Datenbank-Server: MySQL: Tuning vom feinsten

Problem:

Der Server ist zu langsam!

Und nachdem wir uns schon ausführlich mit dem Apache-Tuning befaßt haben und unsere Scripte genauestens nach Optimierungsmöglichkeiten durchsucht haben, gehen wir nun mal auf die Performance-Steigerung von MySQL ein.

Viele Probleme liegen z.T. in der Laufzeit von SQL-Queries begründet. Teilweise müßte man den Apache gar nicht optimieren, wenn MySQL seine Ergebnisse nur schneller ausliefern würde.

Dennoch ist es wichtig zu Begreifen, daß alles zusammen eine Einheit bildet, der Speicher sauber untereinander Verteilt werden muß und die besten Optimierungen nichts helfen, wenn die Scripte unperformant programmiert worden sind.

Dazu sollte man sich auch immer vor Augen halten, daß die Performance-Optimierung ein dynamischer Prozess ist. Gerade bei wachsenden Datenbanken und/oder Websites ist es unabdingbar regelmässig nach dem Rechten zu sehen.

Optimierung:

Die Frage bei MySQL ist immer: "*Wo fang ich an?*"

MySQL hat lediglich eine Konfigurationsdatei: `/etc/my.cnf` oder `/etc/mysql/my.cnf` (Debian).

Bitte Beachten: Alle Änderungen werden in der Sektion `[mysqld]` vorgenommen.

Wichtige Grundsätze bei der Optimierung:

1. Wir können nur ein laufendes System optimieren!

Warum? Weil wir nur auf Systeme die unter "natürlicher Last" stehen erkennen können wo es hängt.

Nur so können wir sehen, ob es überwiegend am Apache, am MySQL oder an den Scripten hängt.

2. Eine Optimierung ist ein dynamischer Prozess und braucht Geduld!

Am Besten immer nur kleine Änderungen machen, und wieder beobachten.

Der Erfolg von kleinen Schritten ist leichter nach zu vollziehen und auch schneller wieder Rückgängig zu machen.

3. Jeder Schritt wird Dokumentiert!

Das sollte sich von selbst verstehen.

4. Wenn geswappt wird, wird zuviel Speicher verbraucht!

Beim Swappen wird der Speicher auf die unperformante Festplatte ausgelagert. Dadurch entsteht zusätzliche IO-Waits und es kostet CPU-Performance.

Wenn hierbei auch noch Speicher ausgelagert wird, welcher eigentlich zum Cachen genutzt werden soll, beißt sich die Ratte in den Schwanz.

Wichtige Parameter:

Die wichtigsten Tuning-Parameter und Ihre Auswirkungen:

Parameter	Erklärung
`max_connections`	Wieviele Connections (und damit auch Threads) sind generell erlaubt? (Dieser Wert kann deutlich kleiner als die Apache-Einstellung `MaxClients` sein)
`table_cache`	Der Table-Cache reserviert Speicher für die häufiger gebrauchten Tabellen.
`query_cache_size`	In dem Speicherbereich werden die SQL-Statements mit einer Ergebnissliste gespeichert um bei einem n
`key_buffer_size`	Der Speicher, der für die referenziellen Keys reserviert wird. Dieser Wert gilt pro Connection.
`sort_buffer_size`	Der Speicher, der für die Sortierung reserviert wird.

Dieser Wert gilt pro Connection.

`read_buffer_size`
Der Speicher, der für das Lesen von Festplatte reserviert wird.
Dieser Wert gilt pro Connection.

Dazu kommen natürlich die Performance-Fresser schlechthin:

Die Logfiles: `log_slow_queries` (evtl. sogar als `log_long_format`) und das Replikations-Log: `log-bin` .

Alle unnötigen Festplattenzugriffe sollten unterbunden werden. Und wer nicht gerade die Slow-Queries bzgl. dem Tuning braucht und auch keine Datenbank-Replikation (`log-bin`) fährt, sollte diese abschalten.

Optimierungshilfen/Analyse

Es gibt inzwischen zwei sinnvolle Tools. Aber beachtet bitte meine Anmerkungen unter *Interpretation*.

a) Matthew Montgomery's Tuning Primer Script (bash-Script)

b) MySQLTuner (Perl-Script)

Beide Scripte lesen in erster Linie die MySQL-Laufzeit-Variablen aus und verrechnen diese. Dies Funktioniert, weil MySQL sowohl die Einstellungen aus der `my.cnf` als auch viele Statistik-Daten darin speichert.

Zum Auslesen wird aber ein Admin-Login gebraucht, welches ggf. abgefragt wird. (Beide Scripte erkennen ein evtl. installiertes Plesk und nutzen das von Plesk im Klartext hinterlegte Passwort.)

Das Tuning-Primer-Script kann zusätzlich die MySQL-Logfiles einlesen und auswerten.

Da beide Scripte ANSI-Zeichen für die kolorierte Ausgabe verwenden, empfiehlt sich das Ergebnis mit `less -R` zu lesen.

Um den dynamischen Prozess auch im Nachhinein mit verfolgen zu können, sollte jeweils das

Ergebnis gespeichert werden. Z.B.:

```
./tuning-primer.sh › tuning-primer.log.1
less -R tuning-primer.log.1
```

Tuning Primer Script

Das Tuning-Primer-Script ist vollständig als Shell-Script geschrieben.

Installation:

```
cd /usr/local/bin
wget http://www.day32.com/MySQL/tuning-primer.sh
chmod +x tuning-primer.sh
```

Der Aufruf startet über `./tuning-primer.sh`. Gerade hier sollte die Ausgabe aufgefangen werden, da die Optimierungstipps direkt unter der jeweiligen Sektion erscheinen.

Bei Fehlermeldungen á la `bc: command not found` muß das Programm `bc` installiert werden. (Z.B. `yast -i bc` oder `apt-get install bc`)

MySQLTuner

MySQLTuner ist im Groben vom Tuning-Primer-Script abgekupfert bzw. nach Perl portiert. Es hat aber einen komprimierteren und leichter verständlichen Output.

Etwas unschön: MySQLTuner sendet bei jedem Aufruf einen Request nach Hause um eine neue Version abzufragen. Diese Funktionalität ist ja ganz sinnvoll, sollte aber doch besser nur auf expliziten Wunsch des Admin ausgeführt werden.

Daher meine Empfehlung: Im Script am Ende nach `validate_tuner_version;` suchen und auskommentieren.

Installation und Aufruf:

```
cd /usr/local/bin
wget http://mysqltuner.com/mysqltuner.pl
chmod +x mysqltuner.pl
```

Ein Aufruf von `./mysqltuner.pl` fördert z.B. folgendes zu Tage:

```
 ››  MySQLTuner 0.9.0 - Major Hayden ‹major@mhtx.net›
 ››  Bug reports, feature requests, and downloads at http://mysqltuner.com/
 ››  Run with '--help' for additional options and output filtering

-------- General Statistics --------------------------------------------------
[OK] You have the latest version of MySQLTuner
[!!] Your MySQL version 4.0.18 is EOL software!  Upgrade soon!
[OK] Operating on 32-bit architecture with less than 2GB RAM

-------- Storage Engine Statistics -------------------------------------------
[--] Status: -Archive -BDB -Federated -InnoDB +ISAM -NDBCluster
[--] Data in MyISAM tables: 1G (Tables: 303)
[!!] ISAM is enabled but isn't being used

-------- Performance Metrics -------------------------------------------------
[--] Up for: 60d 19h 49m 58s (51M q [9.844 qps], 3M conn, TX: 11M, RX: 2B)
[--] Reads / Writes: 77% / 23%
[--] Total buffers: 2.1M per thread and 45.5M global
[OK] Maximum possible memory usage: 148.3M (7% of installed RAM)
[OK] Slow queries: 0% (4K/51M)
[!!] Highest connection usage: 100%  (50/50)
[!!] Key buffer size / total MyISAM indexes: 16.0M/851.0M
[!!] Key buffer hit rate: -19.2%
[OK] Query cache efficiency: 52.3%
[!!] Query cache prunes per day: 25394
[OK] Sorts requiring temporary tables: 0%
[OK] Temporary tables created on disk: 7%
[OK] Thread cache hit rate: 98%
[!!] Table cache hit rate: 0%
[OK] Open file limit used: 28%
[OK] Table locks acquired immediately: 99%

-------- Recommendations -----------------------------------------------------
General recommendations:
    Add skip-isam to MySQL configuration to disable ISAM
    Enable the slow query log to troubleshoot bad queries
    Reduce or eliminate persistent connections to reduce connection usage
    Increase table_cache gradually to avoid file descriptor limits
    Upgrade to MySQL 4.1+ to use concurrent MyISAM inserts
Variables to adjust:
    max_connections (› 50)
    wait_timeout (‹ 28800)
    interactive_timeout (‹ 28800)
    key_buffer_size (› 851.0M)
    query_cache_size (› 9M)
    table_cache (› 150)
```

Interpretation:

Bei der Interpretation sind folgende Bedingungen immer wichtig:

1.) Der MySQL-Server muß bereits eine Weile (ca. 2 Tage) gelaufen sein. Sonst liegen keine brauchbaren Statistikdaten vor.

2.) Nicht alle Tipps sind umsetzbar oder müssen umgesetzt werden!

Gerade zum letzten Punkt:

Beide Scripte missachten an einigen Stellen den real zur Verfügung stehenden Speicher.

Z.B: wenn man viele Tabellen hat, so wird einem empfohlen den `table_cache` zu erhöhen. Dumm nur, wenn man aufgrund von Speichermangel gar nicht mehr erhöhen kann. (Siehe Grundsatz zum Thema Swap.)

Weitere Überlegungen

Wie eingangs schon erwähnt, ist es wichtig die Optimierung auch mit anderen Prozessen abzustimmen. Dazu zählt u.a. daß man den Server beobachtet. Für eine Script-Optimierung lohnt sich die Laufzeit von gewissen Queries anzusehen. Zum einen mit normalen Profiling-Tools, zum anderen in einer Live-Umgebung anhand der Prozessliste. Sehr bequem geht dies übrigens mit `mytop` (Projektseite bei Freshmeat). Es stellt übersichtlich die aktuellen Daten von `SHOW STATUS` und `SHOW PROCESSLIST` dar.

Weiterführende Links:

- huschi.net: Server-Beobachtung
- huschi.net: Hochleistungs-Apache: Performance-Tuning
- huschi.net: Apache-Optimierung auf HTTP-Ebene
- iX 02/2003: MySQL aufpoliert

Eindeutige ID: #1295
huschi

2008-05-20 17:30