

Galera Replication



Galera Replication for MySQL

Seppo Jaakola
Alexey Yurchenko

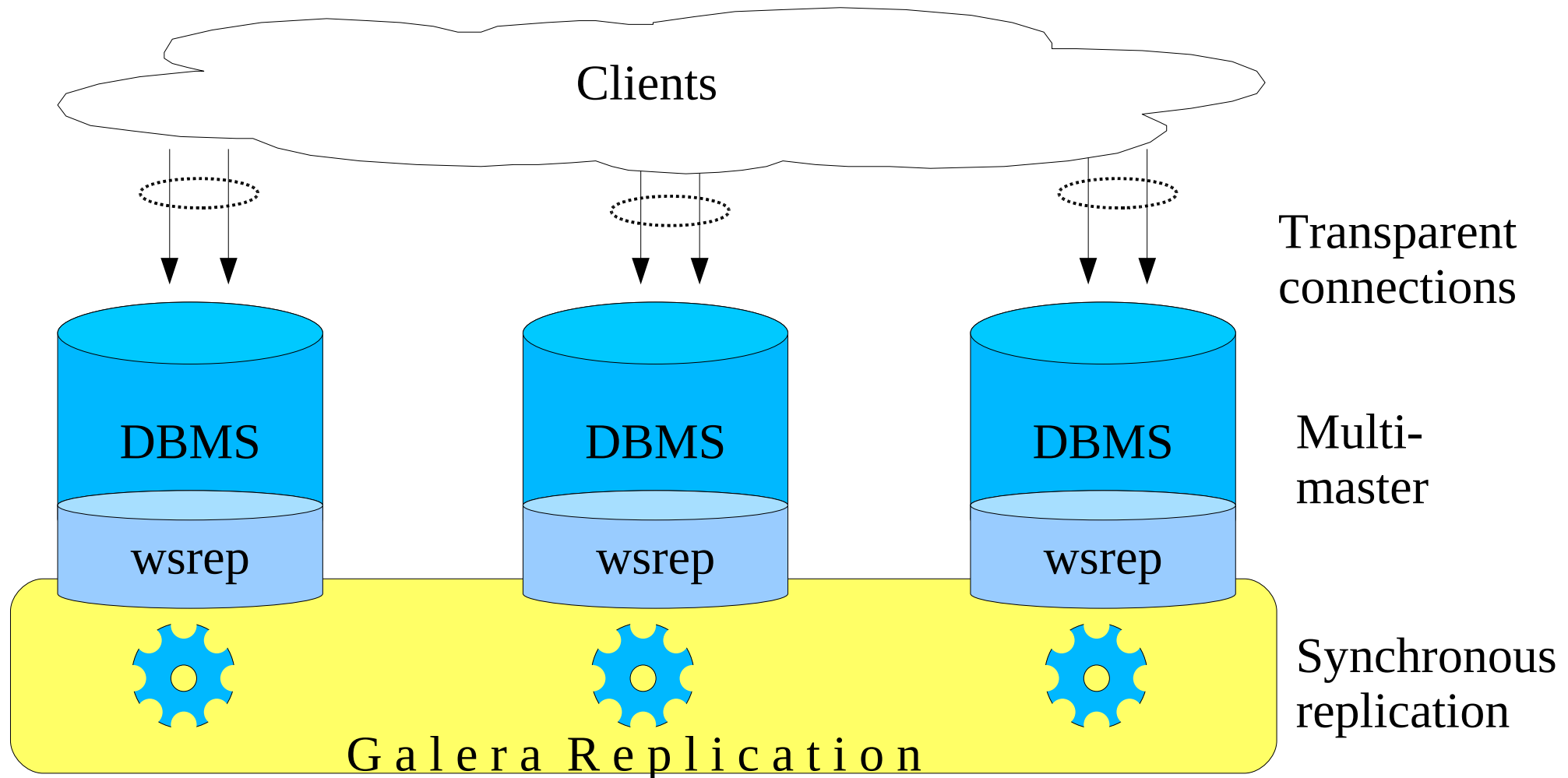
www.codership.com

Contents

1. Galera Overview
2. Replication API
3. MySQL Patch Overview
4. Advanced Features
 1. Connectivity
 2. State Transfer
5. Summary

Galera Replication

Galera Cluster



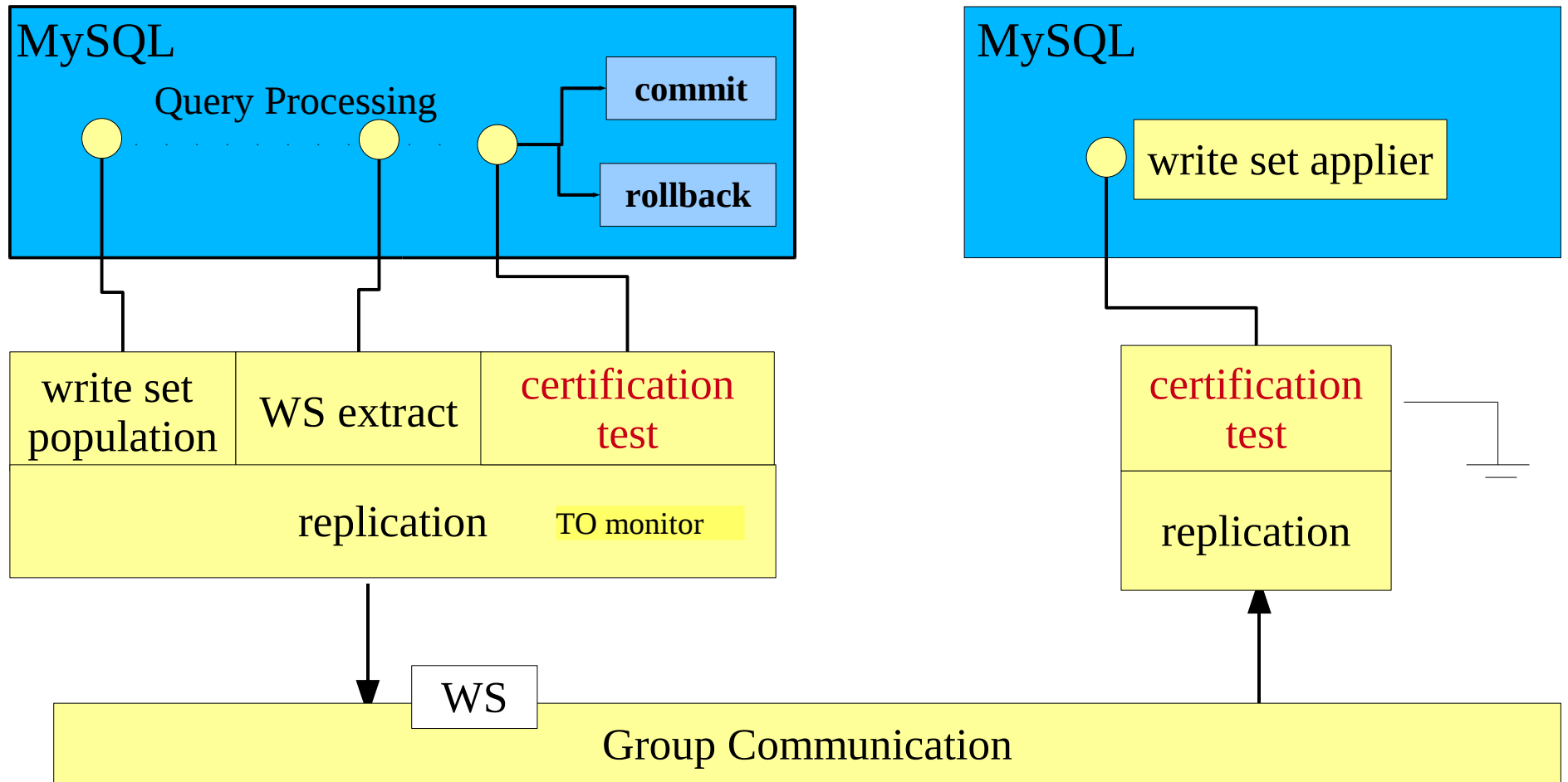
Galera Replication

- Synchronous multi-master replication
 - High Availability
- No middle-ware, connections directly to DBMS
 - Transparency
- RBR events, row level locking
 - Write scalability
- Certification based replication method

Galera Replication

- Galera is pluggable software library
- Generic replication system to make a cluster from any transactional DBMS
- First implementation MySQL/InnoDB cluster

Certification Based Replication



Write Set

Seqno
last_committed_seqno
Keys
Applying info

- Sequence number of the trx
- Sequence number of the last committed trx, which affected the processing state
- All row changes are identified with keys
- SQL statements or RBR events

Write Set

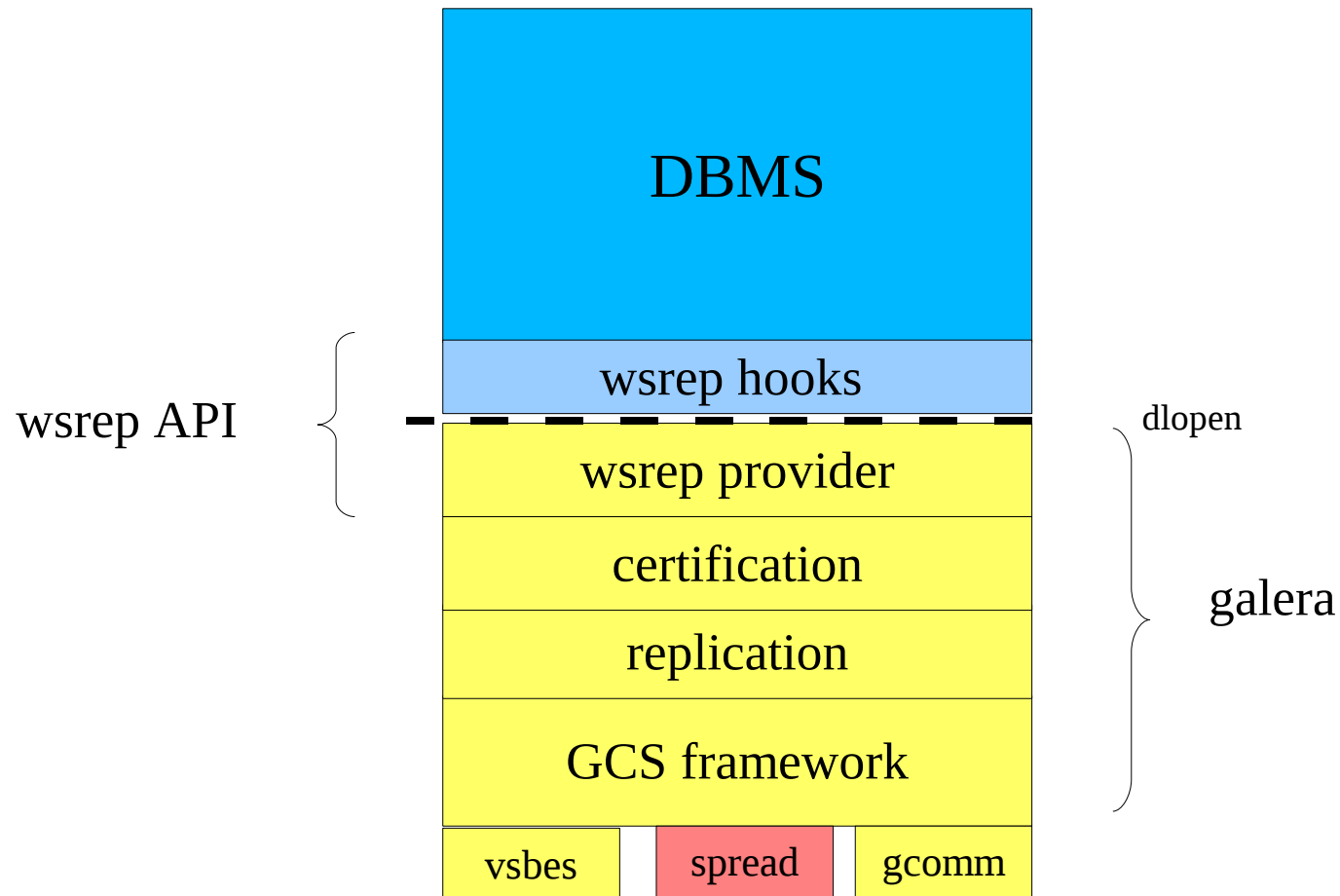
- By default write sets are flushed asap
- However, key information is stored in certification index
- Write sets can also be saved for future needs:
 - `wsrep_ws_persistency = ON/OFF`
 - For debugging purposes
 - For incremental state transfer

Replication API

Replication API

- Galera integrates closely in DBMS transaction processing
- There must be an interface between DBMS and replication system

DBMS and Replication Provider





wsrep_api.h

- Replication provider interface for DBMS
 - Defines types, structures
 - Global Transaction ID
 - Group membership
 - Methods (**DBMS calls provider library**)
 - Callbacks (**provider library calls DBMS**)

wsrep_api.h

- Initialization, connecting

```
int wsrep_load(const char* spec, wsrep_t** hptr, wsrep_log_cb_t log_cb);  
  
wsrep_status_t (*init) (wsrep_t*, const struct wsrep_init_args* args)  
wsrep_status_t (*connect) (...);  
wsrep_status_t (*disconnect) (...)
```

- SST Operations

```
typedef ssize_t (*wsrep_sst_prepare_cb_t) (void** msg);  
typedef int (*wsrep_sst_donate_cb_t) (...)  
  
wsrep_status_t (*sst_sent)(wsrep_t*,...)  
wsrep_status_t (*sst_received)(wsrep_t*,const wsrep_uuid_t* uuid,
```

wsrep_api.h

- Write Set Population

```
wsrep_status_t (*append_query)(wsrep_t*,  
wsrep_status_t (*append_row_key)(wsrep_t*,
```

- Transaction Processing

```
wsrep_status_t (*pre_commit)(wsrep_t*,...  
wsrep_status_t (*post_commit)(wsrep_t*,...
```

```
wsrep_status_t (*post_rollback)(wsrep_t*,...  
wsrep_status_t (*abort_pre_commit)(wsrep_t*,...
```

- Slave Applying

```
wsrep_status_t (*recv)(wsrep_t*, void* ctx);  
typedef enum wsrep_status (*wsrep_bf_apply_cb_t)(void* ctx, ...
```

wsrep_api.h

- DDL Processing

```
wsrep_status_t (*to_execute_start)(wsrep_t*, const void* query,...  
wsrep_status_t (*to_execute_end)(wsrep_t*,...
```

- Group Management

```
typedef struct wsrep_view_info {...  
typedef void (*wsrep_view_cb_t) (wsrep_view_info_t* view);
```

- Status, Configuration

```
struct wsrep_status_var* (*status_get) (wsrep_t*);  
wsrep_status_t (*options_set) (wsrep_t*, const char* conf);  
char* (*options_get) (wsrep_t*);
```



wsrep_api.h

- Provider library is loaded during DBMS startup
- Loading fills **provider handle** (methods, callbacks)
- Project: <https://launchpad.net/wsrep>
- API is “maturing” gradually, pending blueprints for new features
- Current head version is #15

MySQL Patch

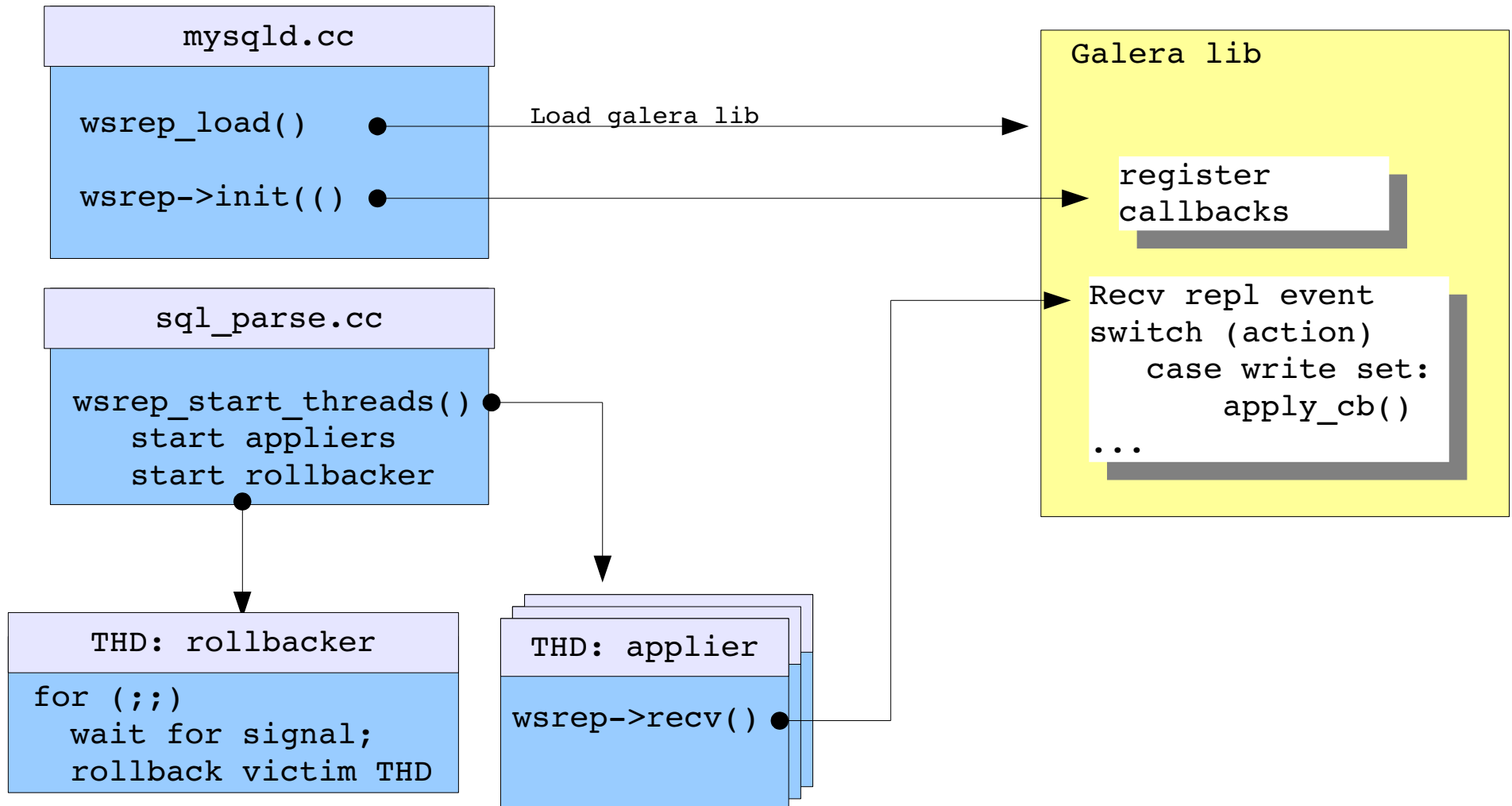
MySQL Patch

- Implements `wsrep_api.h` for MySQL 5.1/InnoDB
- Project: <https://launchpad.net/codership-mysql>
- Patch of ~10 000 lines
- Code surrounded by `#ifdef WITH_WSREP` directive
- Some new modules, named like: `wsrep_*`

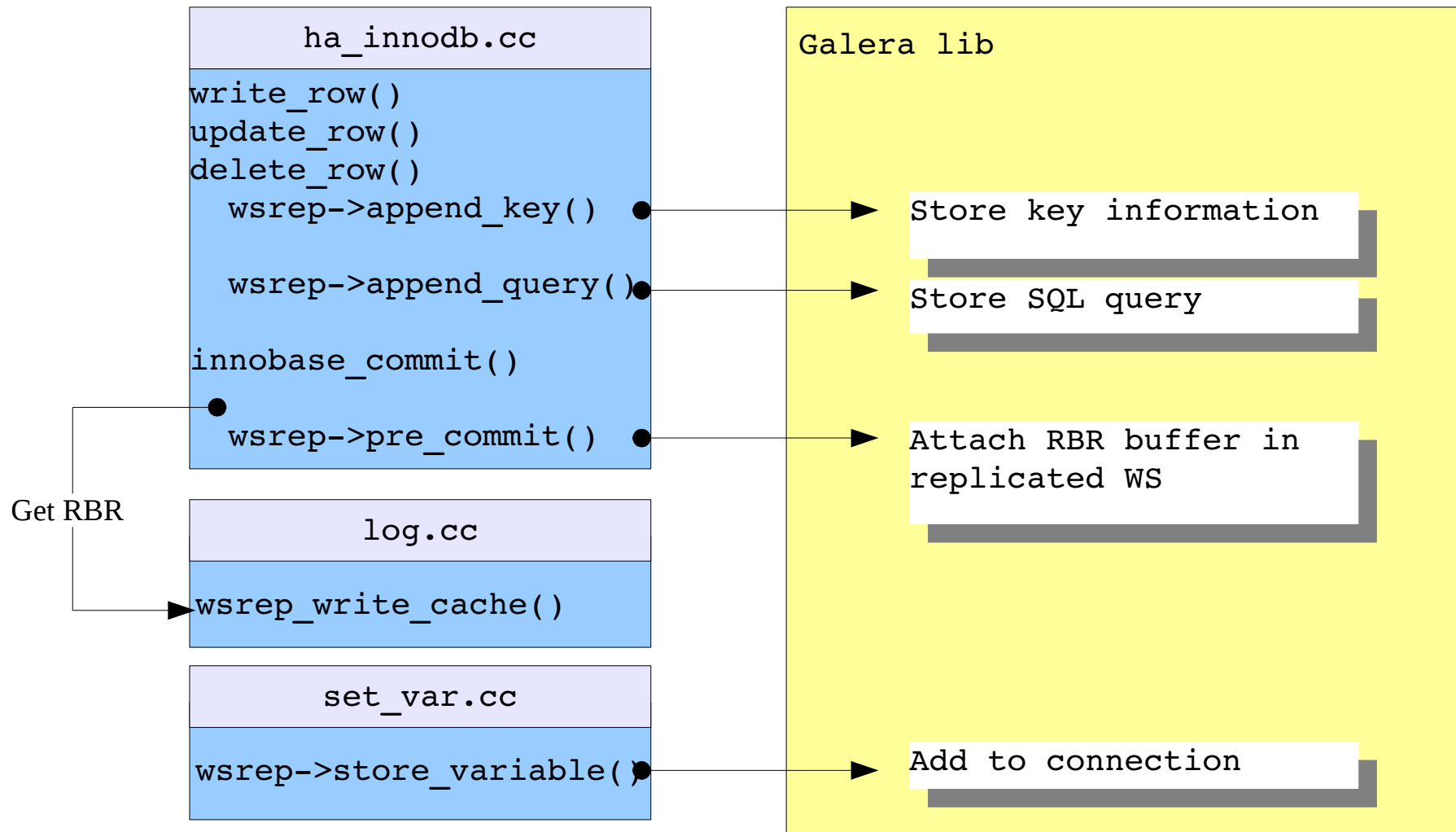
MySQL Patch

- Key information collected in methods:
 `ha_innobase :: write_row, update_row, delete_row`
- Collects RBR events from transaction cache
- By default, no binlog files
- High priority transaction support in InnoDB lock manager

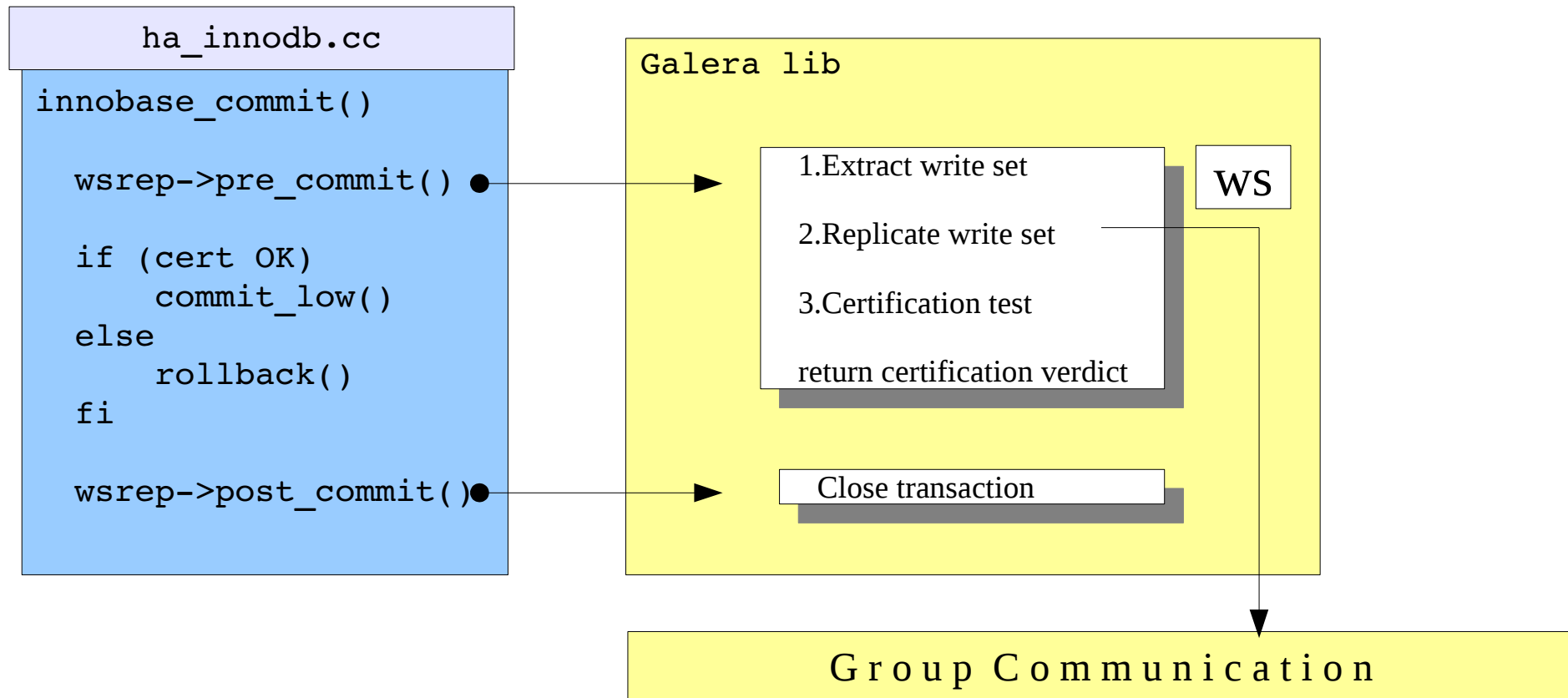
Wsrep Init



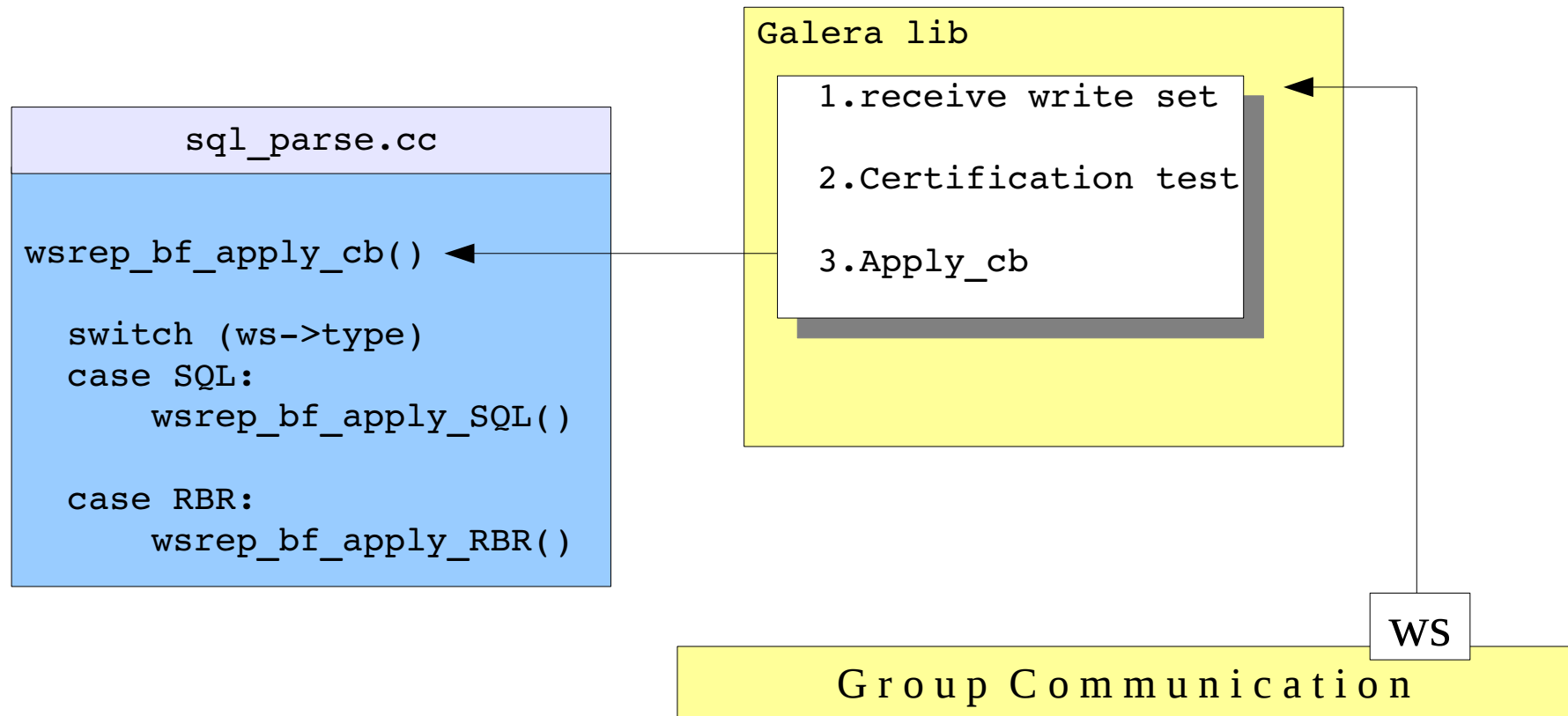
Write Set Population



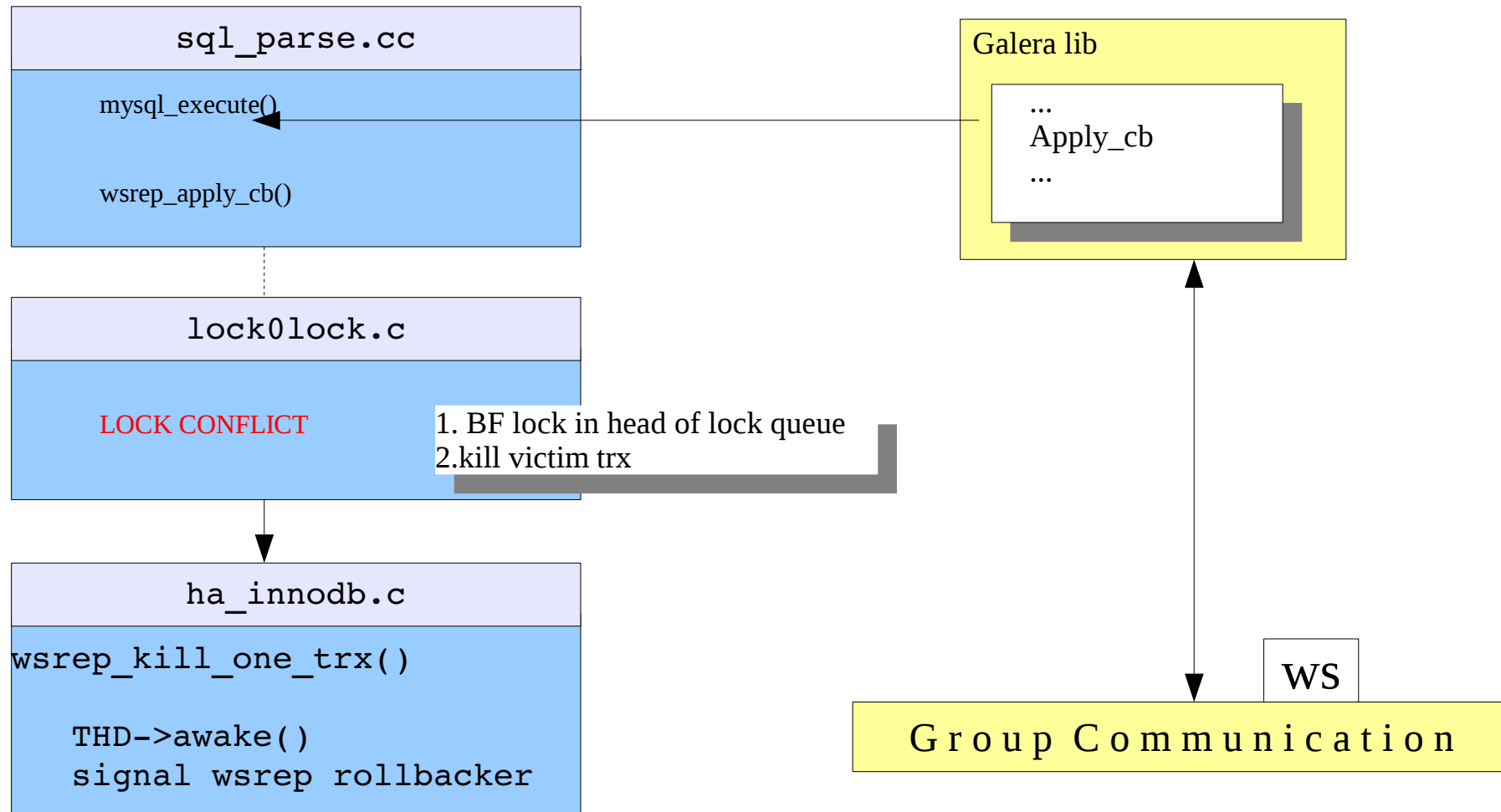
Commit



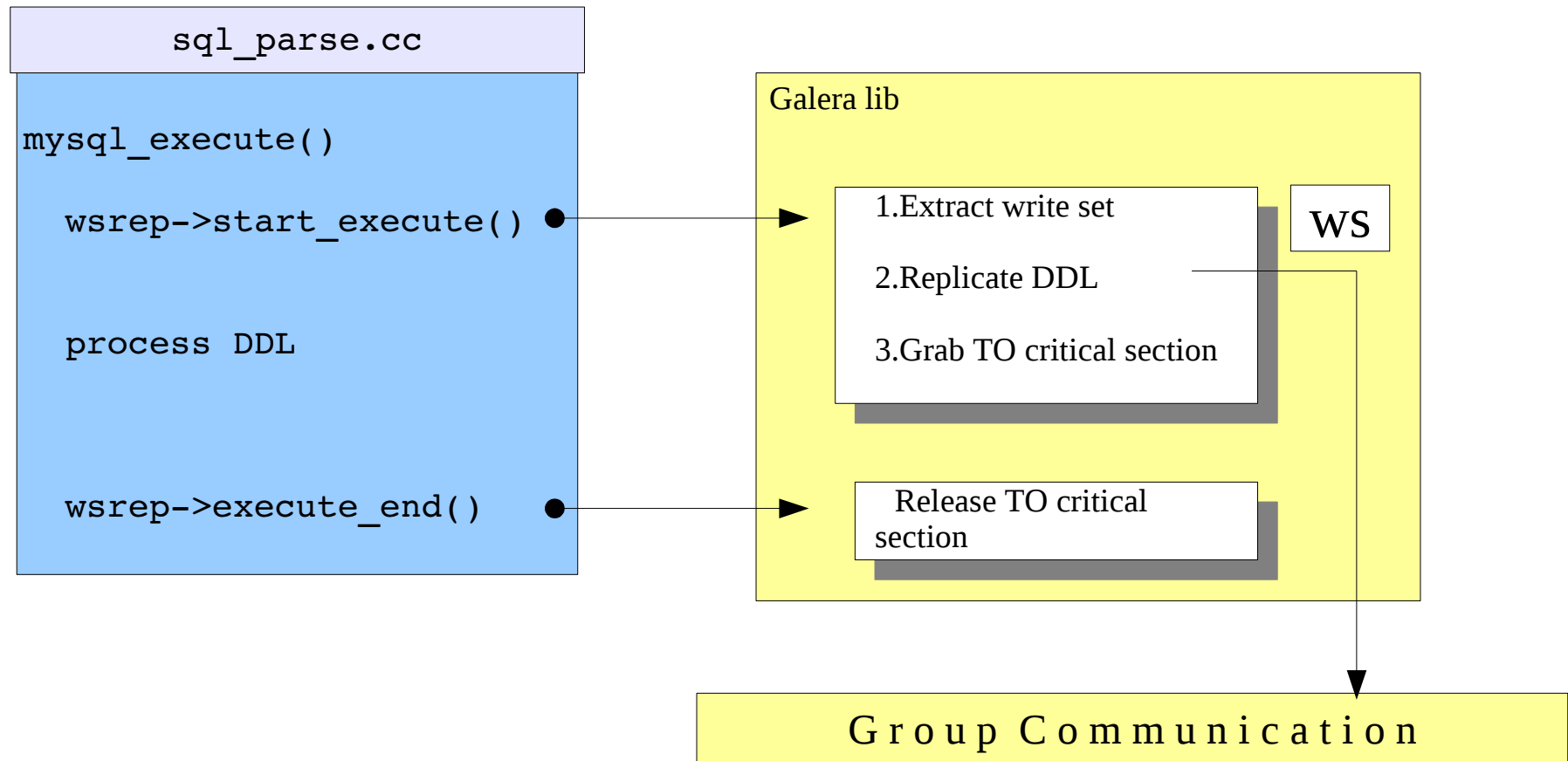
Applying



High Priority Transaction



DDL Replication



Wsrep Variables

```
mysql> show variables like 'wsrep%';
```

Variable_name	Value
wsrep_auto_increment_control	ON
wsrep_cluster_address	gcomm://
wsrep_cluster_name	my_wsrep_cluster
wsrep_convert_LOCK_to_trx	OFF
wsrep_data_home_dir	/home/galera/mysql-5.1.42-2957,1439/mysql/var/
wsrep_debug_option	NULL
wsrep_debug	OFF
wsrep_drupal_282555_workaround	ON
wsrep_local_cache_size	20971520
wsrep_node_incoming_address	10.0.0.121:3306
wsrep_node_name	abyssinian
wsrep_on	ON
wsrep_provider	/home/galera/mysql-5.1.42-2957,1439/galera/lib/libmmgalera.so
wsrep_provider_options	NULL
wsrep_retry_autocommit	ON
wsrep_slave_threads	1
wsrep_sst_auth	root:rootpass
wsrep_sst_donor	NULL
wsrep_sst_method	mysqldump
wsrep_sst_receive_address	AUTO
wsrep_start_position	NULL
wsrep_ws_persistency	OFF

```
22 rows in set (0.00 sec)
```

```
11 Feb 2010
```

Wsrep Status

```
mysql> show status like 'wsrep%';
```

Variable_name	Value
wsrep_local_state_uuid	0eedf650-1694-11df-0800-6227ab0639e3
wsrep_last_committed	3
wsrep_replicated	0
wsrep_replicated_bytes	0
wsrep_received	0
wsrep_received_bytes	0
wsrep_local_commits	0
wsrep_local_cert_failures	0
wsrep_local_bf_aborts	0
wsrep_flow_control_waits	0
wsrep_local_status	Joined (5)
wsrep_cluster_conf_id	1
wsrep_cluster_size	1
wsrep_cluster_state_uuid	0eedf650-1694-11df-0800-6227ab0639e3
wsrep_cluster_status	Primary
wsrep_local_index	0
wsrep_ready	ON

```
17 rows in set (0.00 sec)
```

MySQL Patch

- InnoDB changes are both in builtin and plugin engines
- Some refactoring is happening
 - Handlerton implementation
 - Applier victim aborting
 - DDL processing
- Some refactoring is planned
 - Use MySQL plugin framework
 - Management commands

Advanced Galera Features

Advanced Galera Features

- Optimistic concurrency control
- Flow control
- Connecting
- State Snapshot Transfer
- Auto increment management
- Parallel applying
- Retrying of aborted autocommit trxs
- etc...

Optimistic Concurrency Control

- Transactions proceed independently in each cluster node assuming they can eventually commit
 - There can be cluster wide conflicts
- Victim trx must abort in master node, and avoid committing in other nodes
- ER_DEADLOCK returned for cluster aborts

Optimistic Concurrency Control

- **Hot spots** are bad
 - .e.g. DBT2 shows pretty high conflict rate
- Long lasting transactions are vulnerable
- Rollbacks eat performance, but rollback happens only in one node, all other nodes just avoid applying
- With problematic SQL load, cluster can be adjusted to have a smaller number of write capable nodes

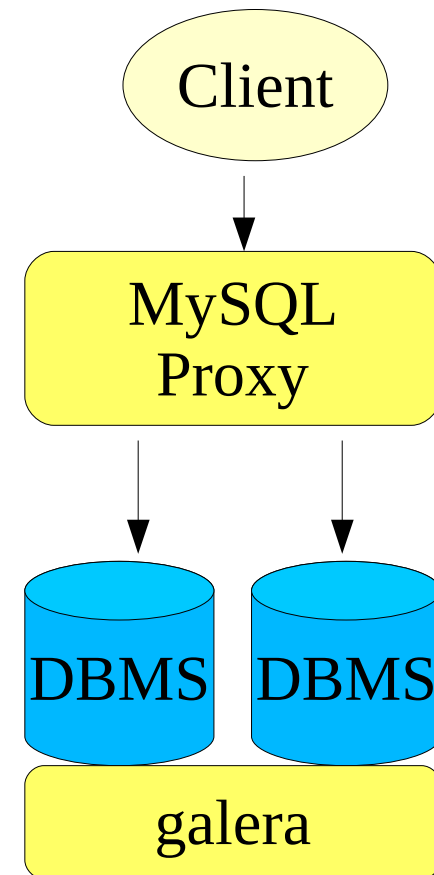
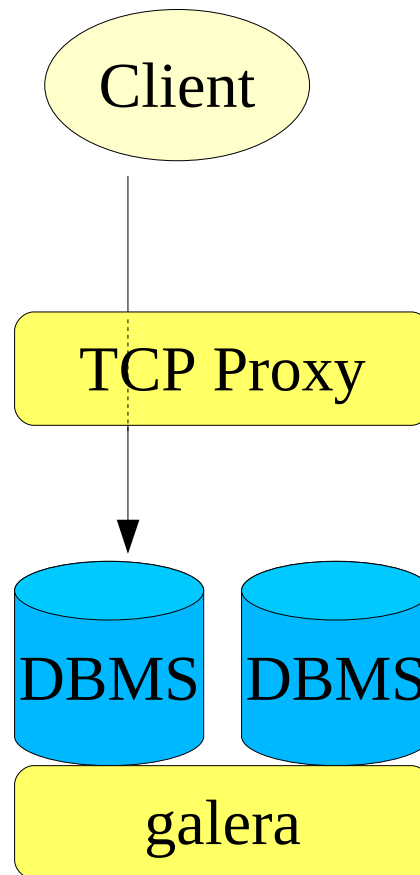
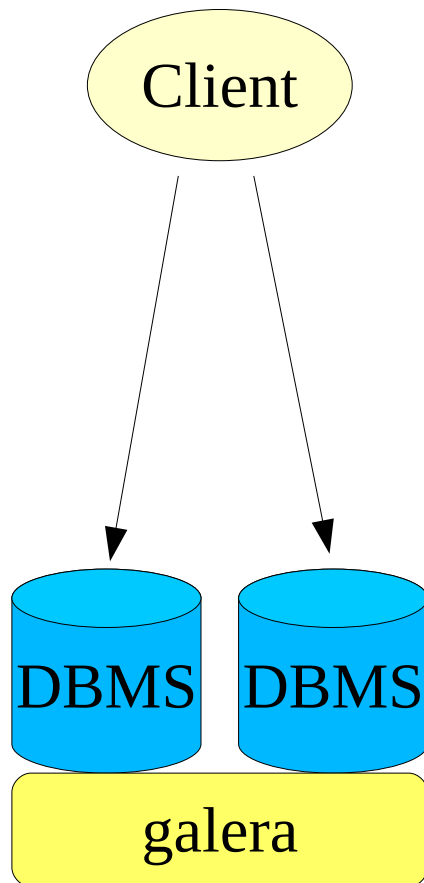
Flow Control

- Synchronous operation requires that each node will process with same frequency
- Without any flow control some node(s) would hijack all master activity => slave queues would grow indefinitely long
- Galera has group communication actions for managing flow control
- There are limits for slave queue length, and triggering for flow control

Connecting

- Galera is true multi-master, clients can directly connect to any node
- Also connection pool can know the node addresses and balance connections to the cluster
- If single connection point is needed:
 - TCP load balancer is viable option (like Galera load balancer, glb)
 - Proxy component (MySQL Proxy)

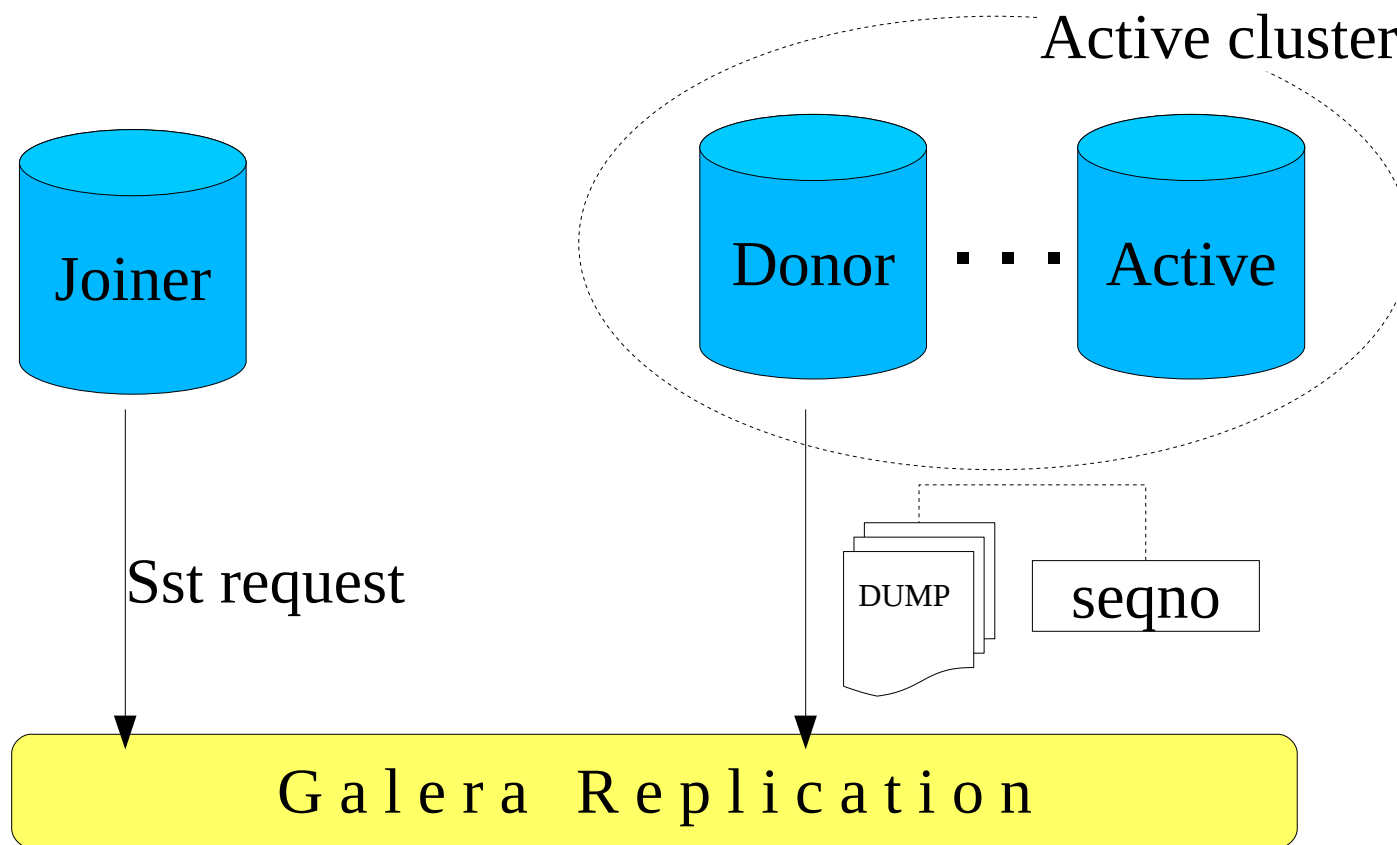
Connecting



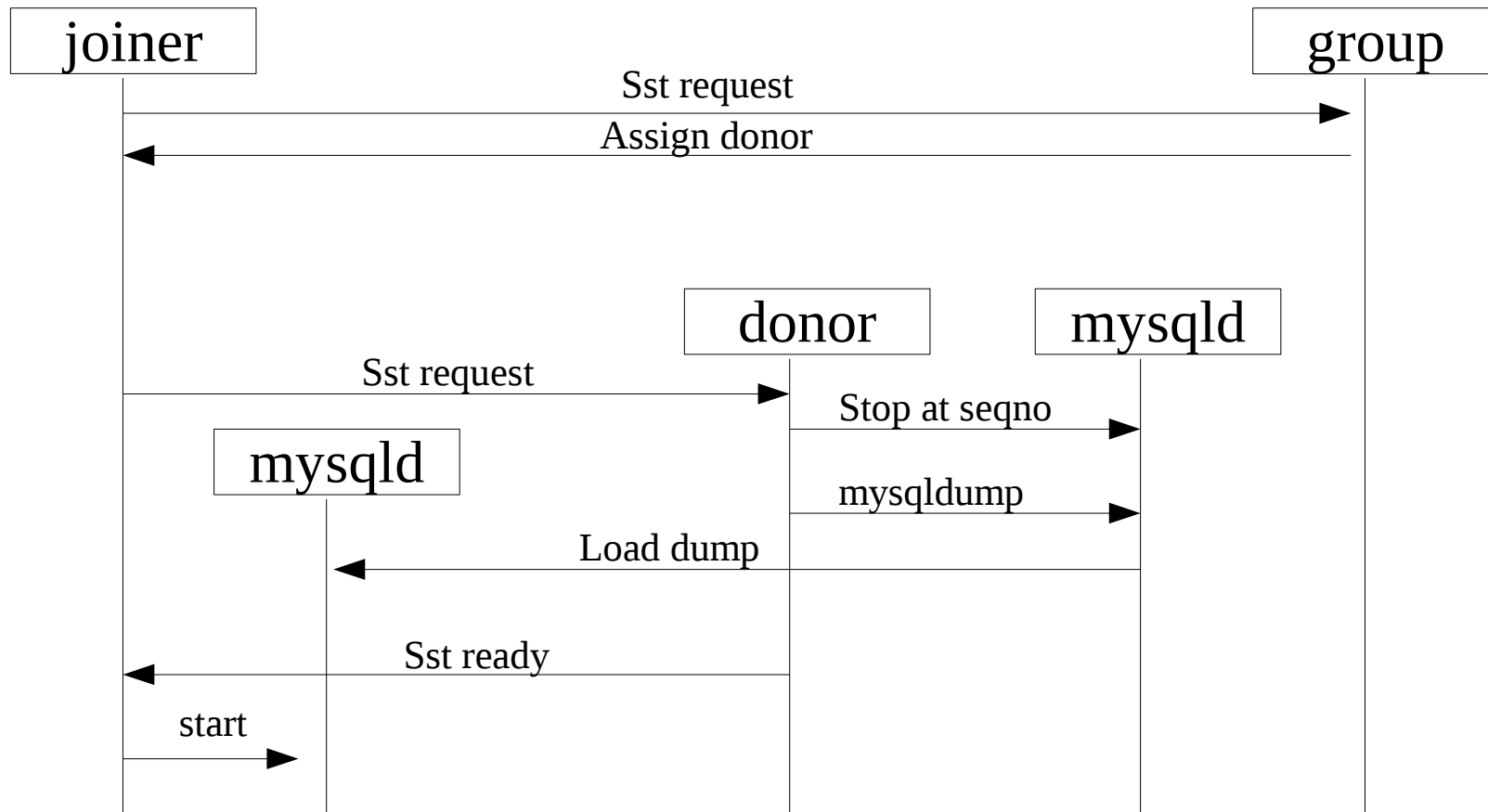
DB State Transfer

- To join new node in running cluster
- Copies DB state from an active node to the joiner
- The task is:
 - Donor must prepare a DB snapshot in known position of replication stream (seqno)
- DB State snapshot transfer options:
 - Mysqldump release 0.7
 - Xtrabackup release 0.8
 - LVM release 0.8
 - InnoDB HotBackup

DB State Transfer



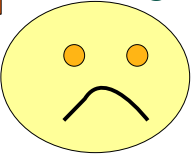
SST in Release 0.7



Autoincrements

- Galera manages autoinc control variables (auto_increment_increment, auto_increment_offset)
- This happens optionally:
wsrep_autoinc_control = ON/OFF
- Autoincrement control is triggered by cluster membership changes:
 - Increment = number of cluster nodes
 - Offset = node id
- Autoincrement control can yield best possible insert performance

Parallel Applying

- It is possible to launch several appliers
 - `wsrep_slaves = n`, option
 - Appliers run in parallel until commit time
-  **But: no performance gain has been observed**
- RBR event applying is very fast compared to SQL query processing
 - Galera cluster scales even with 100% write rate
 - Flow control cuts some of parallelism
- (Implementation suffers from ALG issue)

Retrying

- “Cluster can abort trxs at will”
- If autocommit trx was aborted due to cluster wide deadlock, it is safe to immediately retry the trx
- Option: `wsrep_retry_autocommit=On/OFF`

Maximal Insert Performance

- Even faster inserts?
- Due to autoincrement management, inserts won't conflict
- Transactions with only inserts (...and which get auto inc value) don't need to certify
- We can/could skip certification test and process inserts even faster

Myisam support?

- Can be supported, if myisam writes will go through one selected node
 - Myisam support will fall back to master slave replication (synchronous)
- Galera can reject myisam writes in any other node

Galera Project

State of the Code

- Public releases since Jan 2009
- Current release 0.7.2
 - Node join/drop property
 - SST by mysqldump
 - Gcomm by TCP transport
- fully open source

Release 0.7

- Current release 0.7.2
 - Stable release
 - Production readiness
- Simple management & installation utilities
- Nodes can join/drop active cluster
- State transfer by mysqldump
- “Reasonably” good performance

Road Map

- Maintenance releases for 0.7 series
- Release 0.8 – **Optimization Milestone**
 - Incremental State Transfer
 - Xtrabackup, LVM
 - Multicast GCS
 - Q2/10
- Release 0.9 - **Security Milestone**
 - TLS tunneling
- Release 1.0 - **Management Milestone**
 - Management & Monitoring tools

Summary

- Certification based replication turns out effective
 - High Availability
 - Transparency
 - Good scalability even with high write rates
- wsrep API is “not too hard” to implement
- Any (transactional) DBMS can leverage this replication possibility

Thank You

codership

- Downloads available: <http://www.codership.com>
- info@codership.com
- Mailing list: codership-team@googlegroups.com
- <https://launchpad.net/wsrep>
- <https://launchpad.net/codership-mysql>