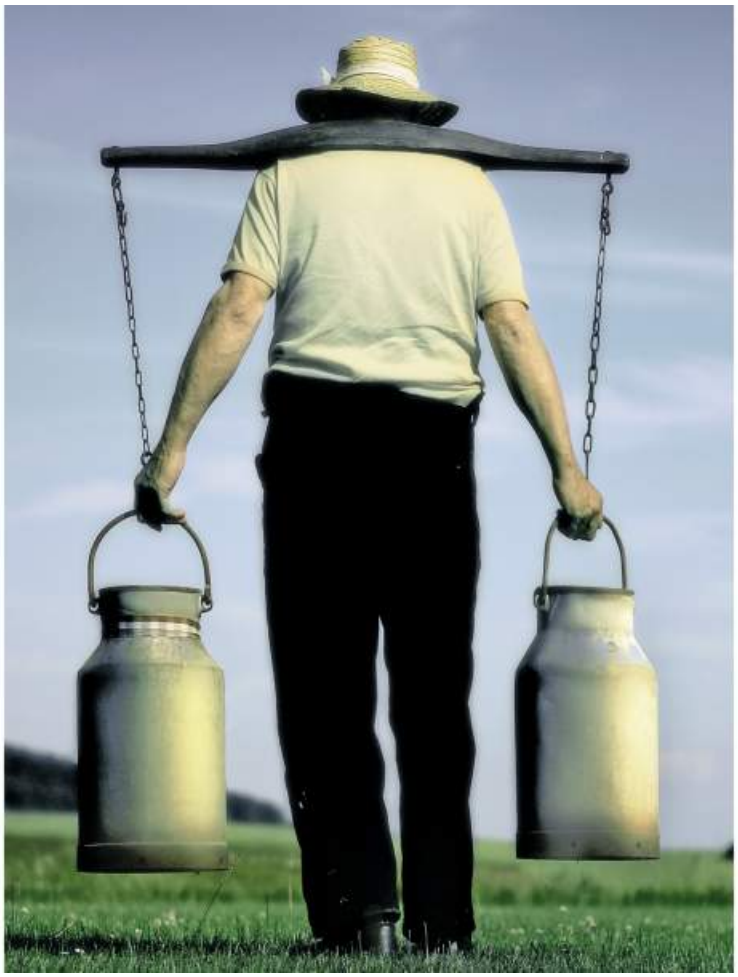
iX 3/2019 S. 106

WISSEN

DATENBANKEN



Fünf Load Balancer und Reverse Proxys für MySQL

Lastenverteiler

Sven Lankes

Mit der klassischen Master-Slave-Replikation, der Galera Engine und Oracles Group Replication gibt es im MySQL-Umfeld inzwischen drei verschiedene Möglichkeiten, hochverfügbare, flexibel skalierende Datenbanksysteme zu bauen. Diese Mechanismen kümmern sich jedoch nicht darum, wie die Querys zu den einzelnen Cluster-Nodes kommen. Verschiedene Werkzeuge füllen die Lücke.

iX-TRACT

- Hochverfügbarkeit und flexible Skalierbarkeit lassen sich für Datenbanken über Replikationsmechanismen realisieren.
- Die Verteilung der eingehenden Anfragen auf die Server übernimmt ein Load Balancer oder ein Reverse Proxy.
- Layer-4-Proxys arbeiten auf der TCP-Transportebene und sind generisch.
- Layer-7-Proxys arbeiten auf der Anwendungsebene des OSI-Modells und leiten Pakete anhand ihres Inhalts weiter.

Der klassische Weg, Daten zwischen mehreren MySQL-Servern auf dem gleichen Stand zu halten, ist die asynchrone Master-Slave-Replikation. Hierbei verarbeitet ein Master-Server alle datenverändernden Querys, während eine im Prinzip beliebige Zahl von Slave-Servern sich die Änderungen bei diesem abholt und damit die lesenden Anfragen bedient. In den letzten Jahren haben sich die zwei recht ähnlichen Cluster-Lösungen Galera und MySQL Group Replication dazugesellt. Sie beherrschen sogar einen fast synchronen Betrieb und erlauben – mit Einschränkungen –

das Schreiben auf mehrere Datenbank-Nodes. Diesen Ansätzen gemein ist, dass ein Datenbestand auf mehreren Serverinstanzen verteilt vorliegt und es einen Mechanismus gibt, der sicherstellt, dass eine Datenänderung zeitnah auf allen Instanzen ankommt.

Wenn sich eine Datenbank auf mehrere Rechner verteilt, stellt sich jedoch unweigerlich die Frage, wie man die hinzugewonnene Ausfallsicherheit und (Lese-)Kapazität am besten nutzen kann. Schließlich kann die Datenbank jetzt auf mehreren Instanzen Anfragen beantworten, während die anfragende Anwendung fast immer nur eine Instanz (IP-Adresse/Port) ansprechen kann. Ausnahmen sind Anwendungen, die auf Konnektoren aufsetzen, die einen Failover-Modus unterstützen, wie den JDBC-Treiber Connector/J von Oracle oder den noch nicht sehr weit verbreiteten PHP-Treiber mysqlnd_ms.

Wenn also Datenbankanfragen ohne Wissen und Unterstützung der Anwendung auf mehrere Nodes verteilt werden sollen, ist ein Load Balancer oder Proxy vonnöten. Ob dieser zur Erhöhung der Verfügbarkeit oder zur Steigerung des Durchsatzes zum Einsatz kommt, ist dabei irrelevant.

Redundanzbedarf reduziert

Auch wenn im Folgenden immer wieder von Proxys die Rede ist, handelt es sich im Grunde um Reverse Proxys. Die Clients verbinden sich nicht mit dem finalen Datenbank-Node (den sie ja nicht kennen), sondern mit dem Reverse-Proxy-Server. Dieser reicht die Anfrage dann nach Parametern wie Verfügbarkeit, Auslastung und Antwortzeiten an einen Datenbank-Node weiter.

Klassisch verwendet man Load Balancer für die zentrale Verteilung eingehender Requests auf eine Reihe von Worker-Nodes. Hierbei ist der Lastenverteiler oder Proxy immer eine eigenständige Komponente im Netzwerk, die – meist auf dedizierter Hardware und mit komplexen HA-Funktionen ausgestattet – ihren Dienst tut.

In modernen, von Virtualisierung und Containern geprägten Umgebungen ist ein solches Setup immer weniger sinnvoll. Stattdessen ist der Platz des Proxys heute direkt

neben der Anwendung, sodass beispielsweise pro Webserver, der eine PHP-Anwendung bereitstellt, eine eigene Proxy-Instanz zum Einsatz kommt. Diese Herangehensweise hat den Vorteil, dass Proxy-Knoten nicht mehr redundant ausgelegt sein müssen. Der in der Vergangenheit für den reibungslosen Übergang zwischen zwei Nodes notwendige Aufwand, bei dem oft über das Virtual Router Redundancy Protocol (VRRP) oder das Ressourcenverwaltungstool Pacemaker eine virtuelle IP-Adresse von einem Node auf einen anderen umgezogen werden musste, entfällt. Das steigert die Flexibilität. Außerdem steigt die Erfolgsrate bei tatsächlichen Problemen. Oft sind nämlich gerade die selten bis nie erprobten Failover-Szenarien die Ursache für längere Ausfälle der Infrastruktur.

Wo laufen sie denn?

	ProxySQL	haproxy	Mysql-router	maxscale	nginx plus
Master-Slave	X	X	-	X	X
Group Replication	X	X	X	-	X
Galera	X	X	-	X	X
AWS RDS Aurora	X	X	-	X	X

Kompatibilität der vorgestellten Load Balancer mit vier Cluster-Lösungen (Abb. 1)

Die im Folgenden vorgestellten Load Balancer und Proxys befinden sich entweder auf Layer 4 oder Layer 7 des OSI-Schichtenmodells. Am weitesten verbreitet sind Lösungen, die auf Layer 4, der Transportschicht, aufsetzen. Die TCP-Verbindung wird zwischen Client und Server aufgebaut, der Proxy leitet die Pakete – ohne von deren Inhalt Kenntnis zu haben – lediglich zum Server weiter.

Ein Layer-7-Proxy hingegen arbeitet auf der Applikationsschicht des OSI-Modells und „versteht" damit das MySQL-Protokoll. Das bedeutet, dass er – je nach Inhalt einer Query – unterschiedliche Aktionen ausführen kann. Wenn erforderlich, kann er eine Query auf dem Weg zum Server auch verändern.

Bei einem Layer-7-Proxy hat man es stets mit zwei TCP-Verbindungen zu tun. Der Cli-

ent baut eine Verbindung zum Proxy auf und der Proxy wiederum eine zum Datenbank-Node. Das hat auch Einfluss auf die Geschwindigkeit. Layer-7-Proxys bieten meist die Option des Connection Pooling, bei dem die Verbindung vom Proxy zum Datenbankknoten dauerhaft bestehen bleibt und nicht bei jeder eingehenden Verbindung neu aufgebaut werden muss.

Proxys: Woher stammt eine Anfrage?

Wer einen zentralen Proxy betreiben möchte, wird bei allen vorgestellten Produkten auf folgendes Problem stoßen: Alle an den Datenbankservern ankommenden Verbindungen haben nun als Absender die Adresse des Proxyservers. Welcher Client die Verbindung ursprünglich aufgebaut hat, lässt sich nur noch mittels Logfile-Analyse auf Proxy und Datenbankserver herausfinden. Für den DBA ist dieser Umstand sowohl aus Debugging- als auch aus Security-Sicht problematisch. Willi Tarreau, der Hauptentwickler von HAProxy, hat deshalb schon 2010 die Proxy Protocol Specification herausgegeben. Sie definiert eine einfache Methode, bei der der Proxy die ursprünglichen TCP-Verbindungsparameter beim TCP-Verbindungsaufbau voranschickt. Diese Methode verbreitet sich langsam, aber sicher in der MySQL-Welt. Neben HAProxy kommt auch MariaDB MaxScale damit zurecht. Auf der Datenbankserverseite unterstützt Percona Server das Protokoll seit Version 5.7, MariaDB seit 10.3.7. Nur Nutzer des Oracle-Produktes schauen in die Röhre. Hier ist die Methode bisher nicht implementiert.

Da es sich bei Layer-4-Proxys um generische Proxys handelt, die auch für alle anderen Protokolle (zumindest TCP-basierte) herhalten müssen, ist das Probing hier oft problematisch. Natürlich möchte der Proxy-Prozess aber wissen, ob ein Node weiterhin funktioniert. Der einfachste Health Check hierfür ist die Prüfung, ob sich zum Datenbankknoten eine TCP-Verbindung auf Port 3306 aufbauen lässt, auf dem MySQL standardmäßig nach Anfragen lauscht. Klappt die Verbindung, sagt das jedoch noch nichts über

die Gesundheit des Datenbankknotens und dessen Fähigkeit, eine eingehende Anfrage tatsächlich zu beantworten, aus. In Layer-4-Setups muss der Administrator daher neben der eigentlichen Einrichtung des Proxys oft zusätzlich komplexe Health-Check-Konstrukte konfigurieren.

Mit Ausnahme von Oracles MySQL Router unterstützen alle im Folgenden vorgestellten Tools auch das von der MariaDB Foundation herausgegebene Konkurrenzprodukt MariaDB sowie die von Amazon vertriebene MySQL-kompatible Cloud-Datenbank RDS Aurora.

HAProxy

Bei HAProxy handelt es sich um einen klassischen Load Balancer auf Schicht 4 des OSI-Modells. Um die Verfügbarkeit eines Nodes zu prüfen, bietet HAProxy den Health-Check-Typ *mysql-check* an. Dieser meldet dann Erfolg zurück, wenn der MySQL-Server den angegebenen User akzeptiert hat. Damit das funktioniert, muss auf den Datenbankservern ein Benutzer eingerichtet werden, den HAProxy für die Überprüfung verwenden kann. An dieser Stelle ist Vorsicht geboten: Gerade in Clusterumgebungen gibt es durchaus Szenarien, in denen der Health Check zwar erfolgreich, eine schreibende oder lesende Abfrage jedoch nicht möglich ist.

HAProxy kann alle in diesem Artikel angesprochenen Clusterkonstrukte bedienen. Lediglich für die Group Replication ist ein zusätzlicher xinetd-gesteuerter Health-Check-Dienst auf den Datenbank-Nodes empfehlenswert (alle Links zum Artikel unter ix.de/ix1903106).

Listing 1: Ein Galera-Cluster lässt sich in HAProxy in wenigen Zeilen konfigurieren

```
<@$p>listen galera 10.44.1.10:3306
     balance source
     mode tcp
     option tcpka
```

```
    option mysql-check user haproxy
    server node1 10.44.1.1:3306 check weight 1
    server node2 10.44.1.2:3306 check weight 1
    server node2 10.44.1.3:3306 check weight 1
```

Statusübersicht eines Galera-Clusters in der Weboberfläche von HAProxy (Abb. 2)

HAProxy ist als generischer Software-Load-Balancer sehr weit verbreitet. Die Einrichtung des Tools erfolgt über die Konfigurationsdatei *HAProxy.cfg*. Listing 1 zeigt, wie sich damit schnell ein Galera-Cluster einbinden lässt. Der Status eines Clusters lässt sich über ein Webinterface anzeigen. Wer keine besonderen Anforderungen hat, ist mit diesem Tool auch für MySQL-Load-Balancing gut bedient.

MySQL Router

Oracles Lösung für hochverfügbare MySQL-Cluster heißt InnoDB Cluster und besteht aus den drei Komponenten MySQL Server, MySQL Shell und MySQL Router. Der MySQL InnoDB Cluster ist wie das Basispaket sowohl in einer Community Edition unter der GPL als auch als Enterprise Edition unter einer proprietären Oracle-Lizenz verfügbar.

Seit Version 5.7.17 bietet Oracle für den MySQL Server das Plug-in Group Replication als Standard-Replikationslösung. Das Tool ist mit der aktuellen MySQL-Version 8.0.14 weitergereift und eignet sich inzwischen durchaus für den Produktiveinsatz.

Die zweite Komponente ist die MySQL Shell, bei der es sich um ein wahlweise in JavaScript-, Python- oder SQL-Syntax zu bedienendes Administrations-Frontend für MyS-

QL-Server handelt. Mithilfe der MySQL Shell lässt sich die initiale Konfiguration des Clusters bewerkstelligen. Später kann die Shell auch als leistungsfähiger und moderner Ersatz für den klassischen MySQL-Kommandozeilenclient eingesetzt werden.

Die dritte Komponente im InnoDB Cluster ist der Load Balancer. MySQL Router agiert wie HAProxy auf Layer 4 und leitet eingehende Verbindungen ungefiltert zu einem Datenbankknoten weiter. Da der Router jedoch speziell für den InnoDB Cluster entwickelt wurde, kann er auf die aufwendige Probing-Logik verzichten, die die anderen vorgestellten Lösungen implementieren müssen. Der MySQL Router macht sich die Tatsache zunutze, dass jeder Knoten in einem funktionsfähigen Group-Replication-Setup immer weiß, welche Knoten gesund sind und funktionieren. Daher hält der Router nur eine einzelne Verbindung zum jeweils primären Group-Replication-Knoten und bezieht die Verfügbarkeitsinformationen aus den Metadaten des Clusters. Ein eigenes Probing erfolgt nicht.

Listing 2: MySQL Router konfiguriert sich selbst

```
[sven@localhost ~]$ sudo mysqlrouter --bootstrap root@172.19.0.2 --user=mysqlrouter
Please enter MySQL password for root:

Bootstrapping system MySQL Router instance...
Checking for old Router accounts
Creating account mysql_router1_klac0bm15ga6@'%'
MySQL Router  has now been configured for the InnoDB cluster 'testCluster'.

The following connection information can be used to connect to the cluster ⤤
 after MySQL Router has been started with generated configuration..

Classic MySQL protocol connections to cluster 'testCluster':
- Read/Write Connections: localhost:6446
- Read/Only Connections: localhost:6447
X protocol connections to cluster 'testCluster':
- Read/Write Connections: localhost:64460
- Read/Only Connections: localhost:64470.
```

Das führt dazu, dass die Konfiguration des Routers erfreulich simpel ist. Im einfachsten Fall reicht die Angabe eines Cluster-Nodes und der Router erstellt im Bootstrap-Modus eine funktionierende Konfiguration (Listing 2). Auch die Konfiguration komplexerer Setups geht dank der guten Dokumentation leicht von der Hand.

Oracles Enterprise-Kunden haben neben der einfachen Konfiguration ein weiteres schlagkräftiges Argument für den MySQL Router: Er ist im Enterprise-Support des Software-Riesen inbegriffen. Bei den anderen Tools sind für den Hersteller-Support separate Wartungsverträge erforderlich.

Für MariaDB-Nutzer steht der MySQL Router nicht zur Verfügung. Wer die binärkompatible MySQL-Alternative Percona Server ab Version 8.0.13 einsetzt, kann sich allerdings einen InnoDB Cluster mit Percona-Kern bauen.

ProxySQL

ProxySQL ist ein aufstrebender quelloffener Load Balancer, bei dem die Verteilung von Querys auf Datenbank-Nodes nur eines von vielen Features ist. Geschrieben hat das Tool der ehemalige Systemadministrator und DBA René Cannaò.

Anders als bei HAProxy und Oracles MySQL Router handelt es sich bei ProxySQL um einen L7-Load-Balancer. Er versteht das MySQL-Protokoll und kann wenn gewünscht sogar die Querys nach vom Administrator definierten Regeln verändern, bevor er sie weiterschickt. Das erlaubt es dem DBA, problematische Querys anzupassen, ohne Zugriff auf die Applikationsschicht zu benötigen.

Außerdem kann ProxySQL eingehende Querys Regex-basiert klassifizieren und unterschiedlich behandeln. So lässt sich der in Cluster-Setups viel gewünschte Read-Write-Split herstellen, bei dem Querys vom Typ *SELECT* auf mehrere Nodes verteilt werden, während der Proxy datenverändernde Querys immer zum gleichen Node weiterleitet.

In der Praxis kann eine solche konfigurationsgesteuerte Unterscheidung zwischen Lese- und Schreibquerys allerdings sehr schnell zu unerwünschten Nebeneffekten füh-

ren. Wenn man zum Beispiel alle SELECT-Querys mit dem regulären Ausdruck ^SELECT zu Slave-Nodes schicken möchte, würden nach diesem Muster auch Querys der Art *SELECT FOR UPDATE* entsprechend verteilt werden. Solche Querys sollten jedoch einen Lock ausführen und müssten wie eine Schreibquery behandelt werden. Deshalb sollte man besser auf die Automatik verzichten und nach einiger Betriebszeit der Datenbankanwendungen die von ProxySQL zur Verfügung gestellten Statistiken heranziehen. Aus diesen lassen sich dann händisch besonders teure Lesequerys identifizieren. HAProxy normalisiert die Querys für die Statistikgenerierung. Aus der normalisierten Form wird anschließend ein Hash berechnet, der alle Querys der gleichen Art abdeckt – ganz egal, mit welchen Parametern sie aufgerufen wurden.

Die Einstellungsmöglichkeiten von ProxySQL sind mächtig, aber gewöhnungsbedürftig. Die Konfiguration ist dreischichtig: Es gibt den Runtime-Layer, den Disk-Layer und den Memory-Layer. Beim ersten Start liest der Proxy eine Konfigurationsdatei (Disk-Layer), die er sowohl in den Memory-Layer als auch in den Runtime-Layer lädt. Änderungen zur Laufzeit macht der DBA über ein MySQL-kompatibles Interface mit jedem MySQL-Client im Memory-Layer. Um die vorgenommenen Änderungen zu aktivieren, muss man sie mit speziellen Befehlen auf dem interaktiven Interface wie zum Beispiel *LOAD MYSQL SERVERS FROM MEMORY* vom Memory- in den Runtime-Layer verschieben.

Besonders gewöhnungsbedürftig ist, dass bei folgenden Starts nicht mehr die Konfigurationsdatei, sondern eine zwischenzeitlich erzeugte Plattenrepräsentation des Memory-Layers geladen wird.

Auch wenn die Konfigurationsvarianten im ersten Augenblick befremdlich wirken mögen, so erlauben sie dem Nutzer komplexe Konfigurationsänderungen zur Laufzeit.

MaxScale

MaxScale ist die Queryverteilungslösung der MariaDB Corporation. Das Unternehmen wurde 2008 von MySQL-Erfinder Michael „Monty“ Widenius als Antwort auf die Übernahme von Sun (und damit MySQL) durch Oracle gegründet. Die Firma mit Stammsitz in Helsinki vertreibt seitdem mit MariaDB ein eigenes Datenbanksystem, das – zumin-

dest auf Protokollebene – zu MySQL kompatibel ist.

Bei MaxScale handelt es sich wie bei ProxySQL um einen Layer-7-Proxy, der umfangreiche Eingriffe in die Querys ermöglicht. Als Alleinstellungsmerkmal gegenüber den Mitbewerbern bietet MaxScale einen automatischen Read-Write-Split an. Lese- und Schreibquerys verteilt das System anhand einer eingebauten Heuristik mit dem *readwritesplit*-Modul. Verbunden mit der in Version 2.2 nachgelieferten Failover-Funktionalität ist MaxScale die einzige Lösung, die es erlaubt, lesende Querys automatisch über mehrere Slave-Nodes zu verteilen. Wenn der Master einmal ausfallen sollte, kann MaxScale automatisch den Slave mit dem aktuellsten Stand zum neuen Master machen.

Während die anderen vorgestellten Lösungen alle unter Open-Source-Lizenzen stehen, sieht das bei MaxScale seit Version 2.0 anders aus. Hier hat die MariaDB Corporation die Business Source License (BSL) aus der Taufe gehoben. Für den Load Balancer bedeutet dies, dass der Sourcecode zwar jederzeit verfügbar ist, aktuelle Versionen der Software allerdings nur bei Datenbankclustern mit weniger als drei Mitgliedern kostenlos verwendet werden dürfen.

Wer mehr als zwei Serverinstanzen einsetzen möchte, muss MaxScale als Teil des MariaDB-Enterprise-Angebotes lizenzieren oder auf eine bereits mindestens zwei Jahre alte Version zurückgreifen. Ein wirklich hochverfügbares System wird jedoch immer mindestens drei Nodes umfassen, damit beim Ausfall eines Nodes noch festgestellt werden kann, welcher Teil des Clusters nun die Mehrheit besitzt.

Zwei Jahre (Change Date) nach der Veröffentlichung einer Version darf sie nach der BSL auch unter einer Open-Source-Lizenz (hier: GPLv2 oder neuer) eingesetzt werden. Dieses Lizenzkonstrukt ermöglicht es, dass Software, an der die Copyright-Inhaber das Interesse verlieren, nach einer gewissen Wartezeit wieder frei weiterentwickelt werden kann. Da es sich bei den aktuellen Versionen von MaxScale aber erst mal nicht mehr um freie Software handelt, hat das Interesse an der Software in der Open-Source-Community mit der veränderten Lizenz sehr nachgelassen – zugunsten von ProxySQL.

nginx(+)

nginx, in erster Linie als Webserver und HTTP-Proxy bekannt, erlaubt seit Version 1.9 auch das Verteilen generischer TCP-Pakete. Damit wird die Software zum Layer-4-Load-Balancer und zu einer Möglichkeit, MySQL-Querys über mehrere Server zu verteilen.

Interessant ist diese Option allerdings hauptsächlich für zahlende Kunden von nginx, die Zugriff auf die nicht quelloffene Variante nginx+ haben. In der Open-Source-Version erlaubt nginx nämlich zwar die Lastverteilung, unterstützt aber kein Probing. Ausgefallene Nodes fallen also nicht auf. Hier gibt es aus der Community mit clustercheck-iptables ein Workaround. Dieses Skript wird auf den Nodes selbst installiert und schaltet beim „ungesunden" Knoten per lokaler Firewall-Regel den Netzwerkzugriff auf die Datenbank ab. Dadurch schlägt der TCP-Verbindungsaufbau fehl, worauf dann wiederum der Proxyserver ohne zusätzliche MySQL-Probing-Features reagieren kann. clustercheck-iptables lässt sich nicht nur für nginx, sondern auch für andere Tools wie HAProxy, keepalived oder das im Linux-Kernel integrierte Transport Layer Load Balancing (IPVS) verwenden. Diese eher fragile Technik wird jedoch aufgrund der Verfügbarkeit besserer Alternativen in der Praxis kaum eingesetzt.

Fazit

Wer Oracles InnoDB Cluster einsetzt und über die saubere Queryverteilung hinaus keine weiteren Wünsche hat, ist mit MySQL Router gut bedient. DBAs, die die steile Lernkurve von ProxySQL nicht fürchtet, haben mit dieser Lösung ein mächtiges und vielseitiges Tool in ihrer Werkzeugkiste. Wer den Aufwand scheut, bekommt mit HAProxy gut abgehangene Software ohne Kinderkrankheiten. MaxScale und nginx(+) hingegen sind vor allem für die Besitzer passender Lizenzen oder Wartungsverträge interessant. (akl@ix.de)

Sven Lankes

arbeitet bei der COCUS AG als Experte für Datenbanken und unterstützt Kunden bei Konzeption und Umsetzung von MySQL-Projekten.

Quellen

[1] Alle Onlineverweise im Artikel: ix.de/ix1903106

Kommentieren

Leserbrief schreiben

Auf Facebook teilen

Auf Twitter teilen