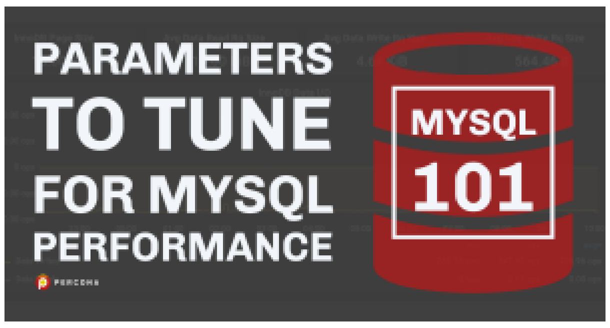
MySQL 101: Parameters to Tune for MySQL Performance

30 Jun 2020

By Brian Sumpter Insight for DBAs, Monitoring, MySQL insight for DBAs, Monitoring, MySQL, mysql-and-variants 3 Comments



While there is no magic bullet for MySQL tuning, there are a few areas that can be focused on upfront that can dramatically improve the performance of your MySQL installation. While much information has been published on this topic over the years, I wanted to break down some of the more critical settings that anyone can implement with no guesswork required.

Depending on the version of MySQL you are running, some of the default values used in this post may differ from your install, but the premise is still largely the same.

Initial MySQL performance tuning can be broken down to the following categories:

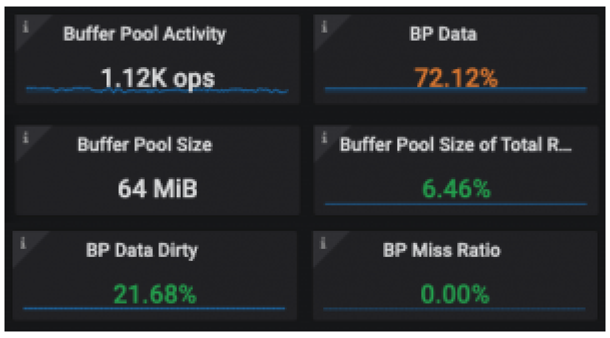
- **Tuning for your hardware**
- **Tuning for best performance / best practices**
- **Tuning for your workload**

Tuning MySQL for Your Hardware

Depending on the hardware you have installed MySQL on, some variables need to be set based on the machine (or VM) specifications. The following variables are largely dependent on your hardware:

innodb_buffer_pool_size

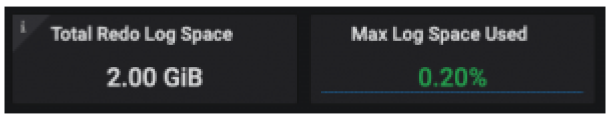
- Generally, set to 50% – 70% of your total RAM as a starting point.
- It does not need to be set any larger than the total database size.
- Percona Monitoring and Management (PMM) can offer additional insight, showing your buffer pool usage and allowing you to tune accordingly.



Buffer Pool Activity 1.12K ops
BP Data 72.12%
Buffer Pool Size 64 MiB
Buffer Pool Size of Total R... 6.46%
BP Data Dirty 21.68%
BP Miss Ratio 0.00%

innodb_log_file_size

- This is generally set between 128M – 2G.
- Should be large enough to hold at most an hour or so of logs.
    - This is more than enough so that MySQL can reorder writes to use sequential I/O during the flushing and checkpointing processes.
- PMM can offer additional insight, as if you are using more than 50% of your log space, you may benefit from a log fi ncrease



Total Redo Log Space 2.00 GiB
Max Log Space Used 0.20%

innodb_flush_log_at_trx_commit

- Setting to “1” (default in 5.7) gives the most durability.
- Setting to “0” or “2” will give more performance, but less durability.

innodb_flush_method

- Setting this to O_DIRECT will avoid a performance penalty from double buffering.

MySQL Tuning for Best Performance/Best Practices

innodb_file_per_table

- Setting this to "ON" will generate an independent InnoDB table space for every table in the database.

innodb_stats_on_metadata

- Setting this to "OFF" avoids unnecessary updating of InnoDB statistics and can greatly improve read speeds.

innodb_buffer_pool_instances

- A best practice is to set this to "8" unless the buffer pool size is < 1G, in which case set to "1".

query_cache_type & **query_cache_size**

- Setting both of these to "0" will entirely disable the query cache.

Tuning for Your Workload

To tune further, more information will be required. The best way to gather this information is to install a MySQL monitoring / graphing tool like Percona Monitoring and Management platform. Once you have a tool installed, we can dive into the individual metrics and start customizing based on the data.

I would recommend starting with one of the most impactful variables **–** the **innodb_buffer_pool_size**. Compare the RAM and number of free pages on your instance to the total buffer pool size. Based on these metrics, you can determine if you need to increase or decrease your overall buffer pool size setting.

Next, take a look at your metrics for the InnoDB Log File usage. The rule of thumb is that your log files should hold approximately one hour of data. If you see that your data written to the log files hourly exceeds the total size of the log files, you would want to increase the **innodb_log_file_size** variable and restart MySQL. You could also verify with "SHOW ENGINE INNODB STATUS;" via the MySQL CLI to assist in calculating a good InnoDB log file size.

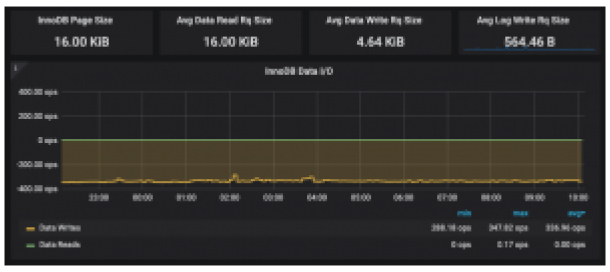
Other Settings

Other InnoDB settings that can be further tuned for better performance are:

innodb_autoinc_lock_mode

- Setting this to "2" (interleaved mode) can remove the need for an auto-inc lock (at the table level) and can increase performance when using multi-row insert statements to insert values into a table with an auto increment primary key. Note that this requires either ROW or MIXED binlog format.

innodb_io_capacity / **innodb_io_capacity_max**

- These settings will impact your database if you are utilizing a write-heavy workflow. This does not apply to read (SELECT) traffic. To tune these values, it is best to know how many iops your system can perform. It is a good idea to run sysbench or another benchmark tool to determine your storage throughput.
- PMM can offer additional insight, showing your IO usage and allowing you to tune accordingly.
  - 

In Summary

While this is by no means a comprehensive article on MySQL tuning, the suggestions above should clear some of the low hanging fruit and get your system closer to an ideal setup. As with all database tuning, your process should be an ongoing one based on current informati

- Examine the settings proposed above, and implement if they make sense for your environment/workload.

- Install a good monitoring tool to give insight into the database (Percona Monitoring and Management is our suggestion).
- Stay current on your monitoring graphs to determine other areas where you may need to tune.

Our solution brief “Get Up and Running with Percona Server for MySQL” outlines setting up a MySQL® database on-premises using Percona Server for MySQL. It includes failover and basic business continuity components.

Download PDF

Related

Tips for MySQL 5.7 Database Tuning and Performance April 1, 2020 In "Insight for DBAs"	MySQL Performance Tuning Settings January 28, 2014 In "Insight for DBAs"	MySQL 5.7 Performance Tuning After Installation October 12, 2016 In "MySQL"

Author

Brian Sumpter

Share this post

Comments (3)

Jim Tommaney

Reply

Hi Brian,
Great write-up! Impact of innodb_stats_on_metadata was completely new to me.
We also evaluated innodb_io_capacity and innodb_read_ahead_threshold at various settings.
workloads that I could find either.
We found good results with changing buffer_pool instances and read_io_threads together in c
Cloud storage, analytic queries, concurrent workload, parallel query enabled (Ali Cloud), 64 cores, o_direct.
innodb_buffer_pool_instances (change from 8 to 32)
innodb_read_io_threads (change from 4 to 16)

Under a 100% physical I/O workload this resulted in a 4x speedup. You mileage will vary, test before production usage. Most query workloads are not 100% PIO, so actual benefits likely much lower.

Cheers,
Jim

June 30, 2020 at 2:08 pm

Jie Zhou

Reply

Why innodb_buffer_pool_instances has a best practice to set it to “8”

July 1, 2020 at 12:04 pm

Brian Sumpter

Reply

Hi Jie Zhou. The MySQL default is to set innodb_buffer_pool_instances to 8 (in MySQL version 5.7 and up) as this is a good starting point for most general use cases. As per the MySQL documentation, this option takes effect only when you set innodb_buffer_pool_size to a size of 1GB or more. The total size you specify is divided among all the buffer pools. For best efficiency, specify a combination of innodb_buffer_pool_instances and innodb_buffer_pool_size so that each buffer pool instance is at least 1GB.

July 8, 2020 at 1:11 pm

Leave a Reply

Enter your comment here...

HOW CAN WE HELP?

Percona's experts can maximize your application performance with our open source database support, managed services or consulting.

Contact us

SUBSCRIBE

Want to get weekly updates listing the latest blog posts? Subscribe now and we'll send you an update every Friday at 1pm ET.

Subscribe to our blog

CATEGORIES

MySQL(3355)

Insight for DBAs(1528)

Percona Software(1479)

Percona Events(870)

MongoDB(549)

Insight for Developers(472)

Benchmarks(340)

Percona Live(332)

Webinars(287)

Cloud(276)

PostgreSQL(176)

Monitoring(164)

MariaDB(157)

Hi, welcome to Percona - home of the database performance experts.

Percona Services(138)

ProxySQL(127)

Security(125)

Hardware and Storage(105)

Storage Engine(52)

Database Trends(46)

Percona Announcements(10)

Percona Blog RSS Feed

UPCOMING WEBINARS

The Open Source Alternative to Paying for MongoDB
Why PostgreSQL Is Becoming A Migration Target For Enterprise
How To Measure Linux Performance Wrong
Converting MongoDB to Percona Server for MongoDB
Moving MongoDB to the Cloud: Strategies and Points To Consider

All Webinars »

Services

Support
Managed Services
Consulting
Training

Products

MySQL Software
MongoDB Software
PostgreSQL Distribution
Kubernetes
Monitoring & Management

Resources

Solution Briefs
White Papers
Webinars
Case Studies
Datasheets

More

Blog
Community Blog
Technical Forum Help

Contact Us

Sales & General Inquiries
(888) 316-9775 (USA)
(208) 473-2904 (USA)
+44 203 608 6727 (UK)
0-808-169-6490 (UK)
0-800-724-4569 (GER)

About

Customers
Newsroom
About
Careers

Terms of Use | Privacy | Copyright | Legal