Bereitstellung von Web-2.0-Anwendungen mit MySQL

Ralf Gebhardt
Oracle Deutschland B.V. & Co. KG
Stuttgart

Schlüsselworte:

DB, MySQL, Cluster, Cache, Web-2.0, Architektur, Replikation, Skalierung, LAMP, Open Source

Einleitung

MySQL wird, nicht zuletzt durch den Erfolg des LAMP-Stacks und der einfachen Nutzung des MySQL Servers, meist für den Betrieb von hochfrequentierten Webseiten eingesetzt. Dies gilt sowohl für Community-Plattformen, kommerziell orientierte Web-Plattformen als auch in internen Unternehmensanwendungen für den Austausch von geschäftskritischen Daten und in besonderem Maße für für Web-2.0-Unternehmen. Dieser Vortrag beschreibt kurz das Thema Web 2.0 und die verschiedenen technischen Elemente einer Web-2.0-Umgebung. Anschließend betrachten wir, wie mit MySQL Replikation eine Lese-Skalierung für Web-2.0-Anwendungen aufgebaut werden kann und wie MySQL Cluster zur Schreib-Skalierung beitragen kann.

Die Definition von Web-2.0-Anwendungen

Tim O'Reilly & Dale Dougherty haben 2004 Web-2.0 so beschrieben:
„... a collection of technologies and companies who leverage users and developers in a collaborative manner in order to rapidly create data and develop applications with a high level of integration across platforms, data sources and other web-enabled services."

Durch den freien Zugriff bietet das Web die optimale Plattform für den Aufbau von Anwendungen und Services nach der Definition von Web-2.0. Zudem ist die Art und Weise, wie das Internet von seinen Nutzern verwendet wird, eine gute Basis, um Daten zu erhalten, durch die Web-2.0-Anwendungen erfolgreich implementiert werden können. Auch die offene Art und Weise der Entwicklung solcher Anwendungen – nicht zuletzt durch die breite Nutzung von Open Source Projekten - ist eine wichtige Voraussetzung, da Dienste auf Basis schon vorhandener Quellen und Daten erstellt werden können.

Erst die Daten einer Web-2.0-Anwendung machen diese erfolgreich. Sie und ihre richtige Verwendung stellen den Wert der Anwendung dar. Diese Daten werden aber nicht von einer zentralen Stelle bzw. vom Betreiber eines Dienstes alleine generiert. Den wesentlichen Anteil der Daten liefert der Nutzer selbst; oft bewusst, teilweise unbewusst.
Die bewusste Generierung von Daten zeigt sich am besten an Wissensdatenbanken, deren Inhalt durch die Anwender selbst generiert werden. Ein bekanntes Beispiel hierfür stellt Wikipedia dar. Eine Web-2.0-Anwendung lebt also von der Mitwirkung der Nutzer.
Eine eher unbewusste Lieferung von Daten kann z.B. durch offen zugängliche Profile des Nutzers selbst stattfinden.

Die Zusammenführung von Daten reduziert den Overhead, Daten für jeden Dienst neu generieren zu müssen - ein weiterer Erfolg von Web-2.0-Anwendungen.

Ein Merkmal von Web-2.0-Anwendungen ist, dass diese keinem festen Entwicklungszyklus unterliegen. Man spricht auch davon, dass eine Web-2.0-Anwendung den BETA-Status nie verlässt.

Dienste wie Blogs, Feeds, Social Networking, Wikis und deren Kombination sind häufig anzutreffende Anwendungen, die von der Definition her Web-2.0-Anwendungen sind.

Web-2.0-Architekturen

Web-2.0-Anwendungen nutzen häufig einen Open Source Stack, in diesem Fall gerne als Web-2.0-Stack bezeichnet. Dieser Stack enthält Betriebssystem, Web/Anwendungsserver, die Datenbank und die Programmiersprache, hier oft eine Skript-Sprache. Die genutzte Software variiert, durch offene Schnittstellen sind verschiedene Kombinationen denkbar.

Dieser Backend-Stack wird oft durch einen Caching-Layer erweitert, um Webinhalte zeitnah darstellen zu können.

Die Schnittstelle zum Anwender ist immer der Browser. Die Herausforderung besteht darin, dem Nutzer unabhängig vom genutzten Browser eine einfach zu nutzende, schnelle und personalisierte Plattform anzubieten.

Die erste Ebene im Backend aus Sicht des Nutzers sind Netzwerk und Elemente zur Lastverteilung. Die Lastverteilung stellt ein wichtiges Element für den Erfolg eines Dienstes dar, da flexibel auf die Last der Nutzer reagiert werden können muss. Für Web-2.0-Anwendungen kann nur selten die aufkommende Last vorherbestimmt werden, meist existieren nur Schätzungen, Tages- oder Wochen-Trends.

Application- und Web-Server, zusammen mit häufig genutzten RAD- und Entwicklungsumgebungen stellen die zweite Ebene dar. Hier muss eine schnelle und parallel zum Betrieb durchführbare Entwicklung ermöglicht werden. Zudem sollte hier die Anwendungslogik positioniert sein. Die richtige Entwicklung der Anwendung ist maßgeblich, um auch hier skalierbar zu bleiben.

Das Caching als dritte Ebene ist vor allem dann wichtig, wenn die Web-2.0-Anwendung sehr leselastig ist – was oft der Fall sein dürfte, zumindest für Teile eines Dienstes. Die Caching-Ebene hat das Ziel, Last vom Datenbankserver zu nehmen für Daten, die sehr oft gelesen und - dies ist wichtig - nicht oft geändert werden. Dies soll nicht bedeuten, dass das Caching nicht auch über einen Datenbankserver implementiert werden kann. MySQL bietet z.B. mit der Memory- und der MySQL Cluster-Storage-Engine Varianten an, die Daten im RAM halten können.
Caching findet immer über RAM-Systeme statt und reduziert so zu stark das Festplatten-IO, das Grundproblem für langsame Anwendungen.

MySQL als Datenbankschicht für Web-2.0-Anwendungen

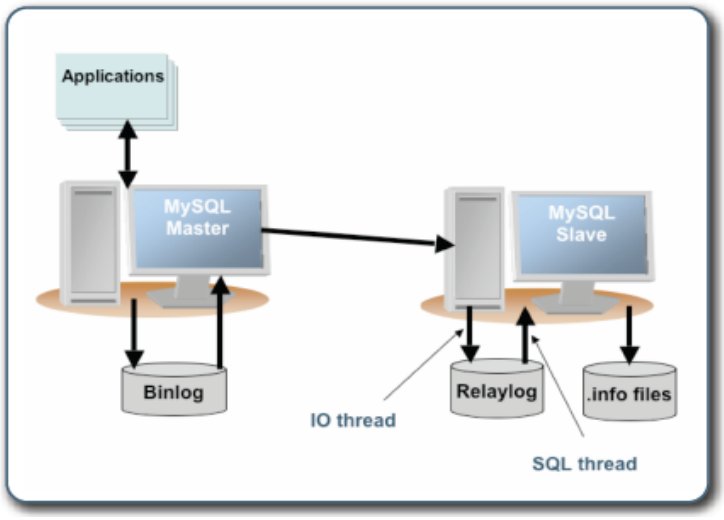
MySQL Server und MySQL Cluster werden oft als Datenbanksystem in Web-2.0-Anwendungen genutzt. Durch die Möglichkeit der Nutzung verschiedener Storage-Engines, - die bekanntesten sind InnoDB und MyISAM-, können mit einem Datenbankserver verschiedenste Anforderungen der Web-2.0-Anwendung erfüllt werden. Die Nutzung der jeweiligen Storage Engine wird auf Tabellenebene definiert. Der Anwendung selbst ist dabei nicht bekannt, welche Engine für die gerade angefragte Tabelle genutzt wird. Dennoch ist die Wahl und Kombination der Storage-Engine entscheidend für Qualität und Performance der Anwendung.

MyISAM wird oft für Tabellen genutzt, die hauptsächlich lesend verwendet werden oder dann, wenn Volltextsuche auf Basis der Datenbank implementiert werden soll. InnoDB wird verwendet, wenn vermehrt Schreibzugriffe erfolgen, Fremdschlüssel für das Datenbank-Design verwendet werden müssen oder Transaktionssicherheit nach dem ACID-Prinzip gebraucht wird.

MySQL Cluster kommt im Bereich von Web-2.0-Anwendungen immer dann zum Einsatz, wenn z.B. Schreibskalierung gefordert wird, der zu Verfügung gestellte Dienst hohe Anforderungen an die Verfügbarkeit hat und administrative Tätigkeiten ohne Unterbrechung des Dienstes erfolgen müssen. Gerne genutzt wird MySQL Cluster für Session-Management. Session-Management ist die zentrale Funktion, wenn einem Benutzer z.B. profil-abhängige Inhalte dargestellt werden sollen.
Da die Datenhaltung von MySQL Cluster auch rein RAM-basiert erfolgen kann, ist auch die Verwendung als Cache interessant - speziell dann, wenn auch vermehrt Schreibzugriffe erfolgen. Hier erzeugen externe Caching Systeme enormen Overhead durch die ständige Aktualisierung der Daten aus der Datenbank.
Aus Sicht MySQL Server stellt sich MySQL Cluster nur als eine weitere Storage-Engine dar. Die Anwendung greift also weiterhin auf eine Tabelle in MySQL Server zu.

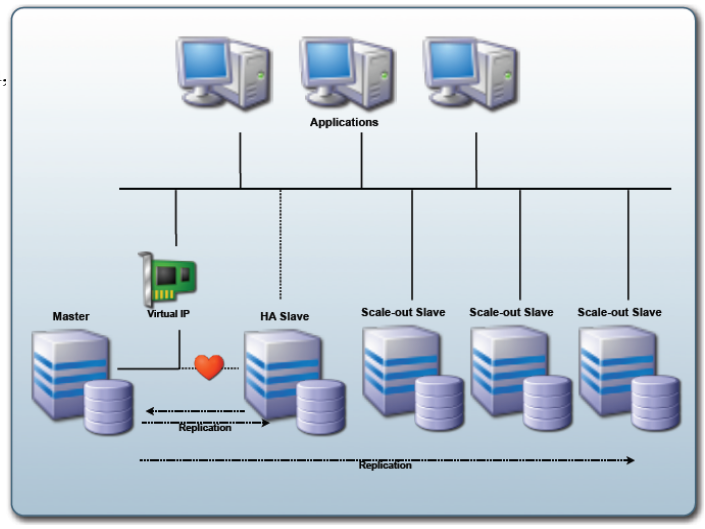
MySQL Replikation

Wie schon erwähnt besteht für viele Web-2.0-Anwendungen die Herausforderung darin, zur Laufzeit skalieren zu können. Oft genügt aber die Lese-Skalierung, da die Schreiblast sehr gering bleibt.
Für die Skalierung auf Datenbankebene wird hier die in MySQL Server integrierte Replikation verwendet.
Wie funktioniert nun die MySQL Replikation? Definiert werden Master und Slave. Der Master dient als Quelle, die Slaves als Replikat. Werden an den Master SQL-Befehle gesendet, die Daten verändern, so werden diese Befehle zusätzlich in das sogenannte Binlog geschrieben. Slaves erhalten nur eine Information, dass sich Änderungen im Binlog ergeben haben. Der Slave wird dann im ersten Schritt neue Einträge in sein eigenes Relaylog übertragen, ein weiterer Thread wird dann diese Befehle auf dem Slave ausführen.
Dieser Vorgang wird asynchron durchgeführt. Dadurch sind Laufzeiten auf Master und Slaves voneinander unabhängig.



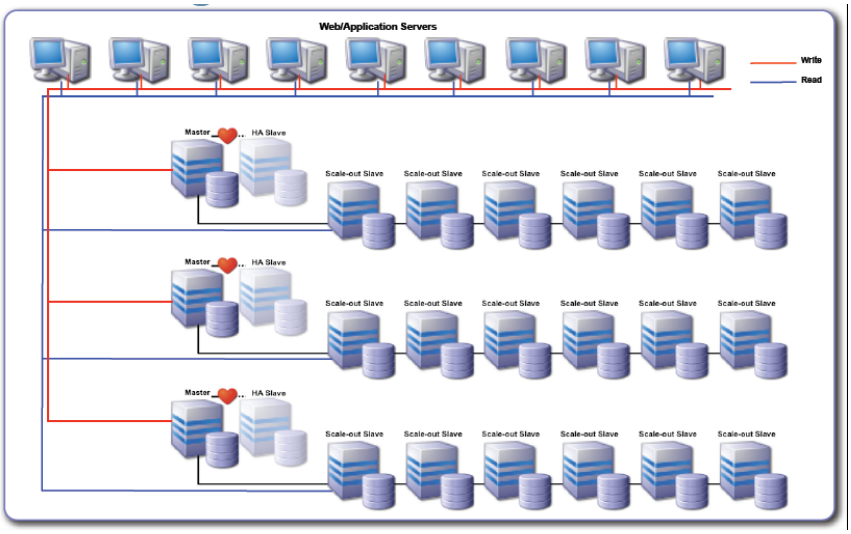
Auf der Basis der 1:n Replikation können verschiedenste Topologien aufgebaut werden. Eine n:1 Replikation ist nicht möglich.

Für eine Lese-Skalierung werden oft mehrere Slaves betrieben. Schreibzugriffe werden an den Master gesendet. Kritische Lesezugriffe, also Zugriffe, für die schon die geringste zeitliche Verzögerung durch die asynchrone Replikation ein Problem wären, werden ebenfalls vom Master bedient. Slaves bedienen die übrigen Lesezugriffe, welche oft 90% der Zugriffe einer Web-2.0-Anwendung ausmachen. Durch Hinzufügen von weiteren Slaves kann auf höhere Leselast reagiert werden.



Erweiterte Architekturen verwenden als Zwischenschicht eine Art Relay-Master. Dies sind Server, die auf der einen Seite als Slave für den zentralen Master dienen, gleichzeitig aber auch als Master für eine Gruppe von Slaves dienen. Diese Architektur wird oft auch zur Implementierung von Sharding verwendet. Hier bedient jeder Serverzweig (Relay-Master + Slaves) eigene Daten, der schreibende Zugriff findet auf dem Relay-Master statt. Daten, die von allen Shards benötigt werden, werden vom zentralen Master zum Relay-Master repliziert, von diesem dann zu den reinen Lese-Slaves.

In bestimmten Bereichen einer Web-2.0-Anwendung kann eine Schreib-Skalierung nötig werden, da die Schreiblast von einem MySQL Server nicht mehr bewältigt werden kann. Sharding kann hier eine Lösung sein, da sich die schreibenden Zugriffe auf die verschiedenen Master der einzelnen Zweige verteilen.

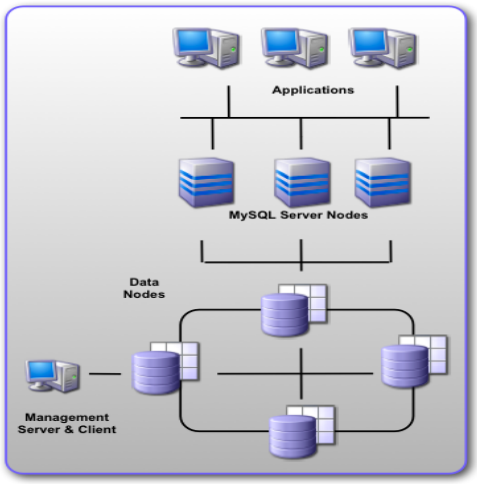


MySQL Cluster

Ist Sharding keine Option, z.B. da die Anwendung eine Verteilung der Zugriffe auf verschiedene Shards nicht zulässt, könnte MySQL Cluster eine Alternative sein. MySQL Cluster ist speziell für Session-Management eine gute Wahl, wenn die Session-Daten als kritisch betrachtet werden und während ihrer Laufzeit nicht verloren gehen sollen.

Auch als Cache eignet sich MySQL Cluster, da die Datenhaltung direkt im RAM stattfinden kann. Da es sich dennoch um ein Datenbanksystem handelt, bestehen auch Vorteile im Vergleich zu von Datenbanken getrennten Caches, wenn des öfteren Schreibzugriffe auf die Daten erfolgen. Hier wird für externe Cache-Lösungen der Overhead für das Erneuern des Caches über die Datenbank zu groß.

Die Architektur von MySQL Cluster erlaubt es, dass mehrere MySQL Server - in diesem Fall SQL Node genannt - lesend und schreibend auf die gleichen Daten zugreifen können und somit ein Active-Active Cluster aufgebaut werden kann. Da es sich hier um einen Applikations-Cluster handelt, wird keine spezielle Hardware benötigt.



MySQL Cluster bietet zudem Funktionalitäten an, um ihn ohne Ausfall des Clusters um neue SQL Nodes zu erweitern. Auch Data Nodes, die Prozesse, die die Datenhaltung übernehmen, können ohne Stop des Clusters hinzugefügt werden. Durch das Hinzufügen neuer Data Nodes oder letztendlich neuer Server und deren RAM kann so der Cache erweitert werden.

Kontaktadresse:

Name
Oracle Deutschland B.V. & Ko. KG
Zettachring, 10a
D-70567 Stuttgart

Telefon:	+49 (0) 160-98928493
E-Mail	ralf.gebhardt@oracle.com
Internet:	www.mysql.de
	www.oracle.de