International Journal of Science and Engineering Applications
Volume 4 Issue 6, 2015, ISSN-2319-7560 (Online)

Accessing and Modifying Sqlite Remotely for Catering Multi Client Access

Sharayu Lokhande
Department of Computer Engineering
Army Institute of Technology, Pune

Vaishali Ganganwar
Department of Computer Engineering
Army Institute of Technology, Pune

Abstract: SQLite is a lightweight database management system and a Stable serverless database with almost zero difficulty in installations. SQLite does not support client server facility due to the write lock issue. For expedite multi-client access to the central database, multiple instances of the database on the central system can be created and later integrating these instances to give the resultant product. Accessing these instances remotely would be a solution to the write lock issue. As a result of creating multiple instances of the database on the same system, there might be a heavy traffic which could lead to reduce performance. To handle this cloud computing concept of High Availability which refers to a system or component that is continuously operational for a desirably long length of time.

Keywords— Remotely Access, SQLite, High Availability

1. INTRODUCTION

A lightweight database system is a high- performance, application-specific Database Management system. It differs from a general- purpose (heavyweight) [1] DBMS in that it omits one or more features and specializes in the implementation of its features to maximize performance. Although heavyweight monolithic and extensible DBMS might be able to emulate LWDB capabilities, they cannot match LWDB performance.

SQLite is a software library that implements a SQL engine. It has been used with great success as on-disk file format: allows the developer to handle data in a simple way, but also have the use of database features (such as undo, redo, etc.). In embedded device environment, in which there is low-concurrency and there are little or medium size datasets, SQLite is the right choice. If we want to save the data in a common place, i.e., Remote Server until now there is no easy mechanism to implement this.

The need for storing information in remote server exists to have centralized access to data by the users. The idea of storing information in remote server is implemented using Web Services (plugin) which can save the data in the Remote database like SQL Server and retrieve as and when required. When a project is developed, a group of developers/testers are involved. They will need concurrent information for development which can be done using a centralized database. For example feedback is collected from different customers for a product and it is more feasible to store it in a centralized repository that can be used by the entire for improvements and further development. So we require a remote access to SQLite [2]

to be used by all of them.The relevant changes need to be reflected and others discarded. SQLite has write lock issues which have to resolve by creation of different instances of the database. Testers can access and debug the problems directly and provide the information without having to install the entire system or database files.

1.1 High availability

Virtualization, a technique to run several operating systems simultaneously on one physical server, has become a core concept in modern data centers, mainly driven by benefit of application isolation, resource sharing, fault tolerance, portability and cost efficiency. A special middleware, hypervisor, abstracts from physical hardware resources and provides so called virtual machines acting like real computers with their own (virtual) hardware resources. High availability system [3] design approach and associated service implementation that ensures a prearranged level of operational performance will be met during a contractual measurement period. Enabling high availability we can detect any point of failure to propagate reliable crossover, if needed. High availability is a characteristic of a system. The definition of availability is Ao = up time / total time. If (total time - down time) is substituted for up time then you have Ao = (total time - down time) / total time. Determining tolerable down time is practical. From that, the required availability may be easily calculated. Here a small network has made with a master, slave (replica of master) backing up data, controller and a user virtual machine. Controller will be constantly checking the master for downtime and doing crossover to slave in case tolerable down time is exceeded. For this purpose we will use open source tools like heartbeat, pacemaker and DRBD.

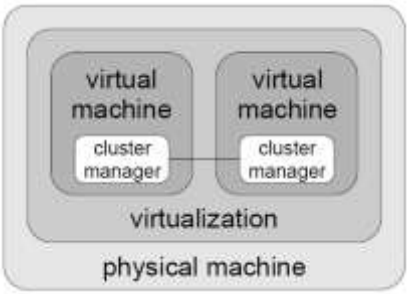

Fig. 1 Virtualization

2. SOLUTION FOR SQLITE

Plugin as an interface has been used for remote accessing of SQLite. A connection to the remote system is made through ssh. As soon as the remote system is accessed the database is copied to the local system and launched through SQLite manager. Now the remote database will be in synch throughout. In case of read operation the local system will not be updated (by copying remote database) as no updations have been made. In case of modification/updation of Database, an instance for it is created corresponding to the developer/ tester which will be used for further development by this particular tester/developer. The local database will be copied again to the remote system. The final product is developed at the remote system by using data from these instances.

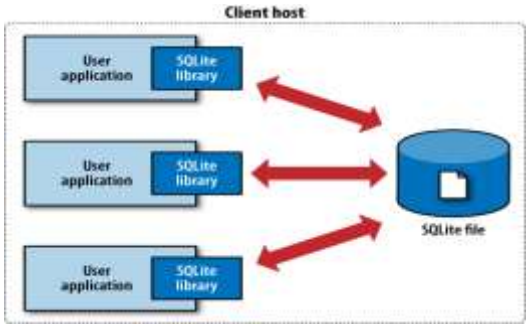

Fig. 2 SQLITE Architecture

3. SOLUTION FOR HIGH AVALABILITY

A controller which sends heartbeat or pace-maker (OS tool) to master, slave and checks the response time.
Availability is calculated by Ao = (total time - down time) / total time. Determining tolerable down time is practical. Using this threshold value is determined. If response time exceeds threshold value, controller shifts from master to slave. All further queries are directed to slave by the controller.
In case master is updated then the last copied time is checked for the slave and synched with the master. For this purpose an open source tool (like DRBD is used). Heartbeat is a daemon that provides services of clustering; this allows the exchange of messages between the machines running Heartbeat and check the health of them. Heartbeat is used for checking if all the nodes are running is recommended to use a dedicated interface for it. Pacemaker is a resource manager that provides a full management of the resources provided by the cluster.

4. GLUSTERFS

GlusterFS [5] is an open source, distributed file system capable of scaling to several petabytes and handling thousands of clients. GlusterFS clusters together storage building blocks over Infiniband RDMA or TCP/IP interconnect, aggregating disk and memory resources and managing data in a single global namespace. GlusterFS [6] is based on a stackable user space design and can deliver exceptional performance for diverse workloads.

Attributes of GlusterFS include:
Scalability and Performance
High Availability
Global Namespace
Elastic Hash Algorithm
Elastic Volume Manager
Standards-based

By considering the advantage of two different features to provide a highly available, scalable NFS and CIFS service. First, the use DNS round robin to have each client use one of the Gluster servers[6] for their mounts. Then, CTDB will provide virtual IPs and failover mechanisms to ensure that, in the case of a server failure, failover is transparent to clients. Define DNS entries for two load balanced services, called glusternfs and glustercifs. Virtual IPs combined with CTDB IP failover; it allows having both load balancing and high availability.

Set a low TTL for the records so, if a virtual IP is down while a client is trying to mount, the client can retry using a different one. To configure CTBD start with a single volume called vol1, configured as distributed + replicated (2 replicas), and export it using NFS and CIFS.

5. DRBD AND DRB DLINKS

5.1 DRBD

One mechanism for sharing data between two machines is to use an external RAID array. The primary drawback to this is cost, with typical array configurations costing no less than $2,000. DRBD is a”Distributed Replicated Block Device”
that allows similar results to be achieved on local discs using a network connection for replication. DRBD can be thought of as a RAID-1 (mirrored drives) system that mirrors a local hard drive with a drive on another computer. DRBD includes mechanisms for tracking which system has the most recent data, ”change logs” to allow a fast partial re-sync, and startup scripts that reduce the likelihood that a system will come up in
”split brain” operation.

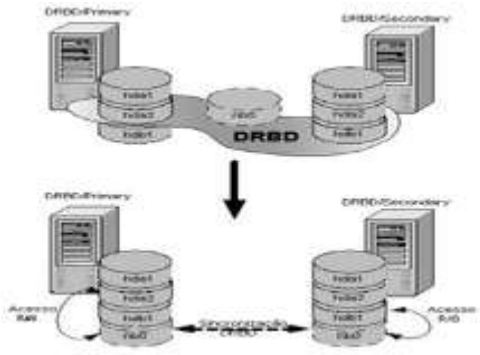

Fig. 3 Overview of DRBD concept

A dedicated network using a direct cross-over network connection is set up between the machines.

5.2 DRBDLINKS

In a typical system running DRBD[8], there will be many directories and files that reside on the shared data partition. A way to handle the

data in the HA Cluster is- Link the normal system files and directories into the shared partition. This means that configuration file and data reside in their familiar locations on the primary system. However, links must be set up when the service starts on the primary and returned to normal when the service is not operating. These links can be maintained in the heartbeat startup and shutdown scripts using the standard ln, mv, and rm commands. The DRBD shared partition will be mounted on"/shared". DRBDLINK Configuration install the"drbdlinks" package Configure the "/etc/drbdlinks.conf" file setup the directories mentioned in /etc/drbdlinks.conf file in the shared partition restart the DRBD close the database to ensure a good copy of the data is made configure heartbeat resource to start and stop DRB- DLinks by modifying the "/etc/ha.d/haresources" file DRBDLinks moves the system "httpd" file to the httpd.drbdlinks" and makes a link to the version in "/shared.

6. CONCLUSION

Environment, in which there is low- concurrency and there are little or medium size datasets, SQLite is the right choice[1]. The drawback of SQLite can be removed by the proposed solution by creating instances and then integrating the modules. HA is implemented with the help of open source tools such as heartbeat, DRDB, GlusterFS, corosync which helps in achieving availability at all times overcoming any failure at the server end.

7. REFERENCES

[1] D. C. Igweze and E. O. Nwachukwu, Lightweight Database System (Lwdbs): An Overview

[2] Kiran Dhokale, Namdeo Bange,Shelake Pradeep, Sachin Malave,Implementation Of Sql Server Based On Sqlite Engine On Android Platform

[3] High availability clustering of virtual machines- possibilities and pitfalls may 2006 Wiesbaden/germany version 1.01

[4] The Expedient Approach for High Availability in Web Server Services for HPC Attained by Clustering using Virtualization International Journal of Computer Applications, Volume 95 No.20, June 2014

[5] GlusterFS: http://www.gluster.org/documentation/ Architecture/internals/Dougw:ANewbie0 sGuidetoGlusterInternals=; V olume95No:20; June2014

[6] http://blog.gluster.org/about/

[7] http://www.linuxlinks.com/article/20130411160756441/GlusterFS.html

[8] https://www.smartseohosting.com/GlusterFS