Nagios Plugins for Monitoring MySQL

Sheeri K. Cabral, PalominoDB, Inc.

Abstract

Monitoring is critical for assessing the health of a MySQL instance. Currently, there is no way to monitor MySQL with a user-defined arbitrary calculation of MySQL variables using Nagios. This paper describes the development of a new Nagios plugin that allows user-defined, arbitrary calculations involving MySQL variables, while minimizing resource usage.

Monitoring MySQL

Monitoring is critical for assessing the health of a MySQL instance. Graphing trends can help debug problems, aid in capacity planning and illuminate potential trouble. Threshold alerting notifies that there is a problem. There are many monitoring tools that can monitor a MySQL instance, each having merits and drawbacks. Minimizing resource usage is important so that monitoring does not cause problems. For example, connecting to the database once for each variable needed in a calculation adds unnecessary overhead.

MySQL has many system and status variables that are useful to monitor, either on their own or combined in a calculation. There are different calculations and variables that are important, depending on how the MySQL instance is used and the configuration of the system. Monitoring must be flexible enough to allow comparisons using standard logical operators and arbitrary calculations that use numbers, arithmetic operators and MySQL system and status variables.

There is an existing set of templates for graphing MySQL trends with minimal overhead[1], but there is nothing similar for threshold alerting. When developing a flexible method for monitoring MySQL, Nagios was chosen due to its open framework and widespread use.

Existing Nagios Plugins for Monitoring MySQL

Nagios is distributed with two plugins to monitor MySQL – `check_mysql` and `check_mysql_query`. `check_mysql` can test connectivity to MySQL and slave lag, using the value of the `Seconds_Behind_Master` field from `SHOW SLAVE STATUS`. Testing that the output of an arbitrary query falls within a certain range is done with the `check_mysql_query` plugin. Additionally, there are currently over thirty plugins for MySQL at the Nagios Exchange[2], the official site for Nagios third-party projects. Of these, there are ten plugins that exclusively check replication and slave lag, seven plugins that check status and system variables, two plugins that exclusively check number of connections, and eleven plugins that check tables, backups or other thresholds.

The seven plugins that allow the user to specify what variables to check only allow specification of the variable names. None allow the user to specify calculations, though two of the plugins can check against pre-defined calculations[3]. Unfortunately, these plugins are not updated frequently enough to handle the new variables and performance output MySQL adds with each new version. They are also not flexible enough to allow monitoring of arbitrary calculations. Only one plugin does caching of any sort, but that plugin only allows one system variable to be checked.

Flexible Monitoring with Minimal Overhead

The flexibility of the new Nagios plugin was modeled after mysqltuner 2.0[4], which allows calculations to be specified using variable names in the configuration file. The program then substitutes the variable names with the appropriate values using simple word replacement. This enables the user to specify arbitrary calculations using any number of MySQL system and status variables. It also means that it is not necessary to upgrade mysqltuner in order to use new variables added in newer versions of MySQL. The Nagios plugin `mysql_health_check.pl` was developed using this concept of word replacement.

To minimize overhead, a cache file can be created to store the monitored items. In this way, a check using this Nagios plugin can use the cache file instead of connecting to the database again. This allows many calculations to be defined without straining database resources.

How mysql_health_check.pl Works

The plugin loads in the metadata to be checked in one of two ways: either by connecting to the database, or by reading the cache file. The plugin connects to the database

1 http://code.google.com/p/mysql-cacti-templates/

2 http://exchange.nagios.org/directory/Plugins/Databases/MySQL/

3 For example, check_mysql_health calculates the query cache hitrate as qcache_hits/(Com_Select+qcache_hits)*100.

4 https://launchpad.net/mysqltuner

when the `--no_cache` parameter is given, the cache file does not exist, or the cache file is older than `--max_cache_age` seconds.

The plugin retrieves and caches the output of:

- `SHOW GLOBAL VARIABLES`
- `SHOW GLOBAL STATUS`
- `SHOW ENGINE INNODB STATUS`
- `SHOW FULL PROCESSLIST`

There are three different modes for mysql_health_check.pl:

- Variable comparison mode (`--mode=varcomp`)
- Long-running query mode (`--mode=long-query`)
- Locked query mode (`--mode=locked-query`)

Variable Comparison Mode

Variable comparison mode requires arguments for an *expression* and a *comparison*. The expression is parsed using word replacement against the variables collected in `SHOW GLOBAL VARIABLES` and `SHOW GLOBAL STATUS`. The parsed expression is then evaluated against the comparison, which represents the threshold. If the comparison matches, the plugin exits with a status of 2, which means `CRITICAL` in Nagios.

For example, to calculate the percentage of connected threads compared to the maximum allowed connections, the parameters are:

```
mysql_health_check.pl --hostname <host>
--user <user> --password <pass> \
--cache_dir=/var/nagios/mysql.cache/
--max_cache_age 60 --mode=varcomp \
--
expression="Threads_running/max_connecti
ons * 100" --comparison=">80"
```

The exit status of this check will be 0 (Nagios `OK`) if the expression evaluates to 80 or less, and 2 (Nagios `CRITICAL`) if the expression evaluates to larger than 80.

Long-running Query Mode

Long-running query mode requires thresholds for warning and critical values. The information collected from `SHOW FULL PROCESSLIST` is parsed to see if there are any queries that have been running for longer than the warning and critical thresholds. All queries are checked, except for those run by `system_user` and those running `Binlog Dump`, which are part of replication that normally run for a long time without any problems arising. The exit status is 0 (Nagios `OK`) if there are no queries running longer than either threshold, 1 (Nagios `WARNING`) if there is at least one query that is running longer than the warning threshold and no queries running longer than the critical threshold, and 2 (Nagios `CRITICAL`) if there is at least one query running longer than the critical threshold.

Locked Query Mode

Locked query mode is very similar to long-running query mode. It requires thresholds for warning and critical values. The information collected from `SHOW FULL PROCESSLIST` is parsed to see if there are any queries in a `Locked` state that have been running for longer than the warning and critical thresholds. All queries are checked. The exit status is 0 (Nagios `OK`) if there are no locked queries running longer than either threshold, 1 (Nagios `WARNING`) if there is at least one locked query that is running longer than the warning threshold and no locked queries running longer than the critical threshold, and 2 (Nagios `CRITICAL`) if there is at least one locked query running longer than the critical threshold.

Limitations

The biggest limitation of `mysql_health_check.pl` currently is that there is no way to induce a `WARNING` in variable comparison mode. Adding this feature requires having two comparisons, which is not difficult. Another limitation is that there is no way to compare the value of the expression against the previous value of the expression. Having this feature would allow monitoring increases, decreases and rate of change. Adding this feature is a bit tricky because it would require keeping older cached data as well as the currently cached data. This is planned for a future version.

Currently there is no Nagios performance data output for long-running query mode and locked query modes. When the check returns `OK` in these modes, there is no further information given. When the check returns `CRITICAL` in these modes, all queries and the times they have been running is printed. Because the queries are SQL, it is difficult to parse this output into performance data.

Conclusion

While `mysql_health_check.pl` is not perfect, it surpasses currently available MySQL Nagios plugins that monitor variables. By allowing the user to specify arbitrary calculations with MySQL variables, this new Nagios plugin does not require an upgrade when new MySQL variables are introduced nor when a new calculation is desired.

About the Author

Sheeri K. Cabral was the first Oracle ACE Director for MySQL, has a master's degree in computer science

specializing in databases from Brandeis University and a background in systems administration. Unstoppable as a volunteer and activist since age 14, Cabral founded and organizes the Boston, Massachusetts, USA, MySQL User Group and is the creator and co-host of OurSQLCast: The MySQL Database Community Podcast, available on iTunes. She is the founder and current treasurer Technocation, Inc., a not-for-profit organization providing resources and educational grants for IT professionals. She wrote the *MySQL Administrator's Bible* and has been a technical editor for high-profile O'Reilly books such as *High Performance MySQL 2nd Edition* and *CJ Date's SQL and Relational Theory*.