MaxScale

Configuration & Usage Scenarios

Mark Riddoch
Last Updated: 24th July 2013

Contents

Document History

Date	Change	Who
21st July 2013	Initial version	Mark Riddoch
23rd July 2013	Addition of default user and password for a monitor and discussion of monitor user requirements New monitor documented for Galera clusters Addition of example Galera cluster configuration	Mark Riddoch

Introduction

The purpose of this document is to describe how to configure MaxScale and to discuss some possible usage scenarios for MaxScale. MaxScale is designed with flexibility in mind, and consists of an event processing core with various support functions and plugin modules that tailor the behaviour of the MaxScale itself.

Terms

Term	Description
service	A service represents a set of databases with a specific access mechanism that is offered to clients of MaxScale. The access mechanism defines the algorithm that MaxScale will use to direct particular requests to the individual databases.
server	A server represents an individual database server to which a clien can be connected via MaxScale.
router	A router is a module within MaxScale that will route client request to the various database server which MaxScale provides a service interface to.
connection routing	Connection routing is a method of handling requests in which MaxScale will accept connections from a client and route data on that connection to a single database using a single connection. Connection based routing will not examine individual quests on a connection and it will not move that connection once it is established.

statement routing	Statement routing is a method of handling requests in which each request within a connection will be handled individually. Requests may be sent to one or more servers and connections may be dynamically added or removed from the session.
protocol	A protocol is a module of software that is used to communicate with another software entity within the system. MaxScale supports the dynamic loading of protocol modules to allow for increased flexibility.
module	A module is a separate code entity that may be loaded dynamically into MaxScale to increase the available functionality. Modules are implemented as run-time loadable shared objects.
monitor	A monitor is a module that can be executed within MaxScale to monitor the state of a set of database. The use of an internal monitor is optional, monitoring may be performed externally to MaxScale.
listener	A listener is the network endpoint that is used to listen for connections to MaxScale from the client applications. A listener is associated to a single service, however a service may have many listeners.
connection failover	When a connection currently being used between MaxScale and the database server fails a replacement will be automatically created to another server by MacScale without client intervention
backend database	A term used to refer to a database that sits behind MaxScale and is accessed by applications via MaxScale.

Configuration

The MaxScale configuration is read from a file which can be located in a number of placing, MaxScale will search for the configuration file in a number of locations.

1. If the environment variable MAXSCALE_HOME is set then MaxScale will look for a configuration file called MaxScale.cnf in the directory $MAXSCALE_HOME/etc
2. If MAXSCALE_HOME is not set or the configuration file is not in the location above MaxScale will look for a file in /etc/MaxScale.cnf

Alternatively MaxScale can be started with the -c flag and the path of an explicit configuration file to load.

The configuration file itself is based on the “ini” file format and consists of various sections that

are used to build the configuration, these sections define services, servers, listeners, monitors and global settings.

Global Settings

The global settings, in a section named [MaxScale]], allow various parameters that affect MaxScale as a whole to be tuned. Currently the only setting that is supported is the number of threads to use to handle the network traffic. MaxScale will also accept the section name of [gateway] for global settings. This is for backward compatibility with versions prior to the naming of MaxScale.

Threads

To control the number of threads that poll for network traffic set the parameter threads to a number. It is recommended that you start with a single thread and add more as you find the performance is not satisfactory. MaxScale is implemented to be very thread efficient, so a small number of threads is usually adequate to support reasonably heavy workloads. Adding more threads may not improve performance and can consume resources needlessly.

```
# Valid options are:
#       threads=<number of epoll threads>
[MaxScale]
threads=1
```

Service

A service represents the database service that MaxScale offers to the clients. In general a service consists of a set of backend database servers and a routing algorithm that determines how MaxScale decides to send statements or route connections to those backend servers.

Several different services may be defined using the same set of backend servers. For example a connection based routing service might be used by clients that already performed internal read/write splitting, whilst a different statement based router may be used by clients that are not written with this functionality in place. Both sets of applications could access the same data in the same databases.

A service is identified by a service name, which is the name of the configuration file section and a type parameter of service

```
[Test Service]
type=service
```

Router

The router parameter of a service defines the name of the router module that will be used to

implement the routing algorithm between the client of MaxScale and the backend databases. Additionally routers may also be passed a comma separated list of options that are used to control the behaviour of the routing algorithm. The two parameters that control the choice of routing are router and router_options.

```
router=readconnroute
router_options=slave
```

To change the router to consider master as well as slave servers for connection distribution would merely require a change of router options.

```
router=radconnroute
router_options=master,slave
```

Servers

The servers parameter in a service definition provides a comma separated list of the backend servers that comprise the service. The server names are those used in the name section of a block with a type parameter of server (see below).

```
servers=server1,server2,server3
```

User

The users parameter, along with the auth parameter are used to define the credentials used to connect to the backend servers to extract the list of database users from the backend database that is used for the client authentication.

```
user=maxscale
auth=maxpassword
```

Auth

The auth parameter may be either a plain text password or it may be an encrypted password. Ses the second on encrypting passwords for use in the MaxScale.cnf file. This user must be capable of connecting to the backend database and executing the SQL statement “SELECT user, password FROM mysql.user”.

Server

Server sections are used to define the backend database servers that can be formed into a service. A server may be a member of one or more services within MaxScale. Servers are identified by a server name which is the section name in the configuration file. Servers have a type parameter of server, plus address port and protocol parameters.

```
[server1]
```

```
type=server
address=127.0.0.1
port=3000
protocol=MySQLBackend
```

Address

The IP address or hostname of the machine running the database server that is being defined.

Port

The port on which the database listens for incoming connections.

Protocol

The name for the protocol module to use to connect MaxScale to the database. Currently only one backend protocol is supported, the MySQLBackend module.

Monuser

The monitor has a username and password that is used to connect to all servers for monitoring purposes, this may be overridden by supplying a monuser statement for each individual server

```
monuser=mymonitoruser
```

MonPasswd

The monitor has a username and password that is used to connect to all servers for monitoring purposes, this may be overridden by supplying a monpasswd statement for the individual servers

```
monpasswd=mymonitorpasswd
```

The monpasswd parameter may be either a plain text password or it may be an encrypted password. Ses the second on encrypting passwords for use in the MaxScale.cnf file.

Listener

The listener defines a port and protocol pair that is used to listen for connections to a service. A service may have multiple listeners associated with it, either to support multiple protocols or multiple ports. As with other elements of the configuration the section name is the listener name and a type parameter is used to identify the section as a listener definition.

```
[Test Listener]
type=listener
service=Test Service
protocol=MySQLClient
port=4008
```

Service

The service to which the listener is associated. This is the name of a service that is defined elsewhere in the configuration file.

Protocol

The name of the protocol module that is used for the communication between the client and MaxScale itself.

Port

The port to use to listen for connections incoming connections.

Monitor

In order for the various router modules to function correctly they require information about the state of the servers that are part of the service they provide. MaxScale has the ability to internally monitored the state of the back-end database servers or that state may be feed into MaxScale from external monitoring systems.

Monitors are defined in much the same way as other elements in the configuration file, with the section name being the name of the monitor instance and the type being set to monitor.

```
[MySQL Monitor]
type=monitor
module=mysqlmon
servers=server1,server2,server3
user=raatikka
passwd=vilho
```

Module

The module parameter defines the name of the loadable module that implements the monitor. This module is loaded and executed on a separate thread within MaxScale.

Servers

The servers parameter is a comma separated list of server names to monitor, these are the names defined elsewhere in the configuration file. The set of servers monitored by a single monitor need not be the same as the set of servers used within any particular server, a single monitor instance may monitor servers in multiple servers.

User

The user parameter defines the username that the monitor will use to connect to the monitored database. Depending on the monitoring module used this user will require specific privileges in order to determine the state of the nodes, details of those privileges can be found in the sections on each of the monitor modules.

Individual servers may define override values for the user and password the monitor uses by setting the monuser and monpasswd parameters in the server section.

Passwd

The password parameter may be either a plain text password or it may be an encrypted password. Ses the second on encrypting passwords for use in the MaxScale.cnf file.

Protocol Modules

The protocol supported by MaxScale are implemented as external modules that are loaded dynamically into the MaxScale core. These modules reside in the directory $MAXSCALE_HOME/module, if the environment variable $MAXSCALE_HOME is not set it defaults to /usr/local/skysql/MaxScale.

MySQLClient

This is the implementation of the MySQL protocol that is used by clients of MaxScale to connect to the gateway.

MySQLBackend

The MySQLBackend protocol module is the implementation of the protocol that MaxScale uses to connect to the backend MySQL, MariaDB and Percona Server databases. This implementation is tailored for the MaxScale to MySQL Database traffic and is not a general purpose implementation of the MySQL protocol.

Telnetd

The telnetd protocol module is used for connections to MaxScale itself for the purposes of creating interactive user sessions with the MaxScale instance itself. Currently this is used in conjunction with a special router implementation, the debugcli.

HTTPD

This protocol module is currently still under development, it provides a means to crete HTTP connections to MaxScale for use by web browser or RESTful API clients.

Router Modules

The main task of MaxScale is to accept database connections from client applications and route the connections or the statements sent over those connections to the various services supported by MaxScale.

There are two flavours of routing that MaxScale can perform, connection based routing and statement based routine. These each have their own characteristics and costs associated with them.

Connection Based Routing

Connection based routing is a mechanism by which MaxScale will, for each incoming connection decide on an appropriate outbound server and will forward all statements to that server without examining the internals of the statement. Once an inbound connection is associated to a particular backend database it will remain connected to that server until the connection is closed or the server fails.

Statement Based Routing

Statement based routing is somewhat different, the routing modules examine every statement the client sends and determines, on a per statement basis, which of the set of backend servers in the service is best to execute the statement. This gives better dynamic balancing of the load within the cluster but comes at a cost. The query router must understand the statement that is being routing and will typically need to parse the statement in order to achieve this. This parsing within the router adds a significant overhead to the cost of routing and makes this type of router only really suitable for loads in which the gains outweigh this added cost.

Available Routing Modules

Currently a small number of query routers are available, these are in different stages of completion and offer different facilities.

Readconnroute

This is a statement based query router that was originally targeted at environments in which the clients already performed splitting of read and write queries into separate connections. Whenever a new connection is received the router will examine the state of all the server that form part of the service and route the connection to the server with least connections currently. This results is a balancing of the connections, however different connections may have different lifetimes and the connections may become unbalanced.

The readconnroute router can be configured to balance the connections from the clients across all the backend servers that are running, just those backend servers that are currently replication slaves or those that are replication masters. These options are configurable via the router_options that can be set within a service. The router_option strings supported are “master”, “slave” and “joined”.

Master/Slave Replication Setup

To setup MaxScale to route connections to evenly between all the current slave servers in a replication cluster a service entry of the form shown below is required.

```
[Read Service]
type=service
router=readconnroute
router_options=slave
servers=server1,server2,server3,server4
user=maxscale
auth=thepasswd
```

With the addition of a listener for this service, which defines the port and protocol that MaxScale uses

```
[Read Listener]
type=listener
service=Read Service
protocol=MySQLClient
port=4006
```

the client can now connect to port 4006 on the host which is running MaxScale. Statements sent using this connection will then be routed to one of the slaves in the server set defined in the Read Service. Exactly which is selected will be determined by balancing the number of connections to each of those whose current state is “slave”.

It is assumed that the client will have a separate connection to the master server, however this can be arranged via MaxScale, allowing MaxScale to manage the determination of which server is master. To do this you would add a second service and listener definition for the master server.

```
[Write Service]
type=service
router=readconnroute
router_options=master
servers=server1,server2,server3,server4
user=maxscale
auth=thepasswd

[Write Listener]
type=listener
service=Write Service
```

```
protocol=MySQLClient
port=4007
```

This allows the clients to direct write requests to port 4007 and read requests to port 4006 of the MaxScale host without the clients needing to understand the configuration of the Master/Slave replication cluster.

Galera Cluster Configuration

Although not primarily designed for a multi-master replication setup it is possible to use the readconnroute in this situation. The readconnroute connection router can be used to balance the connection across a Galera cluster. A special monitor is available that detects if nodes are joined to a Galera Cluster, with the addition of a router option to only route connections to nodes marked as joined. MaxScale can ensure that users are never connected to a node that is not a full cluster member.

```
[Galera Service]
type=service
router=readconnroute
router_option=joined
servers=server1,server2,server3,server4
user=maxscale
auth=thepasswd

[Galera Listener]
type=listener
service=Galera Service
protocol=MySQLClient
port=3336

[Galera Monitor]
type=monitor
module=galeramon
servers=server1,server2,server3,server4
user=galeramon
passwd=galeramon
```

Readwritesplit

The readwritesplit is a statement based router that has been designed for use within Master/Slave replication environments. It examines every statement, parsing it to determine if the statement falls into one of three categories;

- read only statement

- possible write statement
- session modification statement

Each of these three categories has a different action associated with it. Read only statements are sent to a slave server in the replication cluster. Possible write statements, which may include read statements that have an undeterminable side effect, are sent to the current replication master. Statements that modify the session are sent to all the servers, with the result that is generated by the master server being returned to the user.

Session modification statements must be replicated as they affect the future results of read and write operations, so they must be executed on all servers that could execute statements on behalf of this client.

Currently the readwritesplit router module is under development and is limited:

- Only a single slave connection is managed currently, therefore statements can not be balanced across multiple slaves.
- Read statements that use stored procedures and functions are not recognised as having potentially dangerous (write) side effects and so are routed to slave servers, they should be routed to the master.
- Connection failover support has not yet be implemented. Client connections will fail if the master server fails over.

Master/Slave Replication Setup

To setup the readwritesplit connection router in a master/slave failover environment is extremely simple, a service definition is required with the router defined for the service and an associated listener.

```
[Split Service]
type=service
router=readwritesplit
servers=server1,server2,server3,server4
user=maxscale
auth=thepasswd

[Split Listener]
type=listener
service=Split Service
protocol=MySQLClient
port=3336
```

The client would merely connect to port 3336 on the MaxScale host and statements would be directed to the master or slave as appropriate. Determination of the master or slave status may be done via a monitor module within MaxScale or externally. In this latter case the server flags

would need to be set via the MaxScale debug interface, in future versions an API will be available for this purpose.

Debugcli

The debugcli is a special case of a statement based router. Rather than direct the statements at an external data source they are handled internally. These statements are simple text commands and the results are the output of debug commands within MaxScale. The service and listener definitions for a debug cli service only differ from other services in they they require no backend server definitions.

Debug CLI Configuration

The definition of the debug cli service is illustrated below

```
[Debug Service]
type=service
router=debugcli

[Debug Listener]
type=listener
service=Debug Service
protocol=telnetd
port=4442
```

Connections using the telnet protocol to port 4442 of the MaxScale host will result in a new debug CLI session.

Monitor Modules

Monitor modules are used by MaxScale to internally monitor the state of the backend databases in order to set the server flags for each of those servers. The router modules then use these flags to determine if the particular server is suitable for routing connections to particular query classifications to. The monitors are run within separate threads of MaxScale and do not affect the MaxScale performance.

The use of monitors is optional, it is possible to run MaxScale with external monitoring, in which case arrangements must be made for an external entity to set the status of each of the servers that MaxScale can route to.

Mysqlmon

The MySQLMon monitor is a simple router designed for use with MySQL Master/Slave

replication cluster. To execute the mysqlmon monitor an entry as shown below should be added to the MaxScale configuration file.

```
[MySQL Monitor]
type=monitor
module=mysqlmon
servers=server1,server2,server3,server4
```

This will monitor the 4 servers; server1, server2, server3 and server4. It will set the status of running or failed and master or slave for each of the servers.

The monitor uses the username given in the monitor section or the server specific user that is given in the server section to connect to the server. This user must have sufficient permissions on the database to determine the state of replication. The roles that must be granted to this user are REPLICATION SLAVE and REPLICATION CLIENT.

Galeramon

The Galeramon monitor is a simple router designed for use with MySQL Galera cluster. To execute the galeramon monitor an entry as shown below should be added to the MaxScale configuration file.

```
[Galera Monitor]
type=monitor
module=galeramon
servers=server1,server2,server3,server4
```

This will monitor the 4 servers; server1, server2, server3 and server4. It will set the status of running or failed and joined for those servers that reported the Galera JOINED status.

The user that is configured for use with the Galera monitor must have sufficient privileges to select from the information_schema database and GLOBAL_STATUS table within that database.

Encrypting Passwords

Passwords stored in the MaxScale.cnf file may be encrypted for added security. This is done by creation of an encryption key on installation of MaxScale. Encryption keys may be created manually by executing the maxkeys utility with the argument of the filename to store the key.

```
maxkeys $MAXSCALE_HOME/etc/.secrets
```

Changing the encryption key for a MaxScale will invalidate any currently encrypted keys stored in

the MaxScale.cnf file.

Creating Encrypted Passwords

Encrypted passwords are created by executing the maxpasswd command with the password you require to encrypt as an argument. The environment variable MAXSCALE_HOME must be set, or MaxScale must be installed in the default location before maxpasswd can be executed.

```
maxpasswd MaxScalePw001
61DD955512C39A4A8BC4BB1E5F116705
```

The output of the maxpasswd command is a hexadecimal string, this should be inserted into the MaxScale.cnf file in place of the ordinary, plain text, password. MaxScale will determine this as an encrypted password and automatically decrypt it before sending it the database server.

```
[Split Service]
type=service
router=readwritesplit
servers=server1,server2,server3,server4
user=maxscale
auth=61DD955512C39A4A8BC4BB1E5F116705
```