Foglight for MySQL

Cartridge Guide

TABLE OF CONTENTS

Foglight for MySQL Introduction

With over 65,000 downloads per day and six million installations worldwide, MySQL has become the most widely deployed open source database solution and is second most widely deployed database solution period! Originally released in 1995, adoption has steadily grown and with the acquisition by Oracle, MySQL now sets the standard as a robust low-cost alternative to high-cost enterprise data stores.

Early adopters included Internet pioneers such as Google, eBay, Craigslist and Yahoo, but the list of customers has steadily grown and now includes financial industry giants as Dun & Bradstreet, JP Morgan Chase, and telecom giants Nokia as well as pharmaceutical powerhouse Eli Lilly. Whether you are building and supporting commercial websites, distributing enterprise applications or engineering advanced communications networks, the technologies used to run your organization must be readily adaptable for you to remain competitive.

Foglight for MySQL Overview

Based on Quest Software's leading application performance management solution; Foglight for MySQL provides combines world-class monitoring and alerting with operational best-practices designed to ensure the performance and availability of your MySQL databases. Utilizing standard API's, the cartridge provides integration to MariaDB and MySQL Versions 5.0 and above. The robust nature of the Agent ensures its ability to collect granular performance data and display that information through intuitive, easy-to-use dashboards. Based on vast experience building and deploying monitoring solutions, Foglight for MySQL ensures the performance and availability of this critical component. The solution leverages best practices for collecting, storing and representing data as well as detecting environmental abnormalities.

True Enterprise Information Correlation

At the core of the solution is Quest Software's Foglight. Foglight's rich architecture combines a central repository, rules and notification engine, data visualization platform to provide an application performance management that is second to none. This platform permits MySQL data to be combined with data from other enterprise domains to create a true end-to-end view of your critical applications and services. MySQL data collection intervals are pre-defined but are user adjustable for each functional area of the database. Once data has been collected, it is stored in the Foglight repository for a period defined by the administrator. The repository centralizes data collection and facilitates data visualization and trend analysis. Combining MySQL data with other metrics collected by Foglight produces a solution that provides the operator with unprecedented visibility into applications and business services.

Dashboards

Data is displayed through database specific dashboards. Foglight for MySQL provides a series of specific dashboards for MySQL databases and tables. These dashboards are easily complimented and extended utilizing Foglight's powerful drag-and-drop dashboarding capabilities. Operators can easily save dashboards as 'bookmarks' and return to them as needed. Dashboards and/or Reports can be created based on any data stored in the central repository and is not limited to data from a specific database such as MySQL.

Diagnostics is made possible through drill-downs from included dashboards. Each dashboard was designed to provide at-a-glance health state for domains. Additional diagnostic information is made available through drill-downs providing the critical information necessary to troubleshoot complex issues. Since most enterprises rely on a complex technology stack to ensure business continuity, Foglight provides the ability to correlate, view, alert and report on technology from most standard enterprise applications.

Rules Engine

Alerting is often the first indication that a problem exists or may exist if preventative measures are not taken in a timely manner. A notable strength of the solution lies is its ability to leverage both MySQL Error Log alerts and the Foglight rules engine. Error Log messages are propagated to Foglight where they are assigned a severity and can be acknowledged or cleared. Foglight provides the ability for notifications to be sent to administrative and support personnel. The Foglight rules engine permits the operator to easily construct or modify any rule based on any metric using standard operators and variables. A robust set of rules is included with the solution and is defined in this document.

Statement Digests

Statement digest monitoring extends the rich functionality of the MySQL Cartridge, providing the ability for Foglight to collect and analyze information from your MySQL server’s performance_schema for versions 5.6.5 and greater, or the Slow Query Log for lesser versions. Queries are normalized and aggregated in order to provide meaningful results on the performance of unique query structures without exposing non-relevant or protected string values.

This functionality allows users to quickly identify slow running and problematic queries and helps administrators better understand the efficiency of these queries in both development and production environments with a minimal amount of overhead. Further information provides an accurate picture of how queries perform when the server is under a realistic workload with detailed metrics on performance throughout the query’s execution, along with an explain plan available on request. Coupled with Foglight’s historical data collection and powerful rules engine, this ensures you will be alerted on all negative performance trends or spikes in query performance.

Foglight for MySQL Requirements

Foglight for MySQL is compatible with MySQL 5.5+ and equivalent versions for drop-in replacements like MariaDB and managed database services like AWS Aurora/RDS and Microsoft Azure. However, older versions of the server may not provide some monitoring data that Foglight for MySQL presents for later versions. Most notably, statement digest data is not present earlier than 5.6.5.

The agent may be run on a FglAM that is either local or remote to the MySQL server. More information on configuring the MySQL server for monitoring can be found in the MySQL Server Pre-Configuration section of this document.

Installing and Configuring Agents

Installation of Foglight for MySQL is covered in the following sections and should be performed in order:

- MySQL Server Pre-Configuration
- Cartridge Installation
- Creating and Configuring Agents

MySQL Server Pre-Configuration

In order to allow full monitoring of the MySQL Server, the agent will require a user with sufficient privilege to execute system queries. Additional steps may also be required to enable a SSL connection or monitoring the slow query log if desired. These are covered in the following sections:

- MySQL Agent User Permissions
- Configuring an Encrypted Connection
- Configuring the MySQL Slow Query Log

MySQL Agent User Permissions

The Foglight MySQL agent requires certain minimum privileges for the MySQL Database User to be able to monitor a MySQL database.

Create a MySQL Database User by logging into the MySQL server and granting the privileges identified below.

User privileges required for the MySQL agent on the host machine:

- SELECT
- REPLICATION CLIENT (if monitoring replication)
- PROCESS
- SUPER (for MySQL versions below 5.1.24)

If Administration is enabled, the Admin user provided in the agent properties will be required to have privileges necessary to run operations or request explain plans for the functions that you want to have available.

If monitoring a replication slave server with the agent, that database user must have:

- SELECT
- REPLICATION CLIENT

Example 1:

CREATE USER '<user>'@'<localhost or DB HostName or IP>' IDENTIFIED BY '<password>';
GRANT SELECT, REPLICATION CLIENT, PROCESS ON *.* TO '<user>'@'<localhost or DB HostName or IP>';
FLUSH PRIVILEGES;

e.g.
CREATE USER ‘MySQLuser’@’localhost’ IDENTIFIED BY ‘MySQLpassword’;
GRANT SELECT, REPLICATION CLIENT, PROCESS ON *.* TO ‘MySQLuser’@’localhost’;
FLUSH PRIVILEGES;

or

Example 2:

GRANT SELECT, REPLICATION CLIENT, PROCESS ON *.* TO '<user>'@'<localhost or DB HostName or IP>' IDENTIFIED BY '<password>';
FLUSH PRIVILEGES;

e.g.
GRANT SELECT, REPLICATION CLIENT, PROCESS ON *.* TO ‘MySQLuser’@’localhost’ IDENTIFIED BY ‘MySQLpassword’;
FLUSH PRIVILEGES;

Configuring an Encrypted Connection

The below instructions cover common steps used to configure an encrypted connection from the MySQL Agent client. For full information on secure connections and server-side configuration, refer to the Using Secure Connections section of the MySQL documentation for your version.

In order to use SSL, your MySQL server must be built with OpenSSL or yaSSL. To check whether SSL is enabled, run this query:

```
SHOW VARIABLES LIKE 'have_ssl';
```

If the query returns YES, your server can use SSL. If it returns DISABLED, the server must be started with the SSL options listed in the above mentioned section. SSL and RSA certificates and keys must also be generated in order to use SSL. Information on generating those can be found here.

The client requires a client certificate and certificate authority (CA) certificate. These are named client-cert.pem and ca.pem respectively if generated by the MySQL server and should be located in the data directory. First, the client certificate needs to be converted into DER format. This can be performed by downloading OpenSSL and running the following command:

```
openssl x509 -outform DER -in client-cert.pem -out client.cert
```

Then, the certificates must be imported into the FglAM keystore. You can use the bundled keytool, which will be located in the Foglight Agent Manager\jre\1.8.0.72\jre\bin directory, or the equivalent on your system, with these commands:

```
keytool.exe -import -file ca.pem -keystore truststore -alias mysqlServerCACert
```

```
keytool.exe -import -file client.cert -keystore keystore -alias mysqlClientCertificate
```

If you have not changed the password for the keystore, the default password will be “changeit”. Next, edit the baseline.jvmargs.config file in the Foglight Agent Manager\state\default\config directory and add the following parameters with file paths and passwords appropriate for your system. Escape any quotes with a ‘\’.

```
vmparameter.0 = "-Djavax.net.ssl.keyStore=\"C:/Program Files/Common Files/Dell/Foglight Agent
Manager/jre/1.8.0.72/jre/bin/keystore\"";
vmparameter.1 = "-Djavax.net.ssl.keyStorePassword=changeit";
vmparameter.2 = "-Djavax.net.ssl.trustStore=\"C:/Program Files/Common Files/Dell/Foglight Agent
Manager/jre/1.8.0.72/jre/bin/truststore\"";
vmparameter.3 = "-Djavax.net.ssl.trustStorePassword=changeit";
```

Then, restart the FglAM and continue with the agent configuration, setting the “Use SSL” option in the Agent Properties to true.

Configuring the MySQL Slow Query Log

For MySQL servers version 5.6.5 or greater, it is recommended that you monitor statement digests through the performance_schema, as it collects more information. However, some limited information can be gathered through the slow query log.

Configuration of the Slow Query Monitor first requires that slow query logging is enabled. This can be accomplished in one of three ways. 1) Change the server runtime parameters while the server is running. 2) Provide certain command-line options when starting the server. 3) Edit the MySQL configuration file to enable and configure the log.

We recommend using option three (3). Choosing option 3 will ensure that the MySQL Server is always started with the correct parameters. Option one (1) can be used to avoid doing an initial restart of the server in order to enable the slow query log. As of version 5.1.29, the following options may be used:

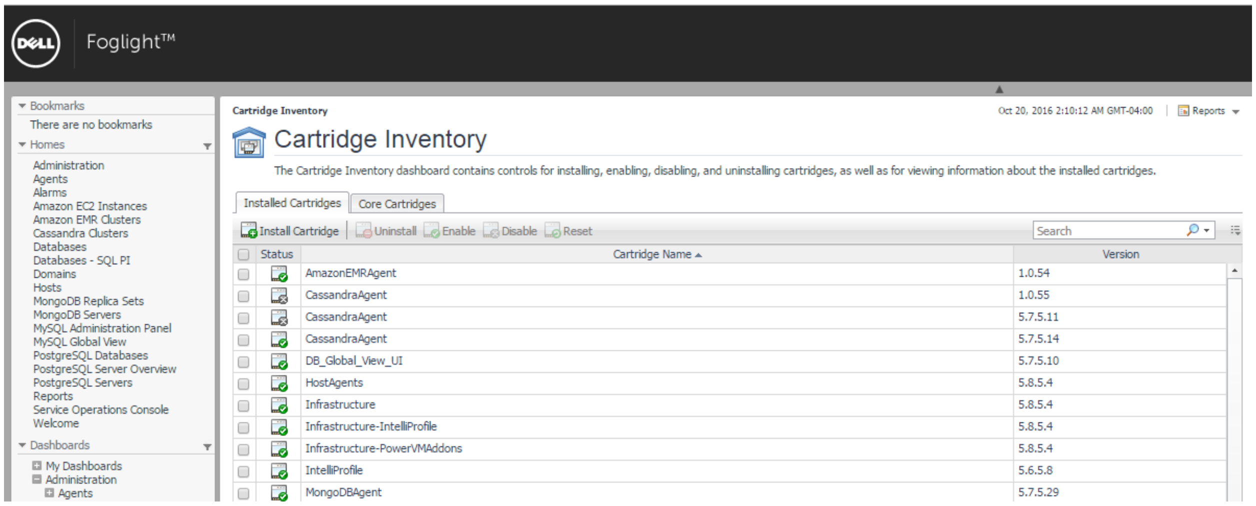
Option	Sample Value	Required	Note
slow_query_log	1	Yes, defaults to 0	1 enables the log, 0 disables it
slow_query_log_file	C:\Program Files\MySQL\slow_query.log	Yes, defaults to *hostname*-slow.log	Any path name is acceptable
long_query_time	0.5	Yes, defaults to 10	Units are in seconds, can also use 0 or microseconds
log_short_format	FALSE	Yes, defaults to FALSE	Must be set to FALSE
log_slow_admin_statements	OFF	No, defaults to OFF	Logs admin statements like ANALYZE, OPTIMIZE, ALTER TABLE, etc.
log_queries_not_using_indexes	ON	No, defaults to OFF	Logs queries expected to retrieve all rows
log_slow_slave_statements	OFF	No, defaults to OFF	Logs statements on a replication slave server
min_examined_row_limit	0	No, defaults to 0	Logs statements that have examined minimum # of rows

See section 5.2.5 of the online MySQL documentation for more information on the slow query log specific to your version. Some of these properties may become deprecated in future versions or not exist in versions before 5.1.12.

Additionally, it is important to understand that the server does not write queries handled by the query cache to the slow query log, nor does it write queries that would not benefit from the presence of an index as the table has either zero or one row. Also of note is that the server does not automatically rotate the slow query log. If you wish to rotate your log file, you may simply delete or rename the current log file manually or with a utility program and the server will create a new log file itself. It is recommended you do this at a period of low activity so as to not cause the agent to miss reading important data.

Cartridge Installation

1. Open Foglight Web Console.
2. From the navigation pane, select: **Administration**.
3. Click “Licenses” in the Administer Server section.
4. Install the appropriate license key to run the cartridge.
5. Return to the **Administration** dashboard.
6. Click “Cartridges” in the Administer Server section.
7. Load the ***MySQLAgent-5_x.car*** file by clicking the Install Cartridge button. Leave the “Enable on Install” check box checked when installing the cartridge.
8. Once the installation is completed on the Foglight Management Server, the MySQL Agent Cartridge will appear in this list below as an installed cartridge.



Cartridge Inventory

The Cartridge Inventory dashboard contains controls for installing, enabling, disabling, and uninstalling cartridges, as well as for viewing information about the installed cartridges.

Status	Cartridge Name ▲	Version
✓	AmazonEMRAgent	1.0.54
✗	CassandraAgent	1.0.55
✗	CassandraAgent	5.7.5.11
✓	CassandraAgent	5.7.5.14
✓	DB_Global_View_UI	5.7.5.10
✓	HostAgents	5.8.5.4
✓	Infrastructure	5.8.5.4
✓	Infrastructure-IntelliProfile	5.8.5.4
✓	Infrastructure-PowerVMAddons	5.8.5.4
✓	IntelliProfile	5.6.5.8
✓	MongoDBAgent	5.7.5.29

Creating and Configuring Agents

Agents can be created in one of two ways:

- Using the Agent Installer Wizard
- Using the Agent Status Dashboard

The Agent Installer Wizard simplifies the agent creation and configuration process and can be accessed from the Databases dashboard. For advanced configuration or modification of agent properties post-creation, use the Agent Status dashboard.

Using the Agent Installer Wizard

Foglight for MySQL provides a graphic, intuitive method for creating and configuring agents, which can be used instead of Foglight's default method for creating agents and editing their properties using the Agent Status dashboard. Foglight for MySQL allows running a wizard that provides a common entry point for adding database instances and then configuring these instances for monitoring.

To run the instance installation wizard:

1. On the navigation panel, click Homes > Databases.
2. Click the MySQL box in the Databases View, and then click Monitor.
3. The Agent Installer Wizard dialog box appears.
4. The first card - Agent Deployment – has two fields:
    a. Agent Name – Provide a name for the agent that will be created. This is not canonical and should be representative of the database instance that this agent will monitor.
    b. Agent Manager - Choose the agent manager on which the agent should run. Considerations for this may include physical or virtual locality to the monitored instance, allocated resources, or grouping with other agents of the same type or monitored environment. If the agent package has not been deployed to this Agent Manager yet, it will be installed when the first agent of this type is created.
5. The second card – Agent Properties – requires a basic set of parameters for connecting to and monitoring the database instance. A full explanation of these properties is available in the Agent Properties section of this document.
6. The third card – Agent Summary – displays a review of the configuration that will be created and an option allowing the agent to be activated after creation. If the configuration looks good, click the Finish button to start the process.
7. When the process completes, a results screen will appear showing the results of agent creation. If the agent was not created, follow the instructions on the results screen. If successful, the database instance should appear in the Databases table within a few minutes.

Note: If the agent was created successfully but data is not appearing, go to the Dashboards > Administration > Agents > Agent Status page and click the icon in the Log File column for the agent you created. In most cases, the reason for the failure will be obvious. You can also refer to the *Foglight for MySQL Installation and Troubleshooting* document for common errors and solutions. If the solution requires reconfiguring the agent properties, follow steps 3-7 of the Using the Agent Status Dashboard section.

Using the Agent Status Dashboard

The Agent Status page can be used to create new agents and configure and manage existing agents. To access the page from the navigation pane, select: Dashboards > Administration > Agents > Agent Status.

Use the following steps to create a new agent instance:

1. If the MySQL agent package has never been deployed to the FglAM that will be used to host the agent, this must be done before an agent has been created. You can use the Deploy Agent Package button on the Agent Status or Agent Managers page to perform this.
2. Click the Create Agent button and follow the instructions for the cards:
   a. **Host Selector** - Choose the Agent Manager on which the agent should run. Considerations for this may include physical or virtual locality to the monitored instance, allocated resources, or grouping with other agents of the same type or monitored environment.
   b. **Agent Type and Instance Name** – Select the MySQLAgent type. Then, select the Specify Name radio button and provide a name for the agent that will be created. This is not canonical and should be representative of the database instance that this agent will monitor.
   c. **Summary** – Click Finish.
3. Once the agent has been created, click the checkbox next to the MySQL agent.
4. Click the **Edit Properties** button.
5. Select **Modify the default properties for this agent**.
6. Edit the agent properties for the MySQL agent instance:
   - Connection Parameters (mandatory)
   - Administration (optional)
   - Replication (optional)
   - Statements (optional)
   - Collection Intervals (optional)
   - Slow Query Log Monitoring (optional)
   - Additional Options (optional)
7. Click the **Activate** button.

To modify the properties for an existing agent, skip to step 3 and Deactivate, then Reactivate the agents after changing the configuration.

Agent Properties

When an agent connects to the Foglight Management Server, it is provided a set of properties that is then used to configure its correct running state.

Default Agent properties are installed with the Foglight Cartridge for MySQL. However, the user will typically edit the default agent properties. Agent properties may apply only to a specific agent instance, or may be applicable across multiple agents.

For more information about working with agent properties, see the *Foglight Administration and Configuration Guide.*

To modify the agent properties for a new agent instance:
Open Foglight.

8. From the navigation pane, select: **Dashboards** > **Administration** > **Agents** > **Agent Status**. The Agent Status screen appears.
9. Deploy the MySQL agent package.
10. Create a MySQL agent instance.
11. Click on a MySQL agent type row. The selected row is highlighted with a yellow background.
12. Click the **Edit Properties** button.
13. Select **Modify the default properties for this agent**.
14. Edit the agent properties for the MySQL agent instance:
    - Setting Connection Parameters (mandatory)
    - Setting Slave Connection Parameters (optional)
    - Setting Collection Intervals (optional)
15. Click the **Activate** button.

Connection Parameters (mandatory)

- **Database Host** – Host where MySQL server is running. Default is “localhost”. (e.g.<hostname> or <IP address>)
- **Database Port** – Port the MySQL database is running. Default is 3306.
- **Database Name** – Name of a valid MySQL database that the user is allowed to connect to. Note: This is for connection purposes only. All non-system databases will still be monitored. The ‘mysql’ database is usually an acceptable entry.
- **Database User** – User that can connect to the MySQL server being monitored.
- **Database Password** – Password of the user that can connect to the MySQL database being monitored.
- **Use SSL** – Requires the connection to the MySQL server to use SSL/TLS. For more information on enabling SSL, refer to the Configuring an Encrypted Connection section.
- **Allow Public Key Remote Retrieval** – Allows the client to retrieve the public key from the server in order to secure the connection. This must be enabled if an SSL connection is not configured and the public key is not available locally.
- **Local Public Key File Path** – The file path to the public key for the MySQL server that the agent is attempting to connect with. The public key must be locally accessible by the agent, which is hosted on the FglAM machine. This can be used instead of allowing public key retrieval, preventing possible MITM interception.
- **Use Cleartext** – Enables the cleartext authentication method that sends the unhashed password to the MySQL server. It is recommended that SSL be enabled if this is done.

Administration (optional)

- **Enable Administration** – Set to true in order to use the MySQL Administration Panel for this agent. See the MySQL Administration Panel for more information. Default is false.
- **Enable Explain Plans** – Set to true in order to enable Explain Plan requests from the Administration Panel and Statement Digests pages. Default is false.
- **Admin User** – User that will be employed only to perform administrative actions on the monitored MySQL server. This user must have the privileges to perform the requested actions or the action will fail.
- **Admin Password** – Password of the user that will be employed to perform administrative actions on the server.

Replication (optional)

- **Monitor Master Status?** – Set to true to monitor the Master replication status on the MySQL server.
- **Monitor Slave Status?** – Set to true to monitor the Slave replication status on the MySQL server.
- **Monitor Replication Slave?** – Set to true to monitor the Slave replication status (and only this) on the replication slave server, of which the parameters below pertain to.
- **Database Host** – Host where MySQL Replication Slave Server is running.
- **Database Port** – Port the Slave Connection is running on.
- **Database Name** – Name of MySQL Replication Server Database to be monitored.
- **Database User** – User that can connect to the MySQL Replication Server database being monitored.
- **Database Password** – Password of the user that can connect to the MySQL Replication Server database being monitored.
- **Use SSL** – Requires the connection to the MySQL server to use SSL/TLS. For more information on enabling SSL, refer to the Configuring an Encrypted Connection section.
- **Allow Public Key Remote Retrieval** – Allows the client to retrieve the public key from the server in order to secure the connection. This must be enabled if an SSL connection is not configured and the public key is not available locally.
- **Local Public Key File Path** – The file path to the public key for the MySQL server that the agent is attempting to connect with. The public key must be locally accessible by the agent, which is hosted on the FglAM machine. This can be used instead of allowing public key retrieval, preventing possible MITM interception.
- **Use Cleartext** – Enables the cleartext authentication method that sends the unhashed password to the MySQL server. It is recommended that SSL be enabled if this is done.

Statements (optional)

- **Statements** – If the MySQL server version is 5.6.5+ and the performance_schema engine is enabled, the agent will collect statement digest information at this interval. Default set to 300 seconds
- **# of Top Statements** – Number of statements that will be collected, ordered by the Sort By property. Default is 1000.
- **Sort By** – Ordering parameter for statement collection limit. Default is Total Executions.

Collection Intervals (optional)

The Collection Interval fields in the agent properties are used to set the sample frequencies. Default collection frequencies (in seconds) have already been set for each table below.

- **Blocked Transactions** – Default set to 60 seconds.
- **Buffer Pool** (i.e. Innodb_Buffer_Pool) – Default set to 300 seconds.
- **Configuration Monitoring** - Default set to 300 seconds.
- **Connection_Status** – Default set to 60 seconds.
- **Database Information** – Default set to 300 seconds.
- **Database Stats** – Default set to 300 seconds.
- **Failed Logins** - If the MySQL server version is 5.6.0+ and the performance_schema engine is enabled, the agent will collect information on failed login attempts at this interval, shown on the Connections dashboard. Default set to 300 seconds.
- **Galera** – Default set to 60 seconds.
- **Handler** – Default set to 300 seconds.
- **Index Structure -** Default set to 3600 seconds.
- **Index/Table-level Compression** – Default set to 1800 seconds.
- **InnoDB Compression** – Default set to 300 seconds.
- **InnoDB Engine** (i.e. Innodb_Storage_Engine) – Default set to 300 seconds.
- **Joins** – Default set to 300 seconds.
- **Key Buffer** – Default set to 300 seconds.
- **Network Interface** – Default set to 300 seconds.
- **Query Cache** – Default set to 300 seconds.
- **Replication** – Default set to 300 seconds.
- **Sort Buffer** – Default set to 300 seconds.
- **Tables** – Default set to 1800 seconds.
- **Table Locks** – Default set to 60 seconds.
- **Thread Pool** – Default set to 300 seconds.
- **Top Sessions** – Default set to 300 seconds.
- **Transaction Log** (i.e. Innodb_Transaction_Log) – Default set to 1800 seconds.
- **Users** – Default set to 3600 seconds.

Slow Query Log Monitoring (optional)

- **Monitor Slow Query Log** - Enable this collection by setting to True.
- **File Access** – If the MySQL Agent is running locally on the same machine as the MySQL server it is monitoring, you may select Local. The other option, Remote_SSH, allows an SSH connection to a remote server in order to retrieve the slow query log.
- **Collection Period** – How often collection occurs, in seconds. It is recommended that this setting be at least 30-60 minutes or more. Currently, unique queries can only be aggregated within their own collection period, so a lengthier collection period will provide a more accurate analysis of a unique query without having to navigate through too many data collections.
- **Line Limit** – Restricts the amount of lines read from the slow query log during each collection. Unless a truly exceptional amount of data is being generated to the slow query log (in which case the long_query_time parameter should probably be increased), this parameter should remain at 0. A value of -1 will cause the agent to read all previous data during its first collection rather than marking the current file position and reading from that point during its next collection.
- **File Path** – File path to slow query log directory.
- **File Name** – File name of slow query log. Optionally, for Local collection only, a regex string may be used and the most recently modified file matching the regex string name will be used.

If using remote file access, the following values must be provided

- **Remote Hostname or IP** – A hostname or IP address valid from the MySQL Agent's location.
- **SSH User** – An SSH user with access to the slow query log's location and file permissions to perform a read action.
- **SSH Password** – The password for the provided SSH user.

Additional Options (optional)

- **Enable Dynamic Memory Allocation** – Allows the agent to request more memory from the FglAM when monitoring a server with over 10,000 tables. Default is false.
- **Agent Host Name** – If a hostname alias is being used for monitoring purposes that is different than the one provided by the MySQL server, provide that here. If using the Infrastructure cartridge for OS monitoring, this name should match any alias provided for OS monitoring in order to link the data correctly.
- **Server Time Zone Override** – Allows the client to override the server time zone, usually for the purpose of correcting a java-incompatible time zone id. A list of acceptable time zone IDs can be found here.
- **Client Time Zone Override** – Allows the client to override the time zone used for time-based performance data. A list of acceptable time zone IDs can be found here.
- **Exclude DB Table/Index Collections** – Table and index collections for individual databases can be disabled here. This will decrease resource usage by the agent and used space in the FMS repository. By default, the list is populated with system databases.
- **Skip Startup Connection Test** – When the agent is first activated, it performs a connection test to ensure that a connection can be made to the MySQL server. If it cannot, it is assumed there is a persistent failure due to incorrect configuration or other issues which need to be resolved and the agent will show as failed until fixed and restarted. In cases like a restart when the agent and the MySQL server are located on the same machine and you expect the agent to activate before the server is accessible, you may want to disable this. If you wish to skip the connection test, set to true.
- **Minimum TRX Block Time (sec)** – This is the minimum wait time for a transaction to be considered deadlocked and thus valid for inclusion in the Blocked Transactions collection. The default time is 15 seconds.
- **Include Views in Table Collection** – Option to include Views in the table collection. Set to false by default.

Upgrading the Agent

1. Go to Dashboards > Administration > Cartridges > Cartridge Inventory and click the Install Cartridge button.
2. Locate the .car file on your system and install it with auto-enable selected. If you get a message that a bundled cartridge is of an older version than the one currently enabled on your FMS and will not be enabled, ignore it and continue.
3. Once the cartridge is installed and enabled, go to Dashboards > Administration > Agents > Agent Managers. Agent Managers that can be upgraded with newer agent packages will show “yes” in the Upgradable **|** Agents column. Select all Agent Managers you wish to upgrade and click the Upgrade button.

Note: If an Agent Manager is not upgradable, check that the Agent Manager version is compatible with the newer agent version. If it is not, the Agent Manager will need to be upgraded first.

Removing Monitored Databases

1. Go to the Databases dashboard.
2. Select the databases you wish to remove.
3. Click the Settings button, then click ok.

Note: Doing this will remove the monitoring agents as well as the historical data already collected. If you wish to delete only the agents, you can do that on the Administration > Agents > Agent Status page. Because the Databases dashboard only shows databases which are being actively monitored, you will only be able to view these databases by going through the MySQL > MySQL Global View dashboard.

Administration

Opening the Databases Administration Dashboard

You can edit agent settings for one or more MySQL instances on the Databases > Administration dashboard.

> **NOTE:** If you attempt to select instances of more than one type of database, such as a MySQL database and an Oracle database, an error message is displayed.

To open the Databases Administration dashboard:

1. In the navigation panel, under **Homes**, click **Databases**.
2. Select the check boxes beside one or more MySQL instances.
3. Click **Settings** and then click **Administration**. The Administration dashboard opens, containing settings for all the selected agents. Settings are broken down into categories, which are organized under a MySQL tree.

Reviewing the Administration Settings

The Databases Administration dashboard allows settings options for collecting, storing, and displaying data, which apply to all the currently selected agents. Click a category of settings on the left (for example: Connection Details) to open a view containing related settings on the right.

To view the full list of selected agents, click the **Selected Agents** button at the upper right corner of the screen. To change the list of agents to which the metrics will apply, exit the Databases Administration dashboard, select the requested agents and re-open the view.

Customizing Alarms for Foglight for MySQL Rules

Many Foglight for MySQL multiple-severity rules trigger alarms. To improve your monitoring experience, you can customize when alarms are triggered and whether they are reported. You can also set up email notifications.

Introducing the Alarms View

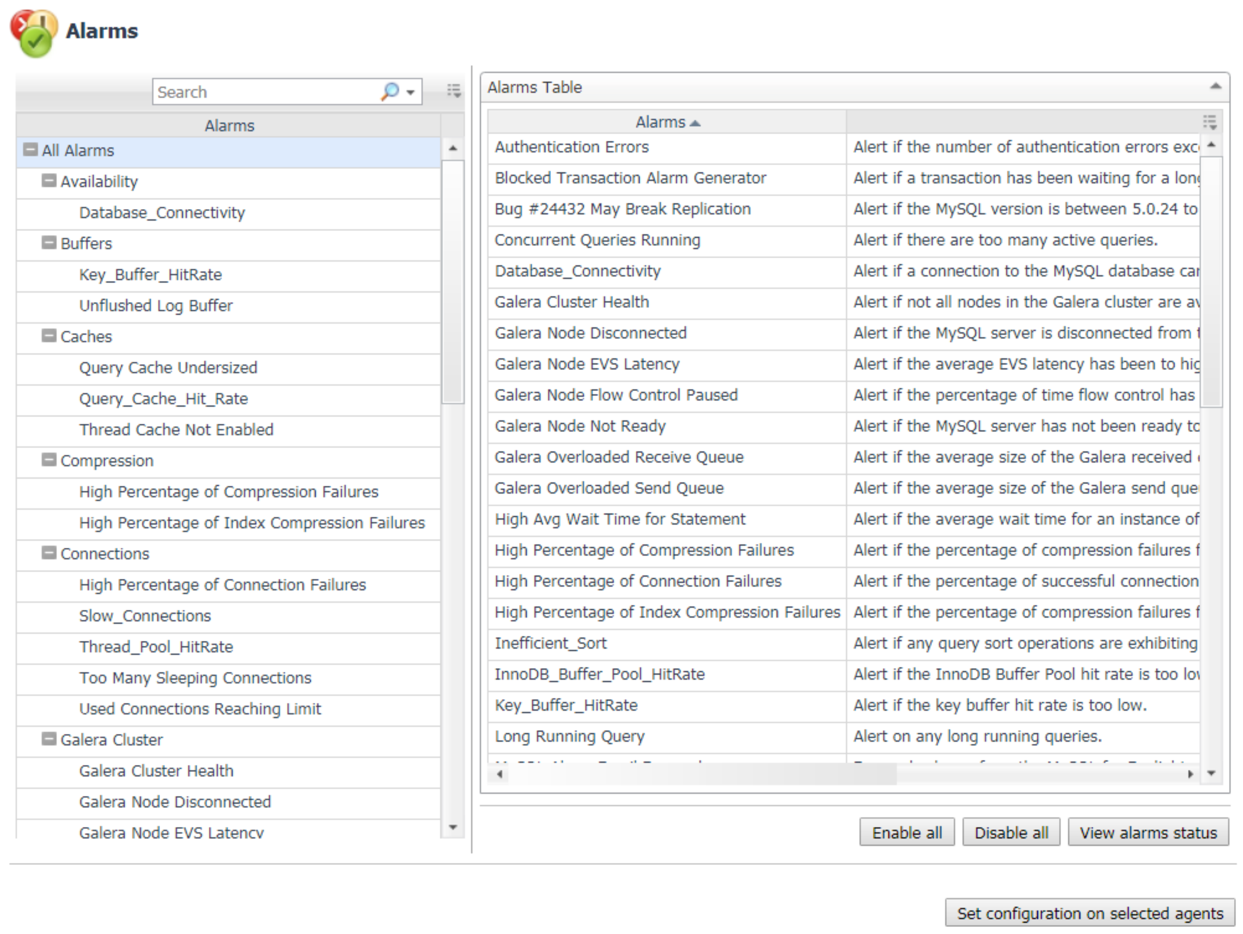
The Alarms view enables you to modify global settings and agent-specific settings for alarms.

To open the Alarms view:

1. Open the Administration dashboard as described in Opening the Databases Administration Dashboard.
2. Select the agents you wish to modify and do one of the following steps:
   a. Select the Settings button and open the Administration dashboard, then click Alarms.
   b. Select the 'Configure Alarm' button.
3. From the Alarms view, you can complete the following tasks:
   a. Modifying Alarm Settings
   b. Reviewing Rule Definitions
   c. Cloning Agent Settings

Modifying Alarm Settings

You can customize how the alarms generated by the default rules are triggered and displayed in the Alarm view. Changes to alarm settings will apply to all selected agents, though thresholds can be customized by individual agent.



Alarms

- All Alarms
  - Availability
    - Database_Connectivity
  - Buffers
    - Key_Buffer_HitRate
    - Unflushed Log Buffer
  - Caches
    - Query Cache Undersized
    - Query_Cache_Hit_Rate
    - Thread Cache Not Enabled
  - Compression
    - High Percentage of Compression Failures
    - High Percentage of Index Compression Failures
  - Connections
    - High Percentage of Connection Failures
    - Slow_Connections
    - Thread_Pool_HitRate
    - Too Many Sleeping Connections
    - Used Connections Reaching Limit
  - Galera Cluster
    - Galera Cluster Health
    - Galera Node Disconnected
    - Galera Node EVS Latency

Alarms Table

Alarms ▲	
Authentication Errors	Alert if the number of authentication errors exc
Blocked Transaction Alarm Generator	Alert if a transaction has been waiting for a long
Bug #24432 May Break Replication	Alert if the MySQL version is between 5.0.24 to
Concurrent Queries Running	Alert if there are too many active queries.
Database_Connectivity	Alert if a connection to the MySQL database car
Galera Cluster Health	Alert if not all nodes in the Galera cluster are av
Galera Node Disconnected	Alert if the MySQL server is disconnected from t
Galera Node EVS Latency	Alert if the average EVS latency has been to hig
Galera Node Flow Control Paused	Alert if the percentage of time flow control has
Galera Node Not Ready	Alert if the MySQL server has not been ready to
Galera Overloaded Receive Queue	Alert if the average size of the Galera received
Galera Overloaded Send Queue	Alert if the average size of the Galera send que
High Avg Wait Time for Statement	Alert if the average wait time for an instance of
High Percentage of Compression Failures	Alert if the percentage of compression failures f
High Percentage of Connection Failures	Alert if the percentage of successful connection
High Percentage of Index Compression Failures	Alert if the percentage of compression failures f
Inefficient_Sort	Alert if any query sort operations are exhibiting
InnoDB_Buffer_Pool_HitRate	Alert if the InnoDB Buffer Pool hit rate is too lo
Key_Buffer_HitRate	Alert if the key buffer hit rate is too low.
Long Running Query	Alert on any long running queries.

The Alarms list controls the contents displayed to the right and the tasks that are available.

- **All Alarms –** Displays all rules with configured alarms and indicates whether alarms are enabled. In this view, you can enable or disable alarms for all the rules at once. You can also set email notifications and define mail server settings.
- **Category of rules –** Displays a set of related rules with configured alarms. In this view, you can enable or disable alarms and also set email notifications for the category of rules.
- **Rule name –** Displays the alarm status for the selected rule. If the rule has multiple severity levels, displays the threshold configured for each severity level. In this view, you can enable or disable the alarm, edit the alarm text, and edit severity levels and their thresholds. You can also set email notifications for the alarm.

You can complete the following tasks:

- Enabling or disabling alarms for selected agents
- Modifying alarm threshold values

- Editing the text of the alarm message

Your changes are saved separately and applied over the default rules. This protects you from software upgrades that may change the underlying default rules.

Enabling or disabling alarms for selected agents

You can override the global alarm sensitivity level setting for the selected agents. You can enable or disable alarms for all rules, a category of rules, or an individual rule.

To see descriptions of the rules, follow the steps described in Reviewing Rule Definitions.

To enable or disable alarms:

1. Navigate to the Alarms view.
2. Decide on the scope for the change: all alarms, a category of rules, or a selected rule.
3. Complete the steps for the selected scope:

Scope	**Procedure**
All alarms	Click **All Alarms**. In the Alarms Settings tab, click either **Enable all** or **Disable all**.
Category of rules	Click a category. Click either **Enable all** or **Disable all**.
Selected rule	Click the rule. In the Alarms Settings tab, click the link that displays the alarm status. Select **Enabled** or **Disabled** from the list and click **Set**.

Modifying alarm threshold values

You can and should modify the thresholds associated with alarms to better suit your environment. If you find that alarms are firing for conditions that you consider to be acceptable, you can change the threshold values that trigger the alarm. You can also enable or disable severity levels to better suit your environment.

When a rule has severity levels, a Threshold section appears in the Alarm Settings tab showing the severity levels and bounds by agent. The threshold values correspond to the lower bounds shown in this table. Many rules do not have severity levels and thresholds.

When editing thresholds, ensure that the new values make sense in context with the other threshold values. For most metrics, threshold values are set so that Warning < Critical < Fatal. However, in metrics where normal performance has a higher value, such as DBSS - Buffer Cache Hit Rate, the threshold values are reversed: Warning > Critical > Fatal.

To change severity levels and thresholds:

1. Navigate to the Alarms view.
2. Click the multiple-severity rule that you want to edit.
3. Click the **Alarms Settings tab.**

4. In the Threshold section, review the defined severity levels and existing threshold bounds for all target agents.
5. Modify the severity levels for one or more agents by following one of the following procedures:

Task	Procedure
Edit severity levels and set threshold values for all agents.	Click **Enhance alarm**. Select the check boxes for the severity levels you want enabled and set the threshold values. Click **Set**.
Change the threshold values for one agent.	Click **Edit** beside the agent name. Set the new threshold values and click **Set**.
Copy the changes made to one agent's threshold values to all other agents.	Click **Edit** beside the agent name that has the values you want to copy. Select **Set for all agents in table** and click **Set**.

Editing the text of the alarm message

For individual rules, you can change the message displayed when an alarm fires. You cannot add or remove the variables used in the message. This is a global setting that affects all agents.

To change the alarm message:

1. In the Alarms view, click the **Settings** tab.
2. Select a rule.
3. Click the **Alarm Settings** tab.
4. Click **Enhance alarm**. A Customize <rule> dialog box opens.
5. In the Message box, edit the message text. To restore the default message, click **Reset message**.
6. Click **Set**.

Reviewing Rule Definitions

If you want to review the conditions of a rule, open the rule in the Rule Management dashboard.

> **IMPORTANT:** Avoid editing rules in the Rule Management dashboard unless you are creating your own rules or copies. These rules may be modified during regular software updates and your edits will be lost.

You can create user-defined rules from the Rule Management dashboard. If you want to modify a rule, we recommend copying the rule and creating a user-defined rule. User-defined rules need to be managed from the Rule Management dashboard; these rules are not displayed in the Alarms view of the Databases Administration dashboard. For help creating rules, open the online help from the Rule Management dashboard.

To open the Rule Management dashboard:

1. On the navigation panel, under **Homes**, click **Administration**.
2. In the Administration dashboard, click **Rules**.
3. Type **MySQL** in the Search field to see the list of predefined rules for MySQL databases. The MySQL rules are displayed. From here, you can review threshold values, alarm counts, and descriptions.
4. To see the full rule definition, click a rule and then click **View and Edit**.
5. In the Rule Detail dialog box, click **Rule Editor**.
6. When you are done with your review, click Rule Management in the bread crumbs to return to the dialog box.
7. Click **Cancel** to avoid changing the rule unintentionally.

Cloning Agent Settings

You may want an agent to have the same settings as another agent. For example, if you add new agents, you may want them to use the same settings as an existing agent. In this case, you can clone the settings from one agent to other agents. This process does not link the agents; in the future if you update the source agent, you also need to update the target agents.

This procedure walks you through selecting the source agent from the Databases dashboard. However, you can also open the Administration dashboard with multiple agents selected. In this case, you select the source agent in Clone Alarm-related Settings to Other Agents dialog box.

To clone alarm-related settings:

1. On the Databases dashboard, select the check box for the agent with the settings you want to clone.
2. Click Settings and then Administration.
3. In the Administration dashboard, click Alarms.
4. Click Set configuration on selected agents. The Clone rule settings across agents dialog box opens.
5. In the Select the source agent drop-down list, you should see the agent you selected.
6. In the Select the target agents table, select the check boxes for agents that should inherit settings from the source agent.
7. Click Apply.
8. When prompted for confirmation, click Yes.

Configuring Email Notifications

We recommend that you set email notifications for the alarms you are most interested in tracking closely. For example, you may want to be notified by email of any Critical or Fatal situation. Or you may want to be informed whenever a key metric is no longer operating within acceptable boundaries.

You can set up email notifications that are generated when an alarm fires and/or on a defined schedule, as described in the following topics:

- Configuring an email server
- Defining Default Email settings
- Enabling or disabling email notifications
- Defining email notifications, recipients, and messages
- Defining variables to contain email recipients

Configuring an email server

You need to define the global mail server variables (connection details) to be used for sending email notifications.

The setting of the email should be configured in Foglight Administration > Email configuration.

Defining Default Email settings

You can define a default email address to be used by every new agent created in the future, by selecting the Default email button when configuring email notification.

The Email addresses entered are applied to all monitored agents not only for the agents that were selected to enter the Alarm administration.

Enabling or disabling email notifications

You can enable or disable email notifications for all alarms, a category of alarms, or a selected rule. Email notifications are sent only if all the following conditions are met:

- The alarm email notification setting is enabled for the affected rule.
- The alarm is triggered by changes in the monitored environment.
- Alarm notification is enabled at the triggered severity level. See Defining email notifications, recipients, and messages.

To enable or disable email notifications:

1. In the Alarms view, click the Settings tab.
2. Decide on the scope for the change: all alarms, a category of rules, or a selected rule.
3. Complete the steps for the selected scope:
    - All alarms - Click All Alarms. Click the Define Email Settings button. Select either Enabled or Disabled from the Alarms notification status list. Click Set.
    - Category of rules - Click a category. Click the Define Email Settings button. Select either Enabled or Disabled from the Alarms notification status list. Click Set.

- Selected rule - Click a rule. In the Alarms Settings tab, click the Define Email Settings tab. Click the link that displays the alarm notification status. Select Enabled or Disabled and click Set.

Defining email notifications, recipients, and messages

You control who receives email messages, the subject line, and some text in the body of the email. The body of the email always contains information about the alarm. This information is not editable. You can also control whether an email is sent based on severity levels. You can set different distribution lists for different rules and different severity levels, or set the same notification policy for all rules.

To configure email notifications:

1. In the Alarms view, click the Settings tab.
2. Decide on the scope for the change: all alarms, a category of rules, or a selected rule.
3. Complete the steps for the selected scope:
   - All alarms - Click All Alarms. Click the Define Email Settings button.
   - Category of rules - Click a category. Click the Define Email Settings button.
   - Selected rule - Click a rule. Click the Email Notification Settings tab.
4. If you selected All Alarms or a category, in the Email Notification Settings dialog box, do one of the following:
   - To change the severity levels that warrant an email notification, from the Messages will be enabled for severities box, select the desired levels of severity.
   - To configure the same email recipients and message for all severity levels, click Configure mail recipients for all Severities and then click All severities.
   - To configure different email recipients and messages for each of the severity levels, click Configure mail recipients for the following options and then click a severity level.
5. In the Message Settings dialog box, configure the email recipients and message. Note that you can use registry variables in place of email addresses. Type the variable name between two hash (#) symbols, for example: #EmailTeamName#. For more information, see Defining variables to contain email recipients.
   - To — Type the addresses of the people who need to take action when this alarm triggers.
   - CC — Type the addresses of the people who want to be notified when the alarm triggers.
   - Subject — Optional. Edit the text of the subject line to better suit your environment. Avoid editing the variables, which are identified with the @ symbol.
   - Body Prefix — Optional. Add text that should appear above the alarm information in the body of the email.
6. Click Set to save the message configuration and close the dialog box.
7. If the Edit Notification Settings dialog box is open, click Set.

Defining variables to contain email recipients

You can create registry variables that contain one or more email addresses, and use these registry variables when defining email notifications. This procedure describes how to create a registry value.

To create a registry variable:

1. On the navigation panel, under Dashboards, click Administration > Rules & Notifications > Manage Registry Variables.

2. Click Add. The New Registry Variable Wizard opens.
3. Select the registry variable type String, and click Next.
4. In the Name field, enter a name, for example: EmailTeamName
5. Click Next.
6. Select Static Value.
7. In the Enter desired value box, enter one or more email addresses (separated by commas).
8. Click Finish.

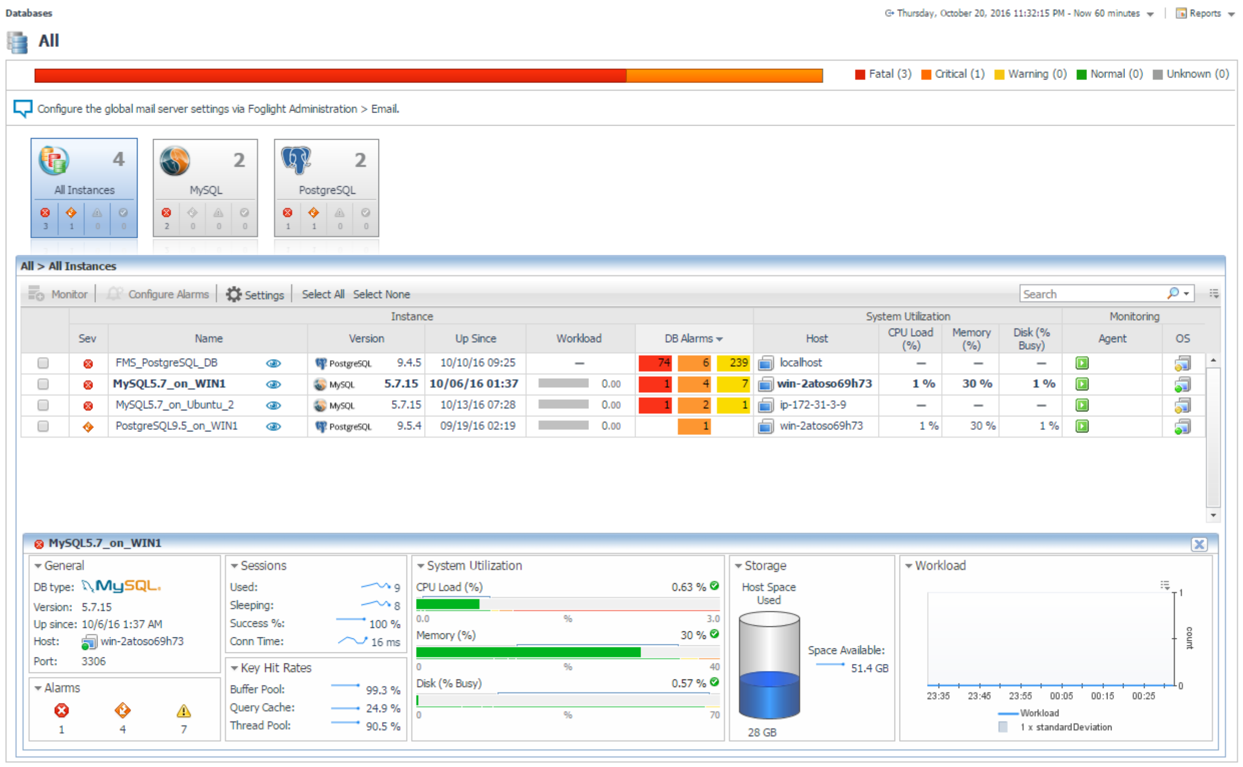
MySQL Dashboards

The installation of the MySQL Cartridge includes the MySQL Dashboards. The MySQL Cartridge offers several principal dashboards:

- Databases
- Global View (deprecated)
- Galera Clusters
- Server Overview
- Server Metrics
- Tables
- Connections
- Statements
- Configuration
- Administration Panel
- Replication Data

Drill downs from these dashboards are available to expose more granular data. It should be noted that there is a great deal of additional information that is collected that is not displayed in these primary dashboards. Should an operator wish to have that information displayed, Foglight allows for the creation of simple drag and drop dashboards.

Databases



Databases

Thursday, October 20, 2016 11:32:15 PM - Now 60 minutes | Reports

All

🟥 Fatal (3) 🟧 Critical (1) 🟨 Warning (0) 🟩 Normal (0) ▨ Unknown (0)

Configure the global mail server settings via Foglight Administration > Email.

All Instances 4	3	1	0	0
MySQL 2	2	0	0	0
PostgreSQL 2	1	1	0	0

All > All Instances

Monitor | Configure Alarms | Settings | Select All Select None | Search

	Instance								System Utilization				Monitoring	
	Sev	Name	Version	Up Since	Workload	DB Alarms			Host	CPU Load (%)	Memory (%)	Disk (% Busy)	Agent	OS
☐	⊗	FMS_PostgreSQL_DB	PostgreSQL 9.4.5	10/10/16 09:25	–	74	6	239	localhost	–	–	–		
☐	⊗	**MySQL5.7_on_WIN1**	**MySQL 5.7.15**	**10/06/16 01:37**	0.00	1	4	7	**win-2atoso69h73**	**1 %**	**30 %**	**1 %**		
☐	⊗	MySQL5.7_on_Ubuntu_2	MySQL 5.7.15	10/13/16 07:28	0.00	1	2	1	ip-172-31-3-9	–	–	–		
☐	◆	PostgreSQL9.5_on_WIN1	PostgreSQL 9.5.4	09/19/16 02:19	0.00		1		win-2atoso69h73	1 %	30 %	1 %		

MySQL5.7_on_WIN1

General

DB type: MySQL
Version: 5.7.15
Up since: 10/6/16 1:37 AM
Host: win-2atoso69h73
Port: 3306

Alarms

1 4 7

Sessions

Used: 9
Sleeping: 8
Success %: 100 %
Conn Time: 16 ms

Key Hit Rates

Buffer Pool: 99.3 %
Query Cache: 24.9 %
Thread Pool: 90.5 %

System Utilization

CPU Load (%) 0.63 %
Memory (%) 30 %
Disk (% Busy) 0.57 %

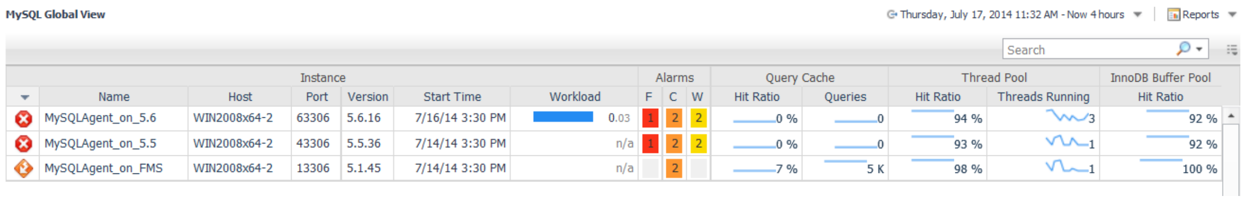
Storage

Host Space Used
Space Available: 51.4 GB
28 GB

Workload

Foglight for MySQL is now incorporated into the Databases dashboard along with any other monitored database types in your environment. Like other products, the list of databases can be filtered by type and severity level and includes basic information, alarms, and host utilization metrics. Clicking on the eye icon will bring up the Quick View with more key information. Clicking the name will drill down to the MySQL Server View.

Global View Dashboard



MySQL Global View

Thursday, July 17, 2014 11:32 AM - Now 4 hours | Reports

Search

	Instance						Alarms			Query Cache		Thread Pool		InnoDB Buffer Pool
	Name	Host	Port	Version	Start Time	Workload	F	C	W	Hit Ratio	Queries	Hit Ratio	Threads Running	Hit Ratio
	MySQLAgent_on_5.6	WIN2008x64-2	63306	5.6.16	7/16/14 3:30 PM	0.03	1	2	2	0 %	0	94 %	3	92 %
	MySQLAgent_on_5.5	WIN2008x64-2	43306	5.5.36	7/14/14 3:30 PM	n/a	1	2	2	0 %	0	93 %	1	92 %
	MySQLAgent_on_FMS	WIN2008x64-2	13306	5.1.45	7/14/14 3:30 PM	n/a		2		7 %	5 K	98 %	1	100 %

The MySQL Global View dashboard displays the following metrics, in row format, for each MySQL agent instance deployed:

- **Instance Group**
  - Health Indicator – Shows the overall health of the deployed instance.
  - Name – The "Instance Name" of the MySQL agent. This is the name entered when the MySQL agent instance was created.
  - Host – Host Name of the MySQL agent instance.
  - MySQL Version – The version of the MySQL database being monitored.

- **Alarms Group**
  - Fatal – Provides a count of the "Fatal" alerts for this agent.
  - Critical – Provides a count of the "Critical" alerts for this agent.
  - Warning – Provides a count of the "Warning" alerts for this agent.

- **Query Cache Group**
  - Hit Ratio – Provides a historical trend and a current value of the Query Cache Hit Ratio for the selected time range.
  - Queries – Provides a historical trend and a current value of the number of SQL statements that are in cache for the selected time range.

- **Thread Pool Group**
  - Hit Ratio – Provides a historical trend and a current value of the ratio of the number of threads that were used from the pool to the total number of connections made for the selected time range.
  - Threads Running – Provides a historical trend and a current value of the number of active (non-sleeping) threads for the selected time range.

- **InnoDB Buffer Pool**
  - Hit Ratio – Provides a historical trend and a current value of the InnoDB BufferPool Hit Ratio for the selected time range.

To get additional information for some of the metrics, the MySQL Global View dashboard provides a dwell capability (by hovering over the metric) and drill down (clicking on the metric) functionality built-in.

The dwell (hovering over the metric) functionality provides a graphical trend for the selected metric over a specified time range

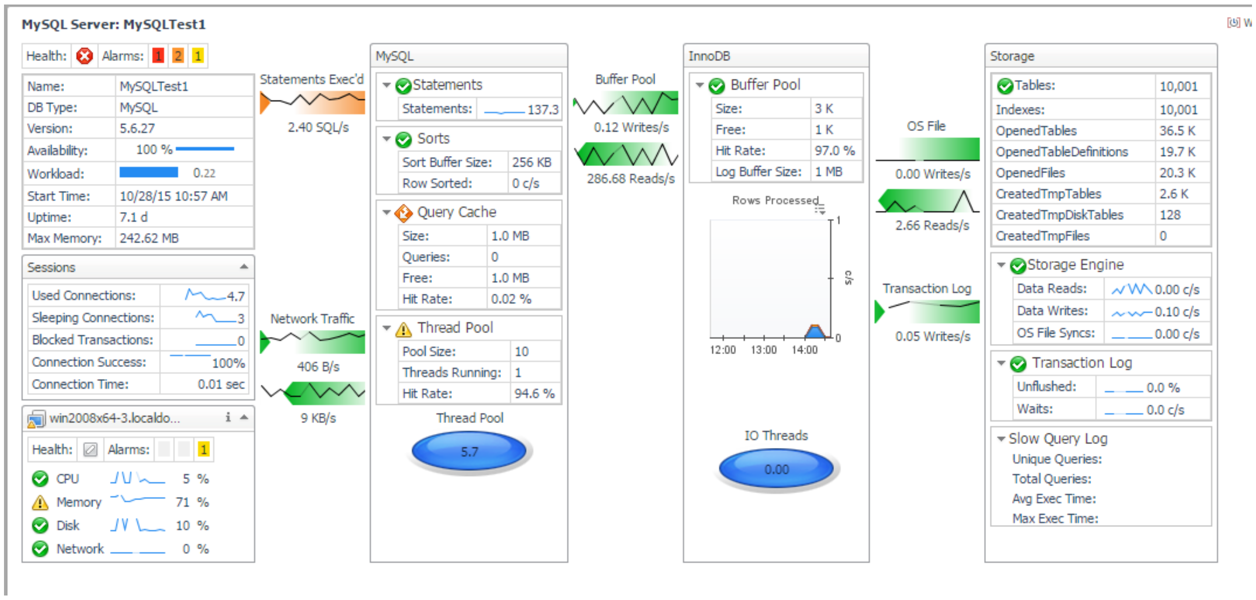
The following metrics in the MySQL Global View dashboard feature the dwell functionality:

- Instance –- Host
- Alarms – Fatal, Critical and Warning (when MySQL alerts are triggered)
- Query Cache – Hit Ratio
- Query Cache – Queries
- Thread Pool – Hit Ratio
- Thread Pool – Threads Running
- InnoDB BufferPool – Hit Ratio

The drill down (clicking on the metric) functionality provides a graphical trend of different metrics for the selected group over a specified time range

The following metrics in the MySQL Global View dashboard feature the drill down functionality:

- Instance (Name) – will drill down to the MySQL Server dashboard. Please see the MySQL Server Dashboard section for more detail information. This dashboard has a "MySQL Agent Selector" option in the right panel.



MySQL Server: MySQLTest1

Health: Alarms: 1 2 1

Name:	MySQLTest1
DB Type:	MySQL
Version:	5.6.27
Availability:	100 %
Workload:	0.22
Start Time:	10/28/15 10:57 AM
Uptime:	7.1 d
Max Memory:	242.62 MB

Sessions

Used Connections:	4.7
Sleeping Connections:	3
Blocked Transactions:	0
Connection Success:	100%
Connection Time:	0.01 sec

win2008x64-3.localdo...

Health: Alarms: 1

CPU	5 %
Memory	71 %
Disk	10 %
Network	0 %

Statements Exec'd 2.40 SQL/s

Network Traffic 406 B/s 9 KB/s

MySQL

Statements

Statements:	137.3

Sorts

Sort Buffer Size:	256 KB
Row Sorted:	0 c/s

Query Cache

Size:	1.0 MB
Queries:	0
Free:	1.0 MB
Hit Rate:	0.02 %

Thread Pool

Pool Size:	10
Threads Running:	1
Hit Rate:	94.6 %

Thread Pool 5.7

Buffer Pool 0.12 Writes/s 286.68 Reads/s

InnoDB

Buffer Pool

Size:	3 K
Free:	1 K
Hit Rate:	97.0 %
Log Buffer Size:	1 MB

Rows Processed

IO Threads 0.00

OS File 0.00 Writes/s 2.66 Reads/s

Transaction Log 0.05 Writes/s

Storage

Tables:	10,001
Indexes:	10,001
OpenedTables	36.5 K
OpenedTableDefinitions	19.7 K
OpenedFiles	20.3 K
CreatedTmpTables	2.6 K
CreatedTmpDiskTables	128
CreatedTmpFiles	0

Storage Engine

Data Reads:	0.00 c/s
Data Writes:	0.10 c/s
OS File Syncs:	0.00 c/s

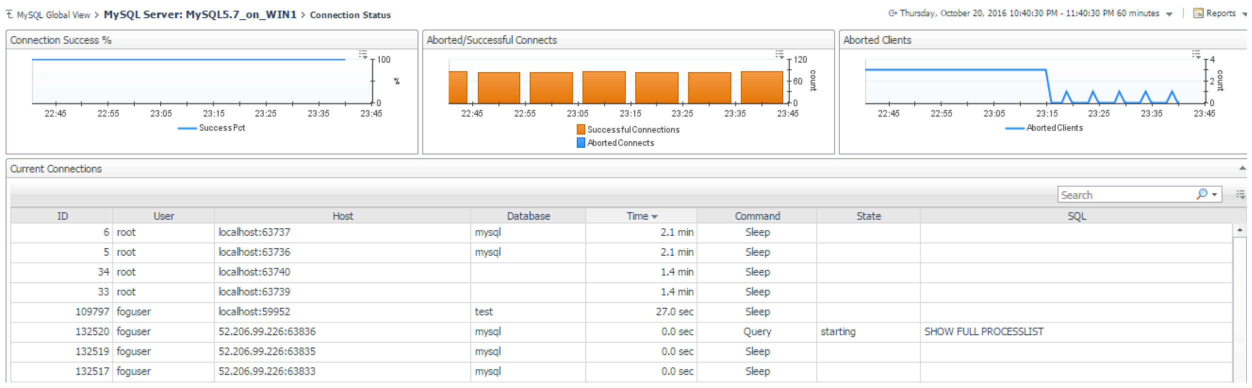
Transaction Log

Unflushed:	0.0 %
Waits:	0.0 c/s

Slow Query Log

Unique Queries:
Total Queries:
Avg Exec Time:
Max Exec Time:

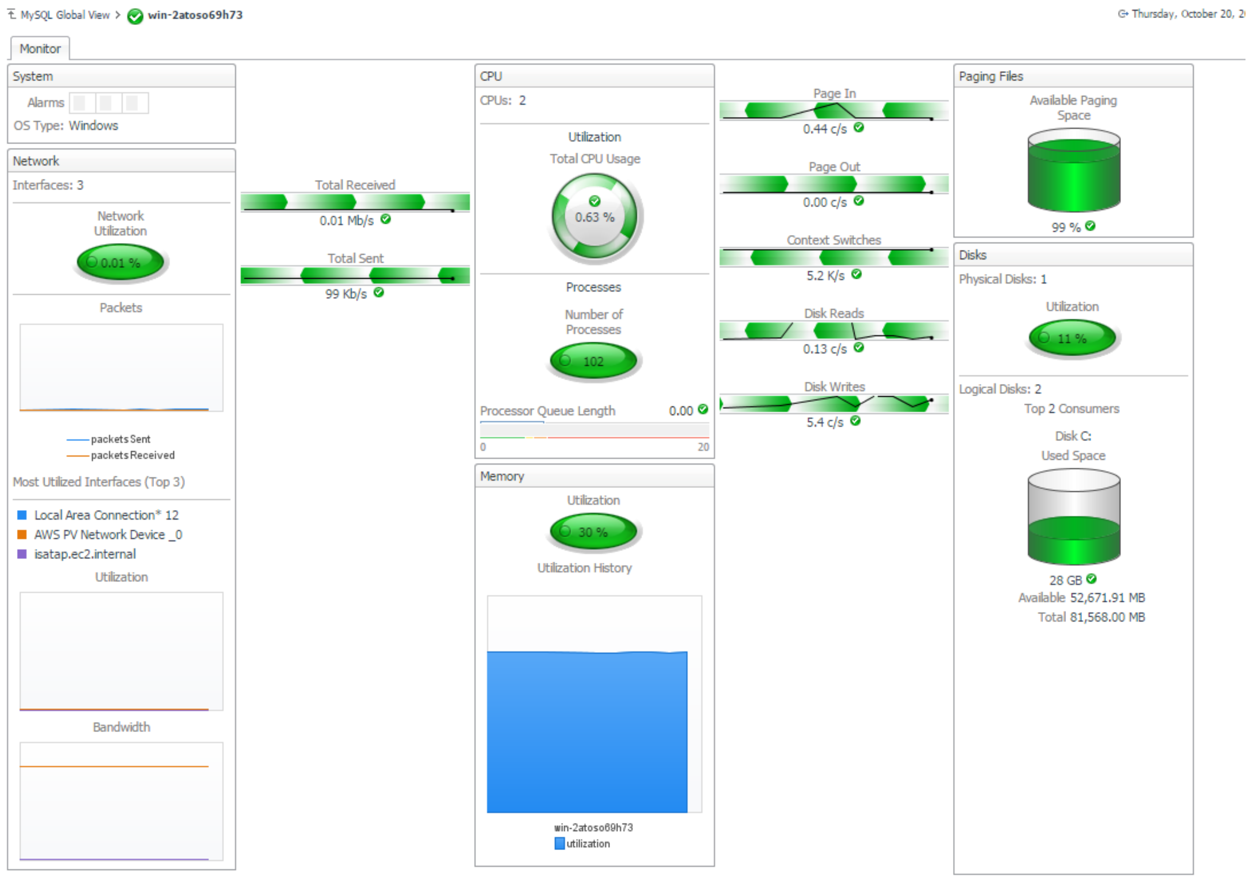
A drill-down from either the Connections or Max Connections component of the MySQL Server view takes the operator to a view of the top sessions and the SQL being executed.



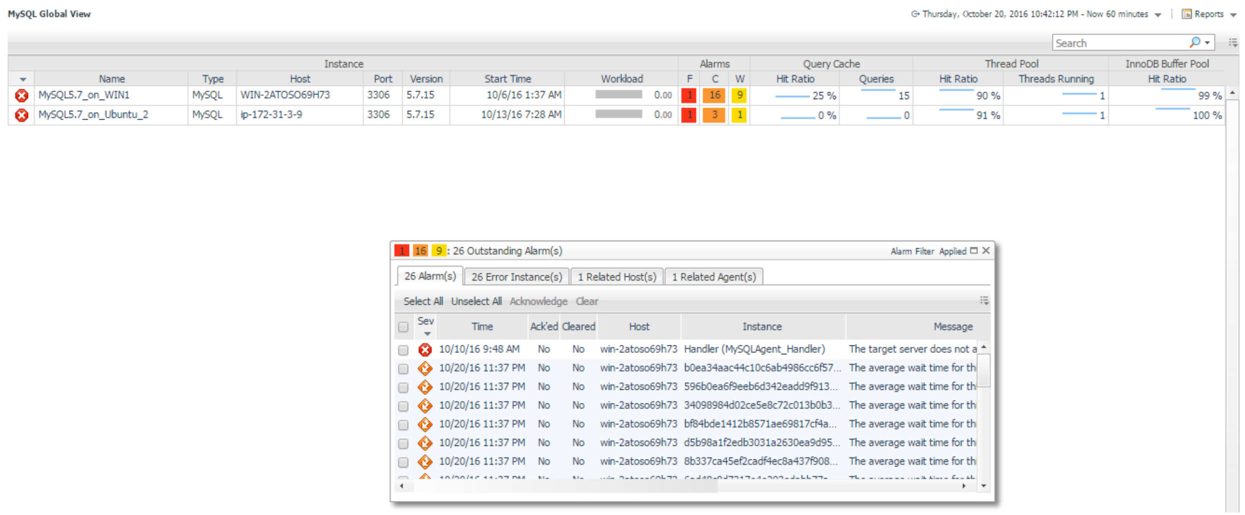
MySQL Global View > MySQL Server: MySQL5.7_on_WIN1 > Connection Status
Thursday, October 20, 2016 10:40:30 PM - 11:40:30 PM 60 minutes
Reports
Connection Success %
Success Pct
Aborted/Successful Connects
Successful Connections
Aborted Connects
Aborted Clients
Current Connections
Search

ID	User	Host	Database	Time	Command	State	SQL
6	root	localhost:63737	mysql	2.1 min	Sleep		
5	root	localhost:63736	mysql	2.1 min	Sleep		
34	root	localhost:63740		1.4 min	Sleep		
33	root	localhost:63739		1.4 min	Sleep		
109797	foguser	localhost:59952	test	27.0 sec	Sleep		
132520	foguser	52.206.99.226:63836	mysql	0.0 sec	Query	starting	SHOW FULL PROCESSLIST
132519	foguser	52.206.99.226:63835	mysql	0.0 sec	Sleep		
132517	foguser	52.206.99.226:63833	mysql	0.0 sec	Sleep		

- Instance (Host) – will drill down to the Host Monitor view for the select “<Host>”. Please see OSCartridges User Guide for more detail information on this dashboard.



- Alarms – Fatal, Critical and Warning will drill down to the MySQL alarms view



MySQL Global View

Thursday, October 20, 2016 10:42:12 PM - Now 60 minutes | Reports

Search

	Instance							Alarms			Query Cache		Thread Pool		InnoDB Buffer Pool
	Name	Type	Host	Port	Version	Start Time	Workload	F	C	W	Hit Ratio	Queries	Hit Ratio	Threads Running	Hit Ratio
●	MySQL5.7_on_WIN1	MySQL	WIN-2ATOSO69H73	3306	5.7.15	10/6/16 1:37 AM	0.00	1	16	9	25 %	15	90 %	1	99 %
●	MySQL5.7_on_Ubuntu_2	MySQL	ip-172-31-3-9	3306	5.7.15	10/13/16 7:28 AM	0.00	1	3	1	0 %	0	91 %	1	100 %

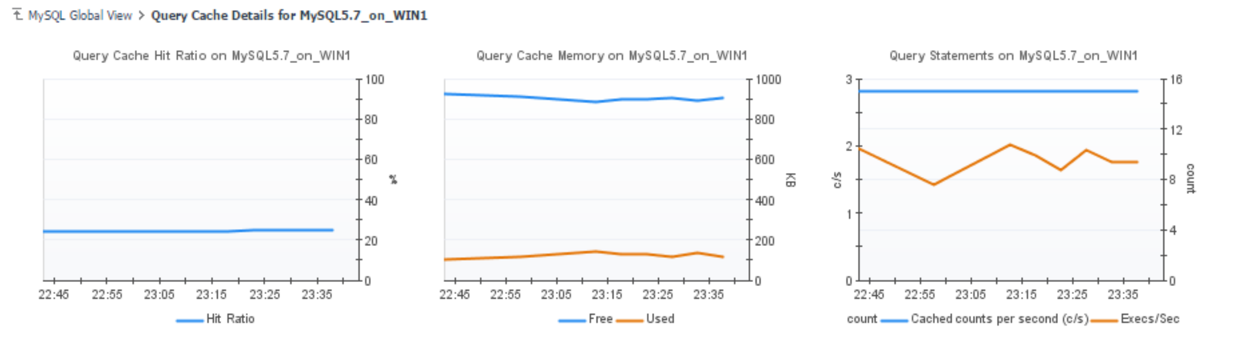
1 16 9 : 26 Outstanding Alarm(s) — Alarm Filter Applied

26 Alarm(s) | 26 Error Instance(s) | 1 Related Host(s) | 1 Related Agent(s)

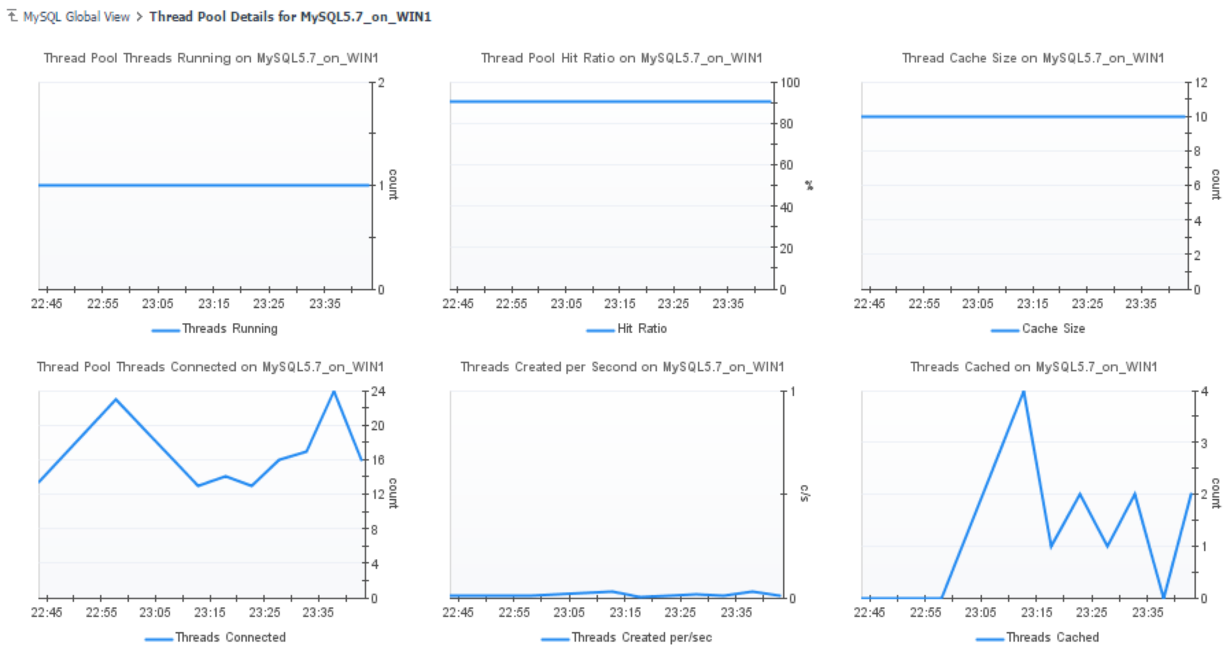
Select All Unselect All Acknowledge Clear

Sev	Time	Ack'ed	Cleared	Host	Instance	Message
●	10/10/16 9:48 AM	No	No	win-2atoso69h73	Handler (MySQLAgent_Handler)	The target server does not a
◆	10/20/16 11:37 PM	No	No	win-2atoso69h73	b0ea34aac44c10c6ab4986cc6f57...	The average wait time for th
◆	10/20/16 11:37 PM	No	No	win-2atoso69h73	596b0ea6f9eeb6d342eadd9f913...	The average wait time for th
◆	10/20/16 11:37 PM	No	No	win-2atoso69h73	34098984d02ce5e8c72c013b0b3...	The average wait time for th
◆	10/20/16 11:37 PM	No	No	win-2atoso69h73	bf84bde1412b8571ae69817cf4a...	The average wait time for th
◆	10/20/16 11:37 PM	No	No	win-2atoso69h73	d5b98a1f2edb3031a2630ea9d95...	The average wait time for th
◆	10/20/16 11:37 PM	No	No	win-2atoso69h73	8b337ca45ef2cadf4ec8a437f908...	The average wait time for th

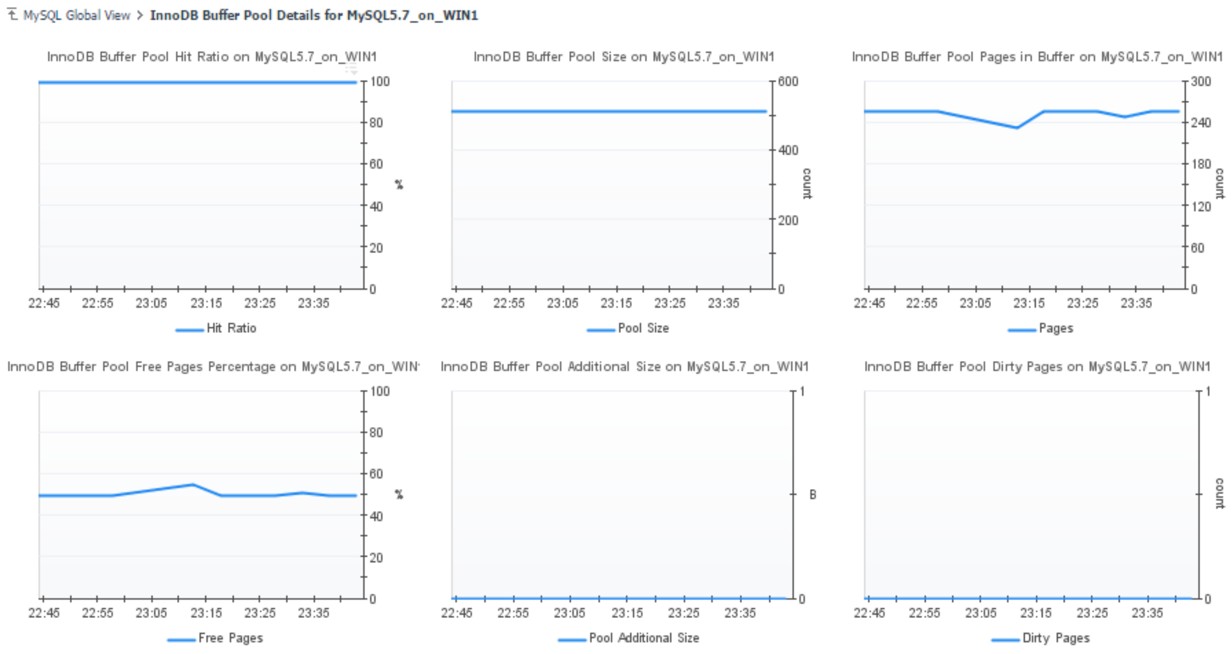
- Query Cache(Hit Ratio and Queries) – will all drill down to the "Query Cache Details for <Instance Name>" view.
    - The Query Cache Detail View provides the following graphs for the selected time range.
        - Query Cache Hit Ratio
        - Query Cache Memory (Free and Used)
        - Query Statements (Caches and Exec/Sec)



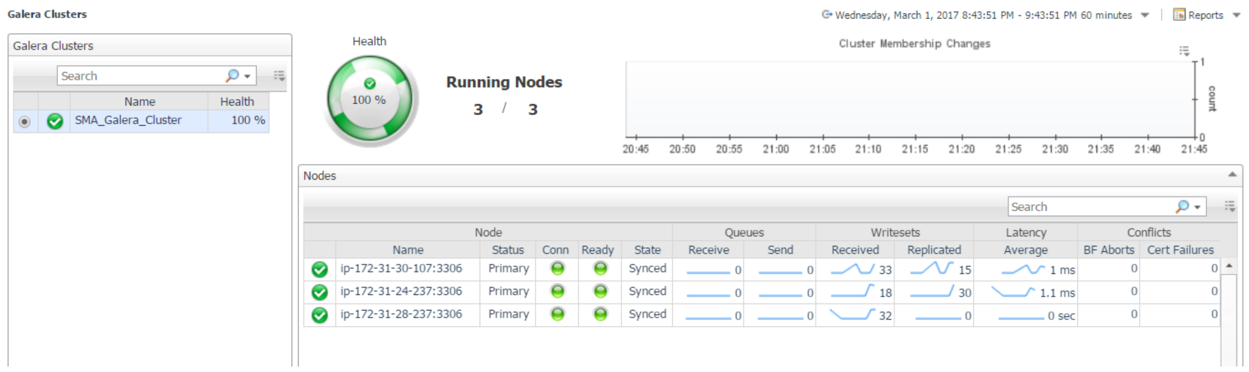
- Thread Pool (Hit Ratio and Threads Running)– will all drill down to the "Thread Pool Details for <Instance Name>" view.
    - The Thread Pool Detail View provides the following graphs for the selected time range.
        - Thread Pool Threads Running
        - Thread Pool Hit Ratio
        - Thread Cache Size
        - Threads Connected
        - Threads Created (per second)
        - Threads Cached


MySQL Global View > Thread Pool Details for MySQL5.7_on_WIN1
Thread Pool Threads Running on MySQL5.7_on_WIN1
Threads Running
Thread Pool Hit Ratio on MySQL5.7_on_WIN1
Hit Ratio
Thread Cache Size on MySQL5.7_on_WIN1
Cache Size
Thread Pool Threads Connected on MySQL5.7_on_WIN1
Threads Connected
Threads Created per Second on MySQL5.7_on_WIN1
Threads Created per/sec
Threads Cached on MySQL5.7_on_WIN1
Threads Cached

- InnoDB BufferPool (Hit Ratio) – will drill down to the “InnoDB Buffer Pool Details for <Instance Name>” view.
    - The Query Cache Detail View provides the following graphs for the selected time range.
        - InnoDB Buffer Pool Hit Ratio
        - InnoDB Buffer Pool Size
        - InnoDB Buffer Pool Pages in Buffer
        - InnoDB Buffer Pool Free Pages (%)
        - InnoDB Buffer Pool Additional Size
        - InnoDB Buffer Pool Dirty Pages


MySQL Global View > InnoDB Buffer Pool Details for MySQL5.7_on_WIN1
InnoDB Buffer Pool Hit Ratio on MySQL5.7_on_WIN1
Hit Ratio
InnoDB Buffer Pool Size on MySQL5.7_on_WIN1
Pool Size
InnoDB Buffer Pool Pages in Buffer on MySQL5.7_on_WIN1
Pages
InnoDB Buffer Pool Free Pages Percentage on MySQL5.7_on_WIN
Free Pages
InnoDB Buffer Pool Additional Size on MySQL5.7_on_WIN1
Pool Additional Size
InnoDB Buffer Pool Dirty Pages on MySQL5.7_on_WIN1
Dirty Pages
%
count
B
22:45
22:55
23:05
23:15
23:25
23:35

Galera Clusters


Galera Clusters
Wednesday, March 1, 2017 8:43:51 PM - 9:43:51 PM 60 minutes
Reports
Health
100 %
Running Nodes
3 / 3
Cluster Membership Changes
count
Nodes
Search

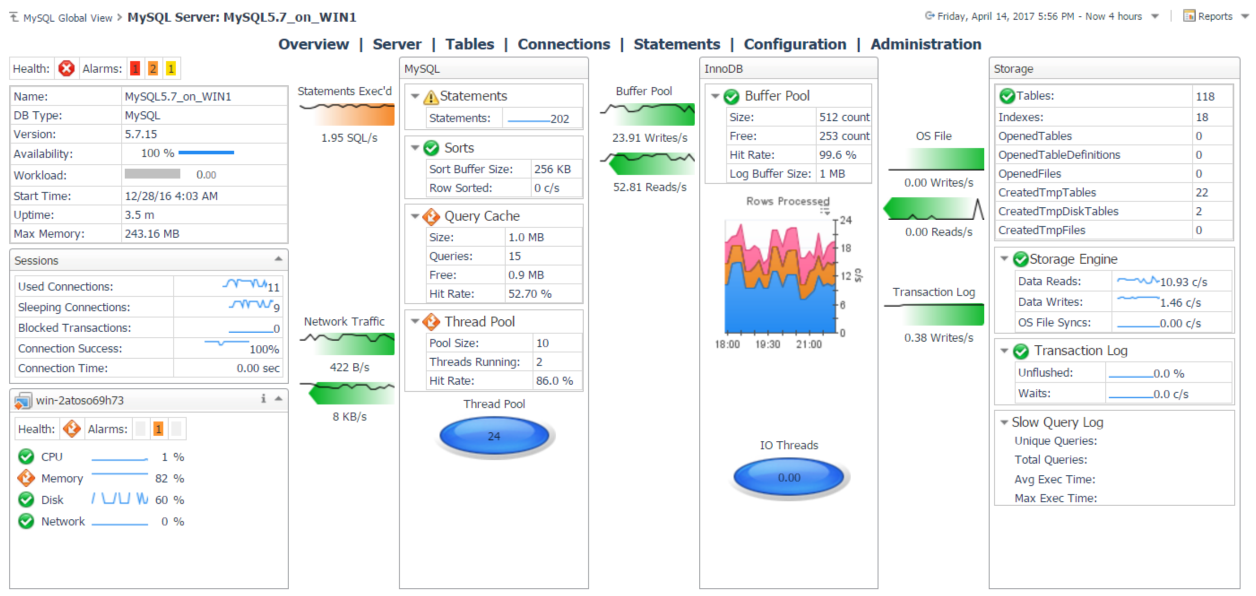
Galera Clusters

		Name	Health
●	✓	SMA_Galera_Cluster	100 %

	Node				Queues		Writesets		Latency	Conflicts	
	Name	Status	Conn	Ready	State	Receive	Send	Received	Replicated	Average	BF Aborts	Cert Failures
✓	ip-172-31-30-107:3306	Primary	●	●	Synced	0	0	33	15	1 ms	0	0
✓	ip-172-31-24-237:3306	Primary	●	●	Synced	0	0	18	30	1.1 ms	0	0
✓	ip-172-31-28-237:3306	Primary	●	●	Synced	0	0	32	0	0 sec	0	0

The Galera Clusters dashboard displays all monitored Galera clusters in your environment. The table on the left lists the Galera clusters and current state and health - the percentage of monitored Galera nodes connected to the cluster. When a cluster is selected, the right panel shows a cluster summary, including a health spinner with a time plot popup, the number of running/monitored nodes, cluster membership changes for the selected time window, and a list of nodes. The nodes table displays each monitored node in the cluster with current states, queue sizes, writeset counts, latency, and conflicts. Clicking on the node name drills down to the Galera Node page.

Server Overview



Name:	MySQL5.7_on_WIN1
DB Type:	MySQL
Version:	5.7.15
Availability:	100 %
Workload:	0.00
Start Time:	12/28/16 4:03 AM
Uptime:	3.5 m
Max Memory:	243.16 MB

Sessions

Used Connections:	11
Sleeping Connections:	9
Blocked Transactions:	0
Connection Success:	100%
Connection Time:	0.00 sec

win-2atoso69h73

Health: Alarms: 1

CPU	1 %
Memory	82 %
Disk	60 %
Network	0 %

MySQL

Statements	
Statements:	202

Sorts	
Sort Buffer Size:	256 KB
Row Sorted:	0 c/s

Query Cache	
Size:	1.0 MB
Queries:	15
Free:	0.9 MB
Hit Rate:	52.70 %

Thread Pool	
Pool Size:	10
Threads Running:	2
Hit Rate:	86.0 %

InnoDB

Buffer Pool	
Size:	512 count
Free:	253 count
Hit Rate:	99.6 %
Log Buffer Size:	1 MB

Storage

Tables:	118
Indexes:	18
OpenedTables	0
OpenedTableDefinitions	0
OpenedFiles	0
CreatedTmpTables	22
CreatedTmpDiskTables	2
CreatedTmpFiles	0

Storage Engine	
Data Reads:	10.93 c/s
Data Writes:	1.46 c/s
OS File Syncs:	0.00 c/s

Transaction Log	
Unflushed:	0.0 %
Waits:	0.0 c/s

Slow Query Log
Unique Queries:
Total Queries:
Avg Exec Time:
Max Exec Time:

The MySQL Server dashboard provides detail information for the MySQL instance being monitored.

This dashboard can be accessed in two ways.

- It is available under the “MySQL” branch in the navigation panel on the left.

- It is also accessible by drilling down on “Name” under the “Instance” section in the MySQL Global View dashboard.

The MySQL Server dashboard displays metrics, in a Spotlight-like format, for each MySQL agent instance deployed. There also exists an agent selector in the right action panel under the general tab so that other deployed instance information can also be accessed through this dashboard.

The following metrics are being collected by the MySQL Server dashboard.

- **Database Information**
  - Name – The name of the database being monitored.
  - Version – The version of the MySQL sever being monitored.
  - Availability – The availability of the MySQL instance being monitored.
  - Uptime – The time that the database has been up.

- **Sessions**
  - Used Connections – The number of current connections to the MySQL server.
  - Sleeping Connections – The number of used connections which are in a sleep state.
  - Blocked Transactions – The number of current transactions that appear to be in a deadlocked state (available in 5.5+).

        - Connection – The number of connection attempts (successful or not) to MySQL server.
        - Connection Success – The percentage of successful connections made to the server in the last sample period.
        - Connection Time – The time it took MySQL took fully establish the connection.

    - **Host Summary**
        - Health – Health indicator for the overall health of the host.
        - Alarms – Number of alerts triggered on the host.
        - CPU – Historical trend and current value of the CPU on the monitored host.
        - Memory – Historical trend and current value of the Memory on the monitored host.
        - Disk – Historical trend and current value of the Disk Utilization on the monitored host.
        - Network – Historical trend and current value of the Network usage on the monitored host.

- **MySQL**
    - **Sorts**
        - Sort Buffer Size – Buffer size that each thread needs to allocate to do a sort.
        - Rows Sorted – The number of rows sorted per second.

    - **Query Cache**
        - Size – Query Cache Size of memory.
        - Queries – The number of SQL statements that are in the cache.
        - Free – Memory in the query cache that are free.
        - Hit Rate – The % of SQL queries that were in the query cache.

    - **Thread Pool**
        - Pool Size – The maximum number of threads that can be cached.
        - Threads Running – The number of active (non sleeping) threads.
        - Hit Rate – The number of threads that were used from the pool in comparison to the total number of connections made.

    - **Thread Pool** – The number of currently open connections. When hovering over Thread Pool, it will provide historical trend for the selected time rang.

- **InnoDB**
    - **Buffer Pool**
        - Size – Size of InnoDB buffer pool
        - Free – Size of InnoDB free buffers
        - Hit Rate – Percent buffer hit rate
        - Log Buffer Size – Size of the InnoDB Log Buffer

    - **Rows Processed** – When hovering over Rows Processed, it will provide historical trend for the selected time range.The graph provides the following:
        - RowsInsertedPerSec – Number of rows inserted per second.

        - RowsReadPerSec – Number of rows read per second.
        - RowsUpdatedPerSec – Number of rows updated per second.
        - RowsDeletedPerSec – Number of rows deleted per second.

    - **IO Threads** – The number of helper threads performing InnoDB I/O. When hovering over IO Threads, it will provide historical trend for the selected time range.

- **Storage**
    - **Tables** – Provides the number of tables in the MySQL instance being monitored.

    - **Storage Engine**
        - Data Reads – Number of physical data reads per second
        - Data Writes – Number of physical data writes per second
        - OS File Syncs – Number of Operating System fsync calls per second

    - **Transaction Log**
        - Unflushed – Percent of InnoDB log buffer that has yet to be flushed to disk. If this is too high then buffer needs to be tuned.
        - Waits – Number of times InnoDB has to wait for log to be flushed to disk per second.
    - **Slow Query Log**
        - Unique Queries – Number of unique queries in the current collection.
        - Total Queries – Total number of queries in the current collection.
        - Avg Execution Time – Average length of time for queries to fully execute in the current collection.
        - Max Execution Time – Maximum length of time for queries to fully execute in the current collection.

The following flows exist in the MySQL Server dashboard. When hovering over each flow, it will provide a historical trend for the selected time range.

- Flows to/from the **Connection Information column** and the **MySQL column**:
    - StatementExecPerSec – The number of SQL statements per second that are currently being executed.
    - BytesReceivedPerSec – The number of bytes received per second.
    - BytesSentPerSec – The number of bytes sent per second.

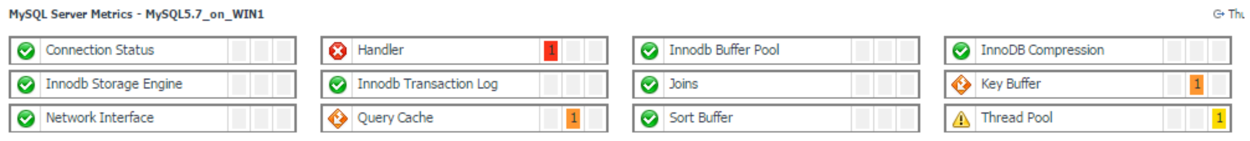
- Flows to/from the **MySQL column** and the **InnoDB column**:
    - BufferPoolWriteReq – The number of logical write requests done to the InnoDB buffer pool.
    - BufferPoolReadReq – The number of logical read requests done to the InnoDB buffer pool.

- Flows to/from the **InnoDB column** and the **Storage column**:
    - OSFileWritesPerSec – Number of Operating System writes per second.
    - OSFileReadsPerSec – Number of Operating System reads per second.
    - WritesPerSec – Number of times InnoDB has written the log to disk per second.

Server Metrics

The Server Metrics Dashboard is accessible through either the navigation panel or by clicking on the Sorts, Query Cache, Thread Pool, Buffer Pool, or Storage Engine table data in the MySQL Server Dashboard. When accessing the Server Metrics dashboard from the MySQL Server dashboard the selected tab will default to the same category as the table that was clicked.

Server Metrics Health Overview

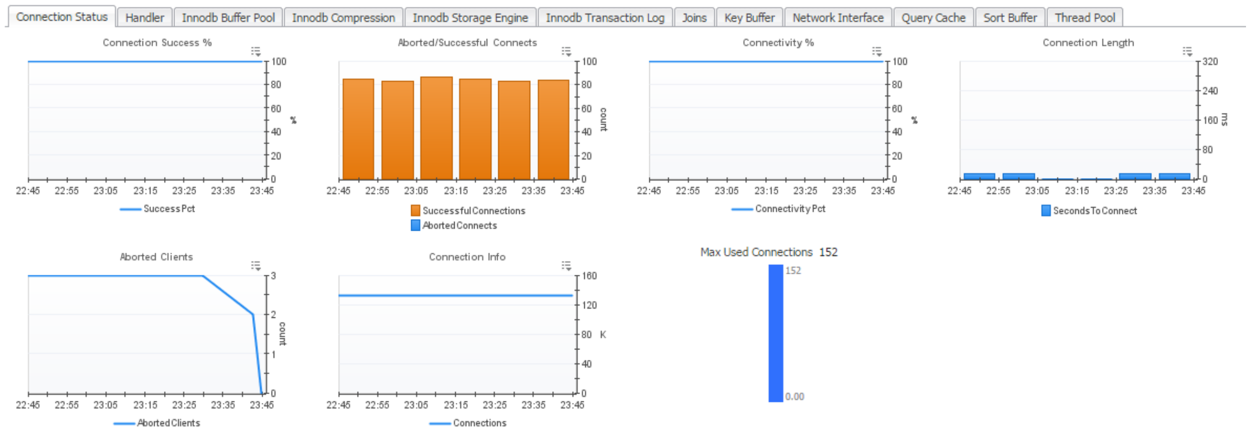


MySQL Server Metrics - MySQL5.7_on_WIN1

Connection Status	Handler 1	Innodb Buffer Pool	InnoDB Compression
Innodb Storage Engine	Innodb Transaction Log	Joins	Key Buffer 1
Network Interface	Query Cache 1	Sort Buffer	Thread Pool 1

Located at the top of the Server Metrics page, overview boxes show the current health state and all active alarms for each component of the agent's data collection. Clicking or dwelling over the health icon or outstanding alarms will bring up additional information related to the state of that component.

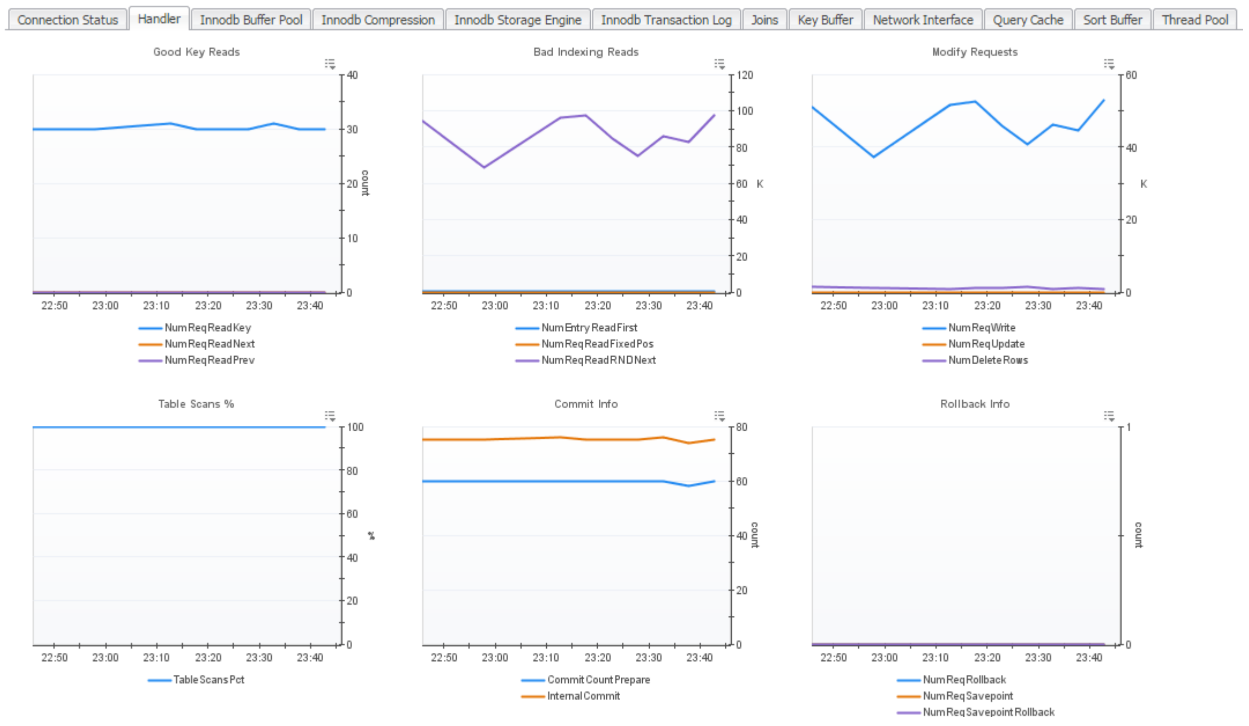
Connection Status



Charts and Metrics (left to right):

- **Max Used Connections** - The maximum number of connections that have been in use simultaneously since the server started.
- **Connection Info** - The number of connection attempts (successful or not) to the MySQL server.
- **Connectivity %** - The probability of making a successful new connection.
- **Connection Length** – The number of seconds MySQL took to establish a new connection.

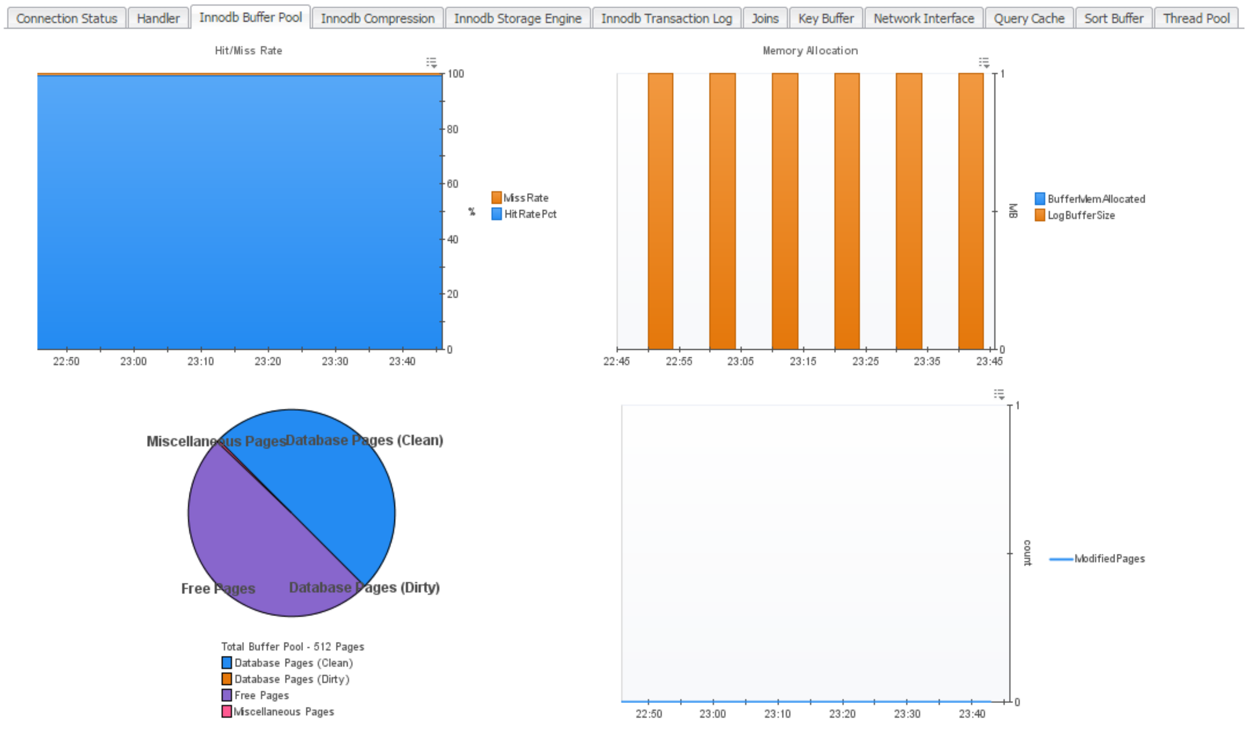
Handler



Charts and Metrics (left to right, top to bottom):

- **Good Key Reads** – Number of reads employing table keys, usually indicative of good indexing.
- **Bad Indexing Reads** – Number of reads which typically indicate bad indexing.
- **Modify Requests** – Requests related to modifying table data through writing, updating, or deleting rows.
- **Commit Info** – MySQL metrics related to transaction commits.
- **Rollback Info** - MySQL metrics related to transaction rollbacks.

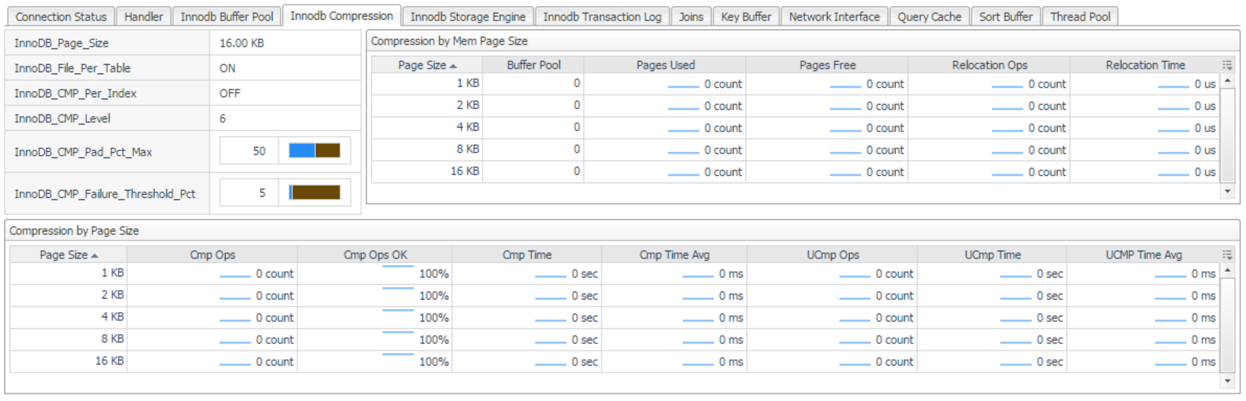
Innodb Buffer Pool



Charts and Metrics (left to right, top to bottom):

- **Hit/Miss Rate** – Hit and Miss rate percentages for InnoDB Buffer Pool.
- **Memory Allocation** – Amount of memory allocated to the InnoDB Buffer Pool and Log Buffer Size.
- **Buffer Pool Pages** – Breakdown of Buffer Pages by type.
- **Modified Pages** – Number of Buffer Pages being modified.

Innodb Compression



InnoDB_Page_Size	16.00 KB
InnoDB_File_Per_Table	ON
InnoDB_CMP_Per_Index	OFF
InnoDB_CMP_Level	6
InnoDB_CMP_Pad_Pct_Max	50
InnoDB_CMP_Failure_Threshold_Pct	5

Compression by Mem Page Size

Page Size	Buffer Pool	Pages Used	Pages Free	Relocation Ops	Relocation Time
1 KB	0	0 count	0 count	0 count	0 us
2 KB	0	0 count	0 count	0 count	0 us
4 KB	0	0 count	0 count	0 count	0 us
8 KB	0	0 count	0 count	0 count	0 us
16 KB	0	0 count	0 count	0 count	0 us

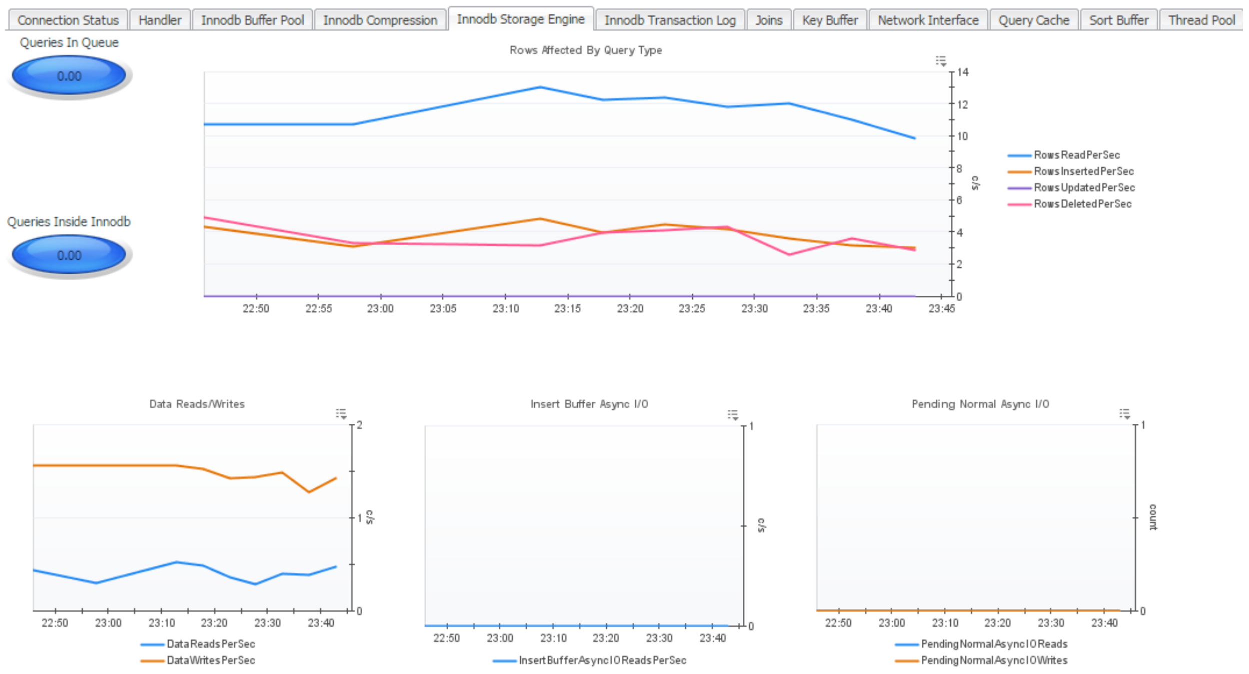
Compression by Page Size

Page Size	Cmp Ops	Cmp Ops OK	Cmp Time	Cmp Time Avg	UCmp Ops	UCmp Time	UCMP Time Avg
1 KB	0 count	100%	0 sec	0 ms	0 count	0 sec	0 ms
2 KB	0 count	100%	0 sec	0 ms	0 count	0 sec	0 ms
4 KB	0 count	100%	0 sec	0 ms	0 count	0 sec	0 ms
8 KB	0 count	100%	0 sec	0 ms	0 count	0 sec	0 ms
16 KB	0 count	100%	0 sec	0 ms	0 count	0 sec	0 ms

Charts and Metrics (left to right, top to bottom):

- **Properties -** InnoDB compression-related variables
- **Compression by Mem Page Size** – Compression metrics grouped by memory page size.
- **Compression by Page Size** – Compression metrics grouped by page size.

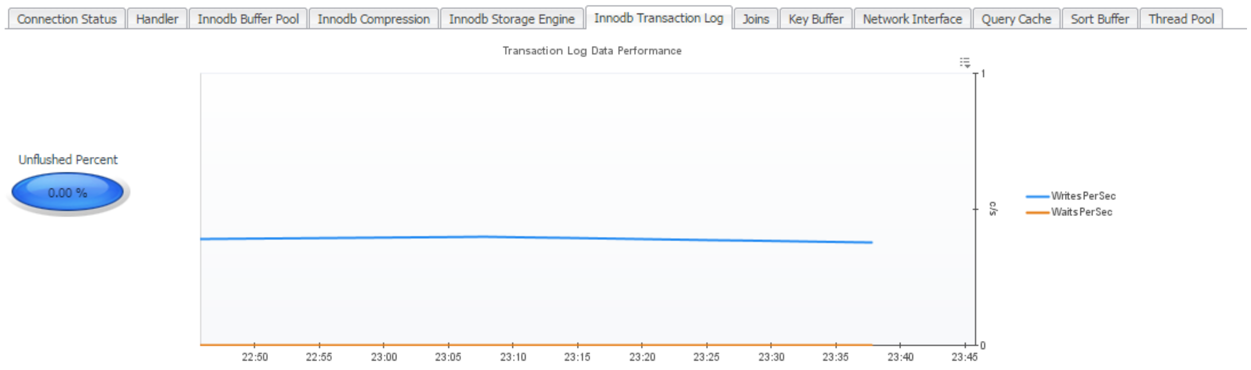
Innodb Storage Engine



Charts and Metrics (left to right, top to bottom):

- **Queries in Queue** – The number of queries waiting to enter InnoDB.
- **Queries inside InnoDB** – The number of queries active inside InnoDB.
- **Rows Affected by Query Type** – The amount of rows per second which are being affected, shown according to the type of query affecting them.
- **Data Reads/Writes** – The number of physical data reads and writes per second.
- **Insert Buffer Async I/O** – The number of insert buffer asynchronous I/O reads and writes per second.
- **Pending Normal Async I/O** – The number of pending normal asynchronous I/O reads and writes per second.
- **Mutex Info** – Information relating to InnoDB waiting for a mutex.
- **RW Locks** – Information relating to InnoDB waiting for a read/write lock to be released.
- **OS File I/O** – The rate of Operating System reads, writes, and fsync operations per second.
- **Transactions Rate** – The InnoDB transaction rate graphed over time.
- **Transaction Purge Lag** – The transaction purge lag graphed over time.

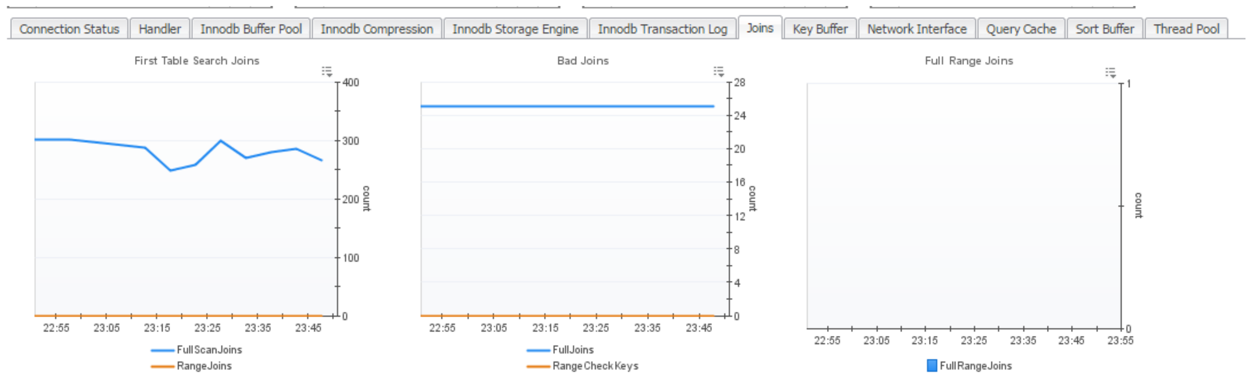
InnoDB Transaction Log



Charts and Metrics (left to right):

- **Unflushed Percent** – The percent of InnoDB log buffer that has yet to be flushed to disk.
- **Transaction Log Data Performance** – The rate of writes and waits for the Transaction Log to be flushed to disk.

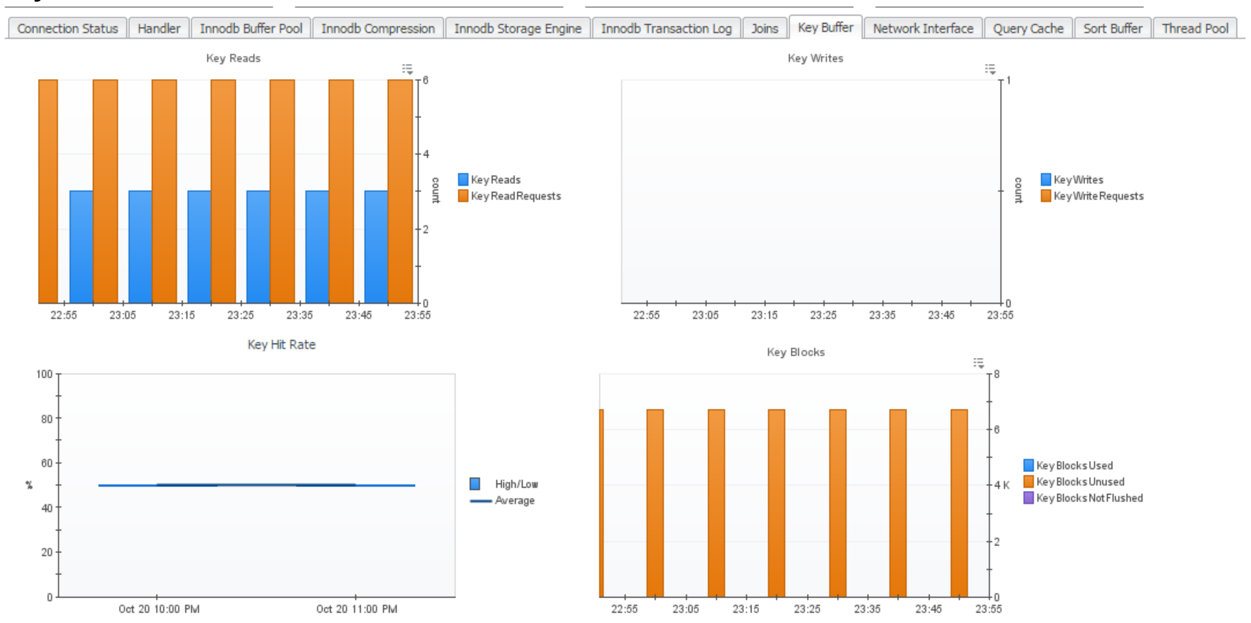
Joins



Charts and Metrics (left to right):

- **First Table Search Joins** – The number of joins operating on the first table.
- **Bad Joins** – Inefficient joins which do not use keys or indexes.
- **Full Range Joins** – The number of joins that used a range search on a reference table.

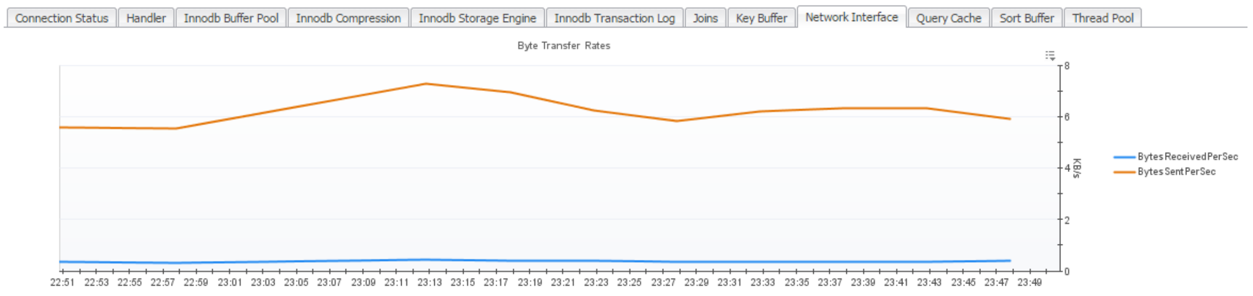
Key Buffer



Charts and Metrics (left to right, top to bottom):

- **Key Reads** – The number of reads or read requests for a key block.
- **Key Writes** – The number of write or write requests for a key block.
- **Key Hit Rate** – The rate of physical reads from a disk to requests to read from the cache.
- **Key Blocks** – A status breakdown of key blocks in the key caches.

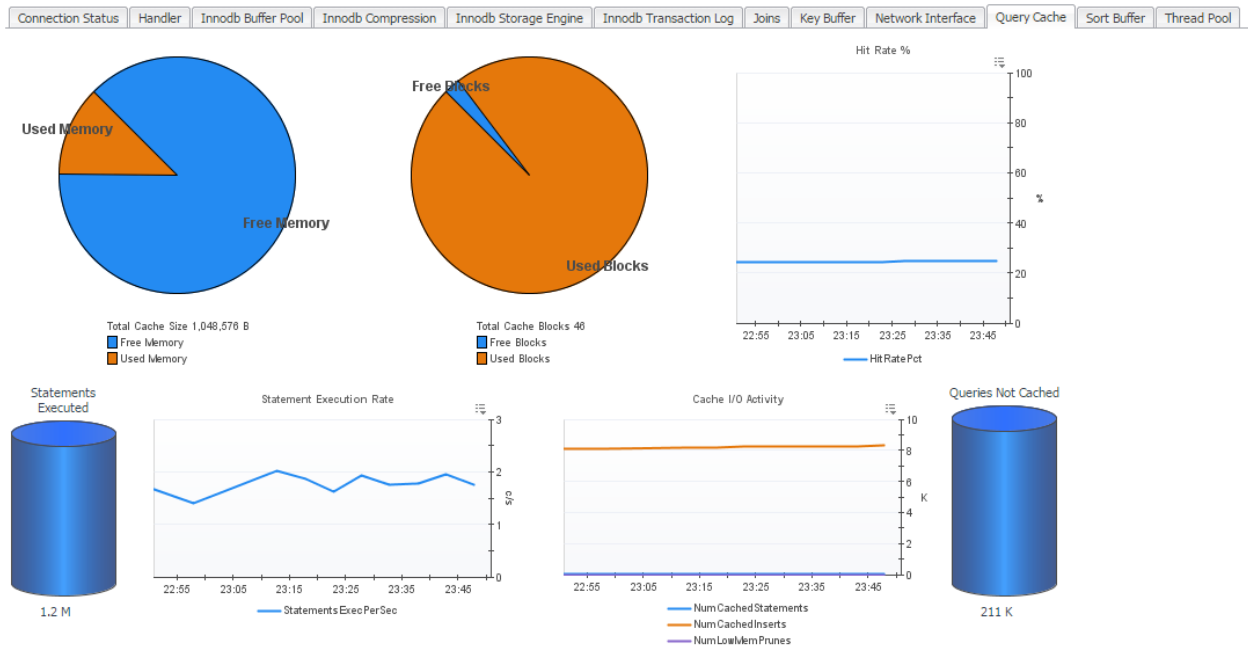
Network Interface



Charts and Metrics (left to right):

- **Byte Transfer Rates** - The number of bytes received and sent per second from and to the network

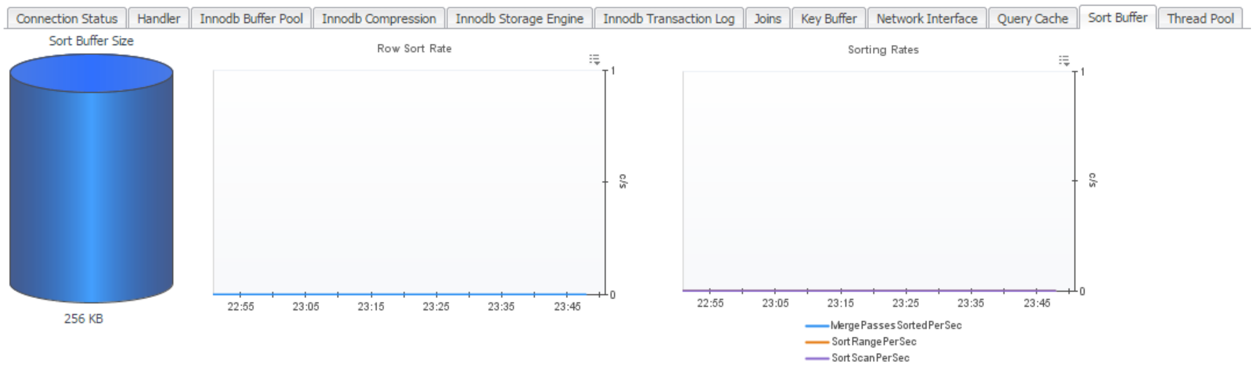
Query Cache



Charts and Metrics (left to right, top to bottom):

- **Cache Memory Usage** – A breakdown of the total cache size into free and used memory.
- **Cache Memory Blocks** – A breakdown of the total cache blocks into free and used blocks.
- **Hit Rate %** - The percent of SQL queries used from the Query Cache.
- **Statements Executed** – The number of SQL statements currently being executed.
- **Statement Execution Rate** – The number of SQL statements per second that are currently being executed.
- **Cache I/O Activity** – The number of statements being added to the cache, existing in the cache, and being dropped from the cache.
- **Queries Not Cached** – The number of non-cached queries.

Sort Buffer

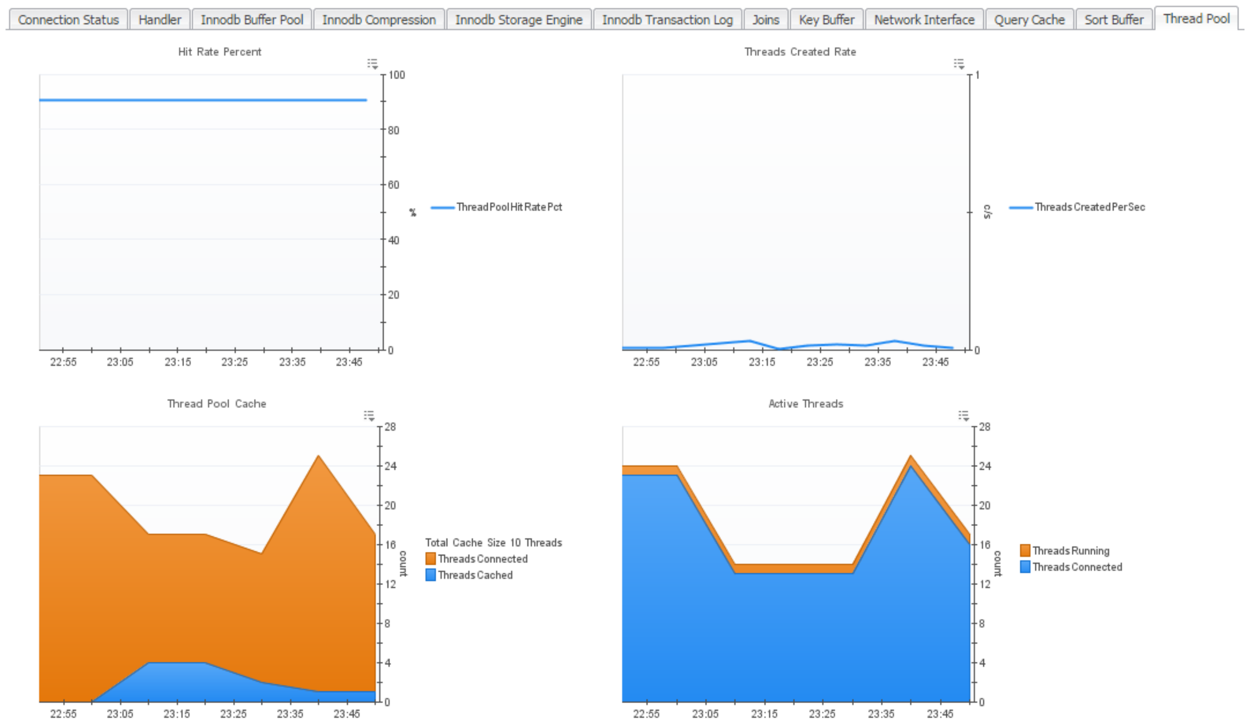


Charts and Metrics (left to right):

- **Sort Buffer Size** – The buffer size that each thread needs to allocate to do a sort.

- **Row Sort Rate** – The number of rows being sorted per second.
- **Sorting Rates** – The sorting rates per second broken down by type of sort.

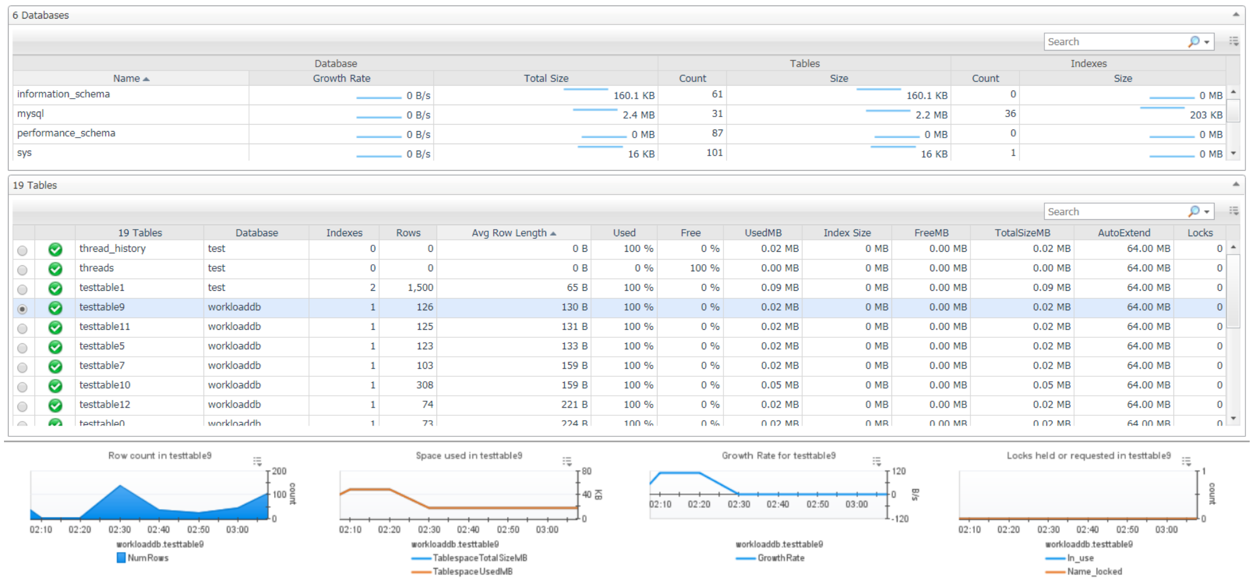
Thread Pool



Charts and Metrics (left to right, top to bottom):

- **Hit Rate Percent** – The number of threads that were used from the pool in comparison to the total number of connections made.
- **Threads Created Rate** – The number of new threads created per second.
- **Thread Pool Cache** – The number of threads in the Thread Pool and currently open connections.
- **Active Threads** – The number of currently open connections and active threads.

Tables



6 Databases

	Database		Tables		Indexes	
Name ▲	Growth Rate	Total Size	Count	Size	Count	Size
information_schema	0 B/s	160.1 KB	61	160.1 KB	0	0 MB
mysql	0 B/s	2.4 MB	31	2.2 MB	36	203 KB
performance_schema	0 B/s	0 MB	87	0 MB	0	0 MB
sys	0 B/s	16 KB	101	16 KB	1	0 MB

19 Tables

		19 Tables	Database	Indexes	Rows	Avg Row Length ▲	Used	Free	UsedMB	Index Size	FreeMB	TotalSizeMB	AutoExtend	Locks
○	✓	thread_history	test	0	0	0 B	100 %	0 %	0.02 MB	0 MB	0.00 MB	0.02 MB	64.00 MB	0
○	✓	threads	test	0	0	0 B	0 %	100 %	0.00 MB	0 MB	0.00 MB	0.00 MB	64.00 MB	0
○	✓	testtable1	test	2	1,500	65 B	100 %	0 %	0.09 MB	0 MB	0.00 MB	0.09 MB	64.00 MB	0
●	✓	testtable9	workloaddb	1	126	130 B	100 %	0 %	0.02 MB	0 MB	0.00 MB	0.02 MB	64.00 MB	0
○	✓	testtable11	workloaddb	1	125	131 B	100 %	0 %	0.02 MB	0 MB	0.00 MB	0.02 MB	64.00 MB	0
○	✓	testtable5	workloaddb	1	123	133 B	100 %	0 %	0.02 MB	0 MB	0.00 MB	0.02 MB	64.00 MB	0
○	✓	testtable7	workloaddb	1	103	159 B	100 %	0 %	0.02 MB	0 MB	0.00 MB	0.02 MB	64.00 MB	0
○	✓	testtable10	workloaddb	1	308	159 B	100 %	0 %	0.05 MB	0 MB	0.00 MB	0.05 MB	64.00 MB	0
○	✓	testtable12	workloaddb	1	74	221 B	100 %	0 %	0.02 MB	0 MB	0.00 MB	0.02 MB	64.00 MB	0
○	✓	testtable0	workloaddb	1	73	224 B	100 %	0 %	0.02 MB	0 MB	0.00 MB	0.02 MB	64.00 MB	0

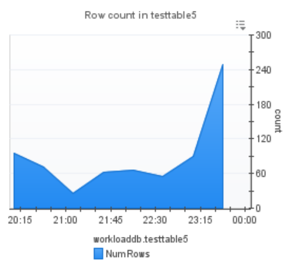
The Tables dashboard provides pertinent table information for the MySQL instance being monitored.

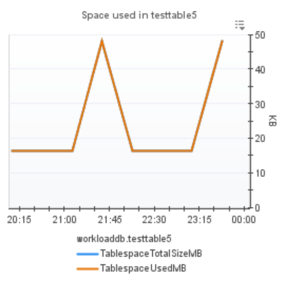
The Databases table shows aggregated metrics for tables in that database, including growth rate and table and index counts and sizes.

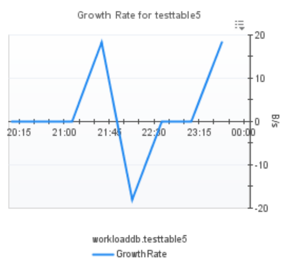
The Tables table displays the following metrics, in row format, for each MySQL agent instance deployed:

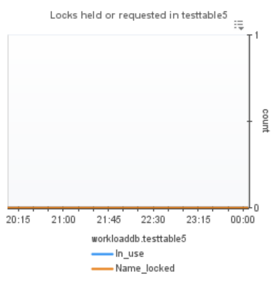
- **Selector button** – used to select a specific table for the graphs below.
- **Health Indicator** – Indicates the overall health of the deployed instance.
- **<Count> Tables** – provides the name and a total count of tables in the MySQL instance being monitored
- **Indexes** – The number of indexes in the table
- **Rows** – The number of rows in the table
- **Avg Row Length** – Average size of rows in the table
- **Used** – Percentage of used space in the table
- **Free** – Percentage of free space in the table
- **UsedMB** – Megabytes of used space in the table
- **FreeMB** – Megabytes of free space in the table
- **TotalSizeMB** – Megabytes of total size in the table
- **AutoExtend** – The increment size for extending the size of an auto-extending table when it becomes full.

The graphs below provide the following information for the selected table.



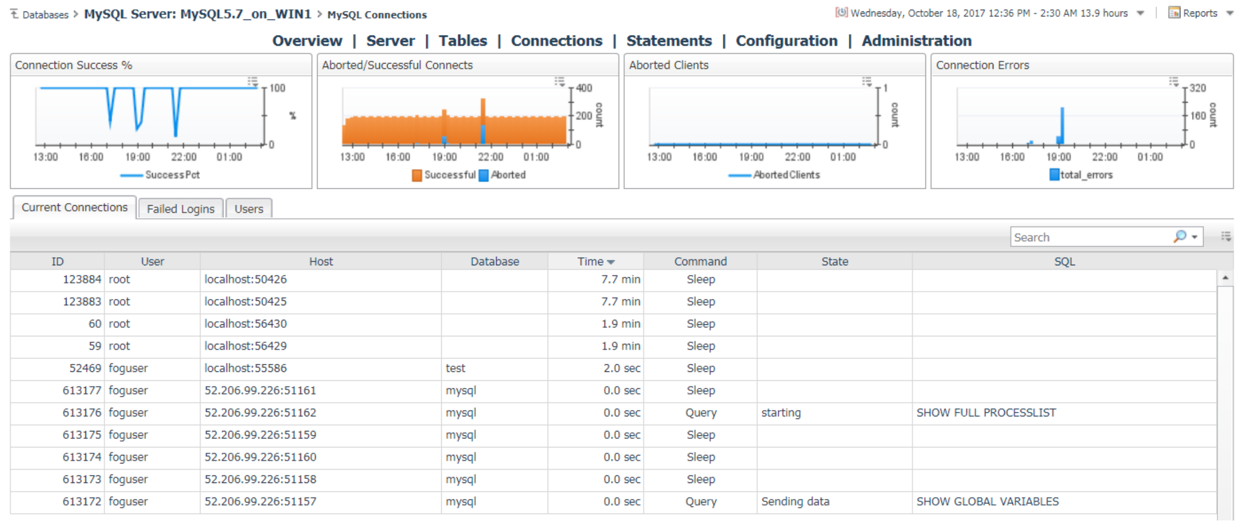






- **Row Count** – Graphs the number of rows in the tablespace for the selected time range
- **Space Used** – Graphs the Table's Used Space vs. Total Size for the selected time range.
- **Growth Rate** – Shows growth/shrinkage of selected table size.
- **Locks Held or Requested** – Shows locks held or requested for selected table/

Connections

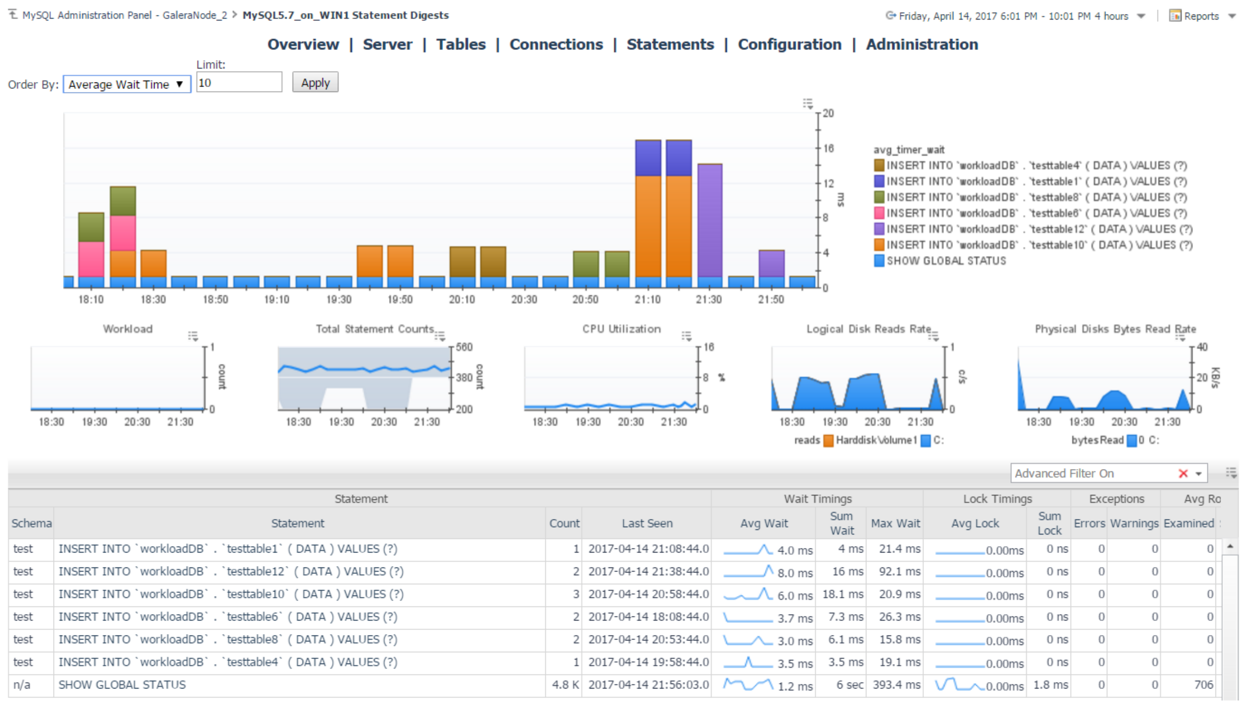

Databases > MySQL Server: MySQL5.7_on_WIN1 > MySQL Connections
Wednesday, October 18, 2017 12:36 PM - 2:30 AM 13.9 hours
Reports
Overview | Server | Tables | Connections | Statements | Configuration | Administration
Connection Success %
100
%
0
13:00 16:00 19:00 22:00 01:00
SuccessPct
Aborted/Successful Connects
400
200
0
count
Successful
Aborted
Aborted Clients
1
0
count
AbortedClients
Connection Errors
320
160
0
count
total_errors
Current Connections
Failed Logins
Users
Search

ID	User	Host	Database	Time	Command	State	SQL
123884	root	localhost:50426		7.7 min	Sleep		
123883	root	localhost:50425		7.7 min	Sleep		
60	root	localhost:56430		1.9 min	Sleep		
59	root	localhost:56429		1.9 min	Sleep		
52469	foguser	localhost:55586	test	2.0 sec	Sleep		
613177	foguser	52.206.99.226:51161	mysql	0.0 sec	Sleep		
613176	foguser	52.206.99.226:51162	mysql	0.0 sec	Query	starting	SHOW FULL PROCESSLIST
613175	foguser	52.206.99.226:51159	mysql	0.0 sec	Sleep		
613174	foguser	52.206.99.226:51160	mysql	0.0 sec	Sleep		
613173	foguser	52.206.99.226:51158	mysql	0.0 sec	Sleep		
613172	foguser	52.206.99.226:51157	mysql	0.0 sec	Query	Sending data	SHOW GLOBAL VARIABLES

The Connections page shows graphs of some key connection metrics and three tabs with more information:

- **Current Connections** - The MySQL process list at the time of the last sample
- **Failed Logins** – This section shows all hosts that have made failed login attempts in the selected time range, along with error counts for all connection error types.
- **Users** - A list of MySQL users and privileges, along with password status, current connections, and total connections in the selected time range.

Statements



MySQL Administration Panel - GaleraNode_2 > MySQL5.7_on_WIN1 Statement Digests
Friday, April 14, 2017 6:01 PM - 10:01 PM 4 hours
Reports
Overview | Server | Tables | Connections | Statements | Configuration | Administration
Order By: Average Wait Time
Limit: 10
Apply
ms
avg_timer_wait
INSERT INTO `workloadDB` . `testtable4` (DATA) VALUES (?)
INSERT INTO `workloadDB` . `testtable1` (DATA) VALUES (?)
INSERT INTO `workloadDB` . `testtable8` (DATA) VALUES (?)
INSERT INTO `workloadDB` . `testtable6` (DATA) VALUES (?)
INSERT INTO `workloadDB` . `testtable12` (DATA) VALUES (?)
INSERT INTO `workloadDB` . `testtable10` (DATA) VALUES (?)
SHOW GLOBAL STATUS
Workload
Total Statement Counts
CPU Utilization
Logical Disk Reads Rate
Physical Disks Bytes Read Rate
count
%
s/s
KB/s
reads Harddisk\Volume1 C:
bytesRead 0 C:
Advanced Filter On

Statement				Wait Timings			Lock Timings		Exceptions		Avg Ro
Schema	Statement	Count	Last Seen	Avg Wait	Sum Wait	Max Wait	Avg Lock	Sum Lock	Errors	Warnings	Examined
test	INSERT INTO `workloadDB` . `testtable1` (DATA) VALUES (?)	1	2017-04-14 21:08:44.0	4.0 ms	4 ms	21.4 ms	0.00ms	0 ns	0	0	0
test	INSERT INTO `workloadDB` . `testtable12` (DATA) VALUES (?)	2	2017-04-14 21:38:44.0	8.0 ms	16 ms	92.1 ms	0.00ms	0 ns	0	0	0
test	INSERT INTO `workloadDB` . `testtable10` (DATA) VALUES (?)	3	2017-04-14 20:58:44.0	6.0 ms	18.1 ms	20.9 ms	0.00ms	0 ns	0	0	0
test	INSERT INTO `workloadDB` . `testtable6` (DATA) VALUES (?)	2	2017-04-14 18:08:44.0	3.7 ms	7.3 ms	26.3 ms	0.00ms	0 ns	0	0	0
test	INSERT INTO `workloadDB` . `testtable8` (DATA) VALUES (?)	2	2017-04-14 20:53:44.0	3.0 ms	6.1 ms	15.8 ms	0.00ms	0 ns	0	0	0
test	INSERT INTO `workloadDB` . `testtable4` (DATA) VALUES (?)	1	2017-04-14 19:58:44.0	3.5 ms	3.5 ms	19.1 ms	0.00ms	0 ns	0	0	0
n/a	SHOW GLOBAL STATUS	4.8 K	2017-04-14 21:56:03.0	1.2 ms	6 sec	393.4 ms	0.00ms	1.8 ms	0	0	706

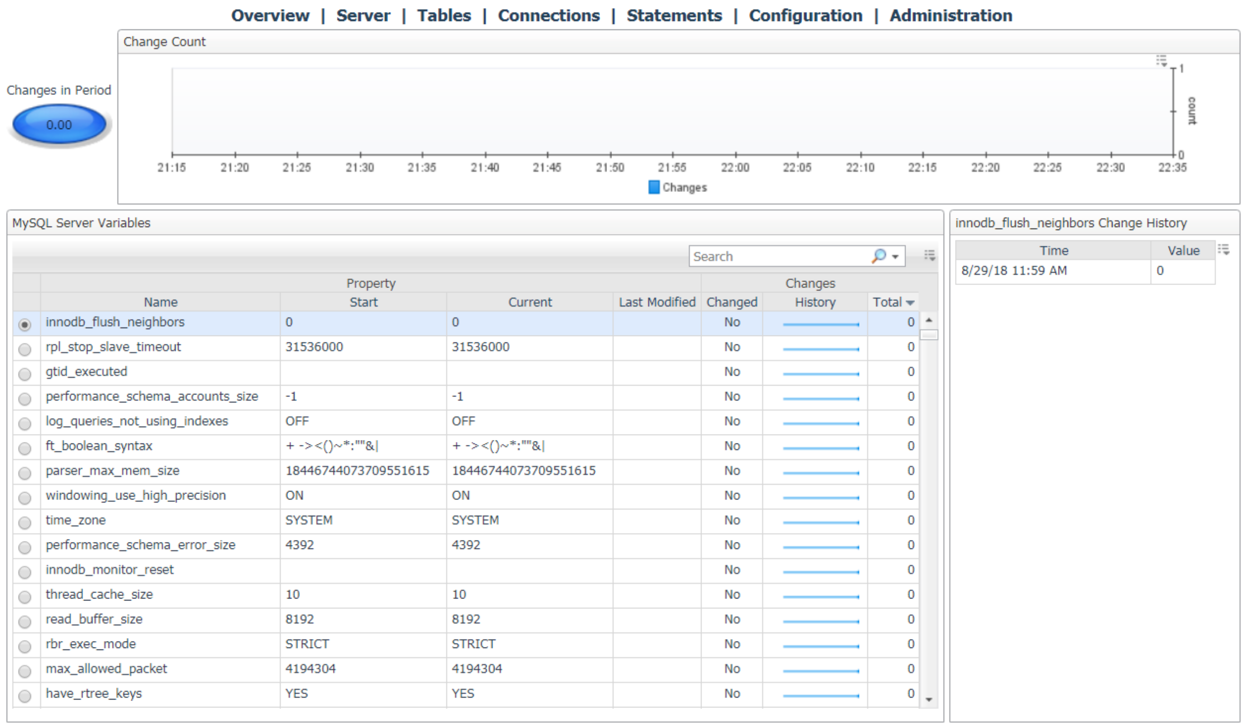
The MySQL Statement Digests dashboard is available for MySQL 5.6.5+ servers with the performance_schema engine enabled. Statement digests are normalized statements which have the same operation plan. Even if values in the statement differ, the server is performing the same operations and can therefore be aggregated for the purposes of analysis. The agent collects these from the server along with raw and calculated statistical data and provides them to the FMS.

The main time plot displays the top *X* statements by one of several metrics that can be selected using the provided dropdown. Below that are several graphs which can be correlated with statement performance to determine impact on the system.

On the bottom half of the page is a table listing all statement digests gathered from the server. The number of statement digests which the server maintains varies by version and is set with performance_schema_digests_size. If the number of digest types exceeds that, excess statements will be put into a single digest, also collected. All statistics shown in the table are period values for the selected time range with the exception of First/Last seen and Min/Max Wait Time. By default, an advanced filter is set for the table to exclude statements performed on the system databases mysql, performance_schema, and information_schema. This can be disabled by clicking the ‘x’ in the filter text box. Selecting a row of the table will update the bottom portion with graphed metrics related to the selected digest.

Page actions available in the right-hand panel include the MySQL Agent Selector and an action to Delete Old Statements. When the MySQL server is restarted or the performance_schema’s statement digests table is truncated, digest statistics will be reset. If a previously collected statement is executed on the server again, it will be merged with the historical data preserved by Foglight. If it is never executed again, it will become a dead object. The Delete Old Statements action is a convenience action to delete these dead topology objects and remove them and their history from the FMS.

Configuration



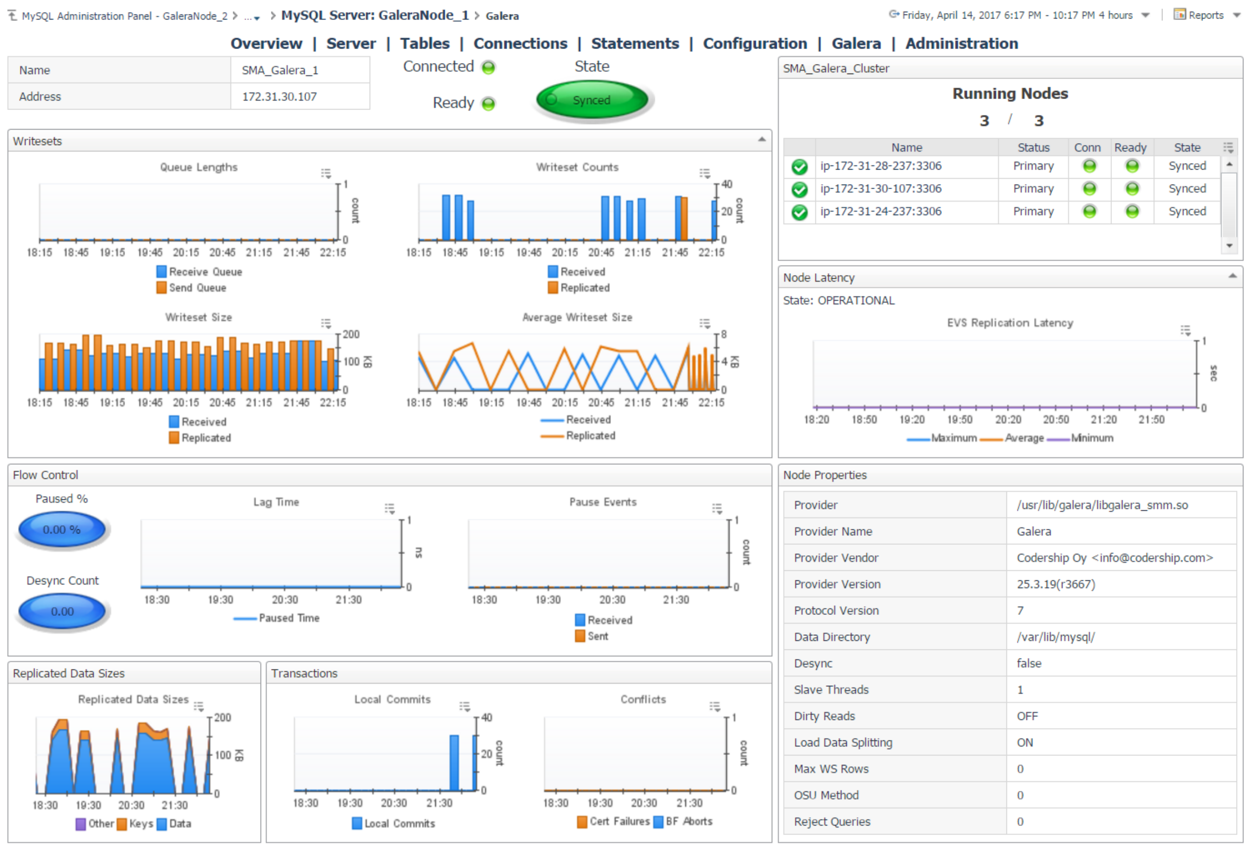
	Property				Changes		
	Name	Start	Current	Last Modified	Changed	History	Total
●	innodb_flush_neighbors	0	0		No		0
○	rpl_stop_slave_timeout	31536000	31536000		No		0
○	gtid_executed				No		0
○	performance_schema_accounts_size	-1	-1		No		0
○	log_queries_not_using_indexes	OFF	OFF		No		0
○	ft_boolean_syntax	+ -><()~*:""&|	+ -><()~*:""&|		No		0
○	parser_max_mem_size	18446744073709551615	18446744073709551615		No		0
○	windowing_use_high_precision	ON	ON		No		0
○	time_zone	SYSTEM	SYSTEM		No		0
○	performance_schema_error_size	4392	4392		No		0
○	innodb_monitor_reset				No		0
○	thread_cache_size	10	10		No		0
○	read_buffer_size	8192	8192		No		0
○	rbr_exec_mode	STRICT	STRICT		No		0
○	max_allowed_packet	4194304	4194304		No		0
○	have_rtree_keys	YES	YES		No		0

Time	Value
8/29/18 11:59 AM	0

The configuration page tracks the MySQL server configuration. The metric indicator and change count graph shows numbers of changes in the selected time range. The table displays every server variable, showing the name, value at beginning and end of the selected time range, the date of the last modification since server monitoring began, and a history of changes for that variable.

By selecting a variable, you can view a history of changes to that variable in the panel on the left for as long as the data is retained.

Galera Node


MySQL Administration Panel - GaleraNode_2 > ... > **MySQL Server: GaleraNode_1** > Galera

Friday, April 14, 2017 6:17 PM - 10:17 PM 4 hours | Reports

Overview | Server | Tables | Connections | Statements | Configuration | Galera | Administration

Name	SMA_Galera_1
Address	172.31.30.107

Connected ● State
Ready ● Synced

SMA_Galera_Cluster

Running Nodes
3 / 3

	Name	Status	Conn	Ready	State
✓	ip-172-31-28-237:3306	Primary	●	●	Synced
✓	ip-172-31-30-107:3306	Primary	●	●	Synced
✓	ip-172-31-24-237:3306	Primary	●	●	Synced

Node Latency
State: OPERATIONAL

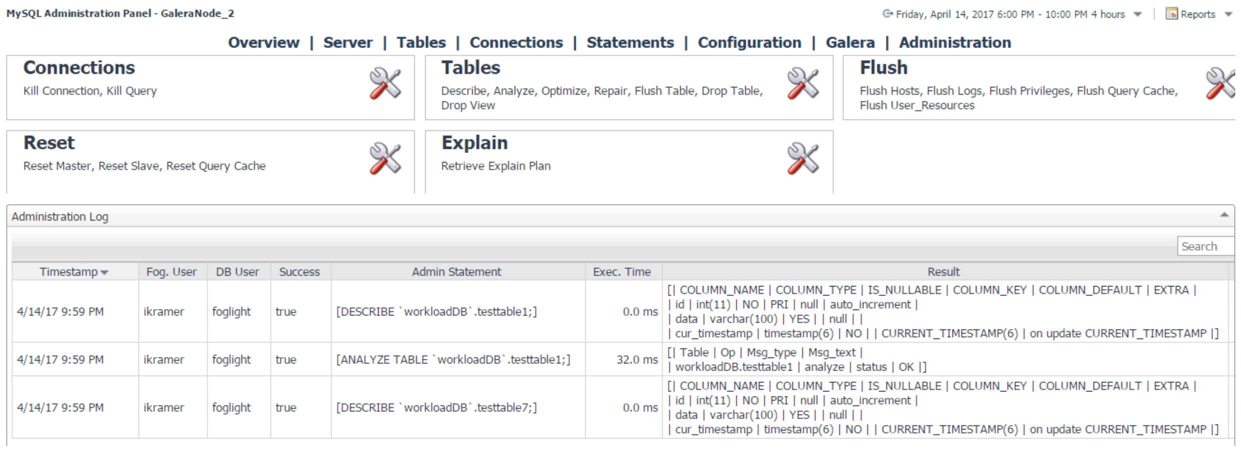
Flow Control
Paused %: 0.00 %
Desync Count: 0.00

Node Properties

Provider	/usr/lib/galera/libgalera_smm.so
Provider Name	Galera
Provider Vendor	Codership Oy <info@codership.com>
Provider Version	25.3.19(r3667)
Protocol Version	7
Data Directory	/var/lib/mysql/
Desync	false
Slave Threads	1
Dirty Reads	OFF
Load Data Splitting	ON
Max WS Rows	0
OSU Method	0
Reject Queries	0

The Galera Node page shows Galera-related information for a MySQL server configured as a Galera node. This page shows the current status of the Galera nodes as well as relevant server variables and informational and performance metrics for writesets, flow control, latency, transactions, and replicated data size. At the top right is a summary for the Galera cluster which the node is a part of, showing the number of collected / monitored nodes for the cluster and listing each node in the cluster and their current states. By clicking the name of a different node, the page will update to feature information on the selected node.

Administration Panel



MySQL Administration Panel - GaleraNode_2

Friday, April 14, 2017 6:00 PM - 10:00 PM 4 hours | Reports

Overview | Server | Tables | Connections | Statements | Configuration | Galera | Administration

Connections
Kill Connection, Kill Query

Tables
Describe, Analyze, Optimize, Repair, Flush Table, Drop Table, Drop View

Flush
Flush Hosts, Flush Logs, Flush Privileges, Flush Query Cache, Flush User_Resources

Reset
Reset Master, Reset Slave, Reset Query Cache

Explain
Retrieve Explain Plan

Administration Log

Search

Timestamp	Fog. User	DB User	Success	Admin Statement	Exec. Time	Result
4/14/17 9:59 PM	ikramer	foglight	true	[DESCRIBE `workloadDB`.testtable1;]	0.0 ms	[| COLUMN_NAME | COLUMN_TYPE | IS_NULLABLE | COLUMN_KEY | COLUMN_DEFAULT | EXTRA | | id | int(11) | NO | PRI | null | auto_increment | | data | varchar(100) | YES | | null | | | cur_timestamp | timestamp(6) | NO | | CURRENT_TIMESTAMP(6) | on update CURRENT_TIMESTAMP |]
4/14/17 9:59 PM	ikramer	foglight	true	[ANALYZE TABLE `workloadDB`.testtable1;]	32.0 ms	[| Table | Op | Msg_type | Msg_text | | workloadDB.testtable1 | analyze | status | OK |]
4/14/17 9:59 PM	ikramer	foglight	true	[DESCRIBE `workloadDB`.testtable7;]	0.0 ms	[| COLUMN_NAME | COLUMN_TYPE | IS_NULLABLE | COLUMN_KEY | COLUMN_DEFAULT | EXTRA | | id | int(11) | NO | PRI | null | auto_increment | | data | varchar(100) | YES | | null | | | cur_timestamp | timestamp(6) | NO | | CURRENT_TIMESTAMP(6) | on update CURRENT_TIMESTAMP |]

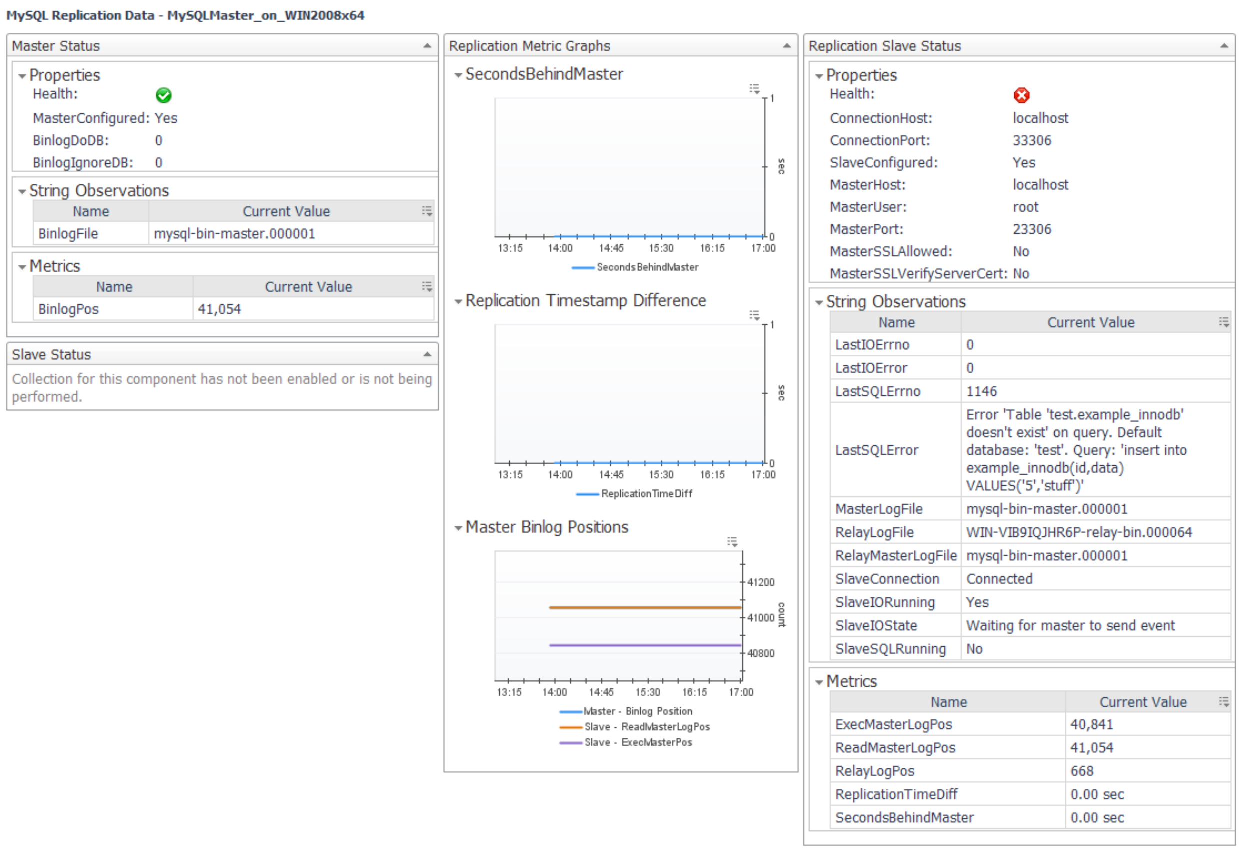
The MySQL Administration Panel features server operations which can be performed through the console by permitted users. The actions currently available represent the initial offering and later versions will incorporate more possibilities. An action is performed by clicking the icon in the relevant category, which will bring up a menu dialog with options and confirmation of the requested action. Actions can only be performed on one server at a time. The active server can be switched using the MySQL Agent Selector in the right-hand pane. Only agents where administration is enabled will be shown.

The Administration Log below records actions performed in the specified time range for auditing purposes. Information includes a timestamp, the Foglight user who performed the action, the database user employed, the actual statement text, execution time, result (if any) and any resulting SQLExceptions or warnings from the statement.

Prerequisites to use Administrative Actions

1) In the Agent Properties of the agent monitoring the MySQL server, Enable Administration must be set to true.
2) Also in the Agent Properties, a DB user and password must be provided. This user must have the necessary privileges to perform actions taken through the Administration Panel or the action will fail. This user will be solely employed to perform actions on request, not for data collection or other purposes.
3) A Foglight user must have the MySQL Administrator role granted to them in the Administration>Users & Security dashboard in order to access and use the Administration Panel.

Replication Data



MySQL Replication Data - MySQLMaster_on_WIN2008x64

Master Status

Properties

Health:	✓
MasterConfigured:	Yes
BinlogDoDB:	0
BinlogIgnoreDB:	0

String Observations

Name	Current Value
BinlogFile	mysql-bin-master.000001

Metrics

Name	Current Value
BinlogPos	41,054

Slave Status

Collection for this component has not been enabled or is not being performed.

Replication Metric Graphs

Replication Slave Status

Properties

Health:	✗
ConnectionHost:	localhost
ConnectionPort:	33306
SlaveConfigured:	Yes
MasterHost:	localhost
MasterUser:	root
MasterPort:	23306
MasterSSLAllowed:	No
MasterSSLVerifyServerCert:	No

String Observations

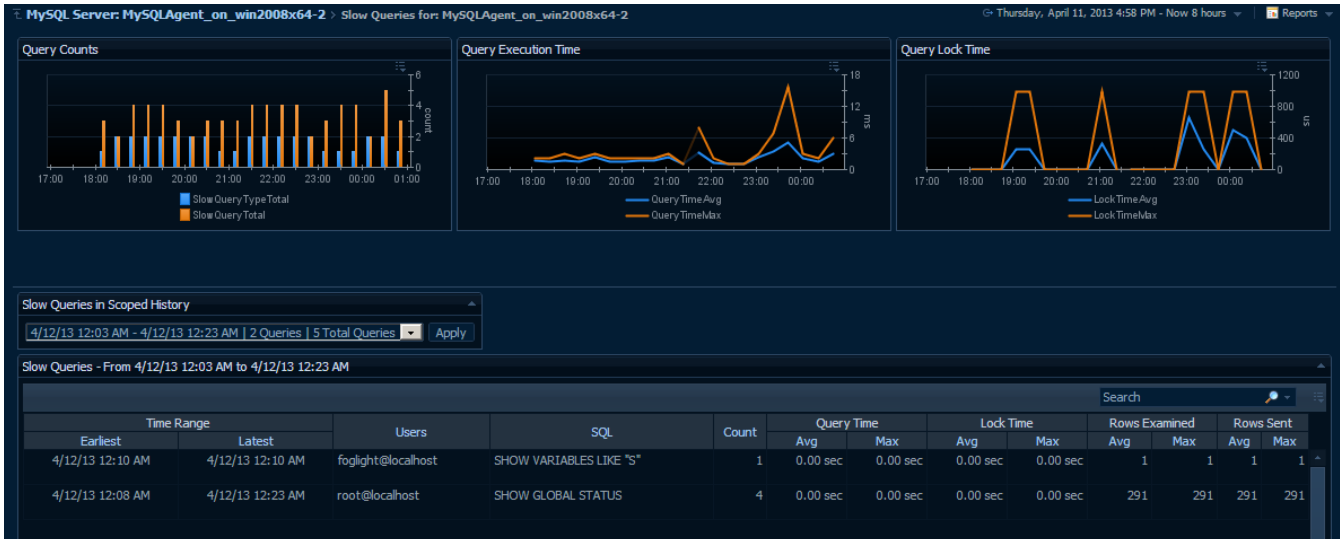
Name	Current Value
LastIOErrno	0
LastIOError	0
LastSQLErrno	1146
LastSQLError	Error 'Table 'test.example_innodb' doesn't exist' on query. Default database: 'test'. Query: 'insert into example_innodb(id,data) VALUES('5','stuff')'
MasterLogFile	mysql-bin-master.000001
RelayLogFile	WIN-VIB9IQJHR6P-relay-bin.000064
RelayMasterLogFile	mysql-bin-master.000001
SlaveConnection	Connected
SlaveIORunning	Yes
SlaveIOState	Waiting for master to send event
SlaveSQLRunning	No

Metrics

Name	Current Value
ExecMasterLogPos	40,841
ReadMasterLogPos	41,054
RelayLogPos	668
ReplicationTimeDiff	0.00 sec
SecondsBehindMaster	0.00 sec

This page shows complete data on MySQL Replication collected by a single MySQL agent and will change depending on which collections have been enabled. This page is primarily meant to serve as a demonstration of data collected by the agent, but can be used for simple Master-Slave replication configurations. More complex configurations can use elements of this page in another dashboard.

Slow Queries (deprecated)


MySQL Server: MySQLAgent_on_win2008x64-2 > Slow Queries for: MySQLAgent_on_win2008x64-2
Thursday, April 11, 2013 4:58 PM - Now 8 hours
Reports
Query Counts
Slow Query Type Total
Slow Query Total
Query Execution Time
Query Time Avg
Query Time Max
Query Lock Time
Lock Time Avg
Lock Time Max
Slow Queries in Scoped History
Apply
Search

4/12/13 12:03 AM - 4/12/13 12:23 AM | 2 Queries | 5 Total Queries

Slow Queries - From 4/12/13 12:03 AM to 4/12/13 12:23 AM

Time Range		Users	SQL	Count	Query Time		Lock Time		Rows Examined		Rows Sent	
Earliest	Latest				Avg	Max	Avg	Max	Avg	Max	Avg	Max
4/12/13 12:10 AM	4/12/13 12:10 AM	foglight@localhost	SHOW VARIABLES LIKE "S"	1	0.00 sec	0.00 sec	0.00 sec	0.00 sec	1	1	1	1
4/12/13 12:08 AM	4/12/13 12:23 AM	root@localhost	SHOW GLOBAL STATUS	4	0.00 sec	0.00 sec	0.00 sec	0.00 sec	291	291	291	291

The Slow Queries dashboard has been deprecated. For MySQL versions 5.6.5+, the Statements dashboard features more data gathered from the Performance Schema database.

The Slow Queries dashboard displays collected queries and their aggregated metrics as well as several derived metrics to show slow query activity overall. At the top of the page, 3 time plots graph these derived metrics. The first graph shows the number of unique queries and the number of total queries found during each collection period. The second graph shows the average and maximum execution time for all collected queries across time. The third graph shows the average and maximum lock times.

Below that, a dropdown list shows all available data snapshots that took place within the scoped time range of the page. By manipulating the Zonar or the calendar feature at the top right, you can change the data snapshots available in the dropdown as well as the scoped time range of the graphs. In the table displaying the unique queries for that snapshot, the following data is shown, in order from left to right:

- **Time Range (Earliest)** – The first appearance of this unique query in the current collection.
- **Time Range (Latest)** – The last appearance of this unique query in the current collection.
- **Users** – A list of the user(s) who have executed this query.
- **SQL** – A generalized version of the SQL string for this unique query.
- **Count** – The number of instances of this query.
- **Query Time (Avg)** – The average execution time for this query.
- **Query Time (Max)** – The maximum execution time for this query.
- **Lock Time (Avg)** – The average lock time for this query.
- **Lock Time (Max)** – The maximum lock time for this query.
- **Rows Examined Time (Avg)** – The average number of rows examined for this query.
- **Rows Examined (Max)** – The maximum number of rows examined for this query.
- **Rows Sent (Avg)** – The average number of rows sent for this query.
- **Rows Sent (Max)** – The maximum number of rows sent for this query.

In addition to these columns, minimum values for these unique query metrics are available, though hidden. To make these columns visible or hide others, use the customizer at the top-right of the table. You can also use the search bar to filter the list of available queries.

Appendix A – Rules

Concurrent_Queries_Running

Purpose:

The Concurrent Queries Running rule will fire when the number of threads currently running exceeds normal usage.

Rule Definition:

Condition	State
#ThreadsRunning#>25	Critical

Problem Description

Too many active queries indicates there is a severe load on the server, and may be a sign of lock contention or unoptimized SQL queries.

Advice

Issue a SHOW FULL PROCESSLIST and examine the SQL output for locked or unoptimized queries. You can also enable the slow query log to log queries that are running too long or are not using an index.

Database_Connectivity Rule

Purpose:

The Database_Connectivity rule will fire when a new connection cannot be made to the database.

Rule Definition:

Condition	State
MySQLAgent_Connection_Status.ConnectivityPct = 0	Fatal
MySQLAgent_Connection_Status.ConnectivityPct =100	Normal

Inefficient_Sort Rule

Purpose:

The Inefficient Sort rule will help determine if the sort buffer is too small. If that is the case then a sort may have to do a number of merge passes using temporary disk files.

Rule Definition:

Condition	State
((MySQLAgent_Sort_Buffer.MergePassesSortedPerSec) / (MySQLAgent_Sort_Buffer.SortRangePerSec + MySQLAgent_Sort_Buffer.SortScanPerSec)) >= registry("InefficientSortCritical")	Critical
((MySQLAgent_Sort_Buffer.MergePassesSortedPerSec) / (MySQLAgent_Sort_Buffer.SortRangePerSec + MySQLAgent_Sort_Buffer.SortScanPerSec)) >= registry("InefficientSortWarning")	Warning

Note
registry("InefficientSortCritical ") is a Foglight Registry Variable with a default value of 2.
registry("InefficientSortWarning") is a Foglight Registry Variable with a default value of 1.
Use the Foglight Registry to change thresholds for rules by modifying registry entries.

InnoDB_Buffer_Pool_HitRate Rule

Purpose:

The InnoDB_Buffer_Pool_HitRate rule will fire when the buffer pool hit rate falls below a user defined percentage. This rule applies only to InnoDB.

Rule Definition:

Condition	State
MySQLAgent_Innodb_Buffer_Pool.HitRatePct <= registry("InnoDBBufferPoolHitRateCritical")	Critical
MySQLAgent_Innodb_Buffer_Pool.HitRatePct <= registry("InnoDBBufferPoolHitRateWarning")	Warning

Note
registry("InnoDBBufferPoolHitRateCritical") is a Foglight Registry Variable with a default value of 90.
registry("InnoDBBufferPoolHitRateWarning") is a Foglight Registry Variable with a default value of 95.
Use the Foglight Registry to change thresholds for rules by modifying registry entries.

Problem Description

Logical I/O is many times faster than physical I/O, and therefore a DBA should strive to keep physical I/O to a minimum. It is true that logical I/O is not free, and that the DBA should work to keep all I/O to a minimum, but it is best if most data access is performed in memory. When using

InnoDB, most data access should occur in RAM, and therefore the InnoDB buffer cache hit rate should be high.

Advice

Increase the size of the InnoDB buffer pool: Set innodb_buffer_pool_size to a larger value in your my.cnf/my.ini file and restart your server. Continue to monitor the cache to be sure there is an improvement.

Key_Buffer_HitRate Rule

Purpose:

The Key_Buffer_HitRate rule will fire when the Key Buffer hit rate falls below a user defined percentage.

Rule Definition:

Condition	State
MySQLAgent_Key_Buffer_KeyHitRate <= registry("KeyBufferHitRateCritical")	Critical
MySQLAgent_Key_Buffer_KeyHitRate <= registry("KeyBufferHitRateWarning")	Warning

Note
registry("KeyBufferHitRateCritical") is a Foglight Registry Variable with a default value of 90.
registry("KeyBufferHitRateWarning") is a Foglight Registry Variable with a default value of 95.
Use the Foglight Registry to change thresholds for rules by modifying registry entries.

Problem Description

The key cache hit ratio represents the proportion of keys that are being read from the key cache in memory instead of from disk. This should normally be greater than 99% for optimum efficiency.

Advice

Increase the key_buffer_size variable and monitor the key cache hit ratio. When it reaches an acceptable level, put the corresponding value of key_buffer_size in your my.cnf/my.ini file so the variable is set properly when the server is restarted.

Query_Cache_Hit_Rate Rule

Purpose:

The Query_Cache_HitRate rule will fire when the query cache hit rate falls below a user defined percentage. A low hit rate is not necessarily indicative of a problem; it may just be due to the types of SQL statements being run.

Rule Definition:

Condition	State
MySQLAgent_Query_Cache.HitRatePct <= registry("QueryCacheHitRateCritical")	Critical
MySQLAgent_Query_Cache.HitRatePct <= registry("QueryCacheHitRateWarning")	Warning

Note
registry("QueryCacheHitRateCritical") is a Foglight Registry Variable with a default value of 92.
registry("QueryCacheHitRateWarning") is a Foglight Registry Variable with a default value of 96.
Use the Foglight Registry to change thresholds for rules by modifying registry entries.

Problem Description

When enabled, the query cache should experience a high degree of "hits", meaning that queries in the cache are being reused by other user connections. A low hit rate may mean that not enough memory is allocated to the cache, identical queries are not being issued repeatedly to the server, or that the statements in the query cache are invalidated too frequently by INSERT, UPDATE or DELETE statements.

Advice

Evaluate whether the query cache is suitable for your application. If you have a high rate of INSERT / UPDATE / DELETE statements compared to SELECT statements, then there may be little benefit to enabling the query cache. Also check whether there is a high value of Qcache_lowmem_prunes, and if so consider increasing the query_cache_size.

Query_Cache_Undersized

Purpose:

This rule is designed to help the DBA determine the optimal size for the Query Cache.

Rule Definition:

Condition	State
#QueryCacheSize# > 0 && #NumLowMemPrunes# > 0	Warning

Problem Description

When the Query Cache is full, and needs to add more queries to the cache, it will make more room in the cache by freeing the least recently used queries from the cache, and then inserting the new queries. If this is happening often then you should increase the size of the cache to avoid this constant "swapping".

Increase the size of the query cache dynamically, then monitor the cache hit rate for improvements. Once you reach an acceptable hit ratio, and NumLowMemPrunes stops increasing, set that value within your my.cnf/my.ini file so the variable is set properly when the server is restarted.

Replication Slave Connection Unavailable

Purpose:

The agent was unable to connect to the replication slave during the last sample collection.

Rule Definition:

Condition	State
#SlaveConnection#!="Connected"	Fatal

Message:

@AgentName failed to connect to the replication slave server during the last sample collection.

Replication Server Times Out of Sync

Purpose:

Checks if the time difference between servers is greater than thresholds set in registry variables. Not enabled by default.

Rule Definition:

Condition	State
#ReplicationTimeDiff#>registry("replicationTimeFATAL")	Fatal
#ReplicationTimeDiff#>registry("replicationTimeCRIT")	Critical
#ReplicationTimeDiff#>registry("replicationTimeWARN")	Warning

Note

registry("replicationTimeFATAL ") is a Foglight Registry Variable with a default value of 60.
registry("replicationTimeCRIT ") is a Foglight Registry Variable with a default value of 30.
registry("replicationTimeWARN ") is a Foglight Registry Variable with a default value of 5.
Use the Foglight Registry to change thresholds for rules by modifying registry entries.

Message:

The replication servers monitored by @AgentName are out of sync. The difference between the UNIX Timestamp of these servers is ___ second(s), which exceeds ___, the current registry value.

Replication Slave Behind Master

Purpose:

Checks if the SecondsBehindMaster value is greater than thresholds set in registry variables. This rule is not enabled by default.

Rule Definition:

Condition	State
#SecondsBehindMaster#>registry("slaveBehindFATAL")	Fatal
#SecondsBehindMaster#>registry("slaveBehindCRIT")	Critical
#SecondsBehindMaster#>registry("slaveBehindWARN")	Warning

Note

registry("slaveBehindFATAL ") is a Foglight Registry Variable with a default value of 30.
registry("slaveBehindCRIT ") is a Foglight Registry Variable with a default value of 15.
registry("slaveBehindWARN ") is a Foglight Registry Variable with a default value of 5.
Use the Foglight Registry to change thresholds for rules by modifying registry entries.

Message:

The Seconds_Behind_Master value for the replication server monitored by @AgentName is ___ second(s), which is higher than ___, the current registry value.

Seconds_Behind_Master effectively calculates how far behind the Replication Slave's SQL thread is from its I/O thread. On a fast network where I/O delay is close to 0, this will be approximately the delay currently experienced between the Master and Slave SQL executions.

Replication Slave I/O In Failed State

Purpose:

Triggers if the message in SlaveIOState contains the string "failed" during two consecutive samples, meaning that the I/O thread has been unable to connect or maintain a connection to the master server.

Rule Definition:

Condition	State

def matcher = (#SlaveIOState# =~ ".*failed.*")
matcher.matches()
	Critical

Message:

The SlaveIOState for the replication server monitored by @AgentName is either reconnecting or waiting to reconnect after a disconnection event. This has occurred during the past two samples. The current state is: ___.

Replication Slave SQL Thread Not Running

Purpose:

Checks whether SQL thread is running.

Rule Definition:

Condition	State
getPropertyObject("SlaveConfigured")=="Yes" && #SlaveSQLRunning#=="No"	Fatal

Message:

The Slave SQL Thread for the Replication Slave server monitored by @AgentName is not running.

Replication Slave I/O Thread Not Running

Purpose:

Checks whether I/O thread is running and connected to master.

Rule Definition:

Condition	State

getPropertyObject("SlaveConfigured")=="Yes" && (#SlaveIORunning#=="No" || #SlaveIORunning#=="Connecting")	Fatal

Message:

The Slave IO Thread for the Replication Slave server monitored by @AgentName is either not running or not connected to a replication master. The current value for Slave_IO_Running is: ___. Not all MySQL versions are capable of making the distinction between these two states. Please refer to MySQL documentation for more information.

Slave Behind Master

Purpose:

Checks if the SecondsBehindMaster value is greater than thresholds set in registry variables. This rule is not enabled by default.

Rule Definition:

Condition	**State**
#SecondsBehindMaster#>registry("slaveBehindFATAL")	Fatal
#SecondsBehindMaster#>registry("slaveBehindCRIT")	Critical
#SecondsBehindMaster#>registry("slaveBehindWARN")	Warning

Note

registry("slaveBehindFATAL ") is a Foglight Registry Variable with a default value of 30.
registry("slaveBehindCRIT ") is a Foglight Registry Variable with a default value of 15.
registry("slaveBehindWARN ") is a Foglight Registry Variable with a default value of 5.
Use the Foglight Registry to change thresholds for rules by modifying registry entries.

Message:

The Seconds_Behind_Master value for the server monitored by @AgentName is ___ second(s), which is higher than ___, the current registry value.

Seconds_Behind_Master effectively calculates how far behind the Slave's SQL thread is from its I/O thread. On a fast network where I/O delay is close to 0, this will be approximately the delay currently experienced between the Master and Slave SQL executions.

Slave I/O In Failed State

Purpose:

Triggers if the message in SlaveIOState contains the string "failed" during two consecutive samples, meaning that the I/O thread has been unable to connect or maintain a connection to the master server.

Rule Definition:

Condition	State
def matcher = (#SlaveIOState# =~ ".*failed.*") matcher.matches()	Critical

Message:

The SlaveIOState for the server monitored by @AgentName is either reconnecting or waiting to reconnect after a disconnection event. This has occurred during the past two samples. The current state is: ___.

Slave SQL Thread Not Running

Purpose:

Checks whether SQL thread is running.

Rule Definition:

Condition	State
getPropertyObject("SlaveConfigured")=="Yes" && #SlaveSQLRunning#=="No"	Fatal

Message:

The Slave SQL Thread for the Slave server monitored by @AgentName is not running.

Slave I/O Thread Not Running

Purpose:

Checks whether I/O thread is running and connected to master.

Rule Definition:

Condition	State
getPropertyObject("SlaveConfigured")=="Yes" && (#SlaveIORunning#=="No" || #SlaveIORunning#=="Connecting")	Fatal

Message:

The Slave IO Thread for the Slave server monitored by @AgentName is either not running or not connected to a replication master. The current value for Slave_IO_Running is: ___. Not all MySQL versions are capable of making the distinction between these two states. Please refer to MySQL documentation for more information.

Slow_Connections Rule

Purpose:

The Slow_Connections rule will fire when the time to establish a connection falls below a user defined time. Under some circumstances MySQL does a DNS lookup of the host name that the user is connecting from and this can delay establishing the connection.

Rule Definition:

Condition	State
MySQLAgent_Connection_Status.SecondsConnect >= registry("SlowConnectionFatal")	Fatal
MySQLAgent_Connection_Status.SecondsConnect >= registry("SlowConnectionCritical")	Critical
MySQLAgent_Connection_Status.SecondsConnect >= registry("SlowConnectionWarning")	Warning

Note
registry("SlowConnectionFatal") is a Foglight Registry Variable with a default value of 10.
registry("SlowConnectionCritical") is a Foglight Registry Variable with a default value of 2.
registry("SlowConnectionWarning") is a Foglight Registry Variable with a default value of 1.
Use the Foglight Registry to change thresholds for rules by modifying registry entries.

Slow Query Average Execution Time Rule

Purpose:

The Slow Query Average Execution Time rule fires when the average execution time for all queries written to the slow query log in the latest collection exceeds the defined threshold.

Rule Definition:

Condition	State
#SlowQueryAvgTime#>registry("SlowQueryAvgTimeFatal")	Fatal

#SlowQueryAvgTime#>registry("SlowQueryAvgTimeCrit")	Critical
#SlowQueryAvgTime#>registry("SlowQueryAvgTimeWarn")	Warning

Note
registry("SlowQueryAvgTimeFatal ") is a Foglight Registry Variable with a default value of 2.5.
registry("SlowQueryAvgTimeCrit ") is a Foglight Registry Variable with a default value of 5.
registry("SlowQueryAvgTimeWarn ") is a Foglight Registry Variable with a default value of 7.5.
Use the Foglight Registry to change thresholds for rules by modifying registry entries.

Slow Query Max Execution Time Rule

Purpose:

The Slow Query Max Execution Time rule fires when the maximum execution time for all queries written to the slow query log in the latest collection exceeds the defined threshold.

Rule Definition:

Condition	State
#SlowQueryMaxTime#>registry("SlowQueryMaxTimeFatal")	Fatal
#SlowQueryMaxTime#>registry("SlowQueryMaxTimeCrit")	Critical
#SlowQueryMaxTime#>registry("SlowQueryMaxTimeWarn")	Warning

Note
registry("SlowQueryMaxTimeFatal ") is a Foglight Registry Variable with a default value of 5.
registry("SlowQueryMaxTimeCrit ") is a Foglight Registry Variable with a default value of 7.5.
registry("SlowQueryMaxTimeWarn ") is a Foglight Registry Variable with a default value of 10.
Use the Foglight Registry to change thresholds for rules by modifying registry entries.

Table Scans Excessive

Purpose:

This rule indicates that the server may not be using indexes efficiently to read from the database tables.

Rule Definition:

Condition	State
(#numReqReadRndNext# > 100000) && ((100-(((#numReqReadRndNext# + #numReqReadFixedPos#) / (#numReqReadRndNext# + #numReqReadFixedPos# + #numEntryReadFirst# + #numReqReadNext# + #numReqReadKey#	Fatal

+ #numReqReadPrev#))*100)) < registry("TableScansExcessiveFatal"))	
(#numReqReadRndNext# > 100000) && ((100-(((#numReqReadRndNext# + #numReqReadFixedPos#) / (#numReqReadRndNext# + #numReqReadFixedPos# + #numEntryReadFirst# + #numReqReadNext# + #numReqReadKey# + #numReqReadPrev#))*100)) < registry("TableScansExcessiveCritical"))	Critical
(#numReqReadRndNext# > 100000) && ((100-(((#numReqReadRndNext# + #numReqReadFixedPos#) / (#numReqReadRndNext# + #numReqReadFixedPos# + #numEntryReadFirst# + #numReqReadNext# + #numReqReadKey# + #numReqReadPrev#))*100)) < registry("TableScansExcessiveWarning"))	Warning

Note

registry("TableScansExcessiveFatal ") is a Foglight Registry Variable with a default value of 60.
registry("TableScansExcessiveCritical ") is a Foglight Registry Variable with a default value of 80.
registry("TableScansExcessiveWarning ") is a Foglight Registry Variable with a default value of 90.
Use the Foglight Registry to change thresholds for rules by modifying registry entries.

Problem Description

The target server does not appear to be using indexes efficiently. The values of Handler_read_rnd_next and Handler_read_rnd together - which reflect the number of rows read via full table scans - are high compared to the sum of Handler variables which denote all row accesses - such as Handler_read_key, Handler_read_next etc. You should examine your tables and queries for proper use of indexes.

Advice

Look for long-running queries issued shortly before the time of this alert, queries where avg or max execution time has increased significantly from one interval to another, or queries where max execution time is much greater than avg execution time. Also, look for queries flagged as requiring a table scan.

Another option is to turn on the Slow Query Log (if it is not already turned on) and monitor what goes into it. Statements that are logged there are candidates for tuning. Note, however, that statements will only be logged there if they take longer than your long_query_time parameter to run, so statements triggering full scans of small tables that execute very quickly may not show up. Once you have found tuning candidates, use the EXPLAIN statement on the queries to see which tables should have indexes added to them.

If you are using MySQL 4.1 or later you can use the --log-queries-not-using-indexes option to log all statements that do a full table scan, even if they would not otherwise qualify for the slow query log.

Note that full table scans are not necessarily bad, as long as they are confined to very small tables, so be sure to take table size into account as you review your queries and their EXPLAIN plans.

Tablespace_Utilization Rule

Purpose:

The Tablespace_Utilization rule will fire when there is less than a user defined amount of tablespace left in an InnoDB database for those databases that do not have auto extend enabled. This rule applies only to InnoDB.

Rule Definition:

Condition	State
#TablespaceFreeMB#<=registry("TablespaceFatal") && #TableSpaceAutoExtend#<1	Fatal
#TablespaceFreeMB#<=registry("TablespaceCritical") && #TableSpaceAutoExtend#<1	Critical
#TablespaceFreeMB#<=registry("TablespaceWarning") && #TableSpaceAutoExtend#<1	Warning

Note
registry("TablespaceFatal") is a Foglight Registry Variable with a default value of 0.
registry("TablespaceCritical") is a Foglight Registry Variable with a default value of 4.
registry("TablespaceWarning") is a Foglight Registry Variable with a default value of 8.
Use the Foglight Registry to change thresholds for rules by modifying registry entries.

Thread Cache Not Enabled Rule

Purpose:

This rule will fire if the Thread Cache is not enabled.

Rule Definition:

Condition	State
#ThreadCacheSize#<1	Fire

Problem Description

Each connection to the MySQL database server runs in its own thread. Thread creation takes time, so rather than killing the thread when a connection is closed, the server can keep the thread in its thread cache and use it for a new connection later.

Advice

Set your thread_cache_size variable high enough that the Threads_created value in SHOW STATUS stops increasing dynamically, then set this value for thread_cache_size within your my.cnf/my.ini file.

Thread_Pool_HitRate Rule

Purpose:

The Thread_Pool_HitRate rule will fire when the Thread Pool hit rate falls below a user defined percentage.

Rule Definition:

Condition	**State**
MySQLAgent_Thread_Pool_KeyHitRate <= registry("ThreadPoolHitRateCritical")	Critical
MySQLAgent_Thread_Pool_KeyHitRate <= registry("ThreadPoolHitRateWarning")	Warning

---

Note

registry("ThreadPoolHitRateCritical") is a Foglight Registry Variable with a default value of 90.
registry("ThreadPoolHitRateWarning") is a Foglight Registry Variable with a default value of 95.
Use the Foglight Registry to change thresholds for rules by modifying registry entries.

Problem Description

Each connection to the MySQL database server runs in its own thread. Thread creation takes time, so rather than killing the thread when a connection is closed, the server can keep the thread in its thread cache and use it for a new connection later.

Advice

Increase the thread_cache_size variable dynamically and monitor the thread cache hit ratio. When it reaches an acceptable level, put the corresponding value of thread_cache_size in your my.cnf/my.ini file so the variable is set properly when the server is restarted. The ideal situation is to get Threads_created as close as possible to thread_cache_size - no new connections having to wait for new thread allocation - staying as close to around a 99% thread cache hit ratio as you can.

Transaction_Purge_lag Rule

Purpose:

The Transaction_Purge_Lag rule will fire when the number of transactions waiting to be purged exceeds a user defined limit. The user can control the lag by adjusting the MySQL innodb_max_purge_lag startup parameter.

InnoDB keeps information about old versions of rows in the tablespace in an area called the rollback segment so that it can support transaction rollback and consistent reads. When a transaction completes the old versions of the rows affected by the transaction can be purged. This purging is done as a background task and can be delayed by active transactions. If the purging lags too far behind then the rollback segments grow and can fill up the tablespace.

Rule Definition:

Condition	State
MySQLAgent_Innodb_Storage_Engine.TransactionPurgeLag >= registry("TransactionPurgeLagCritical")	Critical
MySQLAgent_Innodb_Storage_Engine.TransactionPurgeLag >= registry("TransactionPurgeLagWarning")	Warning

Note

registry("TransactionPurgeLagCritical") is a Foglight Registry Variable with a default value of 1500000.
registry("TransactionPurgeLagWarning") is a Foglight Registry Variable with a default value of 1000000. Use the Foglight Registry to change thresholds for rules by modifying registry entries.

Problem Description

The TransactionPurgeLag is the difference between the current transaction number and the last transaction number to have its row history purged. This variable indicates how far behind purging is lagging. The user can alter innodb_max_purge_lag to control the lag.

Advice

Check your innodb_max_purge_lag to ensure that it is set to a reasonable number for your needs and that the TransactionPurgeLag does not exceed the innodb_max_purge_lag setting. If it does, it may be caused by an old consistent read view that can see rows marked for purging.

Unflushed_Log_Buffer Rule

Purpose:

The Unflushed_Log_Buffer rule will fire when the percentage of log entries in the log buffer waiting to be flushed to disk exceeds a user defined percentage.

Rule Definition:

Condition	State
MySQLAgent_Innodb_Transaction_Log.UnflushedPct >= registry("UnflushedLogCritical")	Critical
MySQLAgent_Innodb_Transaction_Log.UnflushedPct >= registry("UnflushedLogWarning")	Warning

Note
registry("UnflushedLogCritical") is a Foglight Registry Variable with a default value of 50.
registry("UnflushedLogWarning") is a Foglight Registry Variable with a default value of 30.
Use the Foglight Registry to change thresholds for rules by modifying registry entries.

Problem Description

The percentage of entries in the log buffer has exceeded your user-defined threshold.

Advice

If your environment has large or frequent transactions, you may want to increase innodb_log_buffer_size to deal with this. A large log buffer allows large transactions to run without a need to write the log to disk before the transactions commit. Thus, if you have big transactions, making the log buffer larger saves disk I/O.

Appendix B – Reports

Executive Summary – Executive summary of MySQL instance with high-level information including connection, workload, availability, and performance metrics, as well as top 10 alarms by severity.

Failed Logins Report - Shows failed login attempts to MySQL server in selected timeRange and generated error counts.

Health Check – Report showing health indicators for the MySQL instance, including connections, workload, system, and performance metrics, top 10 statement digests by wait and lock times and error counts, and top 10 alarms by severity.

MySQL Current Connections - Displays connections to a MySQL server from latest sample in selected time range.

MySQL Server Configuration – Displays variables for the MySQL instance with values at the start and end of the selected time range and when they were modified.

MySQL Server Configuration Comparison – Compares current variable values for all MySQL instances contained in a Service against a selected MySQL instance (template server) to ensure configuration compliance. The report will optionally return all variables or just the variables with at least one difference in values.

Storage Report – Shows information on MySQL server storage capacity, growth rate, etc.

Top MySQL Servers by Connections – Displays top *X* MySQL instances by the selected connection metric: successful connects, aborted connects, success percentage, or aborted clients.

Top MySQL Tables – Displays top *X* tables in a MySQL instance by the selected table metric: total size, row count, used space percentage, or growth rate.

Top Statement Digests – Displays top *X* statement digests for a MySQL instance by the selected metric: count, average/sum/max wait time, average lock time, errors, or warnings.

Users Report – Shows information on MySQL users, connections, and privileges.

Appendix C – Table and Field Description

MySQL Agent Data Tables

This section includes data tables for the MySQL agent:

Connection_Status Table

Purpose
This table contains information about whether new connections can be made to the database or not.

Table Description

Field	Description
ConnectionDetails	The connection details; host, port, database and user name.
ConnectionPort	The connection port of the database
ErrorMessage	The error message if a connection could not be made or blank if the connection was successful
ConnectivityPct	Shows if a new connection can be made (value is 100) or not (0)
secondsConnect	Number of seconds MySQL took to fully establish the connection
Connections	The number of connection attempts (successful or not) to the MySQL server.
MaxUsedConnections	The maximum number of connections that have been in use simultaneously since the server started.

Database_Information Table

Purpose
This table contains information about the MySQL database being monitored.

Table Description

Field	Description
MySQL_Version	Will contain the numeric portion of the version. E.g. 5.0.22-community-max-nt-log becomes 5.0.22
MajorVersion	The digits to the left of the first point in the database version number. E.g. if the version number is 5.0.22-community-max-nt-log the this value will be 5

MinorVersion	The digits to the right of the first point in the database version number. E.g. if the version number is 5.0.22-community-max-nt-log the this value will be 0
uptime	The time in seconds that this database has been up

Handler Table

Purpose
This table contains handler information within MySQL.

Table Description

Field	Description
internalCommit	The number of internal COMMIT statements.
numDeleteRows	The number of times that rows have been deleted from tables.
commitCountPrepare	A counter for the prepare phase of two-phase commit operations.
numEntryReadFirst	The number of times the first entry was read from an index.
numReqReadKey	The number of requests to read a row based on a key.
numReqReadNext	The number of requests to read the next row in key order.
numReqReadPrev	The number of requests to read the previous row in key order.
numReqReadFixedPos	The number of requests to read a row based on a fixed position.
numReqReadRndNext	The number of requests to read the next row in the data file.
numReqRollback	The number of requests for a storage engine to perform a rollback operation.
numReqRollback	The number of requests for a storage engine to place a savepoint.
numReqRollback	The number of requests for a storage engine to roll back to a savepoint.
numReqUpdate	The number of requests to update a row in a table.
numReqWrite	The number of requests to insert a row in a table.

Innodb_Buffer_Pool Table

Purpose
This table contains information about the InnoDB buffer. The buffer pool size is tunable and should be made as large as possible to increase the buffer hit rate.

Table Description

Field	Description
BufferMemAllocated	Memory allocated for the InnoDB buffer pool.
BufferPoolAdditionalSize	Memory allocated for additional InnoDB buffer pool
BufferPoolSize	Total number of pages in InnoDB buffer
DatabasePages	Number of database pages in buffer
CleanPages	Number of clean pages in buffer
DirtyPages	Number of dirty pages in buffer
MiscPages	Number of miscellaneous pages in buffer
BufferFree	Number of free pages in buffer
ModifiedPages	Number of modified pages in buffer
FreePagesPct	Percent of free pages in buffer. If this is consistently high then the buffer is too big.
HitRatePct	Percent buffer hit rate
MissRate	Percent buffer miss rate
BufferPoolWriteReq	The number of logical write requests done to the InnoDB buffer pool.
BufferPoolReadReq	The number of logical read requests done to the InnoDB buffer pool.
LogBufferSize	Size of the InnoDB Log Buffer

Innodb_Storage_Engine Table

Purpose
This table contains information about the state of the InnoDB storage engine.

Table Description

Field	Description
MutexOsWaitsPerSec	The number of times InnoDB waited on the OS for a mutex.
MutexSpinWaitsPerSec	The number of times InnoDB spun waiting for a mutex to become free.
MutexSpinRoundsPerSec	The number of times InnoDB spun round waiting for a mutex to become free.
RWlockOsWaitsPerSec	The number of times InnoDB waited on the OS for a read/write lock.
RWlockSpinWaitsPerSec	The number of times InnoDB spun waiting for a shared read/write to become free.

TransactionsPerSec	The number of transactions being processed per second
QueriesInsideInnodb	The number of queries active inside InnoDB
QueriesInQueue	The number of queries waiting to enter InnoDB
TransactionPurgeLag	The difference between the current transaction number and the last transaction number to have its row history purged. Indicates how far behind purging is lagging. The user can alter innodb_max_purge_lag to control the lag.
PendingNormalAsyncIOReadsPerSec	Number of pending normal asynchronous I/O reads per second
PendingNormalAsyncIOWritesPerSec	Number of pending normal asynchronous I/O writes per second
InsertBufferAsyncIOReadsPerSec	Number of insert buffer asynchronous I/O reads per second
InsertBufferAsyncIOWritesPerSec	Number of insert buffer asynchronous I/O writes per second
InnoDBIOThreads	The number of helper threads performing InnoDB I/O.
OSFileReadsPerSec	Number of Operating System reads per second
OSFileWritesPerSec	Number of Operating System writes per second
OSFsyncsPerSec	Number of Operating System fsync calls per second
DataWritesPerSec	Number of physical data writes per second
DataReadsPerSec	Number of physical data reads per second
RowsInsertedPerSec	Number of rows inserted per second
RowsReadPerSec	Number of rows read per second
RowsUpdatesPerSec	Number of rows updated per second
RowsDeletedPerSec	Number of rows deleted per second

Innodb_Transaction_Log Table

Purpose

This table contains information about the state of the InnoDB transaction log. The transaction log buffer size is tunable and affects the speed of large transactions.

Table Description

Field	Description
WaitsPerSec	Number of times InnoDB has to wait for log to be flushed to disk per second
WritesPerSec	Number of times InnoDB has written the log to disk per second

UnflushedPct	Percent of InnoDB log buffer that has yet to be flushed to disk. If this is too high then buffer needs to be tuned.

Joins Table

Purpose
This table contains information on joins within MySQL.

Table Description

Field	Description
fullJoin	The number of joins that perform table scans because they do not use indexes.
fullRangeJoin	The number of joins that used a range search on a reference table.
rangeJoins	The number of joins that used ranges on the first table.
rangeCheckKeys	The number of joins without keys that check for key usage after each row.
fullScanJoins	The number of joins that did a full scan of the first table.

Key_Buffer Table

Purpose
This table contains information on key blocks within MySQL.

Table Description

Field	Description
KeyBlocksNotFlushed	The number of key blocks in the key cache that have changed but have not yet been flushed to disk.
KeyBlocksUnUsed	The number of unused blocks in the key cache.
KeyBlocksUsed	The number of used blocks in the key cache.
KeyReadRequests	The number of requests to read a key block from the cache.
KeyReads	The number of physical reads of a key block from disk.
KeyWriteRequests	The number of requests to write a key block to the cache.
KeyWrites	The number of physical writes of a key block to disk.
KeyHitRate	The rate of physical reads from a disk to requests to read from the cache.

Master_Status Table

Purpose

This table stores information about the slave status of the host server.

Table Description

Field	Description
MasterConfigured	The server is set up for master replication
BinlogFile	The name of the current master binary log file
BinlogPos	The position in the current master binary log file
BinlogDoDB	The databases included in replication, if this option is set
BinlogIgnoreDB	The databases not included in replication, if this option is set

Mutex Table

Purpose

This table contains information to the internal Mutexes within MySQL.

Table Description

Field	Description
MutexName	The mutex name
MutexCount	The number of accesses to this mutex
MutexRate	The number of accesses to the mutex per second
SpinWaits	The number of spin waits made while waiting for the mutex to become free
SpinWaitRate	The number of spin waits per second while waiting for the mutex to become free
SpinRounds	The number of spin rounds made while waiting for the mutex to become free
OSWaits	The number of OS waits made while waiting for the mutex to become free
OSWaitsTimeSec	The time spent in OS waits while waiting for the mutex to become free
OSWaitsTimeRateSec	The time per second spent in OS waits while waiting for the mutex to become free
MutexModule	The source module that defines the mutex

Network_Interface Table

Purpose
This table contains information about network I/O.

Table Description

Field	Description
BytesReceivedPerSec	The number of bytes received per second
BytesSentPerSec	The number of bytes sent per second

Query_Cache Table

Purpose
This table caches the SQL of a query and provides its results. It provides information about the cache hit rate.

Table Description

Field	Description
HitRatePct	% of SQL queries that were in the query cache
StatementsExec	The number of SQL statements that are currently being executed
StatementsExecPerSec	The number of SQL statements per second that are currently being executed.
QueryCacheSize	Query Cache Size in megabytes of memory
UsedMemoryMB	The megabytes of memory in the query cache that are used
FreeMemoryMB	The megabytes of memory in the query cache that are free
NumCachedStatements	The number of SQL statements that are in the cache
NumCachedInserts	The number of queries added to the query cache
NumLowMemPrunes	The number of queries that were deleted from the query cache because of low memory.
NumQueriesNotCached	The number of non-cached queries.
NumCachedTotalBlocks	The total number of blocks in the query cache.
NumCachedUsedBlocks	The number of used memory blocks in the query cache.
NumCachedFreeBlocks	The number of free memory blocks in the query cache.

Replication_Slave_Status Table

Purpose

This table stores information about the slave status of the replication slave server used by the master host server. It also stores connection-related and timestamp information.

Table Description

Field	Description
SlaveConnection	Whether the agent is able to connect to the replication slave server
SlaveConfigured	Whether the server is set up for slave replication
ReplicationTimeDiff	The difference in seconds between the time on the master server and replication slave server
ConnectionHost	The hostname or IP that the agent is using to connect to the replication slave server
ConnectionPort	The port that the agent is using to connect to the replication slave server
SlaveIOState	The current state of the replication slave's I/O thread
MasterHost	The master host that the slave is connected to
MasterUser	The user name of the account used to connect to the master
MasterPort	The port used to connect to the master
MasterLogFile	The name of the master binary log file from which the I/O thread is currently reading
ReadMasterLogPos	The position in the current master binary log file up to which the I/O thread has read
ExecMasterLogPos	The position in the current master binary file up to which the SQL thread has read and executed
RelayLogFile	The name of the relay log file from which the SQL thread is currently reading and executing.
RelayLogPos	The position in the current relay log file up to which the SQL thread has read and executed
RelayMasterLogFile	The name of the master binary log file containing the most recent event executed by the SQL thread
SlaveIORunning	Whether the I/O thread is running and has connected successfully to the master
SlaveSQLRunning	Whether the SQL thread is running
MasterSSLAllowed	Whether an SSL connection to the master is permitted and if the replication slave server has SSL enabled
MasterSSLVerifyServerCert	Whether this security feature is enabled
SecondsBehindMaster	The time difference in seconds between the slave SQL thread and the slave I/O thread.
LastIOErrno	The error number of the last error that caused the I/O thread to stop
LastIOError	The error message of the last error that caused the I/O thread to stop
LastSQLErrno	The error number of the last error that caused the SQL thread to stop
LastSQLError	The error message of the last error that caused the SQL thread to stop

Slave_Status Table

Purpose

This table stores information about the slave status of the host server.

Table Description

Field	Description
SlaveConfigured	Whether the server is set up for slave replication
SlaveIOState	The current state of the slave I/O thread
MasterHost	The master host that the slave is connected to
MasterUser	The user name of the account used to connect to the master
MasterPort	The port used to connect to the master
MasterLogFile	The name of the master binary log file from which the I/O thread is currently reading
ReadMasterLogPos	The position in the current master binary log file up to which the I/O thread has read
ExecMasterLogPos	The position in the current master binary file up to which the SQL thread has read and executed
RelayLogFile	The name of the relay log file from which the SQL thread is currently reading and executing.
RelayLogPos	The position in the current relay log file up to which the SQL thread has read and executed
RelayMasterLogFile	The name of the master binary log file containing the most recent event executed by the SQL thread
SlaveIORunning	Whether the I/O thread is running and has connected successfully to the master
SlaveSQLRunning	Whether the SQL thread is running
MasterSSLAllowed	Whether an SSL connection to the master is permitted and if the slave server has SSL enabled
MasterSSLVerifyServerCert	Whether this security feature is enabled
SecondsBehindMaster	The time difference in seconds between the slave SQL thread and the slave I/O thread.
LastIOErrno	The error number of the last error that caused the I/O thread to stop
LastIOError	The error message of the last error that caused the I/O thread to stop
LastSQLErrno	The error number of the last error that caused the SQL thread to stop
LastSQLError	The error message of the last error that caused the SQL thread to stop

Slow Query Entry

Purpose

This table stores information about a unique query collected from the Slow Query Log. When the queries are gathered, they are genericized and standardized in order to aggregate them with other similar queries. This is done by removing column values and other values that are of limited or no importance to the way a query operates. For instance, “SELECT * FROM customers WHERE name=’Jones’ LIMIT 2;” and “SELECT * FROM customers WHERE name=’Smith’ LIMIT 5;” would be considered instances of one unique query type. If the second query were instead to select data from a table named “pets”, they would be considered different.

SQL	A genericized instance of this SQL query string.
EarliestTime	The earliest time this query appeared in the collection period.
LatestTime	The latest time this query appeared in the collection period.
Users	The DB users who executed this query, separated by "|".
Count	The number of times this query was executed.
QueryTimeMin	The minimum execution time for this query.
QueryTimeMax	The maximum execution time for this query.
QueryTimeAvg	The average execution time for this query.
LockTimeMin	The minimum lock time for this query.
LockTimeMax	The maximum lock time for this query.
LockTimeAvg	The average lock time for this query.
RowsSentMin	The minimum number of rows returned from this query.
RowsSentMax	The maximum number of rows returned from this query.
RowsSentAvg	The average number of rows returned from this query.
RowsExaminedMin	The minimum number of rows examined during the execution of this query.
RowsExaminedMax	The maximum number of rows examined during the execution of this query.
RowsExaminedAvg	The average number of rows examined during the execution of this query.

Sort_Buffer Table

Purpose

This table contains information about the MySQL sort buffer.

Table Description

Field	Description
SortBufferSize	Buffer size that each thread needs to allocate to do a sort
RowsSortedPerSec	The number of rows sorted per second
SortRangePerSec	The number of sorts per second that used a range (i.e. used an index)
SortScanPerSec	The number of sorts per second that used a full table scan (i.e. read the entire table). High values generally indicate poor use of indexing.
MergePassesSortedPerSec	The number of merge passes that the sort algorithm has had to do per second

Tables Table

Purpose
This table contains information about tables within the database.

Table Description

Field	Description
DatabaseName	The database that the tablespace is in
TablespaceName	The tablespace name.
TableSpaceAutoExtend	The increment size for extending the size of an auto-extending tablespace when it becomes full.
TablespaceFreeMB	Megabytes of free space in the tablespace
TablespaceFreePct	Percentage of free space in the tablespace
TablespaceUsedMB	Megabytes of used space in the tablespace
TablespaceUsedPct	Percentage of used space in the tablespace
TablespaceTotalSizeMB	Megabytes of total size in the tablespace
NumRows	The number of rows in the tablsepace

Thread_Pool Table

Purpose
This table contains information about how the thread cache is operating.

Table Description

Field	Description
ThreadCacheSize	The maximum number of threads that can be cached
ThreadsCached	The number of threads in the thread cache
ThreadsConnected	The number of currently open connections
ThreadsCreatedPerSec	The number of new threads created per second. A high creation rate is indicative that the cache is too small.
ThreadsRunning	The number of active (non sleeping) threads
ThreadPoolHitRatePct	The number of threads that were used from the pool in comparison to the total number of connections made

Top_Sessions Table

Purpose

This table contains information about sessions that have the longest running commands, excluding the master and slave replication, idle sessions and the session that the agent is connected to.

Table Description

Field	Description
SessionsId	The MYSQL id for the session
SessionsUser	The user ID that the session is connected as.
SessionsDatabase	The database that the session is using
SessionsSeconds	The number of seconds that the command has been executing.
SessionsCommand	A portion of the command that is currently executing.
SessionsState	The state of the session
SessionsSQL	A portion of the SQL that is currently executing