Problem Identification and Resolution Using PMM - A Case Study

Johan Nilsson - Verisure
Michael Coburn - Percona
About Us

Michael Coburn
Product Manager, Percona
• PMM and Percona Toolkit
• Joined Percona seven years ago as a MySQL Consultant

Johan Nilsson
Senior DBA, Verisure Innovation, Sweden
johan.nilsson@verisure.com
• Unix/Linux system administrator since 1999
• MySQL/Oracle DBA since 2008
PMM Background
About PMM

- Open Source database troubleshooting and optimization tool
- Architecture - client/server
- Dashboards for MySQL, MongoDB, and PostgreSQL + OS
- Query Analytics - identify the queries consuming the most amount of time
- Runs in your environment (not a SaaS)
PMM Distribution Methods

1. **docker**
   - `docker pull percona/pmm-server:1`

2. **Virtual Appliance**
   - Supports VMware, RedHat Virtualization, Microsoft Systems Center
   - … and VirtualBox!

3. **AWS Marketplace**
   - Production-ready AMI running in EC2
AWS Marketplace

- Deploy directly to EC2
- Running CentOS 7

Search for "pmm" or "Percona Monitoring and Management"

https://aws.amazon.com/marketplace/pp/B077J7FYGX
● pmm-client (eg. MySQL host)
  ○ mysqld_exporter - MySQL metrics
  ○ node_exporter - Linux/OS metrics
  ○ qan-agent - Query Analytics

● PMM Server
  ○ Query Analytics
    ■ QAN API and QAN Application
  ○ Metrics Monitor
    ■ Prometheus
    ■ Grafana
Verisure Innovation
About Verisure

- European leader in monitored alarm systems
- Present in 14 countries
- 3 million+ customers
- 10k employees
- Building the full stack inhouse
- Two innovation centers
  - Malmö (Sweden)
  - Madrid (Spain)
- Yes, we are hiring!
  
  https://verisure.avature.net/careers
Verisure Versus MySQL

- Core application running in Sybase ASE
- “Bulk” installation data stored in MySQL
  - Driving mobile app and web page
  - Clustered, multi-site setup
- “Always on” - 100% uptime on the service
  - We are in the alarm monitoring business…
- Growing fast, adding 1 MySQL server-pair every 4th week
Verisure vs PMM

● Why PMM?
  ○ Early adopter of Query Analytics, first install 2016
    ■ Recommended and demonstrated by Percona consultant
  ○ Replaced by PMM, first install v1.0.6, 2016

● The journey, together with Percona:
  ○ Some minor bugs found, and reported to Percona
  ○ Performance problems in early versions
    ■ Especially identified when monitoring many servers
      • Forcing us to tweak “everything”
  ○ Close cooperation with Percona support and developer
    ■ Acting in some cases as a benchmark
How Are We Using PMM?
How And What Are We Using PMM For?

Three main uses:
- Preventive statistics gathering and capacity planning
- Troubleshooting during production problems
- PMR and RCA - troubleshooting problems after resolving

Tools in PMM
- Performance graphs
- Query statistics - mysql-prompt

Additional benefits:
- Possible to gather valuable information for bug reports, filed
  - Internally
  - MySQL
  - Percona
Preventive Use

- Trends - using graphs
  - the mysql-instance is leaking memory...
- Pattern finding
  - query statistics (mysql-cli)
Troubleshooting During Issues

• Application grinds to a halt
• mysql hangs on the application database
  ○ long-running query
  ○ recurring problem
• no os-level issues
  ○ some strange patterns
Troubleshooting During Issues
PMRs and RCAs

- “Going back in time”
  - possible to get detailed graphs
  - finding problem originating time
  - correlate to application changes
  - searchable slow query statistics
Conclusions

Great tool!

- Statistics gathering tailor-made for MySQL / Percona Server / ProxySQL
- Low initial effort
- Easily customised graphs and dashboards
  - “all” data already collected by the exporter
Replication delay graphs
Next Steps

Next steps:
- Upgrade PMM version (still on 1.13)
- More tailor-made graphs
  - comparing cross nodes / sites
- Adding alerting based on graphs
- Adding more nodes (we were blocked by MySQL-bug)
- Adding more types of nodes:
  - Cassandra
  - Sybase ASE
Semi-sync Graph Example

- PMM was already collecting semi-sync replication data, but it wasn't getting displayed
- … so let’s do that!
Prepared Statements Versus Executions

- Ever wondered how often you executed a statement versus how many times you issued a prepare?
- … We did, so we made a graph!
What's New and Exciting?

- Query Analytics
  - Support for large environments
  - Filtering using query metadata and labels
  - Sorting and additional columns
  - Support for PostgreSQL
- `pmm-agent`
- Standard and Custom labels
- Inventory Overview
- Inventory API
- Multiple scraping intervals
Filtering

- Standard and Custom labels can be used as filter conditions
- Additional filters include:
  - Client Host
  - Schema
  - Client Username
  - Database Server
## Sorting

- Sort by any column

<table>
<thead>
<tr>
<th>#</th>
<th>Query</th>
<th>Count</th>
<th>Query Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TOTAL</td>
<td>100.00%</td>
<td>0:05:00</td>
</tr>
<tr>
<td>1</td>
<td>show global status</td>
<td>17.30%</td>
<td>51.98 sec</td>
</tr>
<tr>
<td>2</td>
<td>select t.table_schema, t.table__...</td>
<td>13.61%</td>
<td>40.80 sec</td>
</tr>
<tr>
<td>3</td>
<td>select event_name, count_star...</td>
<td>12.20%</td>
<td>36.85 sec</td>
</tr>
<tr>
<td>4</td>
<td>SELECT <code>performance_schema</code></td>
<td>9.52%</td>
<td>28.82 sec</td>
</tr>
<tr>
<td>5</td>
<td>SHOW GLOBAL STATUS</td>
<td>8.73%</td>
<td>26.22 sec</td>
</tr>
<tr>
<td>6</td>
<td>select name, subsystem, type, ...</td>
<td>8.65%</td>
<td>25.99 sec</td>
</tr>
<tr>
<td>7</td>
<td>SELECT <code>EVENT_NAME</code> , <code>...</code></td>
<td>6.06%</td>
<td>18.21 sec</td>
</tr>
<tr>
<td>8</td>
<td>select table_schema, table_name...</td>
<td>5.85%</td>
<td>17.59 sec</td>
</tr>
<tr>
<td>9</td>
<td>SELECT NAME , <code>subsystem</code> ...</td>
<td>4.40%</td>
<td>13.22 sec</td>
</tr>
<tr>
<td>10</td>
<td>select event_name, count_real...</td>
<td>1.52%</td>
<td>4.57 sec</td>
</tr>
</tbody>
</table>
Add Additional Columns

- Add more columns based on query metadata
- Examples:
  - Rows Examined
  - Temporary Tables
  - Filesort

![Query Analysis Table]

- **TOTAL**
  - Query: show global status
  - Duration: 52.64 sec

- **SELECT performance_schema...**
  - Duration: 49.93 sec

- **SHOW GLOBAL STATUS**
  - Duration: 40.93 sec

- **SELECT subsystem, type...**
  - Duration: 49.93 sec

- **SELECT NAME, 'subsystem...**
  - Duration: 49.93 sec

- **select event_name, count_real...**
  - Duration: 49.93 sec

*Results per page: 10*
Thank You to Our Sponsors
Rate My Session

Introducing gh-ost: triggerless, painless, trusted online schema migrations

11:20 - 12:10
Matterhorn 2