MariaDB und Galera

Chemnitzer Linux-Tage 2019

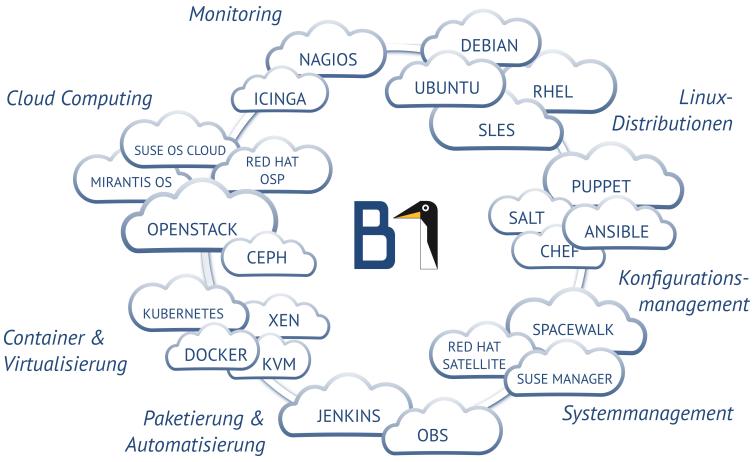
16. März 2019

Ralf Lang
Linux Consultant & Developer
B1 Systems GmbH
lang@b1-systems.de

Vorstellung B1 Systems

- gegründet 2004
- primär Linux/Open Source-Themen
- national & international tätig
- über 100 Mitarbeiter
- unabhängig von Soft- und Hardware-Herstellern
- Leistungsangebot:
  - Beratung & Consulting
  - Support
  - Entwicklung
  - Training
  - Betrieb
  - Lösungen
- Standorte in Rockolding, Köln, Berlin & Dresden

Schwerpunkte



Motivation für MariaDB Galera-Cluster

- Applikationen sollen *immer* laufen, Unterbrechungen vermeiden

Motivation für MariaDB Galera-Cluster

- Applikationen sollen *immer* laufen, Unterbrechungen vermeiden
- keine komplizierte Storage-Architektur (Multipath, SAN)

Motivation für MariaDB Galera-Cluster

- Applikationen sollen *immer* laufen, Unterbrechungen vermeiden
- keine komplizierte Storage-Architektur (Multipath, SAN)
- Verzicht auf teure Spitzenhardware

Begriffe

Cluster Zusammenschaltung mehrerer Computer als eine Einheit

Begriffe

Cluster Zusammenschaltung mehrerer Computer als eine Einheit

Knoten Die einzelnen Mitglieder eines Clusters

Begriffe

Cluster	Zusammenschaltung mehrerer Computer als eine Einheit
Knoten	Die einzelnen Mitglieder eines Clusters
Replikation	Übertragung von Änderungen eines Knotens an alle

Hohe Performance auch bei Lastspitzen

- Lesezugriffe auf beliebig viele Kopien der Datenbank verteilen

Hohe Performance auch bei Lastspitzen

- Lesezugriffe auf beliebig viele Kopien der Datenbank verteilen
- Die Kopien sind immer garantiert auf aktuellstem Stand

Hohe Performance auch bei Lastspitzen

- Lesezugriffe auf beliebig viele Kopien der Datenbank verteilen
- Die Kopien sind immer garantiert auf aktuellstem Stand
- Der Cluster lässt sich bei Bedarf auf mehr Knoten verbreitern

Hochverfügbarkeit

- keine Applikations-Downtime wegen Wartung des DB-Servers

Hochverfügbarkeit

- keine Applikations-Downtime wegen Wartung des DB-Servers
- einfache Wiederinbetriebnahme einzelner Knoten nach Ausfall

Hochverfügbarkeit

- keine Applikations-Downtime wegen Wartung des DB-Servers
- einfache Wiederinbetriebnahme einzelner Knoten nach Ausfall
- auch während eines Applikations- oder Schema-Updates

Free & Open Source

- Grundlage bildet eine MariaDB oder ein Oracle MySQL

Free & Open Source

- Grundlage bildet eine MariaDB oder ein Oracle MySQL
- Galera ist kostenlos und quelloffen verfügbar als Add-on

Free & Open Source

- Grundlage bildet eine MariaDB oder ein Oracle MySQL
- Galera ist kostenlos und quelloffen verfügbar als Add-on
- Galera arbeitet mit der Standard-Engine InnoDB

Das Demosystem

- 4x Raspberry Pi 3

Das Demosystem

- 4x Raspberry Pi 3
- 3x Galera-Knoten, 1x Anwendungsknoten

Master-Master-Prinzip

- alle Knoten sind gleichberechtigt

Master-Master-Prinzip

- alle Knoten sind gleichberechtigt
- Prinzipiell kann auf alle Knoten gleichzeitig geschrieben werden

Master-Master-Prinzip

- alle Knoten sind gleichberechtigt
- Prinzipiell kann auf alle Knoten gleichzeitig geschrieben werden
- aber möglichst nicht auf dieselben Daten ;)

Funktionsweise

- Share Nothing: Kein gemeinsamer Speicher der Knoten

Funktionsweise

- Share Nothing: Kein gemeinsamer Speicher der Knoten
- Semi-synchrone Replikation von Änderungen

Funktionsweise

- Share Nothing: Kein gemeinsamer Speicher der Knoten
- Semi-synchrone Replikation von Änderungen
- Daher: Der langsamste Knoten bestimmt den Schreibdurchsatz

Was funktioniert nicht?

- MyISAM und andere Engines

Was funktioniert nicht?

- MyISAM und andere Engines
- Schreiben des mysql Logs in einer Datenbanktabelle

Was funktioniert nicht?

- MyISAM und andere Engines
- Schreiben des mysql Logs in einer Datenbanktabelle
- kein `LOCK TABLES` bzw. `GET_LOCK()`

Was ist mit DDL, Rechten und Usern?

- Kommandos wie `ALTER`, `GRANT`, `CREATE`, `DROP` werden repliziert

Was ist mit DDL, Rechten und Usern?

- Kommandos wie `ALTER`, `GRANT`, `CREATE`, `DROP` werden repliziert
- Kommandos wie `UPDATE mysql.user ...` werden *nicht* repliziert

Was ist mit DDL, Rechten und Usern?

- Kommandos wie `ALTER`, `GRANT`, `CREATE`, `DROP` werden repliziert
- Kommandos wie `UPDATE mysql.user ...` werden *nicht* repliziert
- Tabellen des Systemschemas mysql sind nicht InnoDB

Optimistisches Locking

- Galera geht davon aus, dass eine lokal ausführbare Transaktion auch auf den anderen Nodes ausführbar ist.

Optimistisches Locking

- Galera geht davon aus, dass eine lokal ausführbare Transaktion auch auf den anderen Nodes ausführbar ist.
- Transaktionen auf verschiedenen Nodes können in der Commit-Phase kollidieren.

Optimistisches Locking

- Galera geht davon aus, dass eine lokal ausführbare Transaktion auch auf den anderen Nodes ausführbar ist.
- Transaktionen auf verschiedenen Nodes können in der Commit-Phase kollidieren.
- Transaktionen können deshalb in der Commit-Phase scheitern/abgelehnt werden.

Verhalten beim Commit

- Alle Änderungen einer Transaktion werden zu einem Write-Set zusammengefasst.

Verhalten beim Commit

- Alle Änderungen einer Transaktion werden zu einem Write-Set zusammengefasst.
- Dieses Write-Set wird nicht sofort geschrieben, sondern allen Nodes zur Entscheidung vorgelegt.

Verhalten beim Commit

- Alle Änderungen einer Transaktion werden zu einem Write-Set zusammengefasst.
- Dieses Write-Set wird nicht sofort geschrieben, sondern allen Nodes zur Entscheidung vorgelegt.
- Die Transaktion wird auf alle oder keinen Knoten angewandt (Mehrheit hat recht)

Notwendige Softwarepakete (OpenSUSE)

```
zypper addrepo https://download.opensuse.org/repositories/server:database/openSUSE_Leap_15.0/ \
server:database.repo

zypper install mysql mariadb-galera galera-3
```

Notwendige Einstellungen

Zeilenbasiertes Binärlog

```
binlog_format=ROW
```

Netzwerk!

```
bind-address=0.0.0.0
```

Nur InnoDB wird repliziert

```
default_storage_engine=innodb
innodb_autoinc_lock_mode=2 #Wichtig für Autoincrement
innodb_flush_log_at_trx_commit=0
```

Konfiguration des wsrep-Plugins

```
wsrep_provider=/usr/lib64/Pfad/zu/libgalera_smm.so
wsrep_cluster_name="mein_cluster"
wsrep_cluster_address="gcomm://knoten1,knoten2,knoten3"
```

Wahl einer Replikationsmethode

Wie sollen neue Cluster-Knoten ihre Daten bekommen?

Wahl einer Replikationsmethode

Wie sollen neue Cluster-Knoten ihre Daten bekommen?

```
wsrep_sst_method=rsync
```

Start des ersten Knotens

Spezielles Kommando nötig:

Start des ersten Knotens

Spezielles Kommando nötig:

```
galera_new_cluster
```

Erster Start der weiteren Knoten

Wie bei einer normalen Datenbank:

Erster Start der weiteren Knoten

Wie bei einer normalen Datenbank:

```
systemctl start mysql
```

Erster Start der weiteren Knoten

Wie bei einer normalen Datenbank:

```
systemctl start mysql
```

Knoten gleichen sich mit vorhandenem Cluster ab, bevor sie aktiv werden

Spätere Neustarts einzelner Knoten

- Ist der Rückstand gering, werden die fehlenden Änderungen eingespielt (*IST - Incremental State Transfer*)

Spätere Neustarts einzelner Knoten

- Ist der Rückstand gering, werden die fehlenden Änderungen eingespielt (*IST - Incremental State Transfer*)
- Ist der Rückstand zu groß, wird erneut der Cluster von null initialisiert (*SST - State Snapshot Transfer*)

Schema Upgrades 1/2

- Rolling Schema Upgrades möglich

Schema Upgrades 1/2

- Rolling Schema Upgrades möglich
- Knoten für Knoten wird aktualisiert

Schema Upgrades 2/2

Dabei sind besondere Regeln zu beachten:

Schema Upgrades 2/2

Dabei sind besondere Regeln zu beachten:

- neue Spalten nur hinten anfügen

Schema Upgrades 2/2

Dabei sind besondere Regeln zu beachten:

- neue Spalten nur hinten anfügen
- alte Spalten nur hinten löschen

Schema Upgrades 2/2

Dabei sind besondere Regeln zu beachten:

- neue Spalten nur hinten anfügen
- alte Spalten nur hinten löschen
- Änderung vorhandener Spalten nur in kompatible Typen

Lastverteilung und Failover mit MaxScale 1/2

- MaxScale ist ein mysql-Protokollproxy

Lastverteilung und Failover mit MaxScale 1/2

- MaxScale ist ein mysql-Protokollproxy
- analysiert Statements, kann Schreib- und Lesekommandos trennen

Lastverteilung und Failover mit MaxScale 1/2

- MaxScale ist ein mysql-Protokollproxy
- analysiert Statements, kann Schreib- und Lesekommandos trennen
- bedingt freie Lizenz

Lastverteilung und Failover mit MaxScale 2/2

- HA-Proxy

Lastverteilung und Failover mit MaxScale 2/2

- HA-Proxy
- free & open

Lastverteilung und Failover mit MaxScale 2/2

- HA-Proxy
- free & open
- Grundsoftware nur TCP-Proxy

Lastverteilung und Failover mit MaxScale 2/2

- HA-Proxy
- free & open
- Grundsoftware nur TCP-Proxy
- Plugins für Erkennung von Schreibkommandos verfügbar

Strategien 1/3

Read-Write Split

Strategien 1/3

Read-Write Split

- einfach umzusetzen

Strategien 1/3

Read-Write Split

- einfach umzusetzen
- günstig für leselastige Applikationen

Strategien 2/3

Vordefinierte Schreibpräferenzen

Strategien 2/3

Vordefinierte Schreibpräferenzen

- verschiedene Applikationen bevorzugen verschiedene Nodes

Strategien 2/3

Vordefinierte Schreibpräferenzen

- verschiedene Applikationen bevorzugen verschiedene Nodes
- komplexer

Strategien 2/3

Vordefinierte Schreibpräferenzen

- verschiedene Applikationen bevorzugen verschiedene Nodes
- komplexer
- kann Gesamtdurchsatz steigern

Strategien 3/3

Round Robin Write

Strategien 3/3

Round Robin Write

- nicht für alle Workloads geeignet

Strategien 3/3

Round Robin Write

- nicht für alle Workloads geeignet
- Gefahr von Kollisionen

Strategien 3/3

Round Robin Write

- nicht für alle Workloads geeignet
- Gefahr von Kollisionen
- Autocommit-Transaktionen können von Galera selbst erneut versucht werden

Vielen Dank für Ihre Aufmerksamkeit!

Bei weiteren Fragen wenden Sie sich bitte an info@b1-systems.de oder +49 (0)8457 - 931096