Graphing MySQL performance with Prometheus and Grafana

Roman Vynar
PromCon 2016, Berlin

PERCONA

About Percona

Build

Percona's industry-leading architecture design consultants have full stack expertise to build databases in hosted and private, public and hybrid cloud environments that are optimized and scale for growth while minimizing application downtime and operational costs.

Fix

Percona's 24 x 7 support experts provide emergency troubleshooting to solve database and server instability, data recovery, performance, response and outage issues, as well as proactive systems monitoring and alert responses.

Optimize

Percona's industry-recognized performance experts can maximize your database, server and application performance, lower infrastructure costs and provide capacity and scalability planning for future growth.

Manage

Percona's managed services team can augment your staff with trusted data advisors to provide project management, best practices for database operations, secure data backups and remote infrastructure management.

PERCONA

Software from Percona

Percona
XtraDB Cluster

Percona
Server for MySQL

Percona
Server for MongoDB

Percona
XtraBackup

Percona
Toolkit

PERCONA

Percona Monitoring and Management project

Percona Monitoring and Management (PMM) is an open-source platform for managing and monitoring MySQL and MongoDB performance. It is developed by Percona in collaboration with experts in the field of managed database services, support and consulting.

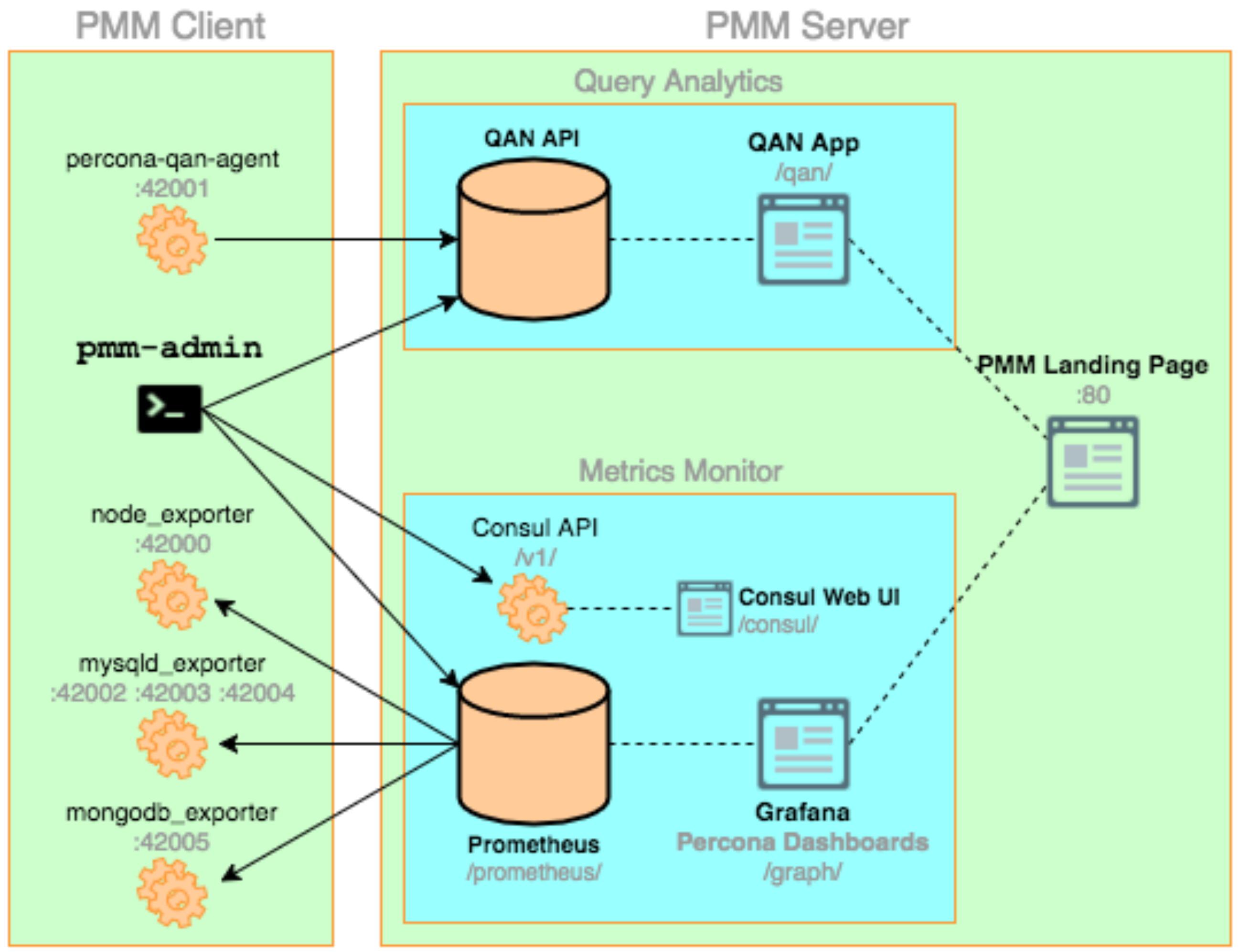
PMM is a free and open-source solution that you can run in your own environment for maximum security and reliability. It provides thorough time-based analysis for MySQL and MongoDB servers to ensure that your data works as efficiently as possible.

Docs: https://www.percona.com/doc/percona-monitoring-and-management/index.html

Live demo: https://pmmdemo.percona.com

PERCONA
Monitoring and Management

PMM components



PERCONA

Monitoring ecosystem

Grafana and Prometheus is a great tandem for monitoring and visualizing MySQL and system performance.

Prometheus:
- simple to use
- efficient
- powerful query language

Prometheus exporters:
- can’t be simpler

Grafana:
- simple use
- nice visualization
- flexible interface

mysqld_exporter flags

mysql high resolution (1s-5s):

- -collect.global_status
- -collect.info_schema.innodb_metrics

mysql medium resolution (5s):

- -collect.slave_status
- -collect.info_schema.processlist
- -collect.info_schema.query_response_time
- -collect.perf_schema.eventswaits
- -collect.perf_schema.file_events
- -collect.perf_schema.tablelocks

mysql low resolution (60s):

- -collect.global_variables
- -collect.info_schema.tables
- -collect.auto_increment.columns
- -collect.binlog_size
- -collect.info_schema.tablestats
- -collect.info_schema.userstats
- -collect.perf_schema.indexiowaits
- -collect.perf_schema.tableiowaits

Other:

- -collect.engine_tokudb_status
- -collect.info_schema.clientstats
- -collect.info_schema.innodb_tablespaces
- -collect.perf_schema.eventsstatements

3-in-1 mysqld_exporter

- https://github.com/percona/mysqld_exporter

```
reg = prometheus.NewRegistry()
reg.MustRegister(NewExporter(dsn))
http.Handle("/metrics-hr", promhttp.HandlerFor(
  reg, promhttp.HandlerOpts{},
))

reg = prometheus.NewRegistry()
reg.MustRegister(NewExporterMr(dsn))
http.Handle("/metrics-mr", promhttp.HandlerFor(
  reg, promhttp.HandlerOpts{},
))

reg = prometheus.NewRegistry()
reg.MustRegister(NewExporterLr(dsn))
http.Handle("/metrics-lr", promhttp.HandlerFor(
  reg, promhttp.HandlerOpts{},
))

http.HandleFunc("/", func(w http.ResponseWriter, r *http.Request) {
  w.Write(landingPage)
})
```

PERCONA

Prometheus config

```
scrape_configs:
  - job_name: mysql-hr
    metrics_path: /metrics-hr
    scrape_interval:    1s
    scrape_timeout:     1s
    ...

  - job_name: mysql-mr
    metrics_path: /metrics-mr
    scrape_interval:    5s
    scrape_timeout:     1s
    ...

  - job_name: mysql-lr
    metrics_path: /metrics-lr
    scrape_interval:   60s
    scrape_timeout:     5s
    ...
```

PERCONA

Running mysqld_exporter

MySQL DSN can be set by 2 ways:

- as environment variable:

  ```
  export DATA_SOURCE_NAME="user:password@(localhost:3306)/
  ```

- through MySQL ini file and the flag:

  ```
  -config.my-cnf=“~/.my.cnf”
  ```

  ```
  # cat ~/.my.cnf
  [client]
  user=root
  password=abc123
  ```

PERCONA

node_exporter

Here is the minimal set of collectors for node_exporter we use for the graphs:

- diskstats
- filesystem
- loadavg
- meminfo
- netdev
- stat
- time
- uname
- vmstat

Grafana patch

To align graph step with interval passed to Prometheus query using rate() there is a need to apple a small patch.
https://github.com/grafana/grafana/pull/5839

Grafana 2.6.0:

```
sed -i 's/step_input:""/step_input:c.target.step/; s/ HH:MM/ HH:mm/;
s/,function(c)/,"templateSrv",function(c,g)/; s/expr:c.target.expr/
expr:g.replace(c.target.expr,c.panel.scopedVars)/' /usr/share/grafana/public/app/plugins/datasource/prometheus/
query_ctrl.js

sed -i 's/h=a.interval/h=g.replace(a.interval, c.scopedVars)/' /usr/share/grafana/public/app/plugins/
datasource/prometheus/datasource.js
```

Grafana 3.x:

```
sed -i 's/expr=\(.\)\.replace(\(.\)\.expr,\(.\)\.scopedVars\(.*\)var \(.\)=\(.\)\.interval/expr=
\1.replace(\2.expr,\3.scopedVars\4var \5=\1.replace(\6.interval, \3.scopedVars)/' /usr/share/grafana/public/
app/plugins/datasource/prometheus/datasource.js

sed -i 's/,range_input/.replace(\/"{\/g,"\\"").replace(\/}"\/g,"\\""),range_input/; s/step_input:""/
step_input:this.target.step/' /usr/share/grafana/public/app/plugins/datasource/prometheus/query_ctrl.js
```

Grafana 4.x (unreleased):

```
There won't be a need to apply this patch.
```

Grafana dashboards

- https://github.com/percona/grafana-dashboards
  - Cross Server Graphs
  - Disk Performance
  - Disk Space
  - Galera Graphs
  - MySQL InnoDB Metrics
  - MySQL InnoDB Metrics Advanced
  - MySQL MyISAM Metrics
  - MySQL Overview
  - MySQL Performance Schema
  - MySQL Query Response Time
  - MySQL Replication
  - MySQL Table Statistics
  - MySQL User Statistics
  - Prometheus
  - Summary Dashboard
  - System Overview
  - TokuDB Graphs
  - Trends Dashboard

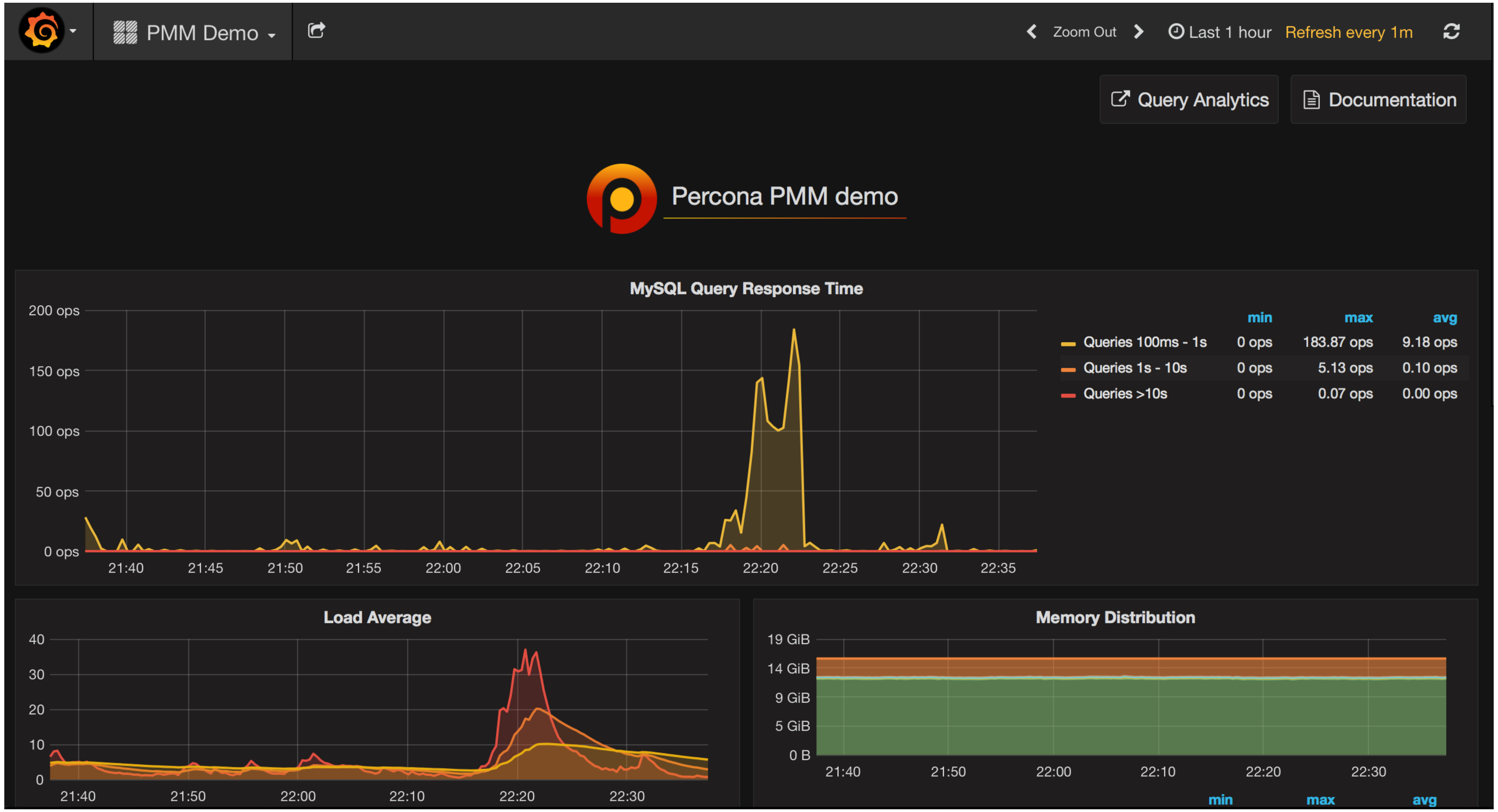
Dashboard graphing tips

Some interesting things we apply to the graphs:

- use Y-min 0 for each graph
- wherever we use `rate()[$interval]` we add the same query with `or irate()[5m]` to make it so the graphs are still shown when you choose interval lower than resolution of data available
- shared cursor on each dashboard
- null value as “null", not “connected” because if you have missed data better to show the gaps rather than connected line and to introduce confusion

PERCONA

Live presentation



	min	max	avg
Queries 100ms - 1s	0 ops	183.87 ops	9.18 ops
Queries 1s - 10s	0 ops	5.13 ops	0.10 ops
Queries >10s	0 ops	0.07 ops	0.00 ops

Prometheus vs InfluxDB (February, 2016)

	Prometheus TSDB	**InfluxDB**
Storage reliability	Corruptible	Unclear
Storage redundancy	Remote storage. Federation	Clustering
Storage backup	No possibility	Available
Data retention	One global	Per database
Data aggregation	Recording rules	Continuous queries
Query language	Quite powerful	Very basic

As of Prometheus **0.17.0rc2** and InfluxDB **0.10.0**.

Thanks!
Questions?

Percona Live Amsterdam

Join us for Percona Live Amsterdam 2016, the premier Open Source Database Conference, to take place on October 3-5, 2016: https://www.percona.com/live/plam16/