1. **An Introduction to MyCAT**

MyCAT is an Open-Source software, “a big database cluster” oriented to the enterprises. MyCAT is an enforced database which is a replacement for MySQL and support transaction and ACID. Regarded as MySQL cluster of enterprise database, MyCAT can take the place of expensive Oracle cluster. MyCAT is also a new type of SQL Server integrated with the memory cache technology, Nosql technology and HDFS big data. And as a new modern enterprise database product, MyCAT is combined with the traditional database and new distributed data warehouse. In a word, MyCAT is a fresh new middleware of database..

MyCAT’s target is to smoothly migrate the current stand-alone database and applications to cloud side with low cost and to solve the bottleneck problem caused by the rapid growth of data storage and business scale.

MyCAT Key Features:

• Supports SQL 92 standard

• Supports Mysql cluster, used as a Proxy

• Supports JDBC connection with ORACLE, DB2, SQL Server, simulated as normal MySQL Server connection

• Supports mysql cluster, percona cluster or mariadb cluster, providing high availability of data fragmentation clusters

• Supports automatic failover and high availability

• Supports separation of read and write, dual-master with multi-slave, single-master with multi-master of MySQL model

• Supports global table, automatically fragment data into multiple nodes for efficient relational query

• Supports the unique fragmentation strategy based on ER-relation for efficient relational query

• Supports multi-platform, easy deployment and implementation

MyCAT adventage:

• Based on Ali open-source product Cobar, whose stability, reliability, excellent architecture and performance, as well as many mature use-case make MyCAT have a good starting. Standing on the shoulders of giants, MyCAT can be able to go farther.

• Extensively drawing on the best open-source projects and innovative ideas, which are integrated into the MyCAT’s gene, make MyCAT be ahead of the other current similar open-source projects, even beyond some commercial products.

• MyCAT behind a strong technical team whose participants are experienced more than five years including some senior software engineer, architect, DBA, etc. Excellent technical team to ensure the product quality of MyCAT.

• MyCAT does not rely on any commercial company. It’s unlike some open-source projects whose important features is enclosed in its commercial products and making open-source projects like a decoration.

MyCAT long-term plan:

• On the basis of Mysql’s support, MyCAT add more support of commercial open-source database, including native support of PosteSQL, FireBird and other open-source databases, as well as indirect support via JDBC of other non-open-source databases such as Oracle, DB2, SQL Server etc.

• More intelligent self-regulating properties, such as automatic statistical analysis of SQL, automatic creating and adjusting indexes. Based on the frequency of reading and writing, MyCAT automatically optimize caching and backup strategies

• Achieve a more comprehensive monitoring and management

• Integrated with HDFS, provide SQL commands, load databases into HDFS for rapid analysis

• Integrated excellent open-source reporting tools to make MyCAT have data analysis capability

1. **MyCAT Architecture**

**Storage**

**Server**

**Front**

**MySQL instance A**

**MySQL instance B**

**MySQL instance C**

**More….**

**Client**

(MySQL CLI/JDBC/ODBC/…)

**MySQL Socket Protocol Handler**

**HeartBeat Checker**

**SQL Router**

**SQL Parser**

**SQL Executor**

**DataNode**

As shown above: MyCAT simulate a MySQL server with Mysql communication protocol and set up a complete logic model of Schema (database), Table (data tables), User and map this logical model to backend storage node DataNode (MySQL Instance) which is a real physical library. Therefore all clients and programming languages can treat MyCAT as MySQL Server to use, without having to develop new client protocol.

When MyCAT receive a SQL request from a client, it will check and parse SQL’s syntax whose results are used for the SQL routing. SQL routing policy has Supports the traditional fragmentation based on table field and also has Supports partitioning strategy based on a unique ER relations of database. For routing to multiple data nodes (DataNode) of SQL, MyCAT will merge all the data sets receiving and then output to the client.

The SQL execution process is to send SQL to a real database backend through the network protocol. For Mysql Server, it sends packets by Mysql network protocol and parses the returned results. If SQL not involving multiple fragments nodes, the results are returned simply and written into the client SOCKET stream. The whole process is in non-blocking mode (NIO).

DataNode is a MyCAT’s logical data node mapped into a backend physical database. In order to achieve high availability, each DataNode is configured with multiple references address (DataSource). When the main DataSource is detected as unavailable, the system will automatically switches to the next available DataSource. And that DataSource can be considered as the Mysql master slave addresses.

1. **MyCAT Schema**

Like any of traditional relational databases, MyCAT also provides a definition of "database" and has user-authorization functions. The following concepts are related to MyCAT logic libraries:

• schema: Logic libraries corresponding to MySQL Database (database). A logical library defines all the logic Tables.

• table: The table is stored in the physical database. Different with traditional database,it needs to be declared its logical data storage-node (DataNode) which is achieved by the definition of table fragmentation rules. The table can define their respective "sub-table (childTable)" whose slice depends on the specific slice address of parent tables. In brief, one record A belonged to a parent table has its all sub-tables stored in the same slice.

• fragmentation rules: It is bundled with a function to a field. According to the value of this field to return the fragmentation (DataNode) number of storage, each table can define a fragmentation rule. All fragmentation rules can be flexibly extended. By default, it provides the fragmentation rule based on the digital and string.

• DataNode: MyCAT logical data node is a specific physical node to store tables, also known as fragmentation node which is through DataSource associated to a specific database on the back end. In general, for high availability, each DataNode is set with two DataSource as Master-Slave mode. When the primary node goes down, the system automatically switches to the slave nodes.

• DataSource: Define an access address of a physical library which is used for bundled onto Datanode.

MyCAT currently define the schema and the relative configurations by the profiles :

• MYCAT\_HOME / conf / schema.xml Definition of the schema, tables, slice nodes, etc.

• MYCAT\_HOME / conf / rule.xml Definition of fragmentation rules

• MYCAT\_HOME definition / conf / server.xml Definition of user and system-related variables, such as ports.

The following figure shows a possible logic library to a physical library (Mysql complete mapping relations) of MyCATd. We can see its powerful ability to slice and flexible integration capabilities of Mysql cluster.

**schema**

**Table B**

**Table A**

**Datanode 4**

**Datanode 3**

**Datanode 2**

**Datanode 1**

**Database c**

**Database e**

**Database a**

**Database b**

192.168.0.1

192.168.0.3

192.168.0.2

**Sharding Strategy**

MyCAT supports horizontal sharding and vertical sharding:

Horizontal Sharding: The data of a table is splitted up into many nodes by row

Vertical Sharding: There are many tables in a database, such as A,B,C. And A is storaged in the first node, B is storaged in the secord node and C is storaged in the third node.

MyCAT implements sharding through defined sharing rule of tables. Every table can bing a sharding rule. Every rule indicates a sharding field and bings a function to implement dynamic sharding algorithm. Take the sharding function org.MyCAT.route.function.PartionByFileMap for example , which bases on integer mapping and is offen used , this function through a configuration file to determined the mapping relation. Take the following sharding-by-intfile sharding rule for example:

<tableRule name="sharding-by-intfile">

<rule>

<columns>sharding\_id</columns>

<algorithm>hash-int</algorithm>

</rule>

</tableRule>

<tableRule name="auto-sharding-long">

<rule>

<columns>id</columns>

<algorithm>rang-long</algorithm>

</rule>

</tableRule>

<function name="hash-int" class="org.MyCAT.route.function.PartitionByFileMap">

<property name="mapFile">partition-hash-int.txt</property>

</function>

<function name="rang-long" class="org.MyCAT.route.function.AutoPartitionByLong">

<property name="mapFile">autopartition-long.txt</property>

</function>

The content of partition-hash-int.txt file as follow：

10000=0

10010=1

It means that when sharding\_id field is set equal to 10000, MyCAT will return the sharding node which ID is zero, and so on.

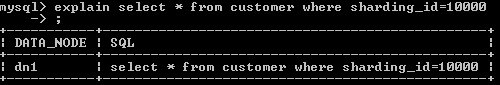
The sharding rule of customer table is the same principle as above in Schema.xml:

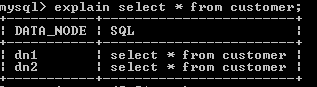
<table name="customer" dataNode="dn1,dn2" rule="sharding-by-intfile" />

So the horizontal sharding of customer table is on sharding\_id field, and the sharding are storaged in two data nodes (dn1 and dn2).

How to konw a SQL was executed in which sharding? Using explain SQL statement:

explain select \* from customer where sharding\_id=10000





Base on your business scenario and data characteristic,you can use the following sharding rules:

auto-sharding-long: Primary key is auto-increment number, and auto-sharding by range. Such the data will be storaged in zeroth sharding node which is between 0 to 2,000,000, in first sharding node which between 2,000,001 to 4,000,000, and so on. Base on the performance of database server, every sharding can be storaged between 100 to 500 of records. And using this way, every sharding table have a independent mechanismof ID auto-increementing to ensure records continuity.

There is a sharding\_id field in the table and its type is integer. The value of sharding\_id corresponds a specific bussiness meaning. Such as 10000 corresponds China Telecom ,10010 corresponds China Unicon. In addition, it is a feasible way to define the value of sharding\_id by encoding provinces.Encoding provinces to integer rather than string for efficiency. The file conf/partition-hash-int.txt defines the mapping relationship that which integer corresponds which sharding id.

mod-long: If We rarely use rang query and only use a field(most likely primary key) to find location for some tables, we can use remainders way to randomly assigned a record into one node.

All of sharding rules are defined in rule.xml. Different tables can be defined different sharding rules depending on the requirement.

For some tables which are non-sharding or same sharding rule, you can use a simplify way to define table rule like :<table name="customer,product" rule="auto-sharding-long" />. For this way, tables which are defined in the name have the same attributes and can’t have childTable element.

Base on E-R Relationshio Sharding Strategy

Traditional database sharding way are based on single table. For table association, this way would be diffcult to handle. Consider the following sharding model , Customer and Orders are sharded in different nodes, parent\_id field of Orders stores primary key of parent table Customer

Dn2(orders)

Parent\_Id:2

Parent\_id :5

Dn1(orders)

Parent\_id:1

Parent\_id:3

Dn3(customer)

Id:5

Id:6

Dn2(customer)

Id:3

Id:4

Dn1(customer)

Id:1

Id:2

orders

customer

Dn3(orders)

Parent\_id :1

Parent\_id:3

Parent\_id:5

In order to execute the union query of Customer and Orders, the data of sharding table must be tranfed on the netword by spanning nodes. In the example above:

When Order table on the DN1 node execute JOIN, it needs DN1 and DN2 nodes.

When Order table on the DN2 node execute JOIN, it needs DN1 and DN3 nodes.

When Order table on the DN3 node execute JOIN, it needs DN1, DN2 and DN3 nodes.

There is not a good method about resolving the current JOIN approach in the industry. It is complex to implement, and the performance of the ways cannot meet the requirements of enterprise application development.

MyCAT leverage the design considerations of rookie Fundation DB in NewSQL field. Foundation DB innovatively propose the concept of Table Group. Storage location of the child tables depend on its primary tables in Fundation DB. The storage location of child tables next to primary tables in physical, so Funddation DB completely resolves the efficiency and performance issues. On the basis of this idea, the author of MyCAT propose the data sharding strategy which bases on E-R relationships, the child table records and its associated parent table records are stored in the same data sharding.

In the example above, schema.xml defines the sharding configurations following:

<table name="customer" dataNode="dn1,dn2" rule="sharding-by-intfile">

<childTable name="orders" joinKey="customer\_id" parentKey="id"/>

</table>

Customer adopt sharding-by-intfile sharding strategy, sharding in dn1 and dn2 nodes. Orders depend on parent table to sharding. Thee relationships between Customer and Orders is Orders.customer\_id = Customer.id. So data sharding and store as follow:

Dn2(orders)

Parent\_Id:3

Parent\_id :4

Dn1(orders)

Parent\_id:1

Parent\_id:2

Dn2(customer)

Id:3

Id:4

Dn1(customer)

Id:1

Id:2

orders

customer

That way, Customer sharding in Dn1 and Orders sharding in Dn1 can execute part JOIN union, so do in Dn2. Then merge the data of two data nodes for finishing overall JOIN. Think of all the fun, in every sharding there is one million records in Orders tables, so ten million for 10 sharding. The data sharding pattern bases on E-R mapping basically solves the problem which more than 80% enterprise application.

How to deal with many-to-many tables? There is some cases as follow for many-to-many tables generally：

Primary table + relationship table + dictionary table

Primary table + relationship table + primary table B

For the first case, dictionary table can be defined as “global table”, and its records size ranged from thousands to hundreds of thousands. Dictionary table change less basically and real-time synchronized to all the sharding by MyCAT automatically. So the three tables can be executed JOIN operation.

For the second case, we should konw that relationship table prefer to which primary table from a business view. That’s “relationship of A” or “relationship of B” to determine the relationship table stored following A or B. At present we does not support the three table union in this pattern temporarily. We will consider the bi-directional replication of the midle table in the future version to complement the bi-directional union query between A-relationship table and B-relationship.

About the implementation of global table, When inserting or updating the global, MyCAT will automatically executes the same operation in all data nodes defined in the global table to guarantee the consistent between all data nodes. Because of this features, global table can be executed JOIN operation with any table no matter sharding table or non-sharding table. Table that update data infrequently and is not larger in scale(within million ) can be defined as global to achieve the goals which trading storage for performance.

Primary Key Sharding VS Non-Primary Key Sharding

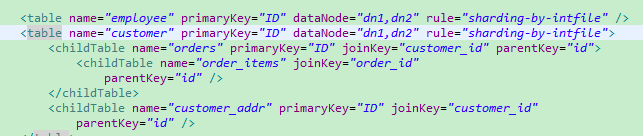
It’s not difficult for using primary key sharding or non-primary key sharding. When you haven’t any field as sharding field, primary key sharding is your only option. The fastest query on primary key is its advantage. When you use auto-incrementing serial number as primary key, the data can be sharded in different nodes more equably.

If there is a fit business field fit for being a sharding field, we support you to use this bussiness field to sharding. The conditions follow for choose a sharding field:

As even as possible distributing all of the data into every nodes.

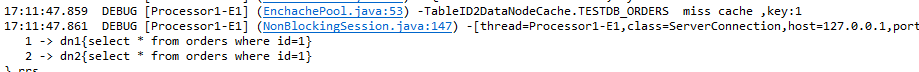
This business field is the most frequent and important query condition.

Excepting primary key, the common sharding fields are that order creation time, shop type, shop province and so on. When you find a fit business field as shardind field, you don’t have to go through losing the query performance on primary key. Because in this case, MyCAT provices the memory cache mechanism from primary key to sharding. The hot data will be query by primary key, and the performance hasn’t suffered. By doing this:



For non-primary key table, Using the properties primaryKey, at the moment MyCAT will analyze the SQL statement result which first executed by primary key to confirm a primary key of this table is on which sharding. And caching primary key to sharing ID. Take the following SQL statement as an example, because id is not the sharding field for orders, this SQL statement will be send to all of the shardings executing:

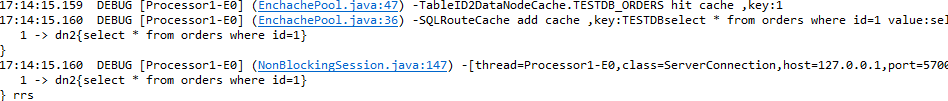
select \* from orders where id=1;



After executed the SQL statement:



MyCAT puts a message into cache TableID2DataNodeCache.TESTDB\_ORDERS that primary key as key and sharding ID as value. When we execute SQL statement above again, MyCAT will send the SQL to dn2.



For multi-primary key query, MyCAT also can automatic optimize. Such as the statement--Select \* from orders where id in(1,2,3), MyCAT will store respectively the relationship into the cache from 1,2,3 primary keys to sharding.

Imagining that, there is fifty million records in every table, hot data ten percent of the records is offen query by primary key. Then 50,000,000 \* 10% = 5,000,000 , it’s need about 1.5G memory space for caching those message. We can tuned this memory the most precise by analyzing the using information of the cache. Executing the command -- show @@cache, it can shows use cases of current cache by analyzing the managemeng port 9066 of MyCAT.



For more content, you can reference MyCAT Performance Tuning Handbook.

**SQL 99 Specification**

MyCAT supports SQL99 Specification, including DDL. Some of sql grammars in MySQL is not supported, however, MyCAT1.2 has made it with MySQL comments.

An example is below.

/\*!mycat select id from travelrecord where id=2\*/ select \* from travelrecord where id=2;

The part which starts with /\*!mycat xxxxx\*/ in the SQL above, is comment of mycat. A SQL meets SQL99 and replaced with xxxx, is used to notify MyCAT that it should be parsed, analyzed, and routed; MyCAT will send the real SQL after the comment to the routed database to execute and receive the result.

For example, MyCAT does not support the sql grammar such as select into, meanwhile, the table to be inserted is not sharding, so the SQL could be written as following:

/\*!mycat insert into B\*/ select \* from A insert into B

NOTE: The grammar of comment in newest version has been changed as below:

/\*!mycat: sql = select id from travelrecord where id=2\*/ select \* from travelrecord where id=2;

GLOBAL TABLES IN MyCAT

Generally, there are many tables as dictionaries in a real scenario. These tables should be considered as LABELS for business tables, but not as primary tables for them. These tables are very seldom changed, so they could be cached based on their primary key.

The following diagram illustrates a typical "label Relations".

**Network device**

Due to business growth, the relation between sharding business tables and dictionary tables, become thorny issues. Considering the following features of dictionary tables:

Changed infrequently

Data quantity is not big

Data scales is not large, there is little more than a few handrand thousand records

So, MyCAT defines a special table, named “global table”, which has following features:

To insert, update, delete global tables, are executed on all nodes immediately, keeping data are consistency on each shard.

To select records of global tables, queries on node only

Global tables could be joined with any other tables

To define dictionary tables as global ones resolves the joining problems. Through global tables and sharding policy based on E-R relation, MyCAT can resolve most of developing problems with data sharding on RDBMS.

Global table configuration is easy. It need not sharding rule. You just add config as following:

<table name="company" primaryKey="ID" type="global" dataNode="dn1,dn2,dn3" />

Note that, DDLs of each global tables must be executed on each DB node.

HA and Read/Write splitting

Read/Write splitting policy of MyCAT are following:

All SQLs will be executed on writing node, in a transaction, except sqlect SQLs which starts with /\*balance\*/

Auto commited select SQL will be executed on reading nodes, with random balancing among them

If one of masters is offline, all of its slaves is unavailable, because the new data could not be updated to reading nodes. MyCAT will load balance in using of another cluster with master/slaves.

If all masters are unavailable, all auto committed select SQL will be sent to alive reading nodes. In the situation, most query accesses could receive datas, but insert/update/delete will fail.

following is configuration of read/write splitting



the attribute named “balance” of dataHost above has following values:

0.close read/write splitting

1.all readHosts and standing by writeHosts will load balance of select SQL. In the other hand, in double masters and double slaves mode(M1->S1, M2->S2), M2, S1, S2 will load balance of select SQL in common situations.

2.all readHosts and writeHosts will load balance select SQLs. In the other words, all nodes will load balance, if there are not too many accesses.

One dataHost element in config file, maps a group of database hosts with data synchronization. DBA should guarantee this. A writeHost maps a master of database server, meanwhile, each sub-node named readhost maps a slave database server which is synchronized with the master. When several writeHosts are configured, MyCAT could detect the offlined ones, and switch next available writeHost.

MyCAT supports HA, according to your application features, it could be configured as following:

To configure database as one-master and multi-slaves, open read/write splitting

To configure database as double-masters and double-slaves(multi-slaves), open read/write splitting

To configure database as multi-masters and multi-slaves, open read/write spliting

The last two configurations, are with higher availabilities, if one of masters is failover, MyCAT will detect by heartbeating, and switch to another master node. In any time, MyCAT will write to one node only.

Following is the typical configuration of double-masters and double-slaves:

MyCAT

)

M2(writeHost)

)

M1(writeHost)

)

S2(readHost)

)

S1(readHost)

)

To set log level is debug in log4j.xml, MyCAT will output following logs.

16:37:21.660 DEBUG [Processor0-E3] (PhysicalDBPool.java:333) -select read source hostM1 for dataHost:localhost1

16:37:21.662 DEBUG [Processor0-E3] (PhysicalDBPool.java:333) -select read source hostM1 for dataHost:localhost1

Based on this information, each SQL could be determined which node is sending to, thus you can determine whether there has separation of read and write.

MYCAT SEQUENCE

MyCAT sequence is a new feature of MyCAT, to ensure the primary key is unique in global with sharding. However, the auto increment feature of MySQL cannot achieve this. MyCAT sequence grammar meets SQL standard specification. You can use it like this:

next value for MYCATSEQ\_GLOBAL

MYCATSEQ\_GLOBAL is sequence name. MyCAT will create new sequence, to simplify the development complexity. In addition, MyCAT also provides a global sequence, named: MYCATSEQ\_GLOBAL.

NOTE that, MYCATSEQ\_ must be UPPER CASE.

NOTE again, in practice, it is suggested that to create a sequence for each table, and sequence name is like MYCATSEQ\_TableName\_ID\_SEQ.

sequence instraction in SQL

Sequence identifier is like MYCATSEQ\_XXX. Examples are following.

To use default global sequence:

insert into tb1(id,name) values(next value for MYCATSEQ\_GLOBAL,'micmiu.com');

To use custom sequence:

insert into tb2(id,name) values(next value for MYCATSEQ\_MY1,'micmiu.com');

To get a new value of sequence:

Select next value for MYCATSEQ\_XXX

MyCAT has provided a local config implementation. The following is configuration instruction.

Config file: sequence\_conf.properties

Config content:

XXX.HISIDS= 1-100,501-800,3001-5000 //used section in history

XXX.MINID=10001 //min value of current section

XXX.MAXID=20000 //max value of current section

XXX.CURID=10000 //current value in current section

Example of global sequence config:

GLOBAL.HISIDS=

GLOBAL.MINID=1

GLOBAL.MAXID=50000

GLOBAL.CURID=10000

Example of custom sequence config:

MY1.HISIDS=

MY1.MINID=101

MY1.MAXID=200

MY1.CURID=152

To config to store sequence in MySQL:  
Edit Serfver.xml, and add this config item   
<property name="sequnceHandlerType">1</property>  
• To create table and function about sequence in one data node. Sqls are stored in $MYCAT\_HOME/doc/sequence-sql.txt, and they should be executed on mysql, but not on MyCAT.   
• $MYCAT\_HOME/conf/quence\_db\_conf.properties contains configuration info of storing sequence database.   
#sequence stored in datanode  
GLOBAL=dn1  
COMPANY=dn1  
CUSTOMER=dn1  
• In sequence table, to insert a sequence record, confirm its initial value and step size. The suggestion is that step size is in a suitable range, such as 50-500. It should be executed on MySQL, but not on MyCAT. For example as following:  
INSERT INTO MYCAT\_SEQUENCE VALUES ('GLOBAL', 0, 100);  
• To update sequence current value as a new one, it should be executed on MySQL, but not on MyCAT.

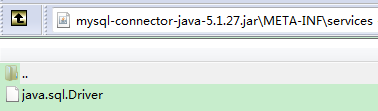
• To select sequence values with sql as following:  
SELECT mycat\_seq\_curval('GLOBAL');

NOTE that, step size depends on your TPS of data insertion. If it is 1000 per second, the step size is 1000\*10=10,000. A batch of sequence values will be selected every 10 seconds. It’s not so large.

To support other databases on JDBC

From MyCATv1.2, it has supported to connect other databases besides MySQL in a common way, such as Oracle, SQL server, DB2, and so on. MyCAT is still a MySQL server to client.

To config it as following, at first to place database driver jar which meets JDBC4 under $MYCAT\_HOME/lib, and check the file which path is META-INF/services/java.sql.Driver in the driver jar file.



The file content is the class name of driver:



To config dataHost in schema.xml as following:

<dataHost name="jdbchost" maxCon="1000" minCon="10" balance="0"

dbType="mysql" dbDriver="jdbc">

<heartbeat>select user()</heartbeat>

<writeHost host="hostM1" url="jdbc:mysql://localhost:3306"

user="root" password="123456">

</writeHost>

</dataHost>

CONFIGURATION OF MyCAT

System arguments and user permissions of MyCAT could be defined in server.xml. Currently, MyCAT supports two kinds of permissions such as readonly and read/write.

Following is the configuration:

<user name="test">

<property name="password">test</property>

<property name="schemas">TESTDB</property>

<property name="readOnly">true</property>

</user>

**SQL intercept**

SQL interception is an useful advanced technique. You can write a java class which can rewrite the SQL incoming to MyCAT and then handed to Mycat to perform. This technique can be shown as some special features:

• Capture and record some special SQL

• In order to optimize the performance, rewrite the SQL, such as changing the order of query conditions or adding a pagination

• set some “Select SQL” forced to “Read” mode, make reading and writing separated (It’s not easy to separate the “Select SQL” from a transaction in a lot of transaction framework.

• Other

The usage is to configure the system parameters in Server.xml, specify your own SQL interceptors with Java implementation class:

<system>

<property name = "sqlInterceptor"> org.opencloudb.interceptor.impl.DefaultSqlInterceptor </ property>

</ system>

The default interceptor implements the filter and conversion of Mysql escape-character, SQL interceptor implementation is very simple:

/ \*\*

\* Escape mysql escape letter

\* /

Override

public String interceptSQL (String sql, int sqlType) {

if (sqlType == ServerParse.UPDATE || sqlType == ServerParse.INSERT || sqlType == ServerParse.SELECT || sqlType == ServerParse.DELETE) {

return sql.replace ("\\ '", "' '");

} Else {

return sql;

}

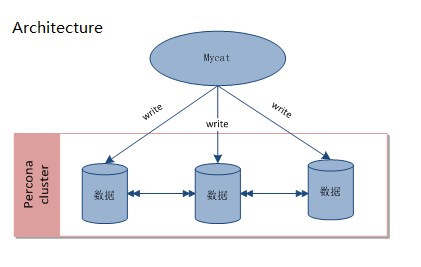
}

**MySQL High Ability**

HAProxy or MyCAT Cluster can play the load-balancing role. Behind them, there could be a group of MySQL database backend. MyCAT Server could have the following two options:

• MYSQL master-slave replication, when the primary node fails, automatically switches to the slave node for writing.

• galera for mysql cluster, percona-cluster or mariadb cluster



MyCAT and percona-cluster: configuration of schema.xml is as follows:

<dataHost name = "localhost1" maxCon = "1000" minCon = "10" balance = "0"

writeType = "1" dbType = "mysql" dbDriver = "native">

<heartbeat> select user () </ heartbeat>

<writeHost host = "hostM1" url = "localhost: 3306" user = "root" password = "123456" />

<writeHost host = "hostM2" url = "localhost: 3317" user = "root" password = "123456" />

<writeHost host = "hostM3" url = "localhost: 3319" user = "root" password = "123456" />

</ dataHost>

writeHost write all percona-cluster nodes, while writeType is set to 1. In this mode, there is no readHost.

When any writeHost has failure, it will automatically be excluded. After restoration, it join the write nodes automatically.

**Quick start**

Mycat is developed by java, it needs Java runtime environment, if you have no java runtime installed in your computer, you need to download jdk and install it first.

<http://www.java.com/zh_CN/>

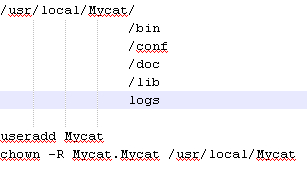
to obtain the latest mycat open source version in the project web site

[http://code.google.com/p/MyCAT/](http://code.google.com/p/opencloudb/)

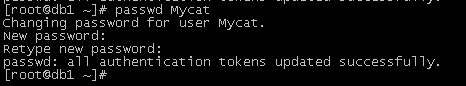
Now the latest code is located in taobao code repository temporarily, you can download the binary distribution here:

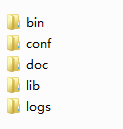
<http://code.taobao.org/svn/openclouddb/downloads>

when using windows operating system, download Mycat-server-xxxx.ZIP and on linux platform, download tar.gz and decompress it to a certain directory, be attention, there should be no blank in the directory path, it’s recommended decompress it to the directory /usr/local/Mycat, such as:



Change the password of user Mycat





The uses of the directories:

bin: the program directory, here stores the windows version and linux version, it provides the version which is packaged as a service, and nowrap’s shell commands, in order to select and change

the version you want to use, switch to directory bin:

• on windows platform: execute mycat.bat console to start the console program, also you can set it as a service, if the program has error too, you can execute startup\_nowrap.bat to ensure the java command can be executed in the command.

• On linux platform: mycat console, first chmod + x mycat

The wrap style command can be installed as a service, you can start it and stop it.

• mycat install (可选)

• mycat start

Be attention, in the wrap style, the configuration params is in the file conf/wrap.conf, you can modify the params refer to [**http://wrapper.tanukisoftware.com/doc/english/properties.html**](http://wrapper.tanukisoftware.com/doc/english/properties.html)**,** hereis the example:

**# Java Additional Parameters**

**wrapper.java.additional.5=-XX:MaxDirectMemorySize=2G**

**wrapper.java.additional.6=-Dcom.sun.management.jmxremote**

**# Initial Java Heap Size (in MB)**

**wrapper.java.initmemory=2048**

**# Maximum Java Heap Size (in MB)**

**wrapper.java.maxmemory=2048**

if lack of memory error reported, modify the memory param value smaller, 1G or 500M.

In the conf directory stores the configuration files, server.xml is used to configure server parameters and user authorization. Schema.xml is used to define the logic database、table and sharding, rule.xml is used to configure the sharding rule, some detail parameter info of the sharding rule is configured in a file independently which is stored in the same directory, when the configure file is modified, mycat needs restarting, or reload on port 9066.

Log is stored in logs/mycat.log, one file per day, the configuration file of the log is conf/log4j.xml, according to your requirement, you can set the log level as debug level, then more information will be written out, helping problem checking.

It’s recommended that a local server is installed, if not, install one, the file download address is <http://dev.mysql.com/downloads/mysql/5.5.html#downloads>

start mycat server, in order to access the data, the command mysql\bin\mysql.exe should be set into PATH environment for convenience.

Use command tool or gui client to access mysql and create three sharding database for demo.

CREATE database db1;

CREATE database db2;

CREATE database db3;

Be attention: if mysql is installed on linux platform, you need to set mysql case insensitive, in case of table not found error.

In the mysql configuration my.ini [mysqld] section, add one line:

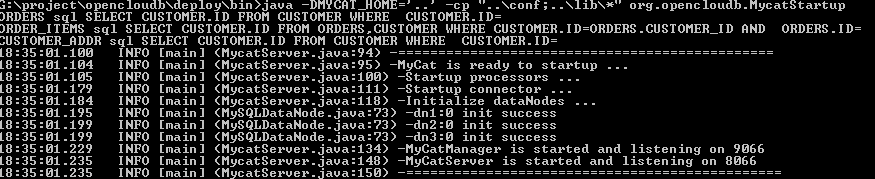
lower\_case\_table\_names=1

edit MYCAT\_HOME/conf/schema.xml file, modify the connection information corresponding to dataHost:



Be attention, the values of location, user, password attributes should be set as your own configuration

When modification finished, switched into MYCAT\_HOME/bin directory, execute the start command: startup.bat, when the command successfully executed, the information will be displayed as below:



Be attention, the default data port is 8066, admin port is 9066.

GUI client is also available, such as mysql workbench, navicat and some other database client based on java. Be attention, set the port as 8066, database as TESTDB.

Execute the command line commnad: mysql -utest -ptest -h127.0.0.1 -P8066 –DTESTDB to access OpenCoundDB, the following operations are executed in the command line environment(JDBC set the port 3306 to 8066 in the mysql url).

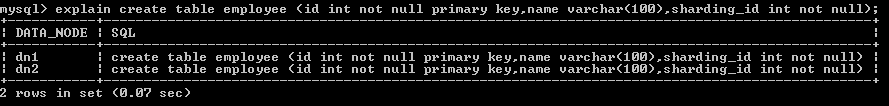
A small tip: the mycat user account and authorization information is configured in the file conf/server.xml, and the username and password used to access endpoint mysql server is in the file conf/schema.xml, they are two separate systems, and mycat logic database(schema), logic table(table) is similar too.

Table Employee, which is according to sharding-by-intfile(the sharding filed is sharding\_id) to shard. create table employee: input the follow sql

create table employee (id int not null primary key,name varchar(100),sharding\_id int not null);

execute explain command, check which shard node the sql is sent to:

explaincreate table employee (id int not null primary key,name varchar(100),sharding\_id int not null);



It’s recommended that refer to the employee table definition in the schema.xml file and sharding rule to check what data is on the node dn1 and what data is on the node dn2.

Warm tip: command explain can be used to check any correct sql, the effect is to tell you on which node the sql is sent to execute, it’s very useful to check sharding related problems, besides, explain can be executed safely many times, it just tells you the sql’s routing shard, not executes actually.

Insert data below:

insert into employee(id,name,sharding\_id) values(1,'leader us',10000);

insert into employee(id,name,sharding\_id) values(2, 'me',10010);

insert into employee(id,name,sharding\_id) values(3, 'mycat',10000);

insert into employee(id,name,sharding\_id) values(4, 'mydog',10010);

table company is sharded by the rule auto-sharding-long(primary key range). Create the following sql:

create table company(id int not null primary key,name varchar(100));

insert data:

insert into company(id,name) values(1,'hp');

insert into company(id,name) values(2,'ibm');

insert into company(id,name) values(3,'oracle');

you will find three data inserted into all of three shading, for employee is defined as a global table, use explain commnad to check the scene.

explain insert into company(id,name) values(1,'hp')

the 3 returned node information is:

| DATA\_NODE | SQL |

+-----------+---------------------------------------------+

| dn1 | insert into company(id,name) values(1,'hp') |

| dn2 | insert into company(id,name) values(1,'hp') |

| dn3 | insert into company(id,name) values(1,'hp') |

+-----------+---------------------------------------------+

Create table customer:

create customer: create table customer(id int not null primary key,name varchar(100),company\_id int not null,sharding\_id int not null);

Insert data:

insert into customer (id,name,company\_id,sharding\_id )values(1,'wang',1,10000); //stored in db1;

insert into customer (id,name,company\_id,sharding\_id )values(2,'xue',2,10010); //stored in db2;

insert into customer (id,name,company\_id,sharding\_id )values(3,'feng',3,10000); //stored in db1;

Query results:

Select \* from customer;

explain Select \* from customer; (make sure the data is stored as sharding)

create table orders, and insert data into orders:

create table orders (id int not null primary key ,customer\_id int not null,sataus int ,note varchar(100) );

insert into orders(id,customer\_id) values(1,1); //stored in db1 because customer table with id=1 stored in db1

insert into orders(id,customer\_id) values(2,2); //stored in db2 because customer table with id=1 stored in db2

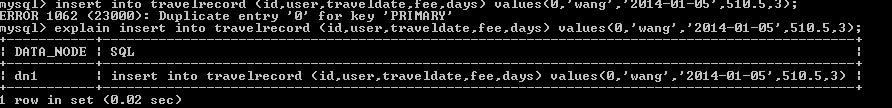
explain insert into orders(id,customer\_id) values(2,2);

select customer.name ,orders.\* from customer ,orders where customer.id=orders.customer\_id;

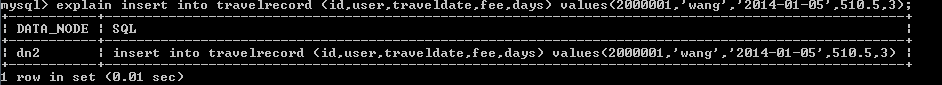
travelrecord is sharded according to the primary key id range:

create travelrecord: create table travelrecord (id bigint not null primary key,user\_id varchar(100),traveldate DATE, fee decimal,days int);

insert into travelrecord (id,user\_id,traveldate,fee,days) values(1,'wang','2014-01-05',510.5,3);



explain insert into travelrecord (id,user\_id,traveldate,fee,days) values(2000001,'wang','2014-01-05',510.5,3); the id field is stored in two shardings.



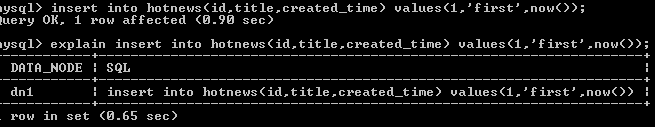
We find it supports cross sharding join.

Hot news, send data to dn1, dn2, dn3 randomly by performing a modulus on hot news id.

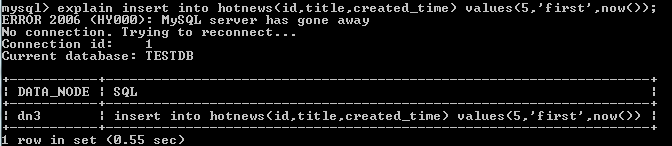
create table hotnews(id int not null primary key ,title varchar(400) ,created\_time datetime);

insert data

insert into hotnews(id,title,created\_time) values(1,'first',now()); on sharding 1



And if id=5, then sent to dn3, 5%3=2, equals to the index of dn3.



Othes :

Table goods

create table goods(id int not null primary key,name varchar(200),good\_type tinyint,good\_img\_url varchar(200),good\_created date,good\_desc varchar(500), price double);

let’s discovery the wonderful new world of MyCat! QQ group: 106088787.