White Paper

MySQL Scalability: Top 5 Tricks to Make Your Databases Fly

ScaleArc

Introduction

The world's most widely used database, MySQL, has a reputation for being extremely easy to use but notoriously hard to master and scale. You can find countless books and articles by experts, and the famous "large scale" giants such as Google and Facebook offer grand presentations on how to "Scale MySQL." Sadly, determining what works for you out of that mass of content can be quite hard, and judging which changes you can make with limited engineering resources is even more difficult. After all, most techniques used by Google, Facebook, and the like are very exotic and require a lot of custom code and hordes of highly skilled engineers to implement.

Most techniques used to scale MySQL are very exotic and require a lot of custom code and hordes of highly skilled engineers to implement.

This list is designed for the rest of us mere mortals – a quick set of things you can do to significantly scale your MySQL database without requiring a ton of time or resources. These tricks offer the biggest bang for the buck – they are simple, cheap, and easy to implement, and they get you 90% there without a lot of time and effort. Let's begin.

1. Don't run MySQL

Wow! What a first tip, right? Let's just take away the database, and then we have no more database troubles! What we mean by "MySQL" is the MySQL community version of the software. It isn't necessarily the most optimized, stable, functional version of MySQL. Percona Server is a version of MySQL software compiled and maintained by Percona, one of the top MySQL performance consultants out there. It is generally more stable and higher-performing than MySQL's community release while being fully compatible with MySQL.

The effort to migrate isn't massive or time consuming; in most cases, you can just take a MySQL dump and restore it on Percona Servers pretty quickly. We were able to migrate a ~3GB WordPress database from MySQL 5.5 to Percona Server 5.6 in shorter than 15 minutes. The performance and stability under load are certainly worth it. Just look here! You can even set up Percona Server as a slave to your existing MySQL server and, once the data is in sync, seamlessly migrate over to it with database load balancing software such as from ScaleArc.

2. Manage connections better

MySQL's query performance isn't too bad, but its connection creation and handling performance leaves a lot to be desired. You could do as many as 20,000 typical WordPress or Drupal queries per second out of a fairly basic Amazon 4XL instance, but at barely 2,000 connections per second, it starts spewing errors or timeouts.

Unless your application performs connection pooling (Apache's DBD or JDBC connection pooling) and reuses existing connections, it's next to impossible to hit MySQL's peak performance. Most LAMP stacks aren't configured out of the box to perform connection pooling, and some apps

aren't fully compatible with connection pooling. But if your app happens to be one that can utilize connection pooling, enable it, and see the massive gains in performance that can be achieved. Database load balancing software can also be used to provide connection pooling to MySQL-powered applications. The software not only improves performance by reducing connection churn but also considerably reduces the number of connections on your MySQL server.

3. Know your bottlenecks, fix top offenders

Many MySQL users we talk to aren't aware of what really consumes most resources on their MySQL servers. Many use the MySQL slow query log as a dipstick to measure which queries are taking longer than 2 to 5 seconds to process, but that tool really isn't sufficient to help you find and solve peak load-related issues. Facebook has entire teams of engineers dedicated to finding that extra 0.5 milliseconds it takes to run a query due to storage, memory, or engine issues, in hopes of optimizing the kind of queries that run millions of times an hour. Even the smallest improvement in performance can cut the server's CPU usage massively.

If you focus on finding your most popular queries that consume a majority of resources and improve the top 10, you'll gain a significant boost in performance and efficiency.

You don't need to go that far, but if you focus on finding your most popular queries that consume a majority of resources and improve the top 10, you'll gain a significant boost in performance and efficiency. You can use tools like Percona's pt-query-digest to find such queries manually through MySQL's general logs or TCPDump, or you can leverage real-time analytics in database load balancing software such as ScaleArc to provide new insights into your application traffic. For example, ScaleArc's built-in, real-time SQL analytics instantly make clear what's hurting your MySQL performance the most and show how improvements will benefit you. Regularly reviewing your SQL query patterns is a very crucial part of maintaining a high performance MySQL environment.

4. Use the right table types and indexes

Now that you know which queries hurt you the most, let's get to fixing them. In many cases, query performance issues can be solved, or significantly reduced, by simply optimizing the table types and indexes associated with them. If you have a custom app, you may not have considered the impact that a query's one extra SQL variable can have on the overall performance of the system. Yet think of the effect when that query is run thousands of times a minute, with each time slower by just 0.3 seconds because the table doesn't have an index for the "userlocation" field.

Find the queries that take the most resources with the pt-query-digest tool or ScaleArc's analytics. ScaleArc analytics plot your most resource-intensive queries and show you and index which columns they query frequently. Then use the right tables for the job; here is a simple primer on MyISAM vs InnoDB for MySQL – if you're on Percona Server, you can use XtraDB instead of InnoDB. Certain tables, such as reporting tables which

are written to all the time but never updated, can also use the Archive engine in MySQL to conserve memory and disk space.

5. Use multiple servers

For this last trick, we debated whether to talk about caching with NoSQL solutions such as memcached or redis or about replication and read/write split and load balancing. We finally came to the conclusion that going from one server to two was a lot simpler from an application-design standpoint, more cost effective, and better in terms of high availability – so here goes.

MySQL has always been ahead of the curve when it comes to the ease of setting up replication and managing replicas. A MySQL slave server that replicates all the data from your current server will offer you many benefits, from being able to have an always-ready failover server to having a second server to serve read queries or support large reporting queries.

Most popular eCommerce, content, or CRM/ERP apps written on top of MySQL perform significantly more reads than writes, so they are relatively easy to scale when you have a second server available. You can take a few simple steps when you set up a slave that will significantly offload your primary server and nearly double your throughput:

- Run large, repeating reporting queries and batch jobs on the slave instead
- Point completely read-only pages on your web app, such as home pages on content sites or main listing pages of forums, to serve from the slave
- Make your application understand HTTP useragent headers, and when it sees a crawler such as Google using your website in the headers, hardwire all queries to go to the slave, as crawlers won't do any write-type actions

These simple tricks can significantly reduce the number of queries going to the primary server and provide a simple way to load balance your read load between both the primary and secondary server. This architecture is known as read/write split. Database load balancing software can also provide you the ability to perform automatic read/write split on all SQL traffic. the ScaleArc software, for example, is also replication aware and capable of load balancing queries to more than just one slave, so it enables seamless, automatic failover in case your master server fails. As an added performance boost, the ScaleArc software can also let you cache your most popular read queries within minutes, reducing the overall traffic to your SQL servers.

By implementing these five tricks, you'll have your MySQL infrastructure scaling performance quickly, improving application performance and availability and simplifying scale out and scale up.

ScaleArc

2901 Tasman Drive, Suite 205
Santa Clara, CA 95054
Phone: 1-408-780-2040
Fax: 1-408-427-3748
www.scalearc.com

ScaleArc is the leading provider of database load balancing software. The ScaleArc software inserts transparently between applications and databases, creating an agile data tier that provides continuous availability and increased performance for all apps. With ScaleArc, enterprises also gain instant database scalability and a new level of real-time visibility for their application environments, both on prem and in the cloud. Learn more about ScaleArc, our customers, and our partners at **www.scalearc.com.**

01/08/15