Alerting With MySQL and Nagios

http://bit.ly/nagios_mysql2013

Sheeri Cabral

Senior DB Admin/Architect

Mozilla

@sheeri

What is Monitoring?

- Threshold alerting
- Graphing/trending

Why Monitor?

- Problem alerting
- Find patterns
  - capacity planning
  - troubleshooting
- Early warning for potential issues

What to Alert?

- Problems you can fix
  - max_connections
  - long running queries
  - locked queries
  - backup disk space

Nagios is great because...

...anyone can write a plugin

The problem with Nagios...

...anyone can write a plugin

Official Nagios Plugins for MySQL

- check_mysql
- check_mysql_query

check_mysql

- db connectivity
- slave running
- slave lag using seconds_behind_master

check_mysql_query

- Checks the output of a query is within a certain range (numerical)

Third party plugins

	System vars	Status vars	Caching	Calculations
check_mysql	yes	yes	no	no
check_mysql (standard)	no	yes	no	no
check_mysqld_status	no	yes	no	no
check_mysql_stats	yes	no	yes	no
check_mysqld	no	many	no	no
check_mysql_health	yes	yes	no	Hard-coded
check_mysql	yes	yes	no	Hard-coded

“Let us know what you'd like to see!”

What I wanted

	System vars	Status vars	Caching	Calculations
mysql_health_check.pl	yes	yes	yes	flexible

mysql_health_check.pl

Caching

- Save information to a file

Caching

- Save information to a file
  - --cache-dir /path/to/dir/

Caching

- Save information to a file
  - --cache-dir /path/to/dir/
  - Use the file instead of connecting again

Caching

- Save information to a file
  - --cache-dir /path/to/dir/
  - Use the file instead of connecting again

--max-cache-age <seconds>

Caching

- Save information to a file
  - --cache-dir /path/to/dir/
  - Use the file instead of connecting again

--max-cache-age <seconds>

--no-cache to force connection

--mode=varcomp

- %metadata{varstatus}

--mode=varcomp

- %metadata{varstatus}
  - SHOW GLOBAL VARIABLES
  - SHOW GLOBAL STATUS

--mode=varcomp

- %metadata{varstatus}
  - SHOW GLOBAL VARIABLES
  - SHOW GLOBAL STATUS

- --expression allows word replacement

--mode=varcomp

- %metadata{varstatus}
  - SHOW GLOBAL VARIABLES
  - SHOW GLOBAL STATUS
- --expression allows word replacement
- --warning --critical are flexible

Sample Command Definition

```
define command {
 command_name    check_mysql_tmp_tables
 command_line    $USER1$/mysql_health_check.pl
--hostname $HOSTADDRESS$ --user myuser --password
  mypass
```

Sample Command Definition

```
define command {
 command_name    check_mysql_tmp_tables
 command_line    $USER1$/mysql_health_check.pl
--hostname $HOSTADDRESS$ --user myuser --password
  mypass
--cache-dir=/var/lib/nagios/mysql_cache
--max-cache-age=300
```

Sample Command Definition

```
define command {
 command_name    check_mysql_cxns
 command_line    $USER1$/mysql_health_check.pl
--hostname $HOSTADDRESS$ --user myuser --password
  mypass
--cache-dir=/var/lib/nagios/mysql_cache
--max-cache-age=300
--mode=varcomp
--expression=
     "Max_used_connections/max_connections * 100"
--warning=">80" -critical=">90"
}
```

Sample Command Definition

```
command_name    check_mysql_cxns

--mode=varcomp
--expression=
     "Max_used_connections/max_connections * 100"
--warning=">80" -critical=">90"
}
```

Sample Service Definition

```
define service {
     use                             generic-service
     host_name                       __HOSTNAME__
     service_description             MySQL Connections
     check_command                   check_mysql_cxns
     }
```

Rates

- Compare to last run

Rates

- Compare to last run
- mode=lastrun-varcomp

Rates

- Compare to last run
- mode=lastrun-varcomp
- current{expr}

Rates

- Compare to last run
- mode=lastrun-varcomp
- current{expr}
- lastrun{expr}

Rates

- Compare to last run
- mode=lastrun-varcomp
- current{expr}
- lastrun{expr}
- --comparison, no warn/crit

Query Rate

mysql_health_check.pl [host,user,pass]

Query Rate

mysql_health_check.pl [host,user,pass]

--mode lastrun-varcomp

Query Rate

mysql_health_check.pl [host,user,pass]

--mode lastrun-varcomp

--expression "(current{Queries} - lastrun{Queries})

Query Rate

```
mysql_health_check.pl [host,user,pass]
--mode lastrun-varcomp
--expression "(current{Queries} - lastrun{Queries})
                / (current{Uptime} – lastrun{Uptime})"
```

Query Rate

```
mysql_health_check.pl [host,user,pass]
--mode lastrun-varcomp
--expression "abs((current{Queries} - lastrun{Queries})
                / (current{Uptime} – lastrun{Uptime}))*100"
--comparison ">80"
```

```
define command {
 command_name    check_mysql_query_rate
 command_line    $USER1$/mysql_health_check.pl
--hostname $HOSTADDRESS$ --user myuser
  --password mypass
--cache-dir=/var/lib/nagios/mysql_cache
--max-cache-age=300
--mode=lastrun-varcomp
--expression=
  "abs((current{Queries} - lastrun{Queries}) /
  (current{Uptime} - lastrun{Uptime}))*100"
--comparison > 100
}
```

Other Modes

- --mode=long-query
- --mode=locked-query

- %metadata{proc_list}
  - SHOW FULL PROCESSLIST

Sample Command Definition

```
define command {
 command_name    check_mysql_locked_queries
 command_line    $USER1$/mysql_health_check.pl
--hostname $HOSTADDRESS$ --user myuser
  --password mypass
--cache-dir=/var/lib/nagios/mysql_cache
--max-cache-age=300
--mode=locked-query
--warning=$ARG1$ --critical=$ARG2$
}
```

Extending Information

sub fetch_server_meta_data {}

add a new hash key to %metadata

```
$metadata{varstatus} =
    $dbh->selectall_arrayref(
        q|SHOW GLOBAL VARIABLES|);
```

Extending Information

sub fetch_server_meta_data {}

add a new hash key to %metadata

```
$metadata{innodb_status} =
    $dbh->selectall_arrayref(
        q|SHOW ENGINE INNODB STATUS|);
```

For example

- %metadata{innodb_status}
  - SHOW ENGINE INNODB STATUS
    - Already exists, unused

For example

- %metadata{innodb_status}
  - SHOW ENGINE INNODB STATUS
    - Already exists, unused
- %metadata{master_status}
  - SHOW MASTER STATUS

For example

- %metadata{innodb_status}
  - SHOW ENGINE INNODB STATUS
    - Already exists, unused
- %metadata{master_status}
  - SHOW MASTER STATUS
- %metadata{slave_status}
  - SHOW SLAVE STATUS

“Standard” checks

% max connections

--expression
'Threads_connected/max_connections*100'

“Standard” checks

% max connections

--expression
'Threads_connected/max_connections*100'

InnoDB enabled

--expression “have_innodb”

--comparison=“ne 'YES'”

```
define command {

    command_name    check_mysql_connections

    command_line    $USER1$/mysql_health_check.pl
  --hostname $HOSTADDRESS$ --port 3306 --user
  <user> --password <password> --mode=varcomp
  --expression="Threads_connected/max_connections
  * 100" --comparison=">80"

}

define command {

    command_name    check_mysql_innodb

    command_line    $USER1$/mysql_health_check.pl
  --hostname $HOSTADDRESS$ --port 3306 --user
  <user> --password <password> --mode=varcomp
  --expression="have_innodb" --comparison="ne
  'YES'"

}
```

Did MySQL Crash?

Nagios set to check every 5 minutes

Did MySQL Crash?

Nagios set to check every 5 minutes

Might miss a crash

Did MySQL Crash?

Nagios set to check every 5 minutes

Might miss a crash

Uptime!

--expression 'Uptime'

--comparison="<1800"

Did MySQL Crash?

```
define command {

    command_name    check_mysql_uptime

    command_line    $USER1$/mysql_health_check.pl
  --hostname $HOSTADDRESS$ --port 3306 --user
  <user> --password <password> --mode=varcomp
  --expression="Uptime" --comparison="<1800"

}
```

“Standard” checks

read_only for slaves

--expression “read_only”

--comparison=“ne 'YES'”

“Standard” checks

read_only for slaves

--expression “read_only”

--comparison=“ne 'YES'”

% of sleeping connections

connected, # running, # max connections

“Standard” checks

read_only for slaves

--expression “read_only”

--comparison=“ne 'YES'”

% of sleeping connections

connected, # running, # max connections

--expression="(Threads_connected-Threads_running)/max_connections * 100"

--comparison=">$ARG1$"

```
define command {

    command_name    check_mysql_read_only

    command_line    $USER1$/mysql_health_check.pl
  --hostname $HOSTADDRESS$ --port 3306 --user
  <user> --password <password> --mode=varcomp
  --expression="read_only" --comparison="ne
  'YES'"

}

define command {

    command_name    check_mysql_connections

    command_line    $USER1$/mysql_health_check.pl
  --hostname $HOSTADDRESS$ --port 3306 --user
  <user> --password <password> --mode=varcomp
  --expression="(Threads_connected-
  Threads_running)/max_connections * 100"
  --comparison=">$ARG1$"

}
```

Limitations

- One check/calculation per Nagios service
  - But, you can use many variables
  - Cached output
- Does not output for performance data
  - Not hard to modify, just no need yet

Where to get it

https://github.com/palominodb/PalominoDB-Public-Code-Repository/tree/master/nagios/

www.palominodb.com->Community->Projects

Other Nagios Plugins

Nagios Plugin for Partitions

table_partitions.pl --host

--user --pass

--database --table

--range [days|weeks|months]

--verify #

Nagios Plugin for Partitions

table_partitions.pl --host db1.mozilla.com

--user nagiosuser --pass nagiospass

--database addons --table user_addons

--range months

--verify 3

Nagios Plugin for Checksums

After using pt-table-checksum (master)

Slaves have a table with checksum

this_crc vs master_crc

Nagios Plugin for Checksums

check_table_checksums.pl -H host

--user username --pass password

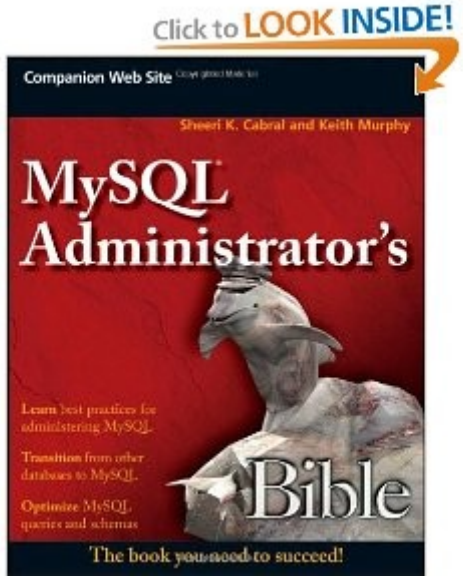
-T checksum_table

-l checksum_freshness

-b dbs,to,skip

More Resources

www.palominodb.com->Community->Projects



www.mysqlmarinate.com

scabral@mozilla.com

OurSQL podcast (oursql.com)

slides: http://bit.ly/nagios_mysql2013

MySQL Administrator's Bible

youtube.com/tcation

http://planet.mysql.com