Realization of the Low Cost and High Performance MySQL Cloud Database

Wei Cao
Alibaba Cloud Computing Ltd.
No. 969，Wenyi Road , Hangzhou, Zhejiang Province,China
+86-13361805660
mingsong.cw@taobao.com

Feng Yu
Alibaba Cloud Computing Inc.
No. 969，Wenyi Road,Hangzhou, Zhejiang Province,China
+86-18605888505
chuba@taobao.com

Jiasen Xie
Alibaba Cloud Computing Inc.
No. 969，Wenyi Road,Hangzhou, Zhejiang Province,China
+86-18691498815
jiasen.xjs@alibaba-inc.com

ABSTRACT

MySQL is a low cost, high performance, good reliability and open source database product, widely used in many Internet companies. For example, there are thousands of MySQL servers being used in Taobao. Although NoSQL developed very quickly in past two years, and new products emerged in endlessly, but in the actual business application of NoSQL, the requirements to developers are relatively high. Moreover, MySQL has many more mature middleware, maintenance tools and a benign ecological circle, so from this perspective, MySQL dominates in the whole situation, while NoSQL is as a supplement. We (the core system database team of Taobao) have done a lot of work in the filed of MySQL hosting platform, designed and implemented a UMP (Unified MySQL Platform) system, to provide a low cost and high performance MySQL cloud database service.

1. INTRODUCTION

UMP (Unified MySQL Platform) system is a low cost and high performance MySQL cloud scheme, which is developed by the core database team of Taobao. It's key module is realized by the Erlang programming language. The system including controller server, proxy server, agent server, API/Web server, analysis and optimization server and log analysis server is depended on the Mnesia[1], LVS, RabbitMQ[2], ZooKeeper[3] and other open source components.

Developers can apply for MySQL instance resources from the platform, and access the data through a single entrance provided by the platform. UMP system maintains and managements a resource pool, to provide master-slave hot standby, data backup, disaster recovery, R/W splitting, database sharding and other service to user transparently.

UMP System reduces the overall cost by a variety of resource virtualization methods, such as, sharing one MySQL instance for multiple small users, letting medium-sized users have a MySQL instance themselves and multiple MySQL instances sharing the same physical machine. In the aspect of resource isolation, UMP system realizes resource virtualization and guarantees the quality of service for users by restricting MySQL process resources through the Cgroup, and limiting QPS in proxy server. In addition, with the integrated use of SSL, IP white list, user log and SQL intercept, UMP system can protect the security of user data.

This paper is organized as follows. Section 2 introduces the overall design of UMP system, the session 3 depicts the system interactions, the session 4 discusses UMP's high availability, the session 5 tells the implementation of the key system function, the session 6 summarizes the whole text.

Proceedings of the VLDB Endowment, Vol. 7, No. 13

2. DESIGH OVERVIEW

To meet the needs of most users, UMP system is designed to serve users transparently, that means users are not aware of perceiving the events of downtime, disaster recovery, flash disconnect and R/W splitting. Our goal is to provide a unified interface, which shields the underlying implementation land provides high availability of services.

2.1 Architecture

The architecture of the first edition of UMP system is based on mysql-proxy 0.8. We have fixed several bugs of mysql-proxy, and made some modifications on the state machine which deals with user connections and database connections in the proxy plug-in. We also developed Lua script to implement some functions, such as obtaining the user authentication information and the address of back-end database from a central database, establishing a connection to the backend database, forwarding packets, and etc.

However, in the process of developing and deploying the first edition, we realized a few problems.

First, mysql-proxy 0.8 only provides simple and crude supports on multithreading where multiple threads share the same message queue and listen to the same socket pair channel, which often causes "Thundering Herd" phenomenon. In addition, mysql-proxy uses global Lua lock, which means only one worker thread can execute Lua scripts (although improvements will be made in version 0.9). As a result, the performance of mysql-proxy under multi-threaded mode is far from keeping linear growth to the number of CPU cores.

Second, the framework of mysql-proxy limits our extension for UMP system, such as achieving the user's connection limit, QPS

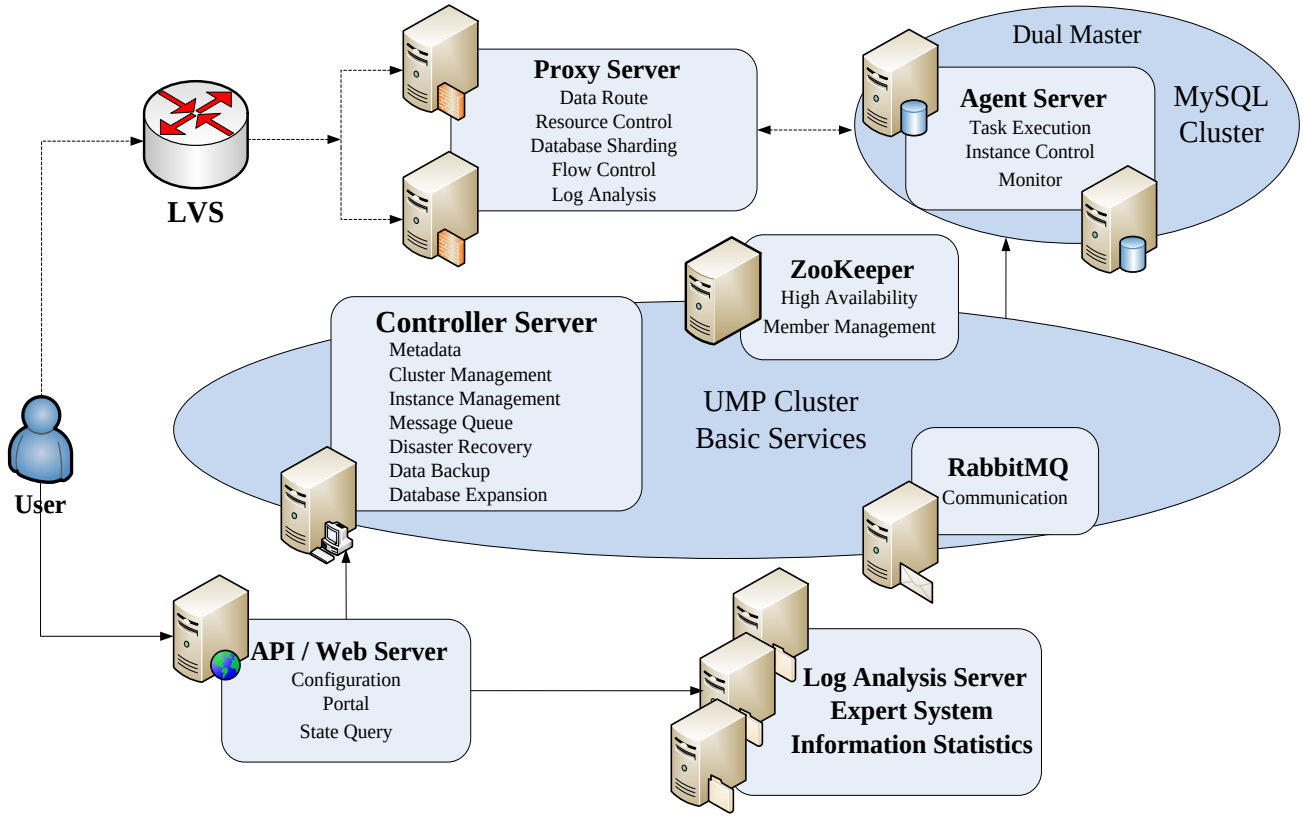

Figure 1. Architecture diagram of UMP system.

limit, master-slave switch, R/W splitting, database sharding, and a series of other functions.

As a result, we decided to rewrite a proxy server to replace the mysql-proxy by using the Erlang programming language. Figure 1 shows the architecture of the current UMP system. The system includes controller server, proxy server, agent server, API/Web server, log analysis server and information statistical server. Also some open source components are involved, such as LVS, RabbitMQ, ZooKeeper and so on. The entire project currently has 50,000 lines' Erlang code, 30000 lines' C/C++ source code, and 20000 lines' source code in other languages.

Erlang is a structured, dynamic and functional programming language. OTP is a framework and platform for the development of distributed and high fault-tolerant Erlang applications. The reason why we used Erlang language to rewrite the proxy server was that Erlang/OTP perfectly abstracts the required elements for developing a distributed, highly fault-tolerant application, including network programming framework, serialization and de-serialization, fault tolerance and hot deployment.

2.2 Interface

We developed a single unified entrance to meet the user's needs which performing all operations through webpages, including create and delete instances, DB control, promotion and demotion, transfer, and etc.

The API/Web server provides the system management interface for users, which is based on open source projects called Chicago Boss and Mochiweb. Mochiweb provides the http/https service, while Chicago provides MVC frameworks like Rails Chicago. Chicago has good support for concurrency, each request of it occupies a lightweight Erlang process.

In addition, the interfaces we designed is compatible with the interfaces of AWS.

2.3 Proxy

The first important reason why we developed the proxy is to fix the defects from mysql-proxy, and the other one is to realize a transparent service mechanism. At the same time, we can add some features on proxy, such as flow control and log analysis.

The proxy server provides the service to access the MySQL database for users, which fully implements the MySQL protocol. The users can use the existing MySQL client to connect to the proxy server, and then the proxy server gets the authentication information of users, resources quotas limit (for example the maximum number of connections, QPS, IOPS, etc.) by user name, as well as the address of MySQL instance (list), after that it transmits the user's SQL queries to the correct MySQL instance. In addition to the basic functions of data routing, the proxy server also implements the following functions, including limiting resource, shielding MySQL instance faults, R/W splitting, database sharding, recording user access logs, and etc. The proxy servers are stateless in order to provide high availability service.

2.4 Metadata

We store the metadata which is used by the UMP system in the metabase of the controller server. The stored metadata is mainly including cluster membership information, user configuration and status information, the mapping between MySQL instances, task

information, and backup migration information. In order to achieve distributed storage of metadata, we put the operations of reading and writing metadata into a set of Mnesia distributed database services. Other server components can obtain the corresponding data only by sending a request to the controller.

2.5 Controller

UMP needs a perfect management and control service for the backend cluster, so our controller server provides a variety of management services for UMP cluster, including metadata storage, cluster membership management, MySQL instance management, backup, migration, expansion and other functions.

The cluster membership management, including cluster management division, host distribution, logical group configuration, and etc. is used to carry out overall control of instances, such as upgrade and migration. The management operations of a MySQL instance, like creation, deletion, migration and expansion, are all encapsulated into a task respectively, these tasks will be broken down into idempotent sub-steps according to business needs, that means even the same steps are executed repeatedly by the agent server, there are not any side effects, so it offers retry operation when the task fails. The agent server is deployed on a machine running MySQL processes, used to manage MySQL instances on the physical machine, performing the actual operation, such as create, delete, backup, migration and master-slave switching.

2.6 Communication

The communication between nodes in UMP system (excluding the transfer of large data in SQL queries, logs and other streams, these go directly to these TCP's) are all through RabbitMQ, which is used as a messaging middleware to ensure the reliability of the message communication.

When initialized, the cluster will create a queue for each node in the cluster in RabbitMQ, which is used as a “mailbox” for each node. When a node want to send a message to other node, it just write a message to the "mailbox" of the other side, regardless of whether the other node is online or not, then the RabbitMQ client running on the other side will receives the message ,and call the appropriate handling routine. After the message is processed, the client will return an ACK packet to RabbitMQ, and the RabbitMQ will delete this message from the "mailbox". We can achieve RPC based on RabbitMQ, the client also writes a message to the sender's Reply "mailbox" in addition to send a ACK packet to RabbitMQ to delete the Request message. RabbitMQ is transaction supported, it can guarantee delete Request and reply message are done in an atomic operation, hereby providing a reliable communication service.

3. SYSTEM INTERACTIONS

Our design should simplify all operation since the UMP system contains multiple components. As a result, we created this cooperation job system with multiple components to help users to do with instance controlling, disaster recovery, dilatation and other functions.

3.1 Control Order

The controlling operation designed in UMP system is launched by API/Web servers, processed by a series of servers and finally sent to the machine holding MySQL instance to do the work. Task processing and scheduling is transparent to users and the unitized interface brings a convenient experience to users.

In Figure 2, we illustrate this process by following the control flow of instance-create through these numbered steps.

1. The clients launch a creating-instance task by accessing the API/Web server.

2. When the controller server receives the notice from API/Web server, it initializes the context of working flow and assigns the steps of the task to memory.

3. The steps to be executed will be distributed to corresponding agent server according to business. The agent server runs the steps and creates a MySQL instance in physical machine.

4. The progress and result that the agent server ran the task will all be fed back to controller server.

5. As soon as the feedback is received by controller server, it will record the behavior and result of the execution to meta database. So will these information be displayed on Web Console UI.

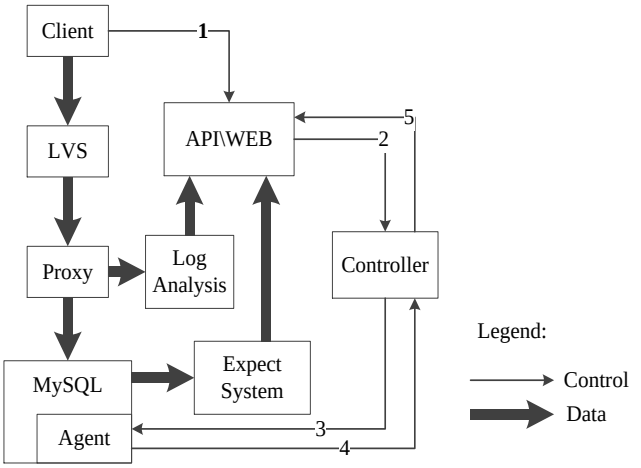

Figure 2. Control order and data flow.

Others controlling processes are similar as above, and controller server can do work concurrently between different instances.

3.2 Data Flow

Data, which refers to the DML statements or Query statements that manipulate the database here, is separated with control flow in system. This will achieve better controlling and management.

Data chain flows through LVS first, converting user's link string to VIP and VPORT, which help to build connection to the server. After the data flow passes LVS and reaches proxy server, the router function will do its work and transfer user's SQL statement to the right SQL instance. The data will be recorded in user access log while passing proxy server which will be processed after fetching by the log analysis server.

3.3 Operation and Maintenance

In order to better understand cloud database running status, UMP party provides an operational log system, mainly is the collection

and analysis of relevant logs and presented to the system management interface, including log analysis, expert system, etc.

3.3.1 Portal

Considering the interaction of unity, the portal is present in the system management interface of API/Web server. The user's access information is fetched by log analysis server and the optimization information provided by expert system will be sent to the Portal on the rendering.

The information statistics server will show the connection numbers, QPS numerical and MySQL instance status on the Portal, for the future implementation of elastic resource allocation and the automated MySQL instance migration provides gist.

3.3.2 Log Analysis

For the convenience of users to analysis their operation of SQL statements, log analysis server will store and analyze the users access logs incoming by proxy server, for the execute slower logs abstract the sentence mode，and implement real-time indexes for user query over a period of slow logs and statistics.

3.3.3 Expert System

Often some users want to know the slow logs which impact their business performance, our expert system will provide the SQL statement analysis and optimization services not only collect the slow logs of MySQL instance for users，but also integrates and analysis these logs, refining to every slow logging statements, analyze the slow reason, and provide our optimization scheme.

4. HIGH AVAILABILITY

As described above, UMP system has many components. The high availability of some key components among them is particularly important. Also their quality of service determines the overall performance of the system and the impact on user.

4.1 Dual Master Replication

For each user system, UMP maintains master and slave MySQL instance with structure of dual master. The proxy will guarantee to write data to master instance. When the database fails, controller server will detect this event and launch a master-slave switching operation, which will inform all proxy server to perform the switching through the RabbitMQ.

When the fault master instance online again, controller server will stop writing slave instance until master and slave databases are consistent and inform the proxy server to switch the write operation to the master instance. This process will make users feel cannot write for a short period of time.

System performs a transparent recovery process to users, so users cannot perceive the master instance downtime and launch event. The proxy provides a persistent connection for user in order to reduce the times of flash disconnection to some extent.

4.2 Controller Leader

In order to achieve high availability, the system will deploy multiple controller servers. They will elect a leader by the distributed lock algorithm provided by the ZooKeeper, which is responsible for scheduling and monitoring various system tasks, such as create or drop instance, backup and migration. These tasks would be divided into multiple steps, and would involve multiple components in the system, such as the main library, the library and the proxy server. In order to provide the function of rollback, we adopt a similar workflow way to achieve it. Each system tasks is divided into Erlang process of multiple stages, after each step and before executing the next step, the middle state is persisted to the Mnesia. If the task is stopped because of a node failure, the leader will detect and re-initiate the task that will go on from the last failed "breakpoint".

4.3 Stateless Proxy

The proxy servers are stateless, so proxy server downtime will not affect the other servers in the system, which will only disconnect a user from the proxy server. Multiple proxy servers using LVS HA program to achieve load balancing, user application will be re-directed to other LVS's proxy after re-connecting, so as to provide high availability level expansion.

5. IMPLEMENTATION

Platform reduces cost through running multiple MySQL instances on a single physical machine, and realizes resource isolation, distribution according to needs and limitations of CPU, memory and IO resources control. The inner maintenance and resource pool management of UMP system provides a series of service such as master-slave hot standby, data backup, migration, R/W splitting, database sharding and data security transparently.

5.1 Resource Control

Referring to the resource management method in some cloud computing system such as the VMware DRS, we implemented a resource pool mechanism to manage the database on the server's CPU, memory, disk and other computing resources. In addition to not affect the user experience, we realized the resource isolation mechanism for MySQL instance.

5.1.1 Resource Allocation

Allocated instance is the unit of resource pool, the administrator can deploy in what rooms, according to the application need to be separately specified computing resources of the library, from the library's resources pool, instance management services choose the lighter server from the resource pools to create an instance of server. Moreover, combined with Cgroup will further resources to facilitate the management and isolation, at the same time it can limit the upper limit of each process group.

Now the system supports three kinds of specifications of the users: small flow users that share the same MySQL instance, medium-sized users that possesses a MySQL instance alone and possession of multiple independent MySQL instance depots table users. For different users, we provide different kinds of service of dynamic expansion and contraction.

5.1.2 Resource Isolation

Isolating resource is particularly important when multiple users share one MySQL instance or multiple MySQL instances share the same host. Currently, we implement the function of resource isolation through combing the use of Cgroup on the database servers to limit MySQL instances resource and the limiting of QPS on proxy servers.

Firstly, we use the cpuset, memcg and blkio sub-modules of Cgroup to limit the maximum CPU usage, memory and IOPS that would be used by MySQL process. Secondly, we increase the delay on proxy server to limit the user QPS and reduce the system resources consumed.

5.2 R/W Splitting

We also realized read/write splitting transparent to users. proxy server will analyze the incoming user SQL statements, will write sent to the master library, read operation load balance distribution to master and slave library. In order to avoid reading before the consistent master-slave synchronization, any data within 300 milliseconds after each write operation to read will be forced to distribute to the master library. Through master-slave multi-threaded replication technology, 300 milliseconds can basically guarantee the data from the master library synchronous to slave library, and this value can be regulated in the configuration.

Besides, in order to share the pressure of reading and writing, system realized the read-only instance, users can put read-only data into read-only instance, so as to reduce the pressure of the master library.

5.3 Database Sharding

The user can specify the user type as multi-instances based on the requirements. This multi-instances type users need to add the rules of database sharding into SQL comments when they are creating tables. The implementation procedure of the database sharding is shown in Figure 3.

Firstly, the proxy server will parse the user incoming SQL statements to extract the information which is needed to rewrite and distribute the SQL statements.

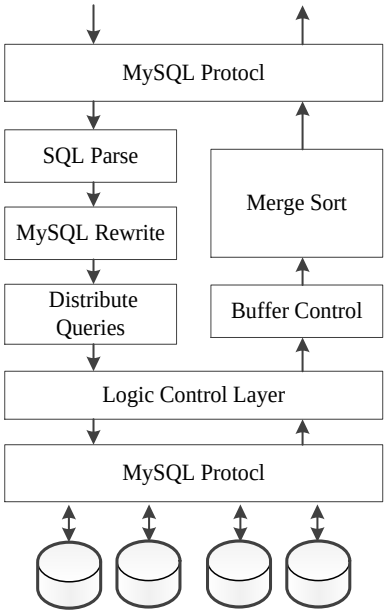

Figure 3. Process of database sharding.

Secondly, the SQL statement is rewritten in the form of sub-statement to be executed on each of the separated table. This mainly include replacing the table names and where conditions. And then the sub-statements are sent to each sub-table to be executed concurrently.

Finally, the result returned from each sub-table is received and merged. To avoid the size of query result set becoming too large, we limit the number of results rows of each MySQL instance by setting the buffer size. At the moment all the sub-tables returned some results the merge sort begin to be performed, and then the merged result is returned to the user.

In order to improving the performance, we used C++ to achieve the parsing and rewriting of the SQL statement and the merging of the results returned from MySQL servers, which is called by the Erlang language state machine through the NIF interface.

5.4 Data Security

User and enterprise security department will be more concerned about the security of their data, we implemented a variety of methods to ensure the safety of the user data:

1. UMP supports the SSL protocol. The proxy server realized the whole MySQL client/server protocol, which can establish an encrypted connection with the client.

2. System generates a list of IP address which is used to access to database by setting up white list. Our users can configure their application server address into the white list in order to increase the security of the account.

3. The proxy server will put all user database operation record to log analysis server, so security department can export log text on a regular basis and scan security vulnerabilities.

4. The proxy server can intercept various types of SQL statements according to the requirement of the security department, such as 'select *' statement on a full table, statements whose result is beyond limit, etc.

6. CONCLUSIONS

Some components in UMP system, such as the proxy server and the server log analysis, has been used in the Poly Tower Platform of Tmall to provide secure data cloud services for electricity and ISV. In addition, the UMP system is also used in the decorating platform of Taobao shop to provide data services for developers. In the next stage, we hope the UMP system can further reduce the cost of internal data storage.

In engineering practice, we adhere to the principle of not reinventing the wheel through taking advantage of open source, mature technology and tools. For example, we implemented the high-performance proxy server on the Erlang network programming framework and the messaging middleware based on RabbitMQ, we manage the server heartbeat using ZooKeeper, and we also make full use of mature tools in the internal group, such as data backup, migration, programs of capacity expansion or contraction and other bin log tools. This principle allows us to concentrate on reducing costs and improving user experience with the limited resources.

7. REFERENCES

[1] Mattsson H, Nilsson H, Wikström C. Mnesia—A distributed robust DBMS for telecommunications applications[M]//Practical Aspects of Declarative Languages. *Springer Berlin Heidelberg*, 1998: 152-163.

[2] Kramer J. Advanced message queuing protocol (AMQP)[J]. *Linux Journal*, 2009, 2009(187): 3.

[3] Hunt P, Konar M, Junqueira F P, et al. ZooKeeper: Wait-free Coordination for Internet-scale Systems[C]//USENIX *Annual Technical Conference*,2010, 8: 9.