MySQL Performance Tuning

Oli Sennhauser
FromDual GmbH
Uster / Schweiz

Schlüsselworte:

MySQL, Performance, Tuning, Explain, Datenbank-Tuning

Einleitung

MySQL Performance Tuning ist früher oder später ein Thema, mit welchem sich jeder Admin zu befassen hat. Es macht hierbei Sinn, sich einen systematischen Weg zuzulegen, wie Performance-Probleme erfasst, analysiert und behoben werden.

Performance Waage

Grundsätzlich lassen sich 5 verschiedene Einflussspähren isolieren, welche einen direkten Einfluss auf die Performance einer Datenbank haben können: Das Betriebssystem, die Hardware, die Datenbankkonfiguration, die auf die Datenbank zugreifende Applikation und das allgemeine Design und die Architektur des Gesamtsystems.
Bei diesen 5 Einflusssphären liegen die grössten Möglichkeiten für Performancesteigerungen in den beiden letztgenannten: Der Applikation und dem Design und der Architektur.

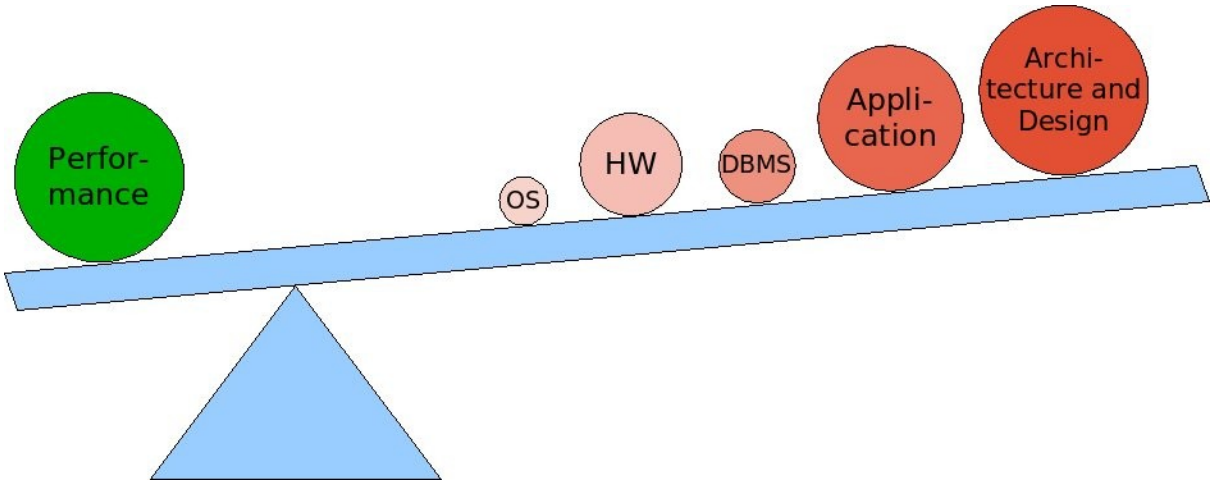

Abb. 1: Performance Tuning Balance

Zusammentragen von Fakten

Wenn der Aufschrei erklingt, dass die Datenbank langsam ist, gilt es als erstes einmal die Fakten zusammenzutragen, wo genau das Problem liegt. Hierbei hilft festzuhalten, wie sich das Problem genau manifestiert.
Dann sollte man abklären, ob historische Messdaten vorliegen, an welchen man ggf. sehen könnte, was aus Systemsicht geändert hat (graphische Daten oder Logfiles). Am besten ist es, wenn man entweder das Problem gezielt simulieren kann oder es vorhersagbar oder periodisch auftritt.

Die Aufgabe bis hierhin lautet: Finde das Muster...

Einflussgrössen auf Performance und kritische Ressourcen

Bei der Isolierung des Problems haben wir auf der Hardwareseite die Möglichkeit durch das Finden der kritischen Ressourcen das Problem weiter einzukreisen. Die Fragestellung lautet hier: Wo liegt der Flaschenhals? Als Kandidaten kommen in Frage: Die CPU, der Speicher (RAM), das I/O-System (IOPS oder Durchsatz) und das Netzwerk (FpS oder Durchsatz).
Wenn wir den Flaschenhals gefunden haben können wir eine weitere Aussage darüber machen, was das Performance-Problem verursachen könnte.
Die kritischen Ressourcen finden wir mit Kommandozeilen-Tools wie: top, vmstat, mpstat, free, ps, iostat, pidstat, dstat oder ifconfig.

Als Stellschrauben bieten sich hier an: Schnellere oder mehr Cores, mehr RAM, ein besseres I/O-System, ein Netzwerk mit 1 Gbit oder mehr Bandbreite. Betriebssystem seitig habe wir die Möglichkeit neue 64-bit Kernel einzusetzen, ein performanteres Filesystem oder für Datenbanksysteme bessere geeignete I/O Scheduler zu verwenden.
Wenn wir eine top-Performance für unser Datenbank erreichen wollen sind sowohl Virtualisierungslösung als auch SAN-Systeme nicht geeignet.

MySQL Grob-Tuning

Bevor mit dem Datenbanktuning angefangen wird, sollte man sich überlegen, welche Storage Engine verwendet wird. Zudem sollte man sich auch Gedanken machen auf einen relativ neuen MySQL Release zu wechseln um von den zahlreichen Performance-Verbesserungen profitieren zu können. MySQL 5.5 oder mindestens MySQL 5.1 mit dem InnoDB Plugin sind hier empfohlen.
Zur Zeit gibt es ca. 330 MySQL Parameter an welchen geschraubt werden kann. Aber nur ca. 8 davon sind primär signifikant für ein MySQL Grobtuning.
Bei Verwendung der InnoDB Storage Engine sind dies:
innodb_buffer_pool_size. Diese Variable steuert, wie viel Speicher InnoDB für das Cachen der Daten-Blocks verwenden soll. Auf einer dedizierten InnoDB Maschine kann dieser Wert auf bis zu 80% des zur Verfügung stehenden RAM gesetzt werden.
innodb_log_file_size sollte für eine Top-Schreibperformance so gross wie möglich gesetzt werden (2 Gbyte). Grosse InnoDB Log Files haben jedoch den Nachteil von längeren Recoveryzeiten nach einem Crash.
Wird mehr Wert auf Schreib-Performance als auf Crash-Resistenz der Daten gelegt, kann auch innodb_flush_log_at_trx_commit auf 0 oder 2 gestellt werden um harte fsyncs auf Platte zu vermeiden.

Wenn das Binary Logging eingeschaltet ist kann aus Performance-Gründen auch sync_binlog ausgeschaltet werden. Dies hat insbesondere dann einen signifikanten Einfluss wenn kein batteriegepuffertes I/O-System zur Verfügung steht.

Wird mit MyISAM-Tabellen gearbeitet, sollte der MyISAM Key Buffer mittels der Variable key_buffer_size auf ca. 25-33% des zur Verfügung stehenden RAM's gesetzt werden um die Indices optimal zu cachen.
Der table_open_cache sowie der table_definition_cache sind für den Zugriff auf die eigentlichen Daten-Tabellen zuständig. Die Default-Werte für diese beiden Variablen sind für grössere Anwendungen meist zu klein gesetzt und sollten daher vergrössert werden.

Eine letzte Möglichkeit, wie das System beschleunigt werden kann ist der Query Cache. Dieser ist per default ausgeschaltet und kann bei Bedarf zugeschaltet werden.

Die entsprechenden Auswirkungen der eingestellten Werte können über den Befehl

```
SHOW GLOBAL STATUS;
```

abgegriffen werden.

Ein Tool um die ca. 330 Variablen und 310 Status Information auszuwerten bietet der MySQL Database Health Check, welcher hier http://www.fromdual.com/mysql-database-health-check kostenlose Hilfe anbietet.

Im weiteren ist die Bedeutung der verschiedenen Parameter und Status Inforation recht gut in der MySQL Dokumentation beschrieben unter: http://dev.mysql.com/doc/refman/5.5/en/mysqld-server.html

Alle weiteren Parameter sollten nur nach ausführlichem Benchmarking unter dem Kapitel Datenbank Fine-Tuning verändert werden.

Applikations-Tuning

Wenn die Datenbank einmal sauber konfiguriert ist, gilt es, sich dem Applikations-Tuning zuzuwenden. Hier hat man die Möglichkeit mit relativ wenig Aufwand sehr viel zu erreichen. Es sind hier durchaus Performancesteigerungen bis zu Faktor 10 für einzelne Abfragen zu erreichen.
Das Hauptaugenmerk ist hier auf fehlenden Indices zu richten.
Eine etwas hemdsärmlige und wenig systematische Methode um langsame Abfragen zu finden besteht darin auf der Konsole mehrmals hintereinander den Befehl:

```
SHOW PROCESSLIST;
```

aufzurufen. Wenn eine Abfrage dabei immer wieder oder die ganze Zeit auftaucht ist diese ein guter Kandidat um optimiert zu werden.
Ein etwas systematischerer Weg besteht darin, das sogenannte Slow Query Log einzuschalten. Der Parameter log_queries_not_using_indexes hilft zusätzlich schnelle Abfragen, welche trotzdem keinen Index verwenden aufzuspüren.
Über das Slow Query Log kann anschliessend mit dem Befehl:

```
mysqldumpslow -s t slow.log > slow.log.profile
```

ein Profil erstellt werden. Dieses hilft dabei, bei der Abarbeitung der langsamen Abfragen eine Priorisierung zu erreichen.
Sind die Abfragen isoliert, besteht der weitere Schritt darin, mit dem EXPLAIN Befehl den sogenannten Query Execution Plan (QEP) zu erstellen, welcher aufzeigt, wie MySQL die Abfrage genau abarbeitet und in welchem Schritt wie viel geschätzte Arbeit anfällt.
Das Lesen eines Query Execution Plans erfordert etwas Übung und es braucht einige Zeit bis man damit umgehen gelernt hat.

Eine weiter Möglichkeit die Applikation zu optimieren besteht darin, die entsprechende Tabellenstruktur von unnötigen Indices, Tabellen und Spalten zu befreien.
Das weg-archivieren von alten, nicht mehr benötigten Daten hilft ebenfalls die Datenbankabfragen zu beschleunigen.
Durch die Wahl der geeigneten kleinstmöglichen Datentypen kann die Tabelle unter Umständen signifikant Verkleinert werden, was abfragen im zweistelligen Prozentbereich beschleunigen kann.
Das utf8 Charakter-Set wir in MySQL intern als 3-byte Charakter-Set dargestellt. Wenn kein utf8 benötigt wird sollte daher aus Performance-Gründen drauf verzichtet werden. Utf8 kann Performanceeinbussen von 20-30 % nach sich ziehen.

InnoDB sortiert seine Daten aufgrund seines clustered Primary Keys automatisch nach dem Primary Key. Mit der Wahl eines geeigneten Primary Keys kann dabei die Sortierreihenfolge der Daten beeinflusst werden was bei entsprechenden Abfragen ebenfalls zu signifikanten Performancesteigerungen führen kann.

Tunning mittels Architektur und Design

Sind alle Möglichkeiten des Query Tunings ausgereizt müssen mit architektonischen mitteln Performance-Steigerungen erreicht werden.
Dies kann mittels der MySQL Replikation durch ein Read-Scale-Out erreicht werden.
Weitere Möglichkeiten der Performance-Steigerung sind der Einsatz von Caches oder die Nutzung von NoSQL-Architekturen mit MySQL um den SQL Overhead zu umgehen.

Bitte fügen Sie Ihre Kontaktadresse hinzu.

Kontaktadresse:

Oli Sennhauser
FromDual GmbH
Rebenweg, 6
CH-8610 Uster

Telefon: +41 79-830-09-33
Fax: +41 43-55-68204
E-Mail oli.sennhauser@fromdual.com
Internet: www.fromdual.com