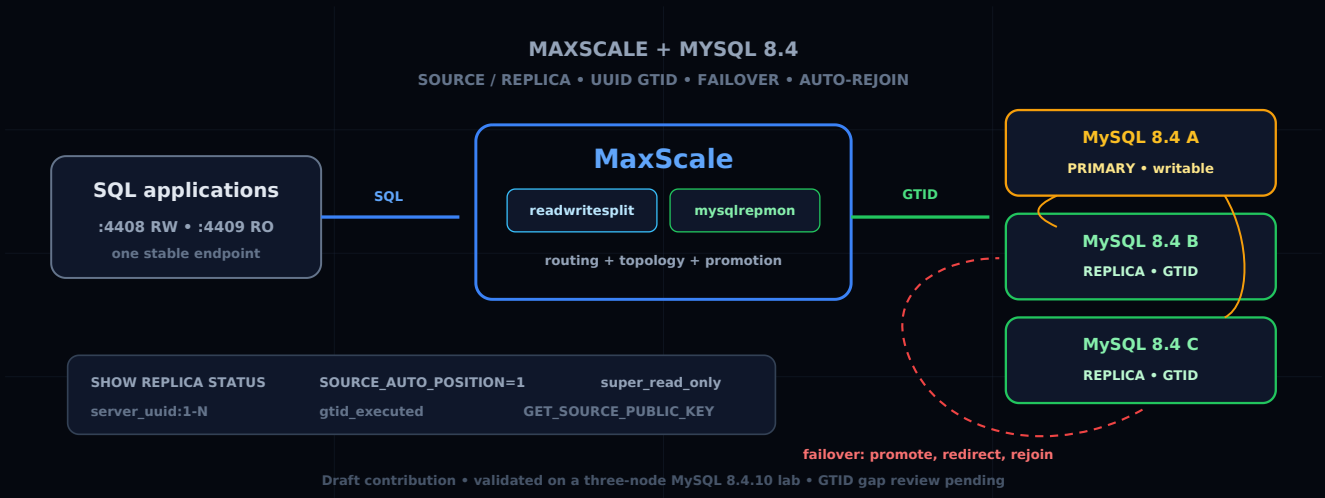
MaxScale и MySQL 8.4: GTID-failover, совместимый с Source/Replica

Aurélien LEQUOY · July 27, 2026

MAXSCALE MYSQL MYSQL-8.4 REPLICATION GTID FAILOVER HIGH-AVAILABILITY PROXY



> ***Статус проекта — 27 июля 2026 года.*** *Upstream PR #421 предлагает для MaxScale модуль `mysqlrepmon`, описанный в статье. PR намеренно оставлен в статусе draft: код собран и проверен в лаборатории MySQL 8.4.10, но ещё не входит в официальный выпуск MaxScale. Не следует представлять его как поддерживаемую производителем функцию для production.*

Проблема в одном предложении

Исторически MaxScale контролирует топологии репликации MariaDB / MySQL с помощью `mariadbmon`. Но MySQL 8.4 удалил прежние команды `SLAVE` / `MASTER`, использует GTID на основе UUID и требует другой синтаксис для перенастройки репликации.

Результат может быть обманчивым: прокси продолжает принимать SQL-соединения, хотя монитор больше не способен правильно прочитать топологию, повысить replica до primary или присоединить прежний primary.

Необходимо разделять три функции:

1. **наблюдать** топологию и состояние потоков репликации;
2. **изменять** топологию во время switchover, failover или rejoin;
3. **маршрутизировать** сеансы приложений на текущий primary и доступные replicas.

Совместимость с MySQL 8.4 должна быть корректной на всех трёх уровнях. Одной замены `SHOW SLAVE STATUS` на `SHOW REPLICA STATUS` недостаточно.

Почему MySQL 8.4 нарушает прежние предположения

MySQL 8.0.22 ввёл терминологию Source/Replica, сохранив несколько исторических псевдонимов. MySQL 8.4 завершает этот переход: устаревшие команды удалены.

Операция	MariaDB / прежний диалект	MySQL 8.4
Прочитать состояние replica	`SHOW SLAVE STATUS`	`SHOW REPLICA STATUS`
Настроить source	`CHANGE MASTER TO`	`CHANGE REPLICATION SOURCE TO`
Запустить репликацию	`START SLAVE`	`START REPLICA`
Остановить репликацию	`STOP SLAVE`	`STOP REPLICA`
Очистить конфигурацию	`RESET SLAVE`	`RESET REPLICA`
Прочитать состояние binlog	`SHOW MASTER STATUS`	`SHOW BINARY LOG STATUS`
Сбросить GTID	`RESET MASTER`	`RESET BINARY LOGS AND GTIDS`

Имена возвращаемых столбцов также изменились:

Старое имя	Имя в MySQL 8.4
`Master_Host`	`Source_Host`
`Master_Port`	`Source_Port`
`Master_Server_Id`	`Source_Server_Id`
`Slave_IO_Running`	`Replica_IO_Running`
`Slave_SQL_Running`	`Replica_SQL_Running`
`Seconds_Behind_Master`	`Seconds_Behind_Source`

Старое имя	Имя в MySQL 8.4
`Master_Log_File`	`Source_Log_File`
`Read_Master_Log_Pos`	`Read_Source_Log_Pos`
`Exec_Master_Log_Pos`	`Exec_Source_Log_Pos`
`Channel_Name`	`Channel_Name`

Монитор, который всё ещё отправляет `SHOW SLAVE STATUS`, немедленно получает синтаксическую ошибку. Монитор, отправляющий правильный запрос, но продолжающий искать `Slave_IO_Running`, ошибочно решит, что репликация остановлена.

Главное различие: модель GTID

Наиболее существенное отличие — не словарь SQL, а представление транзакций.

GTID MariaDB

MariaDB представляет GTID в форме:

```
domain_id-server_id-sequence
```

Пример:

```
0-101-7842
```

Домен является частью модели. Монитор MariaDB умеет сравнивать такие позиции и, помимо прочего, использует `MASTER_USE_GTID`.

GTID MySQL

MySQL представляет множество GTID через UUID серверов и интервалы:

```
155fa720-7623-11f1-9a78-bc241174cab8:1-76
```

История может содержать несколько UUID и разрывные интервалы:

```
155fa720-7623-11f1-9a78-bc241174cab8:1-10:20-30,
7fd8c42a-7630-11f1-99af-bc241174cab8:1-8
```

MySQL предоставляет, среди прочего:

- `@@global.server_uuid` для идентификации источника;
- `@@global.gtid_executed` для применённых транзакций;
- `@@global.gtid_purged` для GTID, binlog которых уже удалён;
- `Retrieved_Gtid_Set` в `SHOW REPLICA STATUS` для полученных транзакций.

При смене источника используется автоматическое позиционирование:

```
CHANGE REPLICATION SOURCE TO
    SOURCE_HOST = '10.0.0.12',
    SOURCE_PORT = 3306,
    SOURCE_USER = 'repl',
    SOURCE_PASSWORD = 'secret',
    SOURCE_AUTO_POSITION = 1,
    GET_SOURCE_PUBLIC_KEY = 1
FOR CHANNEL '';
```

Сервер сам выбирает правильную точку продолжения на основе множеств GTID, без указания файла и позиции binlog.

Зачем нужен модуль `mysqlrepmon`

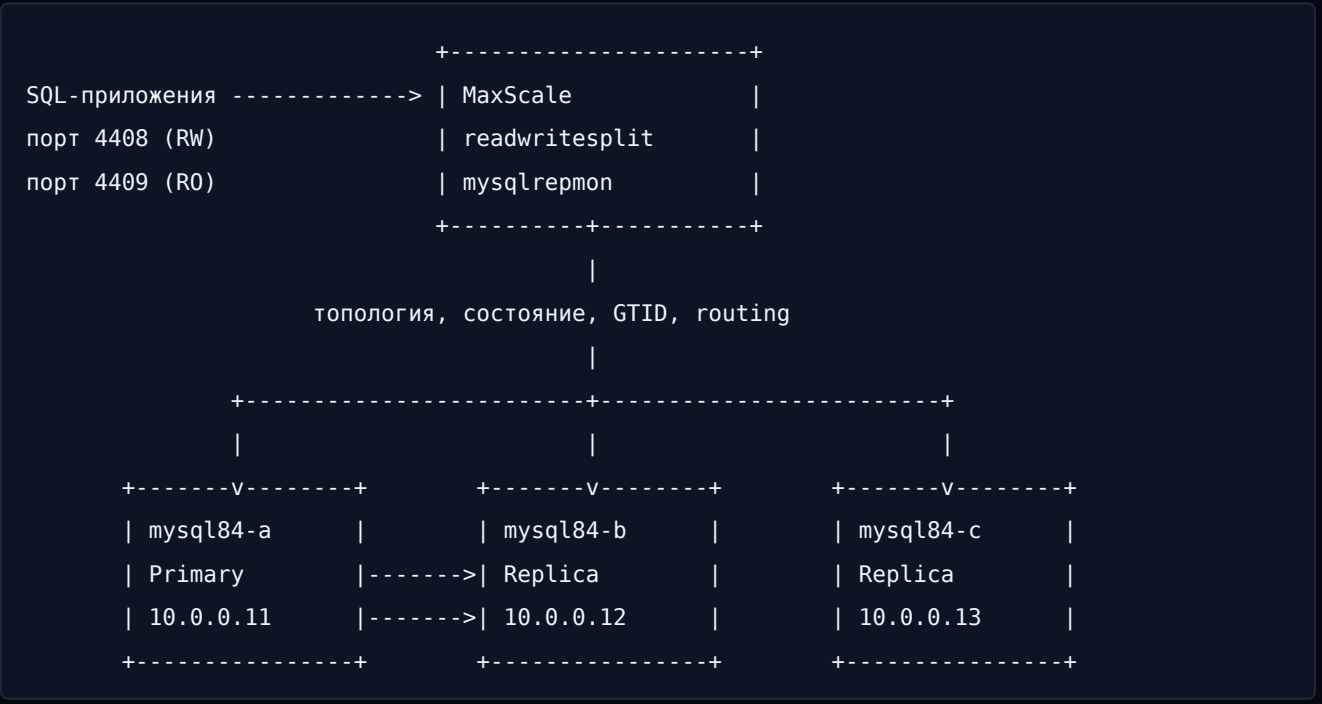
Рядом с `mariadbmon` предложение добавляет монитор, предназначенный для MySQL. Он повторно использует проверенный механизм принятия решений и управления кластером, но заменяет серверно-зависимые элементы:

- чтение `SHOW REPLICA STATUS` и столбцов `Source_*` / `Replica_*`;
- чтение `server_uuid`, `gtid_executed` и `gtid_purged`;
- использование `CHANGE REPLICATION SOURCE TO ... SOURCE_AUTO_POSITION=1`;
- команды `START`, `STOP` и `RESET REPLICA ... FOR CHANNEL`;
- включение `super_read_only=1` при понижении сервера;
- использование `RESET BINARY LOGS AND GTIDS` для ручной команды reset;
- проверку `enforce_gtid_consistency` и `log_replica_updates`.

Модуль сохраняет эксплуатационные параметры, знакомые по `mariadbmon`: `auto_failover`, `auto_rejoin`, `enforce_read_only_slaves`, а также команды switchover, failover и rejoin.

Эталонная архитектура

Проверенная архитектура включает три сервера MySQL 8.4 и один узел MaxScale:



Приложения никогда не знают адрес primary. Они используют listener MaxScale. Монитор решает, какой сервер получает внутреннюю роль `Master` в MaxScale; `readwritesplit` направляет записи на этот сервер и распределяет чтения.

> *Сам MaxScale тоже должен быть отказоустойчивым. Идеальный failover баз данных бесполезен, если единственный SQL-прокси остаётся единой точкой отказа. В production предусмотрите как минимум два узла MaxScale в режиме active/passive, VIP или load balancer и подходящий механизм кооперативных блокировок.*

1. Подготовка каждого сервера MySQL 8.4

Каждый сервер, который может быть повышен, должен создавать собственные binlog и записывать в них полученные транзакции.

Базовая конфигурация:

```
[mysqld]
server_id=101
```

```
log_bin=mysql-bin
binlog_format=ROW

gtid_mode=ON
enforce_gtid_consistency=ON
log_replica_updates=ON

relay_log_recovery=ON
```

На каждом узле задайте отдельный `server_id`, например `101`, `102`, `103`.

`server_uuid` также обязан быть уникальным. Он хранится в файле `auto.cnf` каталога данных. Если машина или datadir клонируются, нельзя запускать несколько серверов с одинаковым `server_uuid`.

На replicas:

```
read_only=ON
super_read_only=ON
```

На primary:

```
read_only=OFF
super_read_only=OFF
```

После перезапуска проверьте инварианты:

```
SELECT
    @@hostname,
    @@server_id,
    @@server_uuid,
    @@gtid_mode,
    @@enforce_gtid_consistency,
    @@log_bin,
    @@log_replica_updates,
    @@read_only,
    @@super_read_only\G

SELECT @@global.gtid_executed\G
SHOW BINARY LOG STATUS\G
SHOW REPLICA STATUS\G
```

Replica, подходящая для повышения, должна иметь `log_bin=1`, `log_replica_updates=1`, оба работающих потока репликации и контролируемое отставание.

2. Создание отдельных учётных записей

Как минимум разделите:

- учётную запись мониторинга и изменения топологии;
- учётную запись, с помощью которой replicas читают binlog;
- учётные записи приложений, проходящие через прокси.

Учётная запись монитора

Только для наблюдения:

```
CREATE USER 'maxscale_mon'@'10.0.0.10'
    IDENTIFIED BY 'a-long-random-password';

GRANT REPLICATION CLIENT
    ON *.* TO 'maxscale_mon'@'10.0.0.10';
```

Для failover, switchover и rejoin монитор также должен останавливать и перенастраивать репликацию, менять read-only переменные и управлять соединениями:

```
GRANT REPLICATION SLAVE,
      REPLICATION CLIENT,
      PROCESS,
      SUPER,
      SYSTEM_VARIABLES_ADMIN,
      REPLICATION_SLAVE_ADMIN,
      CONNECTION_ADMIN
    ON *.* TO 'maxscale_mon'@'10.0.0.10';
```

Историческая привилегия `SUPER` всё ещё требуется некоторыми путями совместимости. В финальной версии модуля нужно повторно проверить точный список и удалить ненужные права.

Учётная запись репликации

```
CREATE USER 'repl'@'10.0.0.%'
    IDENTIFIED BY 'another-long-random-password'
    REQUIRE SSL;

GRANT REPLICATION SLAVE
    ON *.* TO 'repl'@'10.0.0.%';
```

Предпочитайте TLS. При стандартной для MySQL 8.4 аутентификации `caching_sha2_password` через незашифрованное соединение команда смены источника должна указать `GET_SOURCE_PUBLIC_KEY=1` либо путь к открытому ключу RSA. Модуль добавляет эту опцию, если TLS не включён.

Сервисная учётная запись для аутентификации клиентов

Параметр `user` сервиса MaxScale — не та учётная запись, от имени которой выполняются все запросы приложений. Он позволяет User Account Manager в MaxScale загрузить учётные записи и права с backend-серверов:

```
CREATE USER 'maxscale_route'@'10.0.0.10'
    IDENTIFIED BY 'route-password';

GRANT SELECT ON mysql.user
    TO 'maxscale_route'@'10.0.0.10';
GRANT SELECT ON mysql.db
    TO 'maxscale_route'@'10.0.0.10';
GRANT SELECT ON mysql.tables_priv
    TO 'maxscale_route'@'10.0.0.10';
GRANT SELECT ON mysql.columns_priv
    TO 'maxscale_route'@'10.0.0.10';
GRANT SELECT ON mysql.procs_priv
    TO 'maxscale_route'@'10.0.0.10';
GRANT SELECT ON mysql.proxies_priv
    TO 'maxscale_route'@'10.0.0.10';
GRANT SHOW DATABASES ON *.*
    TO 'maxscale_route'@'10.0.0.10';
```

Учётные записи приложений всё равно должны существовать на серверах MySQL с одинаковым паролем и согласованными grant. Поскольку backend видит соединение, пришедшее от MaxScale, часть `@host` должна разрешать адрес прокси.

3. Сборка вклада по воспроизводимому SHA

Поскольку PR пока остаётся draft, стандартный пакет MaxScale не содержит `mysqlrepmon`. Только для лаборатории соберите точный публичный SHA:

```
git clone https://github.com/mariadb-corporation/MaxScale.git
cd MaxScale

git fetch https://github.com/Esysteme/MaxScale.git \
  aurelien/mysql-8.4-replication-support
git checkout --detach f61776a5b19cf295636c94a8a3dea6d81fcd61fb

cmake -S . -B build \
  -DCMAKE_BUILD_TYPE=Release \
  -DCMAKE_INSTALL_PREFIX=/usr/local/maxscale-mysql84

cmake --build build --target mariadbmon mysqlrepmon test_cycle_find -j2
ctest --test-dir build --output-on-failure \
  -R '^test_mariadbmon_cycle_find$'

cmake --build build -j2
sudo cmake --install build
```

Целевая сборка должна создать `libmysqlrepmon.so`. Исправление CMake в предложении явно использует обнаруженные MaxScale значения `LIBSSH_LIBRARY` и `LIBSSH_INCLUDE_DIR`, а не предполагает наличие внутренней цели `libssh`.

Запуск изолированной сборки

Префикс `/usr/local/maxscale-mysql84` не позволяет перезаписать установленный пакет. Однако пакетный `maxscale.service` по-прежнему запускает `/usr/bin/maxscale` и не загрузит новый модуль. Для лаборатории создайте `/etc/systemd/system/maxscale-mysql84.service`:

```
[Unit]
Description=MaxScale MySQL 8.4 compiled build
After=network-online.target
Wants=network-online.target

[Service]
Type=simple
```

```
User=maxscale
Group=maxscale
RuntimeDirectory=maxscale
RuntimeDirectoryMode=0755
ExecStart=/usr/local/maxscale-mysql84/bin/maxscale --nodaemon --config=/etc/maxscale.cnf --
logdir=/var/log/maxscale --datadir=/var/lib/maxscale --cachedir=/var/cache/maxscale --
libdir=/usr/local/maxscale-mysql84/lib/maxscale --piddir=/run/maxscale
Restart=on-failure
RestartSec=5s
LimitNOFILE=65535

[Install]
WantedBy=multi-user.target
```

Системная учётная запись `maxscale` и каталоги данных должны существовать; обычно их создаёт установленный пакет MaxScale, предоставляющий среду выполнения. Затем перечитайте конфигурацию systemd:

```
sudo install -d -o maxscale -g maxscale -m 0755 \
  /var/lib/maxscale /var/cache/maxscale /var/log/maxscale
sudo systemctl daemon-reload
```

4. Настройка MaxScale

Полный минимальный пример:

```
[maxscale]
threads=auto
admin_host=127.0.0.1
admin_port=8989

[mysql84-a]
type=server
address=10.0.0.11
port=3306
protocol=MariaDBBackend

[mysql84-b]
type=server
address=10.0.0.12
```

```
port=3306
protocol=MariaDBBackend

[mysql84-c]
type=server
address=10.0.0.13
port=3306
protocol=MariaDBBackend

[MySQL84-Monitor]
type=monitor
module=mysqlrepmon
servers=mysql84-a,mysql84-b,mysql84-c
user=maxscale_mon
password=a-long-random-password
monitor_interval=2000ms

assume_unique_hostnames=true
auto_failover=safe
auto_rejoin=true
enforce_read_only_slaves=true

replication_user=repl
replication_password=another-long-random-password
replication_master_ssl=true

[MySQL84-RW-Service]
type=service
router=readwritesplit
servers=mysql84-a,mysql84-b,mysql84-c
user=maxscale_route
password=route-password

master_reconnection=true
master_failure_mode=fail_on_write

[MySQL84-RW-Listener]
type=listener
service=MySQL84-RW-Service
protocol=MariaDBClient
port=4408
```

```
[MySQL84-RO-Service]
type=service
router=readconnroute
servers=mysql84-a,mysql84-b,mysql84-c
user=maxscale_route
password=route-password
router_options=slave

[MySQL84-RO-Listener]
type=listener
service=MySQL84-RO-Service
protocol=MariaDBClient
port=4409
```

Пароли оставлены открытыми лишь для читаемости примера. В эксплуатации используйте шифрование секретов MaxScale, строгие права на конфигурационный файл и документированную ротацию.

`assume_unique_hostnames=true` требуется для включения `auto_failover` и `auto_rejoin`. Поэтому каждый backend должен иметь уникальный стабильный адрес или hostname, совпадающий с source, записанным replicas; если сеть репликации использует другой адрес, задайте `private_address`.

Почему сначала `auto_failover=safe`? Этот режим отказывается от операции, если монитор видит, что повышение явно приведёт к потере транзакций. Он не превращает асинхронную репликацию в синхронную, но создаёт полезный защитный барьер во время проверки.

Почему `master_reconnection=true` и `master_failure_mode=fail_on_write`? Сеанс `readwritesplit` может сохранить контекст и подключиться к новому primary, пока в окне без primary не поступила запись и нет открытой транзакции. Без этих параметров failover базы данных может пройти успешно, но все сеансы приложений всё равно оборвутся.

5. Проверка перед первым тестом

Проверьте конфигурацию и запустите MaxScale:

```
sudo /usr/local/maxscale-mysql84/bin/maxscale \
  --config=/etc/maxscale.cnf \
```

```
  --libdir=/usr/local/maxscale-mysql84/lib/maxscale \
  --config-check

sudo systemctl disable --now maxscale.service 2>/dev/null || true
sudo systemctl enable --now maxscale-mysql84.service
systemctl --no-pager --full status maxscale-mysql84.service
```

Убедитесь, что модуль и топология видимы:

```
maxctrl list modules | grep -E 'mariadbmon|mysqlrepmon'
maxctrl list servers
maxctrl list monitors
maxctrl show monitor MySQL84-Monitor
```

Ожидаемое состояние:

- один сервер `Master, Running`;
- два сервера `Slave, Running`;
- ни одного сервера `Down`;
- активный монитор;
- открытые listeners `4408` и `4409`.

Проверьте MySQL и напрямую:

```
mysql -h 10.0.0.12 -e "SHOW REPLICA STATUS\\G"
mysql -h 10.0.0.13 -e "SHOW REPLICA STATUS\\G"
```

Ключевые поля:

```
Replica_IO_Running: Yes
Replica_SQL_Running: Yes
Seconds_Behind_Source: 0
Auto_Position: 1
```

6. Как проходит failover

Когда primary исчезает, монитор не просто меняет метку:

1. ожидает число циклов, заданное `failcount`, и насколько возможно подтверждает реальность сбоя;
2. сравнивает состояние replicas и выбирает кандидата для повышения;
3. останавливает репликацию на кандидате;
4. отключает `read_only` на новом primary;
5. перенаправляет остальные replicas через `CHANGE REPLICATION SOURCE TO ... SOURCE_AUTO_POSITION=1`;
6. перезапускает их потоки репликации;
7. `readwritesplit` обнаруживает новую роль и направляет следующие записи на новый primary.

Для понижения primary `mysqlrepmon` использует:

```
SET GLOBAL super_read_only = 1;
```

Это сильнее, чем `read_only=1`: даже учётная запись с административными правами не должна случайно продолжать запись на прежний primary.

7. Проверка контролируемого switchover

Перед имитацией жёсткого сбоя проверьте switchover:

```
maxctrl call command mysqlrepmon switchover \
  MySQL84-Monitor mysql84-b mysql84-a
```

Затем проверьте:

```
maxctrl list servers

mysql -h 10.0.0.11 -e \
  "SELECT @@hostname, @@read_only, @@super_read_only, @@global.gtid_executed\\G"
mysql -h 10.0.0.12 -e \
  "SELECT @@hostname, @@read_only, @@super_read_only, @@global.gtid_executed\\G"
mysql -h 10.0.0.13 -e \
  "SELECT @@hostname, @@read_only, @@super_read_only, @@global.gtid_executed\\G"
```

Новый primary должен принимать записи. Два других узла должны иметь `super_read_only=1` и реплицироваться с него.

8. Проверка настоящего сбоя без упрощений

Сначала создайте данные через MaxScale:

```
CREATE DATABASE maxscale_ha_test;
CREATE TABLE maxscale_ha_test.events (
    id BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
    source_hostname VARCHAR(255) NOT NULL,
    created_at TIMESTAMP(6) NOT NULL DEFAULT CURRENT_TIMESTAMP(6),
    PRIMARY KEY (id)
) ENGINE=InnoDB;

INSERT INTO maxscale_ha_test.events(source_hostname)
VALUES (@@hostname);
```

Убедитесь, что они появились на replicas, затем остановите primary на уровне службы или машины. Не переводите сервер только в maintenance в MaxScale: это проверит административное решение, а не обнаружение сбоя.

Во время теста:

```
watch -n 1 'maxctrl list servers'
```

В другом терминале:

```
journalctl -u maxscale -f
```

Критерии успеха:

- выбран ровно один новый primary;
- остальные replicas указывают на него;
- запись через listener `4408` проходит после повышения;
- чтение через `4409` остаётся согласованным;
- прежний primary после возвращения присоединяется как replica;
- множества `gtid_executed` сходятся.

Что проверено в лаборатории MySQL 8.4.10

Сценарий выполнен на трёх узлах MySQL 8.4.10:

- первоначальное обнаружение primary и двух replicas;
- остановка primary;
- повышение одной replica;
- перенаправление оставшейся replica на новый source;
- возвращение прежнего primary и автоматический rejoin в роли replica;
- второй failover в обратном направлении;
- итоговое совпадение одного множества `gtid_executed` на всех трёх серверах;
- корректная маршрутизация чтения и записи через MaxScale;
- отказ записей через read-only listener.

Тест подтверждает функциональный путь для непрерывных историй GTID. Он ещё не доказывает корректность всех граничных случаев, необходимых для upstream-интеграции.

Аутентификация: не путайте две неисправности

Сборка MaxScale в лаборатории отклоняла учётные записи приложений с хешем `caching_sha2_password` MySQL 8.4:

```
Stored password hash length is 70 when 40 was expected
```

Это сообщение приходит от authenticator протокола MaxScale в данной сборке, а не от монитора репликации. Репликация и маршрутизация могут быть исправны, даже если аутентификация клиента не работает.

В лаборатории отдельная probe-учётная запись с `mysql_native_password` позволила изолировать проблему. Это не общая рекомендация: MySQL 8.4 по умолчанию отключает этот исторический plugin. В production следует использовать сборку и authenticator MaxScale, совместимые с `caching_sha2_password`, либо официально поддерживаемый метод аутентификации, а не глобально ослаблять конфигурацию MySQL.

То же правило относится к:

```
401 Unauthorized
```

на REST-порту: это означает, что `maxctrl` не получил правильные административные credentials. Это не доказывает сбой монитора или SQL-маршрутизации.

Таблица диагностики

Симптом	Вероятная причина	Проверка
Синтаксическая ошибка на `SHOW SLAVE STATUS`	старый модуль или диалект MariaDB используется с MySQL 8.4	лог MaxScale, фактически отправленный запрос
Сервер показан без роли replica	столбцы `Replica_*` не сопоставлены	прямой `SHOW REPLICA STATUS\G`
Replica не может быть повышена	отключены `log_bin`, `log_replica_updates` или GTID	глобальные переменные и лог монитора
Rejoin отклонён	errant-транзакции или несовместимая история	сравнить `gtid_executed` / `gtid_purged`
`Stored password hash length is 70...`	authenticator несовместим с `caching_sha2_password`	лог MaxScale, plugin учётной записи
`401 Unauthorized` в `maxctrl`	неверные REST-credentials	административная конфигурация MaxScale
Сеанс оборван несмотря на повышение	router не может переподключиться, открыта транзакция или исчерпана история команд сеанса	`master_reconnection`, `master_failure_mode`, лог router
Два доступных для записи primary	недостаточный fencing/read-only либо конкурирующие мониторы	`super_read_only`, кооперативные блокировки, состояние сети

Известное ограничение: разрывы во множествах GTID

Чтобы повторно использовать внутренний движок `mariadbmon`, предложение сопоставляет каждый UUID MySQL с синтетическим доменом и преобразует его интервалы в

существующее внутреннее представление.

В текущем состоянии сохраняется максимальный номер транзакции для каждого UUID. Поэтому:

```
uuid:1-10:20-30
```

сводится к позиции, как будто она достигла `30`. Информация о том, что транзакции с 11 по 19 отсутствуют, передаётся неточно.

В лаборатории истории были непрерывными. В инфраструктуре с внедрёнными, отфильтрованными, очищенными или errant GTID это приближение может исказить сравнение кандидатов.

Это главная причина статуса draft. До признания модуля готовым к production необходимо:

- корректно представлять или сравнивать разрывные интервалы;
- добавить unit-тесты с несколькими UUID и разрывами;
- покрыть errant GTID и очищенные истории;
- выполнить полную upstream CI;
- получить review maintainer'ов MaxScale.

Чего failover не гарантирует

Монитор не отменяет свойства асинхронной репликации.

- **Нулевой RPO не гарантирован**: транзакция, подтверждённая прежним primary, могла ещё не попасть на replica.
- **Открытые транзакции**: сеанс внутри транзакции не всегда можно переместить без ошибки.
- **Split-brain**: если прежний primary доступен части приложений, требуется настоящее сетевое или системное fencing.
- **Сложные топологии**: multi-source, циклическая репликация и relay требуют отдельной стратегии.
- **Расходящиеся данные**: `auto_rejoin` не должен молча перезаписывать errant-транзакции.

- **Единственный прокси**: отказоустойчивость MaxScale обеспечивается отдельно.

Правильный тест высокой доступности измеряет четыре разные вещи: обнаружение, повышение, сходимость данных и непрерывность приложения.

Checklist перед production

- [ ] проверенная восстанавливаемая резервная копия;
- [ ] три уникальных значения `server_id` и `server_uuid`;
- [ ] GTID и `log_replica_updates` включены везде;
- [ ] TLS между MaxScale и backend-серверами;
- [ ] отдельные учётные записи монитора, репликации и приложений;
- [ ] `super_read_only=1` проверен на каждой replica;
- [ ] контролируемый switchover проверен до жёсткого failover;
- [ ] тест записи через MaxScale после повышения;
- [ ] тест rejoin прежнего primary;
- [ ] итоговое сравнение множеств `gtid_executed`;
- [ ] поведение транзакций приложений документировано;
- [ ] fencing и отказоустойчивость MaxScale протестированы;
- [ ] оповещение при каждом повышении и расхождении GTID;
- [ ] upstream-модуль завершён, проверен и поддерживается до внедрения в production.

Зачем публиковать этот вклад

MySQL 8.4 — LTS-выпуск. Исторические команды не вернутся. Бесконечная поддержка псевдонимов в средствах мониторинга не решает различие моделей GTID и синтаксиса перенастройки.

Отдельный модуль явно проводит границу:

- `mariadbmon` остаётся нативным для диалекта и GTID MariaDB;
- `mysqlrepmon` содержит правила, специфичные для MySQL 8.x;
- общий механизм failover остаётся совместным;
- тесты могут проверять каждое семейство без множества разбросанных условий.

PR MaxScale #421 содержит код, документацию, команды проверки, результаты лаборатории и известное ограничение GTID. Цель draft как раз в том, чтобы получить review этого представления, прежде чем утверждать, что любая история MySQL обрабатывается безопасно.

Дополнительные материалы

- MySQL 8.4 — новые возможности и удалённые команды репликации
- MySQL 8.4 — настройка репликации
- MySQL 8.4 — смена source и `GET_SOURCE_PUBLIC_KEY`
- MaxScale — параметры MariaDB Monitor
- MaxScale — router readwritesplit
- Установка MySQL 8.4 на Debian 13
- Переход от Master/Slave к Source/Replica

Хотите увидеть эту поддержку в MaxScale?

Если совместимость с MySQL 8.4 полезна для вас, ознакомьтесь с предложением, протестируйте ветку и поддержите PR. Отзывы из реальных инфраструктур, сложные случаи GTID и технические review помогут превратить этот draft во вклад, который действительно можно интегрировать:

Поддержать PR MaxScale #421