MariaDB

DOAG Konferenz 2015

Open Source Proxy MaxScale

Für MariaDB / MySQL DBAs und Entwickler

Ralf Gebhardt
Principal Sales Engineer

MariaDB Corporation

- Gegründet vom MySQL Kernteam, wie Michael “Monty” Widenius und David Axmark.
- Core-MariaDB Entwickler sind Mitarbeiter von MariaDB Corporation
- Entwicklungs-Unterstützung der MariaDB Foundation
    - Open Source Entity stellt Entwicklung von MariaDB aus der Community sicher – www.mariadb.org
- Innovative Produkte werten MariaDB auf.
- Enterprise Services wie 24/7 Support für MariaDB und MySQL.

Was ist MariaDB MaxScale?

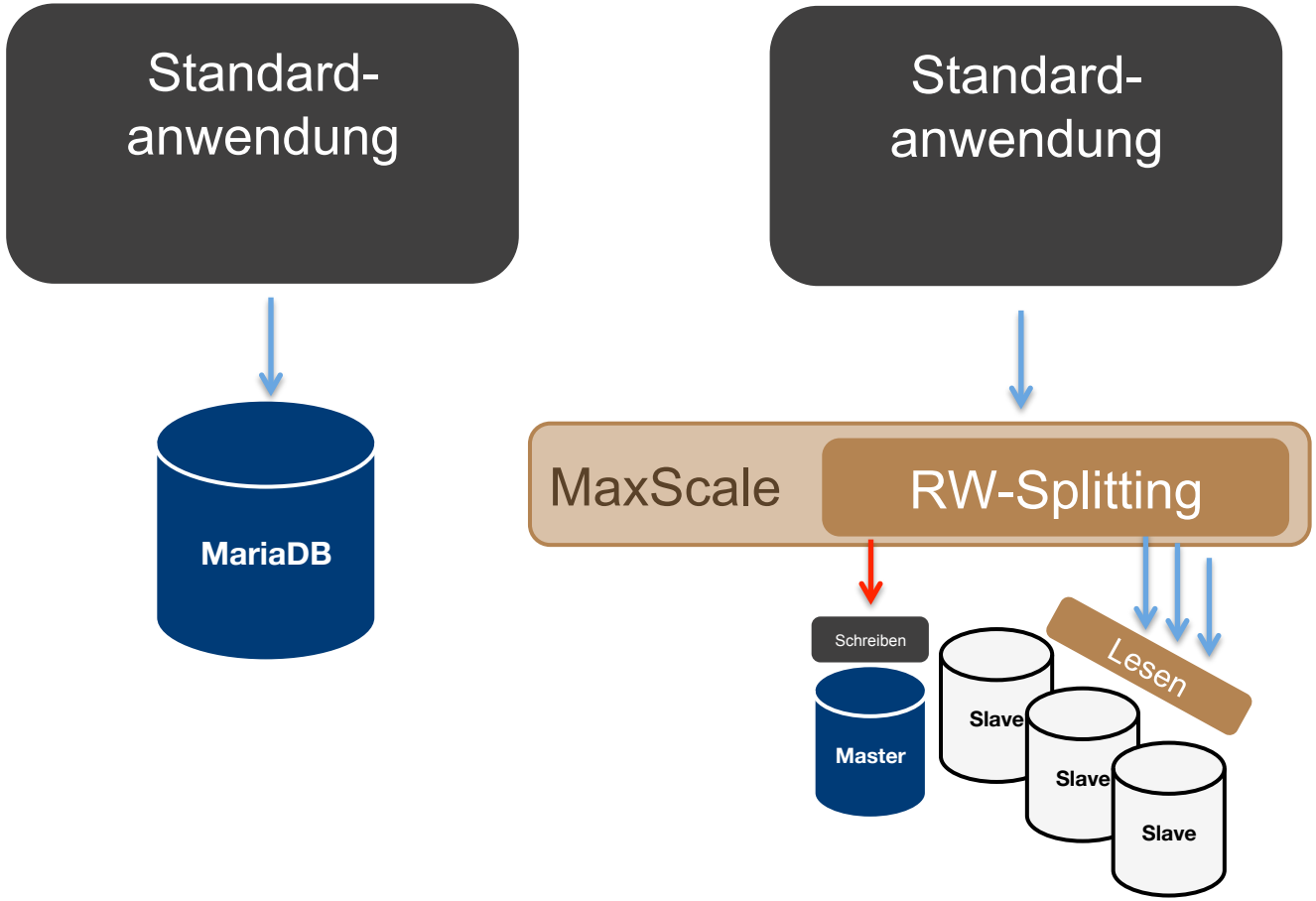
- Ein flexibler Daten-Gateway für
  - Skalierung
  - Hochverfügbarkeit
  - Sicherheit
  - Interoperabilität
- Flexibel konfigurierbare Gateway-Plattform
- „Database aware“
- „Plugable“ Architektur

Warum eine Gateway-Plattform?

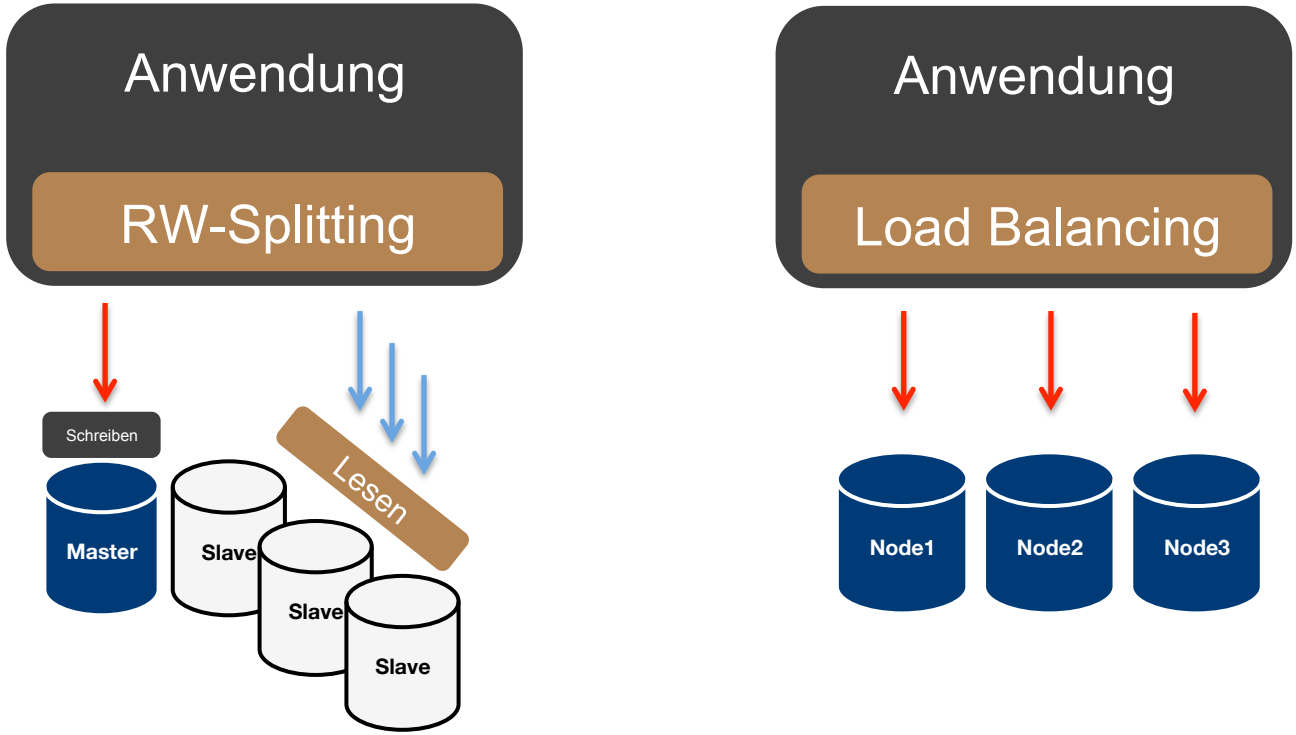
- Vereinfacht den Aufbau neuer Deployment-Architekturen

- Reduziert die Komplexität und Risiken in verschiedensten Umgebungen

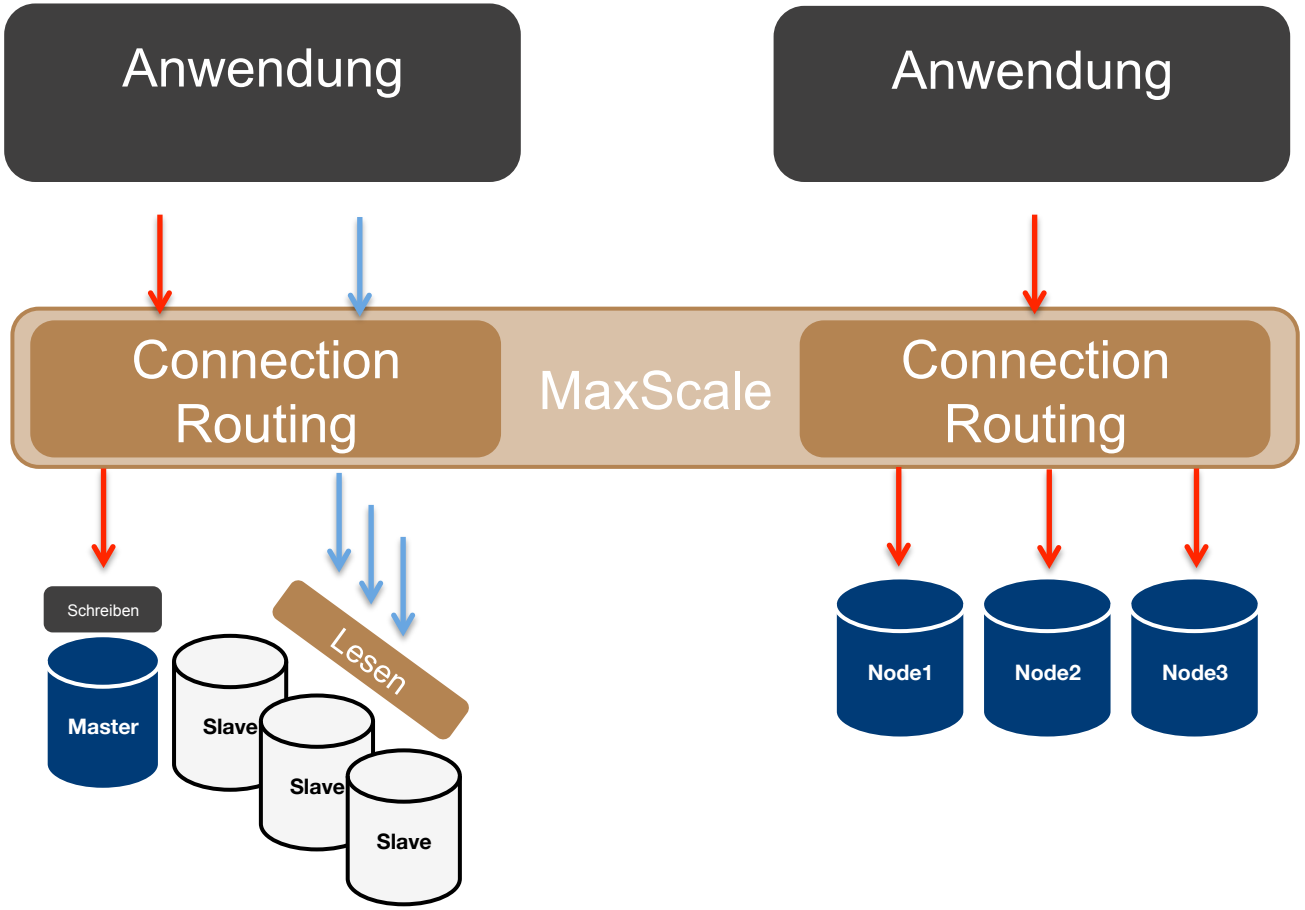
HA-Architektur für Standard-Anwendungen



Routing über Anwendung



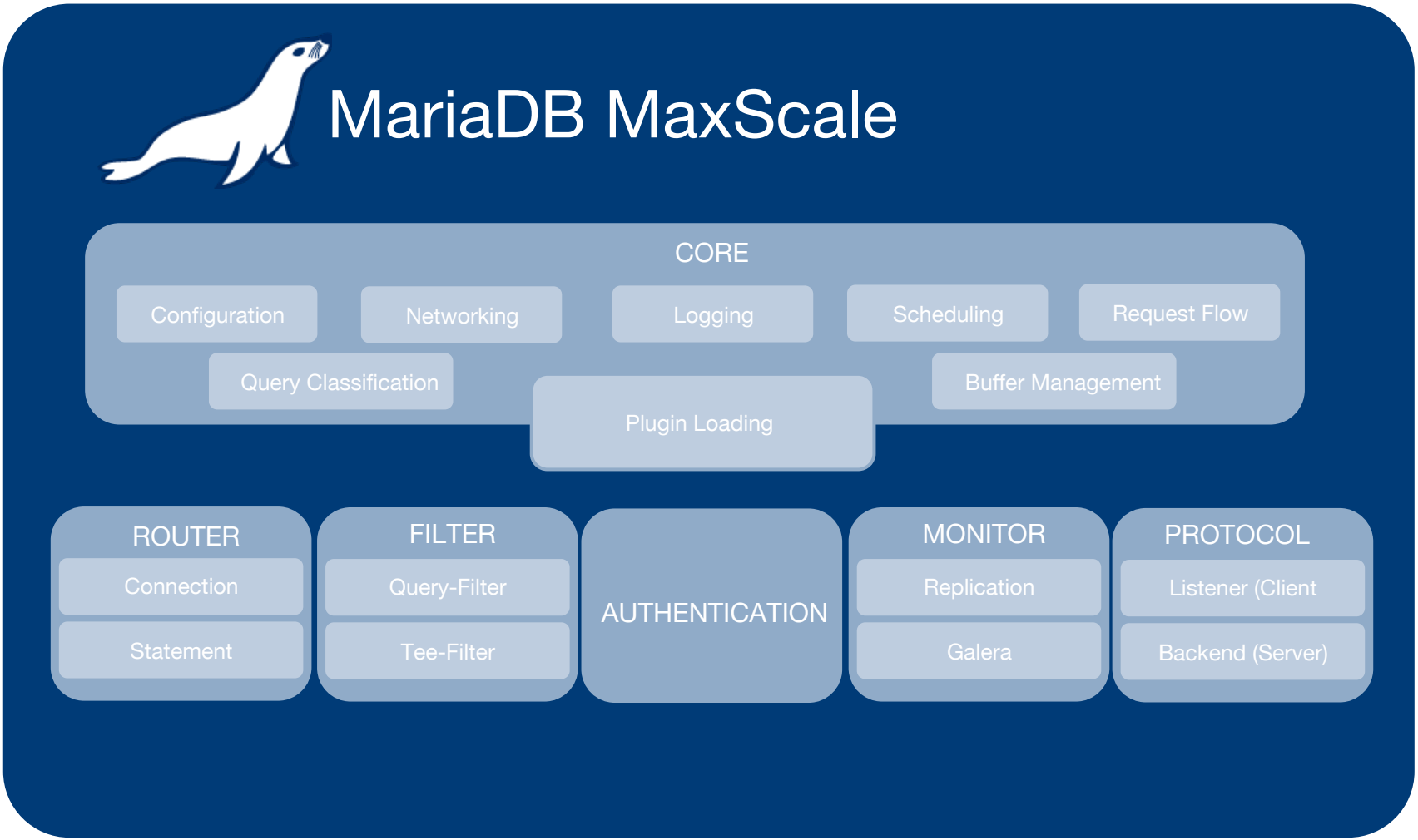
Routing über MaxScale



Warum „Database aware“?

- Angepasst an Datenbank-Umgebungen
- Kennt den Status der Datenbank-Service-Komponenten
- „versteht“ den Datenstrom
- Verteilt Anfragen basierend auf der Kombination von
  - Definierten Algorithmen
  - Status der Komponenten
  - Inhalt der Anfragen
  - Session-Status

MaxScale Architektur



MaxScale Konfiguration

```
[maxscale]
threads=3
log_messages=off
log_debug=on
```

- Allgemeine Optionen in der “maxscale” Sektion der INI-Datei
- Anzahl Threads
- Aktivierung von Logging

- Außerhalb der Sektion “maxscale”
  - Sektions-Namen sind Objekt-Namen innerhalb von MaxScale
  - In jedem Bereich wird ein Objekt-Type definiert

```
[Test-Service]
type=service
router=readconnroute
router_options=slave
servers=server1,server2,server3,server4
user=massi
passwd=6628C50E07CCE1F0392EDEEB9D1203F3
weightby=cores
```

Name über den der Service referenziert werden kann (z.B. von einem Filter.

Sektionen für *server1*, *server2*, *server3* und *server4* werden getrennt in der Konfigurationsdatei definiert

Datenbank-Benutzer

- Zwei Arten von Nutzern werden benötigt
  - Monitor benötigt einen Nutzer

    `REPLICATION CLIENT on *.*`
  - Services benötigen einen Nutzer um Nutzer-Informationen und Datenbank-Namen zu ermitteln

    `SELECT on mysql.user, SELECT on mysql.db and SHOW DATABASES on *.*`
- Nutzer-Passwörter können unverschlüsselt oder verschlüsselt definiert werden
- Erstellung von Encryption Keys über „maxkeys“
- Erstellung von verschlüsselten Passwörtern über “maxpasswd“

Authentifizierung

- MariaDB verwendet Benutzer, Passwort und Host zur Authentifizierung
- MaxScale verwendet die gleiche Logik für eingehende Verbindungen
- MariaDB Backends verwenden Verbindungen von MaxScale, nicht die der Client-Adressen
- Alle MariaDB / MySQL Nutzer müssen sich vom MaxScale Host verbinden können und Berechtigungen müssen für den Benutzer des MaxScale Hosts definiert sein

Einschränkungen zur Authentifizierung

- Per Host Passwörter
  - Umgebungen in denen der gleiche Nutzer verschiedene Passwörter für unterschiedliche Hosts verwendet, werden nicht unterstützt
- Wildcards in Hostnamen
  - Wildcards werden nur in IP-Adressen unterstützt
- Alle Hosts in einem Cluster müssen einen gemeinsamen Pool von Nutzern verwenden
- Momentan wird nur der native MariaDB interne Authentifizierungsmechanismus unterstützt
  - Authentifizierungs-Plugins können nicht verwendet werden

End-to-End Konfiguration für einen Service

```
[ListenerNAme]
type=listener
service=<ServiceName>
protocol=MySQLClient
port=<MaxScale Port for listening>
```

```
[BackEndserverName]
type=server
address=<server-host-address>
port=<port-at-which-database-server-listens>
protocol=MySQLBackend
```

One for each server in cluster

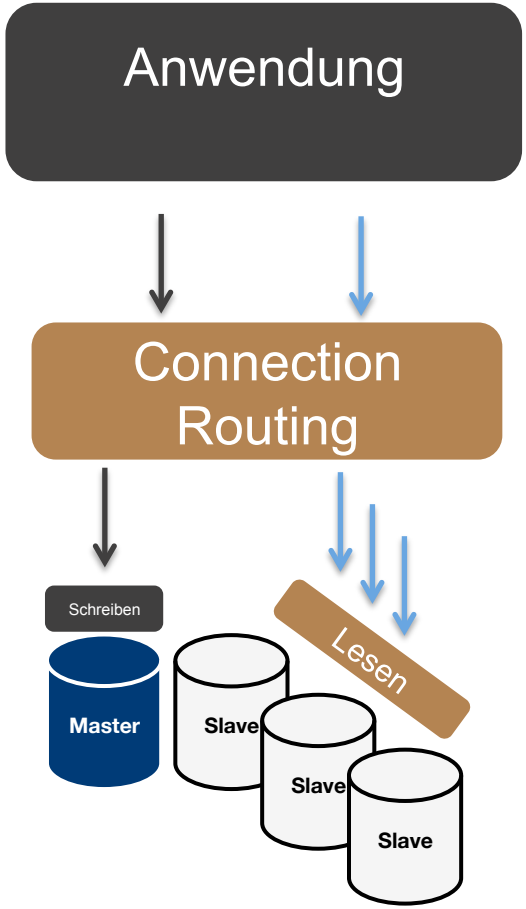
```
[<ServiceName>]
type=service
router=<router-module-name>
router_options=<router-options specific to router>
servers=<list of backend servers to route to srv1, srv2>
user=mper
passwd=6628C50E07CCE1F0392EDEEB9D1203F3
<service-specific-option>=<option-value>
Filter=<FilterName>
```

```
[FilterName]
type=filter
module=<filter-module-name>
<filter-specific-option>=<option-value>
```

```
[MonitorName]
type=monitor
module=<monitor-module-name>
servers=<list of servers in cluster to be monitored srv1, srv2>
user=mper
passwd=massi
monitor_interval=<value>
```

Traditionelles Load Balancing Master / Slave

- Verbindungs-Basiertes Routing
- Verteilung von Verbindungen über mehrere Server
  - Monitoring ermittelt Master und Slave
  - Gewichtung für Verbindung, wenn konfiguriert
  - Round Robin für Slaves
- Externes Failover erforderlich



Master-Slave Cluster Read & Write Services Configuration

Read Service

```
[ReadService]
type=service
router=readconnroute
router_options=slave
servers=server1,server2,server3,server4
user=mper
passwd=6628C50E07CCE1F0392EDEEB9D1203F3
```

Write Service

```
[WriteService]
type=service
router=readconnroute
router_options=master
servers=server1,server2,server3,server4
user=mper
passwd=6628C50E07CCE1F0392EDEEB9D1203F3
```

And network listener

```
[ReadListener]
type=listener
service=ReadService
protocol=MySQLClient
port=4006
```

And network listener

```
[WriteListener]
type=listener
service=WriteService
protocol=MySQLClient
port=4007
```

Client-Anwendungen verwenden die verschiedenen Listener-Ports und optinonales IP-Binding für ihre Lese- und Schreib-Verbindungen

Definition der Server

```
[ReadService]
type=service
router=readconnroute
router_options=slave
servers=server1,server2,server3,server4
user=mper
passwd=6628C50E07CCE1F0392EDEEB9D1203F3
```

- Definition Adresse/Hostname
- Port des Servers
- Das Protocol-Modul für die Verbindung

```
[server1]
type=server
address=127.0.0.1
port=3307
protocol=MySQLBackend
```

Entsprechende Konfiguration
für die restlichen Server
(server2,server3, server4)

- Optional kann der Monitor-Nutzer für den Server überschrieben werden

MariaDB (Master-Slave) Monitor Konfiguration

- Ein Monitor für beide Services
- Verwendung des “mysqlmon” Modules

```
[MySQL Monitor]
type=monitor
module=mysqlmon
servers=server1,server2,server3,server4
user=mper
passwd=massi
```

Gruppe von Servern, Monitoring als Cluster

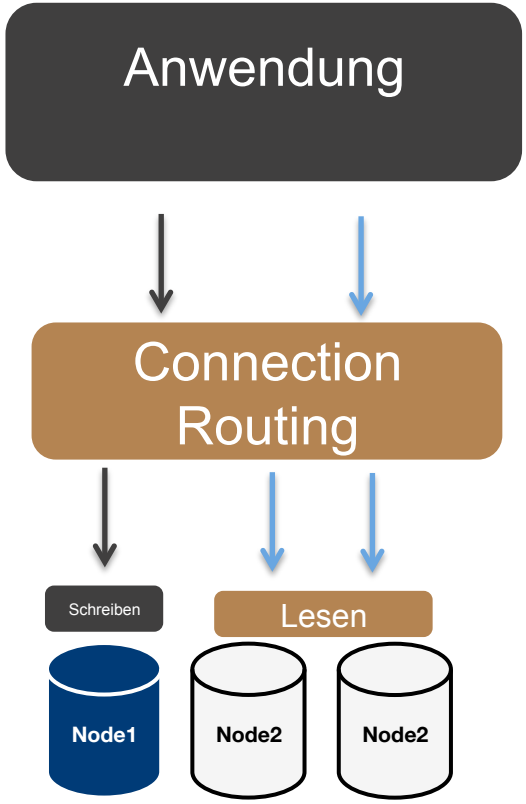
- Optionales Definieren des Monitoring-Intervalls (in Millisekunden)

```
monitor_interval=5000
```

Reduzierung des Monitor Intervalls erlaubt frühere Erkennung von Failover

Traditionelles Load Balancing Galera Cluster

- Verbindungs-Basiertes Routing
- Verteilung von Verbindungen über mehrere Server
  - Monitoring ermittelt „Master-Node“
  - Gewichtung für Verbindung, wenn konfiguriert
  - Round Robin für Slaves
- GaleraMon überwacht Cluster und definiert die Master-Node
- Kein externes Failover erforderlich
- Keine Schreib-Konflikte



Galera Cluster Read & Write Services Configuration

Read Service

```
[ReadService]
type=service
router=readconnroute
router_options=slave
servers=server1,server2,server3
user=mper
passwd=6628C50E07CCE1F0392EDEEB9D1203F3
```

Write Service

```
[WriteService]
type=service
router=readconnroute
router_options=master
servers=server1,server2,server3
user=mper
passwd=6628C50E07CCE1F0392EDEEB9D1203F3
```

And network listener

```
[ReadListener]
type=listener
service=ReadService
protocol=MySQLClient
port=4008
```

And network listener

```
[WriteListener]
type=listener
service=WriteService
protocol=MySQLClient
port=4009
```

Client-Anwendungen verwenden die verschiedenen Listener-Ports für ihre Lese- und Schreib-Verbindungen

Definition der Server

```
[ReadService]
type=service
router=readconnroute
router_options=slave
servers=server1,server2,server3
user=mper
passwd=6628C50E07CCE1F0392EDEEB9D1203F3
```

- Definition Adresse/Hostname
- Port des Servers
- Das Protocol-Modul für die Verbindung

```
[server1]
type=server
address=127.0.0.1
port=3307
protocol=MySQLBackend
```

Entsprechende Konfiguration für die restlichen Server (server2,server3)

- Optional kann der Monitor-Nutzer für den Server überschrieben werden

Galera Monitor Konfiguration

- Ein Monitor für beide Services
- Verwendung des “galeramon” Modules

```
[MySQL Monitor]
type=monitor
Module=galeramon
servers=server1,server2,server3
user=mper
passwd=massi
```

Gruppe von Servern in Galera Cluster

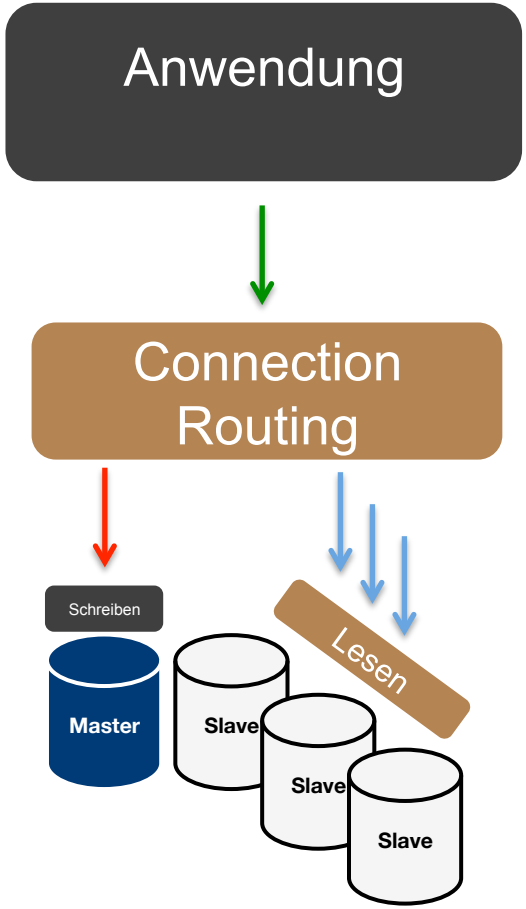
- Optionales Definieren des Monitoring-Intervalls (in Millisekunden)

```
monitor_interval=5000
```

Reduzierung des Monitor Intervalls erlaubt frühere Erkennung von Failover

Statement-Basiertes Routing Master / Slave

- Verteilung von Verbindungen über mehrere Server
  - Monitoring ermittelt Master und Slave
  - Gewichtung für Verbindung, wenn konfiguriert
  - Statement-Basiertes Routing ermittelt lesen, schreiben oder Session-Modifikationen
  - Round Robin für Slaves
- Externes Failover erforderlich



Master-Slave Cluster Read & Write Services Configuration

Read/Write Splitter Service

```
[SplitService]
type=service
router=readwritesplit
servers=server1,server2,server3,server4
user=mper
passwd=6628C50E07CCE1F0392EDEEB9D1203F3
router_option=slave_selection_criteria=LEA
ST_BEHIND_MASTER
```

And network listener

```
[SplitListener]
type=listener
service=SplitService
protocol=MySQLClient
port=4006
```

Client-Anwendungen verwenden die verschiedenen Listener-Ports für ihre Lese- und Schreib-Verbindungen

Read/Write Splitter Optionen

- Router Optionen für Slave Auswahlkriterium
  - Least Global Connections
    - LEAST_GLOBAL_CONNECTIONS
    - Slave mit minimaler Anzahl an Backend Verbindungen aller Services
  - Least Router Connections
    - LEAST_ROUTER_CONNECTIONS
    - Slave mit minimaler Anzahl Backend Verbindungen dieses Services
  - Least Current Operations - Default
    - LEAST_CURRENT_OPERATIONS
    - Slave mit minimaler Anzahl aktiver Anfragen

- Parameter für Slave Auswahlkriterium
- Maximale Anzahl an Slave Verbindungen
  - Maximale Anzahl Slaves für Lastverteilung
  - Definiert als absolute Zahl oder %

- Slaves sind Nodes die zum Lesen verwendet werden
- Nicht Slaves im Sinne eines Master / Slave Cluster

```
max_slave_connections=50%
```

Definition der Server

- Definition Adresse/Hostname
- Port des Servers
- Das Protocol-Modul für die Verbindung

```
[server1]
type=server
address=127.0.0.1
port=3307
protocol=MySQLBackend
```

Entsprechende Konfiguration
für die restlichen Server
(server2,server3, server4)

- Optional kann der Monitor-Nutzer für den Server überschrieben werden

MariaDB (Master-Slave) Monitor Konfiguration

- Optional Monitoring Slave Lag

```
detect_replication_lag=1
```

Nur für read/write-split Router für Master-Slave Cluster

- Ein Monitor für beide Services
- Verwendung des “mysqlmon” Modules

```
[MySQL Monitor]
type=monitor
module=mysqlmon
servers=server1,server2,server3,server4
user=mper
passwd=massi
```

Gruppe von Servern, Monitoring als Cluster

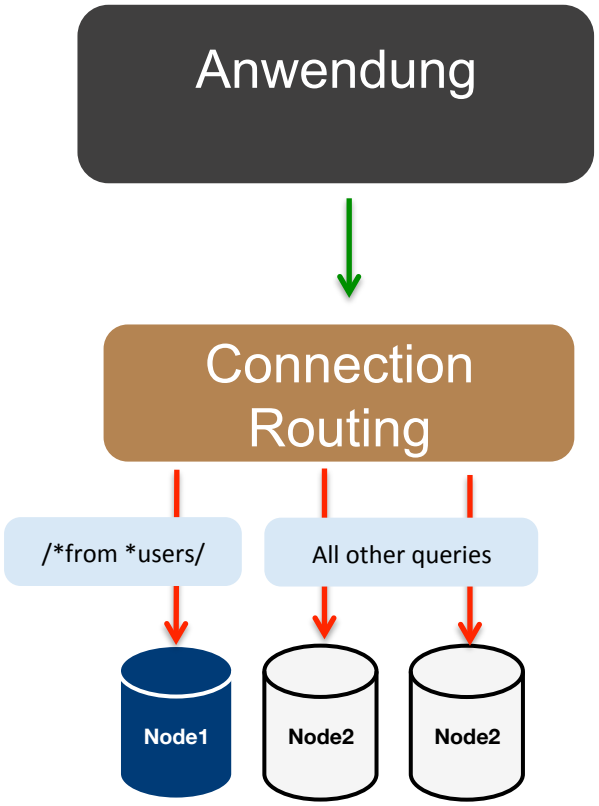
- Optionales Definieren des Monitoring-Intervalls (in Millisekunden)

```
monitor_interval=5000
```

Reduzierung des Monitor Intervalls erlaubt frühere Erkennung von Failover

Abfrageabhängige Lastverteilung

- Verbindungs-Basiertes Routing
- Verteilung von Verbindungen über mehrere Server
  - Monitoring ermittelt „Master-Node“
  - Gewichtung für Verbindung, wenn konfiguriert
  - Round Robin für Slaves
- GaleraMon überwacht Cluster und definiert die Master-Node
- Kein externes Failover erforderlich
- Keine Schreibkonflikte



Regex basiertes Routing von Anfragen

- Definieren des Named Server Filter mit regex und Ziel-Server

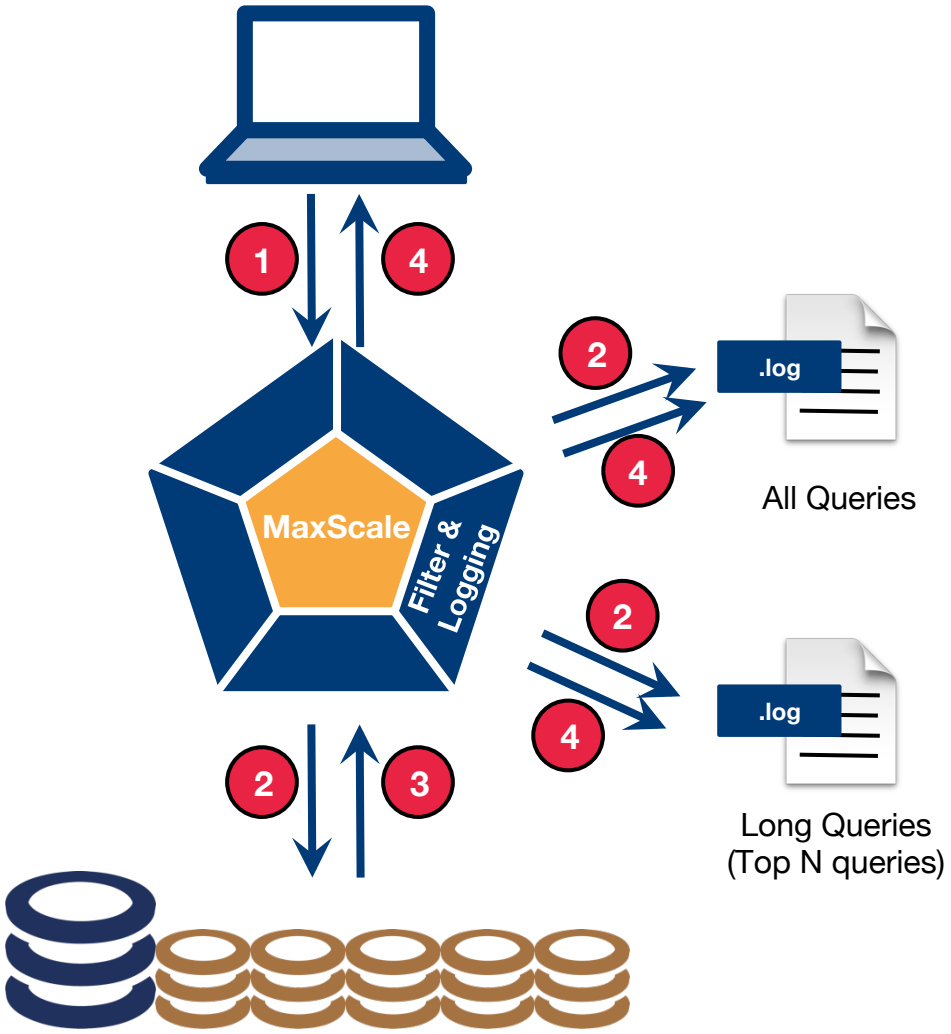
```
[NamedServer1]
type=filter
module=namedserverfilter
match= *from *users
options=ignorecase
server=server1
```

- Definieren von Service und Listener
- Filter hinzufügen zu Service

```
[SplitService]
type=service
router=readwritesplit
servers=server1,server2,server3
user=massi
passwd=6628C50E07CCE1F0392EDEEB9D1203F3
filter=NamedServer1
```

```
[SplitListener]
type=listener
service=SplitService
protocol=MySQLClient
port=4006
```

Query Logging: DBA



1. MaxScale akzeptiert Anfrage von Client Applikation
2. Weiterleitung zum Back-end
   - Log in “All Queries”
   - Log in “Long Running Queries”
3. Empfang des Ergebnisses vom back-end,
4. Weiterleitung des Ergebnisses zum Client
   - Ergebnis in “All Queries” Log
   - Wenn eine der N langsamsten Anfragen, Ergebnis in “Long Running Queries” log File

Weitere Anwendungsfälle

- Schema Sharding
- Anwendungsabhängiges Routing
- Binlog-Server
- Query-Blocking
- Query Transformation

- Reduziert die Komplexität und Risiken in verschiedensten Umgebungen

Vielen Dank!