w: www.mariadb.com

e: info@mariadb.com

Getting Started With MariaDB MaxScale

Filters

Mark Riddoch

Last Updated: 19th November 2014

w: www.mariadb.com

e: info@mariadb.com

Contents

w: www.mariadb.com

e: info@mariadb.com

What Are Filters?

The filter mechanism in MaxScale is a means by which processing can be inserted into the flow of requests and responses between the client connection to MaxScale and the MaxScale connection to the backend database servers. The path from the client side of MaxScale out to the actual database servers can be considered a pipeline, filters can then be placed in that pipeline to monitor, modify, copy or block the content that flows through that pipeline.

Types Of Filter

Filters can be divided into a number of categories

- Logging filters
  Logging filters do not in any way alter the statement or results of the statements that are passed through MaxScale. They merely log some information about some or all of the statements and/or result sets.

  Two examples of logging filters are contained within the MaxScale GA, a filter that will log all statements and another that will log only a number of statements, based on the duration of the execution of the query.

- Statement rewriting filters
  Statement rewriting filters modify the statements that are passed through the filter. This allows a filter to be used as a mechanism to alter the statements that are seen by the database, an example of the use of this might be to allow an application to remain unchanged when the underlying database changes or to compensate for the migration from one database schema to another.

  The MaxScale GA includes a filter that can modify statements by use of regular expressions to match statements and replaced that matched text.

- Result set manipulation filters
  A result set manipulation filter is very similar to a statement rewriting but applies to the result set returned rather than the statement executed. An example of this may be obfuscating the values in a column.

  The MaxScale 1.0 GA release does not contain any result set manipulation filters.

- Routing hint filters

w: www.mariadb.com

e: info@mariadb.com

Routing hint filters are filters that embed hints in the request that can be used by the router onto which the query is passed. These hints include suggested destinations as well as metric that may be used by the routing process.

The MaxScale 1.0 GA release does not contain any hint filters.

- Firewall filters
  A firewall filter is a mechanism that allows queries to be blocked within MaxScale before they are sent on to the database server for execution. They allow constructs or individual queries to be intercepted and give a level of access control that is more flexible than the traditional database grant mechanism.

  The 1.0 GA release of MaxScale does not include any firewall filters.

- Pipeline control filters
  A pipeline filter is one that has an affect on how the requests are routed within the internal MaxScale components. The most obvious version of this is the ability to add a “tee” connector in the pipeline, duplicating the request and sending it to a second MaxScale service for processing.

  The MaxScale 1.0 GA release contains an implementation of a tee filter that allows statements to be matched using a regular expression and passed to a second service within MaxScale.

Filter Definition

Filters are defined in the configuration file, MaxScale.ini, using a section for each filter instance. The content of the filter sections in the configuration file various from filter to filter, however there are always to entries present for every filter, the type and module.

```
[MyFilter]
type=filter
module=xxxfilter
```

The type is used by the configuration manager within MaxScale to determine what this section is defining and the module is the name of the plugin that implements the filter.

w: www.mariadb.com

e: info@mariadb.com

When a filter is used within a service in MaxScale the entry filters= is added to the service definition in the ini file section for the service. Multiple filters can be defined using a syntax akin to the Linux shell pipe syntax.

```
[Split Service]
type=service
router=readwritesplit
servers=dbserver1,dbserver2,dbserver3,dbserver4
user=massi
passwd=6628C50E07CCE1F0392EDEEB9D1203F3
filters=hints | top10
```

The names used in the filters= parameter are the names of the filter definition sections in the ini file. The same filter definition can be used in multiple services and the same filter module can have multiple instances, each with its own section in the ini file.

Filter Examples

The filters that are bundled with the MaxScale 1.0 GA release are documented separately, in this section a short overview of how these might be used for some simple tasks will be discussed. These are just examples of how these filters might be used, other filters may also be easily added that will enhance the MaxScale functionality still further.

Log The 30 Longest Running Queries

The top filter can be used to measure the execution time of every statement within a connection and log the details of the longest running statements.

The first thing to do is to define a filter entry in the ini file for the top filter. In this case we will call it “top30”. The type is filter and the module that implements the filter is called topfilter.

```
[top30]
type=filter
module=topfilter
count=30
filebase=/var/log/DBSessions/top30
```

In the definition above we have defined two filter specific parameters, the count of the number of statement to be logged and a filebase that is used to define where to log the information. This filename is a stem to which a session id is added for each new connection that uses the filter.

w: www.mariadb.com

e: info@mariadb.com

The filter keeps track of every statement that is executed, monitors the time it takes for a response to come back and uses this as the measure of execution time for the statement. If the time is longer than the other statements that have been recorded, then this is added to the ordered list within the filter. Once 30 statements have been recorded those statements that have been recorded with the least time are discarded from the list. The result is that at any time the filter has a list of the 30 longest running statements in each session.

It is possible to see what is in the current list by using the maxadmin tool to view the state of the filter by looking at the session data. First you need to find the session id for the session of interest, this can be done using commands such as list sessions. You can then use the show session command to see the details for a particular session.

```
MaxScale> show session 0x736680
Session 0x736680
       State:              Session ready for routing
       Service:            Split Service (0x719f60)
       Client DCB:         0x7361a0
       Client Address:     127.0.0.1
       Connected:          Thu Jun 26 10:10:44 2014
       Filter: top30
              Report size               30
              Logging to file /var/log/DBSessions/top30.1.
              Current Top 30:
              1 place:
                     Execution time: 23.826 seconds
                     SQL: select sum(salary), year(from_date) from salaries s,
(select distinct year(from_date) as y1 from salaries) y where (makedate(y.y1,
1) between s.from_date and s.to_date) group by y.y1 ("1988-08-01?
              2 place:
                     Execution time: 5.251 seconds
                     SQL: select d.dept_name as "Department", y.y1 as "Year",
count(*) as "Count" from departments d, dept_emp de, (select distinct
year(from_date) as y1 from dept_emp order by 1) y where d.dept_no = de.dept_no
and (makedate(y.y1, 1) between de.from_date and de.to_date) group by y.y1,
d.dept_name order by 1, 2
              3 place:
                     Execution time: 2.903 seconds
                     SQL: select year(now()) - year(birth_date) as age, gender,
avg(salary) as "Average Salary" from employees e, salaries s where e.emp_no =
s.emp_no and ("1988-08-01"  between from_date AND to_date) group by year(now())
- year(birth_date), gender order by 1,2
...
```

w: www.mariadb.com

e: info@mariadb.com

When the session ends a report will be written for the session into the logfile defined. That report will include the top 30 longest running statements, plus summary data for the session;

- The time the connection was opened.
- The host the connection was from.
- The username used in the connection.
- The duration of the connection.
- The total number of statements executed in the connection.
- The average execution time for a statement in this connection.

Duplicate Data From Your Application Into Cassandra

The scenario we are using in this example is one in which you have an online gaming application that is designed to work with a MariaDB/MySQL database. The database schema includes a high score table which you would like to have access to in a Cassandra cluster. The application is already using MaxScale to connect to a MariaDB Galera cluster, using a service names BubbleGame. The definition of that service is as follows

```
[BubbleGame]
type=service
router=readwritesplit
servers=dbbubble1,dbbubble2,dbbubble3,dbbubble4,dbbubble5
user=maxscale
passwd=6628C50E07CCE1F0392EDEEB9D1203F3
```

The table you wish to store in Cassandra in called HighScore and will contain the same columns in both the MariaDB table and the Cassandra table. The first step is to install a MariaDB instance with the Cassandra storage engine to act as a bridge server between the relational database and Cassandra. In this bridge server add a table definition for the HighScore table with the engine type set to cassandra. Add this server into the MaxScale configuration and create a service that will connect to this server.

```
[CassandraDB]
type=server
address=192.168.4.28
port=3306
protocol=MySQLBackend

[Cassandra]
type=service
```

w: www.mariadb.com

e: info@mariadb.com

```
router=readconnrouter
router_options=running
servers=CassandraDB
user=maxscale
passwd=6628C50E07CCE1F0392EDEEB9D1203F3
```

Next add a filter definition for the tee filter that will duplication insert statements that are destined for the HighScore table to this new service.

```
[HighScores]
type=filter
module=teefilter
match=insert.*HighScore.*values
service=Cassandra
```

The above filter definition will cause all statements that match the regular expression `inset.*HighScore.*values` to be duplication and sent not just to the original destination, via the router but also to the service named Cassandra.

The final step is to add the filter to the BubbleGame service to enable the use of the filter.

```
[BubbleGame]
type=service
router=readwritesplit
servers=dbbubble1,dbbubble2,dbbubble3,dbbubble4,dbbubble5
user=maxscale
passwd=6628C50E07CCE1F0392EDEEB9D1203F3
filters=HighScores
```