MySQL Architekturen für Oracle DBA's

Oli Sennhauser
FromDual
Uster

Schlüsselworte:

MySQL, Geschichte, Architektur, Storage Engine, MyISAM, InnoDB, MariaDB, Oracle

Einleitung

Die MySQL Datenbank ist in aller Munde und jetzt gehört sie sogar zur Oracle-Produktpalette. Mancher Oracle-DBA mag sich schon gefragt haben, was man alles mit so einer MySQL Datenbank machen kann, beziehungsweise hat sie bereits schon einmal aufgesetzt und damit herum gespielt.

MySQL ist für Oracle DBA's eine andere Welt. Es herrscht eine andere Philosophie und es braucht seine Zeit bis man sich umgewöhnt hat, wenn man das überhaupt will.

In diesem Vortrag beleuchten wir zuerst kurz die Entwicklungsgeschichte von MySQL, und gehen etwas auf die Philosophie und den Gedanken von Open Source Communities ein. Das hilft schon mal, ein paar Eigenschaft von MySQL zu verstehen.

Dann schauen wir uns die Architektur von MySQL und das Konzept der pluggable Storage Engines, sowie deren Vor- und Nachteile an.

Weiter geht es mit den MySQL Replikations-Architekturen (scale-out) und deren verschiedenen Spielarten bis hin zu MySQL HA (high availability) Architekturen (DRBD + Heartbeat) und (der hochperformanten und hoch verfügbaren) MySQL Cluster - Lösung.

Das ganze wird anschliessend mit ein paar Beispiel-Architekturen, wie sie in den letzten 4 Jahren bei Kunden implementiert wurden, veranschaulicht.

Geschichte von MySQL

Die Geschichte von MySQL beginnt im Jahr 1994 als Monty Widenius, David Axmark und Allan Larson das Projekt ins Leben rufen, welches später die Geschichte des Web stark beeinflussen sollte: Das M im LAMP-Stack oder schlicht MySQL.
Die Firma MySQL AB selbst wurde erst im Jahre 1998 gegründet. Im Jahr 2000 wird MySQL unter die Lizenz GPL gestellt und kann somit ab diesem Zeitpunkt als Open Source Produkt bezeichnet werden.
Im Frühjahr 2001 kommt die InnoDB Storage Engine hinzu welche das Konzept der Transaktionen beherrscht. Von der Firma Ericsson kauft MySQL im Herbst 2004 den NDB Cluster, welcher später unter dem Namen MySQL Cluster vermarktet wird. MySQL besitzt jetzt eine hoch verfügbare und skalierbare Cluster-Lösung.
Ein Jahr später, im Oktober 2005 kauft die Firma Oracle die kleine finnische Firma Innobase OY, welche die InnoDB Storage Engine entwickelt. Dieser Tag wurde in MySQL-Kreisen auch als „InnoDB Black Friday“ bezeichnet.

Als Reaktion hierauf beginnt MySQL 2006 mit der Entwicklung der Falcon Storage Engine. Im selben Jahr versuchte Oracle anscheinend auch die Firma MySQL selber zu kaufen nachdem sie sich die Firma Sleepycat einverleibt hat, welche für die Entwicklung der Berkley DB (BDB) zuständig ist. Ebenfalls im Jahr 2006 verlässt das MySQL Benchmark Team MySQL und gründet eine eigene Firma, welche heute die XtraDB Storage Engine entwickelt, ein Branch der InnoDB Storage Engine. Ende des Jahres wird der IPO (Börsengang) für das Jahr 2008 angekündigt.
Im Januar 2008 der Paukenschlag: Sun Microsystems kauft für 1 Mia USD die Firma MySQL (geschätzter Wert bei Börsengang ca. 350 Mio USD). Sun wiederum wird im April 2009 von der Firma Oracle übernommen in deren Hände MySQL damit übergeht. MySQL und InnoDB sind ab jetzt vereint.

Open Source

Mit MySQL hat das Schlagwort Open Source Einzug in die Datenbankwelt gehalten. Aber was hat es mit Open Source auf sich?
Open Source kann als Philosophie bezeichnet werden, welche teilweise sogar religiöse Ausmasse annehmen kann. Open Source Software wäre somit die praktische Umsetzung dieser Philosophie.
Die Idee dahinter ist, dass der Source Code einer Applikation offen gelegt wird und frei verfügbar ist. Dies erlaubt Einsicht in die implementierte Logik. Die wichtigsten Punkte hierbei aber sind die Möglichkeit aus dem Code anderer zu lernen und Fehler oder problematische Passagen darin zu entdecken. Dies führt, zumindest theoretisch, zu besseren Produkten und einer besseren Verteilung des Know-Hows.
Neben dem Offenlegen des Source Codes beinhaltet Open Source aber auch noch die Regelung von Rechten (Lizenzen), wie mit dem offen gelegten Code umgegangen werden darf. Welche Lizenz sich als Open Source Lizenz bezeichnen darf wird von der Open Source Initiative definiert [1].
Open Source Software ist sehr oft nicht in der Hand von Firmen sondern in der Hand von einzelnen Personen, kleinen Gruppen oder gemeinnützigen Organisationen. Es gilt das Prinzip Jekami – Jeder kann mitmachen.
Je nach gewählter Lizenz lässt sich mehr oder weniger gut Profit aus einer Software schlagen. Dies führt dann öfters zu Interessenkonflikten der verschiedenen involvierten Parteien. Eine Möglichkeit, die hierbei manchmal gewählt wird, ist die der dualen Lizenzierung.
MySQL hat ein solches duales Lizenzierungsmodell gewählt. Im Unterschiede zu den meisten Open Source Projekten ist der Code aber in fester Hand der Firma MySQL und Änderungen aus der Community fliessen nicht so ohne weiteres zurück in den MySQL Code. Dies hat unter anderem mit dem dualen Lizenzmodell zu tun, wird aber nach aussen kaschiert mit: „Der Code genügt nicht unseren Ansprüchen“.

Konzepte hinter MySQL

Um die Eigenschaften und das Verhalten von MySQL zu verstehen ist es hilfreich einige Grundkonzepte zu kennen:
MySQL soll einfach zu benutzen sein. In 15 Minuten muss es laufen. Es wurde auch immer wieder der Spruch herum geboten: „It just works“ (es funktioniert gerade eben so). Um eine hohe Performance, eine einfache Benutzung und eine kurze „Time-to-market“ zu erreichen wird oft auch der Ansatz „relaxation of constraints“ (Aufweichung der Anforderung) verwendet.
Unter Berücksichtigungen dieser Aspekte dürfte es für den Oracle DBA verständlicher werden, warum er sich nach dem felsenfesten Oracle bei MySQL auf etwas sumpfigem Gelände fühlt...

MySQL Architektur

Aus historischen Gründen hat sich MySQL von einer „monolithischen“ Architektur hin zu einer mehr oder weniger modularen Architektur entwickelt. Während wir bei den meisten anderen RDBMS nur eine Art von Tabellentyp kennen, gibt es bei MySQL viele verschieden Tabellentypen.
MySQL besteht intern aus einer zweigeteilten Architektur. Der eine Teil ist Tabellentyp unabhängig und ähnlich aufgebaut wie bei anderen Datenbanken. Der zweite Teil, die Storage Engines, ist für die spezifische Eigenschaften des jeweiligen Tabellentyps verantwortlich. Die Tabellen eines bestimmten Typs werden physikalisch von einer Storage Engine verwaltet.
Im Storage Engine unabhängigen Teil werden die ankommenden Verbindungen von der Applikation entgegengenommen und ggf. aus dem sogenannten Query Cache bedient. Ist eine Abfrage nicht im Query Cache enthalten wird sie geparst und ein Execution Plan erstellt. Anhand dieses Execution Plan's werden dann die Daten aus den verschiedenen Tabellen zusammengetragen und zurück an die Applikation geschickt. Soweit entspricht diese Architektur mehr oder weniger derjenigen der gängigen RDBMS.
Der Grosse Unterschied liegt jetzt aber in den Tabellen welche von den einzelnen Storage Engines verwaltet werden. Je nach gewählter Storage Engine haben die Tabellen unterschiedliche Charakteristiken. Die einen Storage Engines sind mehr geeignet für kurze Transaktionen. Die anderen mehr für Reports und Full-Table-Scans und dritte Tabellentypen mehr für Spalten orientierte Abfragen oder Hochverfügbarkeit und Primary Key Zugriffe.

Pluggable Storage Engines

Einige der Eigenschaften einer Datenbank sind im Storage Engine unabhängigen Teil von MySQL untergebracht. Dies sind unter anderem:

- Stored Procedures / Stored Functions
- Views
- Connection Handling
- Query Cache
- Parsen der Abfragen
- Optimieren der Abfragen

Andere Eigenschaften wiederum sind sehr stark Storage Engine abhängig. Diese sind z.B.

- Transaktionen
- Volltext Suche
- GIS Funktionalität
- Referentielle Integrität
- Hochverfügbarkeits-Eigenschaften
- Datenintegrität nach einem Crash

Je nach Anforderungen und gewünschten Eigenschaften kann man die entsprechende Storage Engine für seine Tabelle wählen. Tabellentype können gemischt werden, sogar innerhalb ein und des selben Statements. Dies zieht aber unter Umständen operative Nachteile nach sich.

Die wichtigsten Storage Engines

Die wichtigsten Storage Engines sind:

- MyISAM / Aria: Waren lange Zeit die voreingestellte Storage Engines. MyISAM ist nicht Crash-sicher. Ihre Nachfolgerin Aria hingegen ist Crash-sicher. Diese Storage Engines können KEINE Transaktionen, beinhalten dafür aber GIS Funktionalität und eine Volltext-Indexierungsmöglichkeit und sind sehr performant bei Full-Table-Scans.
- InnoDB / XtraDB / PBXT: Diese Storage Engines sind geeignet, wenn ein transaktionsorientiertes System benötigt wird. Sie unterstützen zudem referentielle Integrität. PBXT ist besonders für grosse Objekte geeignet.
- NDB: Diese Storage Engine ist auch unter dem Namen MySQL Cluster bekannt. Sie bietet einen hoch verfügbaren und hoch performanten Tabellentyp an der vor allem für hohe Konkurrenz ausgelegt ist. Sie beherrscht ebenfalls Transaktionen.
- Federated-X: Mit dieser Storage Engine erhält man eine ähnliche Funktionalität wie mit Oracle Database-Links.
- InfiniDB/Inforbright: Dies sind Spalten orientierte Storage Engines welche v.a. Bei bestimmten Auswertungen und Reports signifikante Performanzvorteile gegenüber herkömmlicher Storage Engines aufweisen.

Weiter Unterschiede zwischen Oracle und MySQL

Einige weitere Eigenschaften welche zwischen MySQL und Oracle unterschiedlich sind beleuchten wir hier kurz:
Im Unterschied zu Oracle und bedingt durch seine zweischichtige Architektur kennt MySQL zwei verschieden Arten von „Transaktions“-Logfiles: Der eine Typ von Transaktions-Logfiles ist Storage Engine abhängig. Sie existieren nicht für alle Storage Engines. Bei der InnoDB Storage Engine z.B. heissen diese Logfiles ib_logfile.
Der zweite Typ von „Transaktions“-Logfiles ist NICHT Storage Engine abhängig und wird Binary-Log genannt. Diese Logfiles beinhalten DML Statements aller Storage Engines. Die beiden Typen von Logfiles sind nicht direkt kompatibel miteinander. Anders als bei Oracle wo ein Redo-Log automatisch zu einen Archive-Log wird kann bei MySQL ein ib_logfile nicht in ein Binary-Log umgewandelt werden.
Für Oracle DBA's gilt: Ein MySQL Binary-Log hat Ähnlichkeit mit einem Archive-Log und ein ib_logfile ist vergleichbar mit einem Redo-Log.
Während mit Oracle der Aufbau einer Verbindung zur Datenbank sehr teuer ist, können mit MySQL Verbindungen sehr schnell aufgebaut werden. Dies hat zur Folge, dass bei MySQL-Anwendungen sehr oft auf einen Connection-Pool verzichtet wird. Bei Oracle würde das bei einer Webanwendung zu schlechten Antwortzeiten führen.
Währen eine Oracle Instanz aus mehreren Prozessen besteht (pmon, smon, arch, etc.), welche über Shared Memory kommunizieren, verwendet MySQL den Threading-Ansatz: Ein Prozess besteht aus mehreren Threads welche den Speicherbereich gemeinsam nutzen.

MySQL Replikations-Architekturen (scale-out)

Es gibt hauptsächlich 2 Gründe warum MySQL so bekannt geworden ist. Der erste ist, dass MySQL ein Teil des sogenannten LAMP Stacks (Linux, Apache, MySQL, PHP) wurde. Der zweite Grund ist seine einfach aufzusetzende Replikation. Mit dieser Replikation kann man verschiedene Bedürfnisse abdecken und Lastmuster bedienen. In Web-Applikation haben wir sehr oft eine hohes Lese- zu Schreibverhältnis. Dies kann ideal mit einer MySQL scale-out Architektur gelöst werden. 1 Master wird zum schreiben genutzt und das Lesen wird auf mehrere Slaves verteilt.

Im Unterschied zu den grossen Datenbankanwendungen wie Oracle, die den scale-up Ansatz propagieren (grosse Server mit vielen CPU's) verfolgte MySQL den scale-out Ansatz (viele kleine Standardserver und ein Verteilen der Last). Dies hat verschiedene Gründe. Einer davon ist, dass MySQL lange Zeit nicht in der Lage war, auf Mutli-Core Maschinen zu skalieren. Ein anderer, dass grosse Server in Bezug zu Ihrer Leistung überproportional teuer sind und somit für kleine Web-Firmen gar nicht erst in Frage kommen.

MySQL HA (high availability) Architekturen

Wenn nicht Skalierbarkeit das Hauptaugenmerk einer Installation ist, sondern Hochverfügbarkeit, kann mit MySQL ein sogenannter aktiv/passiv Failover-Cluster gebaut werden. Dieses Konzept ist auch von anderen Datenbank HA-Clustern bekannt: Eine Datenbank ist auf einem Server aktiv und schreibt ihre Daten auf ihr Disk-System (SAN) oder über DRBD zusätzlich auf das Disksystem eines zweiten Rechners.
Treten Probleme auf diesem aktiven Rechner auf, wird der Datenbankprozess durch die Cluster-Software auf den zweiten Server gezügelt nachdem das Disksystem umgehängt wurde. Die Datenbank läuft anschliessend auf dem zweiten Rechner weiter.
Wenn es das Budget nicht zulässt, kann anstelle eines SAN auch DRBD verwendet werden, welches die Daten vom aktiven auf das passive System synchron überträgt.

MySQL Cluster

Als weiter Lösung bittet MySQL auch einen hoch performanten und hoch verfügbaren Cluster an. Dieser Cluster glänzt dadurch, dass er bei hoher Parallelität und hoher Schreiblast mithalten kann. Zudem weist diese Shared-Nothing-Architektur im Gegensatz z. B. zum Oracle RAC keinen Single-Point-of-Failure (SPOF) auf. Ein Nachteil dieses Systems ist, dass die Antwortzeiten nicht so gut wie bei einer normalen MySQL Installation sind. Joins können beim MySQL Cluster ebenfalls sehr sehr langsam werden.
Wegen seiner spezifischen Charakteristik kann MySQL Cluster sehr oft NICHT einfach als Ersatz für eine andere MySQL Storage Engine verwendet werden. Applikationen werden typischer Weise auf MySQL Cluster zugeschnitten.

Literatur

[1] Wikipedia: Open Source Software: http://en.wikipedia.org/wiki/Open-source_software

Kontaktadresse:

Oli Sennhauser
FromDual
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