w: www.mariadb.com

e: info@mariadb.com

MariaDB MaxScale Top Filter

Mark Riddoch

Last Updated: 19th November 2014

w: www.mariadb.com

e: info@mariadb.com

Contents

w: www.mariadb.com

e: info@mariadb.com

Overview

The top filter is a filter module for MaxScale that monitors every SQL statement that passes through the filter. It measures the duration of that statement, the time between the statement being sent and the first result being returned. The top N times are kept, along with the SQL text itself and a list sorted on the execution times of the query is written to a file upon closure of the client session.

Configuration

The configuration block for the TOP filter requires the minimal filter options in it's section within the MaxScale.cnf file, stored in $MAXSCALE_HOME/etc/MaxScale.cnf.

```
[MyLogFilter]
type=filter
module=topfilter
```

Filter Options

The top filter does not support any filter options currently.

Filter Parameters

The top filter accepts a number of optional parameters.

Filebase

The basename of the output file created for each session. A session index is added to the filename for each file written.

```
filebase=/tmp/SqlQueryLog
```

The filebase may also be set as the filter, the mechanism to set the filebase via the filter option is superseded by the parameter. If both are set the parameter setting will be used and the filter option ignored.

Count

The number of SQL statements to store and report upon.

```
count=30
```

The default vakue for the numebr of statements recorded is 10.

w: www.mariadb.com

e: info@mariadb.com

Match

An optional parameter that can be used to limit the queries that will be logged by the top filter. The parameter value is a regular expression that is used to match against the SQL text. Only SQL statements that matches the text passed as the value of this parameter will be logged.

```
match=select.*from.*customer.*where
```

All regular expressions are evaluated with the option to ignore the case of the text, therefore a match option of select will match both `select`, `SELECT` and any form of the word with upper or lowercase characters.

Exclude

An optional parameter that can be used to limit the queries that will be logged by the top filter. The parameter value is a regular expression that is used to match against the SQL text. SQL statements that match the text passed as the value of this parameter will be excluded from the log output.

```
exclude=where
```

All regular expressions are evaluated with the option to ignore the case of the text, therefore an exclude option of select will exclude statements that contain both `where`, `WHERE` or any form of the word with upper or lowercase characters.

Source

The optional source parameter defines an address that is used to match against the address from which the client connection to MaxScale originates. Only sessions that originate from this address will be logged.

```
source=127.0.0.1
```

User

The optional user parameter defines a user name that is used to match against the user from which the client connection to MaxScale originates. Only sessions that are connected using this username will result in results being gebnerated.

```
user=john
```

w: www.mariadb.com

e: info@mariadb.com

Examples

Example 1 - Heavily Contended Table

You have an order system and believe the updates of the PRODUCTS table is causing some performance issues for the rest of your application. You would like to know which of the many updates in your application is causing the issue.

Add a filter with the following definition;

```
[ProductsUpdateTop20]
type=filter
module=topfilter
count=20
match=UPDATE.*PRODUCTS.*WHERE
exclude=UPDATE.*PRODUCTS_STOCK.*WHERE
filebase=/var/logs/top/ProductsUpdate
```

Note the exclude entry, this is to prevent updates to the PRODUCTS_STOCK table from being included in the report.

Example 2 - One Application Server is Slow

One of your applications servers is slower than the rest, you believe it is related to database access but you not not sure what is taking the time.

Add a filter with the following definition;

```
[SlowAppServer]
type=filter
module=topfilter
count=20
source=192.168.0.32
filebase=/var/logs/top/SlowAppServer
```

In order to produce a comparison with an unaffected application server you can also add a second filter as a control.

```
[ControlAppServer]
type=filter
```

w: www.mariadb.com

e: info@mariadb.com

```
module=topfilter
count=20
source=192.168.0.42
filebase=/var/logs/top/ControlAppServer
```

In the router definition add both filters

```
filters=SlowAppServer | ControlAppServer
```

You will then have two sets of logs files written, one which profiles the top 20 queries of the slow application server and another that gives you the top 20 queries of your control application server. These two sets of files can then be compared to determine what if anythign is different between the two.

Output Report

The following is an example report for a number of fictitious queries executed against the employees exaple database available for MySQL.

```
-bash-4.1$ cat /var/logs/top/Employees-top-10.137
Top 10 longest running queries in session.
==========================================

Time (sec) | Query
-----------+-----------------------------------------------------------------
    22.985 |  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
     5.304 |  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
     2.896 |  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
     2.160 |  select dept_name as "Department", sum(salary) / 12 as "Salary
Bill" from employees e, departments d, dept_emp de, salaries s where e.emp_no
= de.emp_no and de.dept_no = d.dept_no and ("1988-08-01"  between
```


```
de.from_date AND de.to_date) and ("1988-08-01"  between s.from_date AND
s.to_date) and s.emp_no = e.emp_no group by dept_name order by 1
     0.845 |  select dept_name as "Department", avg(year(now()) -
year(birth_date)) as "Average Age", gender from employees e, departments d,
dept_emp de where e.emp_no = de.emp_no and de.dept_no = d.dept_no and ("1988-
08-01"  between from_date AND to_date) group by dept_name, gender
     0.668 |  select year(hire_date) as "Hired", d.dept_name, count(*) as
"Count" from employees e, departments d, dept_emp de where de.emp_no =
e.emp_no and de.dept_no = d.dept_no group by d.dept_name, year(hire_date)
     0.249 |  select moves.n_depts As "No. of Departments",
count(moves.emp_no) as "No. of Employees" from (select de1.emp_no as emp_no,
count(de1.emp_no) as n_depts from dept_emp de1 group by de1.emp_no) as moves
group by moves.n_depts order by 1
     0.245 |  select year(now()) - year(birth_date) as age, gender, count(*)
as "Count" from employees group by year(now()) - year(birth_date), gender
order by 1,2
     0.179 |  select year(hire_date) as "Hired", count(*) as "Count" from
employees group by year(hire_date)
     0.160 |  select year(hire_date) - year(birth_date) as "Age", count(*) as
Count from employees group by year(hire_date) - year(birth_date) order by 1

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


Session started Wed Jun 18 18:41:03 2014
Connection from 127.0.0.1
Username        massi

Total of 24 statements executed.
Total statement execution time      35.701 seconds
Average statement execution time     1.488 seconds
Total connection time               46.500 seconds
-bash-4.1$
```