A High Availability (HA) MariaDB Galera Cluster Across Data Center with Optimized WRR Scheduling Algorithm of LVS - TUN

Bagus Aditya, Tutun Juhana
Sekolah Teknik Elektro dan Informatika Institut Teknologi Bandung
Jl. Ganesa No. 10 Bandung, Indonesia Phone: +62-22-2502260 Fax: +62-22-2534222
E-mail: bagus.aditya.it@gmail.com, tutun@stei.itb.ac.id

***Abstract*—Data availability and reliability is very important along with the growth of information and communication technology services today. Needs of the powerful and affordable database system infrastructure to be the most considered thing by the application service provider. This paper presents the powerful and scalable database open source solution infrastructure which can be distributed across the data center in the world to provide any application or content services. Database infrastructure in this paper using MariaDB Galera Cluster which geographically distributed across data center and Linux Virtual Server (LVS)- Tunnel with optimized weighted round robin algorithm as the load balancer. Optimization in this weighted round robin algorithm are focused on assignment value of weight, based on the condition of database server status. The condition include total active thread (active connected client) and Query per Second Average.**

***Keywords*—MariaDB Galera Cluster, LVS-TUN, Geo Load Balancing**

I. INTRODUCTION

[13]High availability is essential for any organizations interested in protecting their business against the risk of a system outage, loss of transactional data, incomplete data, or message processing errors. Servers don't run forever. Hardware components can fail. Software can crash. Systems are shutdown for upgrades and maintenance. Whatever the reason, when a server goes down, the applications and the business processes that depend on those applications stop. For a business interested in being available at all times, HA clustering is a practical solution. High availability clusters allow the application and business process to resume operations quickly despite the failure of a server and ensure business is not interrupted. High availability clusters are simple in principle. Two or more servers are joined or clustered together to back each other up. If the primary server goes down, the clustering system restarts the application on one of the other servers in the cluster, allowing the business to continue operating normally. The servers are connected using a network or serial interface so they can communicate with each other. With this kind of clustering there is no need to modify the application.

The principal of this High availability solution inspired us to design the powerful and scalabe Database system as a core of the business. The clustered database server distributed across data center in some country. This method is to reduce the risk in a data center (i.e. bomb,fire, power outage, netowork maintenance,etc.) , so there will no failure response at the system or application because of down database server.

In this paper we used MariaDB as database server. [1]MariaDB is based on the open source MySQL code. It is a branch, or fork, of the source code. Then to clustering the database server we used Galera Cluster. [9]Galera from Codership is a multi master cluster implementation for MySQL and MariaDB.

[16]Load balancer for clustered database server distributed across data center, we used Linux Virtual Server is a software tool that directs network connections to multiple servers that share their workload, which can be used to build highly scalable and highly available services. Prototypes of Linux Virtual Server have already been used to build many sites of heavy load on the Internet, such as Linux portal www.linux.com, sourceforge.net and UK National JANET Web Cache Services.

II. RELATED WORK

1. MariaDB Galera Cluster

[9]Galera from Codership is a multi master cluster implementation for MySQL and MariaDB. Multi master clustering is a type of clustering that allow writes to any server in the cluster, but avoids the complexities of distributed locking, shared disk systems and substitutes these with a more advanced replication system. Galera use the InnoDB Storage Engine of the database server and a compact library is used to communicate between the involved database servers.

978-1-4673-8447-6/15/$31.00 ©2015 IEEE

[9]To run Galera, no additional services than the Galera enabled database servers themselves are needed. In a Galera Cluster, all servers are active and database requests can be addressed to anyone of them. This means that the system is always on. As all servers are active, no recovery is needed when an application switch from using one server to the other. From an availability point of view, this is a great benefit. As there is no shared resources and as all the servers are equal, the issues with a single point of failure is removed.

[9]In terms of scalability, this also has tremendous benefits, as the writes may be scaled beyond a single server. Due to Galera using Optimistic locking, scalability can be retained even for writes, despite the fact that all servers hold the same data. In a Cloud environment, elasticity is key. The ability to add, and remove, servers as they are needed. The ability to replace a server with a different one, while still keeping the service running is necessary in a cloud. Also, the ability to optimize the size of each server in a cloud to create the most cost effective setup is another key. Galera allows all this, servers can be added and remove from an existing cluster while the system is fully online .

2. Linux Virtual Server (LVS)

Linux Virtual Server (LVS) is an open source project which developed by Wensong Zhang. Linux Virtual Server has ability to load balance the workload of the multiple server which running Linux kernel based operating system. [13]Load balancing is a computer networking method for distributing workloads across multiple computing resources, such as computers, a computer cluster, network links, central processing units or disk drives. Load balancing aims to optimize resource use, maximize throughput, minimize response time, and avoid overload of any one of the resources .

[16]Linux Virtual Server directs network connections to the different servers according to scheduling algorithms and makes parallel services of the cluster to appear as a virtual service on a single IP address. Client applications interact with the cluster as if it were a single server. The clients are not affected by interaction with the cluster and do not need modification. Scalability is achieved by transparently adding or removing a node in the cluster. High availability is provided by detecting node or daemon failures and reconfiguring the system appropriately .

In this paper we used IPVS is a software which developed by LVS project that implement advanced IP load balancing software inside the linux kernel. [16]IPVS implements transport-layer load balancing, usually called Layer 4 LAN switching, as part of the Linux kernel. IPVS runs on a host and acts as a load balancer in front of a cluster of servers.

There are 3 IP load balancing techniques developed by LVS. There are LVS/NAT, LVS/TUN and LVS/DR. Each techniques has different purposes and implementation technique [16]. LVS/NAT usually used when LBserver receive request from client at internet/intranet (via Virtual IP (VIP)) then rewrite the request to Real server which connected at private network then real server sending reply back via Lbserver to be forwarded to client. This technique has good security because of the network of real server separated from client because Lbserver acts as gateway. LVS/TUN usually used when LBserver receive request from client at internet (via Virtual IP (VIP)) then forward the request to Real server which connected in any network (WAN/LAN) via tunneling IP then real server sending reply directly to client. [16]The real servers can have any real IP address in any network, and they can be geographically distributed, but they must support IP tunneling protocol and they all have one of their tunnel devices configured with VIP. LVS/DR has setup and testing is the same as LVS-Tun except that all machines within the LVS-DR (ie the director and realservers) must be on the same segment (be able to arp each other). [16]The load balancer and the real servers must have one of their interfaces physically linked by an uninterrupted segment of LAN such as a HUB/Switch. The virtual IP address is shared by real servers and the load balancer. All real servers have their loopback alias interface configured with the virtual IP address, and the load balancer has an interface configured with the virtual IP address to accept incoming packets.

In this paper we used LVS/TUN to load balance the workload of MariaDB Galera Cluster which distributed across data center which located in separate geography. This method was used because of high availability reason. When real server distributed across data center which located in separated country, it can reduce the possibility risk of natural disaster or political issue which can affect the condition of data center. In the other research, load balancer actually can be distributed also in separated data center as failover server using heartbeat, but in this paper we only focused in the real server which distributed accross data center.

III. IMPLEMENTATION

The infrastructure of mariaDB galera cluster across the data center with LVS/TUN as load balancer is shown below.

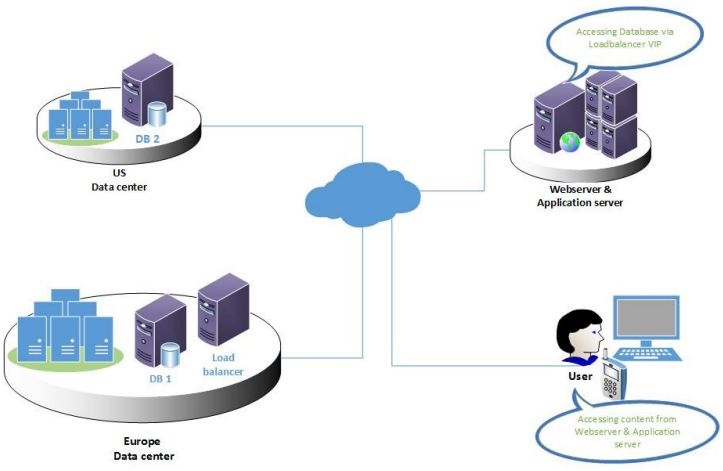

Fig. 3.1 Network Infrastructure

In the above figure shown that DB1 and DB2 located at different data center. In the implementation we used dedicated server hosting at Europe and US which act as

Loadbalancer (LVS) and Real Server (DB) with this following detail of server specification :
CPU: AMD Athlon™ X2
(Dual-Core, 2x 1.8 GHz)
Main Memory : 4 GB DDR2
Hard Disks : 2x 320 GB SATA
Link : 100 Mbit/s
Bandwidth guaranteed : 100 Mbit/s
External Connections: 550 Gbit/s
Operating System CentOS 6

Below is the detail of network Interfaces :

Load Balancer (Location at Europe)
eth0 : 62.75.222.15
eth0:1 : 62.75.223.25

Real Database Server 1 (DB1) (Location at Europe)
eth0 : 62.75.202.135
tunl0 : 62.75.223.25

Real Database Server 2 (DB2) (Location at US)
eth0 : 69.64.38.77
tunl0 : 62.75.223.25

On above details, shown IP address 62.75.223.25 as Virtual IP which shared by load balancer to real database server. This VIP is accessed by application server to access the database server.

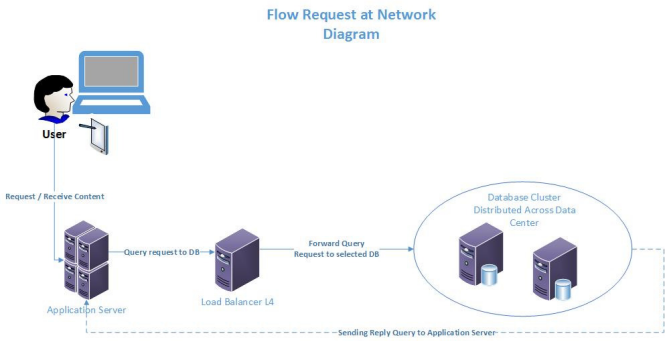

Fig. 3.2 Network Diagram Flow Request

At network diagram above is shown that when user requesting content at application server, the application server sending query to load balancer to selected real database server. [16]The load balancer encapsulates the packet within an IP datagram and forwards it to a dynamically selected real database server. When the real database server receives the encapsulated packet, it decapsulates the packet and finds the inside packet is destined for VIP that is on its tunnel device, so it processes the request, and returns the result to the application server directly. Then, application server replying content to the user.

This implementation usually has an issue at forwarding request to selected real server. This because there are some of ISPs at data center doesn't support IP tunneling protocol. They block spoofing IP activity because of security reason.

IV. PROPOSED ALGORITHM OF HIGH AVAILABILITY MARIADB GALERA CLUSTER WITH OPTIMIZED WRR SCHEDULING ALGORITHM OF LVS-TUN

LVS has some scheduling algorithms to distribute the workload to the real server. There are static and dynamic scheduling algorithms. Round robin (rr) and weighted round robin (wrr) algorithm are included in static algorithms. Then, another dynamic algoritms are least connection (lc), weighted least connection (wlc), etc. The algorithms has strengthness and weakness depends on the infrastructure and the service/application which running in the real server. In this case, we made optimization at weighted round robin (wrr) algorithm at LVS to share the workload of MariaDB Galera Cluster distributed across data center in separated country.

[16]The weighted round-robin scheduling is designed to better handle servers with different processing capacities. Each server can be assigned a weight, an integer value that indicates the processing capacity. Servers with higher weights receive new connections first than those with less weights, and servers with higher weights get more connections than those with less weights and servers with equal weights get equal connections.In the implementation of the weighted round-robin scheduling, a scheduling sequence will be generated according to the server weights after the rules of Virtual Server are modified. The network connections are directed to the different real servers based on the scheduling sequence in a round-robin manner.

We use wrr algorithm because in the implementation, this algorithm is commonly used and more stable than another dynamic algorithms. Additionally we can easily customized the weight of real server destination depends on the MariaDB status and condition. Actually, LVS has dynamic algorithms to check status and condition of the real server. But, the algorithm is running for common server or application infrastructure. In this paper, firstly we check the real status of MariaDB server. Then after knowing the variable status that indicates the best performance of server, we sort the server based on those value of performance variable status. After all, run the algorithm script in every 3 seconds to update the status of the real server. This explanation can be shown in the following flow chart.

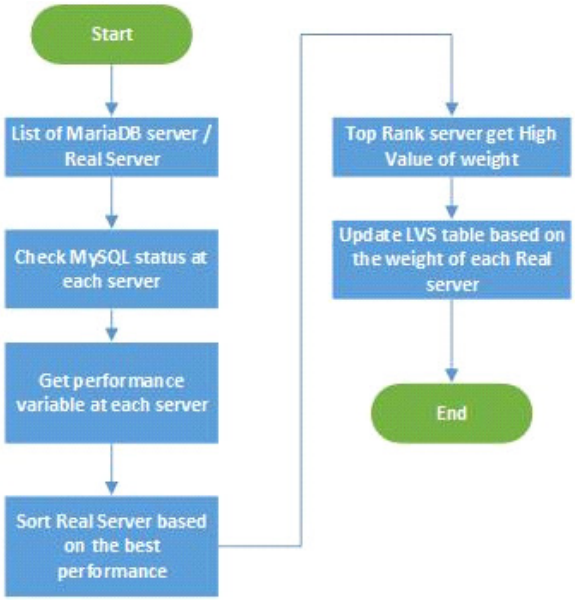

Fig. 3.1 Flow Chart Optimizing wrr in LVS table

This algorithm script is running in the crontab of load balancer server to automatically update the LVS table. This crontab of this script is executed in every 3 seconds. The details of algorithm script to optimize the LVS table is shown below.

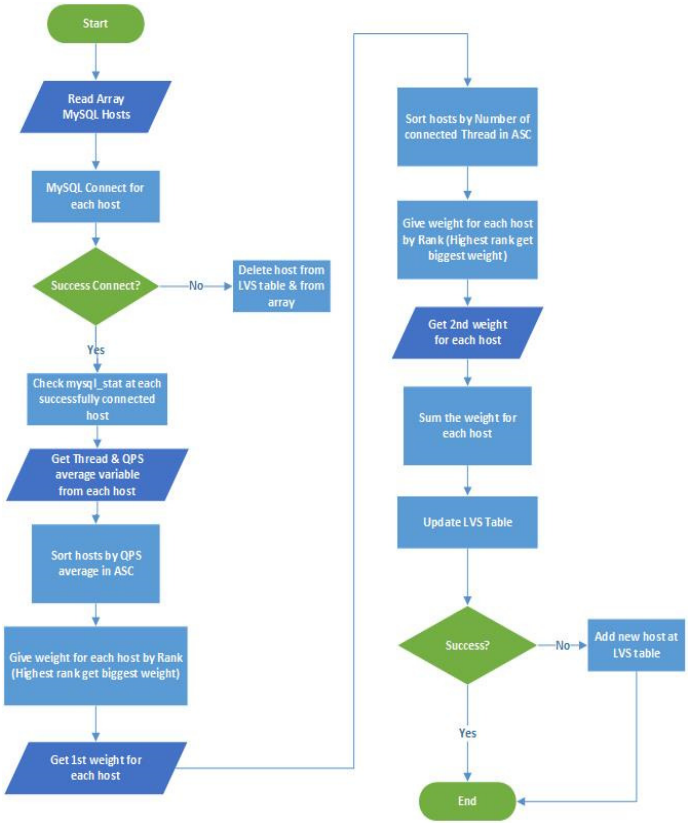

Fig. 3.2 Flow Chart Detail of optimizing wrr in LVS table

At those algorithm we used connected thread and query per seconds average as variable of MariaDB server performance. Connected thread variable indicates the total client that actively connected to MySQL server's thread. So, in this case we can know server capacity's status. If the server has free capacity we can use it as priority server. Then, query per second average indicates that the server load status. If this value has high value, this means the server is busy. Then if server is busy it will execute query longer. So, we can conclude that if the value of average query per second of the server is low we can use it as priority server.

Then as the principal of weighted round robin scheduling algorithm, the top priority server will be assigned to higher weight. After all server has been assigned the weight then it will update the LVS table, so the load balancer can decide then forward the query request to selected priority server.

V. SIMULATION

Infrastructure implementation and simulation of High Availability MariaDB Galera Cluster with optimized wrr algorithm was tested in the real hosted dedicated server in some data centers as described at point III. At simulation test, we were using 2 servers as client which located at the different data center which running sysbench to generate OLTP (online transactions processing) benchmark test to the database server.

The first server generate OLTP transactions through load balancer with various threads, then the second server generate OLTP transactions directly with constant thread to the priority database server based on the weight that has been assigned before depend on the server condition. We're using the second server to simulate when the priority database server is busy, can it handle the new transaction which coming from load balancer? Then, how is the performance (throughput/response)?

OLTP benchmark that has been generated from the first server was using various threads starting from 8 until 512 threads with 10000 maximum requests to 2 million rows of sample data in the database server. The result of benchmarks is shown in the following graph.

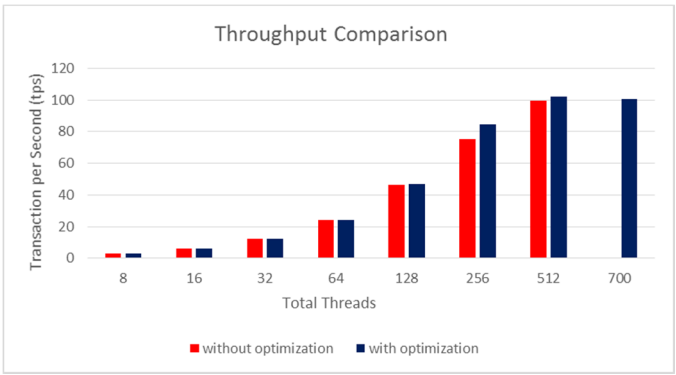

Fig. 5.1 Transaction per Second graph

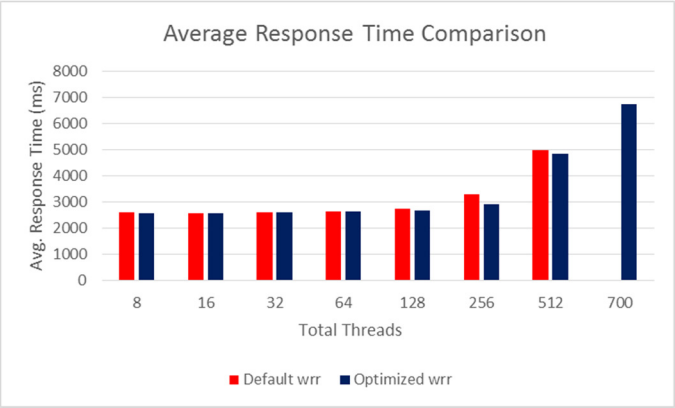

Fig. 5.2 Avg. Response Time Graph

The result of benchmark simulation, shows that the optimized wrr algorithm can handle more threads and the response time is faster than the default wrr algorithm. At the default wrr simulation, when the load balancer receives more than 512 thread, the connection to MySQL server was lost. It means that the default wrr algorithm can not handle the transaction because the priority server is very busy. After we use optimization in the wrr algorithm, when the clustered database server has different load capacity, it can normalize and balancing the load at the each server so the transaction can be handled.

VI. CONCLUSION

The proposed optimization wrr algorithm at LVS/TUN infrastructure of high availability MariaDB Galera Cluster can be work effectively in the dynamic environment (i.e internet). Because when every server connected to the cloud environment, especially in this case of paper we use clustered database server across data center, the load capacity and the condition of server can be dynamically changed. There are many conditions that make priority server busy and not available anymore to handle the new transaction request, i.e. system security attack, data center trouble (i.e fire, power outage, network maintenance), etc. The automated changes at wrr algorithm is very helpful to adapt the condition of dynamic environment (i.e internet) at the clustered database server which distributed across data center in the different geographical location.

VII. REFERENCE

[1] Bartholomew,Daniel."MariaDB vs.MySQL" in MariaDB whitepaper. September. 2012

[2] Chaurasiya,Vijay, Dhyani, Prabhash, Munot,Siddharth. "Linux Highly Available (HA) Fault-Tolerant Servers" in 10th International Conference on Information. DOI Technology.10.1109/ICIT.2007.58

[3] Daryapurkar,Akshay , Mrs. Deshmukh,V.M. "Efficient Load Balancing Algorithm in Cloud Environment" in International Journal Of Computer Science And Applications Vol. 6, No.2, Apr 2013

[4] Dua,Ruchika, Bhandari,Saurabh. "Recovery in Mobile Database System". University of Missouri Kansas City.2006

[5] FromDual."High-availability with Galera Cluster for MySQL".www.fromdual.com

[6] Ingo,Henrik. "How to evaluate which MySQL High Availability solution best suits you" in Percona Live MySQL Conference and Expo.2013

[7] TungYang,Chao, TzuWang,Ko, Li,KuanChing , and Lee,Liang-Teh. "Applying Linux High-Availability and Load Balancing Servers forVideo-on-Demand (VOD) Systems". Tatung University. K.Aizawa,Y. Nakamura, and S. Satoh (Eds.): PCM 2004, LNCS 3332, pp. 455–462, 2004

[8] MarkLogic. High Availability & Disaster Recovery. MarkLogic Data sheet

[9] MariaDB. "MariaDB and MySQL Clustering with Galera" in MariaDB whitepaper

[10] MAZILU,Marius Cristian."Database Replication" in Database Systems Journal vol. I, no. 2/2010

[11] Mohy,Ahmed , Makarem,Mohamed Abul."A Robust and Automated Methodology for LVS Quality Assurance".Egypt.2009

[12] Moniruzzaman,A. B. M. , Hossain,Syed Akther. "A Low Cost Two-Tier Architecture Model for High Availability Clusters Application Load Balancing" in International Journal of Grid and Distributed Computing Vol.7, No.1 (2014), pp.89-98

[13] Moniruzzaman,A. B. M. , Md.Waliullah , Rahman, Md.Sadekur ."A High Availability Clusters Model Combined with Load Balancing and Shared Storage Technologies for Web Servers" in International Journal of Grid Distribution Computing Vol.8, No.1 (2015), pp.109-120

[14] Ni,James. "System High Availability Architecture".Conning Technology.2008

[15] Oracle."MySQL: A Guide to High Availability".A MySQL Strategy Whitepaper.2015

[16] Zhang,Wensong ."Linux Virtual Server for Scalable Network Services". National Laboratory for Parallel & Distributed Processing Changsha, Hunan 410073, China