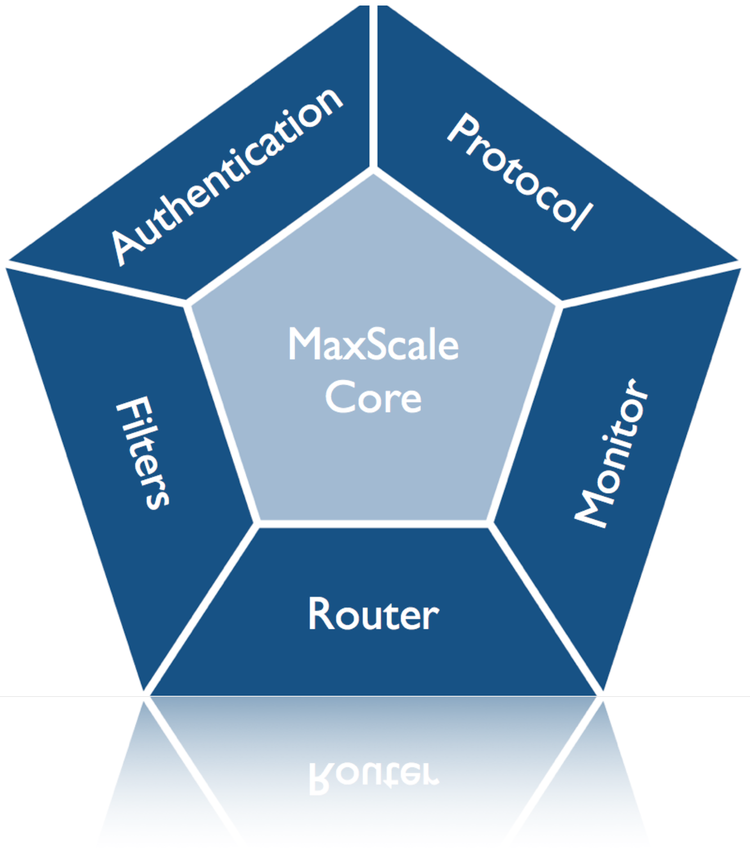
MaxScale

Binlog Router Internal Overview

Mark Riddoch



MySQL Replication Overview

Replication Overview

- Slaves connect to Masters in the same way as normal clients
- Standard MySQL protocol
- Some additional messages (COM_* commands)
- Binary logs are sent as responses to commands

Message Flow

- Slave Connects to master
- Slave issues a series of standard SQL requests to obtain configuration
- Slave registers with the master - COM_SLAVE_REGISTER message
- Slave requests binlog events - COM_BINLOG_DUMP
- Master replies with stream of events

Slave SQL Requests

- SELECT UNIX_TIMESTAMP();
- SELECT @master_binlog_checksum;
- SELECT @@GLOBAL.GTID_MODE;
  - MySQL 5.6 only
- SELECT VERSION();
- SELECT 1;
- SELECT @@version_comment limit 1;

Slave Set Requests

- SET @master_binlog_checksum = @@global.binlog_checksum
- SET @master_heartbeat_period=...
- SET @slave_slave_uuid=...
- SET NAMES latin1
  - May be different if slave is configured differently (e.g. utf8)

Slave Show Commands

- SHOW VARIABLES LIKE 'SERVER_ID'
- SHOW VARIABLES LIKE 'SERVER_UUID'

Slave Registration

- COM_SLAVE_REGISTER message
- Registers slave server-id, port and rank
- Optionally passes other data
  - hostname, username, password
- Server-id should be unique

Binlog Dump

- COM_BINLOG_DUMP message
- Requests a particular binlog file from a position
- Initiates process of master sending back binlog events
  - First event sent is a synthetic rotate event
    - Rotates to specified file and position
  - Send event send is a FORMAT_DESCRIPTION_EVENT
    - Describes how further events will be sent

Event Stream

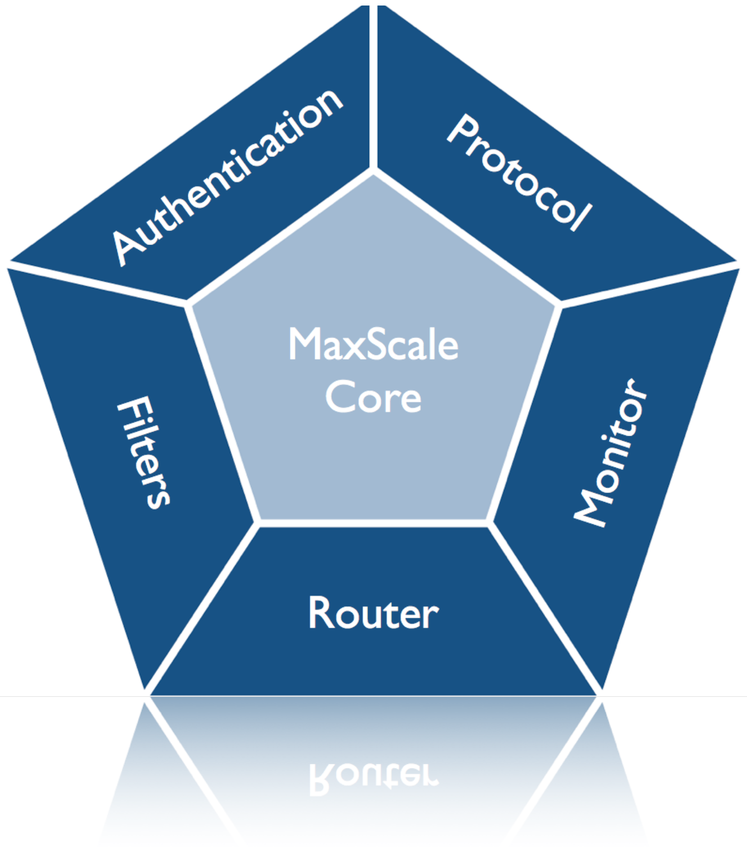
- The master sends events from the binlog file
- Events sent are a binary representation of the binlog file contents with a header prepended
- Once all events have been sent the master pauses until new events are ready to be sent

Rotate Event

- Special event sent at end of file
- Closes current file and moves to next file
  - New filename and position given in event
- Not all files end with rotate event

‘Fake’ Events

- Not all events originate form Master’s binlog file
- Rotate events and format description events
- These should not be saved to binlog file in the receiver



Implementation

Concept

- MaxScale connects to master as if it was a MySQL Slave
- MaxScale caches binlog events sent by master
- Slaves connect to MaxScale and request binlogs
- MaxScale appears to slaves as if it was the master

MaxScale Modules

- New router to manage replication
- Same Client/Backend Protocols
- No monitor used
- Filter interface not implemented

Binlog Router - Backend

- The binlog router has a single backend server
- The backend server is actually the master server
- Unlike other routers connection to backend occurs before client connections

Binlog Router - Clients

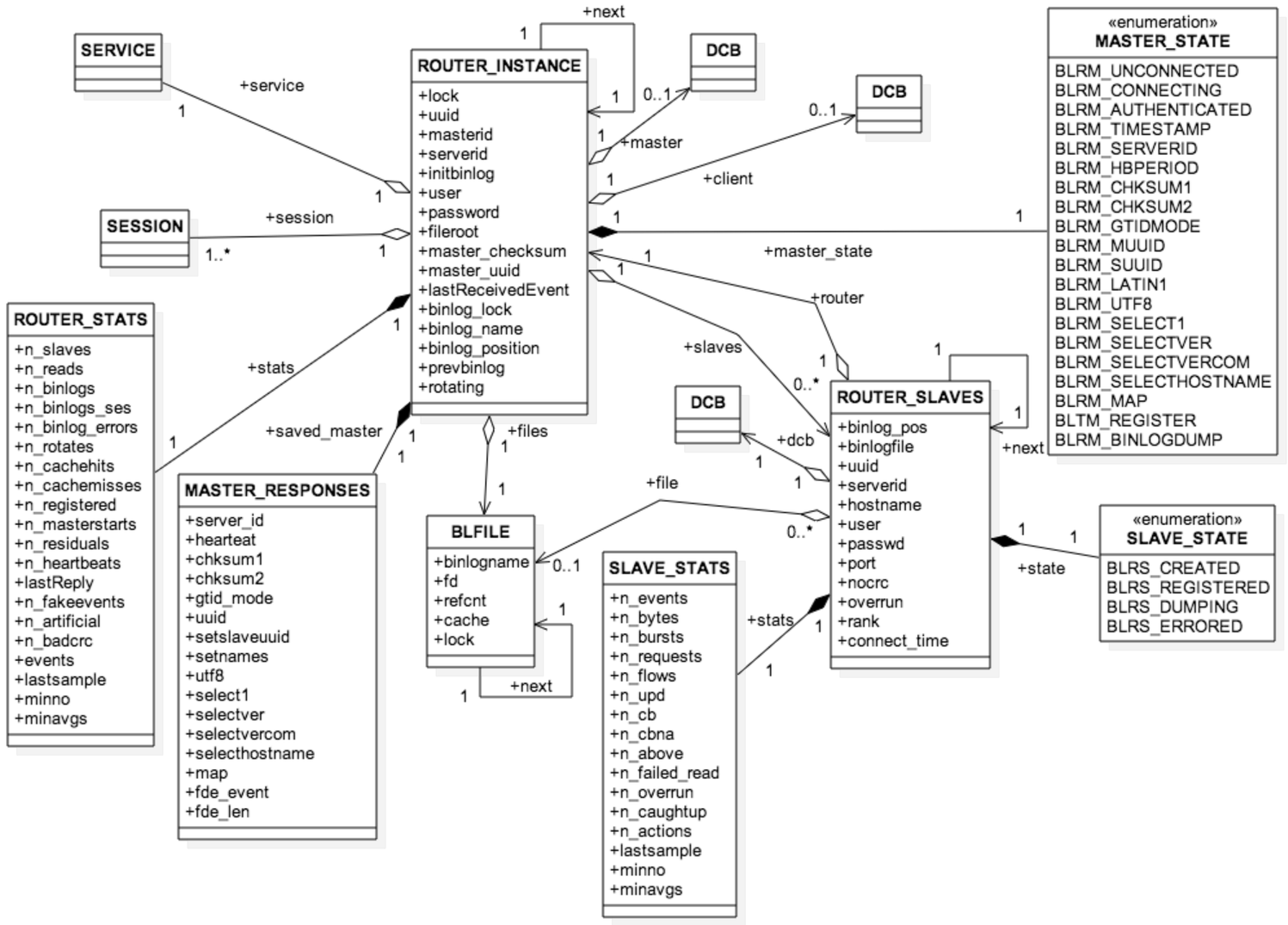
- The clients of the binlog router are the slaves
- Multiple slave connections
- Slave requests are not routed to the backend
- MaxScale creates responses from cached information previously received from the backend (master)

Binlog Router Code

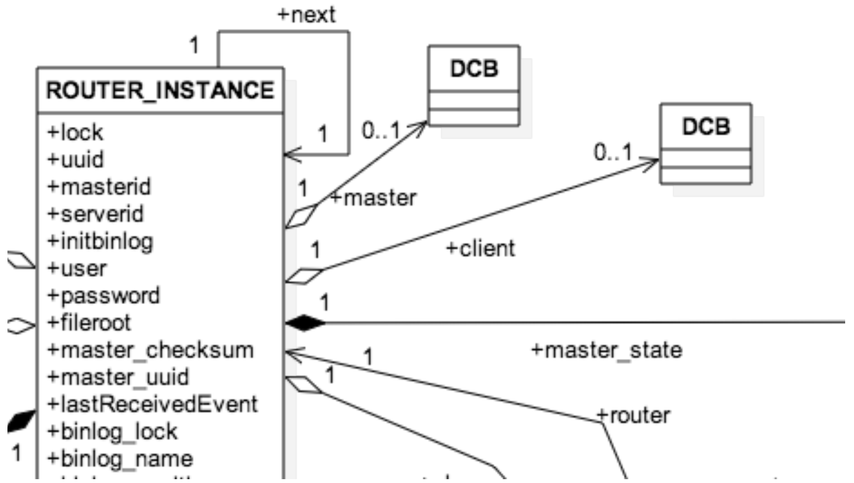
- MaxScale/modules/routing/binlog
  - blr.c - Module interface and entry points
  - blr_master.c - Interaction with master server
  - blr_slave.c - Interaction with the slave servers
  - blr_file.c - All file I/O related functions
  - blr_cache.c - Memory cache for binlog events (not currently in use)
- Header file MaxScale/Modules/include/blr.h

Binlog Router Structure

- ROUTER_INSTANCE is the state of the master connection
- ROUTER_SLAVE is the router session structure
- Pseudo session has NULL session



Router Startup



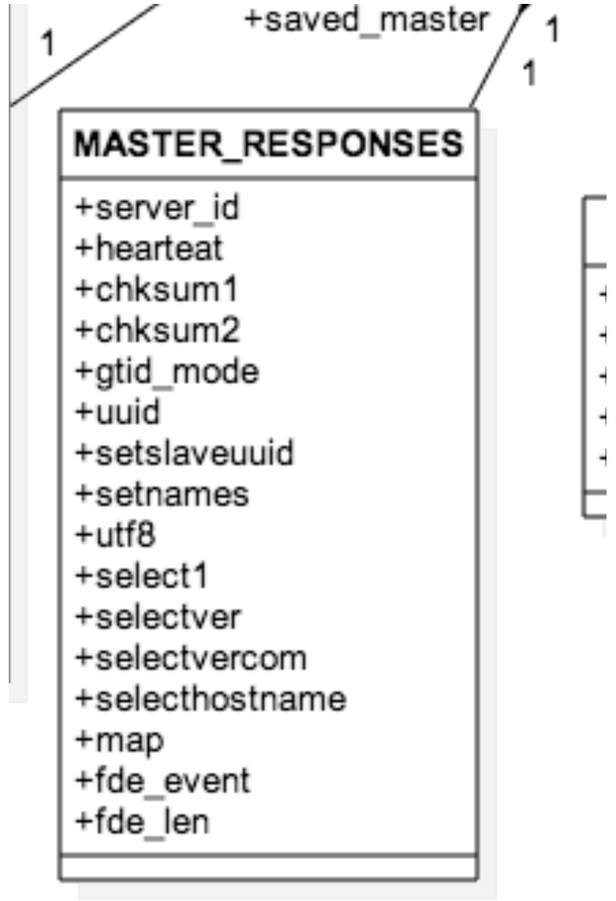
- Need connection to master on startup
- Creates fake client in blr_start_master routine - called from createInstance entry point
- Creates authentication information internally - no clients to copy from
- Router instance client dcb is the fake client
- Initiates state machine used to obtain master information

Master State Machine

- Master state machine is mechanism to send set of SQL requests to master
- Allows non-blocking semantics to be maintained
- Set of select, show etc commands
- Responses are cached in the MASTER_RESPONSES structure for replay to the slaves
- If master connection fails MASTER_RESPONSES are loaded from cache of previous values

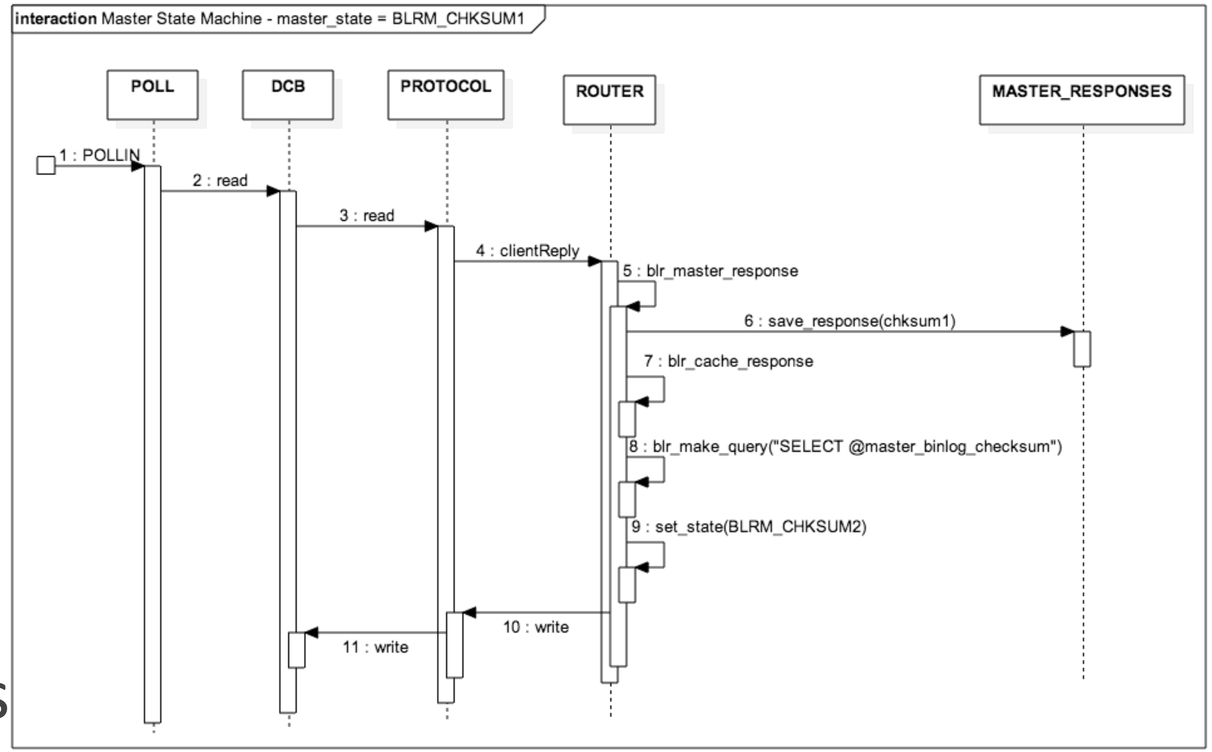
Master State Machine - Statements

- Set of commands sent
  - SELECT UNIX_TIMESTAMP()
  - SHOW VARIABLES LIKE ‘SERVER_ID’
  - SET @master_heartbeat_period = …
  - SET @master_binlog_checksum = @@global.binlog_checksum
  - SELECT @master_binlog_checksum
  - SELECT @@global.gtid_mode
  - SHOW VARIABLES LIKE ‘SERVER_UUID’
  - SET @slave_uuid = …
  - SET NAMES latin1
  - SET NAMES outfit
  - SELECT 1
  - SELECT VERSION()
  - SELECT @@version_comment LIMIT 1
  - SELECT @@hostname
  - SELECT @@max_allowed_packet
- Slave Register
- Binlog Dump



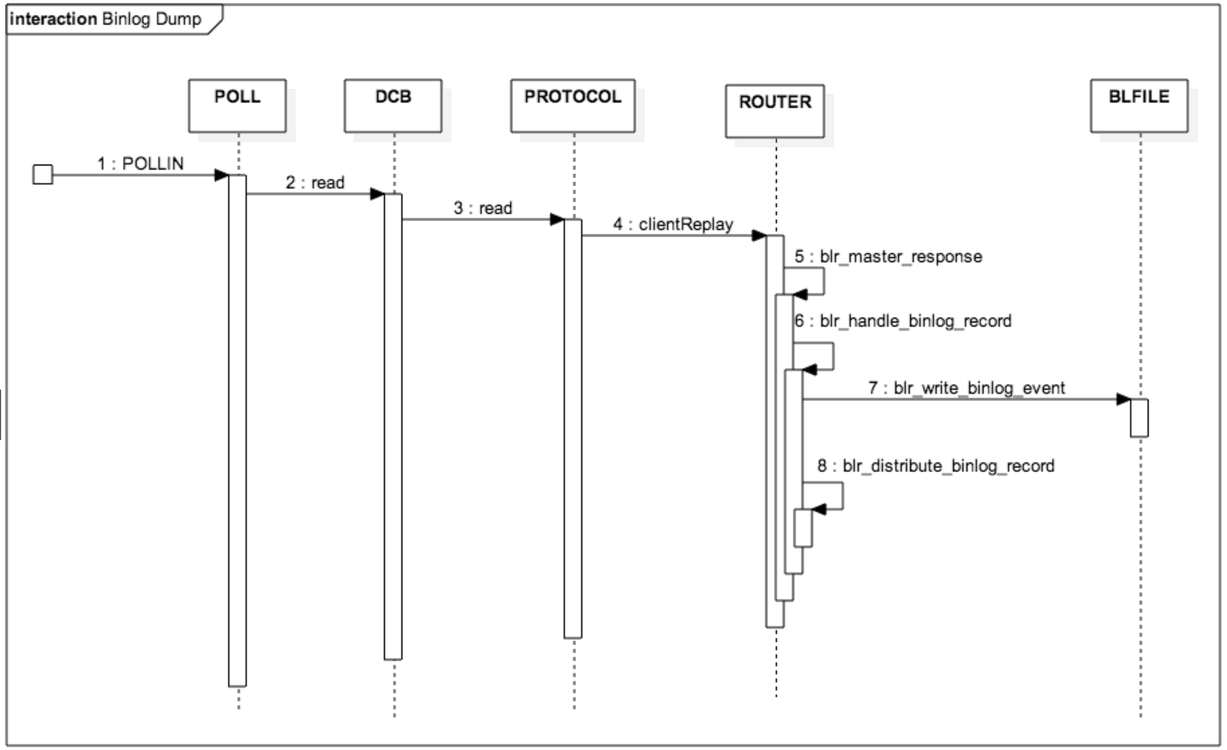
Master State Machine Implementation

- Assume current state BLRM_CHKSUM1
- State machine triggered on packet arrival
- Responses stored internally for later use
- Written to cache file for later invocations



MSM - Steady State

- "Steady State" of Master State Machine is BLRM_BINLOGDUMP
- Packets from master are BINLOG Events
- Events always written to disk
- If clients are registered events distributed

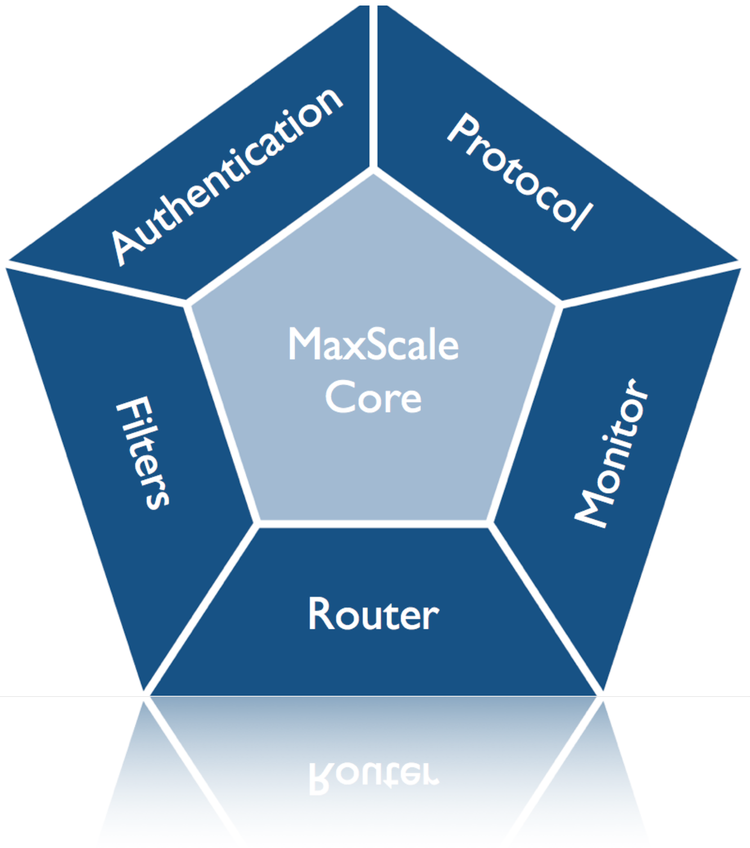


Binlog Event Packets

- Routine blr_handle_binlog_record is called with GWBUF that contains many packets
  - router->residual holds any unprocessed data from previous packet
  - Routine prepends any residual
  - Loops over each event making them contiguous if straddles buffers
  - Each event extracted and handled separately
  - Only copies data if need to make contiguous
  - On return may leave residual data
- Events flushed to disk at the end of each packet

Special Binlog Events

- Rotate Events
  - Written to file
  - Close file
  - Open next file
  - Distributed to slaves
- Heartbeat Events
  - Not written to file
  - Not distributed to slaves
- Events LOG_EVENT_ARTIFICAL_F bit set
  - Generated (fake) event
  - Not written to disk

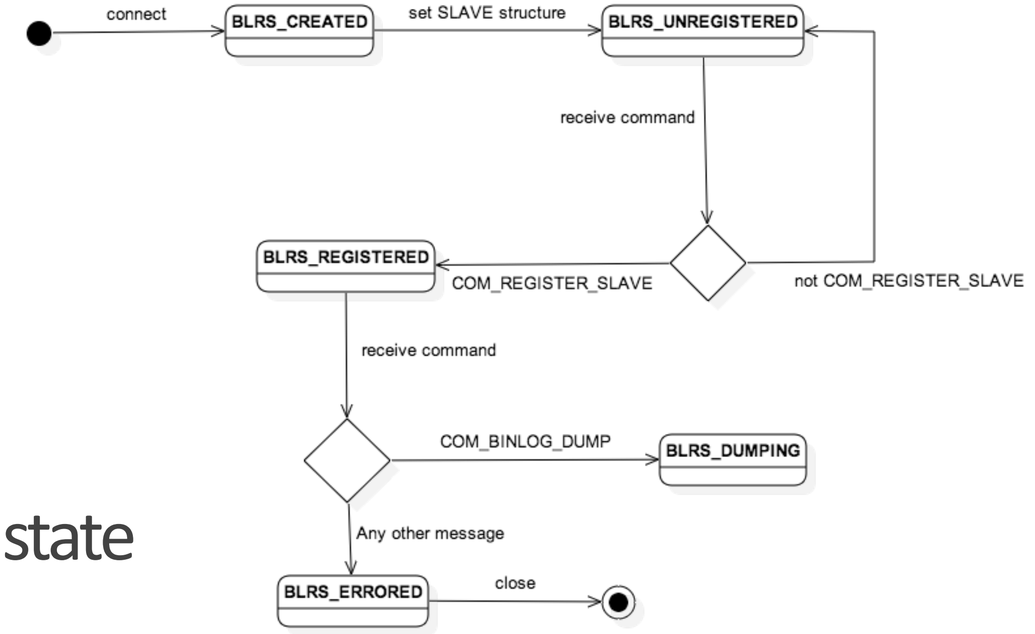

Authentication
Protocol
MaxScale
Core
Filters
Monitor
Router

Slave Side

Slave Connection

- Slaves connection to MaxScale using same mechanism as any client
- Authentication is done at the protocol level
- Slave sends commands to router
- Router parses and responds from MASTER_RESPONSES cache

Slave State Machine



- New slaves enter the slave state machine
- No binlog events are sent until BLRS_DUMPING state
- In BLRS_UNREGISTERED state all responses generated from saved master responses

Slave Modes

- Once BLRS_DUMPING state achieved slaves may be in two modes
  - Catchup mode
    - Slave is lagging behind the master and receiving events from the saved binlog files on the MaxScale server
  - Up-to-date mode
    - Slave has received all binlog events in the MaxScale binlog files
    - New events will be forwarded to the slave as they arrive at the MaxScale binlog router

Catchup Mode

- Sending all events required to cause slave to catchup in a single call would block a MaxScale thread
- There are no new messages sent from slave to master (MaxScale)
- Solution:
  - Send at most “burstsize” binlog events in a single call
  - Insert fake POLLOUT event for the slave
  - Results in future thread continuing process to bring slave up to date
- Implemented in blr_slave_catchup function

Up-to-date Mode

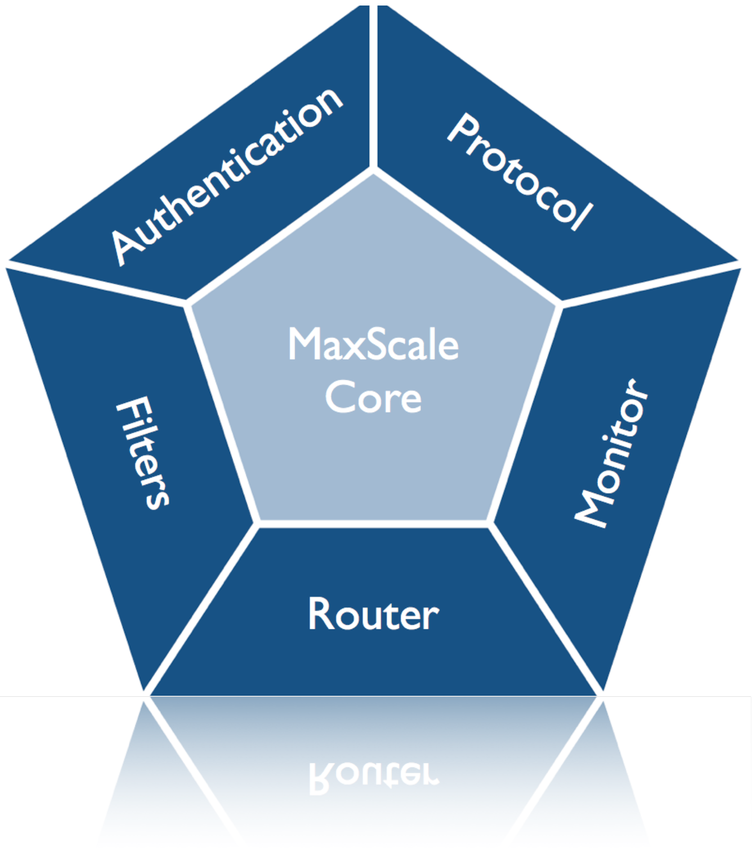
- Binlog events are sent as they are received
- Implemented in blr_distribute_binlog_record
  - Loops over all slaves
  - Sends single event to each slave that is up-to-date
- Events written to local binlog directory before sending to slaves

Slave Mode Transition

- Vital that transition for catchup to up-to-date mode is secure
- Must stop any new logs being sent during transition
- Two stage locking process used
  - First hold binlog_lock in the router
  - Then take slave catch_lock
  - **Never** do this in the opposite order or deadlock might occur

End of File Handling

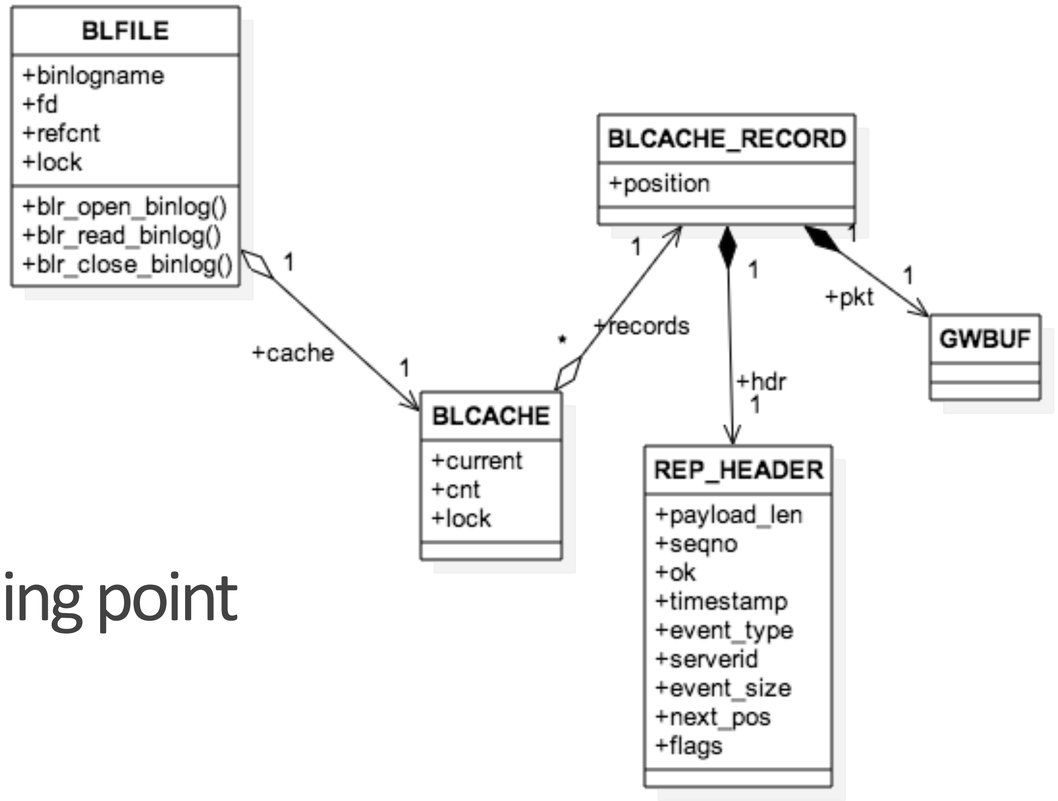
- If slave reaches end of file 2 things may have happened
  - The slave is up to date and streaming of binlog records can pause
  - The master crashed at some point in history and the rotate event at the end of the file is missing
- Detection
  - If current file is index N does file N + 1 exist
- If the binlog file is missing a trailing rotate event then a fake rotate event is generated


Authentication
Protocol
MaxScale
Core
Filters
Monitor
Router

File I/O

Shared Binlog File Handles

- BLFILE used as shared binlog file handles
- Shared between master and multiple slaves
- Allows shared descriptors and common caching point

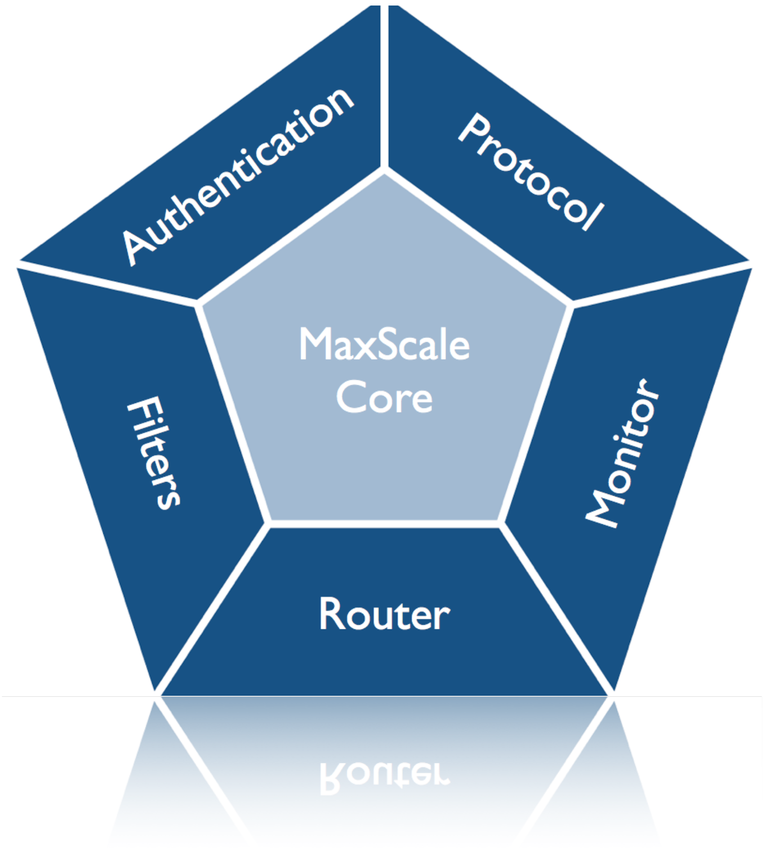


Binlog Storage

- By default binlog files stored in $MAXSCALE_HOME/<service name>
- Binlog files have standard binlog header (4 bytes)
- Remainder of binlog files are raw events as per the master
- Only difference is no close marker written to binlog file
- MaxScale does not support binlog index file

Cached Master Responses

- Router caches responses from master during handshake to file
- Allows router to read these cached responses if master is down when MaxScale starts
- If master is down MaxScale can serve binlog records it has previously cached
- Cached responses stored in $MAXSCALE_HOME/<service name>/.cache
- User credentials also stored to allow authentication without a master



Monitoring Binlog Server

MaxAdmin

- Uses standard router diagnostic entry point for show service command
- Provides details stats for master and slaves

MySQL Protocol Support

- As a side effect of router responding to saved commands these can detect MaxScale status
- COM_PING response to allow mysqladmin ping to determine if MaxScale is up
- COM_STATISTICS - allows basic MaxScale statistics to be returned
  - uptime
  - No. of threads
  - No. of binlog events sent
  - No. of slaves connect
  - Master State Machine state

MySQL Protocol Support (Contd)

- SELECT commands
  - SELECT @@maxscale_version
  - SELECT @@hostname
  - SELECT @@server_id
- SHOW commands
  - SHOW MASTER STATUS
  - SHOW SLAVE STATUS
  - SHOW SLAVE HOSTS
  - SHOW VARIABLES LIKE ‘MAXSCALE%’