Cluster Replication

Comparison of cluster topologies in a simulated telecom environment.

Master's Thesis in Computer Science

Kristin Aanby

May 13, 2014

Østfold University College www.hiof.no

Abstract

The main focus of this thesis is to look at the advantages and disadvantages of the two selected topologies and how they are able to handle a lot of traffic with a reduced bandwidth or high latency.

This thesis looks at different replication strategies in today's market in connection with telecom for a greenfield mobile telecom company named IPCO. There are many replication topologies to choose between, and different DBMS systems support different replication strategies. MySQL only supports lazy replication between sites, but has come up with a cluster solution to support eager replication between sites within the cluster. Based on the two selected replication techniques, two cluster test environments have been established on MySQL7.3.3. The first test environment is named Big Cluster and it uses eager replication. The second test environment is named Asynchronous Replication and it uses a combination of eager and lazy replication. Two test cases are executed on the test environments and set out to answer the question "Given the two cluster topologies, in what way are the characteristics different?"

The analysis performed in this thesis is based on document analysis of MySQL documentation, research documents, articles and statements from actors in the marked in general. The analysis is also based on the results of the replication topology simulation.
This thesis has two main contributions: one for database replication in general and one specific to IPCO.

Preface

This master thesis is the final project for finish my Master's Degree in Computer Science at Østfold University College.
I wish to thank my thesis supervisors Harald Holone at Østfold University College, and Jens Erik Rive and Rolf Røsok at Morecom for their advise, support and encouragement. They have shown patience and interest in my studies beyond what could be expected. Special thanks to my supervisors at Morecom for allowing me to complete my trainee program at Morecom as a part of my master's program.
I would also like to thank my family for all their support through this period.

Halden May 5, 2014

Kristin Aanby

Abbreviations and Acronyms

IPCO	Internet Protocol Communication
MNO	Mobile Network Operator
MVNO	Mobile Virtual Network Operator
VSS	Visitor Server Subsystem
VOV	Virtual Operator Visitor
IMSI	International Mobile Subscriber Identity
HLR	Home Location Register
NTP	Network Time Protocol
JDBC	Java DataBase Connectivity
OS	Operating System
TC1	Test case 1
TC2	Test case 2
C1	Cluster1 in Asynchronous Replication test environment
C2	Cluster2 in Asynchronous Replication test environment
$\mathbf{C1_0}$	Cluster1.0 in Big Cluster test environment
$\mathbf{C1_1}$	Cluster1.1in Big Cluster test environment

Contents

Chapter 1

Introduction

The telecom market is constantly changing. People are much more dependent on their cellphone now and feel like they have to be available at all time. It has become more common to travel, both in Norway and abroad according to Statistisk sentralbyrå (SSB) [1]. People bring their cellphone almost everywhere. Since more people are using their cellphones abroad, there is an increased need to transfer large amounts of subscriber data over long distances. For the subscribers it can be expensive to use their cellphone abroad, and a substantial cost can incur. The use of cellphone abroad means that one must adhere to agreements that are made across the home country mobile network operator (MNO) and the visiting countries MNO. IPCO is a telecom provider which will try to reduce this cost by allowing their partners to roam as a local subscriber, with local charges.

1.1 Motivation

The author's personal motivation for writing this master thesis is her interest in databases and her wish to learn more about this, especially replication over long distances and over different geographical areas. Replication is a mechanism that is used a lot in today's society, and it is a central topic for distribution of data. IPCO's motivation for using database replication is that it is a good way to ensure fault-tolerance of the IPCO system and to increase performance, availability and scalability of the data.

1.2 Method and Research Question

The method used for this thesis is a combination of qualitative and quantitative methods based on document analysis and the results of replication topology simulation.
IPCO is a telecom provider which require carrier grade performance from their database system. This means requirements for high availability, high performance and replication demands, because IPCO has operators in several countries. Network latency and bandwidth are important factors when it comes to distribution of data. These factors can affect the performance, data consistency and replication delay, which can result in slower distribution of data. In worst case it can result in loss of data. Based on the requirements there are two different cluster topologies that are relevant for this project.
To see how replication behaves when the network is affected by high latency and low bandwidth, the research question and sub-research questions this thesis is going to answer are:

- Given the two cluster topologies, in what way are the characteristics different?

[1] https://www.ssb.no/transport-og-reiseliv/statistikker/reise/kvartal/2014-03-06#content

1A: In what way does bandwidth and latency affect the performance on the two cluster topologies.

1B: In what way does bandwidth and latency affect the replication delay on the Asynchronous Replication topology.

1C: Which combinations of bandwidth and latency can cause error situations like package lose, cluster downtime or replication downtime.

1.3 History of telecom

Telecommunication is a compound of the Greek prefix "tele-", meaning far off and Latin "communication" meaning to share. The history of telecommunication is an important part of the larger history of communication, and dates back to the use of smoke signals and drums in Africa. The first breakthrough into modern electrical telecommunication was the development of the telegraph in the 1830's. The modern world of telecom includes use of electrical devices such as telephones. In the beginning of the1990's there was a revolution in wireless telecommunication. This revolution revolved around the pioneering developments in radio cellphone communications. [2]

1.4 Today's mobile telecom market

In the mobile market today there are two types of mobile operators, Mobile Network Operators (MNOs) and Mobile Virtual Network Operators (MVNOs). MNOs are operators which own a mobile network, which means that they can provide services to mobile phone subscribers. The MNOs in Norway are Telenor, NetCom and Tele2. The MVNOs are operators that do not own a mobile network and are depending on a business agreement with a MNO to rent their mobile network. There are over 600 active MVNOs operated by over 360 companies in the world, per February 2010[2]. Some of the MVNOs in Norway are Chess, Ventelo, OneCall, Lyca mobile, Talkmore etc.

As the situation in the mobile market is today, it is difficult for the MVNOs to compete in the market on the same terms as the MNOs. This is because of the roaming agreement in the market. A roaming agreement is a mutual agreement between network operators. Roaming [3] gives the mobile customers access to mobile services (ie. to call and receive calls, to receive and send SMS and MMS and to use data services) in another mobile network. In order to offer a global roaming coverage for their subscribers, a mobile operator signs a large number of roaming agreements. As an alternative to sign roaming agreements with operators in all countries around the globe, a mobile operator can offer roaming through roaming hubs [1].

From the mobile operators point of view, there are two different types of roaming in the market, outbound roaming and inbound roaming. When people are traveling in foreign countries, inbound roaming gives them the opportunity to roam in the country they are visiting. Only MNOs are able to handle inbound roaming because MVNOs do not have their own mobile network to route this traffic on. When people visit Norway they have to roam on one of the Norwegian MNOs mobile network. This is either Telenor's-, NetCom's- or Tele2's mobile network, but which of these three networks they can use, depends on the agreement between the traveler's local home operator and

[2]http://www.prepaidmvno.com/definitions/#0002 (26.04.2013)
[3]http://en.wikipedia.org/wiki/Roaming

the MNO in Norway.
Outbound roaming occurs when a subscriber is traveling in a foreign country and roams on a foreign mobile network. Customers that are traveling can roam on all foreign mobile networks which their home MNO has an outbound roaming agreement with. If your home operator is a MVNO, the outbound roaming agreement depends on the agreement that your serving MNO has.

MNOs have access to both inbound and outbound roaming and gets an income when:

- Their own subscribers uses the mobile network at home.
- Their own subscribers uses outbound roaming.
- Subscribers who belong to a MVNOs that they are serving, like mobile network at home.
- Subscribers who belong to the MVNOs that they are serving outbound roaming.
- Subscribers who are travelling abroad use inbound roaming on the MNOs mobile network.

MVNOs have only access to outbound roaming and get an income when:

- Their own subscribers use the mobile network at home.
- Their own subscribers use outbound roaming when they are travelling.

As seen above, it is more difficult for the MVNOs to compete in the market in line with the MNOs. IPCO's idea is to offer a service to all the MVNOs, so that they can get into the market and compete on the same terms as the MNOs.

1.5 IPCO

Visitor Server Subsystem (VSS) is a system offered by IPCO. The VSS is offering a new service called "roam as a local" to MVNOs which traditionally have to rely on incumbent telecom operators for international roaming services. The "roam as a local" service includes:

- Inbound roaming to MVNOs.
- Cheaper outbound roaming to MVNOs.
- International calls between inbound and outbound MVNO through IPCO's international IP communication network.

The VSS takes the mobile operator visitor concept one step further by introducing a Virtual Operator Visitor (VOV). A VOV who travels to a foreign country is registered as a local MVNO subscriber in the visited MNO network, and is automatically given a visitor International Mobile Subscriber Identity (IMSI). The visitor IMSI allows the visitor to present himself as a local MVNO subscriber, and he receives the "roam as a local" service from the local MVNO. The VOV is authenticated through the VSS towards his home operator as a roaming subscriber, thus being able to use his own mobile telephone number.

The VSS architecture

The main focus is going to be the database system which is a part off the VSS system.

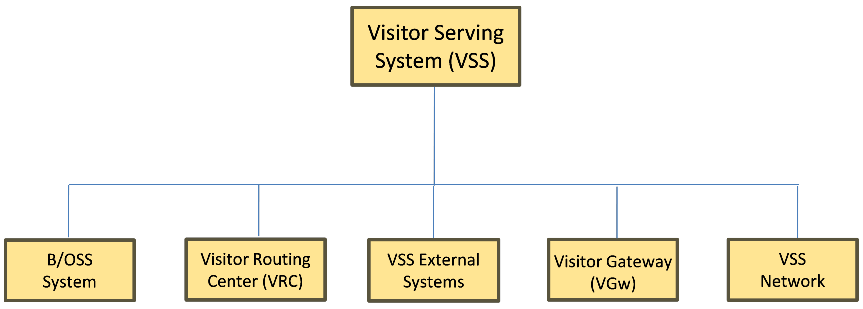


Figure 1.1: *IPCO VSS Subsystem*

Figure 1.1 shows the architecture of the VSS subsystem. The system consists of a B/OSS system, the Visitor Routing Centre (VRC), Visitor Gateway (VGw), and the VSS Network.
IPCO's database system is further down in the VSS architecture, and into the B/OSS System. The B/OSS System is a subsystem to the VSS system.

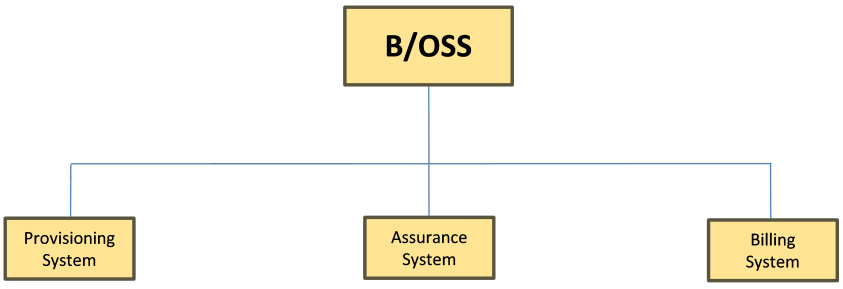


Figure 1.2: *B/OSS System*

The B/OSS System consists of the following subsystems; Provisioning System, Assurance System and Billing System. The Provisioning System consists of a VSS configuration application and a VSS database. The VSS database is the focus of this project[1].

IPCO has different requirements to the database system. Different replication techniques are going to be looked at to see if some of these can accommodate IPCO's requirements.

1.6 Existing systems

There is heavy competition between mobile operators, both MNOs and MVNOs, when it comes to roaming prices with MNO making most of the revenue. Many mobile operators in Norway have good deals for voice and SMS to countries within Scandinavia and some also outside Scandinavia. However, none of the mobile operators especially not the MVNOs, have the opportunity to make good roaming deals for their customers outside Norway. This is also reflected in other countries. Alternatives to cheap voice and SMS in a foreign country is to use services like Skype, Lync, OfficeSIP and other chat services, but they all require internet access and they can not be integrated using the subscribers mobile phone number in a transparent manner.

1.7 Database history

Connolly et. al. have stated the following about database research:

> The history of database research over the last 30 years is one of exceptional productivity that has led to the database system becoming arguably the most important development in the field of software engineering [3].

Today, databases are an important part of the IT infrastructure. Both companies and state institutions rely on database systems to store most of their data. The database has now become the underlying framework of the information system since it is used to store and access data. This has made the society to become more and more dependent on it. Over the last years the development of technologies have created systems that are more powerful and more intuitive to use. Examples of technologies improvements are; RAM has become cheaper and CPUs more powerful in memory solutions. This development has resulted in increasing availability of database systems for a wider variety of users. A workshop was held in the beginning for the 1990s, and a subsequent workshop was held in 1996. The workshops emphasized that developments in database system was not over, as some people thought. In fact, to paraphrase an old saying, it may be that we are only at the end of the beginning of the development. The application they have to handle in the future are much more complex and many of the algorithms that are being used today have to be reconsidered, like file storage, file access and query optimization. [3] [9]

Since society is more and more dependent on database systems in today's IT infrastructure, it is important to have a fault-tolerant system. One way to ensure fault-tolerance of a system is by using replication technology. [9]

1.7.1 What is replication

Replication is the process of generating and reproducing multiple copies of data at one or more sites, and keep the copies synchronized so they behave as identical as possible. Data means file, file system, database and so on. [3] Replication is a natural way of dealing with failures, if one replica fails, another takes over.

> The term "replica" mostly refers to a site running the database system software and storing copy of the entire database. [5]

The purposes of replication are multiple: [16, Chapter 13]

- Server availability - By using replication, data is available on multiple sites and this helps to avoid single point of failure. Even when some sites are down, the data can still be accessed from other sites.
- Performance - Replication allows us to locate data closer to their access points, thereby the response time will be reduced since the data is closer.
- Scalability - As systems grow geographically, the number of sites and the number of access request will increase. Replication supports this growth with an acceptable response time.

As shown above, data replication has clear benefits. When it comes to the design choices, there are two alternatives; full replication or partial replication. Full replication means that the whole state is available at all replicas, while partial replication means that each replica contains only a subset of the state. In other words full replication means that all replicas have a full copy of the database, and with partial replication each data item of the database has a physical copy only on a subset of sites. [5] The design choice can influence the performance and practicality of the replication solution.

1.7.2 Database replication

Database replication had its first boom in the early 80s, and has been extensively studied in the past years. Database replication is the distribution of information across a series of database servers that may be located in the same place or scattered across the globe. All servers are sharing the same level of information. There are no limitations when it comes to the replication model, any number of databases can take place into this model. [4]
From a database administrator's perspective, database replication requires that all data is kept synchronized without the need for user intervention and that the data is available where and when it is needed. [4]

1.8 Report structure

Chapter 1 is the introduction chapter of the report. It contains a brief introduction of the telecom market, the IPCO system and database history. Chapter 2 Background, the main focus of this chapter is replication techniques in general, MySQL replication and network theory.
Chapter 3 Method, this part of the thesis describes the research question, how the tests will be executed and how the results of the tests will be analyzed. Chapter 4 Experimental setup, describes IPCO's demands, demands to the database system and choice of database. Two test environments are used for testing in this thesis, the lab setup of these environments and the test preparations are described in detail. Chapter 5 Result, describes how the tests are executed, how the data is collected and a summary of all the results. Chapter 6 Discussion, is divided into two parts; the first part is a database technical discussion based on the results from the previous chapter and theory. The second part is a recommendation of the two test environments to IPCO. Chapter 7 Conclusion and Future work, this chapter answers the main research question, first for databases in general and then for IPCO. This chapter also looks at future work for this thesis.

[4] http://frext.sourceforge.net/replication-concepts.html

Chapter 2

Background

This chapter is structured by following a description of replication in general where the two- and three parameter classification are looked at, MySQL replication in general and cluster replication, followed by network theory.

2.1 Replication

There are many reasons why replication should be used, like fault-tolerance of the system and high-availability. The primary purpose of database replication is to increase the performance and improve the scalability of the database engines.

> Having more database replicas distributed across geographic regions provides fast access to local copies, having cluster of database replicas provides throughput and for some applications replication is a natural choice, e.g., in the context of mobile users that are frequently disconnected from the corporate data server. [5]

Replication is a useful component in backup and disaster recovery strategies. The threat of natural disasters is always present, regional power outages occur, building electrical and mechanical systems can fail. Often when a disaster occurs, organizations are left with day-old backups to relay on. By using data replication the organization can enable their backup systems to contain the latest data up to the point of the disaster. It is obvious that site replication is not enough for critical applications to ensure that even when a disaster occurs, critical applications and data remains available. Geographic replication can make this possible by having their database servers in more than one geographical location. [8]

2.1.1 Replica control mechanisms

Kemme et. al. refers to a replica as following:

> The term "replica" mostly refers to a site running the database system software and storing a copy of the entire database. [5]

The main task of replica control is to translate the operation that transactions submit on the logical data items into operations on the physical data copies. It will also make sure that the copies are consistent. In reality there are many different levels of consistency.
A main challenge of database replication is replica control: when data is updated, replica control is in charge of keeping the copies consistent and providing a globally correct execution. [5] The standard protocol for concurrency control is two Phase-Locking (2PL), but there are also several other ways to ensure concurrency control. See [5] for a detailed description.

There are several ways to replicate information across a network. Here are examples of some commonly used techniques. [4] [19]

- Dump and reload: The data is extracted "dumped" from one location and loaded "reloaded" into one or more destinations. From the creation of new data and until this data is propagated to one or more destinations it can take hours, days or even longer time. There is a risk that business units can make decisions based on data that may be days or weeks old.
- Two-phase commit (2PC): This protocol is used in the two parameter classification for Eager Update Anywhere and Eager Primary Copy, see 2.1.2. A transaction will only be accepted if all sites agree on it. This is how data is replicated inside the MySQL cluster. [28]
- Snapshot: This method is more advanced than "dump and reload". The snapshot method takes an image of the data at a given point in time, this image is loaded up to one or several sites. This method has some drawbacks, one of them is that this is not an appropriate technique when real-time data is needed.
- Database triggers: This mechanism is divided into two, with transactions and non-transactions. A trigger copies data to one or several remote locations.
- Log-based: This technique is used in MySQL for Lazy Update Anywhere and Lazy Primary Copy, and this is usually the most efficient and cost effective way of delivering replication.
- Group communication (GC): This protocol is widely used in conjunction with distributed systems but it can also be used for database replication. GC has an ordering and a delivery guarantee. The two main GC primitives are Atomic Broadcast (ABCAST) and View Synchronous Broadcast (VSCAST). For a more detailed description see [17] and [7].

As shown above there are several ways to transfer data between sites. The protocol 2PC is going to be used in the Big Cluster test environment, to replicate data within the cluster. The Asynchronous Replication test environment is going to use the log-based replication technique for the lazy replication between clusters and the protocol 2PC for the eager replication within the clusters. See chapter 4, Experimental setup for a detailed description of the test environments.

2.1.2 Replication Techniques

A replication technique describes how and when data is going to be transferred from one site to one or several other sites. In the past years there has been a rapid development in database replication, a plethora of replication protocols have been proposed, some of them with only subtle differences. A large variety of replication techniques have been proposed, yet the relationship between those techniques are not always clear. Similar techniques can behave different because they use different models and terminologies. To better understand database replication and the different replication techniques, the literature has proposed a two parameter classification and a three parameter classification of the replication techniques. P. Koszalinska et. al. [18] have created a short overview of the various classifications in their publication.

Two parameter classification

Gray et. al. [10] have come up with a two parameter classification, this is a well-known classification of database replication techniques. The advantage of this classification is that it makes it possible to see which techniques that are related. This classification of the replica control solution uses two parameters: transaction location and synchronization strategy. In other words; where and when updates take place. The two parameter classification is also described by other authors such as M. T. Özsu et. al, M. Wiesmann et. al and B. Kemme et. al. All have described the two parameter classification in the same way, but M. T. Özsu et al. are using the name “centralized“ and “distributed” instead of Primary copy and Update anywhere.

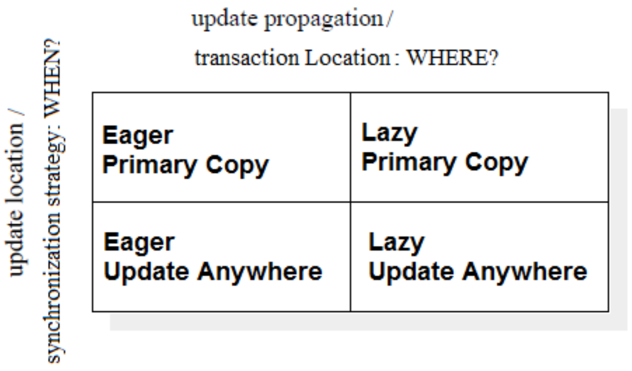


Figure 2.1: *Two parameter classification of the replication techniques in the database systems*

The first parameter: transaction location indicates where transactions can be executed. In general, a transaction containing a read only operation can be executed at any replica. An update transaction is a transaction that has at least one write operation. For this transaction there are two possibilities when it comes to where they can be executed; primary copy and update anywhere. These two parameters are generally referred to as update management strategies. In a primary copy approach, all update transactions are executed at a given site, unlike the update anywhere approach where the update transaction can be executed at any site, just like the read-only transaction.

Primary copy This is perhaps the most common replication method in use today, and it is also called master, master-slave, primary-backup or centralized. The data is owned by one site, this site is called master or primary. The remaining sites can only be updated by the master site, and they are considered as backups. The remaining sites are often called slaves or backups. With primary copy only, the master site is allowed to be updated and all other sites are just reflecting the change of the master. There are positive and negative aspects with primary copy. Conflicts can not occur between sites with primary copy because only the master site can perform the updates and there is no risk of updating the same data items twice. In other words, the master has full control when it comes to the updates at all time and will always contain up-to-date values. If there are too many who are trying to update the master site at once, the master can be overloaded and be a bottleneck. The reason is that there is no load balancer that can take some of the load of the master site and divide it to other sites[3, 9, 6].
According to M. Wiesmann et al:

> Primary copy has a single point of failure and a bottleneck, this limitation can be solved by making the protocol more complicated. Thus, if the master crashes, one of the slaves can take over the master role which requires an election protocol.

In other words, the system will have a failover control that make sure that the system avoids bottlenecks and system downtime. This will create a more robust system.

Update anywhere This is also called group, update everywhere, peer-to-peer, multi-primary or multi-master. All sites are allowed to be updated by others, unlike the primary copy. The updates which are made on the site or on several sites can be altered by any of the other sites. All sites which owns a copy are allowed to update the value of the copy.

The second parameter in Gray et. al.'s classification is synchronization strategy, it determines when replicas coordinate in order to achieve consistency. There are two possibilities for achieving constancy; eager and lazy replication. In eager replication, the updates of a transaction take place before the transaction commits. With lazy replication the update is asynchronously propagated after the transaction commits. Many replica protocols have been developed , but often it is not always clear whether they are eager or lazy.
Eager replication does often result in longer client response time since communication between master and slave takes place before a commit confirmation can be returned. It can provide strong consistency earlier than lazy replication. [5] [9]

Eager replication Is also known as synchronous replication. This mechanism updates the data immediately when the source data is updated. It is preferred for applications with low recovery time objectives (RTOs) that can not tolerate data loss and is also often used for disaster recovery purposes. Eager replication writes data to the primary and secondary sites at the same time so that the data remain current between sites. It is designed to work over short distances.[1]

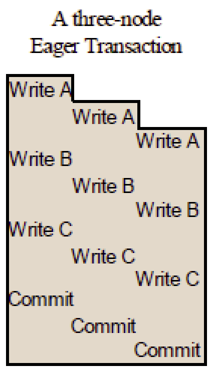


Figure 2.2: *Eager Transaction from Gray et al. [10]. Updates are applied to all replicas of an object as part of the original transaction.*

Lazy replication Also known as asynchronous replication. With this mechanism the target database is updated after the source database has been modified. It may take from a few seconds up to several hours or days to regain consistency in the database. Unlike eager replication, lazy replication is designed to work over long distances[31] because it does not require as much bandwidth and can tolerate some degradation in connectivity [32]. Lazy replication writes the data to a primary storage array and then, depending on the implementation approach, commits data to be replicated to memory or a disk-based journal. It then copies the data at scheduled intervals to replication targets, see figure 2.3.[2][3]

The parameters in the two parameter classification gives four possible strategies to implement replication solutions depending on whether it is eager or lazy, primary copy or update anywhere. The four strategies are; Eager Primary Copy, Eager Update Anywhere, Lazy Primary Copy and Lazy Update Anywhere, as shown in figure 2.1. Each strategy has advantages and disadvantages which are more or less obvious given the way they work. These strategies are described in detail in a book written by Kemme et. al., Replication - Theory and Practice [5].

[1]http://searchdisasterrecovery.techtarget.com/definition/synchronous-replication
[2]http://searchdatabackup.techtarget.com/definition/asynchronous-replication

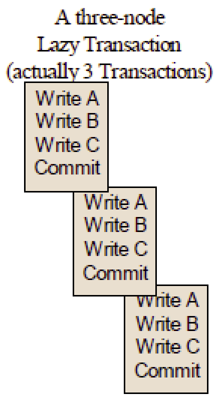


Figure 2.3: *Lazy Transaction from Gray et al. [10]. One replica is updated by the orginal transaction. Updates to other replicas propagate asynchronously, typically as a separate transaction for each node.*

Three parameter classification

Wiesmann has written the paper group communication and database replication: techniques, issues and performance[9], which are based on the paper by Wiesmann et. al. [11]. Wismann's paper argues that the two parameter classification are rather coarse and also have problems coping with non-votion replication techniques. The author has focused on the eager part of the three parameter classification, in his paper. The three parameters in the three parameter classification are; the server architecture, how changes or operations are propagated across servers and the transaction termination protocol. The two parameter classification and the three parameter classification have one thing in common, they both distinguish between eager and lazy replication. The first parameter, server architecture in the three parameter classification matches Gray et. al's. classification. The three parameter classification has extended the method with two additional parameters, Server interaction and Transaction termination.

Server architecture The server architecture matches Gray's classification, it distinguishes between primary copy and update anywhere as mentioned above. Which technique to chose is often based on which type of data is being replicated and the importance of data availability to the dependent application.

Server interaction Many sites may be located far away from each other and the network interaction between these sites are often very time-consuming. Server interaction is therefore a very important issue, this parameter is related to the degree of communication among database sites during the execution of a transaction. It determines the number of messages transmitted during execution of a single transaction. This number can be expressed by a function f(k), where k is the number of operations (insert, update, delete) in a given transaction. Two cases are considered in server interactions, constant interaction f(k) = O(1) and linear interaction f(k) = O(k). Both are presented with the big-O notation. The paper written by W. Khadzynov og M. Maksymiuk [18] has a detailed description of server interaction.

Constant Interaction In this case a constant number of messages is used to synchronise the sites for a given transaction, regardless the number of operations in the transaction. This gives f(k) = O(1).

Linear interaction Unlike the constant interaction case the number of messages used to synchronise the sites are variable. This gives f(k) = O(k).

Transaction termination The final parameter is the way a transaction terminates and this is how atomicity is guaranteed. To maintain data consistency and integrity, appropriate techniques must be used. Depending on the way of implementation there are two groups of algorithms: voting techniques and non-voting techniques.

Voting termination There are different degrees of voting, weak voting and strong voting. Weak voting means that the decision is unilateral, which means that only the master that can decide to abort a transaction. Strong voting implies that the decision to abort a transaction is multilateral, meaning that all the replicas have the possibility to abort a transaction, not only the master. In strong voting it is necessary to have a network communication between the sites to commit transaction. Such a network interaction is known as voting. During voting each site votes if the transaction should be committed or aborted and if at least one votes no, the transaction will be aborted. Some methods used for voting techniques are 2PC (Two-Phase Commit) this technique is also used in two parameter classification, 3PC (Three-phase Commit) and quorum based methods. The main drawback with strong voting is that the system always waits for the slowest replica. [9] [18]

Non-voting termination In this case each site can decide whether to commit or abort the transaction.

Compared to the two parameter classification that had four different replication solutions, the three parameter classification gives sixteen different possibilities to implement a replication solution. Each strategy has advantages and disadvantages which are more or less obvious given the way they work. The strategies that are for eager replication, i.e eight of them, are described in detail in the paper by Wiesmann [9, Chapter 3].

2.2 MySQL Replication

As described earlier, the focus of the research will be database replication within the telecom market. This section is going to look at how replication works in MySQL database and see which opportunities there are when using this database. It is a requirement from IPCO to use MySQL as the database engine.

In the past, improving replication involved implementing different replication topologies and increasing throughput with fast servers and networks. While these methods are still used today, third party software products have been developed to break the lazy replication model. [21, Chapter 3] Replication in MySQL enables data to be shared from one MySQL server (Master) to one or several MySQL servers (slaves). The Replication is lazy by default and has a dial-up service. The dial-up service means that the slave does not need to be permanently connected to the master. MySQL database replication can replicate all databases, selected databases or selected tables[12, Chapter 16].

MySQL has three built-in types of binary logging: (SBR) Statement based replication, (RBR) Row based replication and (MIXED) Mixed based replication. SBR logs the entire SQL statement to the binary log-file, RBR logs only the rows that are changed instead of the statement itself to the binary log and MIXED uses SBR logging by default, but automatically switches to RBR logging in particular cases. SBR has been on the market the longest, and it is also default in MySQL. MySQL replication does not normally need a lot of bandwidth since the database uses SBR and lazy replication as default. If RBR is used instead of SBR, it is dependent on a higher bandwidth.[15, Chapter 8][21, Chapter 2]

2.2.1 How replication works in MySQL

Replication in MySQL between servers are based on a binary logging mechanism. It means that the master adds all changes to its own binary log file and the slave/slaves main task is to keep itself updated by reading the masters binary log file. This process can be divided into three steps, but in reality each of these steps are much more complex, see figure 2.4.

1. The master writes all changes and updates to his binary log.
2. The slave is configured to read and copy the master's binary log file and applies the changes to its own relay log.
3. The slave adds the changes from the relay log into its own database.

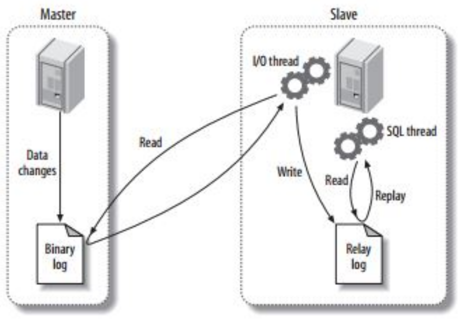


Figure 2.4: *How MySQL replication work from Schwartz et al. [15].*

All statements from the master are recorded in the binary log and it is the responsibility of the slave to decide which statement that is going to be executed. The master can not be configured to log only certain events, but the slave can be configured not to execute all events in the binary log, for example to only execute the events that apply in particular databases or tables. As mentioned earlier, this is called a partial replication. The slave can also execute all events in the binary log, this is called a full replication.

Replication of earlier versions of MySQL worked quite different. In many ways, for example, earlier versions of MySQL did use a relay log and it used two threads, instead of three as used today.

If one of the replication channels goes down MySQL does not have any automatic failover solution. To get a failover solution MySQL has to use a third part software.[3]

2.2.2 MySQL Replication Topologies

This section will look at some command topologies in MySQL and their limitations and strengths. Almost any combination of master and slave can be configured in MySQL, but the rules which have to be followed are:

- The slave instance can only have one master.
- Every slave must have a unique server ID.
- A master can have many slaves.

[3]http://www.edgewebhosting.net/blog/2011/12/mysql-high-availability-technologies/

- A slave can propagate changes from its master and be the master of other slaves.
- MySQL does not support Multimaster replication.

It is possible to have very complex topologies in MySQL but it is important to remember that simpler topologies can be very flexible. The variety of ways to use replication are so many that is easily could fill in a book of its own.
The two parameter classification and the three parameter classification are not fully supported by MySQL. MySQL supports both primary copy and update anywhere. It also supports eager replication in some degree, lazy - and semisynchronous replication.

Lazy Lazy replication is default in MySQL. With lazy replication, the master writes events to its binary log and slaves request them when they are ready. There is no guarantee that any event will ever reach any slave.

Eager The standard MySQL server distribution does not currently support eager replication. MySQL provides a different product called a MySQL Cluster that support eager replication. It is important to understand that MySQL Cluster is a different product than a MySQL server. When using eager replication, the master first commits a transaction, all slaves have to commit the transaction as well, before the master can return to the session that performed the transaction. The drawback of this is that there might be a lot of delay to complete a transaction. Today various products like Galera, Tungsten, Schooner Tech, etc. exists. These are a third party products that provides eager replication features. The third party software products have been developed to break the asynchronous replication model. It can provide seamless failover, replication between different database versions and building complex topologies such as multi-master replication.

Semisynchronous One major new feature in addition to the performance improvements in MySQL 5.5 is semisynchronous replication. This new replication method is an intermediate mechanism, compared to lazy replication and eager replication.
Semisynchronous replication falls between lazy and eager replication. The master waits after "commit" only until at least one slave has received and logged the events. It does not wait for all slaves to acknowledge receipt, and it requires only a receipt and not that the events have been fully executed and committed on the slave side. Semisynchronous replication is a step in the direction towards eager replication[21, Chapter 3].
With lazy replication there is no guarantee that the master event reaches, or has been committed to the slave host. On the opposite side of the replication spectrum is eager replication. All transactions are acknowledged and committed on all slaves before returning to the master that initiated the transaction. Semisynchronous, in this case, means that after a commit on the master host, a wait will occur until one of the configured slave hosts has received and logged the event. However, there is a drawback implied here, in the case where a system has more than one slave there is no confirmation that all the slaves have acknowledged the event since only one of the slave has to acknowledge it. Semisynchronous replication does not wait for all the slaves to commit the event and it does not require those events to be fully executed and committed on the slaves. Compared to lazy replication, semisynchronous replication provides improved data integrity. When commit returns successfully, it is known that the data exists in at least two places, on the master and at least on one slave. Semisynchronous replication works on MySQL 5.5 and newer versions and can be installed as a plugin. This type of replication can be used as an alternative to lazy replication. [13, Chapter 16.3.8], [21, Chapter 3]

MySQL does not support multimaster replication. Multimaster is used to describe a slave with more than one master. In some cases there can be confusion about what a multimaster replication is. Some uses the word "multimaster" to describe that there is more than one master in the entire replication topology, it is also called "update anywhere", since the changes can be written to any of the sites in this topology. This type of multimaster is legal in MySQL. The multimaster topology that is unsupported in MySQL is replication where one slave has two or more masters at once, as shown in figure 2.5.

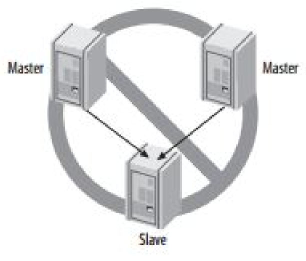


Figure 2.5: *MySQL does not support multimaster from Schwartz et al. [15].*

2.2.3 MySQL Cluster Replication

MySQL Cluster consists of a set of computers known as hosts, each running one or more processes. These processes are known as nodes. There are three types of nodes in a MySQL Cluster: management nodes, data nodes and SQL nodes. The only table engine running with MySQL Cluster is NDB, Network DataBase, also known as NDBCLUSTER [25][12].

Management Node The role of this node is to manage the other nodes within the cluster and performing such functions as providing configuration data, starting and stopping nodes and running backups. This is the first node to be started in the cluster.

SQL Node This node access the cluster data. In the case of MySQL Cluster, this node is a traditional MySQL server that uses NDB as the storage engine. In this particular situation, clients do not interact directly with data nodes but with SQL nodes which access the data stored in the data nodes. There can be several SQL nodes in a MySQL Cluster accessing the same clustered database. Each change made from a SQL server will be seen instantly by other SQL servers as data is synchronized on all the data nodes. It is also possible not to use SQL nodes in the cluster and access data in the data node directly from clients using the NDBAPI.

Data Node This nodes stores the cluster data and are the nodes that are hosting the database. As the database is distributed over the cluster, each data node keeps an active partition of the database.

Replication between clusters

It is important to understand regular replication, usually referred to simply as "replication", to easier to understand cluster replication. Cluster replication is conceptually very similar to replication, but can be more complex in practice. MySQL depends on the NDB storage engine to get cluster functionality. Some years ago replication between clusters was not available when using MySQL Cluster. It was not before MySQL 5.1.6 was introduced on the market that clusters could be replicated, but it was only the primary copy (master-slave) topology which was available. MySQL Cluster replication is available in MySQL version 5.1.6 and newer versions, but it is not yet available in the newest version of MySQL 5.7. As mentioned earlier, MySQL Cluster is different from MySQL server for the following reasons [21, Chapter 6]:

- There has not been any referential integrity until now. Foreign key is now available in the newest version 7.3.

- NDB is the only storage engine that can be used in MySQL Cluster and is limited to READ-COMMITTED transaction level.
- Access is faster than a regular MySQL server, more complex queries, including table joins and group by operations is more expensive.

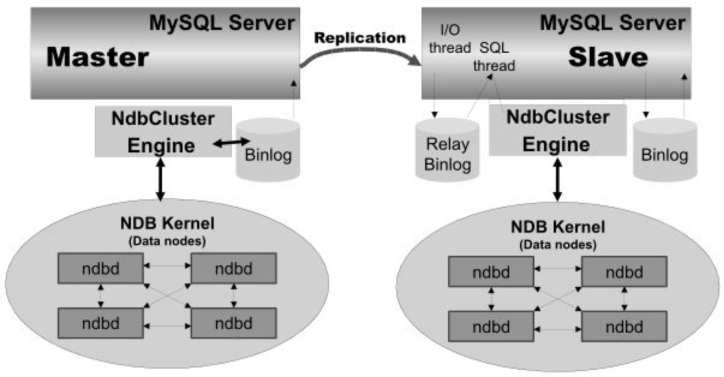


Figure 2.6: *from MySQL Reference Manual [13, Chapter 17].*

The scenario shown in figure 2.6, there is replication between two clusters. All updates in the master cluster are logged and saved in the slave cluster. This is accomplished by a special thread known as the NDB binlog injector thread. It runs on each MySQL server and produces a binary log. This thread ensures that all changes in the cluster are inserted into the binary log with the correct serialization order. It is almost identically to a regular MySQL replication but the main difference is the setup configurations in the database, and as mentioned the cluster replication is more complex.

MySQL Cluster replication uses lazy replication and does not support eager replication or semisynchronous replication between clusters. It supports eager replication between data nodes and SQL nodes within a cluster.

To guarantee high availability in the cluster, it is important to maintain a backup replication channel (secondary replication channel). MySQL is dependent on an external application to monitor the primary replication channel and to fail over to the secondary replication channel when necessary.

2.3 Network theory

In the real world networks can be divided into several classes: Social networks, Information networks, Technological networks, Biological networks[4]. In this thesis the technological network will be described.

A network is a collection of computers and other devices that can send and receive data from each other, in more or less real time. [22, Chapter 2] Each machine of a network is called a node.

When the first computers were build during the second world war, they where expensive and isolated. However, after about twenty years as the cost gradually decreased, the first experiments to connect computers together began. In the 1960's the first research papers describing the idea of building computer networks were published, one of these papers was written by Paul Baran [20]. In

[4]http://phys.cts.ntu.edu.tw/nuclear/2009/Lee2.pdf

the 1970's the telecommunication and computer industry became interested in computer networks. The telecommunication industry used X25[5]. X25 is a standard protocol suited for packet switched WAN communication, the service is still in use and available in niche and legacy applications. The computer industry took a completely different approach compared to the telecom industry by designing the LAN (Local Area Network). Ethernet and Token Ring among others are LAN technologies that were designed at that time. Some terminologies that are used in networking literature are [23, Chapter 2]:

- LAN (Local Area Networks) interconnect hosts that are up to a few kilometers apart.
- MAN (Metropolitan Area Network) interconnect devices that are up to a few hundred kilometers apart.
- WAN (Wide Area Network) interconnect hosts that can be located anywhere on Earth.

Whether it is wired or wireless, today most data communications happens by packages with information traveling over one or more networks. But before these networks can work together, they must use a common protocol or a common set of rules to transmit and receive these packets of data. Many protocols have been developed for this purpose. The most widely used protocols are TCP/IP (Transmission Control Protocol/Internet Protocol) and the OSI model (Open System Interconnection). These models are described in the article "What's The Difference Between The OSI Seven-Layer Network Model and TCP/IP"[29]. The OSI model consist of 7-layers while the TCP/IP model consist of 4-layers. Figure 2.7 compares the OSI model and the TCP/IP model.

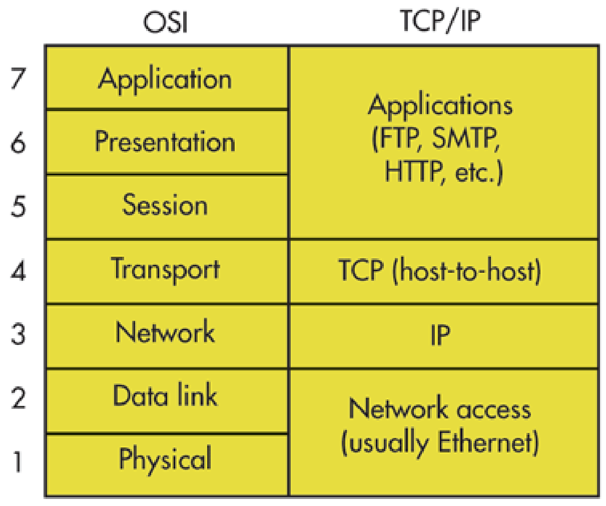


Figure 2.7: *Compare the 7-layers OSI model and the 4-layer TCP/IP model* [29].

Performance, also referred to as the speed of the network can be affected by various factors, two key elements of network performance are bandwidth and latency.

Latency In a network, latency is often used as a synonym for delay. Latency is describing how long it takes for a packet of data to get from one destination to another. [6] A so-called low latency network connection is one that generally experiences short delay times, while a high latency connection generally suffers from long delays.[7]

[5]http://en.wikipedia.org/wiki/X.25
[6]http://whatis.techtarget.com/definition/latency
[7]http://compnetworking.about.com/od/speedtests/a/network_latency.htm

Bandwidth In a network, bandwidth is often used as a synonym for data-transfer-rate that is supported by a network connection or interface. It means the amount of data which can be carried from one point to another, in a given time period. [8]

Numerous tools exist to measure the bandwidth of network connections and the latency. To measure the bandwidth, netperf and tcp can be used, there are also numerous online bandwidth and speed test programs available.[9] Network tools like ping tests and traceroute measures latency by determining the time it takes a given network packet to travel from source to destination and back, the so-called round-trip time[10]. Round-trip time is not the only way to specify latency, but it is the most common.

This chapter has described replication in general and replication which is used in MySQL. In addition, this chapter also described network history and network theory focusing on the technological network. The next chapter Method, will look at the research question, tests and how the data are analyzed.

[8]http://searchenterprisewan.techtarget.com/definition/bandwidth

[9]http://compnetworking.about.com/od/speedtests/a/network-speed.htm

[10]Round-trip time (RTT) is the time it takes for a client to send a request and for the server to send a response over the network, not including the time required for data transfer.

Chapter 3

Method

IPCO wants to find out which replication technique, eager replication or a combination of eager- and lazy replication, is most suitable for their production environment. Considering IPCO's requirements to the database system, two test environments are created. The environments are created to see how the different replication techniques behave and to test how each environment acts when latency and bandwidth are adjusted. The test environments are described in detail in chapter 4 Experimental setup. A combination of quantitative and qualitative methods are used to answer the research question by evaluating and comparing the two test environments replication behavior.

This chapter is structured as follows: a description of the main research question and the underlying research question. It is followed by a description of the tests which are going to be executed in this thesis and how the data collected during the tests, will be analyzed.

3.1 Research question

To see how the replication behave when the network is affected by high latency and low bandwidth. The main research question is:

- Given the two cluster topologies in what way are the characteristics different?

  1A: In what way does bandwidth and latency affect the performance on the two cluster topologies.

  1B: In what way does bandwidth and latency affect the replication delay on the Asynchronous Replication topology.

  1C: Which combinations of bandwidth and latency can cause error situations like package lose, cluster downtime or replication downtime.

Based on the theory and IPCO's requirements, it felt natural to select a cluster solution for test environments. Two cluster test environments stood out as particularly relevant in this case. The first environment is Big Cluster and the second is Asynchronous Replication. Each of these environments use different methods to transfer data between the sites. The Big Cluster uses eager replication while Asynchronous Replication uses both eager and lazy replications. Eager- and lazy replication were chosen based on the two parameter classification theory. This two replication methods are most commonly used and therefore most likely to work as expected.

Both test environments use eager replication within the cluster, they do not support eager replication between clusters. Big Cluster solution is the only choice when it comes to having an environment with eager replication. The Asynchronous Replication environment is chosen to get a lazy replication environment. Big Cluster and Asynchronous Replications are two very different environments. Therefore it is difficult to perform exactly the same measurements on the two environments. For the Big Cluster environment the performance and error rates are measured. For Asynchronous Replication environment performance, replication delay and error rates are measured. Below follows a description to easier understand what the research question means with performance, replication delay and error rate for each of the test environments and how they are measured.

3.1.1 Research question 1A

Performance In this test case, performance has two different meanings. For the Big Cluster test environment it means the time it takes from the procedure is started on one location and until the data is spread to all the other locations in the cluster. This requires not only eager replication over the same network but also eager replication over to another network segment, since the cluster is spread over two network segments. The replication between the segments can be affected be latency and lower bandwidth.
For the Asynchronous Replication test environment, performance means the time it takes from the procedure is started and the data is spread to the other sites within the cluster. This requires only eager replication over the same network segment.

The performance is measured for both environments to see how much variations in bandwidth and latency affects the performance.
To measure the performance a timer is added to the Java program which is running on the MySQL-runner server. The Java program is used to run the tests against the SQL nodes in the two environments. The timer which is used is java.sql.Timestamp. The timer counts the time on each AllocateIMSI procedure[1] that is executed in the database and the total time for all executions. The time is then written to a file.

3.1.2 Research question 1B

Replication Delay This means the time it takes to transfer data from one cluster to another cluster. In practice this means how far behind the slave is compared to the master. This is measured to see how latency and bandwidth affects the replication delay, and what happens to the replication when the delay is big. To view the replication delay, MySQL Cluster has a parameter named `Seconds_Behind_Master`. This parameter can be found under slave status. `Seconds_Behind_Master` shows the difference on the current time stamp on the slave and the original timestamp logged on the master. NTP (Network Time Protocol) is used on the master and slaves to get an identical clock time. This make the calculation of `Seconds_Behind_Master` more reliable. `Seconds_Behind_Master` field is a good approximate of how delayed the slave SQL thread is compared to the master. If the network is slow, the slave SQL thread may quite often be caught up with the slow-reading slave I/O thread. Therefore the `Seconds_Behind_Master` often shows a value of 0, even if the I/O thread is late compared to the master. Getting `Seconds_Behind_Master` readings of 0 alternating with readings of `Seconds_Behind_Master` of a high value, indicates that this is happening.

[1]The procedure AllocateIMSI allocates visitor IMSIs (International Mobile Subscriber Identity) in the database to IPCO subscribers which are traveling in foreign countries which have IPCO partners. This makes it possible to use the “roam as a local” service.

There are several alternatives how to monitor the replication delay between the two clusters which are in the test environment Asynchronous Replication. In this case a shell script from Nagios[2] written by C. Kuenzler is used to monitor the replication delay. To get it tailored for the Asynchronous Replication test environment some small changes were made in the script. This shell script is called check_mysql_slavestatus.sh. It checks if the Mysql replication is active, the delay and some other additional info and performance data. There is also a script with the name rep_mon.sh. This script calls the check_mysql_slavestatus.sh every second and writes the output of the check_mysql_slavestatus.sh file to a log file. Both scripts are placed on Mysql01 and Mysql03 because the test environment has an update anywhere approach. This means that the Mysql01 and Mysql03 are both slaves and masters for each other. Since each slave has one master, this must not be confused with the multimaster configuration that MySQL does not support.

3.1.3 Research question 1C

Error Rate This means the amount of data that is lost during the execution and if the environments are up and running at all time. Error rate is to measure that all data are spread correctly to the nodes in the test environments. In the Big Cluster environment the error rate is used to measure that data are spread to all the nodes that are in the cluster. In the Asynchronous Replication environment it is used to measure that all data is replicated correctly between the two clusters. To be able to measure the error rate on the test environments three scripts, named compare_himsidetails.sh, compare_imsievent.sh and compare_vimsiinfo.sh are created, one script for each of the tables H_IMSI_DETAILS, IMSI_EVENT and V_IMSI_INFO, which are located in the VRC_DB database. These three tables are checked for missing data, because they are affected during the execution of the procedure AllocateIMSI. The scripts are added on MySQL-runner and connects to the database VRC_DB through the clients Mysql01 and Mysql03. The content of the selected table in Mysql01 are written to textfile1, while the content of the selected table on Mysql03 are written to textfile2. The script then compares the two files to see if there are any differences.

The scripts and Java programs that measure performance, replication delay and error rate can be found in appendix E on the CD that comes with this thesis.

3.2 Tests

The ideal configuration for testing eager replication and a combination of eager- and lazy replication would have been to perform it with variation in latency and bandwidth. This can be achieved by using netem. Netem is a tool which can introduce latency and bandwidth limitations in the Linux TCP/IP communication stack. By using netem it is possible to simulate that the clusters are located in different countries or in different continents by adjusting latency and bandwidth. In this case two test cases are found to be appropriate.

Test case 1 (TC1): The AllocateIMSI procedure is going to allocate 1.002 IPCO subscribers concurrent in both clusters.

Test case 2 (TC2): The AllocateIMSI procedure is going to allocate 10.002 IPCO subscribers concurrent in both clusters.

The combination of parameters, latency and bandwidth that would have been interesting to test for both the test cases; allocate 1.002 IPCO subscribers and allocate 10.002 IPCO subscribers are

[2]http://exchange.nagios.org/directory/Plugins/Databases/MySQL/Check-MySQL-Replication-Slave-Status/details

described in the matrix, see figure 3.1. In this case the latency is given and measured in round-trip, not one-way. In total this would have led to 20 measurements per test case. In total 80 measurements for both Big Cluster and Asynchronous Replication test environments.

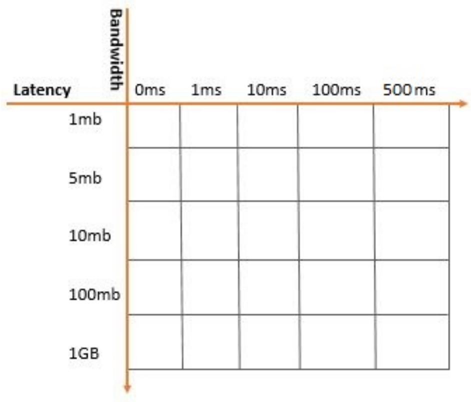


Figure 3.1: *Matrix shows the combination of latency and bandwidth which would have been ideal to test for TC1 and TC2.*

Unfortunately, these tests are difficult to perform because it appeared to be a challenge with the virtual network adapter offered by VMware ESXi 5.1 when it came to simulating limited bandwidth [3]. When bandwidth is adjusted on the WanSIM server to for example 10mb/s or 100mb/s, the bandwidth is measured between 100Kbit/s - 46Kbit/s. The rate control used by the communication stack in VMware is much closer to the hardware drivers compared with the delay/latency functions. This causes comparability issues which the VMware virtual network adapter drivers for netem's rate control. The tests for rate control, i.e. bandwidth, are therefore carried out in a slightly different manner.

To be able to get an answer on the research question it is not necessary to perform the tests with varying in latency and bandwidth. The important thing to test is varied latency. Latency will not be possible to avoid when transferring data between sites which are not located in the same building. Varied bandwidth is not so important because having too little bandwidth is not a case for IPCO. IPCO has simply stated that too little bandwidth is unacceptable if the bandwidth is too low for replication all data changes in the system will eventually fail.
The purpose of the tests is to see if there is enough bandwidth, if no data loss occurred during execution and if the database system is stable enough to handle and recover from delay due to too little bandwidth at peak time. Since it is possible to conduct the test without a variety in bandwidth, time has not been spent to solve this problem.
The tests which are performed had the same two test cases as the ideal test:

TC1: The AllocateIMSI procedure is going to allocate 1.002 IPCO subscribers concurrent in both clusters.

TC2: The AllocateIMSI procedure is going to allocate 10.002 IPCO subscriber concurrent in both clusters.

The two test cases are not tested with the combination of parameters: bandwidth and latency.

[3]https://bugs.launchpad.net/qemu/+bug/899140

TC1 and TC2 have two tests each, one with enough bandwidth and one with too little bandwidth. The tests with enough bandwidth has a bandwidth of 1Gb/s. The round-trip latency tested in combination with bandwidth 1Gb/s is 0ms, 1ms, 10ms, 100ms and 500ms, see figure 3.2.

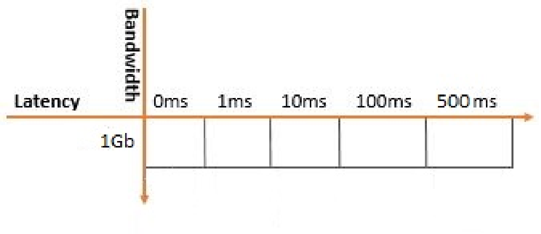


Figure 3.2: *Shows the combinations of latency which is tested on bandwidth 1Gb/s.*

The test with too little bandwidth is performed with a bandwidth between 100Kbit/s and 47Kbit/s for TC1 and TC2. This is because the network bandwidth is unstable since the traffic shaping is not working correctly on VMware. There is no purpose to add latency to the low bandwidth because the latency will automatically be larger with low bandwidth. Six measurements are going to be performed for TC1 and for TC2, in both test environments. In total 24 measurements measurements are going to be performed.

To be sure that the latency and bandwidth are affected by netem during the test, the network testing tools iperf and ping will be used. Iperf is used to measure the bandwidth and the ping command is used to measure the round-trip latency between the two network segments 10.20.1.128 and 10.20.1.160.

3.3 Analysis

Since all script for data collection during the tests are made especially for these specific tests, there is no need for any additional tool to help analyze the data. All data output are created with comma-separated lines to add it directly into Excel. Excel is used as a tool to make charts to help to analyze the data, to see what happened when the parameters AllocateIMSI, latency and bandwidth were adjusted during the tests. Some of the test data results are also typed manually into Excel. This is mainly done for results like total time for all AllocateIMSI to be executed and average time for one procedure to execute. Some of the data is analyzed manually by looking at the test results from the scripts, like error rate and max replication delay.

This chapter has described the main research question, the underlying research question and the two test cases that are going to be executed on two different test environments. In addition, this chapter has described how the data is collected during the tests are going to be analyzed. The next chapter will look in detail at IPCO's demands for a database environment and the two test environments.

Chapter 4

Experimental setup

Replication behavior over short and long distances with networks problems that can occur are tested on the two test environments. The networks problems are simulated by adjusting the bandwidth and latency. “Short” distances means replication in the same town, and “long” distances means replication to other continents, like for instance to USA.

This chapter is structured as follows: a description of IPCO’s requirements and database choice. These demands and choice of database forms the basis for the test environments that are described in detail under Lab setup. Following is a description of the databases, tables and test data that are used for the testing on both test environments, and finally the test preparations. The test preparation part also includes possible time thieves that can have an affect on the tests.

4.1 IPCO’s demands

IPCO has a list of functional and non-functional requirements that have to be fulfilled for the entire VSS system. These requirements must be met and followed all the way down in the system architecture to get the VSS system as solid and as good as possible. This means that the system database has to accommodate these requirements as well. When it comes to the database system, it is the non-functional requirements which are most important to fulfill in order to meet IPCO’s demands.

There are four non-functional requirements which IPCO regards as important for the database system. These four demands are performance, availability, scalability and cost. All these non-functional requirements are important in their own way, but at slightly different levels. Following is a description of the database choice and the requirements this database fulfills.

4.1.1 Choice of database

The IPCO company is a greenfield mobile telecom company which has specialized in offering roaming solutions to allow mobile telecom operators to roam their subscribers from mobile networks. The mobile networks roaming solution requires that data is available 99.999% of the time, and that data can be requested and modified in a real-time mode (The time a subscriber is willing to wait before a call is connected through to the recipient is limited).
Many different database solutions have been considered by IPCO, but based on their demands, the conclusion is that IPCO needs an in-memory, high availability cluster solution.
Other non-relational databases like Mongo DB were also considered but rejected due to problems with integration across technologies and platforms, and because of reported error rates above what IPCO considered acceptable. At the end, the decision stood between Soliddb and MySQL Cluster.

Soliddbs advantage is that some IPCO partners have more experience with this technology from previous solutions, but in the end MySQL Cluster was selected for the following reasons: [13]

- Proven high availability from business critical applications
- Large user base in the telecommunications industry. Popular choice as HLR (Home Location Register) database among Mobile operators (The HLR database is the most business critical database in the operation of a mobile telecom operator).
- Large number of API's
- Wide support among telecom equipment vendors - SQL support
- Reported performance among the best database systems available
- Availability of support from reliable support institution (Oracle)
- Partitioned tables to increase scalability [13, chapter 15]

MySQL meets the non-functional requirement which IPCO has to a database system, by fulfilling the requirement of availability, scalability, performance, and cost. Cost is an important requirement since IPCO is in a development phase. During the development it is important to keep the cost as low as possible. MySQL is an open source database and is therefore free to use during the development phase. To achieve IPCO's non functional requirements to the database system, some mechanism can be used. An important mechanism is replication. Replication will help with availability, scalability and performance.

4.1.2 MySQL Cluster 7.2 versus 7.3

When the decision to use MySQL Cluster as the primary database was made at the end of 2012, the MySQL Cluster version 7.2 was available. The first release of MySQL Cluster version 7.3 that was stable enough for use by the general public, became available on June 18th 2013, as MySQL Cluster 7.3.2[26]. At this time IPCO had already six months experience with MySQL Cluster 7.2. Based on the experience they knew that 7.2 was stable, and that 7.2 had the necessary performance to scale to the projected number of IPCO subscribers. Due to problems during installation of the test cluster using MySQL Cluster 7.3 and uncertainty about bugs and a short time frame for testing and verification, IPCO decided to use MySQL Cluster 7.2 for the production solution.

Since September 2013 a lot of changes have occurred that has caused IPCO to change from the MySQL Cluster version 7.2 to 7.3.3 for production [27].

- A new MySQL Cluster release (7.3.3) with a lot of bugfixes was released on November 18th 2013 [26].
- The IPCO production release was delayed until summer 2014.
- Tests done by MySQL showed that MySQL Cluster 7.3.3 was significantly faster and more stable than 7.2
- Extended support for standard SQL features like foreign keys and vastly improved table join.
- Initial tests done in this project showed that MySQL Cluster 7.3.3 performed better than 7.2.

Since the IPCO release had been delayed, IPCO changed their plans by the end of January 2014 and decided to use MySQL Cluster 7.3.3 in the production environment.

4.2 Lab setup

In this lab setup IPCO has proposed two different cluster topologies to be assessed. Both environments uses MySQL Cluster database. The first test environment is named Big Cluster and the second environment is named Asynchronous Replication. The environments are created based on requirements from IPCO, see 4.1 IPCO's demands. Below follows a description of the two test environments, Big Cluster and Asynchronous Replication and other components and configurations that are important for the test environments and the testing part, like VMware, WanSIM, MySQL-runner and the network configurations. A detailed description of the environments setup can be found in the appendices.

4.2.1 Test environments

Both test environments are running on a VMware ESXi server, called Kit, which is located in Halden. Ten virtual machines are created on Kit, these virtual machines are divided over three different network segments. In segment 10.20.1.128/5 the virtual machines mysql01, mysql02, ndb01 and ndb02 are placed. In the segment 10.20.1.160/5 the virtue machines mysql03, mysql04, ndb03 and ndb04 are placed. In the last network segment 10.20.1.0/8, the servers WanSIM and MySQL-runner are located. These two machines also have a foot in both of the other segments as well. See figure 4.1 for an overview of the network segments. The network configuration are described in appendix A.

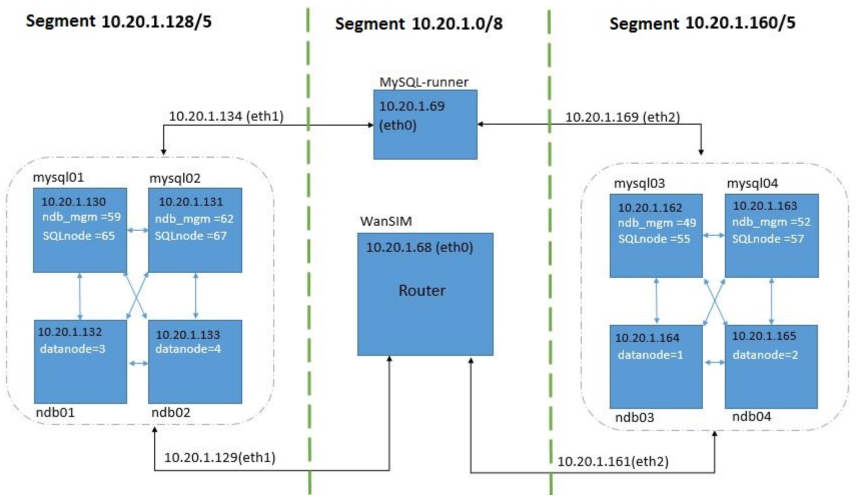


Figure 4.1: *Overview of the network segments, showing which machines that are located in the different segments.*

Both test environments are running on the same host. If the environments are going to be run on different hosts, twice as many hosts would be needed. It is possible to have two different environments running on the same host by changing the configuration files and create new instances of MySQL. MySQL Cluster 7.3.3 makes it natural to run several instances of MySQL on the same host. Because both environments are sharing IP-addresses only one of the environments is running at the time. This avoids possible conflicts between the environments.

WanSIM runs Debian as the operating system (OS), all other machines are running Oracle Linux 6. The default installation of Oracle Linux Server is a basic server installation. All the Oracle Linux 6 installations which are used in this lab setup, uses Software Development workstations as software, this is configured during the installation. All hosts that are running Oracle Linux are also running MySQL Cluster 7.3.3, except MySQL-runner.

In the beginning both environments were running on another VMware ESXi server, named Birk, this server is also located in Halden. Both test environments on Birk were running servers with Oracle Linux 5 and MySQL Cluster 7.2.9-1. All data nodes and SQL nodes were running Oracle Linux 5 because the newest version of Oracle Linux, Oracle Linux 6, does not support MySQL 7.2 only 7.3.
It turned out that iSCSI and all Ethernet access were running on the same network, this led to problems with writing and reading to/from disk. The disk used on Birk was not running in a raid and it was shared with other systems. When other systems are saving and reading from the disc, network congestion can occur. The problem was not solved within reasonable time and to avoid the problem, the environments were switched to another platform called Kit. The test environments had to be set up from the beginning, and in cooperation with IPCO it was decided to use MySQL Cluster 7.3 instead of 7.2 as described earlier in this chapter.

WanSIM This server plays an important role in both environments. WanSIM is set up to act as a router, to route the network traffic mainly between two different network segments. The two segments are 10.20.1.128/5 and 10.20.1.160/5. A third segment was created that lead out on the world wide web.
To get the server to act as a router the IP forwarding has to be enabled, see appendix B. It is disabled as default on any modern Linux distributions. WanSIM uses the network emulation, netem for traffic shaping of the different network segments. Netem is integrated in the OS Debian and Oracle Linux. Netem makes it possible to emulate wide area networks delay, by adjusting the latency and bandwidth parameters. Netem is used to create networks problems between the two segments 10.20.1.128/5 and 10.20.1.160/5, by adjusting the latency and bandwidth.

MySQL-runner This server runs the OS Oracle Linux 6, 64-bit and is created to run the test against the two test environments, Big Cluster and Asynchronous Replication. MySQL-runner is standing in the network segment 10.20.1.0/8, and has a foot in each of the two other network segment 10.20.1.128/5 and 10.20.1.160/5 as well, like WanSIM. This gets a fast connection directly to the selected network segment without going through the WanSIM router. To run the tests, eclipse standard [1] and a mysql JDBC driver [2] are installed.

Test environment 1 - Asynchronous Replication This test environment consists of two clusters, C1 and C2. C1 consists of the four servers; mysql01, mysql02, ndb01 and ndb02 which are running Oracle Linux 6, 64-bit. C2 also consists of four servers, mysql03, mysql04, ndb03 and ndb04 and which also are running Oracle Linux 6. Figure 4.2 shows the Asynchronous Replication test environment. This test environment is a fully replicated system with update anywhere approach. It is lazy replication between the two clusters and eager replication within the clusters. In this environment there are two clusters that are located on two different network segments and the SQL nodes in each cluster are switching between being master and slave. Which of the SQL nodes that are master or slave in each cluster depends on which node the load balancer is running against. There are two load balancers in this environment. They are placed between the host MySQL-runner and the test environment. It helps to distribute the traffic coming towards the clusters so that the load reaches the node in each cluster that has free capacity. Eager replication is used between the nodes that are located in the same cluster, lazy replication is used to transfer data between the two clusters.

The following section contains a description of how the clusters and replication are configured.

[1] https://www.eclipse.org/downloads/
[2] http://dev.mysql.com/downloads/connector/j/

A more detailed description of each part, is referred to in the appendix.

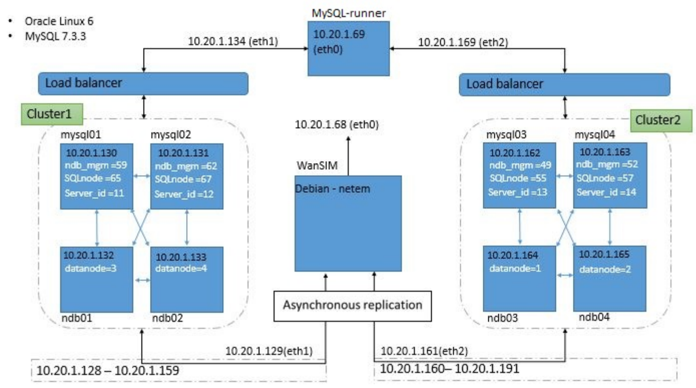


Figure 4.2: *Asynchronous Replication test environment - The environment uses eager replication within the clusters and lazy replication between C1 and C2*

Configuration of MySQL Cluster C1 is running a total of four hosts, two data nodes (ndb01 and ndb02) and two SQL nodes (mysql01, mysql02). The management nodes are running on the same hosts as the SQL nodes. C2 also contains of four hosts, two data nodes (ndb03, nndb04) and two SQL nodes (mysql03, mysql04). Also here the management nodes are running on the same host as the SQL nodes, just like C1. There are only minor changes between C1 and C2 in the cluster configuration. One of the differences is that the clusters are placed in different network segments, this means that the IP-addresses in each cluster are different. This is because they belong to different network segments and can only reach each other through the router that is between the two segments. Other changes are the ndb_mgm id, SQL node id and data node id, have different values. Since there only are minor changes between the two clusters, only the configuration of C1 is described in detail in appendix C. The differences between C1 and C2 will also be mentioned in the description.

A new feature in MySQL Cluster 7.3 is the auto-installer. This is a web-based wizard that helps creating the configuration files, deploy and start the cluster. The configuration files and the startup command files that are created by the auto-installer are my.cnf, config.ini, start_ndbdmgm.sh and ndb_start.sh. The auto-installer is running on ndb01.

In this case, the wizard is only used as a help tool, to get the configuration files right and to set up the cluster to be a real time, high performance cluster. The configuration files and startup files from the wizard are placed out manually on the different hosts.

In MySQL Cluster 7.3.3 each SQL node requires a my.cnf file and a config.ini file. The start_ndbdmgm.sh file and ndb-start.sh are startup commands to make it easier to start the management server.
Below follows a description of the different files which are generated by the auto-installer and where to place them on the different hosts.

The two SQL nodes/ management nodes (mysql01 and mysql02) contain the configuration files my.cnf and config.ini, and the startup command file start_ndbdmgm.sh. The my.cnf file provides two pieces of information: a connection string that tells the node where to find the management node and a line telling the MySQL server on this host to enable the NDBCLUSTER storage engine. The configuration file my.cnf is located in the /etc directory on both nodes.

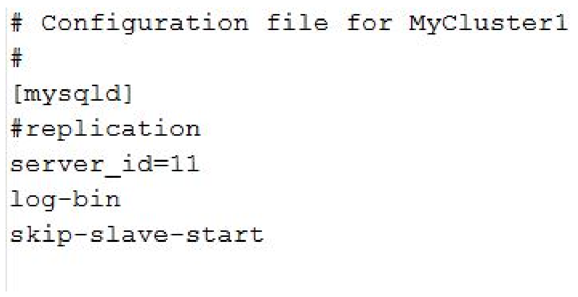
The config.ini configuration file is the configuration file for the management node. It tells how many replicas to maintain, how much memory to allocate for data and indexes on each data node, where to find the data nodes, where to save data and where to find any SQL nodes. This file is added to the directory /var/lib/mysql-cluster/ on both the two SQL nodes. The config.ini files are identical so it does not matter which files that are on which SQL nodes.
The last file start_ndbmgm.sh is placed on /root/MySQL7.3/ on both nodes. By running this file on both nodes, the management nodes will start.

The two data nodes (ndb01 and ndb02) contain the startup command file ndb-start.sh. The folder /var/lib/mysql-cluster is created on both data nodes with mysql privileges. The script ndb-start.sh is added on /root/MySQL7.3

When all the configuration files and startup command files are created and placed out on the correct nodes, the replication channel between the two clusters were added.

Setting up replication between the two cluster sites. In this test environment there are four replication channels. The two first replication channels are called primary replication channels and replicate data from mysql01 to mysql03 and from mysql03 to mysly01. The last two channels are called secondary replication channels and replicate data between mysql02 and mysql04 and between mysql04 and mysql02. The secondary replication channel is just a backup channel and will only be applied when the primary replication channel is down.

To set up the replication channels between the two cluster sites it is necessary to prepare the my.cnf files or the command line on all the SQL nodes. In this case the configurations are done in the my.cnf file. Figure 4.3 shows an excerpt from mysql01s my.cnf of the replication configuration. All changes done in the my.cnf file are permanent changes, while the changes written in the command line will be deleted each time MySQL is down or is restarted. To prepare the nodes for replication each SQL node in either cluster must be uniquely identified by a server id, the binary logging needs to be enabled on each cluster site that will act as a replication master and enable the skip slave start. The skip slave start is not necessary but will prevent the replication to start immediately. The replication will not start before the START SLAVE statement has been issued.



```
# Configuration file for MyCluster1
#
[mysqld]
#replication
server_id=11
log-bin
skip-slave-start
```

Figure 4.3: *Shows an excerpt from the replication configuration in mysql01s configuration file my.cnf*

The next step is to create a slave account on the masters in each cluster. In this case the mysql01 server is acting as the master in C1, and mysql03 in C2 for the primary replication channel, since this is an update anywhere approach. It is important that the slave accounts have the appropriate privileges for replication.
When the slave accounts are created, the slaves needs to be configured to use the master. For the primary replication channel when mysql01 is the slave, it will use mysql03 as the master. If

mysql03 is the slave, it will use mysql01 as the master. This needs to be configured before starting the slaves.

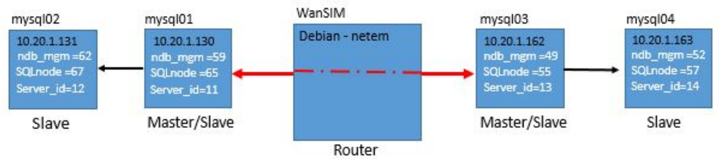


Figure 4.4: *Illustration of the replication on the SQL nodes.*

Starting the cluster When the cluster is going to be started it is important that each of the node hosts are started separately. First the management nodes on mysql01 and mysql02 are started, followed by the data nodes on ndb01 and ndb02, and then finally the SQL nodes on mysql01 and mysql02. C2 is started in the same way.
To start the management nodes, the following script start_ndbmgm1.sh is executed on both hosts. The first time the management nodes are started the ndb_mgmd must be told where to find the configuration file, config.ini. The scripts start_ndbmgm1.sh contains this information.
To start the ndbd process on the data nodes, the following script start-ndb.sh is executed on both hosts. To see if ndb_mgm is running on both the SQL nodes and the data nodes, it is possible to go into the management client and see the status.
Finally the MySQL is started on both SQL nodes.

Test environment 2 - Big Cluster: This test environment only consist of one Big Cluster, unlike test environment 1 that consists of two smaller clusters. MySQL does not support eager replication between sites, MySQL has provided another product called MySQL Cluster that supports eager replication. MySQL Cluster is the closest way to get a test environment with update anywhere, eager replication. The test environment topology is one big cluster that is fully replicated and spread over two different network segments. Figure 4.5 shows the Big Cluster test environment. It is an active-active, also called hot-hot solution, because if one of the SQL nodes or data nodes are down, there is always a SQL and a data node that can take over. The Big Cluster test environment has a total of 4 SQL nodes and 4 data nodes and consists of the servers mysql01, mysql02, mysql03, mysql04, ndb01, ndb02, ndb03 and ndb04. All servers are running Oracle Linux 6 with 64-bit, just like test environment 1. The replication between the data nodes are eager.

The Big Cluster test environment uses the same hosts as the Asynchronous Replication test environment. To get the environments to run on the same hosts the configuration files config.ini and my.cnf have to be changed. By starting the mysqld as mentioned in test environment 1 there can be several instances of MySQL running with different configuration files.

In this test environment there is one big cluster that is spread over two different networks. The network segment 10.20.1.128/5 has two SQL nodes (mysql01 and mysql02) and two data nodes (ndb01 and ndb02) and the network segment 10.20.1.160/5 also has two SQL nodes (mysql03 and mysql04) and two data nodes (ndb03 and ndb04). The nodes on segment 10.20.1.128/5 are called $C1_0$ and the nodes on the segment 10.20.1.160/5 are called $C1_1$, to make it easier to understand which part of the cluster it is referred to.

The environment setup in this environment is similar to the cluster setup in test environment 1- Asynchronous Replication. The difference is that this is a cluster which is spread over two network segments and it has 4 SQL nodes and 4 data nodes as mentioned over. The Big Cluster environment are described in details in appendix D.

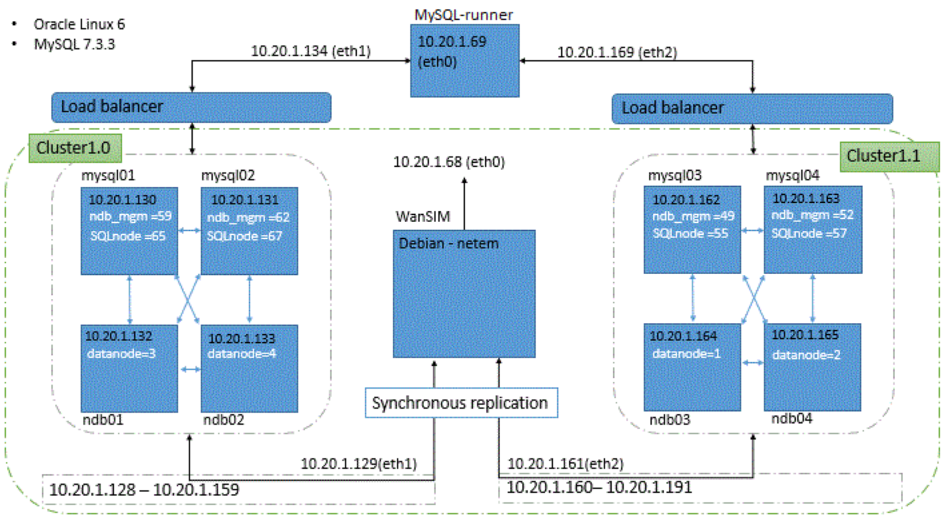


Figure 4.5: *Big Cluster test environment.*

Cluster configuration The Big Cluster is running a total of eight hosts, four data nodes (ndb01, ndb02, ndb03 and ndb04) and four SQL nodes (mysql01, mysql02, mysql03, mysql04). The management nodes are running on the same hosts as the SQL nodes.

The auto-installer that is running on ndb01 is used to generate configuration files and startup command files for the Big Cluster. The wizard is only used as a help tool to generate the files and to set the cluster to be a real time, high performance cluster. The configuration files and startup command files from the wizard are placed out manually on the different hosts. The configuration files my.cnf and config.ini, and the startup command files start_ndbdmgm2.sh are generated for each of the four SQL nodes. The startup command files ndb-start1.sh are generated for each of the four data nodes.
There are no major changes between the configuration files on the SQL nodes on test environment1 and test environment2. For my.cnf, the connection string that tells the node where to find the management nodes has two more management nodes added. The config.ini file has now a total of four management nodes, data nodes, SQL nodes and API nodes.

My.cnf is placed under /etc, config.ini is placed under /var/lib/mysql-clusterSync and start_ndbdmgm2.sh is placed under /root/MySQL7.3 on each of the SQL nodes. The ndb-start1.sh command startup file is placed under /root/MySQL7.3 on the data nodes.

Starting the cluster When the cluster is going to be started it is important that each of the node hosts are started separately, just like the Asynchronous Replication test environment. First the management nodes on mysql01, mysql02, mysql03 and mysql04 are started, followed by the data nodes on ndb01, ndb02, ndb03 and ndb04, and then finally the SQL nodes on mysql01, mysql02, mysql03 and mysql04.
To start the management nodes, the following script start_ndbmgm1.sh is executed on both hosts. The first time the management nodes are started the ndb_mgmd must be told where to find the configuration file, config.ini. The scripts start_ndbmgm1.sh contains this information.
To start the ndbd process on the data nodes, the following script start-ndb.sh is executed on both hosts. To see if ndb_mgm is running on both the SQL nodes and the data nodes it is possible to

go into the management client and see the status.

4.2.2 Database model, tables and test data

To get the test environments up and running properly, the VRC_DB database and the tables had to be added to both the environments. A schema called vrc_Create_NDB.sql is created to add all the tables into the VRC_DB database in both test environments, see figure 4.6 to see the tables in the VRC_DB database. Since MySQL Cluster 7.2 was the original database chosen, the tables are created without foreign keys because MySQL Cluster 7.2 do not support relation databases. In addition IPCO has stated that they will not use foreign keys unless it is strictly necessary, since each foreign key requires a index. Unnecessary indexes slows down inserts, updates and delete operations. Sql files with test data are generated from perl scripts. Fifteen MVNOs with 35.000 H-IMSIs and 35.000 V_IMSIs each are generated. This makes a total of 525.000 H_IMSIS and 525.000 V_IMSIS that are inserted into to the VRC_DB database. Both environments contains the same tables and test data. The script, procedures and SQL files for generation and adding the test data to the database can be found in appendix E on the CD that comes with this thesis.

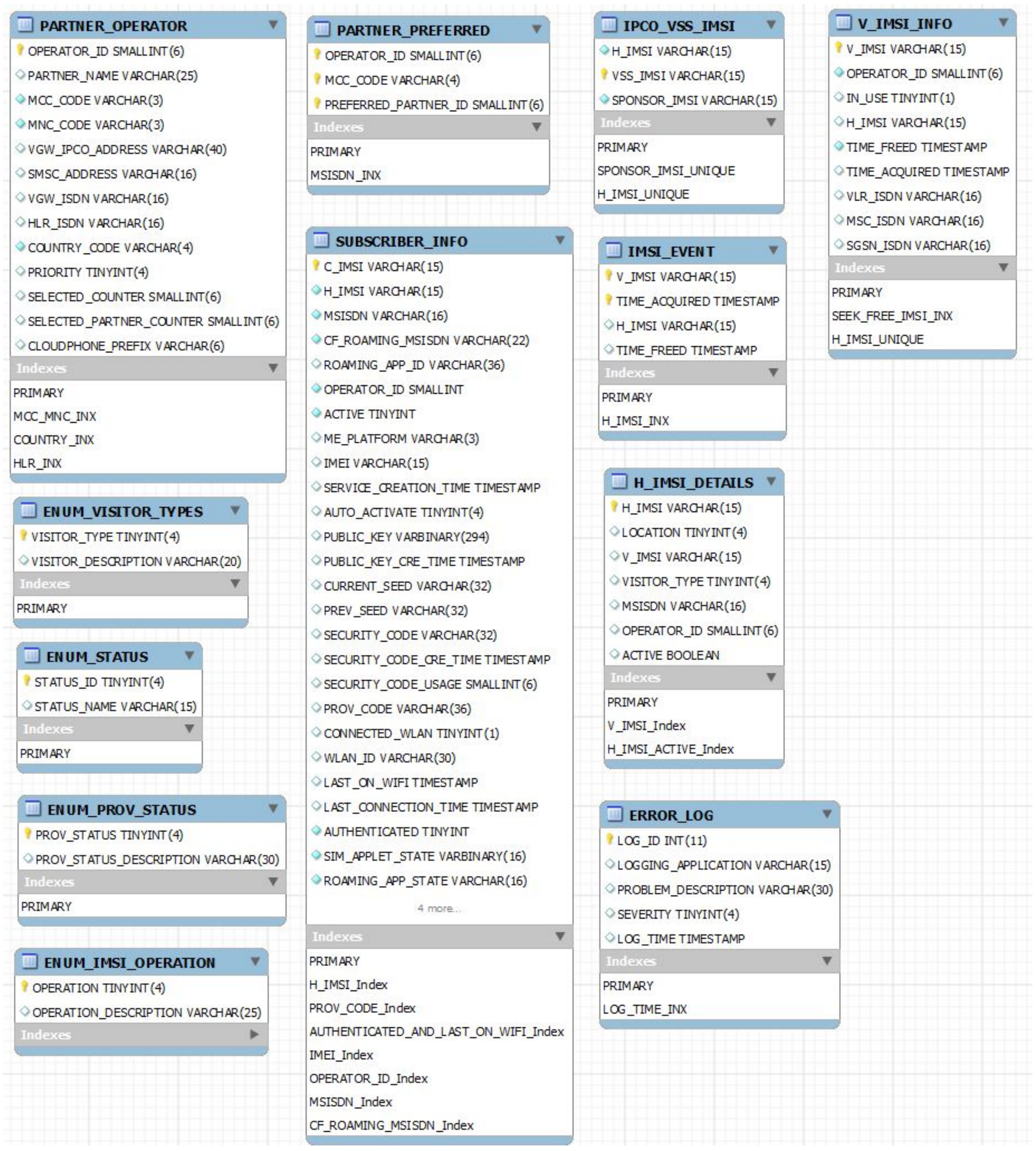
Procedures The procedures AllocateIMSI and ReleaseIMSI are added to both test environments. AllocateIMSI is a procedure that allocates visitor IMSIs to IPCO subscribers who are traveling in a foreign country which has IPCO partners. This makes it possible for the subscribers to use the "roam as a local" service IPCO provides. ReleaseIMSI is used by the AllocateIMSI procedure if one subscriber already has a visitor IMSI and tries to allocate a new visitor IMSI. The releaseIMSI procedure deletes the old visitor IMSI, if there is any.
There are three ways to commit a procedure:

1. Auto commit. This is default, but if the procedure goes wrong it will not roll back the transactions. This can lead to inconsistencies in the data.
2. Start transaction and commit. This is written in the beginning and in the end of the procedure. This will automatically turn off the auto commit. If the procedure goes wrong it will role back the transaction. In this way the data will not be added to the database unless everything goes ok.
3. Turn of auto commit manually and use start transaction and commit outside the procedure. In this way one can add batches of data in the database. If one transaction in the batch goes wrong it will role back the entire batch. On batch can be 10, 100, 1.000 etc.

Both the AllocateIMSI and the ReleaseIMSI procedure uses alternative 2, start transaction and commit to commit the procedure. It is important that all the data is correctly transferred to the database, if an error occurs the transaction will roll back and avoid losing data. The procedure is created to allocate the users one by one. This is the reason why alternative 2 is chosen as the preferred way to commit the procedures. If alternative 3, turn of auto commit manually and use start transaction and commit outside the procedure had been selected, the allocation could risk to get rolled back on the entire branch even if only one allocation failed. This can result in longer time for the allocations than necessary.

4.3 Test Preparation

It is important to perform some research before the tests are conducted to get an overview of the test and which factors that can affect the tests. For both Big Cluster and Asynchronous Replication test environments the performance can be affected by time thieves. A list of possible time thieves is created, these are parameters which theoretically could affect the performance of the AllocateIMSI procedure.



PARTNER_OPERATOR
OPERATOR_ID SMALLINT(6)
PARTNER_NAME VARCHAR(25)
MCC_CODE VARCHAR(3)
MNC_CODE VARCHAR(3)
VGW_IPCO_ADDRESS VARCHAR(40)
SMSC_ADDRESS VARCHAR(16)
VGW_ISDN VARCHAR(16)
HLR_ISDN VARCHAR(16)
COUNTRY_CODE VARCHAR(4)
PRIORITY TINYINT(4)
SELECTED_COUNTER SMALLINT(6)
SELECTED_PARTNER_COUNTER SMALLINT(6)
CLOUDPHONE_PREFIX VARCHAR(6)
Indexes
PRIMARY
MCC_MNC_INX
COUNTRY_INX
HLR_INX

ENUM_VISITOR_TYPES
VISITOR_TYPE TINYINT(4)
VISITOR_DESCRIPTION VARCHAR(20)
Indexes
PRIMARY

ENUM_STATUS
STATUS_ID TINYINT(4)
STATUS_NAME VARCHAR(15)
Indexes
PRIMARY

ENUM_PROV_STATUS
PROV_STATUS TINYINT(4)
PROV_STATUS_DESCRIPTION VARCHAR(30)
Indexes
PRIMARY

ENUM_IMSI_OPERATION
OPERATION TINYINT(4)
OPERATION_DESCRIPTION VARCHAR(25)
Indexes

PARTNER_PREFERRED
OPERATOR_ID SMALLINT(6)
MCC_CODE VARCHAR(4)
PREFERRED_PARTNER_ID SMALLINT(6)
Indexes
PRIMARY
MSISDN_INX

SUBSCRIBER_INFO
C_IMSI VARCHAR(15)
H_IMSI VARCHAR(15)
MSISDN VARCHAR(16)
CF_ROAMING_MSISDN VARCHAR(22)
ROAMING_APP_ID VARCHAR(36)
OPERATOR_ID SMALLINT
ACTIVE TINYINT
ME_PLATFORM VARCHAR(3)
IMEI VARCHAR(15)
SERVICE_CREATION_TIME TIMESTAMP
AUTO_ACTIVATE TINYINT(4)
PUBLIC_KEY VARBINARY(294)
PUBLIC_KEY_CRE_TIME TIMESTAMP
CURRENT_SEED VARCHAR(32)
PREV_SEED VARCHAR(32)
SECURITY_CODE VARCHAR(32)
SECURITY_CODE_CRE_TIME TIMESTAMP
SECURITY_CODE_USAGE SMALLINT(6)
PROV_CODE VARCHAR(36)
CONNECTED_WLAN TINYINT(1)
WLAN_ID VARCHAR(30)
LAST_ON_WIFI TIMESTAMP
LAST_CONNECTION_TIME TIMESTAMP
AUTHENTICATED TINYINT
SIM_APPLET_STATE VARBINARY(16)
ROAMING_APP_STATE VARCHAR(16)
4 more...
Indexes
PRIMARY
H_IMSI_Index
PROV_CODE_Index
AUTHENTICATED_AND_LAST_ON_WIFI_Index
IMEI_Index
OPERATOR_ID_Index
MSISDN_Index
CF_ROAMING_MSISDN_Index

IPCO_VSS_IMSI
H_IMSI VARCHAR(15)
VSS_IMSI VARCHAR(15)
SPONSOR_IMSI VARCHAR(15)
Indexes
PRIMARY
SPONSOR_IMSI_UNIQUE
H_IMSI_UNIQUE

IMSI_EVENT
V_IMSI VARCHAR(15)
TIME_ACQUIRED TIMESTAMP
H_IMSI VARCHAR(15)
TIME_FREED TIMESTAMP
Indexes
PRIMARY
H_IMSI_INX

H_IMSI_DETAILS
H_IMSI VARCHAR(15)
LOCATION TINYINT(4)
V_IMSI VARCHAR(15)
VISITOR_TYPE TINYINT(4)
MSISDN VARCHAR(16)
OPERATOR_ID SMALLINT(6)
ACTIVE BOOLEAN
Indexes
PRIMARY
V_IMSI_Index
H_IMSI_ACTIVE_Index

ERROR_LOG
LOG_ID INT(11)
LOGGING_APPLICATION VARCHAR(15)
PROBLEM_DESCRIPTION VARCHAR(30)
SEVERITY TINYINT(4)
LOG_TIME TIMESTAMP
Indexes
PRIMARY
LOG_TIME_INX

V_IMSI_INFO
V_IMSI VARCHAR(15)
OPERATOR_ID SMALLINT(6)
IN_USE TINYINT(1)
H_IMSI VARCHAR(15)
TIME_FREED TIMESTAMP
TIME_ACQUIRED TIMESTAMP
VLR_ISDN VARCHAR(16)
MSC_ISDN VARCHAR(16)
SGSN_ISDN VARCHAR(16)
Indexes
PRIMARY
SEEK_FREE_IMSI_INX
H_IMSI_UNIQUE

Figure 4.6: Shows the database tables in the VRC_DB database.

Procedures and Index configuration The procedure itself, combined with indexes can affect the performance. To illustrate this two of the queries in the procedure AllocateIMSI were running without indexes. As a result two table scans were found in the procedure. The time it took to run the AllocateIMSI procedure once was reduced from 0.200ms to 0.016ms when all the queries were properly indexed.

The performance can also be affected by the type of indexes that are used. Binary indexes are more flexible than hash indexes, but slower when it comes to lookups in the database. A hash index is very fast for lookups but can only search one specific row at the time. A binary index is more flexible in the way that it can find ranges of data. Binary indexes will be slower by time, but the hash index time will always be constant. In the Big-O notation for binary index = O(lg n) and Hash index =O(n)[3].

MySQL query optimizer can also be a possible time thief by using the wrong index on the query. MySQL query optimizer analyzes the selected row to see if any optimization can be used to process the query more quickly. It is possible to overwrite the optimizer. This can be useful in some cases. The optimizer has a set of guidelines to follow to take advantage of the indexes. [24, Chapter 5]

Database design A poor logical database design can impact the performance of the entire system. Database design is a big topic and includes table design, row size and page density, correct and consistent data type usage, column nullability and data integrity. R. Bradford has written a guideline for good database design in MySQL database, the name of the guideline is "The Top 20 Design Tips - For MySQL Enterprise Data Architects"[4].

JDBC To amplify the hypothesis that JDBC (Java DataBase Connectivity) can be a possible time thief, 1.002 AllocateIMSI procedures call in one sql-file has been created. The sql-file was executed directly in the MySQL client, called mysql01. The time it took to execute all the procedures in the client was about 16 seconds. This is approximately 0.016 seconds per procedure. When the same amount of procedures were executed on the MySQL-runner server through JDBC, it took 23 seconds. This is about 0.023 seconds per procedure. In this case JDBC is a substantial time thief.

VMware It is important to check for error situations which are outside of the MySQL Cluster. For instance, in the preliminary rounds there were problems when adding large amount of data in the cluster. It turned out to be caused by an error on the birk-server which was running VMware. iSCSI (internet Small Computer System Interface) [5] is an IP-based, Internet Protocol which is used as an interface for disk access on Birk. iSCSI and all Ethernet access like the eager replication in each cluster, the lazy replication between the clusters and all other internet access were running on the same network. This led to problems with writing and reading to/from disk. It used up to 2-3 seconds to access the disk. The solution was to move iSCSI to its own dedicated network.

The disk size when using VMware is another important aspect to be aware of. The disk used on Birk was not running in a Raid, the disk was shared with other systems and the access was through an Ethernet switch. When other systems are saving and reading from the disc, network congestion can occur. To avoid this problem, the test environments were moved to another VMware, named Kit.

Network Performance can for instance be affected by network latency and bandwidth. It is impossible to avoid latency when data is transported over long distances. Bandwidth defines the

[3]http://www.cs.cornell.edu/courses/cs312/2008sp/lectures/lec20.html
[4]http://www.slideshare.net/ronaldbradford/top-20-design-tips-for-mysql-data-architects-presentation
[5]http://searchstorage.techtarget.com/definition/iSCSI

maximum capacity of a network. Low bandwidth in the network can cause bottlenecks and unacceptable delays, this can results in slow communication speeds and limits user efficiency and productivity on a network[6][30]. The research question is going to see how the two test environments handles low bandwidth and high latency.

This chapter has looked at the demands IPCO has for the VSS system and how these demands has helped to influence the layout on both test environments. Further, the database, tables and test data that are used for the test environments Asynchronous Replication and Big Cluster have been looked at, as well as the test preparation. The main focus of this chapter was the lab setup of the Asynchronous Replication test environment and the Big Cluster test environment.
The next chapter looks at how the tests were conducted and the results that came from the executed tests.

[6]http://www.techopedia.com/definition/24819/network-bottleneck

Chapter 5

Result

As described in chapter 4 Experimental setup, two test environments have been established. The first environment is Big Cluster which is a cluster with four data nodes and four SQL nodes with eager replication between the nodes. The second environment, Asynchronous Replication consists of two separate clusters with lazy replication between, and eager replication within the clusters. Each cluster consists of two data nodes and two SQL nodes. Both test environments are located in Norway. To make the tests seem more like a real situation, the tests are performed by simulating that each cluster in the environment is placed in individual countries. A typical example here would be to simulate that half of the environment is placed in Norway and the other half is placed in France, as shown in figure 5.1. To make is easier to understand, the clusters in the Asynchronous Replication environment are named C1 and C2, while half the cluster in Big Cluster is named $C1_0$ and the other half is named $C1_1$. C1 is placed in Norway and C2 is placed in France. The same applies for the Big Cluster environment where half the cluster ($C1_0$) is located in Norway and the other half ($C1_1$) is located in France. Latency and bandwidth are adjusted to simulate distance between the clusters that are placed in Norway and France.

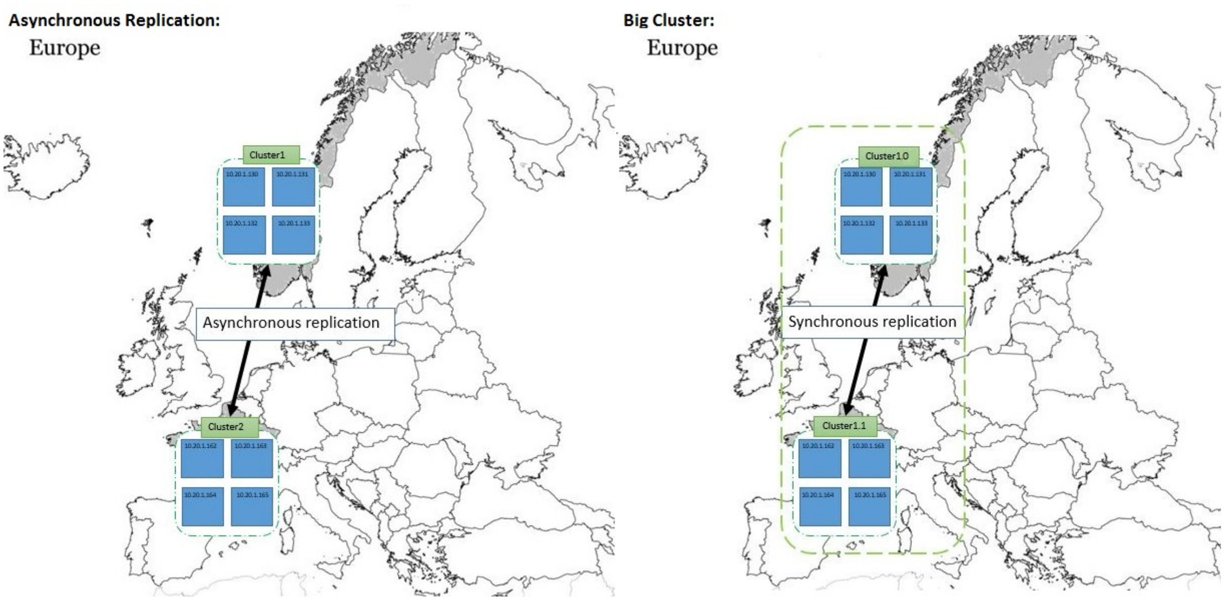


Figure 5.1: *C1 and $C1_0$ are located in Norway while C2 and $C1_1$ are located in France.*

C1 and $C1_0$ which are placed in Norway serves three Norwegian operators, ICE, Ventelo and Lyca.no. The subscribers of these operators travel in this case to either France, Switzerland or Sweden. C2 and $C1_1$ which are placed in France serves three French operators, Vivendi, Iliad and Lyca.fr. The subscribers of these operators travel in this case to either Norway, Germany or Spain. All the operators in Norway, France and the countries they are traveling to are Mobile Virtual Network Operators (MVNO), see section 1.4 for description.

This chapter is structured as following; a description of how the tests are conducted and how the test result data is collected during the tests. Followed by the test results which are presented as statistics for both test environments, Asynchronous Replication and Big Cluster.

5.1 Test execution

A set of tests on the two test environments will be conducted.
The Big Cluster test environment will run these tests:

- TC1: Allocate 1.002 IPCO subscribers on $C1_0$ which is located in Norway and 1.002 IPCO subscribers on $C1_1$ which is located in France concurrent, with the procedure AllocateIMSI. Each subscriber is represented by a home IMSI (International Mobile Subscriber Identity) which is used by the AllocateIMSI procedure. This TC is going to test the performance and error rate in the test environment.
    - Bandwidth 1Gb/s: This is the first test in TC1, this test is executed with a bandwidth of 1Gb/s and with varying combinations of round-trip latency. The combinations shown in table 5.1 are tested.

Latency in ms	**Latency=0**	**Latency=1**	**Latency=10**	**Latency=100**	**Latency=500**
Bandwidth=1Gb	X	X	X	X	X

Table 5.1: *Shows the combinations of bandwidth = 1GB and round-trip latency that is tested for the Big Cluster environment.*

-
    - Bandwidth 57Kbit/s: This is the second test in TC1, this test is executed with a bandwidth of 57KBit/s. Latency will not be adjusted in this case.
- TC2: Allocate 10.002 IPCO subscribers on $C1_0$ which is located in Norway and 10.002 IPCO subscribers on $C1_1$ which is located concurrent in France with the procedure AllocateIMSI. Each subscriber is represented by a home IMSI which is used by the AllocateIMSI procedure. This TC is going to test the performance and error rate in the test environment.
    - Bandwidth 1Gb/s: This test is executed with a bandwidth of 1Gb/s and with varying combinations of round-trip latency. TC2 will test the same combinations of bandwidth and round-trip latency as TC1, the combinations are shown in table 5.1.
    - Bandwidth 57Kbit/s: This test is executed with a bandwidth of 57KBit/s. Latency will not be adjusted in this case.

The Asynchronous Replication test environment will run the same tests as the Big Cluster test environment, the replication delay will also be measured:

- TC1: Allocate 1.002 IPCO subscribers on C1 which is located in Norway and 1.002 IPCO subscribers on C2 which is located concurrent in France with the procedure AllocateIMSI. Each subscriber is represented by a home IMSI which is used by the AllocateIMSI procedure. This TC is going to test the performance, error rate and replication delay for this test environment.

  - Bandwidth 1Gb/s: This test is executed with a bandwidth of 1Gb/s and with varying combinations of round-trip latency. The combinations shown in table 5.2 are tested, these are the same combinations which were tested on the Big Cluster Test environment.

Latency in ms	**Latency=0**	**Latency=1**	**Latency=10**	**Latency=100**	**Latency=500**
Bandwidth=1Gb	X	X	X	X	X

Table 5.2: *Shows the combinations of bandwidth = 1GB and round-trip latency that is tested for the Asynchronous Replication environment.*

  - 57Kbit/s: This test is executed with a bandwidth of 57KBit/s. Latency will not be adjusted in this case.

- TC2: Allocate 10.002 IPCO subscribers on C1 which is located in Norway and 10.002 IPCO subscribers on C2 which is located concurrent in France with the procedure AllocateIMSI. Each subscriber is represented by a home IMSI which is used by the AllocateIMSI procedure. This TC is going to test the performance, error rate and replication delay for this test environment.
  - Bandwidth 1Gb/s: This test is executed with a bandwidth of 1Gb/s and with varying combinations of round-trip latency. TC2 will test the same combinations of bandwidth and round-trip latency as TC1, the combinations are shown in table 5.2.
  - 57Kbit/s: This test is executed with a bandwidth of 57KBit/s. Latency will not be adjusted in this case.

The tests as described above for both test environments are simulating Norwegian mobile subscribers and French mobile subscribers who are traveling abroad. The Norwegian subscribers are traveling to either France, Switzerland or Sweden, the subscribers are equally divided between these three countries. The French subscribers are traveling to either Norway, Germany or Spain, also here the subscribers are equally divided between these three countries. The system does all registrations for the Norwegian subscribers in the database in Norway. All data which is located in the Norwegian database needs to be replicated to the database in France. As for the French subscribers, the system does all the registrations in the database in France. All data in this database needs to be replicated to the Norwegian database. It is the replication between these two countries which is tested.

To test the replication between Norway and France and to look at the variations in the two test environments, the tests are simulating that all the subscribers are arriving the country they are visiting at the same time and turn on their cellphones immediately after arrival. This will lead to a high number of subscribers which the system needs to register at the same time. The registration is against the database which is located in Norway and in France. One example can be that several planes arrives at an airport in the same country or in different countries at the same time. Each plane can contain 100-500, people depending on the size of the plane[1] [2]. As soon as the plane has landed and the seatbelt light has been turned off, most of the passengers will immediately turn on their cellphones. If many of the passengers on the planes are IPCO subscribers and they come from the same country, this will cause a high load on the system.
To get the VSS service IPCO offers to run, the system has to register all their cellphones in the database back home. This service makes it possible to "roam as a local" in the visiting country, see section 1.5.
When the system does an allocation against the database back home for each subscriber, there

[1]http://no.wikipedia.org/wiki/Boeing_747,
[2]http://no.wikipedia.org/wiki/Boeing_737

will be a heavy load on the clusters in the Asynchronous Replication test environment and on $C1_0$ and $C1_1$ in the Big Cluster environment. The tests will show how these environments handle the replication between the countries during heavy load. This is tested by measuring performance, error rate and replication delay. Section 3.1 describes the details about the test parameters, performance, error rate and replication delay.

The tests are executed with 1.002- and 10.002 subscribers against each cluster in Norway and France. Originally, it was also supposed to be executed with 100.000 subscribers against each cluster in the environment. After analysing the test results from the test which was performed with 10.002 subscribers, the test with 100.000 subscribers was dropped due to the time it would take to run these tests. To use the example with the planes again; It would be likely to get around 1.000 subscribers to turn on there cellphones at the same time, but the possibility that 100.000 IPCO, subscribers from the same country would turn on there cellphones at the same time is quite unlikely. Unless for example 100.000 Norwegian IPCO subscribers were in France and the network they were connected to went down, then all subscribers had to roam to another IPCO partner. This situation is a special case and will not be taken into account in this thesis.

The first test is going to allocate 1.002 IPCO subscribers on each cluster that is located in the two test environments. The second test is going to allocate 10.002 subscribers on each cluster in both test environments. It was decided to run the tests with 1.002 and 10.002 subscribers against each cluster. 1.002 and 10.002 can be divided with three, since both Norway and Franc uses three different countries each for the testing. The tests are set up to have the same amount of subscribers in each of the three countries, this is the same for both Norwegian subscribers and French subscribers.

Below is a description of how the tests are conducted. To simulate that the clusters are located in different countries, the same continent or within different continents, the tests are performed with variation in latency and bandwidth. Norway and France are used as test countries for all tests which are performed. This is to simplify the tests as much as possible. It did not have any impact on the test results. Both test environments are update anywhere.

Two Java programs are used for testing, MysqlClusterTest and MysqlCluster2Test[3]. They both run on the host MySQL-runner. These two Java programs are very similar, the only difference is that they connect to different clusters and allocate subscribers from two different countries. The red line in figure 5.2 and figure 5.3 shows that MysqlClusterTest connects to the database that is located in Norway, while MysqlCluster2Test connects to the database that is located in France. This might as well be done with multiple threads in the same project. To start the test the two Java programs in the environments had to be started simultaneously. MasterClusterTest and MasterCluster2Test first created ranges of home IMSIs for the Norwegian and French operators. The home IMSI ranges that are created and the MCC code (mobile country code)[4] of the country the subscriber had traveled to, are used as input parameters to the AllocateIMSI procedure[5]. The procedure AllocateIMSI are executed concurrently in C1 and C2 in the Asynchronous Replication environment, and concurrently in $C1_0$ and $C1_1$ on the Big Cluster environment.

The combinations of round-trip latency as shown in Table 5.1 corresponds to transactions in the same building (L=0), in the same town (L=1), in the same country (L=10), in the same continent (L=100) and in different continents(L=500). The numbers are based on Verizon,[6] statistical data.

[3]The Java programs MysqlClusterTest and MysqlCluster2Test can be found in appendix E on the CD that comes with this thesis.

[4]http://en.wikipedia.org/wiki/Mobile_country_code

[5]The procedure AllocateIMSI allocates visitor IMSIs (International Mobile Subscriber Identity) in the database to IPCO subscribers which are traveling in foreign countries which have IPCO partners. This makes it possible to use the "roam as a local" service.

[6]http://www.verizonenterprise.com/about/network/latency/

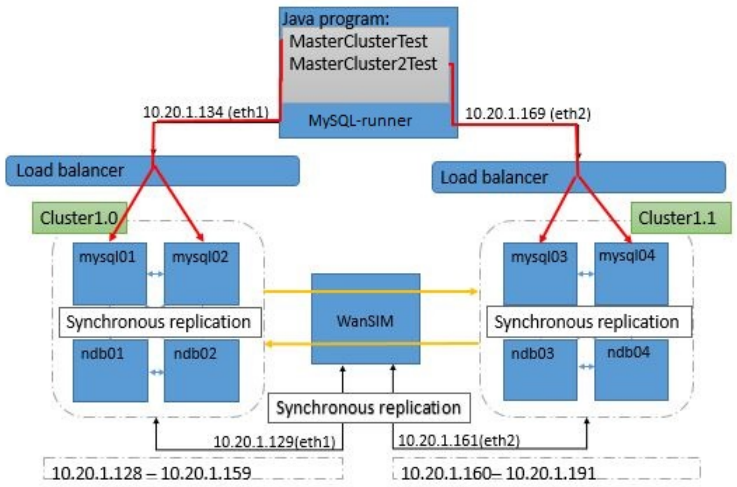


Figure 5.2: *The red lines in the Big Cluster shows which Java program that is connected to which half of the cluster. The yellow lines shows the eager/synchronous replication between* $C1_0$ *and* $C1_1$*.*

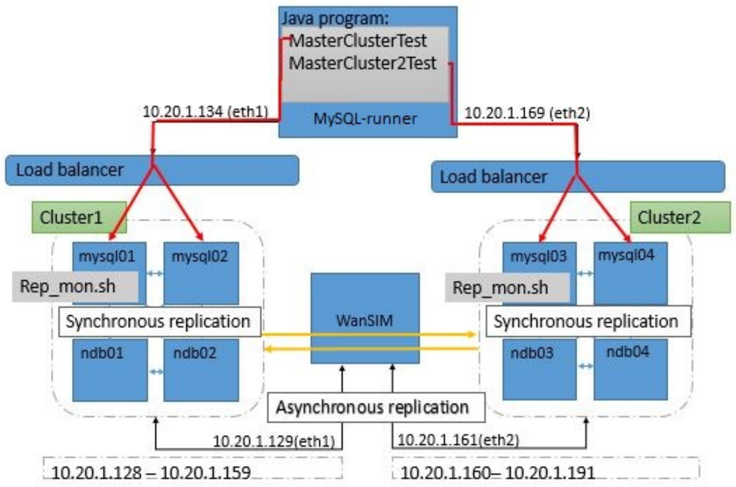


Figure 5.3: *The red lines in the Asynchronous Replication environment shows which Java program that is connected to the clusters. The yellow lines shows the lazy/asynchronous replication between C1 and C2.*

The latency is adjusted in the WanSIM router, by using netem. The same tool is used for the bandwidth. The yellow line in figure 5.2 and figure 5.3 shows the eager replication between $C1_0$ and $C1_1$ and the lazy replication between C1 and C2. These lines also show that adjustments of latency or bandwidth in WanSIM affects the network between the $C1_0$ and $C1_1$ which are on two different networks, as well as the network between C1 and C2.

5.2 Data collection

Below is a description of different scrips which collect the data on both test environments during the execution of the tests. The scripts are located on different hosts, MySQL-runner, mysql01 and mysql03. All the scripts can be found in appendix E on the CD that comes with this thesis.

MySQL-runner The Java program MysqlClusterTest and MysqlCluster2Test are both designed to be the main programs for the testing. These two Java programs do also contribute to the collection of data during the tests. They measure the performance of the environment by measuring the total time for all transactions, average time per transaction and logs the total time of every tenth allocations.
The scripts compare_vimsiinfo.sh, compare_himsidetails.sh and compare_imsievent.sh compare tables from mysql01 and mysql03. This is to see if there are any differences in the tables due to data loss during the tests.

Mysql01 and mysql03 These nodes do only collect data when the Asynchronous Replication test environment is tested. This is the only time it is necessary to measure the replication delay. The file that is used is a shell script from Nagio with the name check_mysql_slavestatus.sh and a script called rep_mon.sh.

5.3 Test results

The test results from the Big cluster and Asynchronous Replication are presented as statistics. The results will be discussed and described in chapter 6 Discussion. The Performance and error rate are tested on the Big Cluster while performance, error rate and replication delay are tested on the Asynchronous Replication environment. The first test results which are presented are the results from the Big Cluster, followed by the results from the Asynchronous Replication environment.

5.3.1 Big Cluster

The aim of these tests are to verify how $C1_0$, $C1_1$ and the replication between them managed to cope with the system allocating IMSIs for several subscribers at the same time, over a bad network connection that is affected by latency or low bandwidth. To see how this has influenced the performance and the possibility to lose data over eager replication, the performance and error rate are tested for this test environment.

TC1: The AllocateIMSI procedure allocated 1.002 IPCO subscribers concurrent against both $C1_0$ and $C1_1$.
The first tests which were conducted in TC1 had a bandwidth of 1Gb/s, followed by a bandwidth of 57Kbit/s.

	Cluster 1.0 ($C1_0$)					**Cluster 1.1 ($C1_1$)**				
Bandwidth: 1Gb/s	Latency:0	L:1	L:10	L:100	L:500	Latency:0	L:1	L:10	L:100	L:500
Subscribers	**1.002**	**1.002**	**1.002**	**1.002**	**1.002**	**1.002**	**1.002**	**1.002**	**1.002**	**1.002**
Total time for all transactions	25,35s	44,52s	215,60s	1943,50s	9510,96s	25,97s	43,35s	219,40s	1900,20s	9279,30s
Average time per transaction	0,0253s	0,0453s	0,215s	1,94s	9,49s	0,0256s	0,0432s	0,219s	1,89s	9,26s
Error rate	0	0	0	0	0	0	0	0	0	0

Table 5.3: *This table shows the test results from TC1, where 1.002 subscribers were added to $C1_0$ and $C1_1$ concurrently.*

1Gb/s bandwidth Five combinations of round-trip latency and bandwidth are tested for all subscribers in TC1.

Table 5.3 shows the test results that comes from the combination of bandwidth = 1Gb and varying round-trip latency for $C1_0$ and $C1_1$. The table shows amongst other the amount of subscribers, this means the number of times the test was executed, the total time for all transactions to be executed, the average time per transaction and if the test had any package loss or down time during the execution of the tests. 0 means that there was no package loss or down time during the tests.

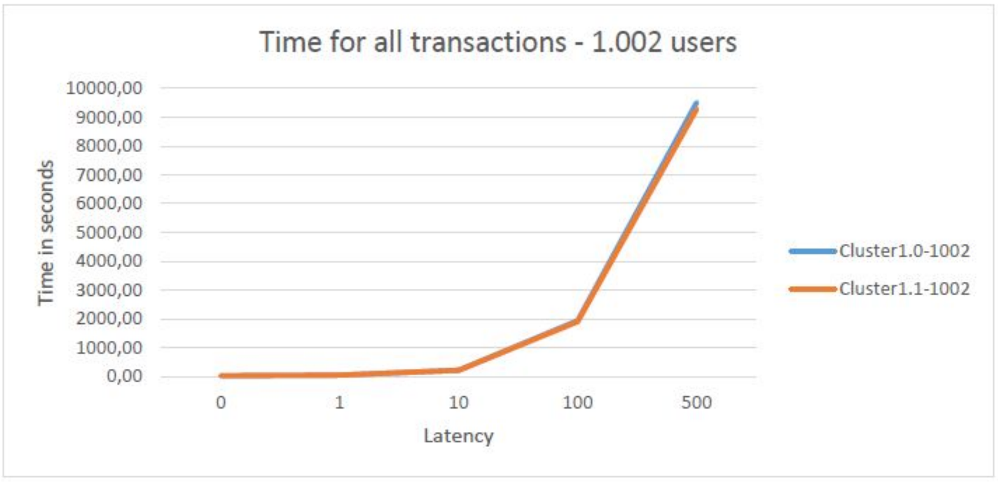


Figure 5.4: *Graph shows the total time for all transactions when the bandwidth is 1Gb/s. X-axis shows the latency from 0-500, while the Y-axis shows time in seconds.*

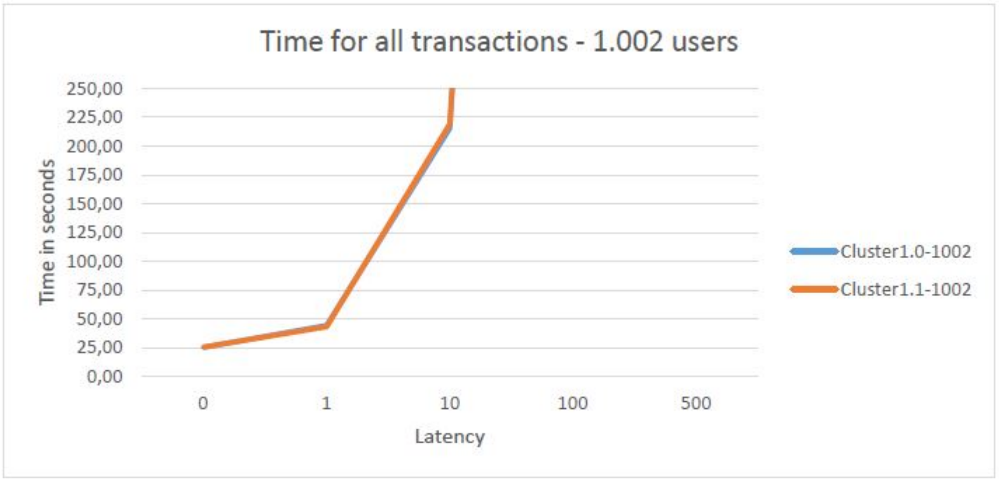


Figure 5.5: *This graph is the same graph as shown in figure 5.4, but is enlarged for a close-up view at the values when latency=0, 1 and 10.*

The graph shown in figure 5.4, shows the total time in seconds for all transactions for $C1_0$ and $C1_1$. The total time for all transactions when latency=0 is 22,35 in this graph and not 0 as it seems to be. It is difficult to see the lines for both clusters because the values are very similar and therefore they overlap each other. The values used in this graph can be seen in table 5.3, total time for all transactions.

Shown in figure 5.4 and figure 5.5 the graphs have a breakpoint on the scale when latency is 1, 10 and 100. Possible reasons for this will be discussed in the discussion part.

Performance are measured for $C1_0$ and $C1_1$, by logging the total time of every tenth allocations for all the variations of round-trip latency.

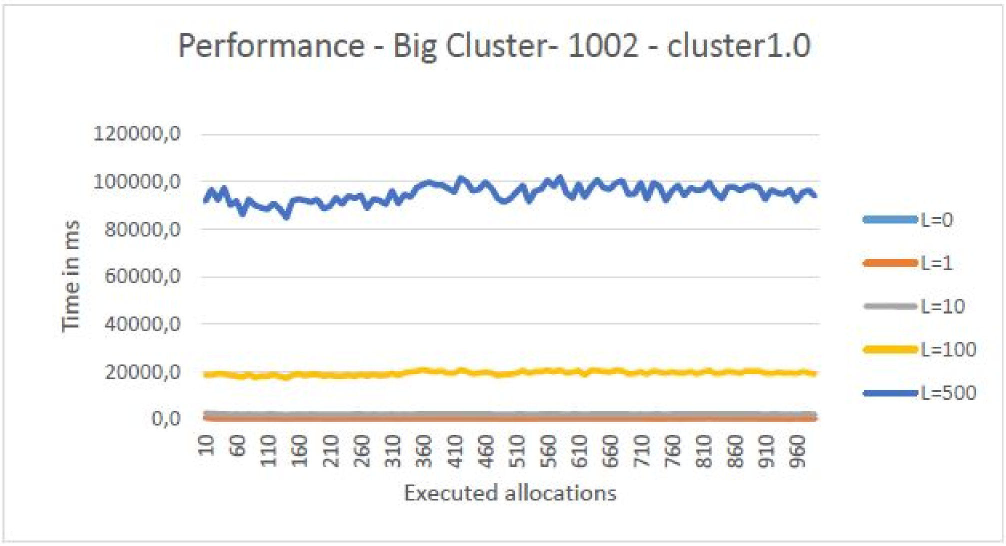


Figure 5.6: *The graph shows the performance of every tenth allocation for $C1_0$ when the bandwidth is 1Gb/s. The X-axis shows the number of allocations, the Y-axis shows time in milliseconds.*

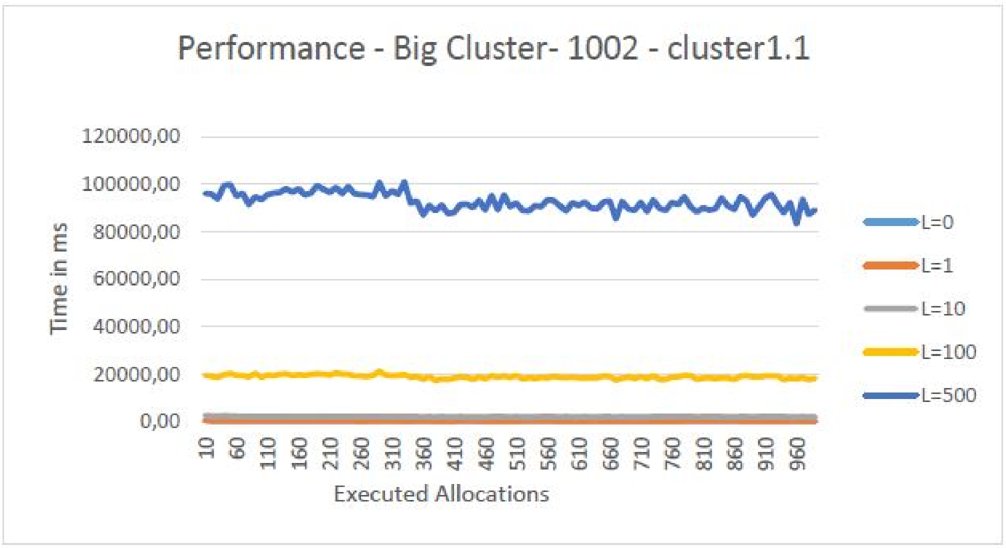


Figure 5.7: *The graph shows the performance of every tenth allocation for $C1_1$ when the bandwidth is 1Gb/s. The X-axis shows the number of allocations, the Y-axis shows time in milliseconds.*

The graphs from figure 5.6 and figure 5.7 show the performance that was measured when latency were 0, 1, 10, 100 and 500 from $C1_0$ and $C1_1$.
The measurements from $C1_0$ and $C1_1$ looks close to each other when the Y-axis is from 0ms - 120.000ms. To show that the measurements varies more when the latency increases, both graph in figure 5.6 and figure 5.7 are enlarged on latency =0 and latency =500.

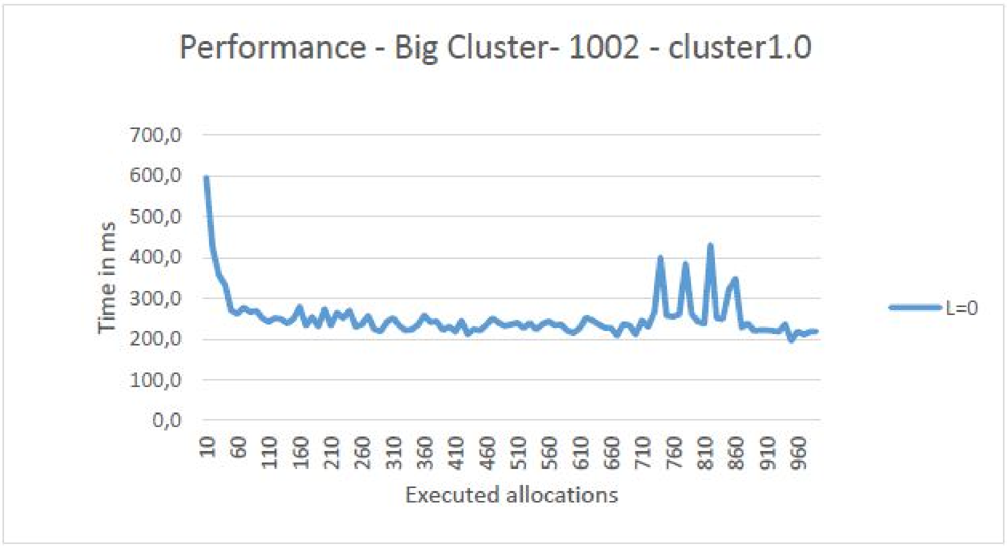


Figure 5.8: *This graph is the same graph as shown in figure 5.6, performance from $C1_0$. It is enlarged to get a close look at the time for executing ten allocations when latency=0.*

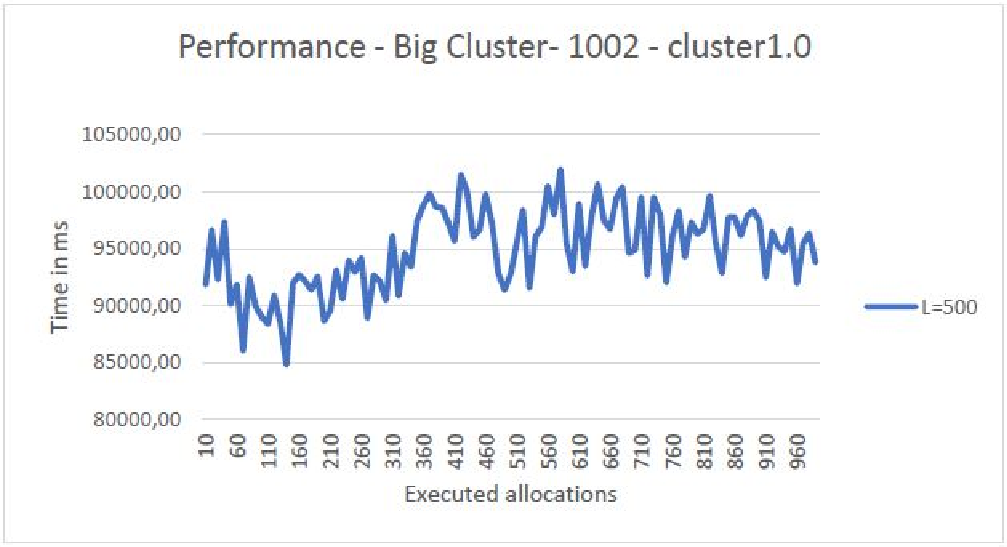


Figure 5.9: *This graph is the same graph as shown in figure 5.6, performance from $C1_0$. It is enlarged to get a close look at the time for executing ten allocations when latency=500.*

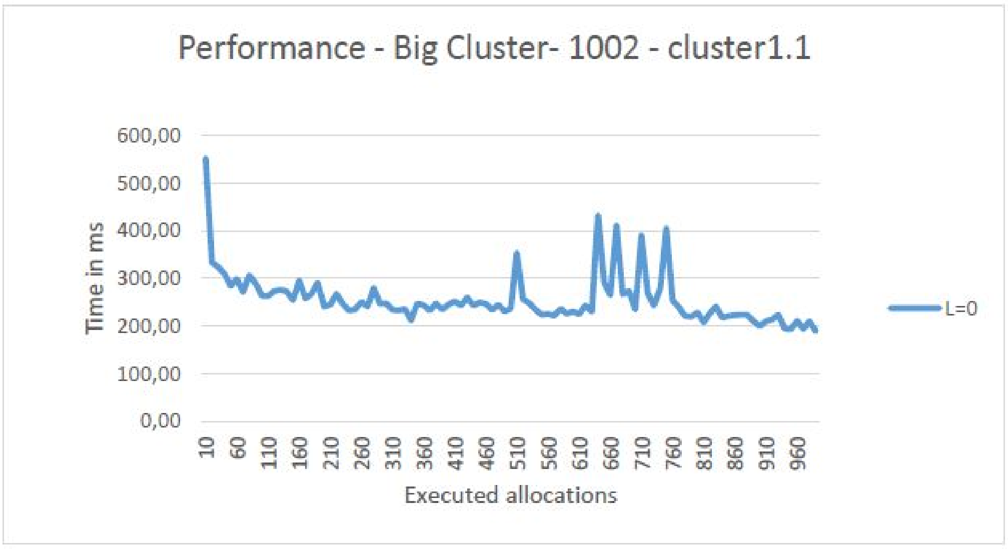


Figure 5.10: *This graph is the same graph as shown in figure 5.7, performance from $C1_1$. It is enlarged to get a close look at the time for executing ten allocations when latency=0.*

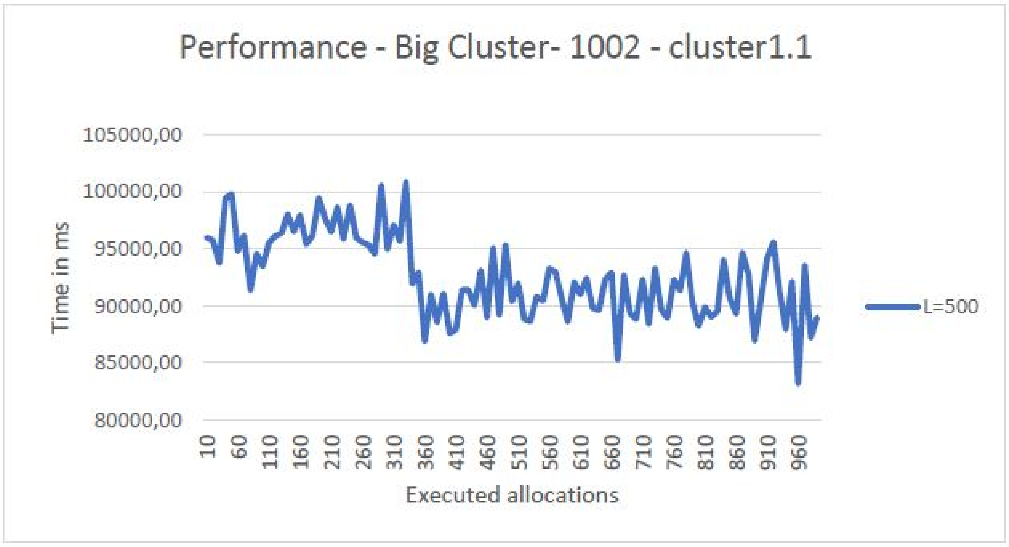


Figure 5.11: This graph is the same graph as shown in figure shows 5.7. It is enlarged to get a close look at the time for executing ten allocations when latency=500.

Figure 5.8 and figure 5.9 shows an enlarged view of the performance graph from figure 5.6, when latency = 0 and latency = 500 on $C1_0$. The same applies for figure 5.10 and figure 5.11 which shows an enlarged view of the performance graph from figure 5.7, when latency = 0 and latency = 500 on $C1_1$.

57Kbit/s bandwidth Low bandwidth limits the amount of traffic that transmits and when the traffic transmits, it can consist of all possible combinations of latency in the connection. It is therefore difficult to have control over latency and it will not be adjusted in this tests.

	Cluster 1.0 ($C1_0$)	**Cluster 1.1 ($C1_1$)**
Bandwidth: 57Kbit/s		
Subscribers	**1.002**	**1.002**
Total time for all transactions	3854,68s (1,07 t)	3735,916s (1,03 t)
Average time per transaction	3,84s	3,73s
Error rate	0	0

Table 5.4: *TC1 - 1.002 subscribers*

Table 5.4 shows the results from the test when the bandwidth was 57Kbit/s for $C1_0$ and $C1_1$. It shows the amount of subscribers, the total time for all transactions, the average time per transaction and if the test had any package loss or down time during the execution of the test. 0 means that there was no package loss or down time during the test.

Performance was measured for $C1_0$ and $C1_1$ by logging the total time of every tenth allocation.

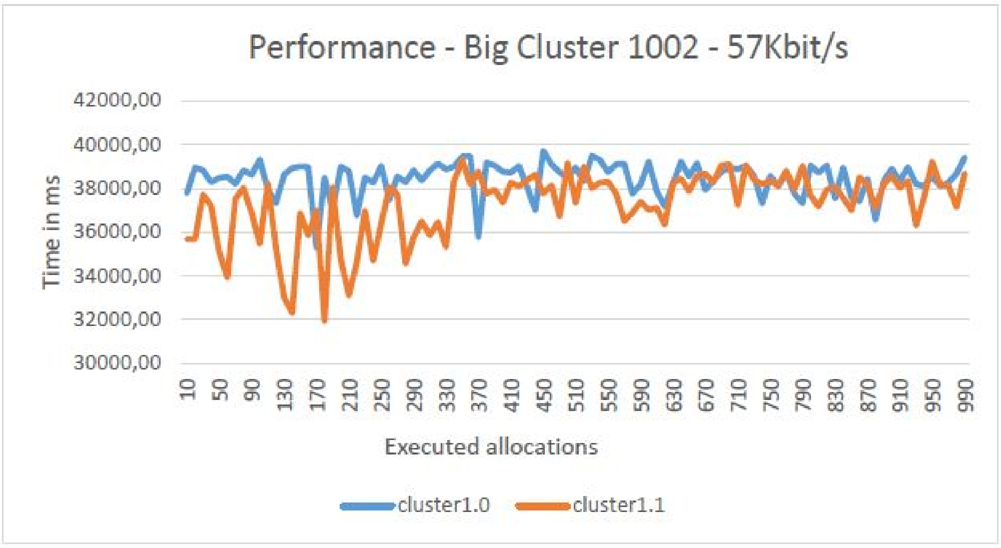


Figure 5.12: *The graph shows the performance of every tenth allocation for $C1_0$ and $C1_1$ when bandwidth is 57Kbit/s. The X-axis shows the number of allocations, the Y-axis shows time in milliseconds.*

The graph in figure 5.12 shows the performance that was measured for $C1_0$ and $C1_1$ when bandwidth was 57Kbit/s, latency was not adjusted. The Y-axis from this graph goes from 30.000ms - 42.000ms. The differences between the performance when the bandwidth is 1Gb/s compared to when the bandwidth is 57Kbit/s will be looked at in the discussion part.

TC2: The AllocateIMSI procedure allocated 10.002 IPCO subscribers concurrent in clusters1.0 and $C1_1$.
The first test result to be presented is the result from the test where bandwidth = 1Gb/s, followed by bandwidth = 57Kbit/s.

1Gb/s bandwidth: Just like TC1, five combinations of round-trip latency and bandwidth were tested for all subscribers in TC2.

The amount of subscribers used in this test, the total time for all transactions, the average time per transaction and the error rate can be seen in table 5.5. 0 means that there were no package loss or down time during the test.

The graph shown in figure 5.13 shows the total time in seconds for all transactions for both $C1_0$ and $C1_1$. The total time for all transactions when latency=0 is 225,20 seconds and not close to 0, as it looks like in the graph.
It is difficult to see the lines for both clusters because the values are very similar and therefore they overlap each other. The values used in the graph can be seen in table 5.5, total time for all transactions.

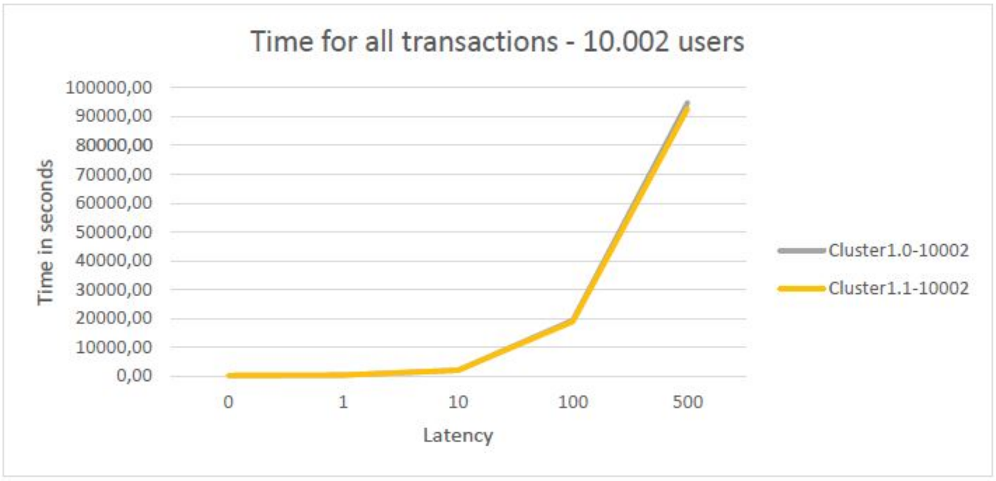


Figure 5.13: *Graph shows the total time for all transactions when the bandwidth is 1Gb/s. X-axis shows the latency from 0-500, while the Y-axis shows time in seconds.*

	Cluster 1.0 ($C1_0$)					**Cluster 1.1 ($C1_1$)**				
Bandwidth: 1Gb/s	Latency:0	L:1	L:10	L:100	L:500	Latency:0	L:1	L:10	L:100	L:500
Subscribers	**10.002**	**10.002**	**10.002**	**10.002**	**10.002**	**10.002**	**10.002**	**10.002**	**10.002**	**10.002**
Total time for all transactions	225,20s	401,40s	2087,60s	19374,20s	94950,88s	224,70s	395,00s	2037,10s	18907,78s	92792,59s
Average time per transaction	0,0225s	0,0401s	0,2080s	1,937s	9,49s	0,0224s	0,0390s	0,2030s	1,89s	9,27s
Error rate	0	0	0	0	0	0	0	0	0	0

Table 5.5: *TC2 - 10.002 subscribers*

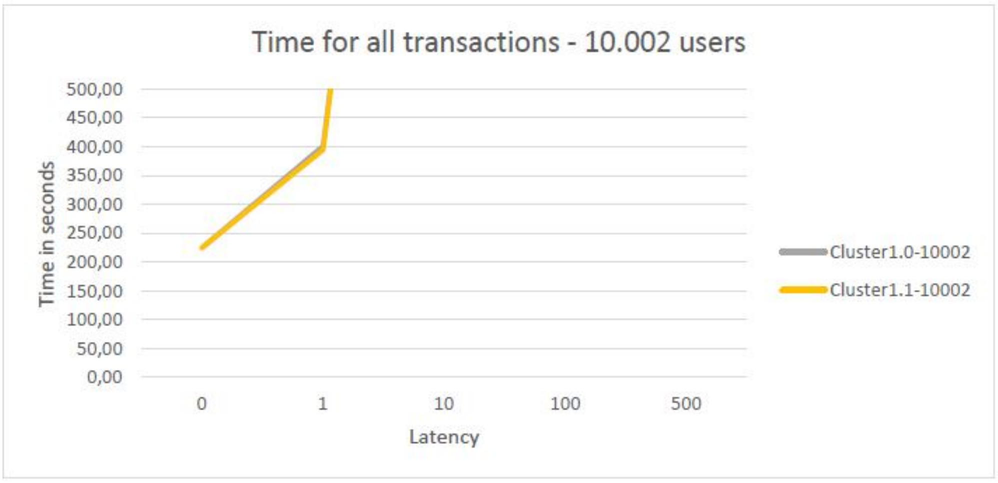


Figure 5.14: *This graph is the same graph as shown in figure 5.13. This graph is enlarged for a closer look at the values when latency=0 and 1.*

As shown in figure 5.13 and 5.14, the graphs have breakpoints on the scale when latency is 1, 10 and 100, just like the graph in figure 5.4 from TC1. This result will be discussed in the discussion part together with the results from TC1.

Performance is measured for $C1_0$ and $C1_1$ by logging the total time of every tenth allocation for all the variations of round-trip latency.

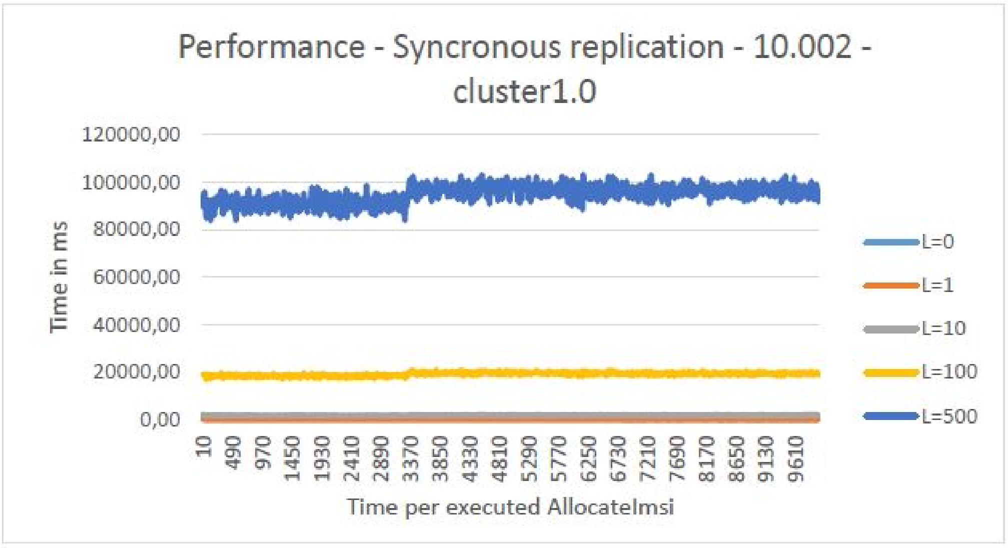


Figure 5.15: *The graph shows the performance of every tenth allocation for $C1_0$ when the bandwidth is 1Gb/s. The X-axis shows the number of allocations, the Y-axis shows time in milliseconds.*

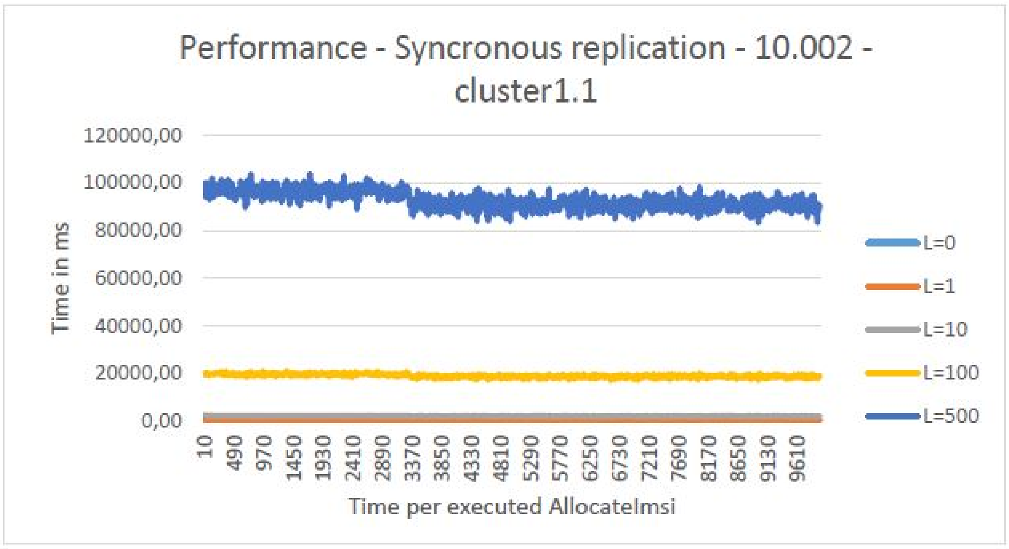


Figure 5.16: *The graph shows the performance of every tenth allocation for $C1_1$ when the bandwidth is 1Gb/s. The X-axis shows the number of allocations, the Y-axis shows time in milliseconds..*

The graphs shown in figure 5.15 and figure 5.16 show the performance measured for $C1_0$ and $C1_1$ when latency was 0, 1, 10, 100 and 500. The measurements done on the different variations of latency for $C1_0$ and $C1_1$ look similar when the two graphs are compared. This is because of the large scale on the graphs. Y-axis ranges from 0ms - 120.000ms and X-axis is from 10 - 9.610. To see that the measurements varies more when the latency increases, both graphs in figure 5.15 and figure 5.16 are enlarged on latency = 0 and latency = 500.

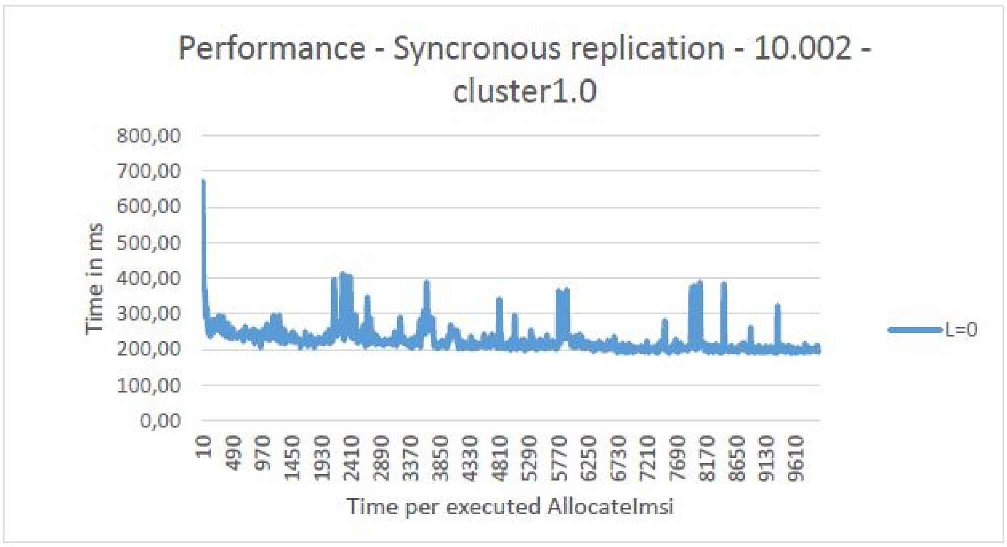


Figure 5.17: *This graph is the same graph as shown in figure 5.15, performance from $C1_0$. It is enlarged to get a closer look at the time for executing ten allocations when latency=0.*

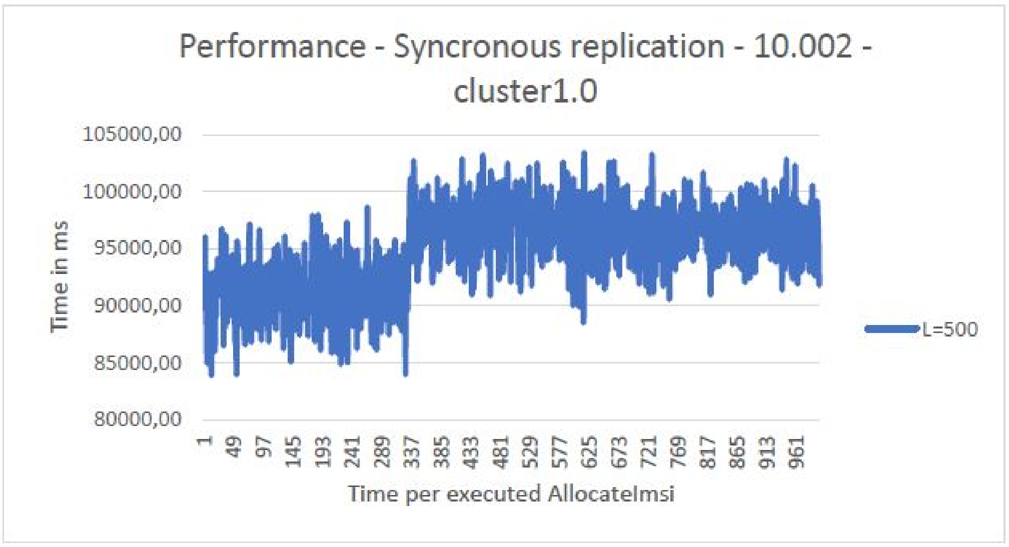


Figure 5.18: *This graph is the same graph as shown in figure 5.15, performance from* $C1_0$. *It is enlarged to get a closer look at the time for executing ten allocations when latency=500.*

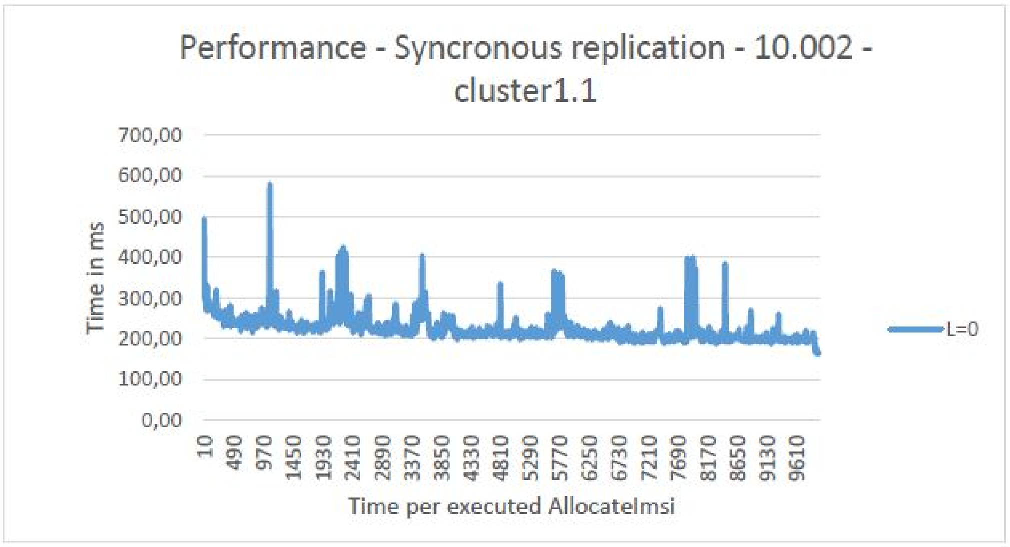


Figure 5.19: *This graph is the same graph as shown in figure 5.16, performance from* $C1_0$. *It is enlarged to get a closer look at the time for executing ten allocations when latency=0.*

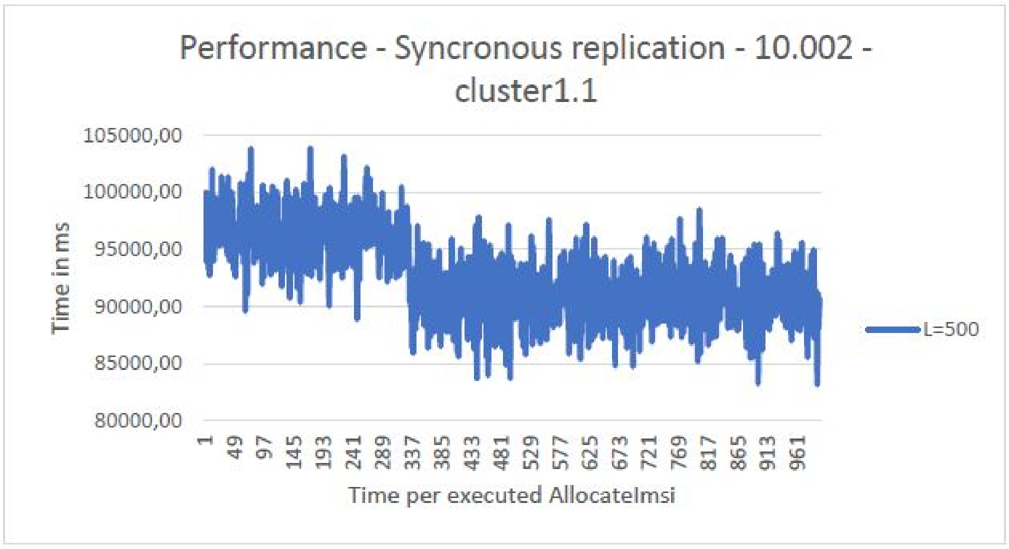


Figure 5.20: *This graph is the same graph as shown in figure 5.16, performance from $C1_0$. It is enlarged to get a closer look at the time for executing ten allocations when latency=500.*

Figure 5.17 and figure 5.18 shows an enlarged view of the performance graph from figure 5.15, when latency = 0 and latency = 500 on $C1_0$. The same applies for Figure 5.19 and figure 5.20, which shows an enlarged view of the performance graph from figure 5.16, when latency = 0 and latency = 500 on $C1_1$.

57Kbit/s bandwidth: As mentioned in TC1, the latency will not be adjusted in this test.

	Cluster 1.0 ($C1_0$)	**Cluster 1.1 ($C1_1$)**
Bandwidth: 57Kbit/s		
Subscribers	**10.002**	**10.002**
Total time for all transactions	38287,87s (10,63 t)	37550,44s (10,66 t)
Average time per transaction	3,828s	3,754s
Error rate	0	0

Table 5.6: *TC1 - 10.002 subscribers*

Table 5.6 shows the results from the test when the bandwidth was 57Kbit/s for $C1_0$ and $C1_1$. It shows the amount of subscribers, the total time for all transactions, the average time per transaction and if the test had any package loss or down time during the execution of the tests. 0 means that there were no package loss or down time during the tests.

Performance are measured for $C1_0$ and $C1_1$ by logging the total time of every tenth allocation, like TC1.

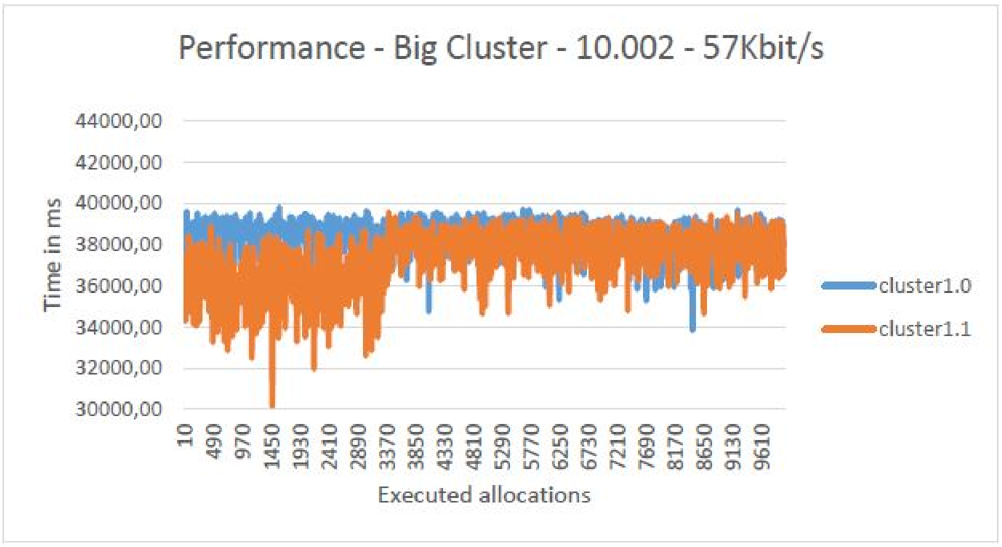


Figure 5.21: *The graph shows the performance of every tenth allocation for* $C1_0$ *and* $C1_1$ *when bandwidth is 57Kbit/s. The X-axis shows the number of allocations, the Y-axis shows time in milliseconds.*

The graph in figure 5.21 shows the performance that is measured for $C1_0$ and $C1_1$ when bandwidth is 57Kbit/s, latency is not adjusted. The Y-axis from this graph goes from 30.300ms - 44.000ms. The differences between the performance when the bandwidth is 1Gb/s compared to when the bandwidth is 57Kbit/s will also here be looked at in the discussion part.

5.3.2 Asynchronous Replication

The aim of this test was to see how C1 and C2 and the replication between them managed to cope with the system which allocates IMSIs for several subscribers at the same time over a bad network connection that is affected by latency and low bandwidth. This is also used to see how this influenced the performance, replication delay and the possibility to lose data over lazy replication. The performance error rate and replication delay are tested for this environment.

TC1: The AllocateIMSI procedure allocated 1.002 IPCO subscribers concurrent in both clusters.

The first test that was conducted in TC1 was with a bandwidth of 1Gb/s, followed by a bandwidth of 57Kbit/s.

1Gb/s bandwidth: Five combinations of round-trip latency and bandwidth are tested for subscribers in TC1.

Table 5.7 shows the amount of subscribers that was used in this test, the total time for all transactions, the average time per transaction, maximum replication delay and the error rate. 0 means that there was no package loss or down time during the test.
The result from total time for all transactions is based on the average of ten measurements per test. This is done to get the total time for all transactions as exactly as possible because the times vary depending on which of the SQL nodes the load balancer is running against.

	Cluster 1 (C1)					**Cluster 2 (C2)**				
Bandwidth: 1Gb/s	Latency:0	L:1	L:10	L:100	L:500	Latency:0	L:1	L:10	L:100	L:500
Subscribers	**1.002**	**1.002**	**1.002**	**1.002**	**1.002**	**1.002**	**1.002**	**1.002**	**1.002**	**1.002**
Total time for all transactions	22,10s	22,60s	21,00s	20,71s	21,11s	20,20s	21,30s	20,25s	19,67s	20,64s
Average time per transaction	0,022s	0,022s	0,021s	0,020s	0,021s	0,020s	0,021s	0,020s	0,019s	0,020s
Max replication delay	1s	1s	1s	1s	1s	1s	1s	1s	1s	1s
Error rate	0	0	0	0	0	0	0	0	0	0

Table 5.7: *TC1 - 1.002 subscribers*

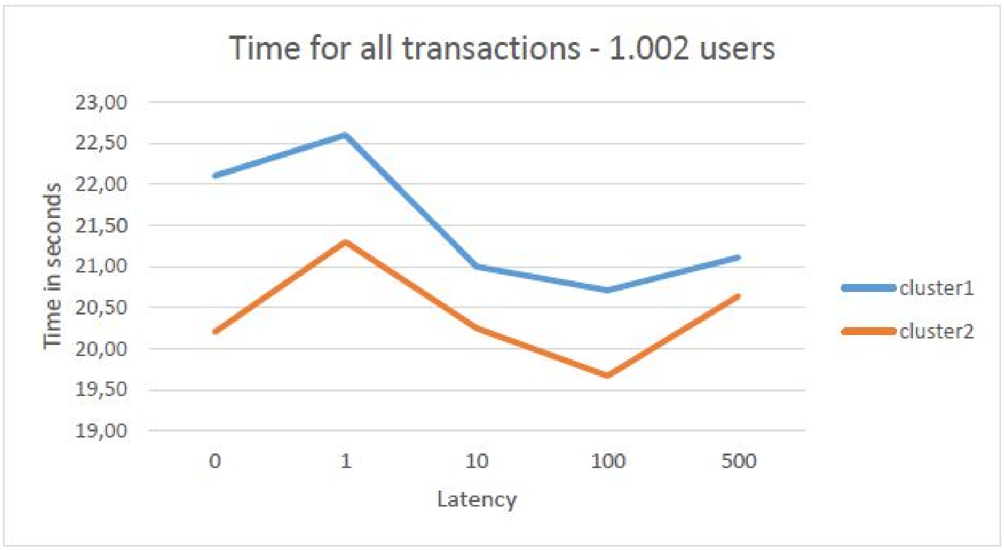


Figure 5.22: *The graph shows the total time for all transactions when the bandwidth is 1Gb/s. X-axis shows the latency that is tested from 0-500, while Y-axis shows time in seconds from 19s-23s.*

The graph shown in figure 5.22, shows the total time in seconds for all transactions for C1 and C2. The values in this graph can be found in table 5.7, total time for all transaction.

The performance from each tenth allocation was measured for C1 and C2. This is measured by logging the total time of every tenth allocations for all variations of round-trip latency.

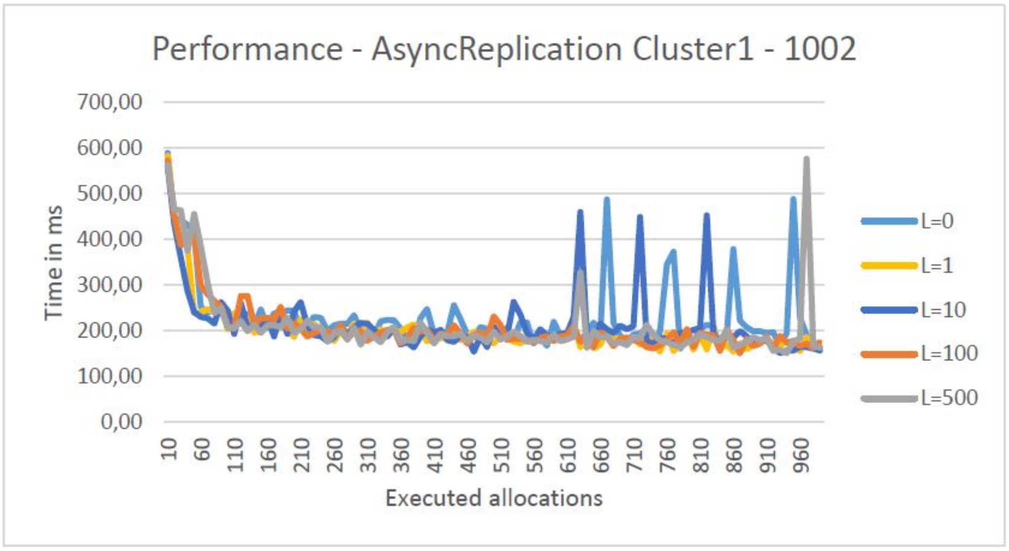


Figure 5.23: *The graph shows the performance of every tenth allocation for C1 when the bandwidth is 1Gb/s. The X-axis shows the number of allocations, the Y-axis shows time in milliseconds.*

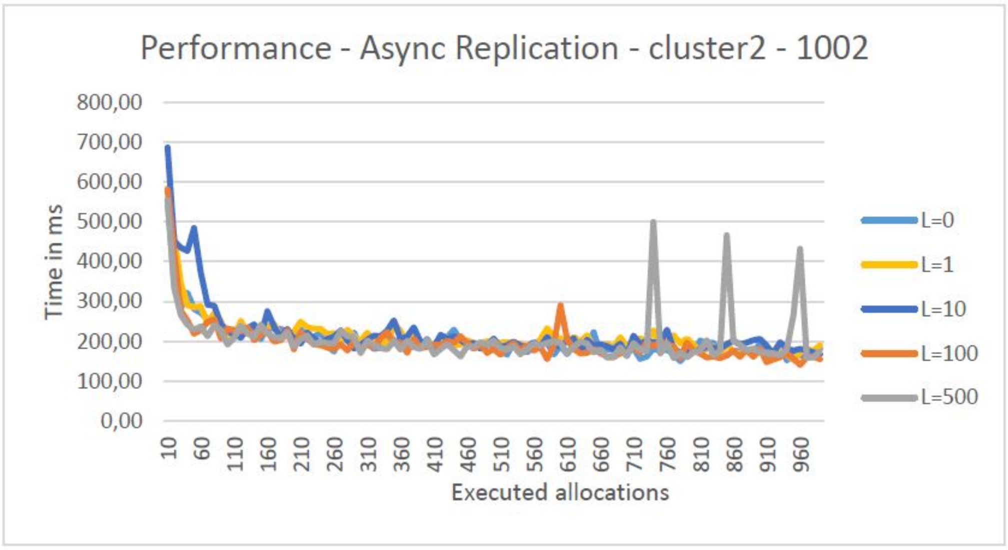


Figure 5.24: *The graph shows the performance of every tenth allocation for C2 when the bandwidth is 1Gb/s. The X-axis shows the number of allocations, the Y-axis shows time in milliseconds.*

The graph from figure 5.23 and figure 5.24 shows the performance that was measured when the round-trip latency was 0, 1, 10, 100 and 500 on C1 and C2. The measurements done on C1 and C2 showed that the first 10 allocations took the longest time 600ms-700ms, while the following allocations used around 200ms, with a few exceptions. This graphs will be looked at in the discussion.

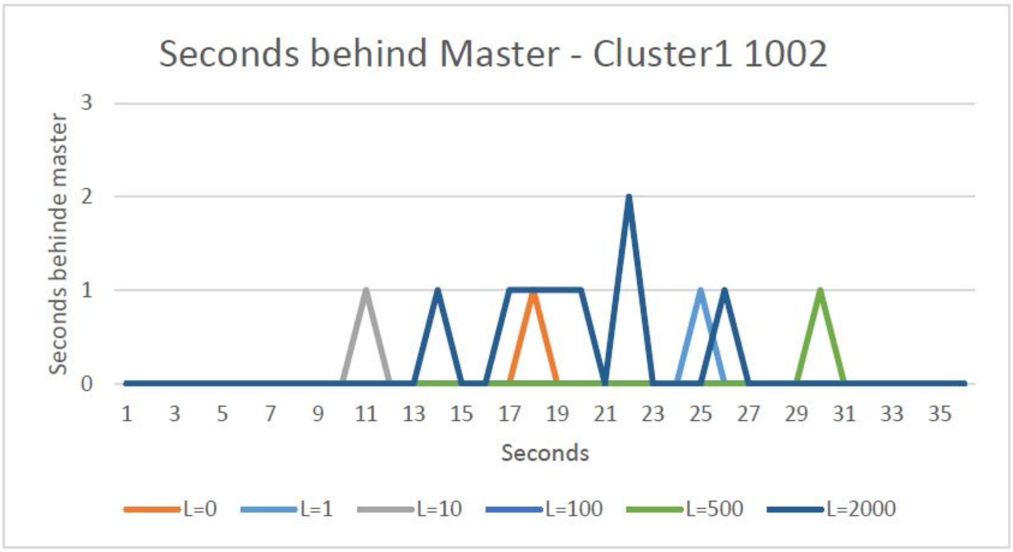


Figure 5.25: *The graph shows how many seconds C1 is behind the Master (C2) when the bandwidth is 1Gb/s. X-axis shows seconds from 0-3, while Y-axis shows seconds for the transaction.*

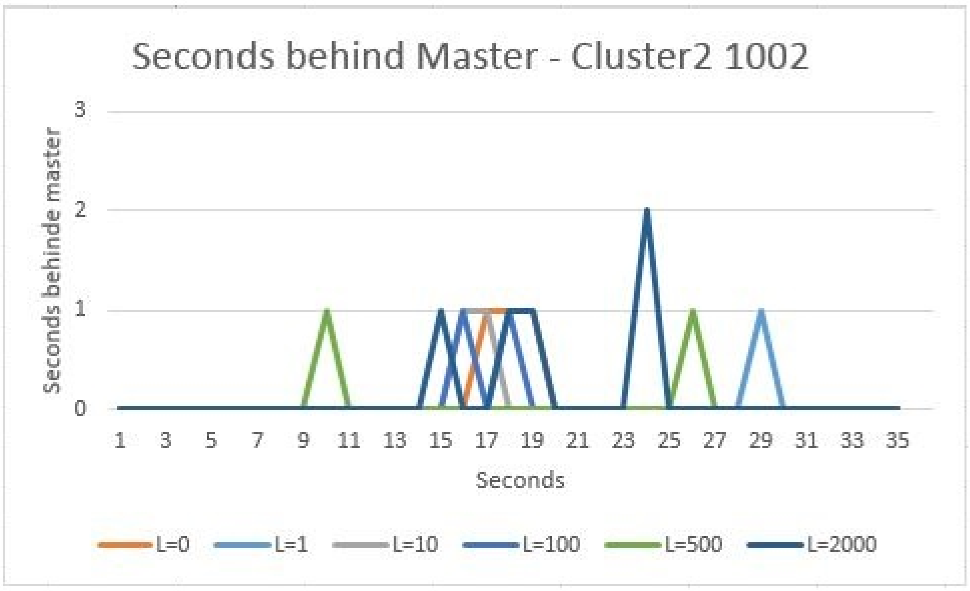


Figure 5.26: *The graph shows how many seconds C2 is behind the Master (C1) when the bandwidth is 1Gb/s. X-axis shows seconds from 0-3, while Y-axis shows seconds for the transaction.*

The graphs in figure 5.25 shows how many seconds C1 is behind C2. Figure 5.26 shows how many seconds C2 is behind C1. The replication delay behind the master is also tested with a round-trip latency that is 2000, to see how the replication delay reacted.

57Kbit/s bandwidth: Low bandwidth limits the amount of traffic that transmits and when the traffic transmits, it can consist of all possible combinations of latency in the connection. It is therefore difficult to have control over the latency, and it will not be adjusted in this tests.

	Cluster 1 (C1)	**Cluster 2 (C2)**
Bandwidth: 57Kbit/s		
Subscribers	**1.002**	**1.002**
Total time for all transactions	28,1s	24,20s
Average time per transaction	0,028s	0,0242s
Max replication delay	22s	5s
Error rate	0	0

Table 5.8: *TC1 - 1.002 subscribers*

Table 5.8 shows the results from the test where bandwidth was is 57Kbit/s for C1 and C2. The table also shows the amount of subscribers used in this test, the total time for all transactions, the average time per transaction, max replication delay and the error rate. 0 means that there was no package loss or down time during the test.
The result from total time for all transactions are based on the average of ten measurements per

test. This is done to get the total time for all transactions as exactly as possible because the times vary depending on which of the SQL nodes the load balancer is running against.

The performance is measured for C1 and C2 by logging the total time of every tenth allocation.

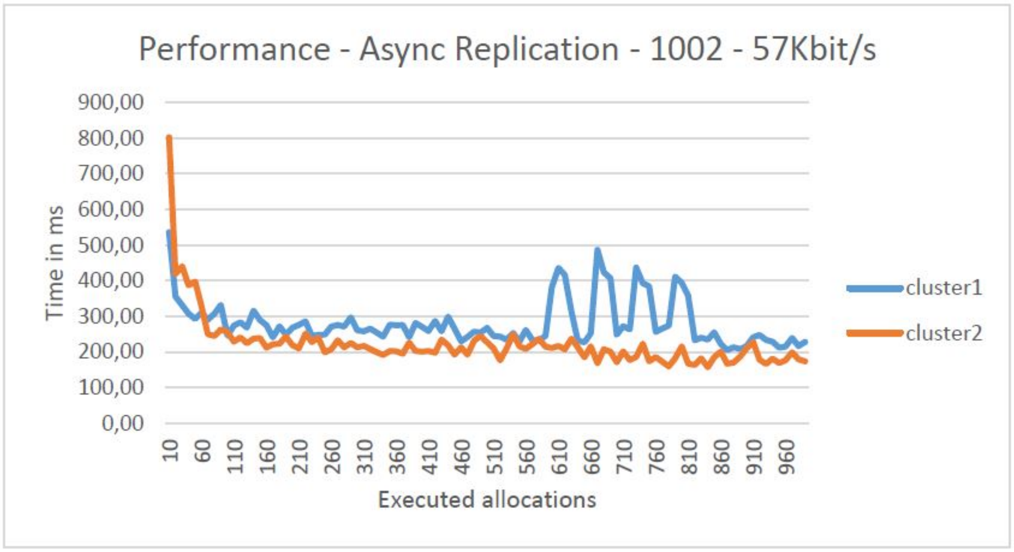


Figure 5.27: *The graph shows the performance of every tenth allocation for C1 and C2 when bandwidth is 57Kbit/s. The X-axis shows the number of allocations, the Y-axis shows time in milliseconds.*

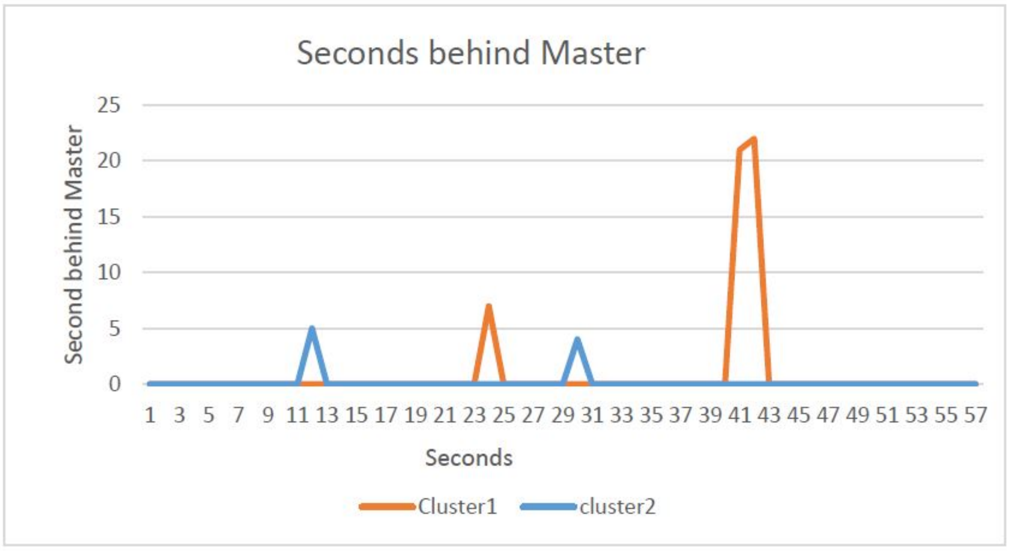


Figure 5.28: *The graph shows by the orange line how many seconds C1 is behind the Master (C2) and the blue line shows how many seconds C2 is behind the Master (C1) when bandwidth is 57Kbit/s. X-axis shows seconds from 0-25, while Y-axis shows seconds for the transaction.*

Figure 5.27 shows the performance graph where the results from C1 and C2 are presented. C1 which is the blue line has a start value of the ten first allocations on 536ms. The other allocations in C1 uses between 300ms - 200ms with a change in the pattern between allocation No. 560-810. C2, which is the orange line shows a start value for the ten first allocations on 803ms, while the other allocations used around 200ms.

The graph in figure 5.28 shows the second behind for C1 and C2 in the same graph. C1 is 22 seconds behind C2 at the most, while C2 is 5 seconds behind C1 at the most.

TC2: The AllocateIMSI procedure is going to allocate 10.002 IPCO subscribers concurrent in both clusters.

1Gb/s bandwidth: Table 5.9 shows the amount of subscribers used in this test, the total time for all transactions, the average time per transaction, max replication delay and the error rate. 0 means that there was no package loss or down time during the test.
The result from total time for all transactions are based on the average of ten measurements per test. This is done to get the total time for all transactions as exactly as possible because the times vary depending on which of the SQL nodes the load balancer is running against.

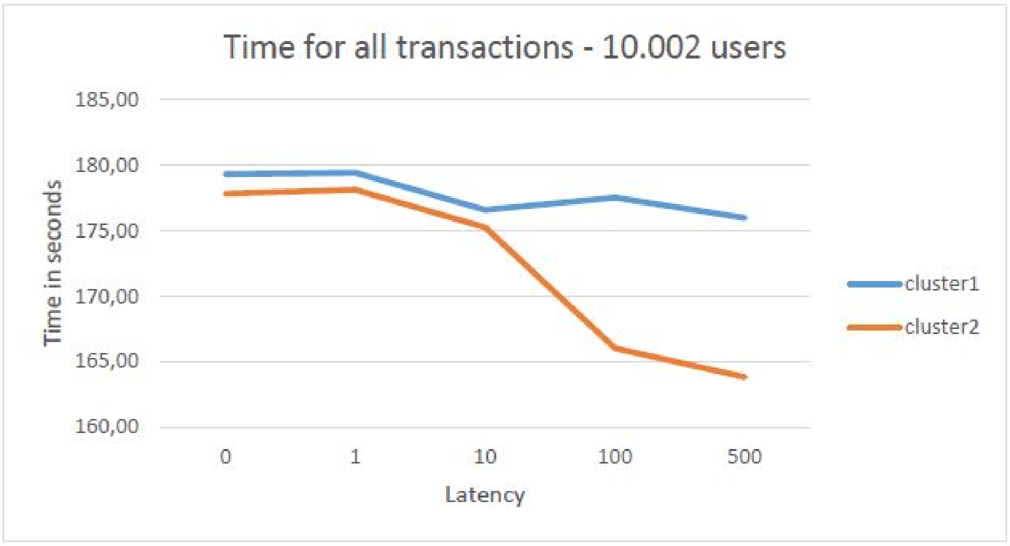


Figure 5.29: *Graph shows the total time for all transactions when the bandwidth is 1Gb/s. X-axis shows the latency that is tested from 0-500, while Y-axis shows time in seconds from 160s-180s.*

The graph shown in figure 5.29, shows the total time in seconds for all transactions for both C1 and C2. The values in this graph can be seen in table 5.9, total time for all transactions.

The performance from each tenth allocation is measured for C1 and C2. This is measured by logging the total time of every tenth allocation for all variations of round-trip latency.

	Cluster 1 (C1)					**Cluster 2 (C2)**				
Bandwidth: 1Gb/s	Latency:0	L:1	L:10	L:100	L:500	Latency:0	L:1	L:10	L:100	L:500
Subscribers	**10.002**	**10.002**	**10.002**	**10.002**	**10.002**	**10.002**	**10.002**	**10.002**	**10.002**	**10.002**
Total time for all transactions	179,30s	179,40s	176,56s	177,50s	175,95s	177,80s	178,10s	175,21s	166,00s	163,80s
Average time per transaction	0,017s	0,017s	0,017s	0,017s	0,017s	0,017s	0,017s	0,017s	0,017s	0,016s
Max replication delay	1s	1s	1s	1s	1s	1s	1s	1s	1s	1s
Error rate	0	0	0	0	0	0	0	0	0	0

Table 5.9: *TC2 - 10.002 subscribers*

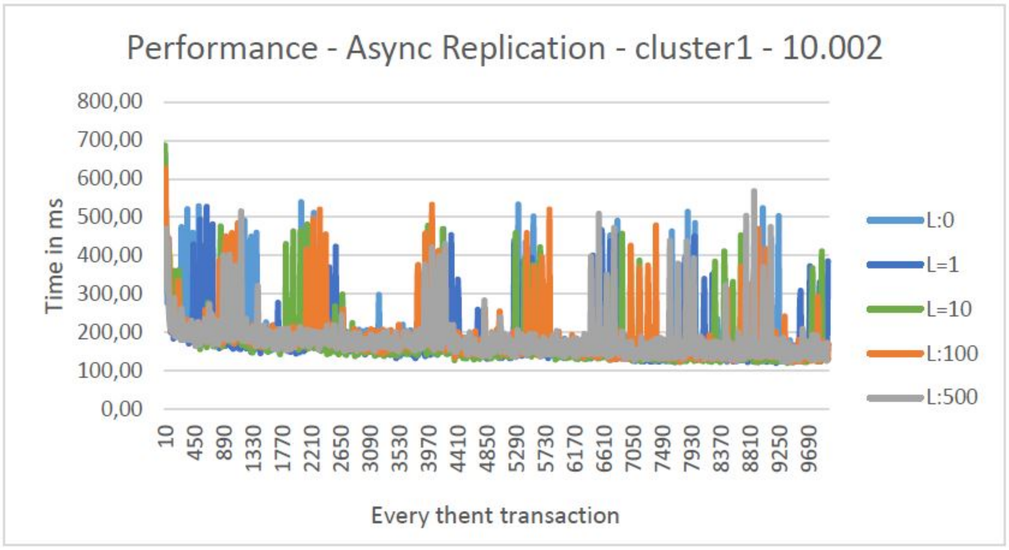


Figure 5.30: *The graph shows the performance of every tenth allocation for C1 when the bandwidth is 1Gb/s. The X-axis shows the number of allocations, the Y-axis shows time in milliseconds.*

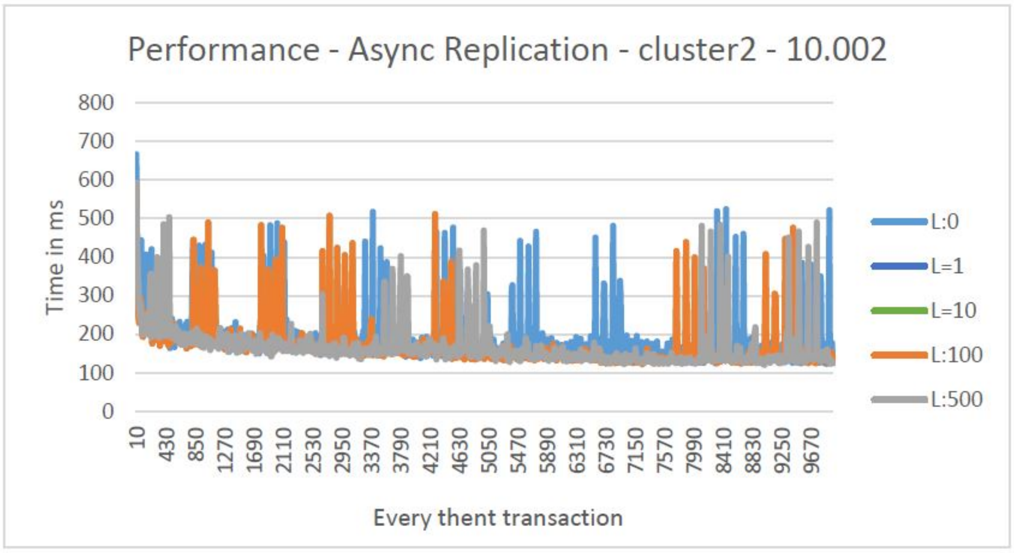


Figure 5.31: *The graph shows the performance of every tenth allocation for C2 when the bandwidth is 1Gb/s. The X-axis shows the number of allocations, the Y-axis shows time in milliseconds.*

The graph in figure 5.30 and figure 5.31 shows the performance that was measured when the round-trip latency was 0, 1, 10, 100 and 500 on C1 and C2. The measurement done on C1 and C2 showed that the first 10 allocations used around 700ms, while the following allocations varied between 500ms - 150ms. This graphs will be looked at in the discussion.

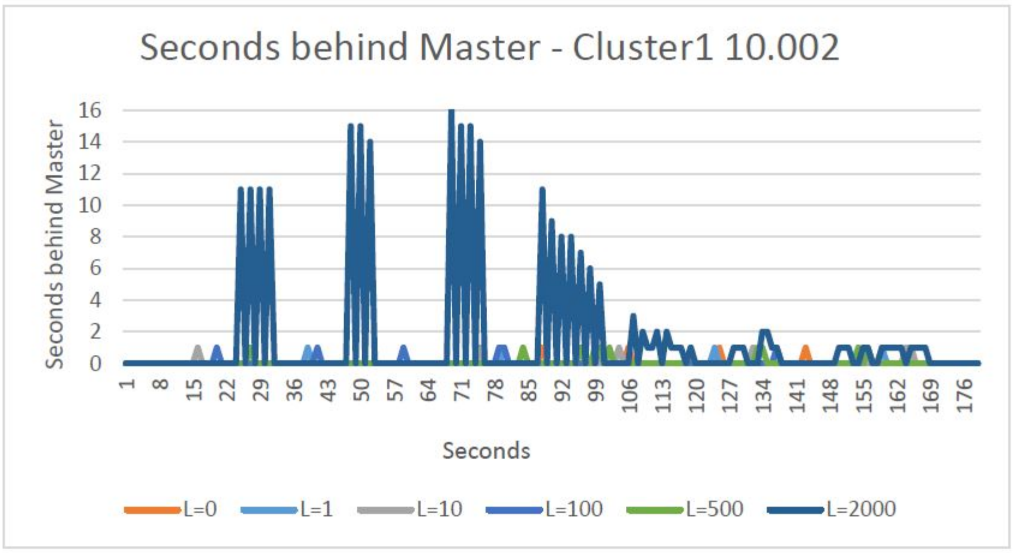


Figure 5.32: *The graph show how many seconds C1 is behind the Master (C2) when the bandwidth is 1Gb/s. X-axis shows seconds from 0-16, while Y-axis shows seconds for the transaction C1 second behind the Master.*

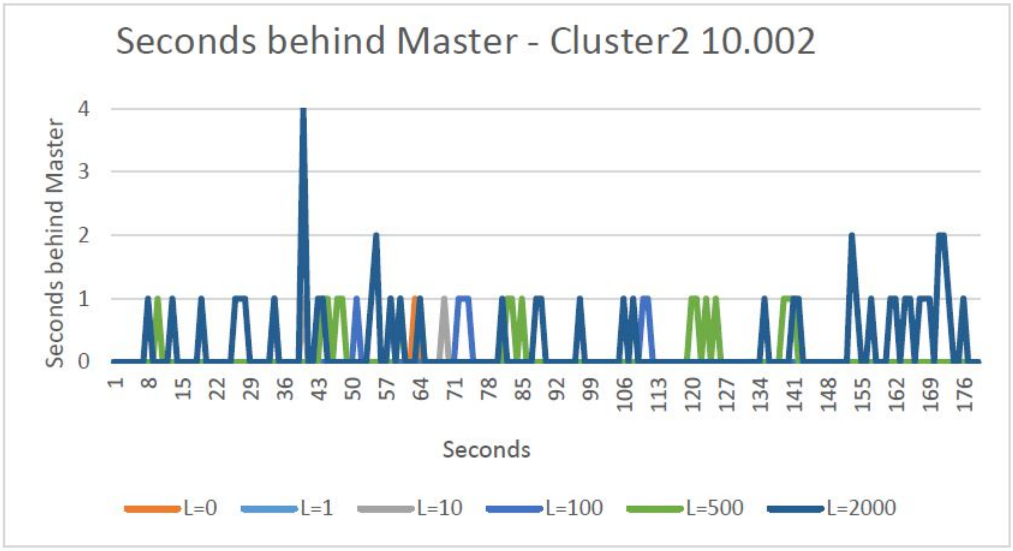


Figure 5.33: *The graph show how many seconds C2 is behind the Master (C1) when the bandwidth is 1Gb/s. X-axis shows seconds from 0-4, while Y-axis shows seconds for the transaction.*

The graph in figure 5.32 shows how many seconds C1 is behind C2 and figure 5.33 shows how many seconds C2 is behind C1. The replication delay behind the master is also tested with a round-trip latency that is 2000, to see how the replication delay reacted. C1 was 16 seconds behind behind C2, at the most with a round-trip latency on 2000. C2 was 4 seconds behind C1 at the most, with a round-trip latency on 2000.

	Cluster 1 (C1)	**Cluster 2 (C2)**
Bandwidth: 57Kbit/s		
Subscribers	**10.002**	**10.002**
Total time for all transactions	190s	195,9s
Average time per transaction	0,019s	0,0195s
Max replication delay	422s	189s
Error rate	0	0

Table 5.10: *TC1 - 10.002 subscribers.*

57Kbit/s bandwidth: Table 5.10 shows the results from the test where bandwidth was 57Kbit/s for both C1 and C2. The table also shows the amount of subscribers used in this test, the total time for all transactions, the average time per transaction, max replication delay and the error rate. 0 means that there was no package loss or down time during the tests.
The result from total time for all transactions is based on the average of ten measurements per test. This is done to get the total time for all transactions as exactly as possible because the times vary depending on which of the SQL nodes the load balancer is running against.

The performance is measured for C1 and C2 by logging the total time of every tenth allocation.

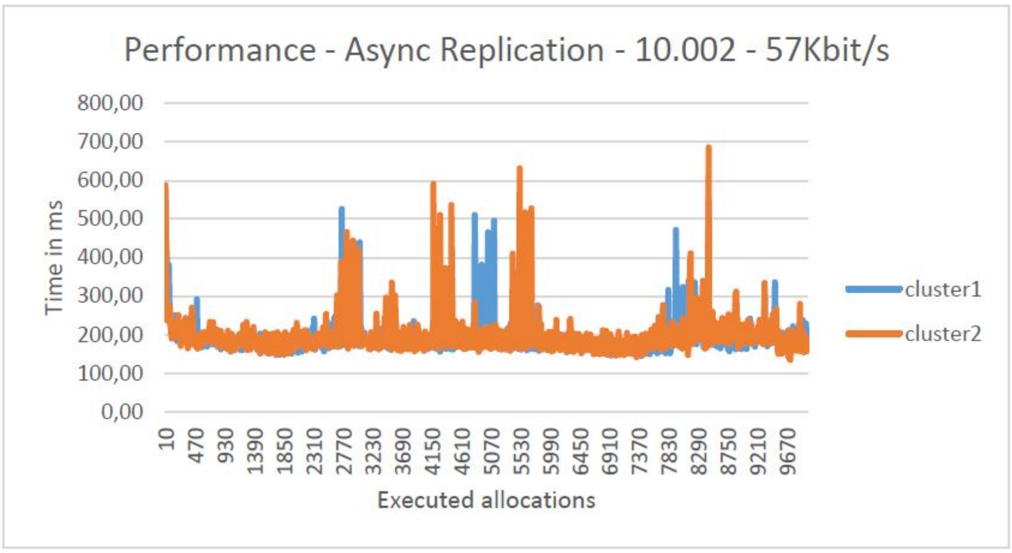


Figure 5.34: *The graph shows the performance of every tenth allocation for C1 and C2 when bandwidth is 57Kbit/s. The X-axis shows the number of allocations, the Y-axis shows time in milliseconds.*

Figure 5.34 shows the performance graph where the results from C1 and C2 are presented. C1 which is shown as a blue line, has a start value for the ten first allocations on 572ms. The other allocations in C1 uses between 300ms - 150ms, with a change in the pattern between allocation

No. 2770-5990 and between 7830-8750. C2 which is shown as a orange line, shows a start value for the ten first allocations on 588ms while the other allocations used around 200ms with a change in the pattern between allocation No. 2310-3230, 4610-5070 and between 7830-8290.

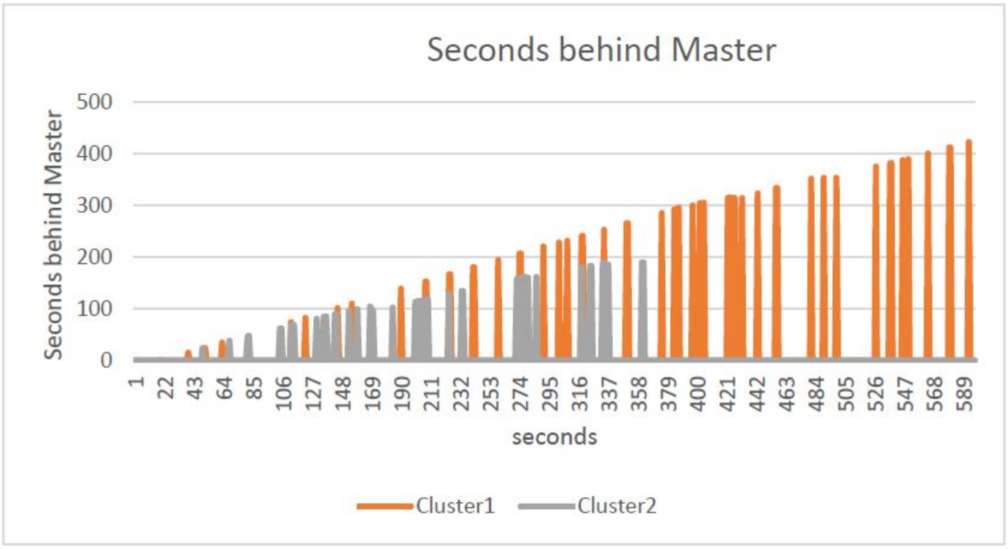


Figure 5.35: *The graph shows by the orange line how many seconds C1 is behind the Master (C2) and by the gray line how many seconds C2 is behind the Master (C1) when bandwidth is 57Kbit/s X-axis shows seconds from 0-500, while Y-axis shows seconds for the transaction.*

The graph in figure 5.35 shows the second behind for C1 and C2 in the same graph. C1 is 422 seconds behind C2 at the most, while C2 is 189 seconds behind C1 at the most.

This chapter has looked into how the tests were conducted and how the test data was collected, followed by a review of the test results from both environments presented in a statistical way. Based on the test results from this chapter, each of the underlying research questions will be discussed in the next chapter. The discussion chapter will consist of two parts. The first part is a database technical discussion based on the theory and the second part is a recommendation of the test environments to be presented for IPCO.

Chapter 6

Discussion

This discussion chapter is divided into two parts. The first part is a database technical discussion based on the database theory and the test results from the two test environments. The second part it is a recommendation to IPCO, recommending which of the test environments that are best suited for use in the production environment. The recommendation is based on the test results and IPCO's demands for environments.
Each of the two parts in this chapter discusses the underlying research questions, while the conclusion is based on the main research question.

6.1 Database technical

This part of the discussion is a general database discussion and looks at the test results in relation to the database theory from chapter 2, Background.

1A: In what way does bandwidth and latency affect the performance on the two cluster topologies.

Asynchronous Replication environment: When the network connection is bad or the replication distance is long, the replication may be affected by low bandwidth and high latency.
The time it takes for the Asynchronous test environment to execute the procedure AllocateIMSI [1], can be described as following: $T_{transaction}$.
$T_{transaction}$ is the measured time seen from the node on which the transaction was initiated. $T_{latency}$ is the applied latency in the network. In theory, $T_{latency}$ will not have any impact on $T_{transaction}$, since the transaction on the Asynchronous Replication environment is the local transaction time. One of the purposes with this thesis is to investigate that the Asynchronous Replication environment does not cause increased transaction time when the network latency increases. The latency is added between C1 and C2, and not inside the clusters. Since the replication between the clusters is lazy, the replication will not take place before the master cluster transaction is committed and the slave has time to update the transaction. For the Asynchronous Replication test environment, the average time per transaction in TC1[2] is between 0,019 - 0,022 seconds, while the average time per transaction in TC2[3] is 0,016-0,017 seconds. The difference between the results from the average

[1]The procedure AllocateIMSI allocates visitor IMSIs (International Mobile Subscriber Identity) in the database to IPCO subscribers which are traveling in foreign countries that have IPCO partners. This makes it possible to use the "roam as a local" service.
[2]Test case 1: The AllocateIMSI procedure is going to allocate 1.002 IPCO subscribers concurrent in both clusters.
[3]Test case 2: The AllocateIMSI procedure is going to allocate 10.002 IPCO subscribers concurrent in both clusters.

time per transaction for TC1 and for TC2 can be explained by looking at the performance graphs. An example of a performance graph is shown in figure 6.1. The same graph is also presented in figure 5.23, section 5.3.2 Asynchronous Replication. The performance graphs for the Asynchronous Replication environment for TC1 shows that the first ten transactions used between 600-700ms to execute, while the other used around 200ms.

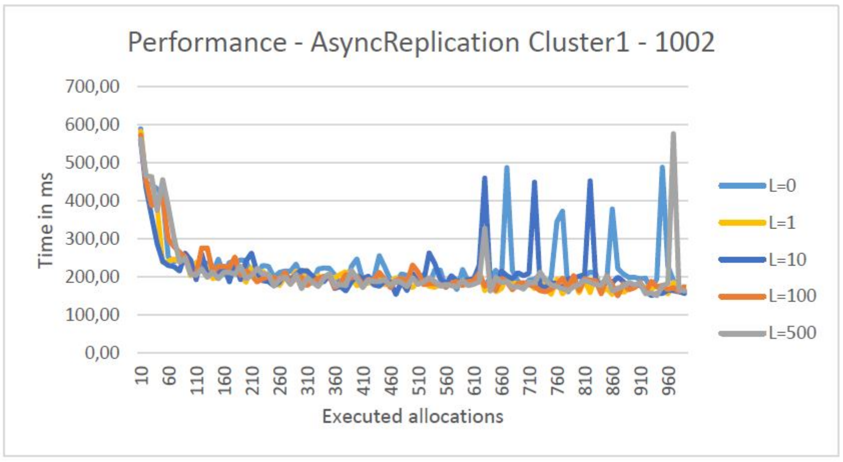


Figure 6.1: *TC1- Graph shows the performance of every tenth allocations for C1 when the bandwidth is 1Gb/s. X-axis shows how many allocations are performed, while the Y-axis shows the time in ms.*

The performance graph for the Asynchronous Replication environment for TC2 shows that the first ten transactions used 700ms, while the rest of the transaction used around 200ms to execute. As an example, see graph in figure 6.2 (the same graph is also presented in figure 5.30, section 5.3.2 Asynchronous Replication)

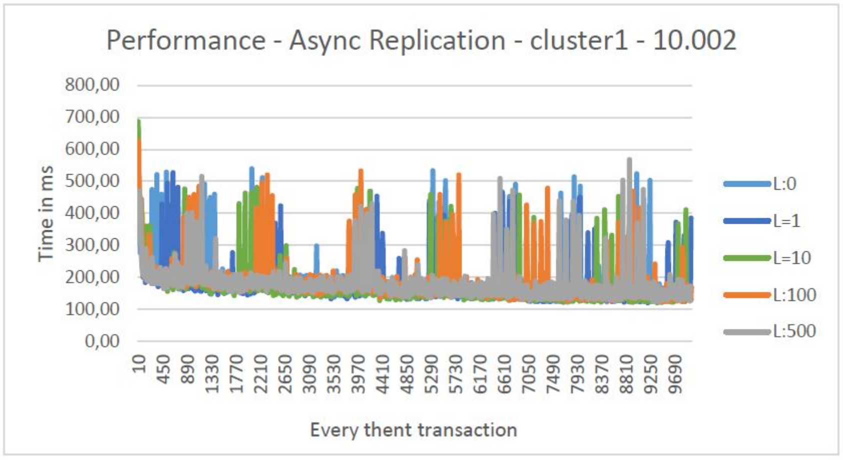


Figure 6.2: *TC2 - Graph shows the performance of every tenth allocations for C1 when the bandwidth is 1Gb/s. X-axis shows how many allocations are performed, while the Y-axis shows the time in ms.*

The performance graphs from TC1 and TC2 where the bandwidth is 1Gb/s, show a few exceptions where the time for every tenth transaction takes more than 200ms, this may be caused by:

- How the indexes are sorted.
  During data modifications, indexed columns requires modifications to the indexes. Usually there is space within the index structure to insert new items, but sometimes the indexes needs to be reordered (new "pages" added) which requires additional resorting of parts of the indexes.
- Hash index recalculation.
  MySQL calculates hash values for the hash indexes that allow for a finite number of entries in the hashed tables. If the number of entries in a hashed table becomes larger than the number allowed in the hash value, MySQL automatically recalculates the hash values to allow a larger number of entries (In other words, allow for a larger hash table). It is possible to configure the number of entries a hashed index shall hold to avoid ever reaching the limit, but this is beyond the scope of this thesis.
- I/O delay in connection with writing to the disk.
  Due to hardware limitations, each node in the MySQL Cluster runs on only one CPU kernel. This will result in some delay to the cluster operation when the CPU is busy persisting the data to the disk.
- Network devices and shared CPU resources in VMware.
  The cluster share network resources with other virtual machines in the VMware installation. This will cause some variation in the network performance offered to the MySQL Cluster nodes by VMware. The CPUs are shared by other virtual machines in the VMware installation. Switching of CPU resources between virtual machines by the VMware OS can result in variation in performance and delay of the virtual machines making up the MySQL Cluster. These are a well-known issues when running on VMware. Because of these issues, IPCO will not run the MySQL Cluster on VMware but on dedicated hardware in the production environment instead.
- Other issues which are beyond VMware.
  Part of the variance in performance may be caused by switches and routers that the VMWare networks are connected to. As an example, the Linux server used for routing between the MySQL Clusters is not running on hardware suited for real-time solutions. The MySQL Clusters are sharing the same SAN with more than 30 other virtual machines, this can for short write bursts cause I/O contention on the SAN.

How the indexes are sorted and hash index recalculation are mentioned in the MySQL 5.6 documentation.

In both test cases, the first batch with ten transactions with 1Gb/s bandwidth, uses longer time because it has to establish a connection to the database. The more transactions that are executed at at the same time, the less influence the first transaction has on the average time per transaction. The first batch of ten transactions will have a bigger influence on the average time per transaction on TC1 than on TC2. This is because TC1 only has 10% of the subscribers compared to the subscribers in TC2.

Low bandwidth does not have any impact on the data inserts within the same cluster for the Asynchronous Replication test environment, according to the paper “The Art of Data Replication” published by Oracle. Figure 6.3 and 6.4 shows the performance graphs from the Asynchronous Replication test environment where bandwidth is 57Kbit/s (the same graphs were presented in figure 5.27 and 5.34, section 5.3.2 Asynchronous Replication). These graphs show variation in time measured for every tenth allocation, just like the performance graphs in figure 6.1 and 6.2 as described above. These variations in performance are most likely one of the same reasons as listed above for the performance graphs where bandwidth is 1Gb/s in the Asynchronous Replication test environment.

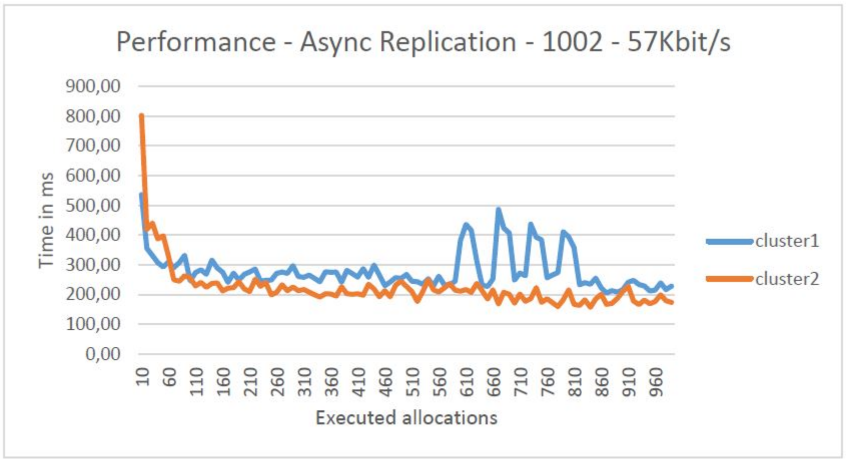


Figure 6.3: *The graph shows the performance of every tenth allocations for C1 and C2 when bandwidth is 57Kbit/s. X-axis shows how many allocations are performed, while Y-axis shows the time every tenth allocations uses in milliseconds.*

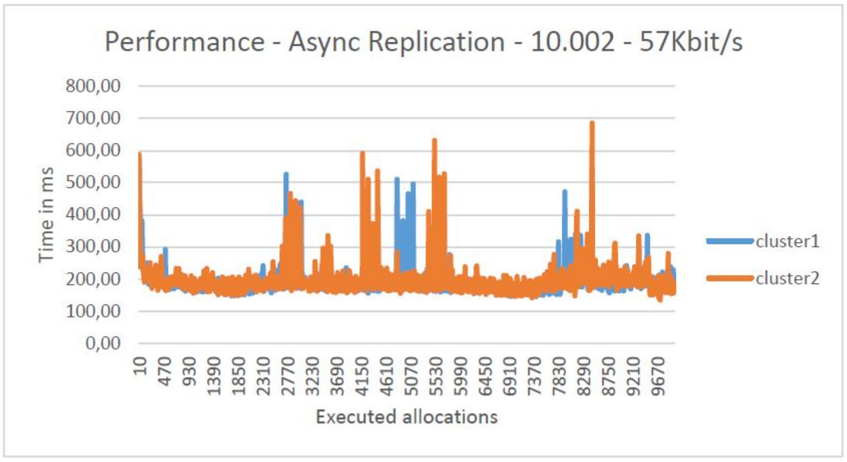


Figure 6.4: *The graph shows the performance of every tenth allocations for C1 and C2 when bandwidth is 57Kbit/s. X-axis shows how many allocations are performed, while Y-axis shows the time every tenth allocations uses in milliseconds.*

Big Cluster environment: Table 5.3 and 5.5 from section 5.3.1 Big Cluster shows the results from TC1 and TC2 where the bandwidth is 1Gb/s. These tables shows that the total time for all transactions and the average time per transaction increases when latency increases. This is expected because when latency increases the time it take for the eager replication to replicate between $C1_0$ and $C1_1$ increases as well.

The time it takes for the Big Cluster environment to execute each AllocateIMSI procedure can be described as following: $T_{transaction}$. Unlike the Asynchronous test environment, $T_{latency}$ has impact on $T_{transaction}$ because the replication takes place before the transaction commits.

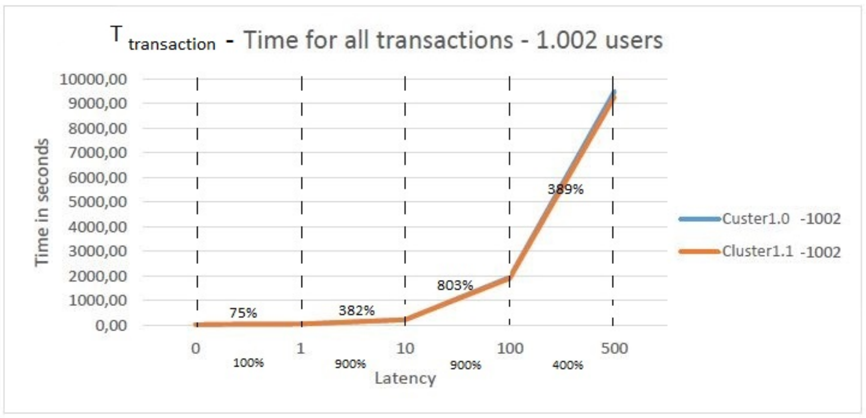


Figure 6.5: *This graph shows the time for all transactions that are measured in the Big Cluster environment. Y-axis shows the time in seconds while the X-axis shows the latency and the percentage increase in latency. The lines that shows $C1_0$ and $C1_1$'s time on the different latencies, also shows the percentage increase of time.*

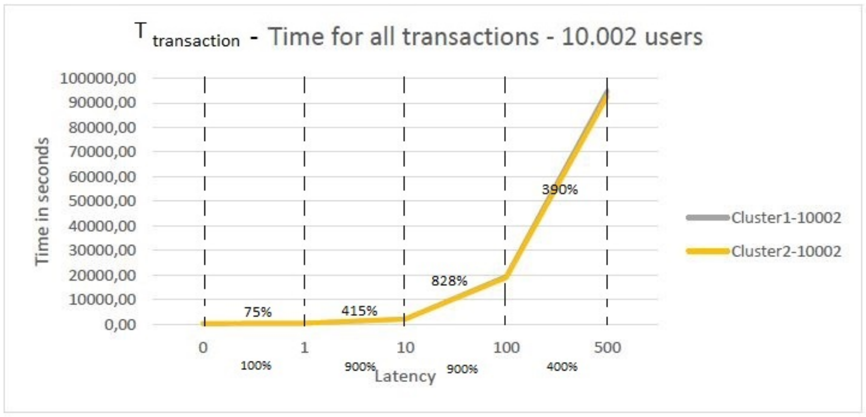


Figure 6.6: *This graph shows the time for all transactions that are measured in the Big Cluster environment. Y-axis shows the time in seconds while the X-axis shows the latency and the percentage increase in latency. The lines that shows $C1_0$ and $C1_1$'s time on the different latencies, also shows the percentage increase of time.*

The graphs showing $T_{transaction}$ in figure 6.5 and 6.6 show the same numbers as figure 5.4 and figure 5.13 from section 5.3.1 Big Cluster, but the percentage increment between latency 0-1ms, 1-10ms, 10-100ms and 100-500ms are added.
T_{local} is transaction time without latency, $T_{local} = T_{transaction} - T_{latency}$. To calculate T_{local} this formula is used: $T_{local} = T_{transaction}/T_{latency}$. Here $T_{transaction}$ represents the total time for all transactions and $T_{latency}$ represent the total latency. In this case the latency will never be 0ms, but closer to 1ms. This is because the routing goes through WanSIM. WanSIM is not designed to be a router, it does not have IP and routing in hardware but it is handled by the software. When the latency in these test cases are set to be 0ms and 1ms, the real measured latency are 1ms and

2ms because of the delay caused by WanSIM. When the latency is set to be 10ms, 100ms and 500ms the delay will not be noticed anymore.

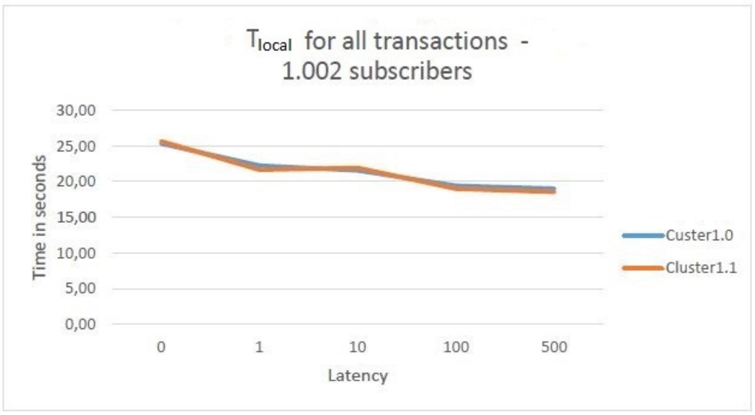


Figure 6.7: *The graph shows the total time of T_{local} from all 1.002 subscribers. X-axis shows the latency from 0ms-500ms, while Y-axis shoes the time in seconds from 0-30seconds.*

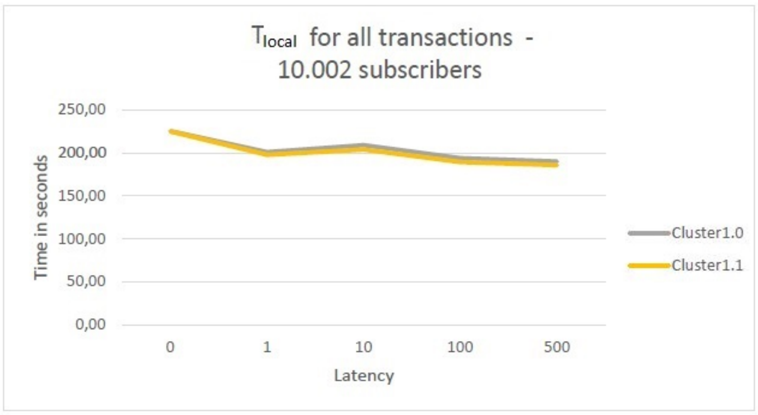


Figure 6.8: *The graph shows the total time of T_{local} from all 10.002 subscribers. X-axis shows the latency from 0ms-500ms, while Y-axis shoes the time in seconds from 0-250seconds.*

Figure 6.7 and figure 6.8 shows the T_{local} for all transactions executed in the Big Cluster test environment. One can see that the larger the latency is, the shorter time it takes for T_{local} to be inserted. This may be due to the two phase-commit (2PC) protocol used in the Big Cluster test environment. If there is no latency as shown in figure 6.9, the 2PC protocol sends a request to other sites in the cluster and if all sites agree, the changes are written to RAM and a commit is sent back. The transaction starts to build indexes if there are any, and writes the changes to disk before the job is completed. If there is latency as shown in figure 6.10, the 2PC protocol sends a request to all other sites in the cluster and because of the latency it will get time to build indexes if necessary and write the changes to disk before the commit is sent back. It is called "housekeeping". This is one of the reason why T_{local} time falls when the latency increases.
There are not enough measurements to be able to state whether the T_{local} will continue to decline with increased latency, if it will stagnate or if the environment will collapse.

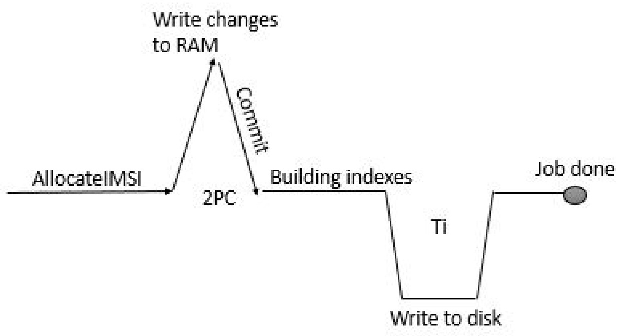


Figure 6.9: *Shows how an insert without latency works with the 2PC protocol used in the Big Cluster test environment.*

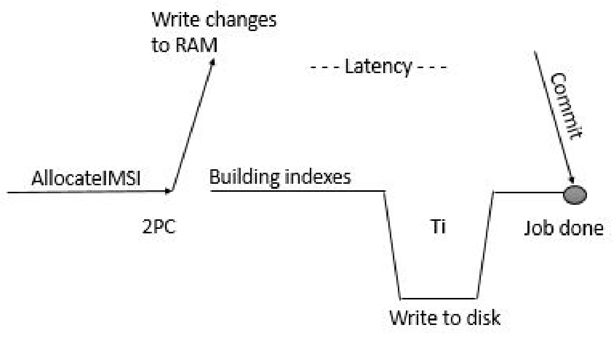


Figure 6.10: *Shows how an insert with latency works with the 2PC protocol used in the Big Cluster test environment.*

Figure 6.11 and 6.12 (the same graphs were presented in figure 5.6 and 5.15, section 5.3.1 Big Cluster) shows the results of the performance graphs.

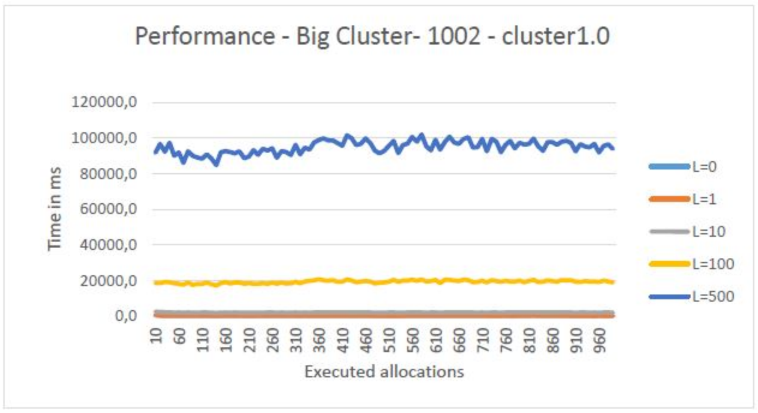


Figure 6.11: *TC1 - The graph shows the performance of every tenth allocations for $C1_0$ when the bandwidth is 1Gb/s. X-axis shows how many allocations are performed, while the Y-axis shows time in milliseconds.*

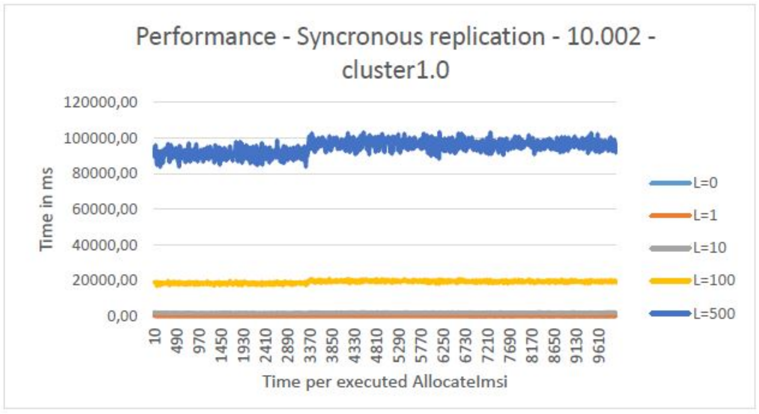


Figure 6.12: *TC2 - The graph shows the performance of every tenth allocations for $C1_0$ when the bandwidth is 1Gb/s. X-axis shows how many allocations are performed, while the Y-axis shows time in milliseconds.*

These graphs show that the higher latency becomes, the more the transaction time varies in time. This is also valid for TC1 and TC2, where bandwidth is 1Gb/s. When latency is 0ms, $T_{transaction}$ time varies with 400ms, from 200ms - 600ms. While when latency is 500ms the transaction varies between 80.000ms-100.000ms. High latency increases the risk of timeout in cluster applications, which can lead to large variations in performance.

Low bandwidth has a huge impact on the Big Cluster environment, when the bandwidth is 57Kbit/s as shown in table 5.8 and 5.4, section 5.3.1, Big Cluster. A bandwidth of 57Kbit/s corresponds to a network with 1Gb/s bandwidth and a latency that is more than 100ms, this is the same for TC1 and TC2 in the Big Cluster test environment. Figure 6.13 and 6.14 shows the performance graphs for the Big Cluster test environment where bandwidth is 57Kbit/s (the same graphs were presented in figure 5.12 and 5.21, section 5.3.1 Big Cluster). These graphs show that the time varies between 30000ms to 40000ms for every tenth executed transactions. This is due to the low bandwidth, when packets are sent over a slow network they risk being dropped by the router if the package TTL (time to live) has expired. The package has to be retransmitted. The packages TTL varies from package to package. This leads to time of transactions corresponding to a latency average over 100ms.

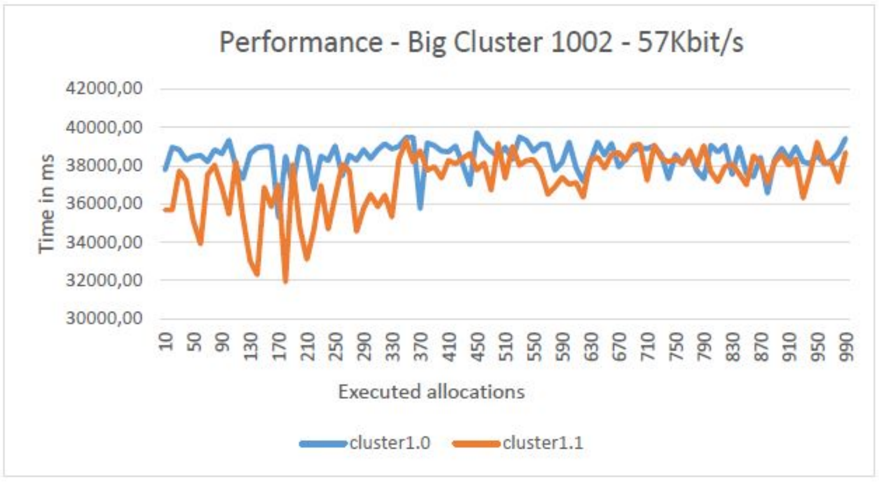


Figure 6.13: *The graph shows the performance of every tenth allocations for $C1_0$ and $C1_1$ when the bandwidth is 57Kbit/s. X-axis shows how many allocations are performed, while Y-axis shows the time every tenth allocations uses in milliseconds.*

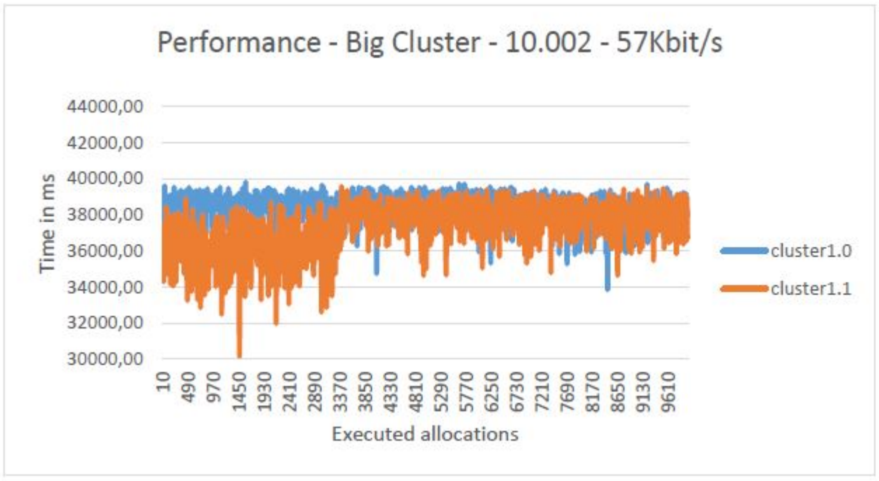


Figure 6.14: *The graph shows the performance of every tenth allocations for $C1_0$ and $C1_1$ when the bandwidth is 57Kbit/s. X-axis shows how many allocations are performed, while Y-axis shows the time every tenth allocations uses in milliseconds.*

1B: In what way does bandwidth and latency affect the replication delay on the Asynchronous Replication topology.

Figure 6.15 and figure 6.16 shows two of the replication delay graphs, one from TC1 and one from TC2 where the bandwidth is 1Gb/s (the same graphs were presented in figure 5.25 and 5.32, in section 5.3.2 Asynchronous Replication).

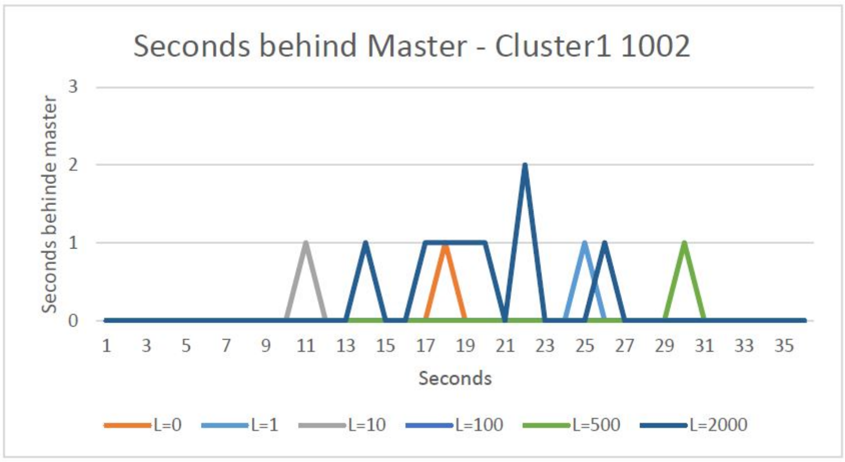


Figure 6.15: *The graph show how many seconds C1 is behind the Master (C2) when the bandwidth is 1Gb/s. X-axis shows seconds from 0-3, while Y-axis shows seconds for the transaction.*

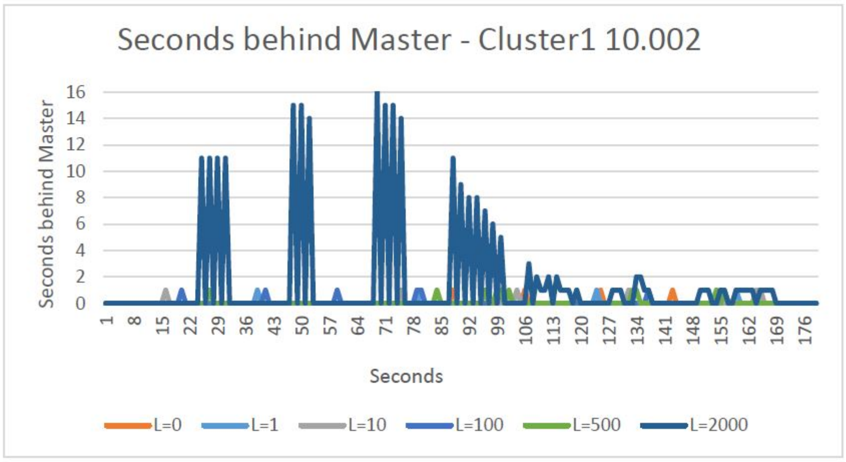


Figure 6.16: *The graph show how many seconds C1 is behind the Master (C2) when the bandwidth is 1Gb/s. X-axis shows seconds from 0-16, while Y-axis shows seconds for the transaction.*

The graphs show that the latency does not affect the lacy replication very much. To see how the seconds behind master was affected when the latency was high, the same test was executed with a latency that was 2000ms. It can be discussed whether the values shown in the graphs are reliable or not. According to MySQL the `Seconds_Behind_Master` parameter often shows a value of 0, even if the I/O thread is late compared to the master. This can occur when the network is slow. This also applies to TC2 where the amount of subscribers are 10.002. The seconds behind master parameter was checked by comparing that the tables in C1 and C2 contained the same amount of data, to see if it was reliable.

Figure 6.17 and 6.18 shows the second behind master when bandwidth is 57Kbit/s (the same graphs were presented in figure 5.28 and 5.35, section 5.3.2 Asynchronous Replication). It was observed that the seconds behind master parameter is unstable and the slave is far behind the master when all transactions were executed. This was observed by comparing the database tables in C1 and C2 to check that they contained the same data. When the bandwidth is low, the replication between the clusters is slow and is not prioritized by the clusters.

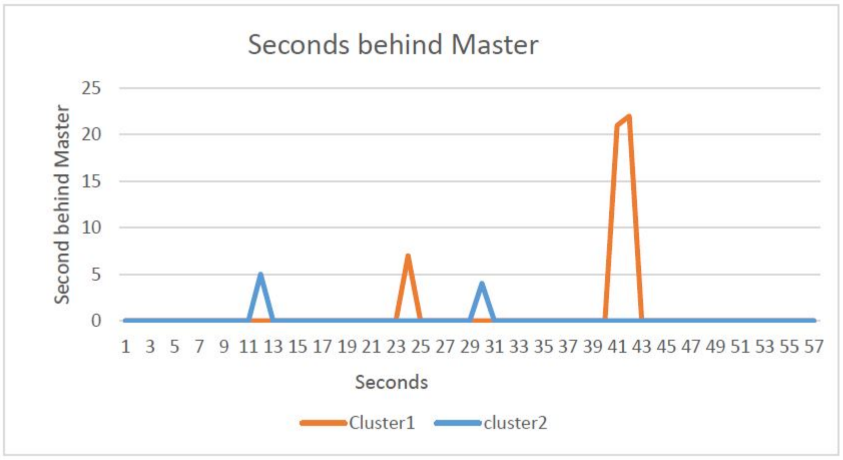


Figure 6.17: *The graph shows the seconds behind master when the bandwidth is 57Kbit/s. The orange line shows how many seconds C1 is behind the Master (C2) and the blue line shows how many seconds C2 is behind the Master (C1) X-axis shows seconds from 0-25, while Y-axis shows seconds for the transaction.*

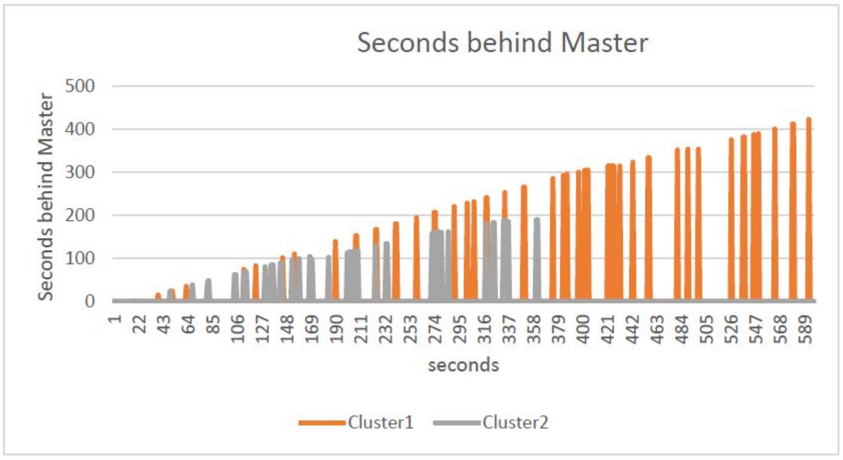


Figure 6.18: *The graph shows the seconds behind master when the bandwidth is 57Kbit/s. The orange line shows how many seconds C1 is behind the Master (C2) and the gray line shows how many seconds C2 is behind the Master (C1) X-axis shows seconds from 0-500, while Y-axis shows seconds for the transaction.*

1C: Which combinations of bandwidth and latency can cause error situations like package lose, cluster downtime or replication downtime.

During the tests, no error situations with package lose or replication downtime were observed on the two test environments. The results shown in the tables 5.3, 5.4, 5.5 and 5.6 from section 5.3.1 Big Cluster, shows that when latency increases or bandwidth is low the performance time is high. This is not formally a technical error but it is an undesirable situation which later can lead to errors in the system. It can also be discussed if the performance result measured when bandwidth is 57Kbit/s and when latency for example is 10ms or higher, is beyond what can be considered as acceptable performance.

It can also be discussed if the graphs seconds behind master in figure 6.17 and 6.18 as shown above, from the Asynchronous Replication environment when bandwidth is 57Kbit/s, are considered acceptable. If the `Seconds_Behind_Master` parameter is considered as correct, the seconds behind master is at the master over 400 seconds behind the master, this can be looked upon as a possible error.

6.2 Recommendation to IPCO

This part of the discussion will in particular be focused towards IPCO's production environment. The test results and the database technical discussion will be assessed to give a recommendation about which of the test environments that are best suited for IPCO.

1A: In what way does bandwidth and latency affect the performance on the two cluster topologies.

The Big Cluster environment is stable, but performance and latency has a major impact on the performance. If the performance is bad, as shown in some of the test results in table 5.3, 5.5, 5.4 and 5.6 in section 5.3.1 Big Cluster, IPCO risks that the system is unable to allocate enough IMSIs. If the performance is bad, the service that IPCO provides will not work as expected. When a subscriber arrives in a country which is an IPCO partner, the "roam as a local" service will not work immediately. If 10.002 Norwegian Ventelo subscribers arrives in a country which is an IPCO partner and they all turn on their cellphones at the same time, in worst case and if the latency is high it could take more than 25 hours before the IPCO service is working for all the subscribers. In general, other tasks involving modifications to the IPCO subscribers will be affected by a similar delay as well:

- Insert new subscriptions
- Delete subscriptions
- If applicable, modify subscriptions

If this environment is going to be used by IPCO, the environment has to have high bandwidth and zero latency. This can turn out to be difficult since IPCO's production environment is going to be spread over different countries, which will lead to latency between the sites. Oracle support has stated "We have our best people working on increasing the speed of light, but have not succeeded yet". This was a joke from Oracle, but it illustrates the absolute physical limitation for eager replication.

The Asynchronous Replication environment performance measurements have shown that varying the network latency and bandwidth has little impact on the MySQL cluster transaction performance. This is because the latency and bandwidth is between the clusters and the replication starts after the transaction is committed. This is an important result to consider when IPCO plans their database architecture.

1B: In what way does bandwidth and latency affect the replication delay on the Asynchronous Replication topology.

Assuming the `Seconds_Behind_Master` parameter can be trusted and if the bandwidth is high and the latency is between 0ms-500ms, it will not have a big affect on the replication delay for the Asynchronous Replication environment assuming. But if the bandwidth is as low as shown in figure 6.17 and 6.18 above, the seconds behind master parameter is unstable and the replication time is long. Long replication delay time can cause trouble if one of the clusters goes down. If C1 goes down, C2 has to take over and C2 will contain much data which is not updated due to the big

replication delay. IPCO can also, in worst case risk that a subscriber has time to get to another site before the data is updated.
According to IPCO, networks which do not have the bandwidth necessary to replicate data between master and slave without the slave falling behind master, is defined as slow.

1C: Which combinations of bandwidth and latency can cause error situations like package lose, cluster downtime or replication downtime.

Both the Big Cluster and the Asynchronous Replication test environments are stable and had no downtime during the tests. However, from IPCO's point of view it can be discussed if some of the test results can be looked upon as errors.
As mentioned earlier, bad performance can lead to that subscribers, in worst case have to wait over 25 hours to be able to use the IPCO service. In today's society people expect it to work immediately when the cellphone is turned on. Good performance is one of the absolute requirements IPCO has to the "roam as a local" service, since it is a real time service. Bad performance can therefor be seen as an error.
For the Asynchronous Replication test environment low bandwidth or high replication delay can also be seen as an error since this can cause problems for IPCO if one of the clusters fail or if the subscribers have time to go to another site before the data arrives.

In this chapter the results from tests against database and network theory have been discussed. In the last part of this chapter, the results and theory from IPCO's point of view have been discussed. The next chapter, Conclusion and Future work will be based on the main research question and this discussion chapter.

Chapter 7

Conclusion and Future work

This chapter presents the conclusion of this thesis and future work. The conclusion is presented in two ways, database replication in general and a recommendation to IPCO. Future work presents what could have been done differently in this project and show new research areas that can benefit database replication in general and also benefit IPCO.

7.1 Conclusion

The work of this thesis has investigated how latency and bandwidth affects the performance, replication delay and error rate on two different MySQL Cluster environments with replication within and between clusters. These are tested for a telecom provider named IPCO.
There are several replication techniques on the market today. There are two ways to classify replication techniques according the authors of the literature. The first technique is named two parameter classification and is based on two parameters: transaction location (primary copy, update anywhere) and synchronization strategy (eager replication, lazy replication). These combinations of parameters give four possible replication strategies. The second classification is named three parameter classification and is based on the parameters: server architecture (primary copy, update anywhere), server interaction (constant interaction, linear interaction) and transaction termination (voting termination, Non-voting termination). These combinations of parameters give sixteen possible replication strategies. MySQL Cluster 7.3 as used in this thesis, does not support all combinations of replication techniques that these two classifications are offering. It does not support eager replication between sites or between clusters, only within the cluster.
The two test environments have been created and tested with different test cases. The first environment is named Big Cluster and the second environment is named Asynchronous Replication. The Big Cluster test environment consists of a large cluster that uses eager replication within the cluster with update date anywhere. The Asynchronous Replication test environment consists two clusters with eager replication within the clusters with update anywhere and lazy replication between the clusters.

Given the two cluster topologies in what way are the characteristics different?

The conclusion of the main research question is divided into two parts. First the databases technical findings will be concluded before IPCO's recommendation is presented.

Database technical The investigation performed for database replication in general through this thesis, shows that the Big Cluster test environment is stable, but high latency has a big influence on the environment when it comes to performance. The performance of the Asynchronous Replication test environment is not so much affected by high latency and low bandwidth. The time lag in the Asynchronous Replication test environment enables data to be replicated over a lower-bandwidth network, but it does not provide the same level of protection as the Big Cluster environment. The Big Cluster environment uses the 2PC protocol that ensures that the data is spread to all the four data nodes. The Asynchronous Replication test environment uses the protocol 2PC within the clusters, the data is consistent between the two data nodes in the cluster but there is no guarantee that C1 and C2 contains the same data.

IPCO Through this thesis, the investigation performed on behalf of IPCO shows that the Big Cluster test environment is affected by low bandwidth and high latency and will therefor be too slow compared to IPCO's requirements to a real time system. According to the theory, the Asynchronous Replication test environment is best suited for replication over longer distances. It turned out to be correct in this case too.
The author of this thesis has come up with a recommendation to IPCO. It is based on the results from the tests that were executed on both test environments and on a theoretical background. As the situation is today, the ideal production environment for IPCO is Asynchronous Replication environment. To get the environment to work in the best possible way, the nodes making up a MySQL Cluster should be connected with the smallest latency possible. This requires that the nodes in the cluster are at the same geographical site and on the same LAN. This will ensure enough bandwidth as well. The Asynchronous Replication environment works fine, even if the latency is high, but it is depending on enough bandwidth. It is possible to buy or rent network connections with high bandwidth where the latency is within reasonable limits.
The measurements made on the Asynchronous Replication test environment shows that it is possible to guarantee that data older than 2 seconds are consistent. This is done by giving the clusters in each geographical area ownership to the data modifications initiated by roaming events, originating from their geographical area. For example, the cluster in France handles events from mobile networks in France and the cluster in Norway handles events from mobile networks in Norway. This ensures that the data integrity is not compromised by replication delay. The cluster in Norway will never modify data belonging to the cluster in France, and vice versa.

7.2 Future work

This part of the chapter looks at what could have been done differently if the project was to be repeated and what would be a natural next step for this project, both in relation to database replication in general and towards IPCO.

If this project was to be repeated and would have been given unlimited resources, it would be desirable to run test the environments on dedicated hardware and dedicated network devices. In this thesis the two test environments are simulated by using VMware and using WanSIM as router between the networks. Dedicated hardware and network devices would have benefits like:

- Reduced uncertainty in the measurements by elimination CPU contention between MySQL and operation system process (Delegate CPU threads to MySQL Cluster processes).
- Better functions for traffic shaping, like latency, jitter and bandwidth.

Below are some proposals which would be interesting to look at as follow-up steps in this thesis. These proposal are written for IPCO, but will also apply for database replication in general.

There are several areas which could be interesting for IPCO to evaluate. One area which would

benefit IPCO in the future is to see how multiple threads (when there are two or more threads) running against each cluster affects the performance and error rate on both test environment, and the replication delay on the clusters for the Asynchronous Replication test environment.
Another test which could be of interest for IPCO would be to measure how much data that actually comes over the network when the bandwidth is 1Gbit/s. If the exact amount of data that comes over the network is known, it can for example be used to calculate the bandwidth IPCO actually needs. The bandwidth can be measured by enabeling Simple Network Management Protocol (SNMP) with network interface instrumentation and use a statistical data traffic measurement tool like Multi Router Traffic Grapher (MRTG) to measure the bandwidth.

Measuring jitter can be useful and interesting. Jitter shows the variation in time between packets arriving caused by network congestion, timing drift or route changes[1]. IPCO uses voice over IP (VoIP), this is exposed for network behavior referred to as delay and jitter which can degrade the voice application to the point of being unacceptable to the average user.

[1]http://searchunifiedcommunications.techtarget.com/definition/jitter

Bibliography

[1] Faller Torbjørn, 2012 *Visitor Serving System Services for Mobile Operators - The Solution*

[2] Wikipedia, 2013 *Telecommunication. URL:*

[3] T. Connolly, C. Begg, 2010 *Database systems - A Practical Approach to Design, Implement, and Management*

[4] Robert D.Schneider, 2010 *Replication For Dummies - Sybase Edition*

[5] B. Kemme, R. Jimènez-Peris, M. Patino-Martinez and G. Alonso, 2010 *Database Replication: A Tutorial, pp. 219-250. Springer, Replication (2010)*

[6] R.Van Renesse and R. Guerraoui, 2010 *Replication Techniques for Availability, pp. 19-39. Springer, Replication (2010)*

[7] F. Pedone. A. Schiper, 2010 *From Object Replication to Database Replication, pp. 207-218. Springer, Replication (2010)*

[8] M. Lyle, 2011 *Redundancy in Data Storage - Part 2: Geographical Replication*

[9] M. Wiesmann, 2002 *GROUP COMMUNICATIONS AND DATABASE REPLICATION: TECHNIQUES, ISSUES AND PERFORMANCE*

[10] J. Gray, P. Helland, P. O'Neil, D. Shasha, 1996 *The Dangers of Replication and a Solution*

[11] M. Wiesmann, F. Pedone, A. Schiper, B. Kemme, G. Alonso, 2000 *Database Replication Techniques: a Three Parameter Classification*

[12] Oracle, 2013 *MySQL 5.7 Referemce Manual*

[13] Oracle, 2013 *MySQL 5.6 Referemce Manual*

[14] Oracle, 2013 *MySQL 5.5 Referemce Manual*

[15] B. Schwartz, P. Zaitsev, V. Tkachenko, J. D. Zawodny, A. Lantz, D. J. Balling, 2012 *High Performance MySQL, Second Edition*

[16] M. Tamer Özsu, Patrick Valduriez, 2011 *Principles of Distributed Database Systems*

[17] M. Wiesmann, F. Pedone, A. Schiper, B. Kemme, G. Alonso, 2000 *Understanding Replication in Database and Distributed Systems*

[18] P. Koszalinska, K. I. Komputerowej 2007 *Model and classification of database replication techniques*

[19] Sybase, 2003 *Replication Strategies: Data Migration, Distribution and Sunchronization*

[20] P. Baran, 1962 *On distributed communications network*

[21] R. Bradford, C. Schneider, 2012 *Effective MySQL: Replication Techniques in Depth*

[22] E. R. Harold, 2004 *Java Network Programming*

[23] O. Bonaventure, 2011 *Computer Networking: Principles, protocoles and Parctice*

[24] P. Dubois, 2013 *MySQL - Developer's Library*

[25] T. Nam-Luc, 2009 *MySQL Cluster Design and Architecture Principles*

[26] Oracle, 2014 *MySQL Cluster 7.3 Release Notes*

[27] A. Morgan, 2013 *MySQL Cluster 7.3 is now Generally Available - an overview*

[28] J. Lechtenbörger, Uvinerity of Münster, Germany *2-Phase Commit Protocol*

[29] L. Frenzel, 2013 *Whats The Difference Between The OSI Seven-Layer Network Model And TCP/IP*

[30] A. Muthitacharoen, B. Chen and D. Mazieres *A Low-bandwidth Network File System*

[31] E. Cecchet, G. Candea, A. Ailamaki, 2008 *Middelware-based Database Replication: The Gaps Betweem Theory and Practice*

[32] Sun Oracle, 2011 *The Art of Data Replication*

List of Figures

Appendix A

Configuration of the network

Configuration of the network

The network 10.20.1.0/24 is divided into three sub networks to simulate that the environments are spread over different networks. The three sub networks are:

- 10.20.1.128/5
- 10.20.1.160/5
- 10.20.1.161/8

The two first networks are divided into networks with 30 usable hosts while the third network has 126 usable hosts. Table A.1 shows the three sub networks and which host that belongs to which network.

Setting up network on the hosts When a virtual machine is created on VMware it is possible to add several networks adapters per machine, the default is one. WanSIM and MySQL-runner needs three networks adapters each, because they both have access to all the three networks segments 10.20.1.128/5, 10.20.1.160/5 and 10.20.1.0/8.
During the installation of the OS the "networks configurations" are selected and the networks are configured.
It is possible to add networks adapters and do network configurations after the machines are installed as well. The rest of the hosts only need one network adapter each, the networks in these hosts are also configured during the installation of the OS.

Server Name	Filesystem Size	RAM	IP-Address	IP-Address	IP-Address	Note:	OS
	(GB)	Subnet alias	10.20.1.128/5	10.20.1.160/5	10.20.1.0/8	root - password	
			255.255.255.224	255.255.255.224	255.255.255.128		
mysql01	24	2	10.20.1.130			Master	Oracle 6
mysql02	24	2	10.20.1.131			Master	Oracle 6
ndb01	26	6	10.20.1.132			Master	Oracle 6
ndb02	26	6	10.20.1.133			Master	Oracle 6
mysql03	24	2		10.20.1.162		Master	Oracle 6
mysql04	24	2		10.20.1.163		Master	Oracle 6
ndb03	26	6		10.20.1.164		Master	Oracle 6
ndb04	26	6		10.20.1.165		Master	Oracle 6
WanSIM	20	2	10.20.1.129	10.20.1.161	10.20.1.68	Master	Debian
MySQL-runner	20	2	10.20.1.134	10.20.1.166	10.20.1.69	Master	Oracle 6

Table A.1: This table gives an overview of the three sub networks that comes from the network 10.20.1.0/24 and the hosts in the networks.

Appendix B

Setting up WanSIM

To setup routing on WanSIM, IP forwarding has to be enabled. It is important that the network segments are configured correctly, otherwise the routing will not work. This can cause that packages will be sent on the wrong interfaces causing them to be dropped. To enable IP forwarding on WanSIM, type the command:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

To disable the routing switch from 1 to 0. Only the root users can type these commands.

Appendix C

Test environment 1 - Asynchronous replication

Configuration of MySQL Cluster Only the configuration of cluster 1 will be described in detail, because there are only minor changes that separates the two clusters. The changes will be described below:

Download and install MySQL Cluster

1. The directory /root/MySQL7.3 was created on all hosts

   ```
   mkdir /root/MySQL7.3
   ```

2. Download the MySQL7.3.3-1 RPM-packages on all four hosts in the cluster, place it under /root/MySQL7.3/
3. Install MySQL7.3.3-1 RPM-packages on all four hosts in the cluster. Command for installing the RPM packages on one host:

   ```
   rpm -ivh MySQL-Cluster*
   ```

   Note: make sure to delete earlier versions of MySQL installations if there exists any, before installing MySQL7.3.3-1 RPM-package.

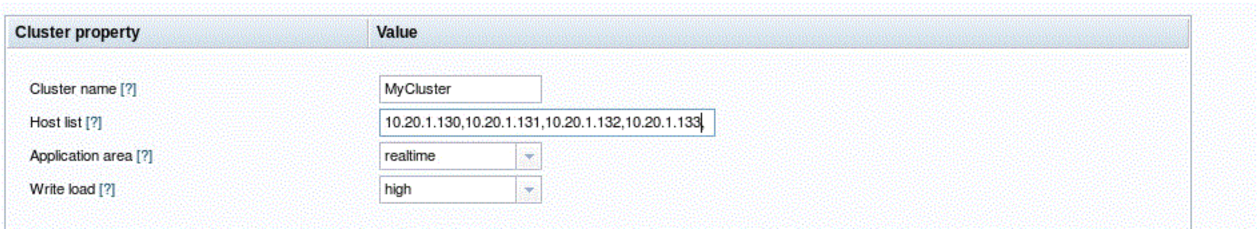
Auto-installer

1. Download Python 2.6, Paramiko 1.7.7.1 and Pycrypto 2.6 on ndb01, it is necessary to get the auto-installer to work.
2. The auto-installer opens automatically in the browser by typing:

   ```
   ndb_setup.py
   ```

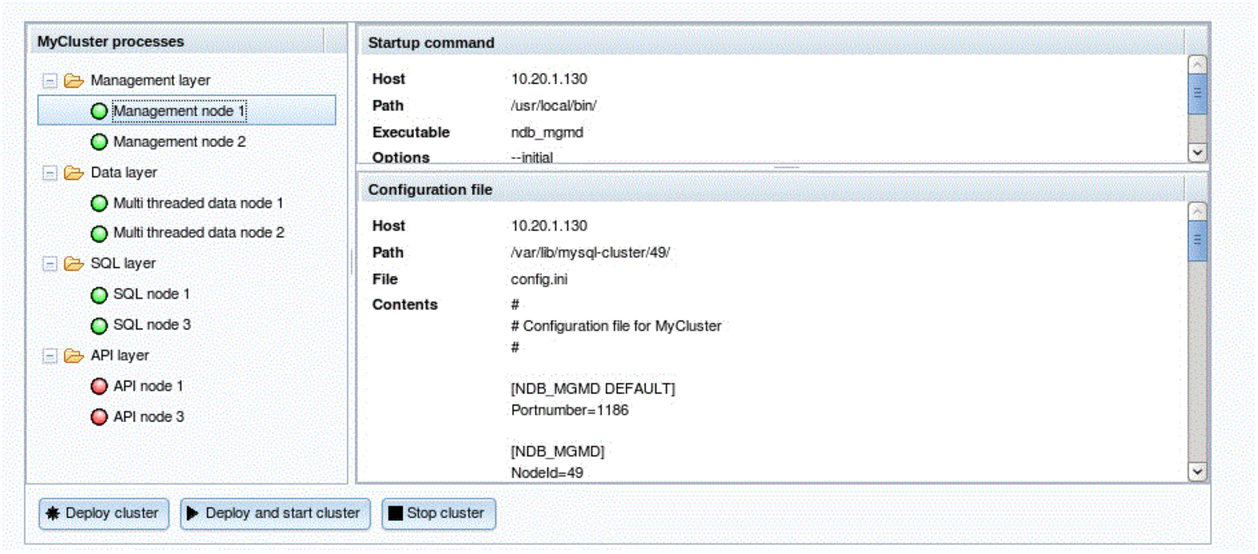
3. Select “create new MySQL Cluster” in the wizard.
4. Define the cluster: see figure C.1.
   (a) Specify the host list of servers by typing in all hosts (mysql01, mysql02, ndb01 and ndb02) that are in the cluster by IP-address.

   ```
   10.20.1.130, 10.20.1.131, 10.20.1.132, 10.20.1.133
   ```



Cluster property	Value
Cluster name [?]	MyCluster
Host list [?]	10.20.1.130,10.20.1.131,10.20.1.132,10.20.1.133
Application area [?]	realtime
Write load [?]	high

Figure C.1: Auto-installer wizard for setting up C1, specify hosts and application type.



Startup command

Host	10.20.1.130
Path	/usr/local/bin/
Executable	ndb_mgmd
Options	--initial

Configuration file

Host	10.20.1.130
Path	/var/lib/mysql-cluster/49/
File	config.ini
Contents	

```
#
# Configuration file for MyCluster
#

[NDB_MGMD DEFAULT]
Portnumber=1186

[NDB_MGMD]
NodeId=49
```

Figure C.2: Auto-installer for C1 views the configuration files and startup commands

Note: for configuration of cluster 2 type the IP addresses 10.20.1.162, 10.20.1.163, 10.20.1.164, 10.20.1.165

(b) Application area = real time

(c) Write load = high

5. Define hosts: this should be the path to where the MySQL Cluster installation is and where the data (and configuration files) should be stored.

   (a) MySQL Cluster install directory: /usr/local/bin/

   (b) MySQL Cluster data directory: /var/lib/mysql-cluster/

6. At the end of the wizard all the nodes and their configuration files shows up, see figure C.2. The configuration files were copied down to the host they belonged to.

Configuration files and startup command files on mysql01 and mysql02

1. Place the my.cnf file for mysql01 in the /etc directory.
2. Place the my.cnf file for mysql02 in the /etc directory.
   Note: for configuration of C2 the files are located on the same directory.
3. Create the directories listed under with mysql privileges on mysql01

- /var/lib/mysql-cluster/
  ```
  mkdir /var/lib/mysql-cluster/
  ```
- /var/lib/mysql-cluster/65/
  ```
  mkdir /var/lib/mysql-cluster/65/
  ```
- /var/lib/mysql-cluster/65/tmp
  ```
  mkdir /var/lib/mysql-cluster/65/tmp
  ```
- Give the directories mysql privileges, from the directory /var/lib/
  ```
  chown -R mysql:mysql mysql-cluster
  ```

4. Create the corresponding catalogs on mysql02. Instead of using SQL node id = 65 that belongs to mysql01, the mysql02 SQL node id that is 67 is used. Give the directories mysql privileges.
   - /var/lib/mysql-cluster/
   - /var/lib/mysql-cluster/67/
   - /var/lib/mysql-cluster/67/tmp
5. The config.ini configuration files were added to the directory /var/lib/mysql-cluster/ on both the two SQL nodes. The config.ini files are identical.
6. The last file start_ndbmgm.sh is placed on /root/MySQL7.3/ on both nodes. This file starts the management nods mysql01 and mysql02.

Startup command files on ndbl01 and ndb02

1. The start up command file was placed under the directory /root/MySQL7.3/ on both nodes. This file starts the management nods on ndb01 and ndb02.
2. Create the directory /var/lib/mysql-cluster/ on both nodes with mysql privileges.
   ```
   mkdir /var/lib/mysql-cluster/

   chown -R mysql:mysql mysql-cluster
   ```

Starting the environment

1. Start the NDB manager on mysql01
   ```
   ./start_ndbdmgm1.sh
   ```
2. Start the NDB manager on mysql02
   ```
   ./start_ndbdmgm1.sh
   ```
3. Start the NDBD on ndb01
   ```
   ./ndb-start.sh
   ```

4. Start the NDBD on ndb02

   ```
   ./ndb-start.sh
   ```

5. Start the mysql on mysql01

   (a) Create the SQL files - this step is only necessary the first time the environment is started.

      i. Run as root user

         ```
         mysql_install_db --no-defaults --datadir=/var/lib/mysql-cluster/65
         --basedir=/usr
         ```

      ii. Change the privileges to all directories under /var/lib/mysql-cluster/65

         ```
         chown mysql:mysql -R 65
         ```

   (b) Start mysql

      i. Only mysql users can start the mysql

         ```
         su mysql
         ```

      ii. To start the mysql

         ```
         nohup mysqld --defaults-file=/etc/my.cnf &
         ```

6. Start the mysql on mysql02

   (a) Create the SQL files - this step is only necessary the first time the environment is started.

      i. Run as root user

         ```
         mysql_install_db --no-defaults --datadir=/var/lib/mysql-cluster/67
         --basedir=/usr
         ```

      ii. Change the privileges to all directories under /var/lib/mysql-cluster/67

         ```
         chown mysql:mysql -R 67
         ```

   (b) Start mysql

      i. Only mysql users can start the mysql

         ```
         su mysql
         ```

      ii. To start the mysql

         ```
         nohup mysqld --defaults-file=/etc/my.cnf &
         ```

         Note: Can use the service mysql start, but will only start one mysql, not a new instance.

Note: mysql03 and mysql04 are started in the same way, but the SQL node id are not 65 and 67, but 55 and 57.

Setting up replication between two clusters sites

1. To setup a replication channel between the clusters there are some changes which needs to be done in the my.cnf configuration files on all the SQL nodes (mysql01, mysql02, mysql03 and mysql04)

   (a) Each SQL node used for replication must be uniquely identified by a server id. This server id is a unique integer.

      i. set server id on mysql01

      ```
      server_id=11
      ```

      ii. set server id on mysql02

      ```
      server_id=12
      ```

      iii. set server id on mysql03

      ```
      server_id=13
      ```

      iv. set server id on mysql04

      ```
      server_id=14
      ```

   (b) Binary log, log-bin, needs to be enabled on each cluster site that will act as a replication master.

      i. Enable binary logging on mysql01, mysql02, mysql03 and mysql04

      ```
      bin-log
      ```

   (c) Skip slave start, this option is not necessary but it will prevent the replication to start immediately. The replication will not start before the START SLAVE statement has been issued.

      i. Enable skip slave status on mysql01, mysql02, mysql03 and mysql04

      ```
      skip-slave-start
      ```

Restart the mysqld to start using the changes in my.cnf

1. Stop mysql on mysql01, mysql03 by using the command on both SQL nodes.

   ```
   service mysql stop
   ```

2. Start mysql on mysql01, mysql03 by using the command on both SQL nodes.

   ```
   nohup mysqld --defaults -file=/etc/my.cnf &
   ```

Create slave accounts on the masters for the primary replication channel

1. Create a slave account on the masters mysql01 and mysql03, with the appropriate privileges. In this case the mysql01 server is acting as the master in C1, and mysql03 in C2.

   (a) Create a slave account in the MySQL client on mysql01

   ```
   mysql> GRANT REPLICATION SLAVE ON *.* TO 'repl'@'10.20.1.162'
   IDENTIFIED BY 'MASTER';
   ```

(b) Create a slave account in the MySQL client on mysql03

```
Mysql> GRANT REPLICATION SLAVE ON *.* TO 'repl'@'10.20.1.130'
IDENTIFIED BY 'MASTER';
```

- In the previous statement 'repl' is the slave account user name, '10.20.1.xx' is the IP address of the replication slave, and 'MASTER' is the password to assign the account.
  Note: for the secondary replication channel create a slave account for mysql04 on the master mysql02 and opposite.

2. Configure the slave to use the master. When mysql01 is the slave, it is going to use mysql03 as the master, and the other way around.

   (a) Configure the slave mysql03 to use the master mysql01 - on mysql03 in the MySQL client.

   ```
   mysql>
   CHANGE MASTER TO
   MASTER_HOST='10.20.1.130',
   MASTER_PORT=3306,
   MASTER_USER='repl',
   MASTER_PASSWORD='MASTER';
   ```

   (b) Configure the slave mysql01 to use the master mysql03 - on mysql01 in the MySQL client.

   ```
   mysql>
   CHANGE MASTER TO
   MASTER_HOST='10.20.1.162',
   MASTER_PORT=3306,
   MASTER_USER='repl',
   MASTER_PASSWORD='MASTER';
   ```

3. Start the slaves in the MySQL client on both mysql01 and mysql03.

   ```
   START SLAVE
   ```

Note: for the secondary replication channel section No. 2 and 3 can be done when the primary replication channel is down.

Appendix D

Test environment 2 - Big Cluster

Configuration of MySQL Cluster The MySQL 7.3 RPM package is already downloaded and installed on all hosts, since test environment 1 - Asynchronous replication also use this nodes. Make sure that test environment 1 is not running.

Auto-installer

1. Ndb01 has already installed Python 2.6, Paramiko 1.7.7.1 and Pycrypto 2.6 from the installation of test environment 1, it is necessary to get the auto-installer to work.
2. Type the command below in the command lint to get the auto-installer to open:
   ```
   ndb_setup.py
   ```
3. Select "create new MySQL Cluster" in the wizard.
4. Define the cluster:
   (a) Specify the host list of servers by typing in all hosts (mysql01, mysql02, ndb01, ndb02, mysql03, mysql04, ndb03 and ndb04) that are in the cluster by IP-address.
   ```
   10.20.1.130, 10.20.1.131, 10.20.1.132, 10.20.1.133,
   10.20.1.162, 10.20.1.163, 10.20.1.164, 10.20.1.165
   ```
   (b) Application area = real time
   (c) Write load = high
5. Define hosts: this should be the path to where the MySQL Cluster installation is and where the data (and configuration files) should be stored.
   (a) MySQL Cluster install directory: /usr/local/bin/
   (b) MySQL Cluster data directory: /var/lib/mysql-clusterSync/
6. At the end of the wizard all the nodes and their configuration files shows up. The configuration files were copied down to the host they belonged to.

Configuration files and start up command files on mysql01, mysql02, mysql03 and mysql04

1. Place the my.cnf file for mysql01 in the /etc directory.
2. Place the my.cnf file for mysql02 in the /etc directory.
3. Place the my.cnf file for mysql03 in the /etc directory.
4. Place the my.cnf file for mysql04 in the /etc directory.
5. Create the directories listed under with mysql privileges on mysql01
   - /var/lib/mysql-clusterSync/
     ```
     mkdir /var/lib/mysql-clusterSync/
     ```
   - /var/lib/mysql-clusterSync/65/
     ```
     mkdir /var/lib/mysql-clusterSync/65/
     ```
   - /var/lib/mysql-clusterSync/65/tmp
     ```
     mkdir /var/lib/mysql-clusterSync/65/tmp
     ```
   - Give the directories mysql privileges, from the directory /var/lib/
     ```
     chown -R mysql:mysql mysql-clusterSync
     ```
6. Create the corresponding catalogs on mysql02. Give the directories mysql privileges.
   - /var/lib/mysql-clusterSync/
   - /var/lib/mysql-clusterSync/67/
   - /var/lib/mysql-clusterSync/67/tmp
7. Create the corresponding catalogs on mysql03. Give the directories mysql privileges.
   - /var/lib/mysql-clusterSync/
   - /var/lib/mysql-clusterSync/55/
   - /var/lib/mysql-clusterSync/55/tmp
8. Create the corresponding catalogs on mysql04. Give the directories mysql privileges.
   - /var/lib/mysql-clusterSync/
   - /var/lib/mysql-clusterSync/57/
   - /var/lib/mysql-clusterSync/57/tmp
9. The config.ini configuration file was added to the directory /var/lib/mysql-clusterSync/ on all four SQL nodes. The config.ini files are identical.

10. The last file start_ndbmgm.sh is placed on /root/MySQL7.3/ on all SQL nodes. This file starts the management nods mysql01, mysql02, mysql03 and mysql04.

Start up command files on ndbl01, ndb02, ndbl03 and ndb04

1. The start up command file was placed under the directory /root/MySQL7.3/ on all the data nodes. This file starts the management nods on ndb01, ndb02, ndb03 and ndb04.
2. Create the directory /var/lib/mysql-clusterSync/ on all data nodes with mysql privileges.
   ```
   mkdir /var/lib/mysql-clusterSync/

   chown -R mysql:mysql mysql-clusterSync
   ```

Starting the environment

1. Start the NDB manager on mysql01
   ```
   ./start_ndbdmgm2.sh
   ```
2. Start the NDB manager on mysql02
   ```
   ./start_ndbdmgm2.sh
   ```
3. Start the NDB manager on mysql03
   ```
   ./start_ndbdmgm2.sh
   ```
4. Start the NDB manager on mysql04
   ```
   ./start_ndbdmgm2.sh
   ```
5. Start the NDBD on ndb01
   ```
   ./ndb-start1.sh
   ```
6. Start the NDBD on ndb02
   ```
   ./ndb-start1.sh
   ```
7. Start the NDBD on ndb03
   ```
   ./ndb-start1.sh
   ```
8. Start the NDBD on ndb04
   ```
   ./ndb-start1.sh
   ```
9. Start mysql on mysql01
   (a) Create the SQL files - this step is only necessary the first time the environment is started.
      i. Run as root user
         ```
         mysql_install_db --no-defaults --datadir=/var/lib/mysql-clusterSync
         /65 --basedir=/usr
         ```
      ii. Change the privileges to all directories under /var/lib/mysql-clusterSync/65
         ```
         chown mysql:mysql -R 65
         ```
   (b) Start mysql

   i. Only mysql users can start the mysql

      ```
      su mysql
      ```

   ii. To start the mysql

      ```
      nohup mysqld --defaults -file=/etc/my.cnf &
      ```

10. Start mysql on mysql02

    (a) Create the SQL files - this step is only necessary the first time the environment is started.

       i. Run as root user

          ```
          mysql_install_db --no-defaults --datadir=/var/lib/mysql-clusterSync
          /67 --basedir=/usr
          ```

       ii. Change the privileges to all directories under /var/lib/mysql-clusterSync/67

          ```
          chown mysql:mysql -R 67
          ```

    (b) Start mysql

       i. Only mysql users can start the mysql

          ```
          su mysql
          ```

       ii. To start the mysql

          ```
          nohup mysqld --defaults -file=/etc/my.cnf &
          ```

11. Start mysql on mysql03, follow step 10.
    Important: change SQL id from 67 to 55.

12. Start mysql on mysql04, follow step 10.
    Important: change SQL id from 67 to 57.

Note: Can use the service mysql start, but will only start one mysql, not a new instance.