Im Vergleich: Hochverfügbarkeitslösungen für die MySQL®-Datenbank

Ralf Gebhardt
SkySQL Ab
Vävarsvägen 11

02630 Esbo

Finland

Schlüsselworte
MySQL, MariaDB, Hochverfügbarkeit, Replikation, MHA, DRBD, HA, Cluster, Galera, Open Source, Tungsten, Continuent

Einleitung

In diesem Vortrag werden Produkte und Lösungsansätze vorgestellt und verglichen, welche in MySQL-Umgebungen zum Einsatz kommen, um für die Datenbank eine höhere Verfügbarkeit gewährleisten zu können.
Die meisten der vorgestellten Lösungen können sowohl mit MySQL® Server als auch mit MariaDB® verwendet werden.
Zum Vergleich der verschiedenen Lösungsansätze werden Themen wie die Art der Replikation, synchron oder asynchron, zu verwendende Storage-Systeme, Einschränkungen bezüglich der MySQL-Funktionalität und Art der Hochverfügbarkeitslösung (HA) betrachtet.
Die im Vergleich enthaltenen Lösungen sind die MySQL-Replikation, MHA®, Continuent Tungsten® Replicator und Continuent Tungsten® Enterprise, DRBD®, Galera und MySQL® Cluster.

Im Vergleich: Hochverfügbarkeitslösungen für die MySQL®-Datenbank

Für den Vergleich der verschiedenen Hochverfügbarkeitslösungen sollten einige der zu betrachtenden Merkmale kurz beschrieben werden.

Asynchron vs. Synchron

Um ein redundantes Datenbank-System aufzubauen, müssen die Daten auf einen weiteren Server gespeichert werden. Dies findet entweder asynchron oder synchron statt.
Das wesentliche Merkmal für den asynchronen Transfer der Daten ist, dass der sendende Prozess nicht auf eine Bestätigung wartet, ob die gesendeten Daten auf dem zweiten System genutzt werden können. Der Vorteil dieser Variante ist, dass das Zielsystem nicht die Performance des Quell-Systems beeinflusst. Man erkauft sich diesen Vorteil mit dem Risiko, dass die Daten der beiden Systeme inkonsistent sein können, da z.B. bei einem Fehler im Zielsystem kein Rollback auf dem Quell-System erfolgen würde. Hier müsste manuell eingegriffen werden, um die Konsistenz wieder herzustellen.

Kann dieses Risiko nicht in Kauf genommen werden, muss ein synchroner Daten-Transfer verwendet werden. In diesem Fall garantiert eine Rückmeldung des Zielsystems, dass die Daten erfolgreich empfangen werden konnten. Das Zielsystem kann, abhängig von der Rückmeldung, entscheiden, ob die Aktion im Quell-System bestätigt werden oder rückgängig gemacht werden soll.
Bei Nutzung des synchronen Datentransfers hängt die Gesamt-Performance allerdings von allen Systemen einschließlich des Netzwerk-Durchsatzes ab.
Der synchrone Daten-Transfer kommt daher in WAN-Verbindungen kaum in Frage.

Switchover, Failover und Failback

Wir sprechen von „Switchover“, wenn das Umschalten zu einem redundanten Standby-System manuell erfolgt. Im Unterschied dazu wird bei automatischer Umschaltung zum Standby-System von „Failover“ gesprochen.
In manchen Fällen ist ein Failover allerdings nicht möglich, oder nicht nötig. Dies hängt vom Gesamt-Konzept und den zu beachtenden SLAs der Einzel-Systeme ab. In Umgebungen mit MySQL-Datenbank kommt es z.B. oft vor, dass ein SLA für lesenden Zugriff auf Daten höher ist als ein SLA für den Server, auf den Daten geschrieben werden. Dies bedeutet, dass für den schreibenden Zugriff ein Switchover genügen kann, Lese-Zugriffe über mehrere Lese-Server (Slaves) aber immer gewährleistet sein muss.
Oft unterschätzt wird im Bezug auf eine Hochverfügbarkeitslösung die Wiederherstellung des ausgefallenen Systems, auch als „Failback“ bekannt. Oft ist der einzige Grund, dass ein Failback so bald als möglich durchgeführt sein muss, der Verwendung von schlechter ausgestatteten Systemen im Standby-System. Daher stellt sich die Frage, ob eher in eine gut Failback-Umgebung oder in die Ausstattung des Standby-Systems investiert werden sollte.

Hochverfügbarkeitslösungen für MySQL® Server

Die verschiedenen Lösungen, um MySQL-Datenbanken hochverfügbar zu machen, können unterteilt werden in Lösungen basierend auf:

- hauptsächlich Funktionalitäten von MySQL® Server
- Storage-Systemen
- Patches von MySQL® Server mit spezieller HA-Funktionalität
- HA spezialisierten Storage-Engines
- HA Middleware

Wir werden im Folgenden einige diese Lösungen betrachten.

HA-Lösungen basierend auf MySQL Server Funktionalitäten

MySQL® Server bringt schon eigene Funktionalitäten mit, um Daten zu anderen Datenbank-Systemen zu transferieren – die MySQL Replikation.

MySQL-Replikation ist ein bekannte Funktion, um auf einfache Weise Lese-Skalierbarkeit oder, mit komplexeren Strukturen, skalierbare Umgebungen auf Basis von Sharding-Architekturen zu implementieren.

Da auch hier letzten Endes Daten auf ein Zweit-System übertragen werden, kann das Zweit-System auch als Standby-System angesehen werden. Wir wollen daher kurz die MySQL-Replikation selbst, und die erweiterte Semi-Synchone Replikation betrachten. Die Lösung MHA erweitert die Funktionalitäten durch die Verwendung einiger Skripte.

MySQL-Replikation

Wie erwähnt ist die MySQL-Replikation an sich keine Verfügbarkeitslösung, da hier aber sehr einfach eine Daten-Redundanz auf einem zweiten Server erzeugt werden kann, wird sie dennoch gerne als Einstiegs-Lösung in dieses Thema verwendet. Und nicht selten bleibt man bei diesem Ansatz.

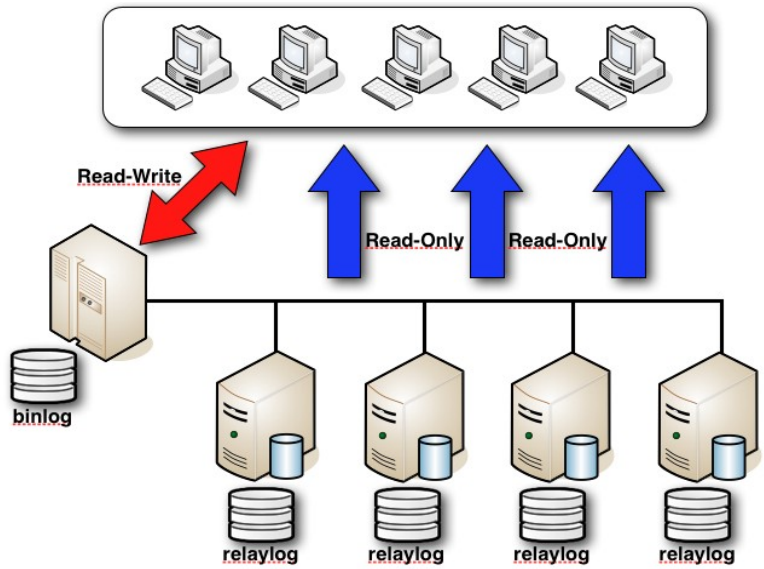

Abb. 1: MySQL-Replikation

Kurz beschrieben, werden Änderungen auf dem “Master” zusätzlich in einer Log-Datei gespeichert, diese Änderungen dann zu einem – oder mehreren – weiteren Datenbank-System(en) (Slaves) übertragen, genauer gesagt wiederum in eine Log-Datei eingetragen, um dann in der Datenbank aktualisiert zu werden. Hierbei wird der asynchrone Ansatz verwendet. Das bedeutet, die Performance des “Masters” ist nicht abhängig von den “Slaves”. Sollte es zum Ausfall des “Masters” kommen, könnte nun einer der “Slaves” die Master-Rolle übernehmen, es existiert allerdings kein Automatismus hierfür.

Merkmale dieser Lösung:

- Asynchron
- Local Storage
- Hot-Standby
- Kurze Umschaltzeit
- Datenverlust möglich
- Unterstützt gängige Storage-Engines
- Single-Threaded Replikation
- Failback komplex

- Open Source

Semi-Synchrone Replikation

Die Semi-Synchrone Replikation soll das Risiko des Verlustes von Daten reduzieren, ohne die Performance des Masters wesentlich zu beeinflussen. Diese Zusatzfunktion hat daher den Fokus, die MySQL-Replikation zur Implementierung einer Verfügbarkeitslösung zu nutzen.

Im Unterschied zur schon beschriebenen Replikation wartet der Master in diesem Fall auf eine Bestätigung des Slaves, dass Änderungen zumindest in der Log-Datei eines der Slaves protokolliert wurden. Dieser Modus wird allerdings automatisch verlassen, wenn der Master zu lange warten müsste.

Merkmale dieser Lösung:

- Semi-Synchron
- Local Storage
- Hot-Standby
- Kurze Umschaltzeit
- Kein Datenverlust
- Unterstützt gängige Storage-Engines
- Single-Threaded Replikation
- Failback komplex
- Open Source

MHA® - Master High Availability Manager

MHA - Master High Availability Manager - automatisiert einige Schritte eines manuellen Switchover einer MySQL-Replikation, und verbessert dadurch die Nutzung als Hochverfügbarkeitslösung.

Wie schon erwähnt enthält MySQL® Server selbst keine Funktionalität, eine Failover-Lösung aufzusetzen – also eine automatisierte Umschaltung. Zudem kann es vorkommen, dass eine Transaktion zwar in der Log-Datei des Masters protokolliert ist, die Replikation auf den Slave aber nicht mehr durchgeführt werden konnte. Hier kommt MHA ins Spiel.

Basierend auf Perl-Scripts adressiert MHA folgende Punkte:

- Monitoring des Masters
- Automatisches Failover vom Master zu ein einem Slave
- Automatische Synchronisation der nicht übertragenen Transaktion, wenn Zugriff auf die LOG-Datei des Masters besteht
- Automatische Neu-Konfiguration anderer Slaves zum neuen Master

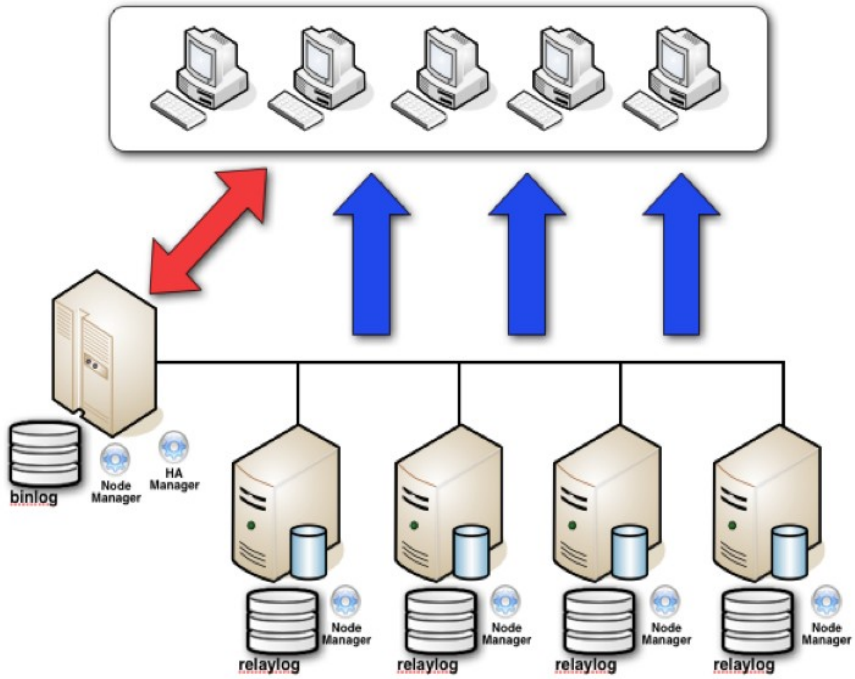

Abb. 2: MHA® – Master High Availability Manager

Abbildung 2 zeigt die aus der MySQL-Replikation bekannte Umgebung, die MySQL-Datenbanken Server erhalten einen Node Manager, zusätzlich wird ein HA Manager benötigt, welcher das Monitoring des Masters übernehmen kann und die Steuerung des Failover übernimmt.

Merkmale dieser Lösung:

- Asynchron oder Semi-Synchron
- Local Storage
- Hot-Standby
- Kurze Umschaltzeit
- Kein Datenverlust (Semi-Synchrone Replikation)
- Unterstützt gängige Storage-Engines
- Single-Threaded Replikation
- Failback komplex
- Open Source

Auf Storage aufbauende Verfügbarkeitslösungen

Ein anderer Ansatz, eine MySQL- oder MariaDB-Instanz hochverfügbar zu machen, stellt der Aufbau einer Aktiv/Passiv-Lösung dar. Hier wird ein redundantes System auf Basis der vom Datenbank-Server genutzten Dateien erzeugt. Zwei Ansätze wollen wir hier betrachten:

- Synchrone Replikation über DRBD®
- Shared Storage Cluster

Synchrone Replikation über DRBD®

DRBD – Distributed Replicated Block Device – ist eine weit verbreite Lösung in Verbindung mit MySQL Server, wenn ein synchroner HA Cluster aufgebaut werden soll. Hier wird ein Block-Device über Netzwerk zwischen zwei Linux-Servern gespiegelt, vergleichbar mit einem Netzwerk-RAID. In Kombination mit Linux Heartbeat und z.B. Pacemaker kann ein HA Cluster mit Failover implementiert werden.

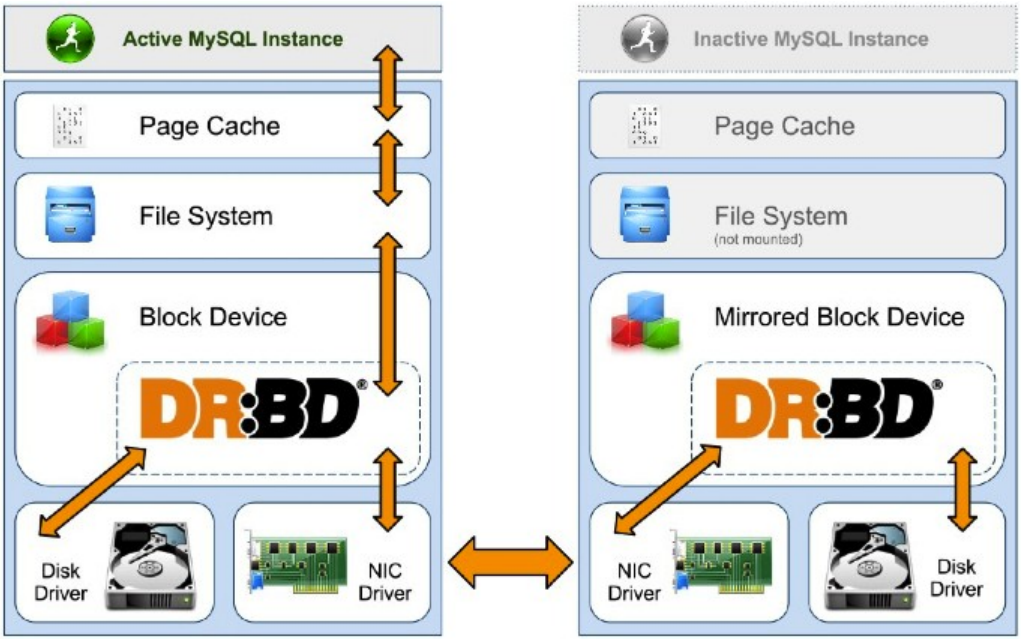

Abb. 3: DRBD – Distributed Replicated Block Device

DRBD kennt hier zwei Stati, Primär und Sekundär. Der als Primär definierte Knoten enthält die die aktive DB-Server Instanz, welche Lese- und Schreib-Anforderungen bedient. Im Unterschied zur Replikation kann der sekundäre Knoten nicht für Lese-Zugriffe genutzt werden, da die DB-Server Instanz auf diesem Knoten erst bei einem Failover gestartet wird. Dies sorgt auch für längere Umschalt-Zeiten, dafür enthält man die Vorteile einer synchronen Replikation.

Da der im Fehlerfall übernehmende Knoten letztendlich auf einen identischen Datei-Bestand zugreifen wird, sind andere Gegebenheiten zu beachten. Die Dateien müssen sich nicht in einem konsistenten Zustand befinden, es ist daher ein Recovery nötig. Eine Lösung mit DRBD sollte daher nur mit „Crash-Save“ Storage Engines wie XtraDB oder InnoDB verwendet werden.

Merkmale dieser Lösung:

- Synchrone Replikation
- Active / Passive
- Local Storage
- Umschaltzeit abhängig von Recovery-Zeit
- Kein Datenverlust
- Crash-Save Storage Engines (XtraDB, InnoDB)
- Failback teilweise durch Re-Sync möglich
- Linux

Shared Storage Cluster

Die Nutzung eines Shared Storage Clusters verfolgt den gleichen Ansatz wie für DRBD beschrieben, anstatt des Einsatzes einer Replikation eines Block-Devices wird hier ein Shared Storage verwendet, typischerweise ein Shared Storage Area Network (SAN) oder Network Attached Storage (NAS). Die Daten-Redundanz muss hier durch das Shared-Storage-Device garantiert werden.

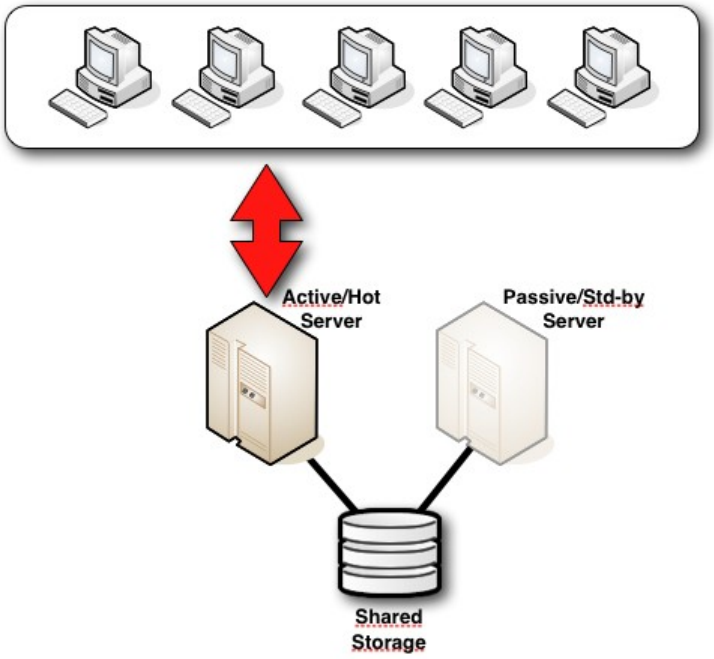

Abb. 4: Shared Storage

Der komplette HA-Cluster wird wiederum z.B. über Linux Heartbeat und Pacemaker, oder andere OS-Cluster implementiert.

Da auch hier die Datenbank-Instanz des passiven Knoten im Failover-Fall gestartet wird, diese wiederum auf die gleichen Dateien zugreift, wird auch hier nur die Nutzung von Storage-Engines wie XtraDB und InnoDB empfohlen, die ein Auto-Recovery durchführen können.

Ein Failback stellt sich in diesem Fall sehr einfach dar, da auch dann wieder auf die identischen Dateien zugegriffen werden kann.

Merkmale dieser Lösung:

- Synchron, keine Replikation nötig
- Active / Passive
- Shared Storage
- Umschaltzeit abhängig von Recovery-Zeit
- Kein Datenverlust
- Crash-Save Storage Engines (XtraDB, InnoDB)
- Einfaches Failback
- Höhere Systemkosten
- OS Cluster für Failover

Patches von MySQL® Server mit spezieller HA-Funktionalität

Einige Hochverfügbarkeitslösungen für MySQL-Datenbanken verfolgen den Ansatz, eigene Replikations-Ansätze in innerhalb des Servers zu implementieren. Hierzu ist es jedoch nötig, den Code von MySQL® Server zu modifizieren. Das führt dazu, dass Standard-Binaries nicht verwendet werden können. Bekannte Lösungen sind hier SchoonerSQL™ oder Galera.

Galera Cluster

Galera Cluster besteht aus der Galera Library, welche die Galera Replikation implementiert, sowie einer Erweiterung von MySQL® Server um eine Replikations-API.

Nicht zuletzt durch die Galera Replikation können Funktionen wie die synchrone Replikation, Global Transaction IDs, Active/Active Multi-Master genutzt werden. Die parallele Replikation kann zu besserer Performance bei schreiblastigen Anwendungen führen. Die synchrone Replikation basiert auf der Nutzungen von durch die Storage-Engine angebotene Transaktions-Sicherheit. Galera Cluster kann daher nur für InnoDB / XtraDB Tabellen verwendet werden.

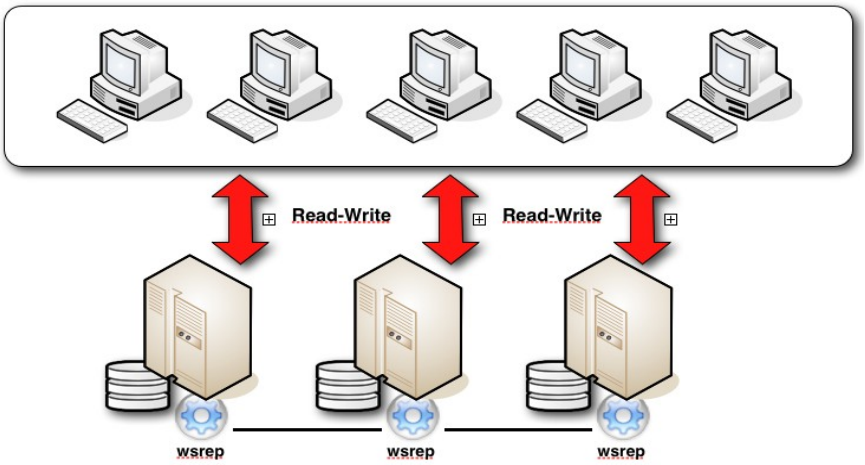

Abb. 5: Galera Cluster

Galera erlaubt das parallele Lesen und Schreiben auf allen Knoten und führt die Konflikt-Lösung durch “Aborted Commit” durch. Dies führt noch zu einigen Einschränkungen, doch auch die klassische Verwendung mit einem schreibenden Knoten bringt den Vorteil der synchronen parallelen Replikation. Ein automatisches Failover ist für diesen Fall allerdings nicht implementiert. Galera Cluster benötigt 3 Knoten.

Merkmale dieser Lösung:

- Synchron
- Active / Active (Mit Einschränkungen)
- Local Storage
- Schnelle Umschaltzeit
- Kein Datenverlust
- Transaktionale Storage Engines (XtraDB, InnoDB)
- Global Transaction ID
- Parallele Replikation
- Einfaches Failback
- Open Source

HA spezialisierten Storage-Engines

Hochverfügbarkeitslösungen wie MySQL® Cluster implementieren die Hochverfügbarkeit direkt in der Strorage Engine. Um diese Lösung nutzen zu können, muss also der Wechsel auf eine andere Storage-Engine durchgeführt werden. Dabei muss speziell auf die angebotenen Funktionalitäten der Storage-Engine geachtet werden, im Gegenzug wird aber eine hochwertige Hochverfügbarkeitslösung angeboten, die speziell auf das Thema Hochverfügbarkeit hin implementiert wurde.

MySQL® Cluster

MySQL Cluster kann als eine Hochverfügbarkeitslösung beschrieben werden, die den den höchsten Ansprüchen genügen kann, dies unter Nutzung von Standard Server Hardware. MySQL Cluster verfolgt das Konzept, in keinem Prozess einen Single Point of Failure dazustellen. Die MySQL Cluster Storage Engine ist dabei voll transaktionssicher und wurde ursprünglich speziell für Anwendungen mit hohem Transaktions-Aufkommen, aber kleinen Transaktionen konzipiert. MySQL Cluster ist daher speziell für Telekommunikations-Anwendungen und Session-Management sehr gut geeignet.

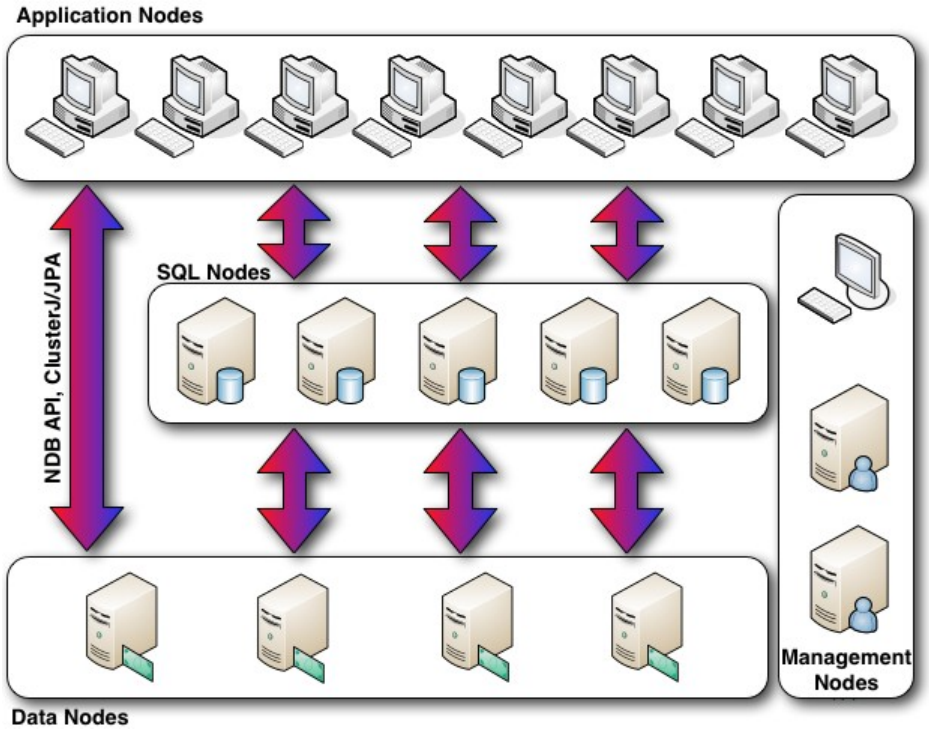

Abb. 6: MySQL® Cluster

Die Architektur und Konfiguration von MySQL Cluster ist allerdings sehr komplex. Wir nennen hier daher nur kurz die verschiedenen Elemente.

Die Anwendung verbindet sich im Allgemeinen über MySQL® Server, in diesem Fall SQL Knoten genannt. Jeder SQL Knoten kann Lese- und Schreib-Anforderungen bearbeiten. Die Transaktionen, Datenhaltung, HA Funktionalität usw., werden durch die Daten-Knoten abgedeckt. Die Daten-Knoten garantieren auch die synchrone Replikation der Daten. Management Knoten enthalten die Cluster-Konfiguration.

Merkmale dieser Lösung:

- Synchron
- Active / Active
- Local Storage

- Schnelle Umschaltzeit
- Kein Datenverlust
- Nur NDB Storage Engine
- Applikations-Cluster
- Komplex
- Einfaches Failback
- Open Source

HA Middleware

Unter HA Middleware kann man Lösungen betrachten, die auf Standard-Funktionalität und Standard-Binaries von MySQL-Datenbanken aufbauen, jedoch durch eigene Erweiterungen zusätzliche Funktionalität bezüglich der Hochverfügbarkeit implementieren. Continuent Tungsten Replicator und Continuent Tungsten Enterprise z.B. verfolgen diesen Ansatz.

Continuent Tungsten®

Der Continuent Tungsten Replicator setzt auf der Replikations-LOG-Datei von MySQL-Datenbanken auf, wandelt diese in eine eigene asynchrone Replikation um, in der z.B. die in MySQL-Datenbanken fehlende Global Transaction ID eingefügt wird, durch welche Switchover und Failover zu anderen Servern einfacher werden. Zudem kann die parallele Replikation genutzt werden, sofern mehrere Datenbank-Schemen zum Einsatz kommen. Dadurch können bei MySQL-Datenbanken bekannte Symptome schreiblastiger Anwendungen umgangen werden, wo die Single-Threaded Replikation nicht mehr in der Lage ist, alle ankommenden Änderungen durchzuführen.

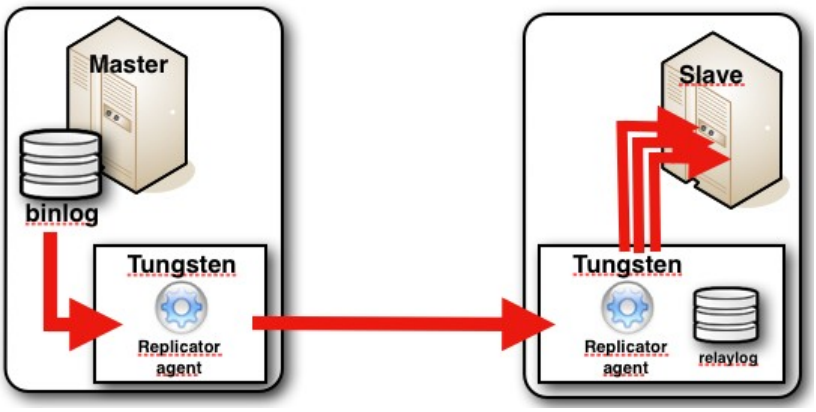

Abb. 7: Continuent Tungsten®

Wie der Abbildung zu entnehmen ist, werden auf Master und Slave jeweils Replicator Agents genutzt, um die Replikation durchzuführen. Ein Failover erfolgt hier noch manuell, jedoch wird bei er Nutzung mehrerer Slaves die Neu-Konfiguration durch die vorhandene Global Transaction ID einfacher. Zudem sind Architekturen wie Multi-Source möglich.

Mit Continuent Tungsten Enterprise, und den darin enthaltenen zusätzlichen Modulen, wird aus der Replikationslösung eine komplette Hochverfügbarkeitslösung mit automatischem Failover, zentralem Management des Clusters, eigenem Backup- und Restore-Verfahren. Anwendungen verbinden sich über einen speziellen Connector zum Datenbank-Server, welcher wiederum mit dem Cluster-Manager und Replikator verbunden ist. Dies erlaubt eine für die Anwendung transparente HA-Lösung, die zudem im Connector eine Lese-Schreib-Verteilung durchführen kann.

Merkmale dieser Lösung:

- Asynchron
- Read-Scalability
- Local Storage
- Schnelle Umschaltzeit
- Hot Standby
- Automatisches Failover
- Datenverlust möglich
- Global Transaction ID
- Parallele Replikation (pro Schema)

# # #

# # #

Kontaktadresse:
Ralf Gebhardt
SkySQL Ab
Vävarsvägen 11
Finland - 02630 Esbo

Telefon: +49 (0) 7457-930 646
E-Mail ralf.gebhardt@skysql.com
Internet: www.skysql.com