any port 3306
sudo ufw allow from member2_server_ip to any port 33061
```

4.3.5.5 Configuring Replication Users and Enabling Group Replication Plugin

In order to establish connections with the other servers in the replication grid, each MySQL instance must have a dedicated replication user.

On each of the MySQL servers, we need to log into MySQL instance with the administrative user to start an interactive session, the following command is run on each server nodes. Figure 4.3.23 shows the output.

sudo mysql

```
fathul@ubuntu-s-1vcpu-1gb-sgp1-02:~$ sudo mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.32-0ubuntu0.20.04.2 (Ubuntu)

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> █
```

Figure 4.3.23: sudo mysql

Because each server will have its own replication user, binary logging need to be turned off during the creation process. Otherwise, once replication begins, the server nodes would attempt to propagate the replication user from the primary to the other servers, creating a conflict with the replication user already in place. The following command from the MySQL was ran prompt on each of your servers.

SET SQL_LOG_BIN=0;

Now we can run a CREATE USER statement to create our replication user. We will run the following command, which creates a user named repl. This command specifies that the replication user must connect using SSL. Also, we need to make sure to use a secure password in place of password when creating this replication user. Figure 4.3.24 shows the output.

```
CREATE USER 'repl'@'%' IDENTIFIED BY 'password' REQUIRE SSL;
```

```
mysql> CREATE USER 'repl'@'%' IDENTIFIED BY 'password' REQUIRE SSL;  
Query OK, 0 rows affected (0.01 sec)
```

Figure 4.3.24: CREATE USER Syntax

Next, we need to grant the new user replication privileges on the server with the below command. Figure 4.3.25 shows the output.

```
GRANT REPLICATION SLAVE ON *.* TO 'repl'@'%';
```

```
mysql> GRANT REPLICATION SLAVE ON *.* TO 'repl'@'%;  
Query OK, 0 rows affected (0.01 sec)
```

Figure 4.3.25: Grant Replication Slave

Then, we need to flush the privileges to implement the changes and then re-enable binary logging to resume normal operations. Next, we need to set the group_replication_recovery channel to use our new replication user and their associated password. Each server will then use these credentials to authenticate to the nodes.


```
FLUSH PRIVILEGES;
SET SQL_LOG_BIN=1;
```

```
CHANGE REPLICATION SOURCE TO SOURCE_USER='repl',
SOURCE_PASSWORD='password' FOR CHANNEL 'group_replication_recovery';
```

```
mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.01 sec)

mysql> SET SQL_LOG_BIN=1;
Query OK, 0 rows affected (0.00 sec)

mysql> CHANGE REPLICATION SOURCE TO SOURCE_USER='repl', SOURCE_PASSWORD='password' FOR CHANNEL 'group_replication_recovery';
Query OK, 0 rows affected, 2 warnings (0.04 sec)
```

With the replication user in place, you can enable the `group_replication` plugin to prepare to initialize the server nodes. Verify that the plugin is active by running the following command. We can use the below command. Figure 4.3.26 shows the output.

```
INSTALL PLUGIN group_replication SONAME 'group_replication.so'
SHOW PLUGINS;
```

```
mysql> INSTALL PLUGIN group_replication SONAME 'group_replication.so';
Query OK, 0 rows affected (0.04 sec)

mysql> SHOW PLUGINS;
```

Name	Status	Type	Library	License
binlog	ACTIVE	STORAGE ENGINE	NULL	GPL
mysql_native_password	ACTIVE	AUTHENTICATION	NULL	GPL
sha256_password	ACTIVE	AUTHENTICATION	NULL	GPL
caching_sha2_password	ACTIVE	AUTHENTICATION	NULL	GPL
sha2_cache_cleaner	ACTIVE	AUDIT	NULL	GPL
daemon_keyring_proxy_plugin	ACTIVE	DAEMON	NULL	GPL
CSV	ACTIVE	STORAGE ENGINE	NULL	GPL
MEMORY	ACTIVE	STORAGE ENGINE	NULL	GPL
InnoDB	ACTIVE	STORAGE ENGINE	NULL	GPL

Figure 4.3.26: Output

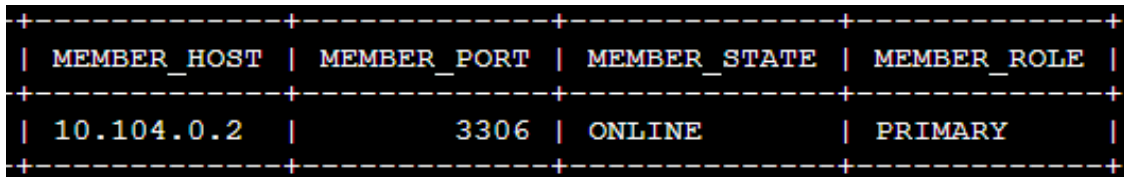
4.3.5.6 Starting BVAG Replication on First Node

Now that each MySQL server has a replication user configured and the replication plugin enabled, we can begin to bring up our server nodes in the grid using the command below.

```
START GROUP_REPLICATION;
```

The nodes grid will be started with this server as the only member. We can verify this by checking the entries within the replication_group_members table in the performance_schema database. Figure 4.3.27 shows the output.

```
SELECT * FROM performance_schema.replication_group_members;
```



MEMBER_HOST	MEMBER_PORT	MEMBER_STATE	MEMBER_ROLE
10.104.0.2	3306	ONLINE	PRIMARY

Figure 4.3.27: Group Replication Member 1 Status

The ONLINE value for MEMBER_STATE indicates that this node is fully operational within the server grids. Next we need to create a test database and table with some sample data. Once more members are added to this group, this data will be replicated out to them automatically. We will create a sample database called “playground”.

```
CREATE DATABASE playground;
```

```
CREATE TABLE playground.equipment (  
id INT NOT NULL AUTO_INCREMENT,  
type VARCHAR(50),  
quant INT,  
color VARCHAR(25),  
PRIMARY KEY(id)  
);
```

This table contains the following four columns:

- i) id: This column will contain integer values that increment automatically, meaning we won't have to specify values for this column when we load the table with sample data
- ii) type: This column will contain string values describing what type of playground equipment the row represents
- iii) quant: This column will contain integer values to represent the quantity of the given type of playground equipment
- iv) color: This column will hold string values specifying the color of the given equipment

Lastly, we run the following command to insert one row of data into the table. Then we will query the table to make sure the data is inserted correctly. Figure 4.3.28 shows the output.

```
INSERT INTO playground.equipment (type, quant, color) VALUES ("slide", 2, "blue");  
SELECT * FROM playground.equipment;
```

```
mysql> INSERT INTO playground.equipment (type, quant, color) VALUES ("slide", 2, "blue");  
Query OK, 1 row affected (0.01 sec)  
  
mysql> SELECT * FROM playground.equipment;  
+----+-----+-----+-----+  
| id | type | quant | color |  
+----+-----+-----+-----+  
|  1 | slide |     2 | blue  |  
+----+-----+-----+-----+  
1 row in set (0.00 sec)
```

Figure 4.3.28: Query from Table

After verifying that this server is a member of the server grid and that it has write capabilities, the other servers can join the server grid.

4.3.5.7 Starting BVAG Replication on remaining Node

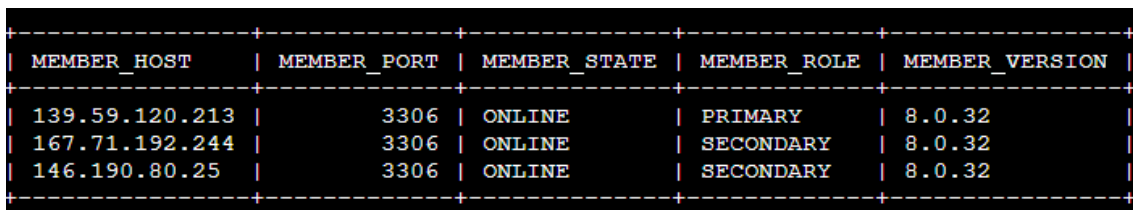
Next, we will start group replication on member2. Since we already have an active member, we don't need to bootstrap the server grid and this member can join straightaway.

```
START GROUP_REPLICATION;
```

On member3, we start group replication the same way.

Then, we will check the membership list again on any of the three servers. This time, there will be three servers listed in the output. We will check it on member2 nodes. Figure 4.3.29 shows the output.

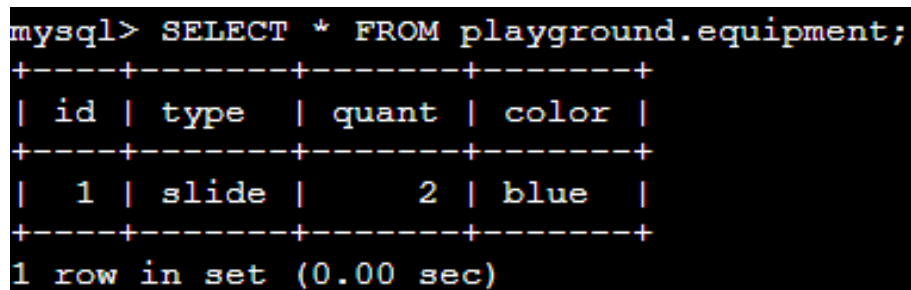
```
SELECT * FROM performance_schema.replication_group_members;
```



MEMBER_HOST	MEMBER_PORT	MEMBER_STATE	MEMBER_ROLE	MEMBER_VERSION
139.59.120.213	3306	ONLINE	PRIMARY	8.0.32
167.71.192.244	3306	ONLINE	SECONDARY	8.0.32
146.190.80.25	3306	ONLINE	SECONDARY	8.0.32

Figure 4.3.29: Check Membership Replication

Now, we will check if the replication is working by checking the database information that we created on the secondary server (member2) using the below command. Figure 4.3.30 shows the results.



```
mysql> SELECT * FROM playground.equipment;  
+----+-----+-----+-----+  
| id | type  | quant | color |  
+----+-----+-----+-----+  
|  1 | slide |     2 | blue  |  
+----+-----+-----+-----+  
1 row in set (0.00 sec)
```

Figure 4.3.30: Successful Replicate Data using BVAG

4.4 BVAG TESTING

Next, we can try testing to write to the database from our new replication grid. We can try to insert some data into the existing table by using the below syntax. Figure 4.4.1 shows the output and results.

```
INSERT INTO playground.equipment (type, quant, color) VALUES ("swing", 10, "yellow");
```

```
mysql> SELECT * FROM playground.equipment;
+-----+-----+-----+-----+
| id | type  | quant | color  |
+-----+-----+-----+-----+
| 1  | slide | 2     | blue   |
| 2  | swing | 10    | yellow |
+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

Figure 4.4.1: Data replicated on member2

From here we can see the data is successfully replicated to member2 from member1.

4.5 RESULT AND CONCLUSION

The test conducted on the system have been very successful, even though there are more rooms for improvement and amendments that can be made into the system. From the result and testing, it is concluded that the BVAG works as proposed and able to replicate data to other server as proposed.

In conclusion, even though this system is able to work, there are many improvements that can be made with this system overtime to ensure that it keeps up with the current software and hardware to ensure the process between this system can run smoothly

CHAPTER 5

CONCLUSION

5.1 Introduction

This chapter, will discuss on the closure of the development of this system. We will discuss the limitations and constraints in this system and the future work that can be done for this. This work has been addressed using Binary Vote Assignment Grid replication techniques to produce a highly-available and fault tolerant in managing replication. It also includes some suggestions for future work in each of the areas covered during this research.

5.2 Limitation and Constraint

There are some difficulties that have been through during the development of this system. Below are some of the key points of the issues that have limited or given constraints to the development of the system.

i) Lack of development time

The lack of development time has caused some hindrances in the development of the system as there are some other responsibilities that needed to be taken care of and also the lack of understanding has cause more time needed to understand the topic which took up more time then the development of the system.

ii) Lack of resources

The lack of resources has caused some constraint in the development of the system. There are not many published papers related to the project. Especially when going to the developers' GitHub or website they have lack of information on installation, or implementing them or some projects of BVAG replication techniques.

iii) Lack of Obtain Quorum, Initiate and Propagate Lock phase

The lack of said phases is due to the lack of knowledge and the lack of development time as well as the lack of access to the hardware at the cluster lab due to the problem in faculty management. If the development had more time, it is possible to implement the quorum, initiate and propagate lock phase.

5.3 Future Work

BVAG can be improved in many different ways. As we know, server failure can happen anytime. In future, it can make a significant improvement for commercial usage. A web-based BVAG can also introduce for customizing replication service.

It is hoped also that the BVAG prototype can be expanded much larger so it can work more efficiently and replicate data at a much faster speed while also making sure it is fault-proof and highly-available.

Other than that, BVAG can be improved in the future by implementing the BVAG replication techniques in a web-based application in order to monitor in real time the process of data replication as well as able to extract the time taken for each executed task.

REFERENCES

- Ahmad. (2010). Data Replication Using Read-One-Write-All Monitoring Synchronization Transaction System in Distributed Environment. *Journal of Computer Science*, 6(10), 1095–1098. <https://doi.org/10.3844/jcssp.2010.1095.1098>
- Ahmad, N., Noraziah, A., Deris, M. M., Ahmed, N. A., Saman, M. Y. M., Norhayati, R., & Alfawaer, Z. M. (2007). Preserving Data Consistency through Neighbor Replication on Grid Daemon. *American Journal of Applied Sciences*, 4(10), 751–758.
- Avram, A. (n.d.). *Geographically Distributed Database Management at the Cloud's Edge*.
- Azila, A., Fauzi, C., Fariza, W., Rahman, W. A., Fauzi, A., & Weigelt, F. (2021). Managing Fragmented Database in Distributed Database Environment. In *Journal of Mathematics and Computing Science* (Vol. 7, Issue 1).
- Budiarto, Nishio, S., & Tsukamoto, M. (2002). Data management issues in mobile and peer-to-peer environments. *Data & Knowledge Engineering*, 41(2–3), 183–204. [https://doi.org/10.1016/S0169-023X\(02\)00040-X](https://doi.org/10.1016/S0169-023X(02)00040-X)
- Deris, M. M., Abawajy, J. H., Taniar, D., & Mamat, A. (2009). Managing data using neighbour replication on a triangular-grid structure. *International Journal of High Performance Computing and Networking*, 6(1), 56. <https://doi.org/10.1504/IJHPCN.2009.026292>
- Foster, I., Kesselman, C., & Tuecke, S. (2001). The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *The International Journal of High Performance Computing Applications*, 15(3), 200–222. <https://doi.org/10.1177/109434200101500302>
- Francine Berman, Geoffrey Charles Fox, & Tony Hey. (2003). *Grid Computing: Making The Global Infrastructure a Reality* (F. Berman, G. Fox, & T. Hey, Eds.). John Wiley & Sons, Ltd. <https://doi.org/10.1002/0470867167>
- Linesch, M., & Marketing, H. P. (2007). *GWD-I*.
- Pérez, J. M., García-Carballeira, F., Carretero, J., Calderón, A., & Fernández, J. (2010). Branch replication scheme: A new model for data replication in large scale data grids. *Future Generation Computer Systems*, 26(1), 12–20. <https://doi.org/10.1016/j.future.2009.05.015>

- Sathya, S. S., Kuppaswami, S., & Ragupathi, R. (2006). Replication strategies for data grids. *Proceedings - 2006 14th International Conference on Advanced Computing and Communications, ADCOM 2006*, 123–128. <https://doi.org/10.1109/ADCOM.2006.4289868>
- Sathya, S. S., & Seshu, K. N. (2008). Synchronous Replica Consistency Protocol with Notification and Response. *2008 International Conference on Information Technology*, 71–74. <https://doi.org/10.1109/ICIT.2008.50>
- Tarun, S., Batth, R. S., & Kaur, S. (2019). A Review on Fragmentation, Allocation and Replication in Distributed Database Systems. *2019 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE)*, 538–544. <https://doi.org/10.1109/ICCIKE47802.2019.9004233>
- Ubaidillah, S. H. S. A., Alkazemi, B., & Noraziah, A. (2021). An Efficient Data Replication Technique with Fault Tolerance Approach using BVAG with Checkpoint and Rollback-Recovery. *International Journal of Advanced Computer Science and Applications*, 12(1). <https://doi.org/10.14569/IJACSA.2021.0120155>
- Voicu, L. C., Schuldt, H., Breitbart, Y., & Schek, H.-J. (2009). Replicated data management in the grid. *Proceedings of the 1st ACM Workshop on Data Grids for EScience - DaGreS '09*, 7. <https://doi.org/10.1145/1531786.1531789>

APPENDIX A
SAMPLE APPENDIX 1

For Appendices Heading, use TITLE AT ROMAN PAGES style.

APPENDIX B
SAMPLE APPENDIX 2

For Appendices Heading, use TITLE AT ROMAN PAGES style.