MariaDB/MySQL Galera Cluster

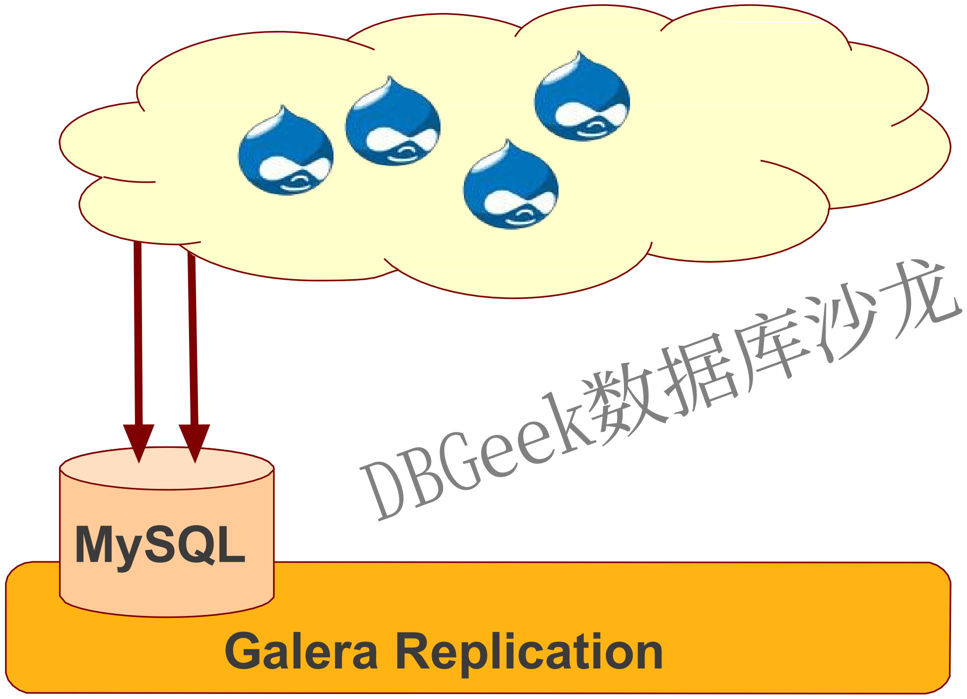
杭州沃趣网络科技有限公司
Hangzhou WOQU Technology Co., Ltd.

Let Data Drive!

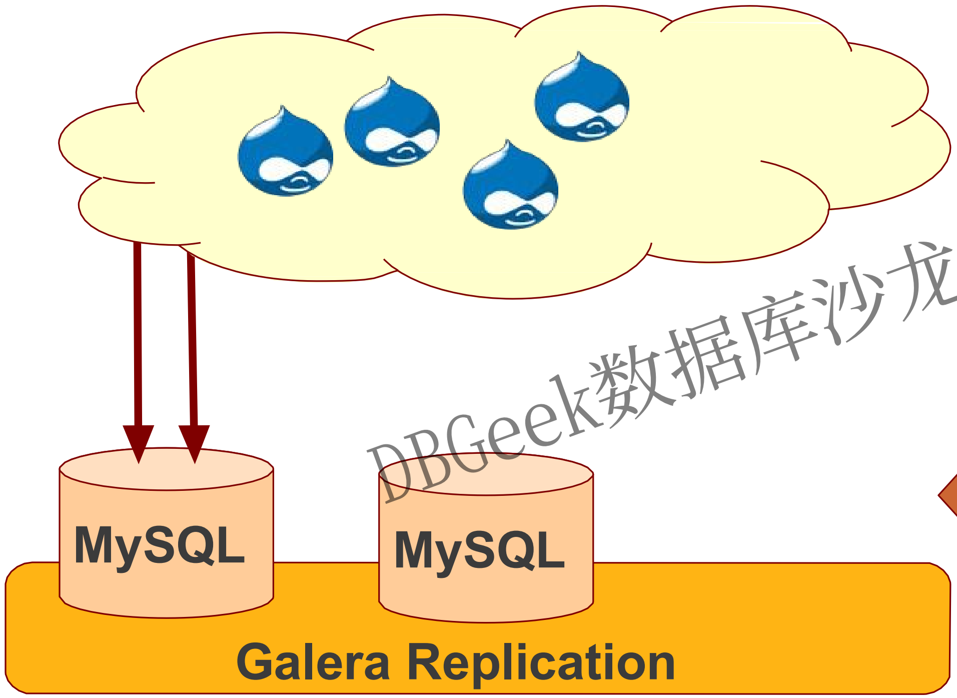
MariaDB/MySQL Galera Cluster

- General intoduction:
  - The Cluster consists of Nodes.
  - Each Node is regular *MySQL / Percona / MariaDB Server* setup
  - Each Node contains the full copy of data

Multi-Master Replication

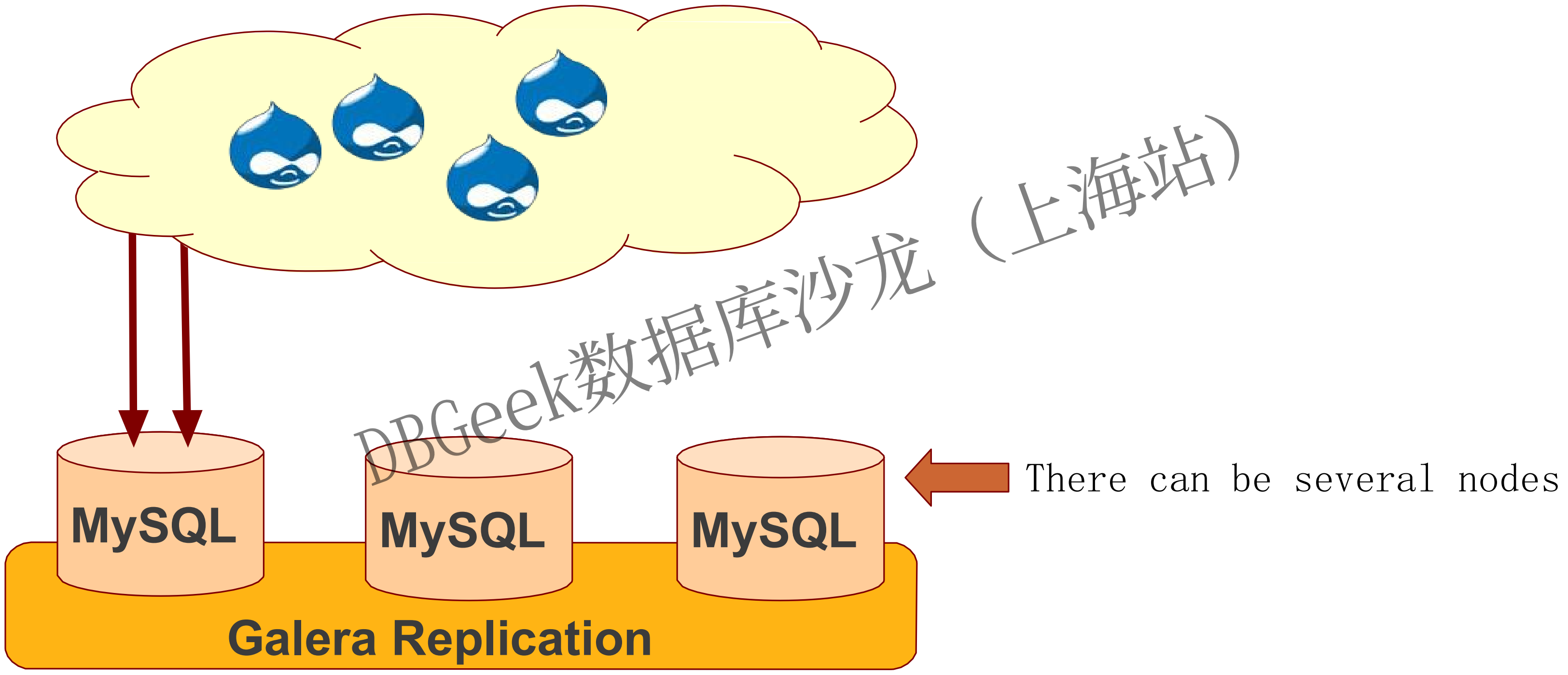


Multi-Master Replication

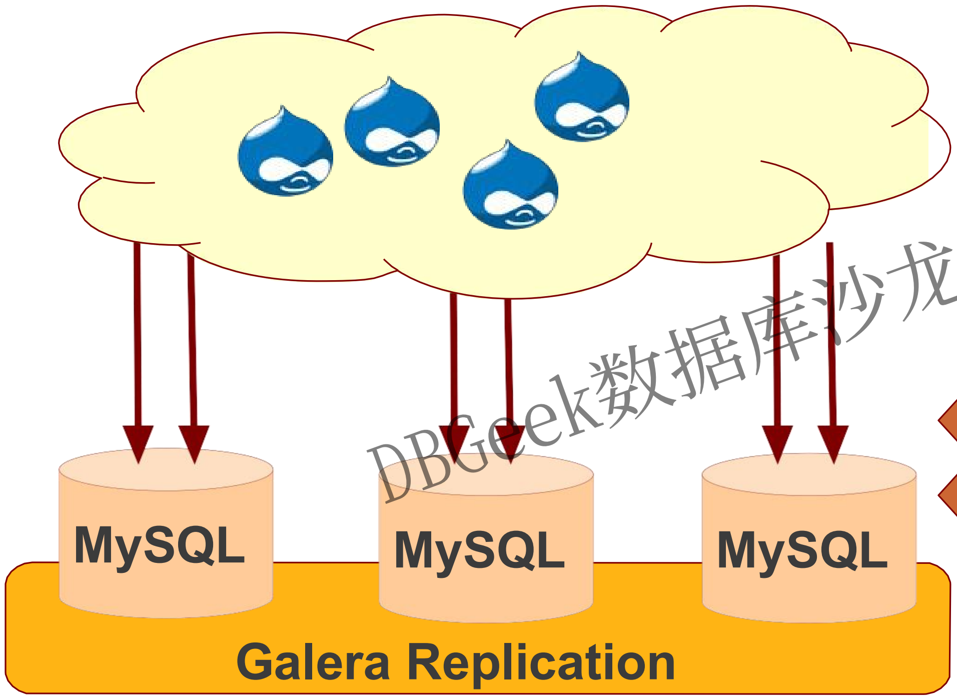


There can be several nodes

Multi-Master Replication



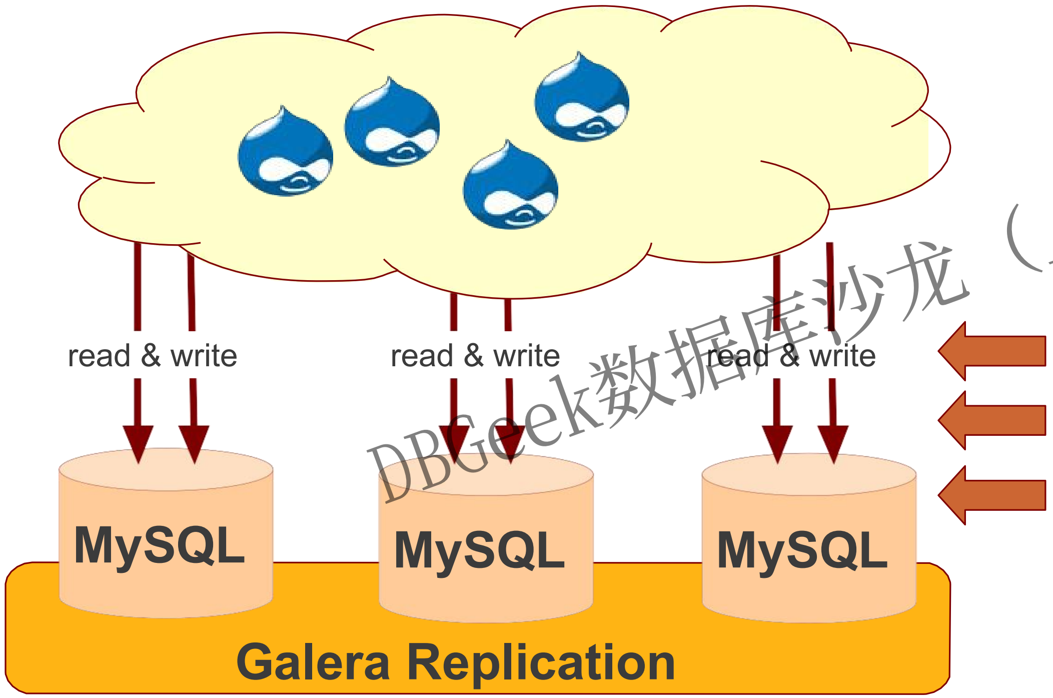
Multi-Master Replication



Client can connect to any node

There can be several nodes

Multi-Master Replication

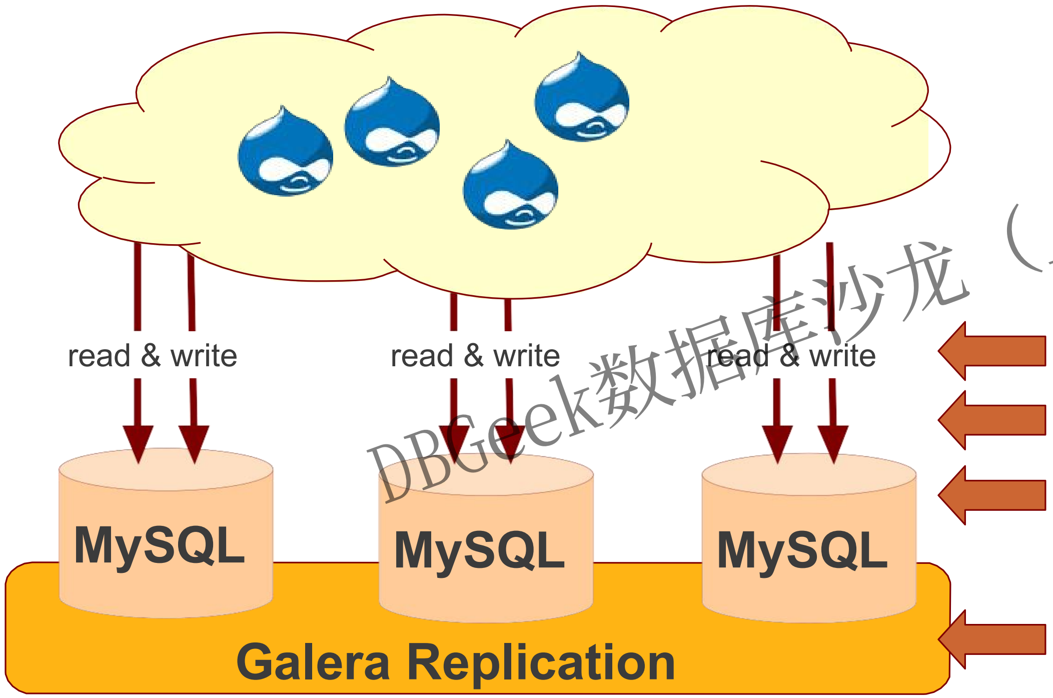


Read & write access to any node

Client can connect to any node

There can be several nodes

Multi-Master Replication



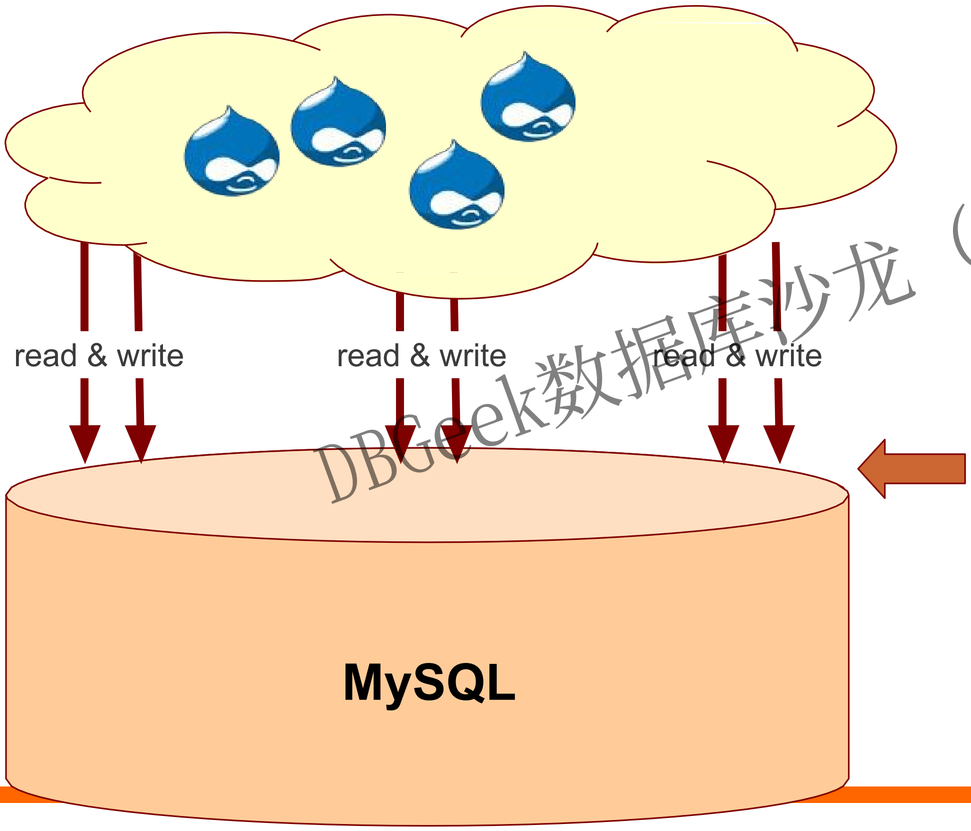
Read & write access to any node

Client can connect to any node

There can be several nodes

Replication is synchronous

Multi-Master Replication



Multi-master cluster looks like one big database with multiple entry points

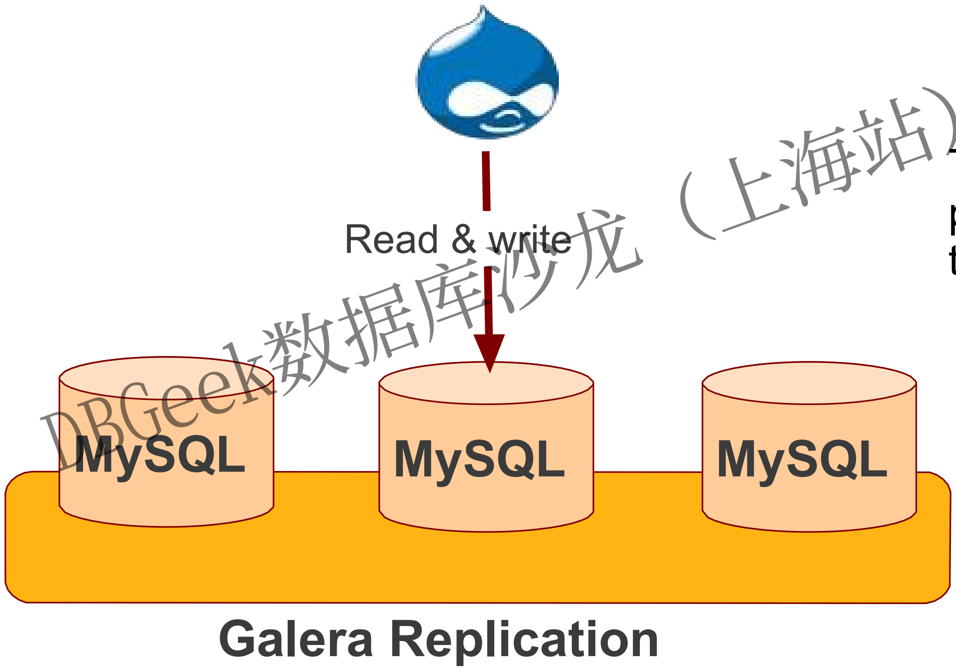
- Benefits
  - When you execute a query, it is executed locally on the node
  - No central management. You can loose any node at any point of time, and the cluster will continue to function.
  - Good solution for scaling a read workload. You can put read queries to any of the nodes.

- Drawbacks
  - Overhead of joining new node.
  - This can’t be used as an effective write scaling solution.
  - You have several duplicates of the data, for 3 nodes - 3 duplicates.

Galera Cluster

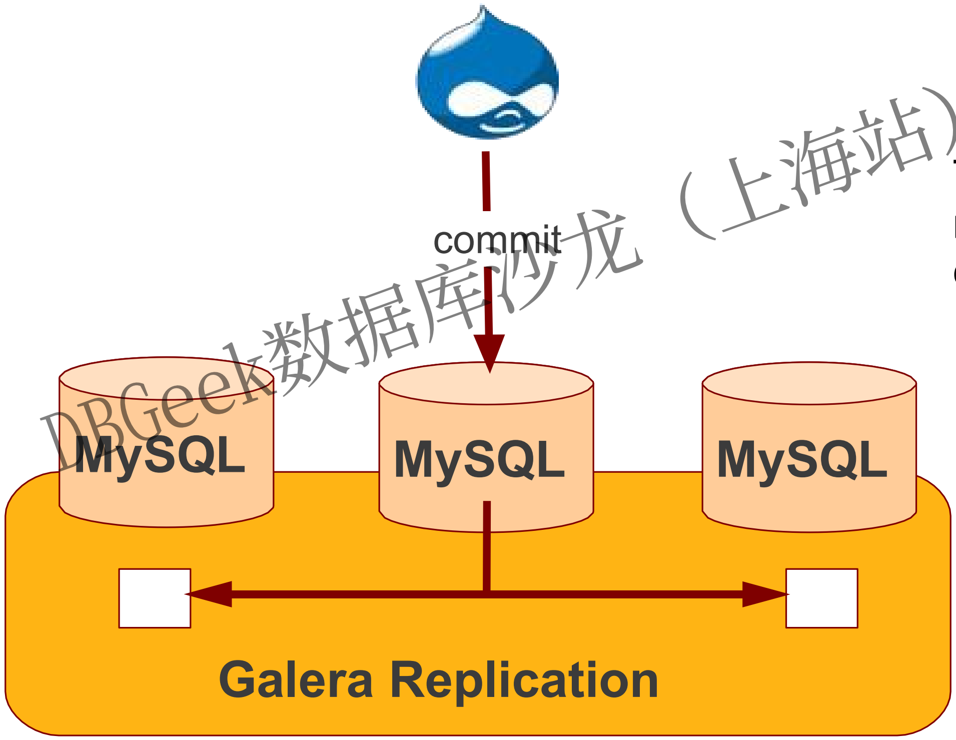
- Synchronous multi-master cluster
- For InnoDB Engine
- 3 or more nodes needed for HA
- Automatic node provisioning
- Works in LAN / WAN / Cloud

Synchronous Replication



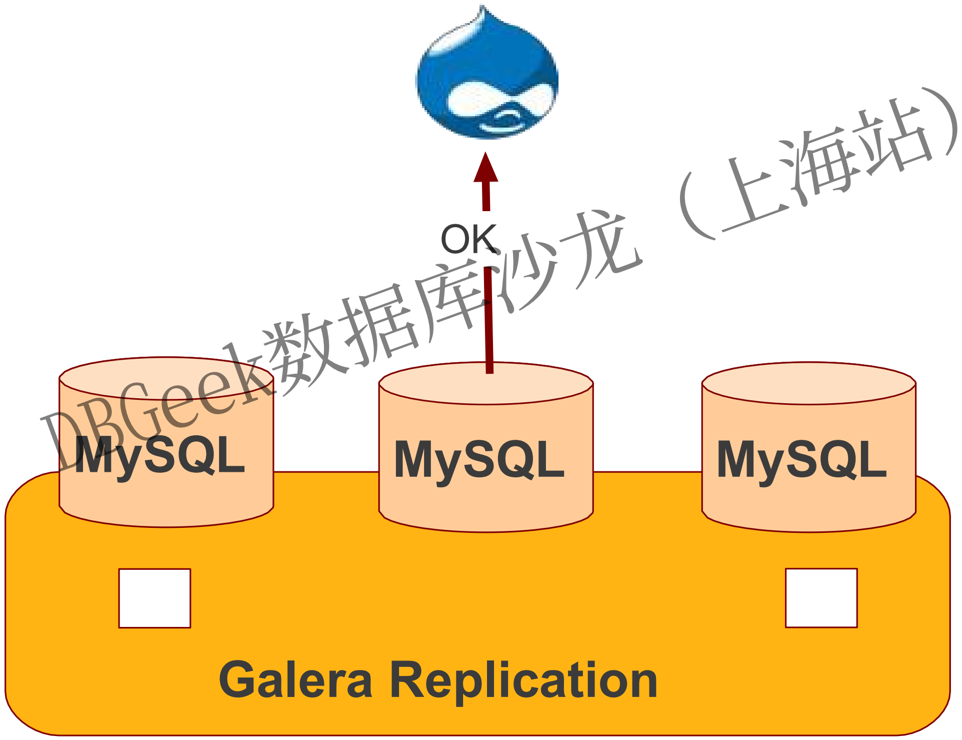
Transaction is processed locally up to commit time

Synchronous Replication



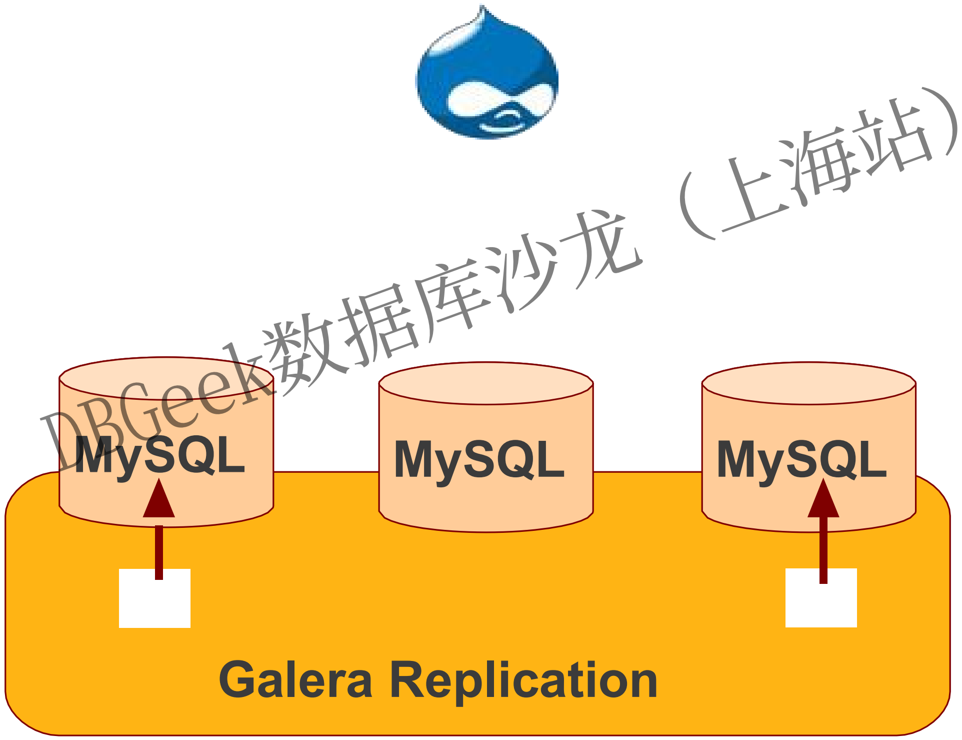
Transaction is replicated to whole cluster

Synchronous Replication



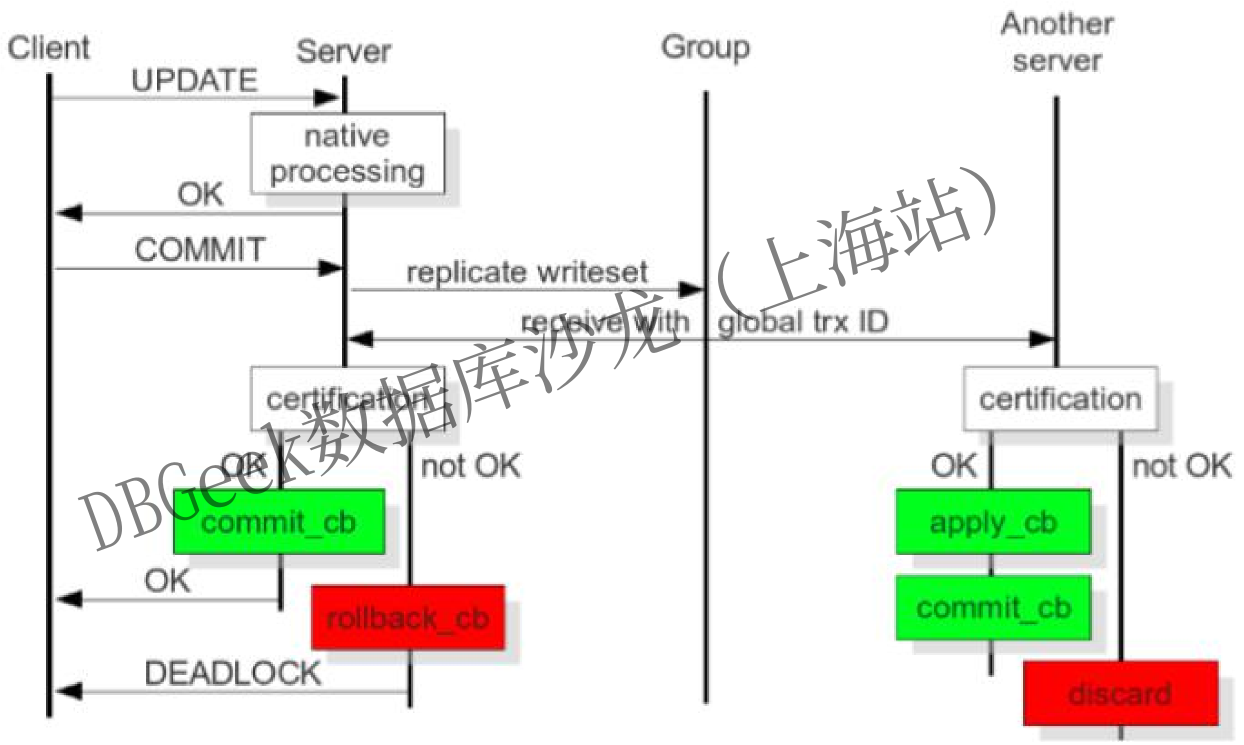
Client gets OK status

Synchronous Replication



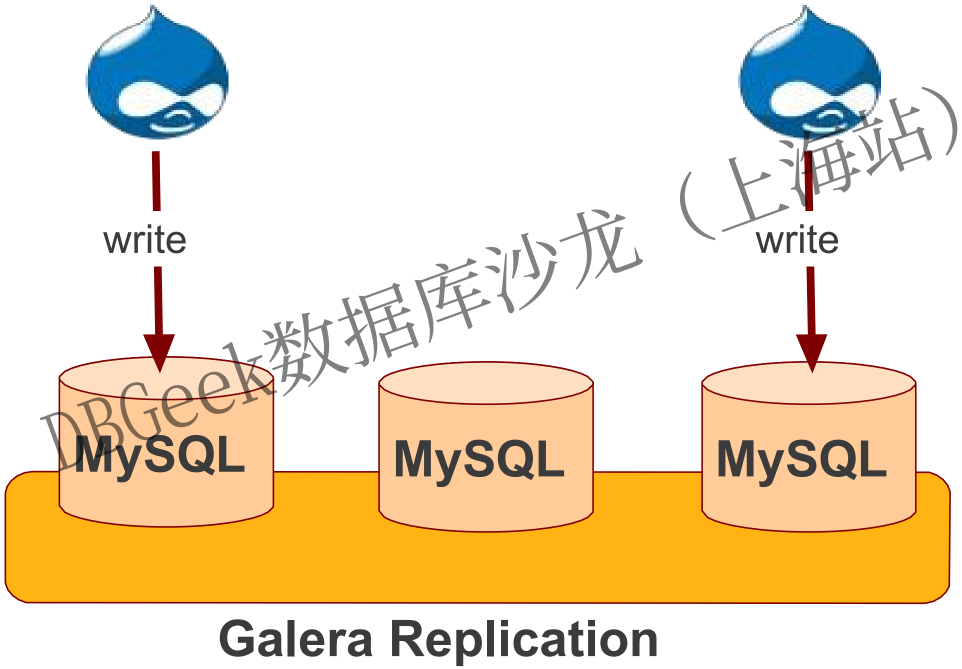
Transaction is applied in slaves

Certification-Based Replication

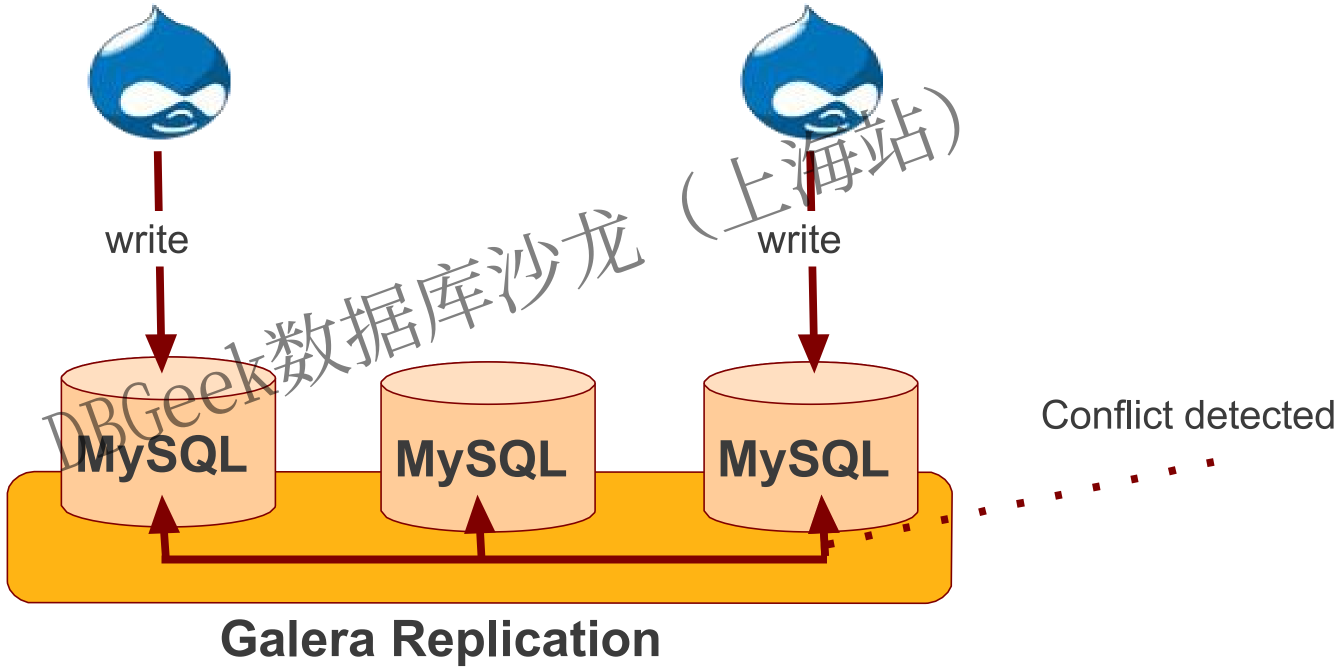


Multi-Master Conflicts

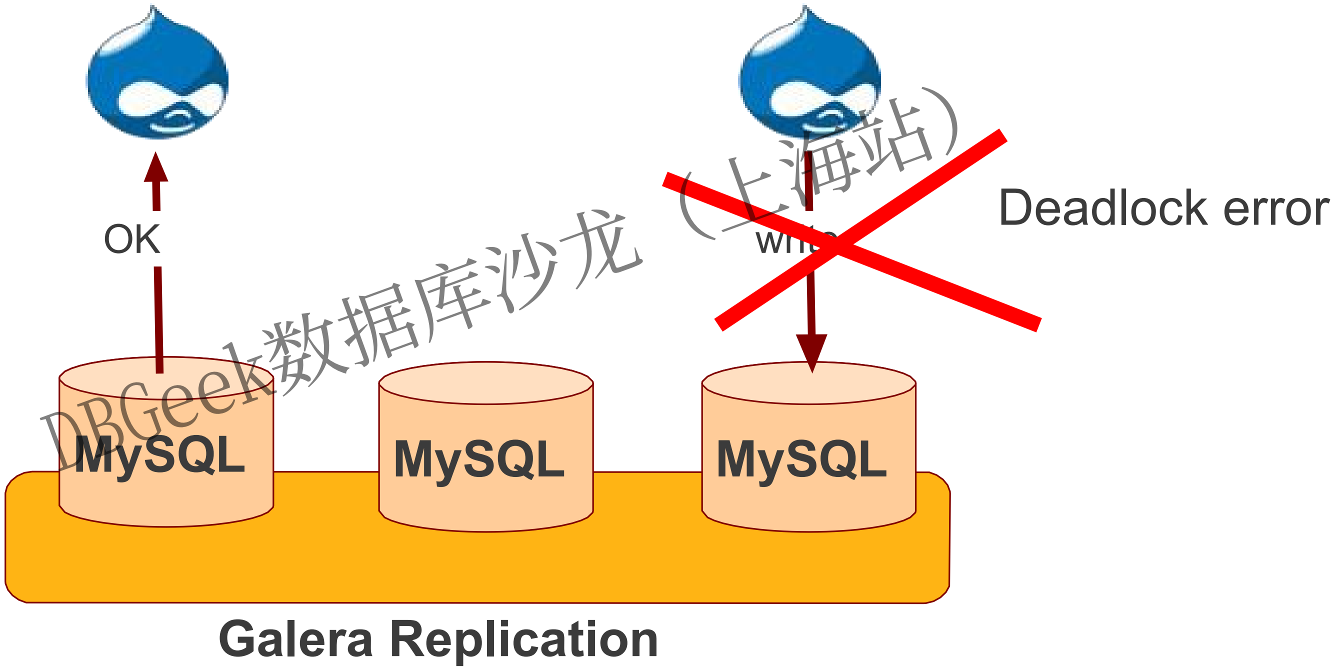
Multi-Master Conflicts



Multi-Master Conflicts



Multi-Master Conflicts



Multi-Master Conflicts

- Galera uses optimistic concurrency control
- If two transactions modify same row on different nodes at the same time, one of the transactions must abort

  ➔ Victim transaction will get deadlock error
- Application should retry deadlocked transactions, however not all applications have retrying logic inbuilt

Diagnosing Multi-Master Conflicts

- using wsrep_debug configuration, all conflicts (...and plenty of other information) will be logged
- wsrep_log_conflicts which will cause each cluster
- conflict to be logged in mysql error log Monitor
- wsrep_local_bf_aborts
- wsrep_local_cert_failures

Node A	Node B

wsrep_local_bf_aborts=0
begin;
update tb_a set c='a' where i=1	

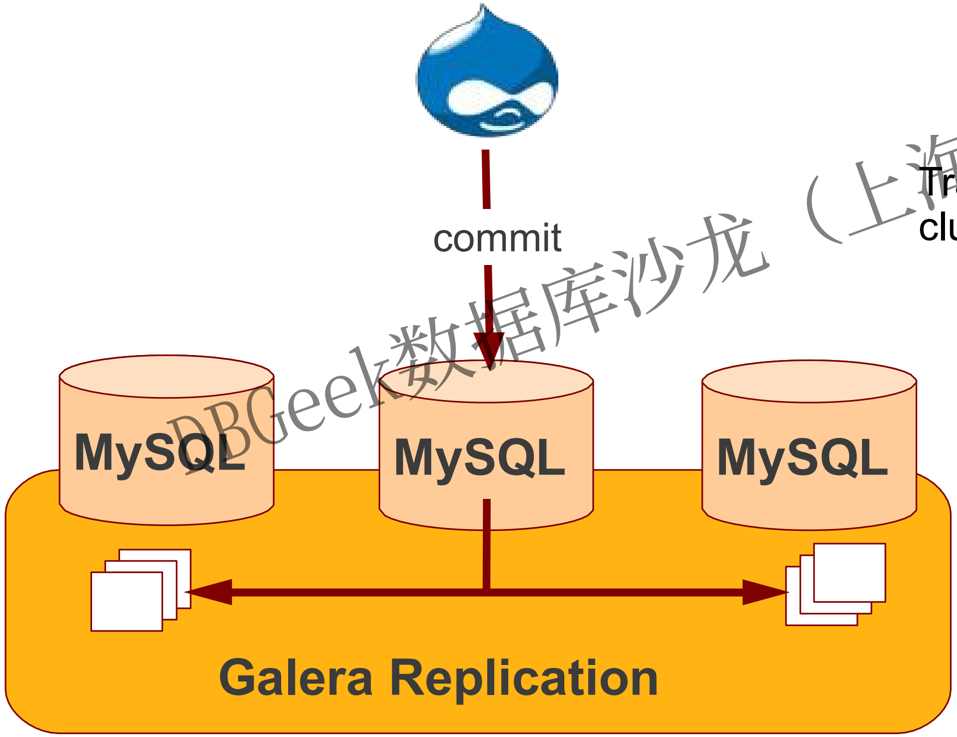
	begin;
	delete from tb_a where i=1;
	commit;

ERROR 1213 (40001): Deadlock found when trying to get lock; try restarting transaction	
wsrep_local_bf_aborts=1	

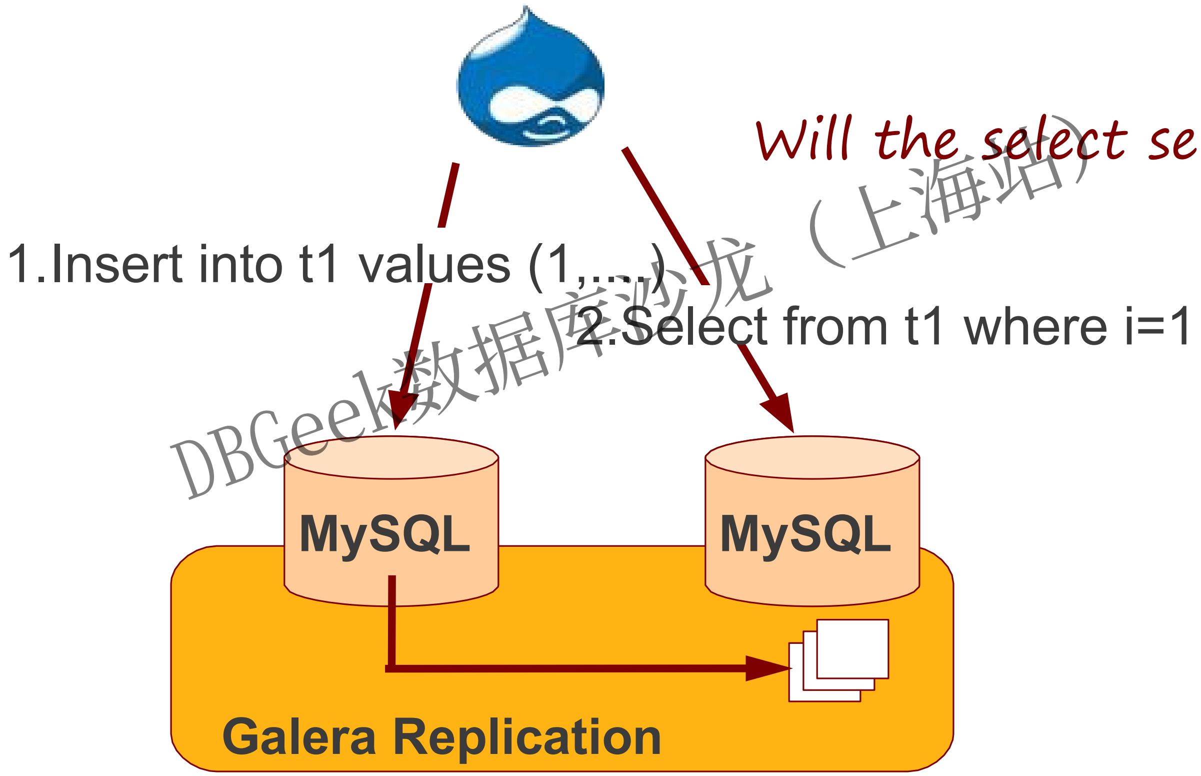
```
………
Victim thread:
   THD: 9, mode: local, state: executing, conflict: cert failure, seqno: 6198670
   SQL: delete from dd where i=4

*** Priority TRANSACTION:
………
*** Victim TRANSACTION:
……..
*** WAITING FOR THIS LOCK TO BE GRANTED:
……
2016-11-17 15:59:45 139742395034368 [Note] WSREP: Winning thread:
   THD: 2, mode: applier, state: executing, conflict: no conflict, seqno: 6198669
   SQL: (null)
2016-11-17 15:59:45 139742395034368 [Note] WSREP: Victim thread:
   THD: 9, mode: local, state: executing, conflict: cert failure, seqno: -1
   SQL: delete from tb_a where i=1
```

Consistent Reads

Replication is virtually synchronous...



Transaction is replicated to whole cluster



Will the select see the inserted ro

Consistent Reads

- Aka read causality
- There is causal dependency between operations on two database connections
  - Application is expecting to see the values of earlier write

Consistent Reads

Use: wsrep_causal_reads=ON

➔ Every read (select, show) will wait until slave queue has been fully applied

There is timeout for max causal read wait

- replicator.causal_read_keepalive

State Transfers

Joining node needs to get the current database state

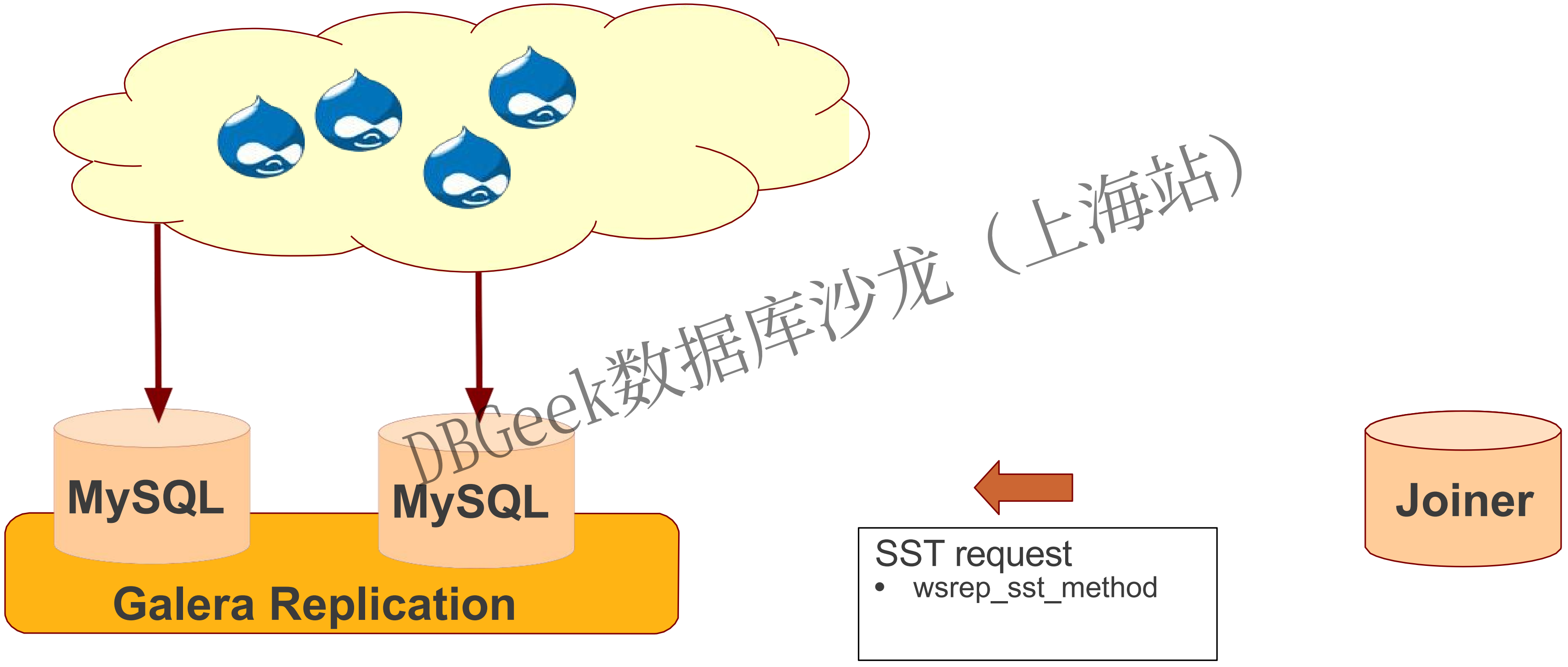
- ➢ Two choices:
  - ➢ IST: incremental state transfer
  - ➢ SST: full state transfer
- ➢ If joining node had some previous state and gcache spans to that, then IST can be used

State Snapshot Transfer

To send full database state

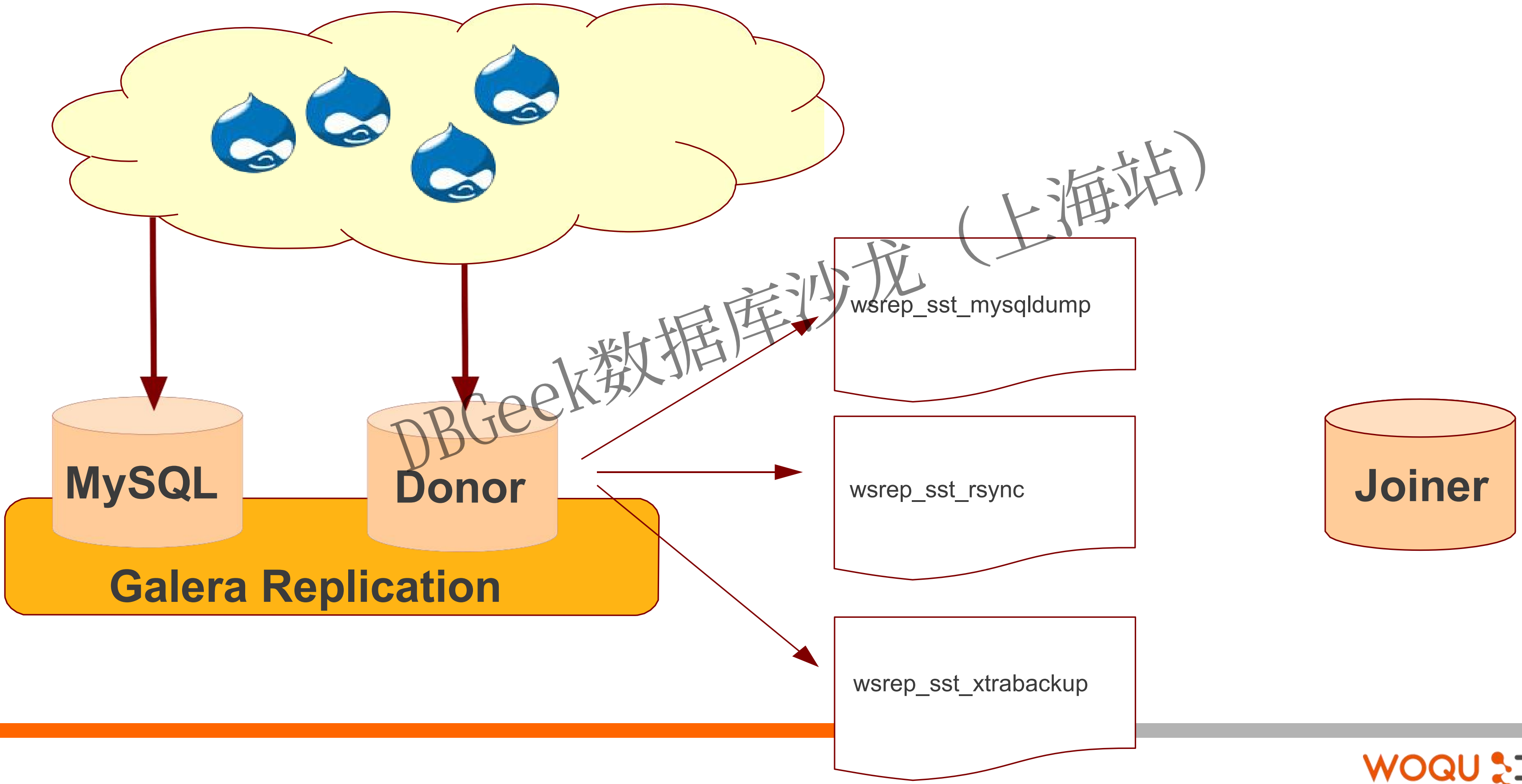
- wsrep_sst_method to choose the method:
  - ➢ mysqldump
  - ➢ rsync
  - ➢ xtrabackup

SST Request



SST request
- wsrep_sst_method

SST Method


MySQL
Donor
Galera Replication
wsrep_sst_mysqldump
wsrep_sst_rsync
wsrep_sst_xtrabackup
Joiner

SST API

- SST is open API for shell scripts
- Anyone can write custom SST
- SST API can be used e.g. for:
  - Backups
  - Filtering out part of database

wsrep_sst_mysqldump

- Logical backup
- Slowest method

Configure authentication

  - ➢ wsrep_sst_auth=”root:rootpass”
  - ➢ Super privilege needed

- Make sure SST user in donor node can take mysqldump from donor and load it over the network to joiner node
- You can try this manually beforehand

wsrep_sst_rsync

- Physical backup
- Fast method
- Can only be used when node is starting
- ➢ Rsyncing datadirectory under running InnoDB is not possible

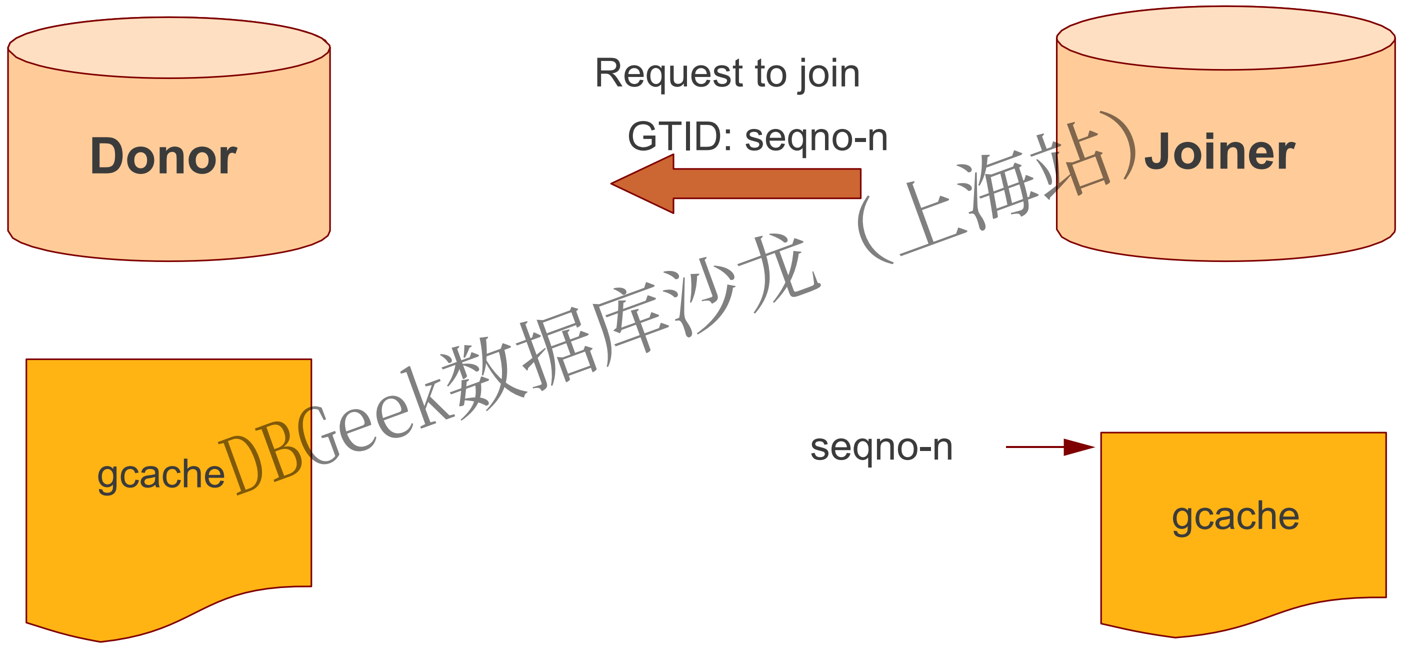
wsrep_sst_xtrabackup

- Contributed by Percona
- Probably the fastest method
- Uses xtrabackup
- Least blocking on Donor side (short readlock is still used when backup starts)

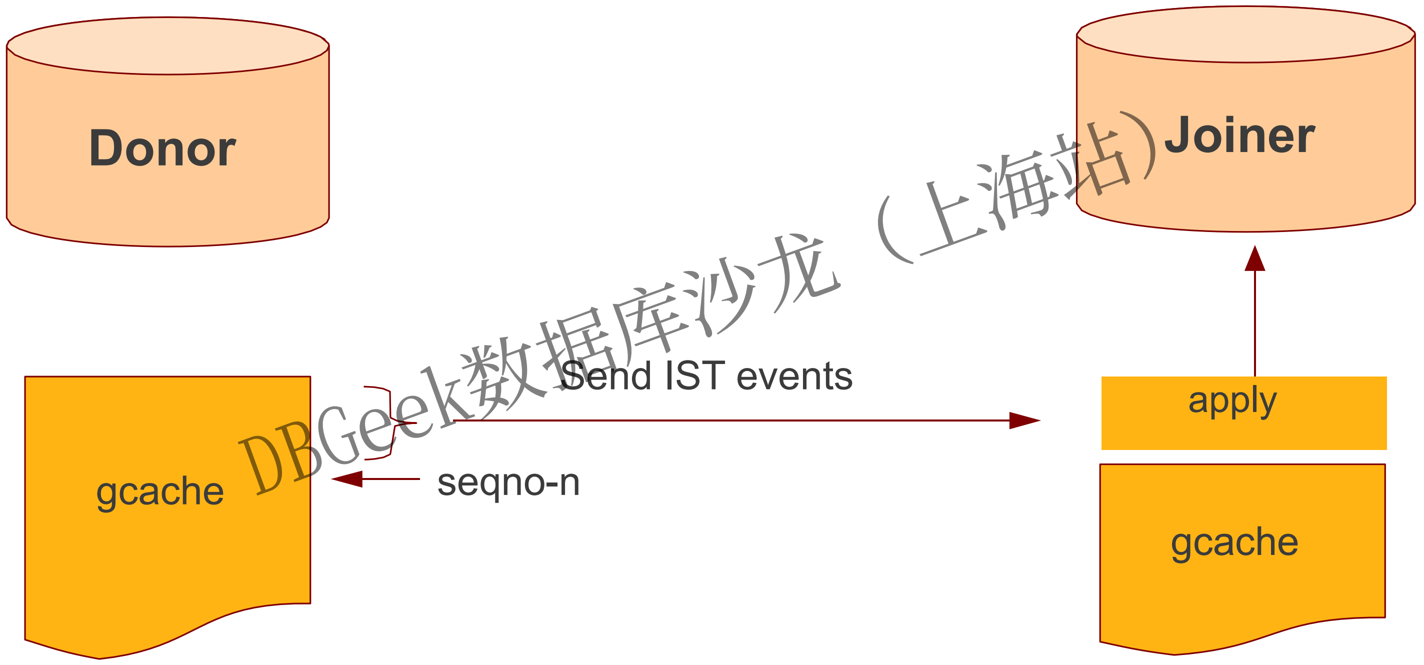
SST Donor

- All SST methods cause some disturbance for donor node
- By default donor accepts client connections, although committing will be prohibited for a while
- If wsrep_sst_donor_rejects_queries is set, donor gives unknown command error to clients
- ➔Best practice is to dedicate a reference node for donor and backup activities

Incremental State Transfer



Incremental State Transfer



Incremental State Transfer

- Very effective
- gcache.size parameter defines how big cache will be maintained
- gcache is mmap, available disk space is upper limit for size allocation

Incremental State Transfer

Use database size and write rate to optimize gcache:

- gcache < database
- Write rate tells how long tail will be stored in cache

Incremental State Transfer

- You can think that IST Is
  - A short asynchronous replication session
  - If communication is bad quality, node can drop and join back fast with IST

Backups Backups Backups

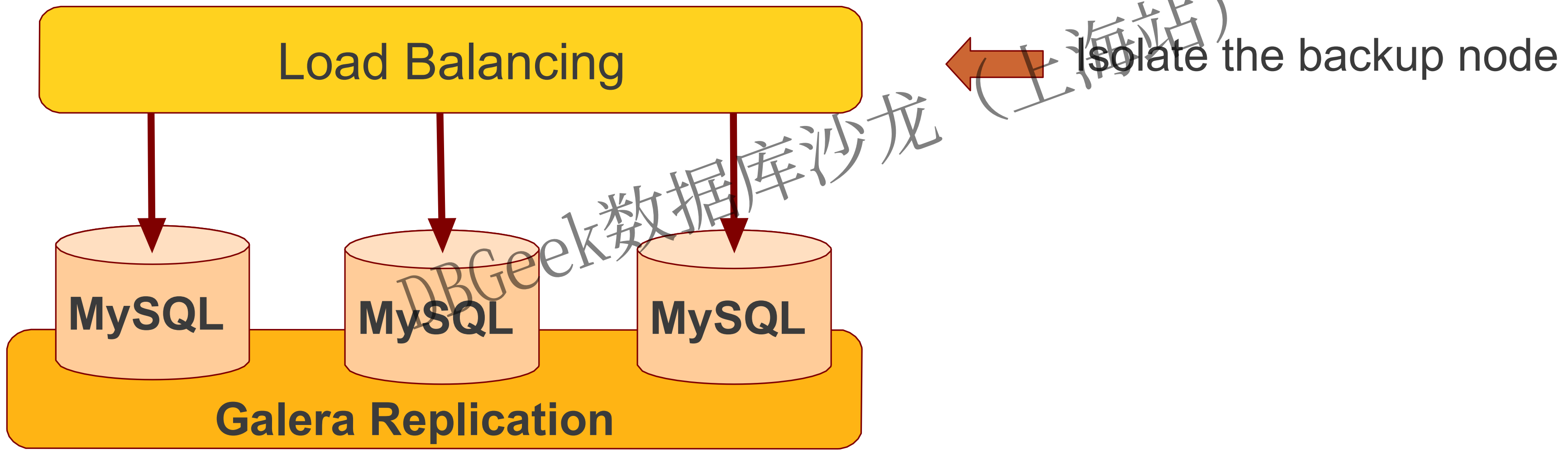
Backups

- All Galera nodes are constantly up to date Best practices:
  - ➢Dedicate a reference node for backups
  - ➢Assign global trx ID with the backup
- Possible methods:
  1. Disconnecting a node for backup
  2. Using SST script interface
  3. xtrabackup

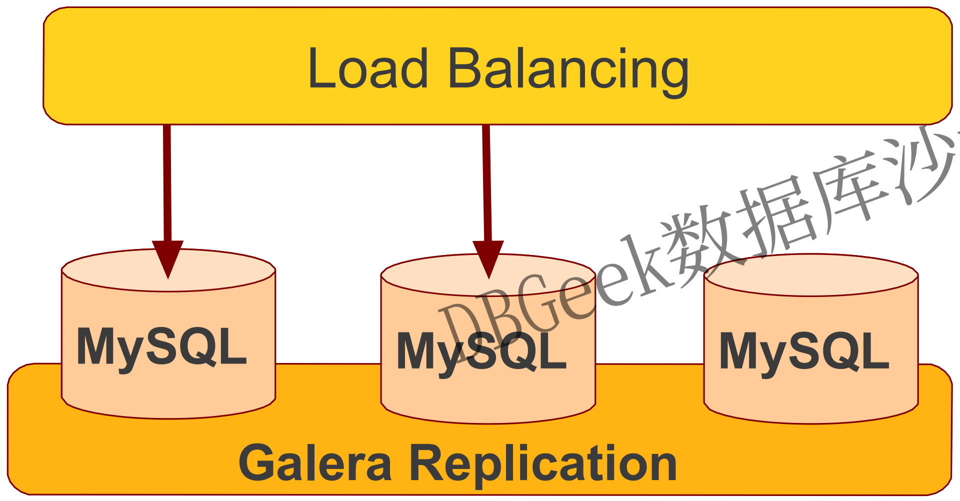
Backups with global Trx ID

- Global transaction ID (GTID) marks a position in the cluster transaction stream
- Backup with known GTID make it possible to utilize IST when joining new nodes, eg, when:
  - Recovering the node
  - Provisioning new nodes

Backup by Disconnecting a Node

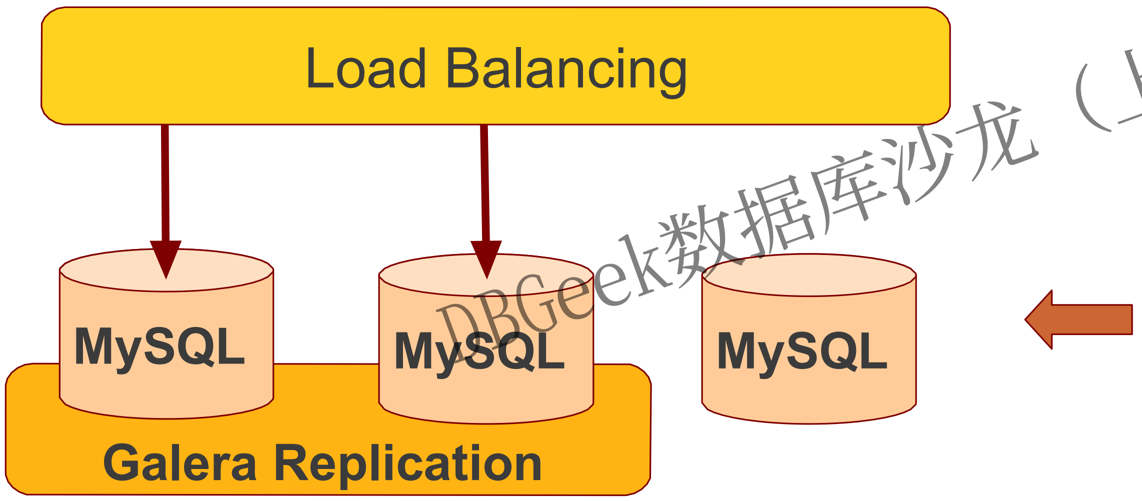


Backup by Disconnecting a Node



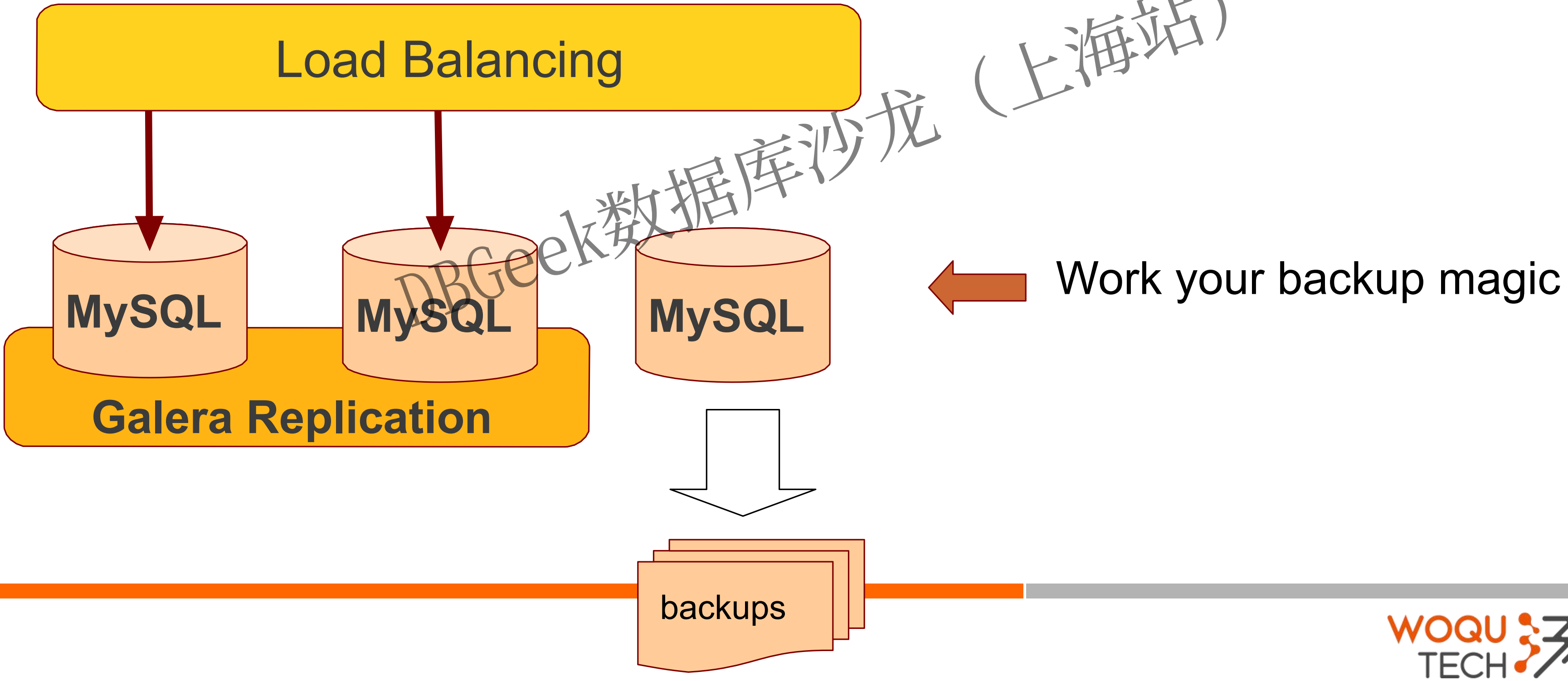
Disconnect from group
e.g. clear wsrep_provider

Backup by Disconnecting a Node

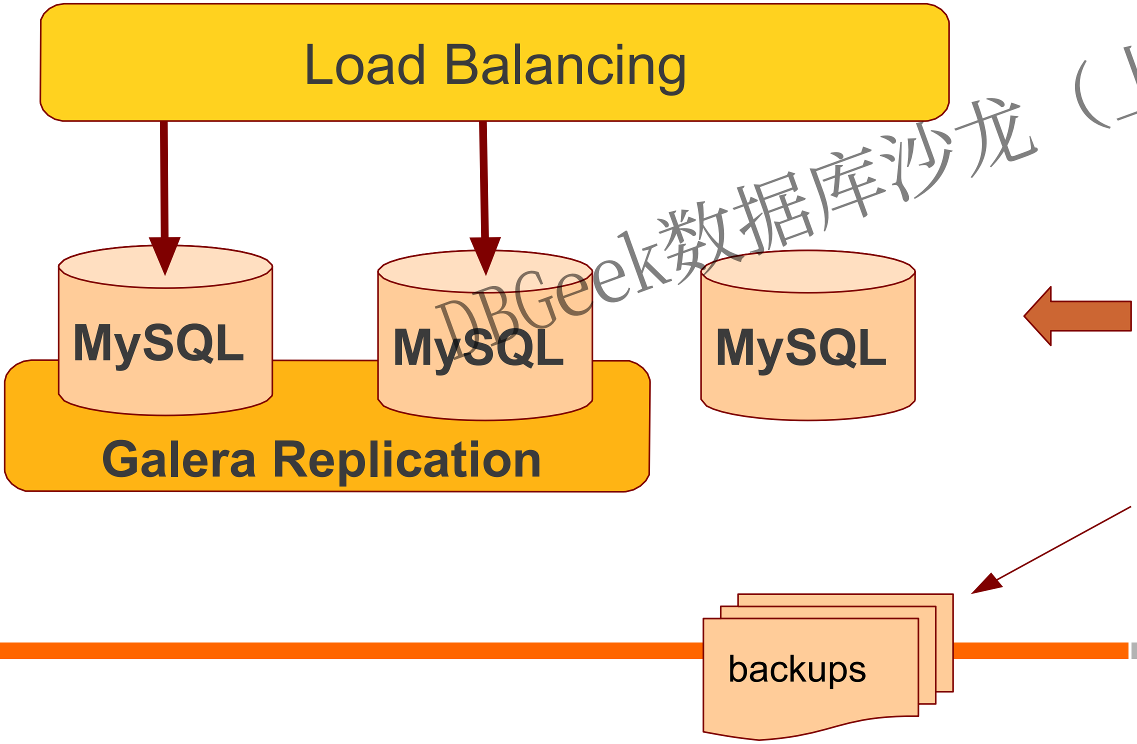


Disconnect from group e.g. clear wsrep_provider

Backup by Disconnecting a Node



Backup by Disconnecting a Node

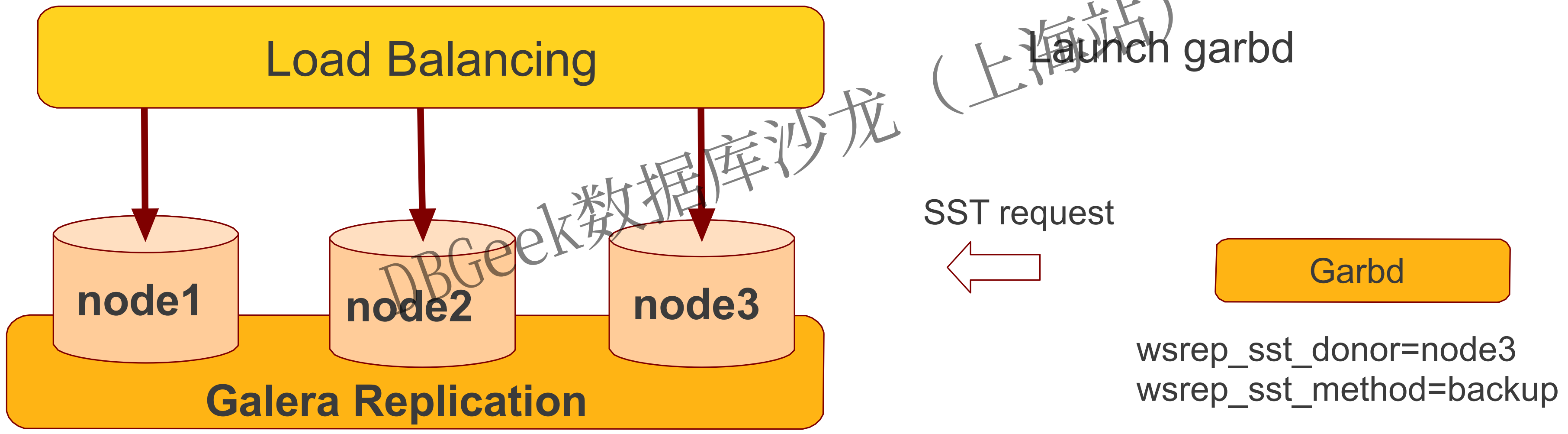


Read global transaction ID from status and assign to backup
wsrep_cluster_uuid
wsrep_last_committed

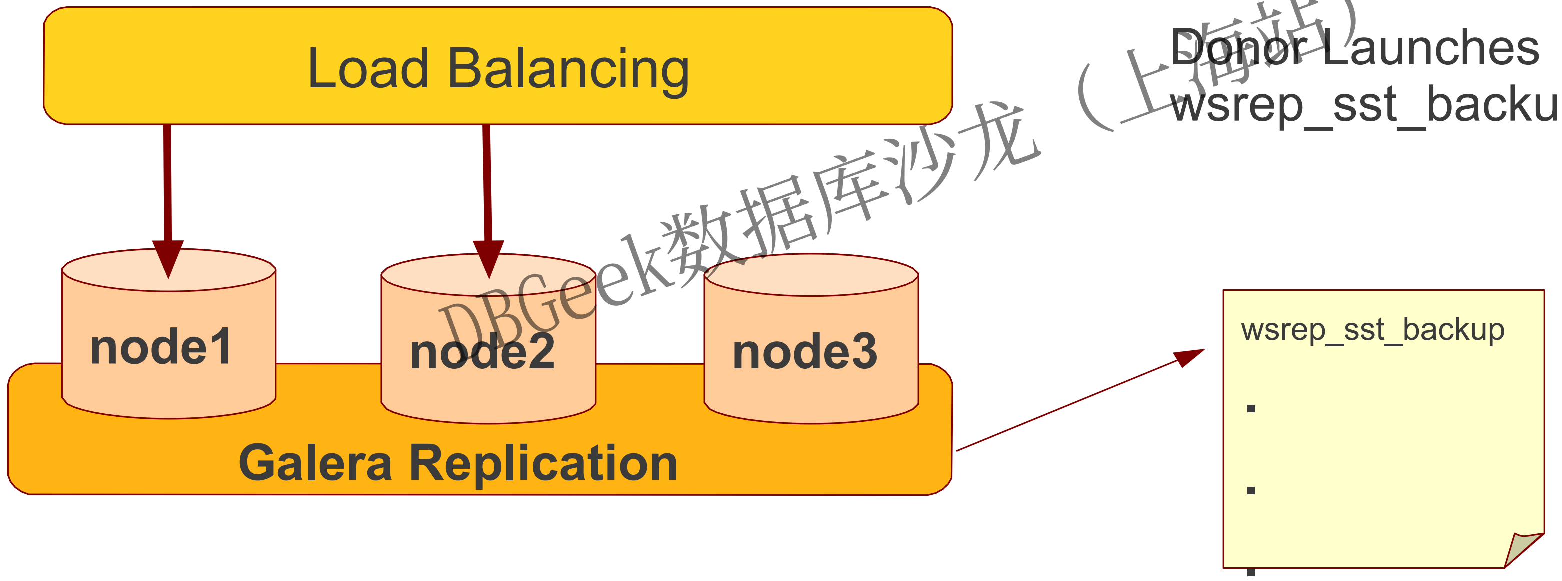
Backup by SST

- Donor mode provides isolated processing environment
- A special SST script can be written just to prepare backup in donor node: wsrep_sst_backup
- Garbd can be used to trigger donor node to run the wsrep_sst_backup

Backup by SST API

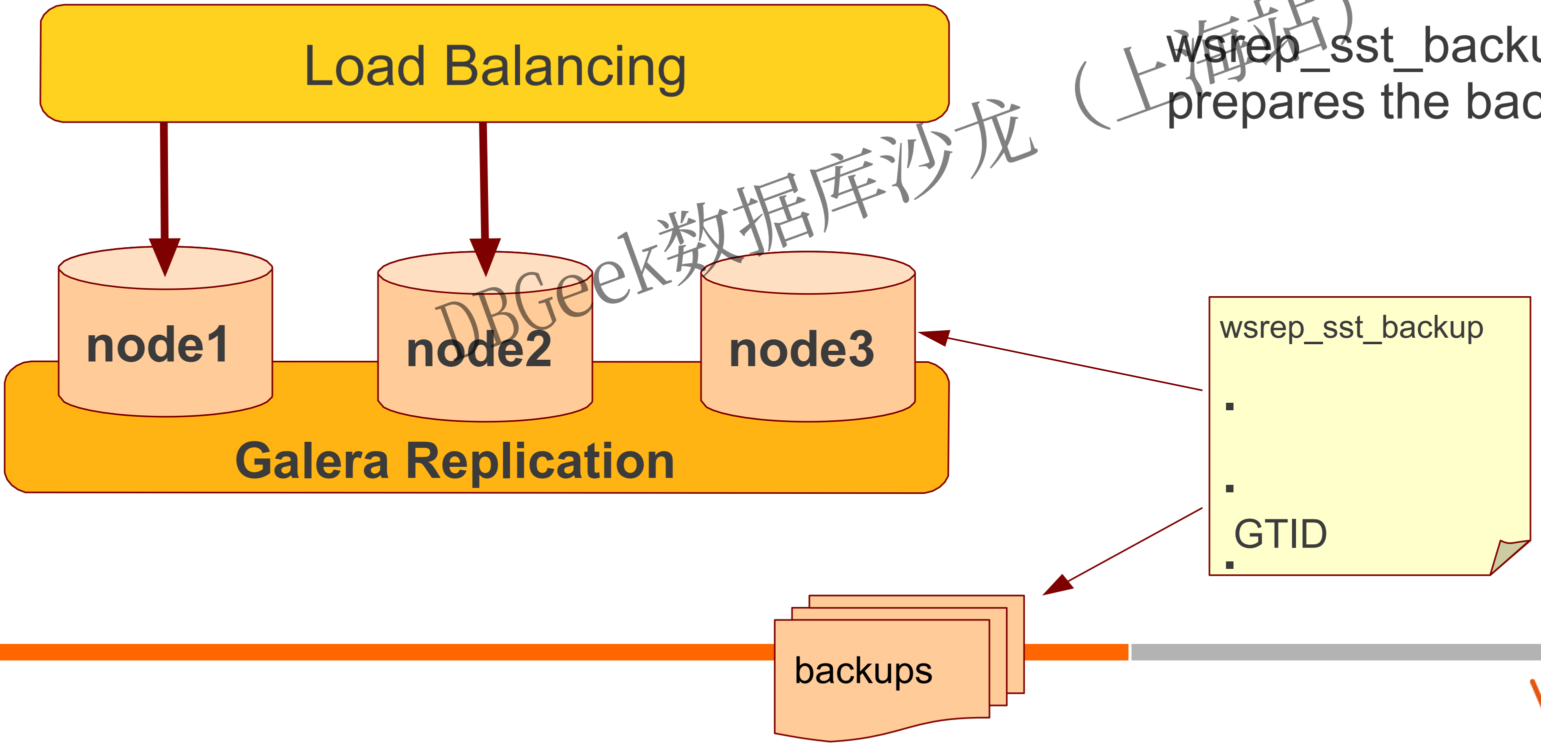


Backup by SST API



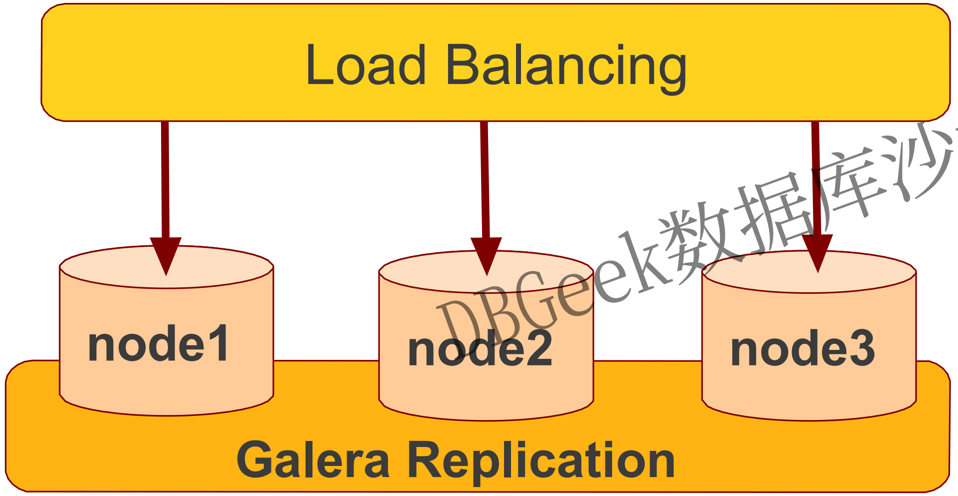
Donor Launches
wsrep_sst_backup

Backup by SST API



wsrep_sst_backup prepares the backup

Backup by SST API



Backup node returns to cluster

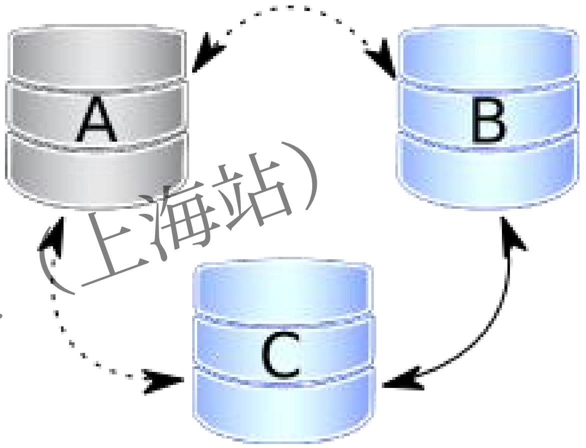
Backup by xtrabackup

- Xtrabackup is hot backup method and can be used anytime
- Simple, efficient
- Use –galera-info option to get global transaction ID logged into separate galera info file

Quorum and Availability of the cluster

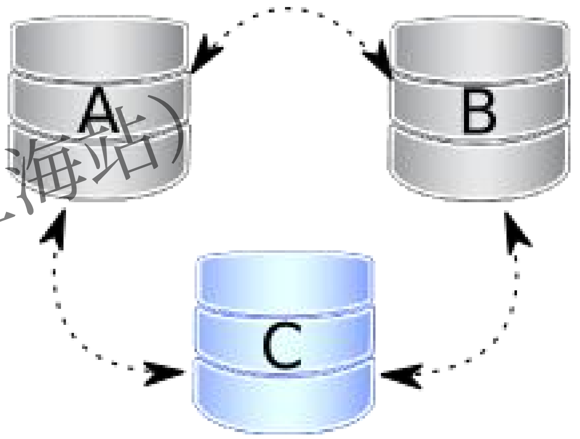
Nodes leaving gracefully

- Node A will instruct the other nodes that it is leaving the cluster.
- 2-node cluster and the remaining members have 2/2 = 100% of the votes. The cluster keeps running normally



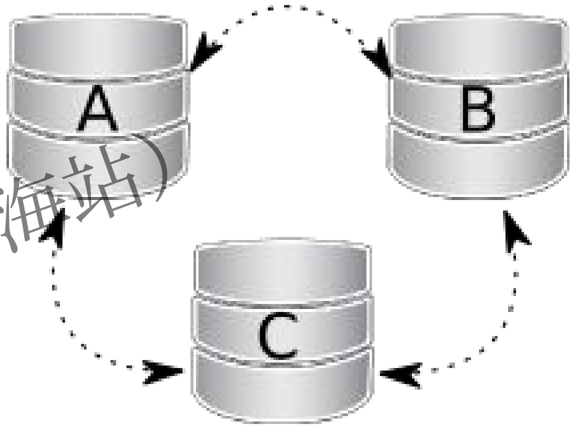
Nodes A and B are gracefully stopped

C will be switched to “Donor/Desynced”



All three nodes are gracefully stopped

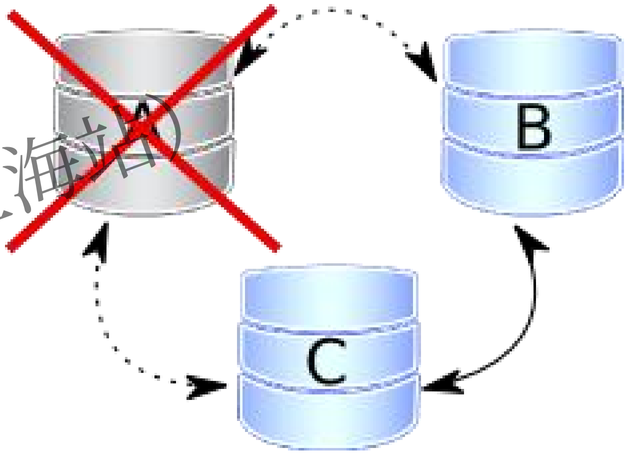
- PXC node writes it’s last executed position into the grastate.dat file
  - the most advanced one (most likely the last one stopped). Cluster must be bootstrapped using this node



Nodes becoming unreachable

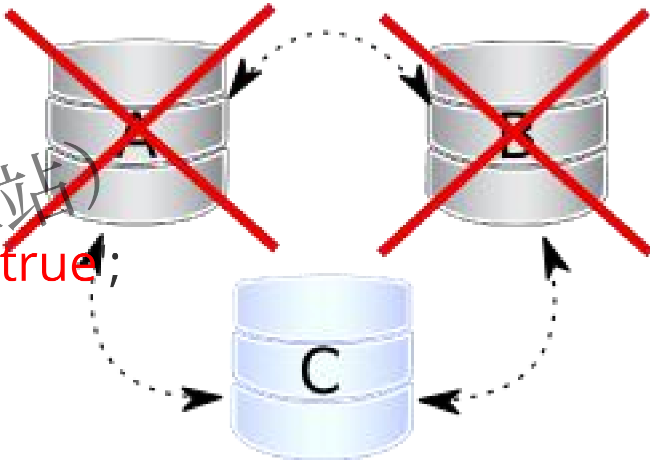
It only happens after the ‘suspect timeout’ (evs.suspect_timeout) which is 5 seconds by default.

- Node A disappears from the cluster
- wsrep_cluster_status it will show NON_PRIMARY

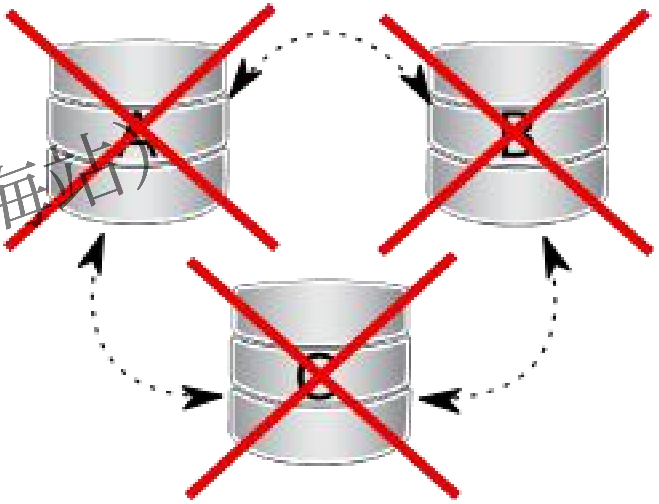


WOQU TECH 沃趣

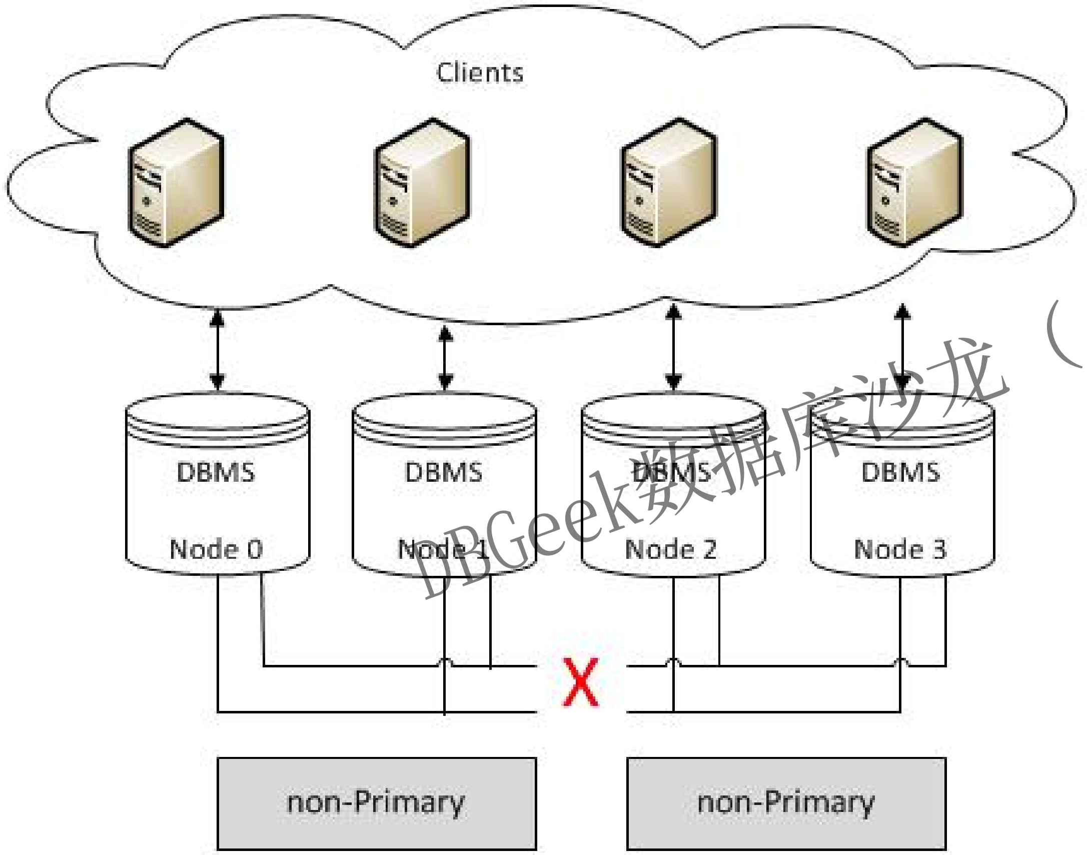
Nodes A and B disappear.
the cluster is switching into a non-primary mode
SET GLOBAL wsrep_provider_options='pc.bootstrap=true';



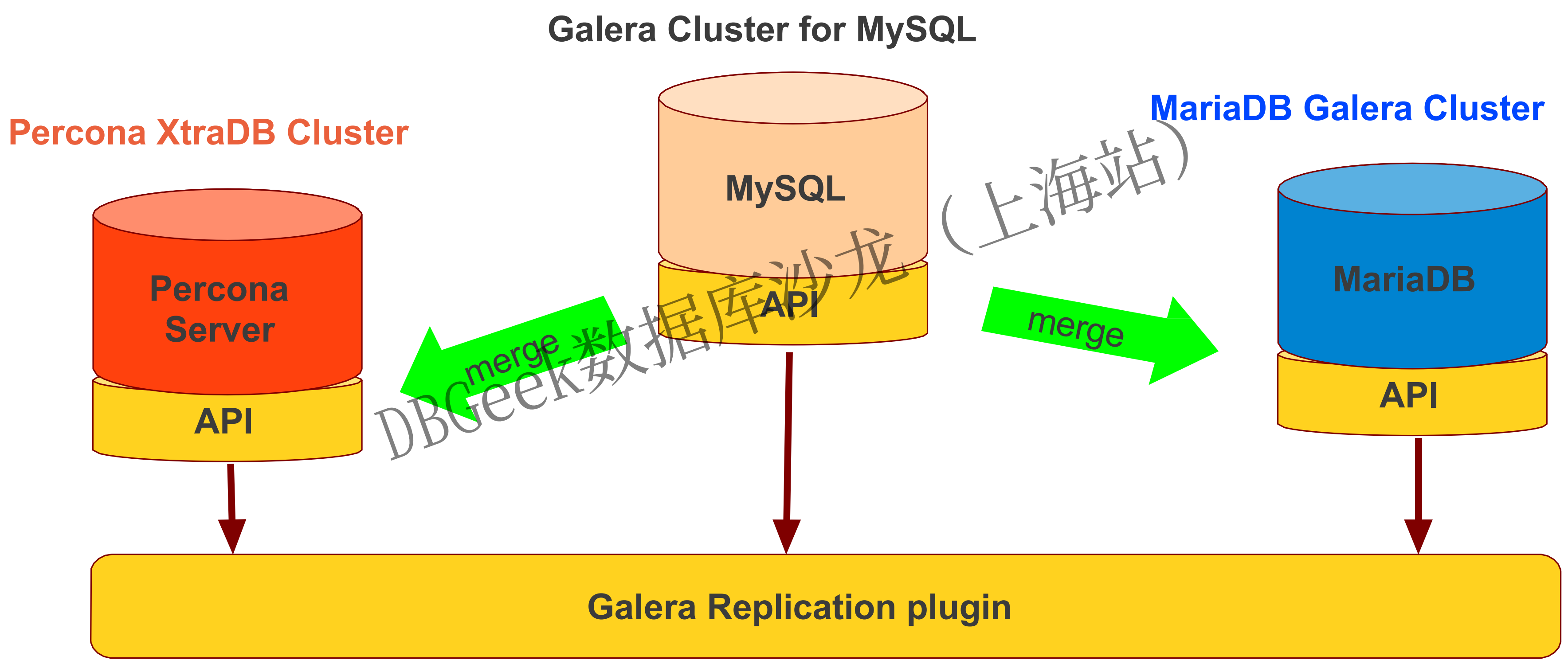
- All nodes went down without proper shutdown procedure
- grastate.dat file is not updated and does not contain valid sequence number (seqno)
- Mysqld_safe –wsrep_recover



Split Brain


Clients
DBMS
Node 0
DBMS
Node 1
DBMS
Node 2
DBMS
Node 3
X
non-Primary
non-Primary

Galera Project



沃趣

WOQUTECH

Let Data Drive!