ORACLE®

MySQL®

MySQL Cluster Deployment Best Practices

Johan ANDERSSON
MySQL Cluster practice Manager

Joffrey MICHAÏE
MySQL Consultant

The presentation is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions.
The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Agenda

- Cluster Setup
  - Minimal & Recommended Setup
  - Networking & Hardware Selection
- Disk Data Tables
- Configuration
- Administration
  - Online/Offline Operations
  - Backup and restore
- Monitoring

Minimal Setup

4 Computers having:

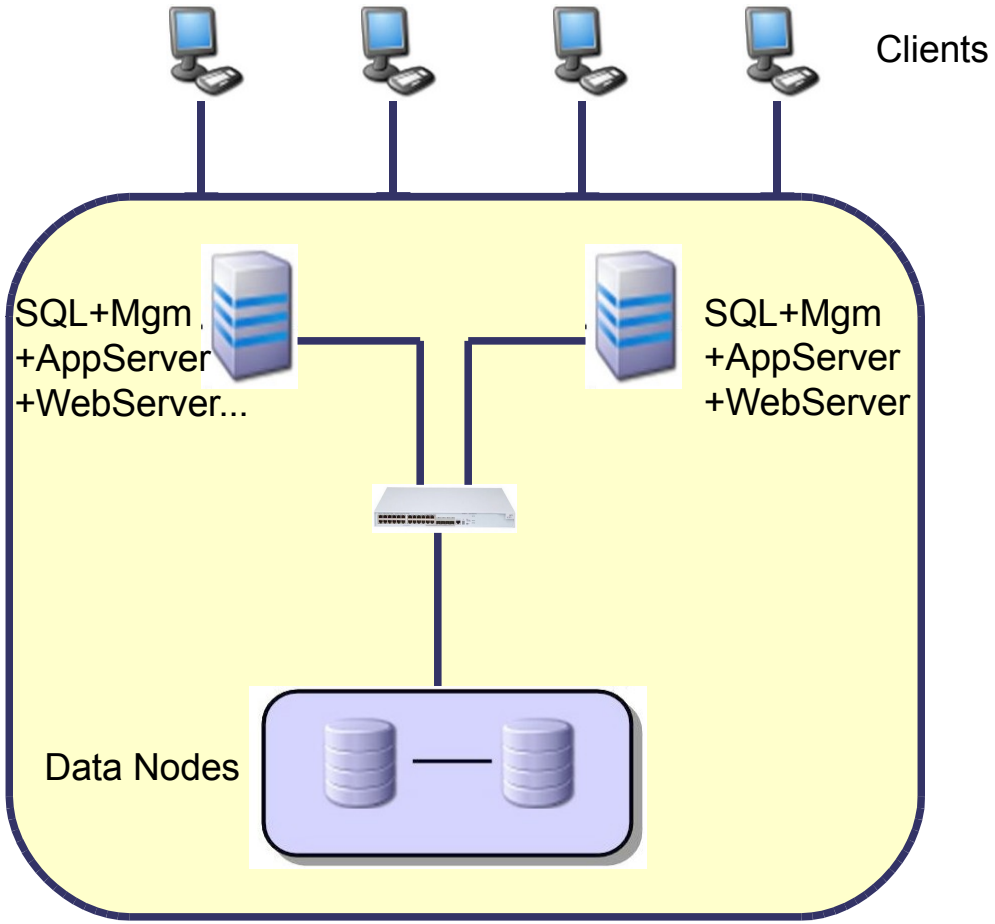
- 2 Data nodes
- 2 MySQL Servers
- 2 Management Nodes

Never:

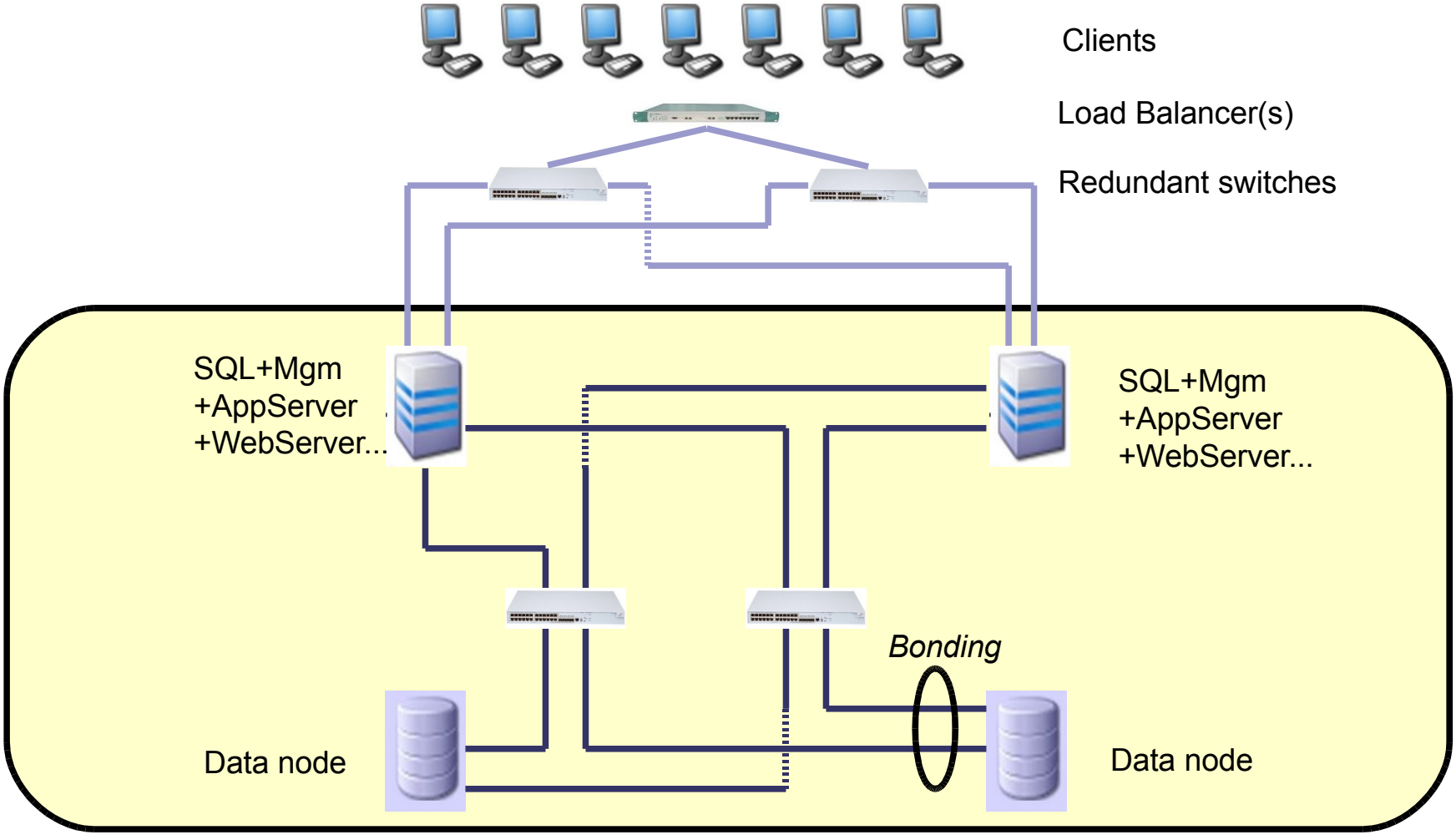
- Co-locate data node and management node! (split brain/network partitioning)

However :

- No redundant network
- SPOF
- No DRP



Recommended Setup



Networking

- Dedicated >= 1GB/s networking
- Prevent network failures (NIC x 2, Bonding)
- Use Low-latency networking (Dolphin...) if >= 8 data nodes or want higher throughput and lower latency
- Use dedicated network for cluster communication
- No security layer to management node (remote shutdown allowed ....)
- Enable port 1186 access only from cluster nodes and administrators

Hardware Selection - RAM & CPU

- Storage Layer (data nodes)
  - One data node can (7.0+) use 8 cores
  - **CPU**: 2 x 4 core (Nehalem works really well). Fast CPU → fast processing of messages.
  - **RAM**: As much as you need
    - a 10GB data set will require 20GB of RAM (because of redundancy
    - Each node will then need 2 x 10 / #of data nodes. (2 data nodes → 10GB of RAM → 16GB RAM is good
  - **Disk space:** 10xDataMemory + space for BACKUP + TableSpace (if disk data tables)
- SQL Layer (MySQL Servers)
  - **CPU**: 2 – 16 cores
  - **RAM**: Not so important – 4GB enough (depends on connections and buffers)

Hardware Selection - Disk Subsystem

low-end	mid-end	high-end
		
1 x SATA 7200RPM • For a read-most, write not so much • No redundancy (but other data node is the mirror)	1 x SAS 10KRPM • Heavy duty (many MB/s) • No redundancy (but other data node is the mirror)	4 x SAS 10KRPM • Heavy duty (many MB/s) • Disk redundancy (RAID1+0) hot swap

- REDO, LCP, BACKUP – written sequentually in small chunks (256KB)
- If possible, use Odirect = 1

Filesystem

- Most customers uses EXT3(Linux) and UFS (Solaris)
  - Ext2 could be an option (but recovery is longer)
- XFS – we haven't experienced so much...
- ZFS
  - You must separate journal (Zil) and filesystem
- Mount with noatime
- Raw device is not supported

Hardware Selection - Disk Data Storage

Minimal recommended

LCP
REDOLOG
UNDOLOG

TABLESPACE

2 x SAS 10KRPM (preferably)

high-end

UNDOLOG
(REDO LOG)

TABLESPACE 1

TABLESPACE 2

(REDO LOG / UNDO LOG)
LCP

4 x SAS 10-15KRPM (preferably)

- Use High-end for heavy read write (1000's of 10KB records per sec) of data (e.g Content Delivery platforms)
- SSD for TABLESPACE is also interesting – not much experience of this yet
- Having TABLESPACE on separate disk is good for read perf.
- Enable WRITE_CACHE on devices

Configuration - Disk Data Storage

- Use Disk Data tables for
  - Simple accesses (read/write on PK)
  - Same for innodb – you can easily get DISK BOUND (iostat)
- Set
  - `DiskPageBufferMemory=3072M`
    - is a good start if you rely a lot on disk data – like the Innodb_Buffer_Pool, but set it as high as you can!
    - Increased chance that a page will be cached
  - `SharedGlobalMemory=384M-1024M`
  - `UNDO_BUFFER=64M to 128M` (if you write a lot)
    - You cannot change this BUFFER later!
    - Specified at LOGFILE GROUP creation time
  - `DiskIOThreadPool=[ 8 .. 16 ]` (introduced in 7.0)

Configuration - General

- Set
  - `MaxNoOfExecutionThreads<=#cores`
    - Otherwise contention will occur → unexpected behaviour.
  - `RedoBuffer=32-64M`
    - If you need to set it higher → your disks are probably too slow
  - `FragmentLogFileSize=256M`
  - `NoOfFragmentLogFiles= 6 x DataMemory (in MB) / (4x 256MB)`
    - Most common issue – customers never configure big enough redo log
- The above parameters (and others, also for MySQL) are set with
  - www.severalnines.com/config

Application : Primary Keys

- To avoid problems with
  - Cluster 2 Cluster replication
  - Recovery
  - Application behavior (KEY NOT FOUND.. etc)
- ALWAYS DEFINE A PRIMARY KEY ON THE TABLE!
  - A hidden PRIMARY KEY is added if no PK is specified. BUT..
    - .. NOT recommended
    - The hidden primary key is e.g not replicated (between Clusters)!!
    - There are problems in this area, so avoid the problems!
- So always, at least have
  `id BIGINT AUTO_INCREMENT PRIMARY KEY`
  - Even if you don't “need” it for you applications

Application : Query Cache

- Don't cache everything in the Query Cache
  - It is very expensive to invalidate over X mysql servers
  - A write on one server will force the others to purge their cache.
- If you have tables that are read only (or change very seldom):
  - my.cnf:
    - `query_cache_type=2 (DEMAND)`
  - `SELECT SQL_CACHE <cols> .. FROM table;`
- This can be good for STATIC data

Application : Transactions

- Failed transactions must be retried by the application
- If the REDOLOG or REDOBUFFER is full the transaction will be aborted
  - This differs from INNODB behaviour
- There are also other resources / timeouts
  - `"Lock wait timeout"` – transaction will abort after `TransactionDeadlockDetectionTimeout`
  - `MaxNoOfConcurrent[Operations/Transactions]`
    - You can then increase this
- Nodefail/noderestart will cause transaction to abort

Application : Transactions

- Transactions (large updates)
  - Remember NDB is designed for many and short transactions
  - You are recommended to UPDATE / DELETE in small chunks
  - Use LIMIT 10000 until all records are UPDATED/DELETED
- MaxNoOfConcurrentOperations sets the upper limit for how many records than can be modified simultaneously on one data node.
  - MaxNoOfConcurrentOperations=1000000 will use 1GB of RAM
  - Despite it is possible, we do recommend DELETE/UPDATE in smaller chunks.

Application : Table locks

- FLUSH TABLE WITH READ LOCK;
- LOCK TABLES <table> READ;
  - Only locks the table(s) on the LOCAL mysql server
  - You must get the LOCK on all mysql servers

Application : Schema Operations

- Don't use too much `CREATE/DROP TABLE` of NDB tables
  - It is a heavy operation within Cluster
  - Takes much longer than with standard MySQL

Log Only What Needs To Be Recovered

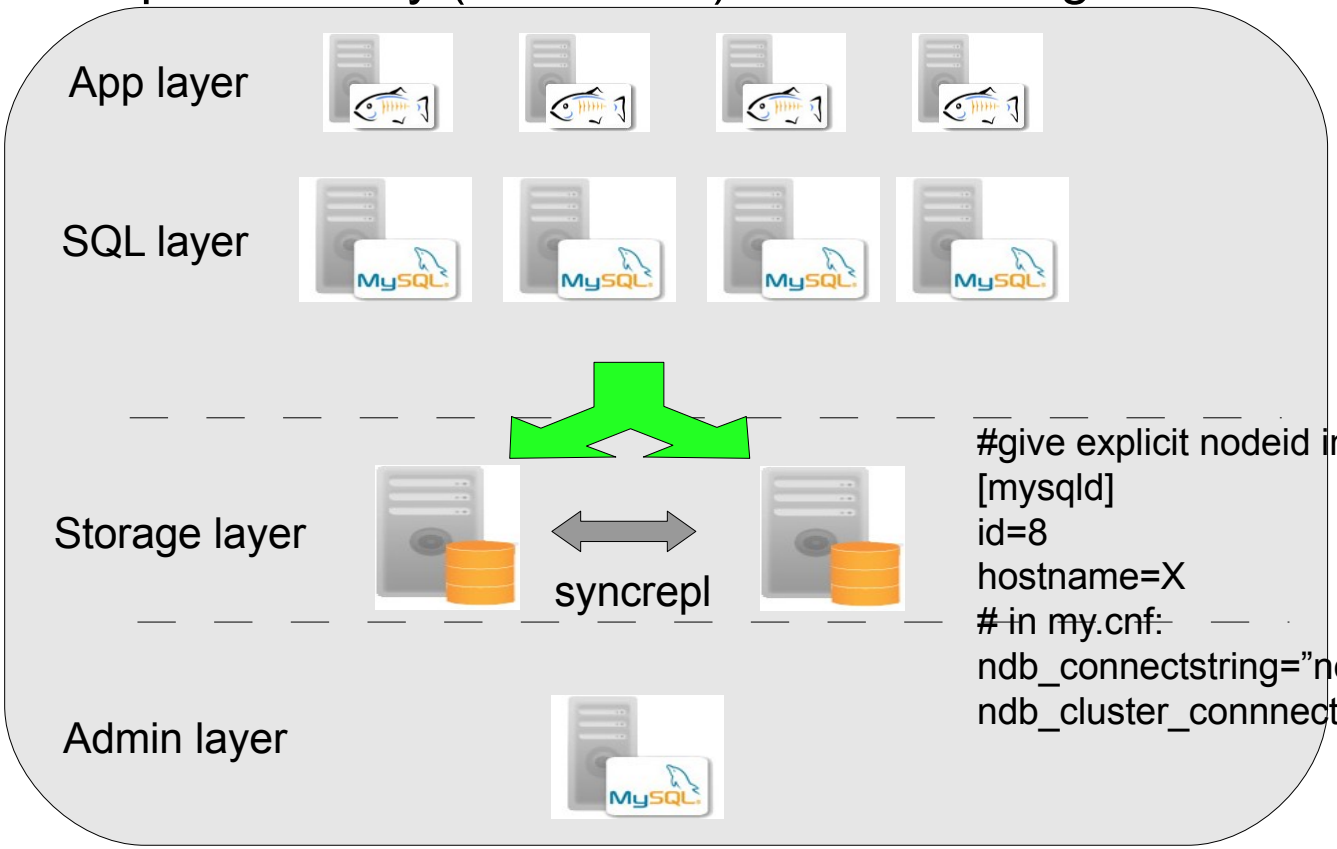
- Some types of tables account for a lot of WRITEs, but does not need to be recovered (E.g, Session tables)
- A session table is often unnecessary to REDO LOG and to CHECKPOINT
- Create these tables as 'NO LOGGING' tables:

```
mysql> set @ndb_curr_val=@@ndb_table_no_logging;

mysql> set ndb_table_no_logging=1;

mysql> create table session_table(..) engine=ndb;

mysql> set ndb_table_no_logging=@ndb_curr_val;
```

- 'session_table' will not be
  - REDO logged or Checkpointed → No disk activity for this table!
- After System Restart it will be there, but empty!

Administration Layer

- Introduce a MySQL Server for administration purposes!
  - Should never ever get application requests
  - Simplifies heavy (non online) schema changes

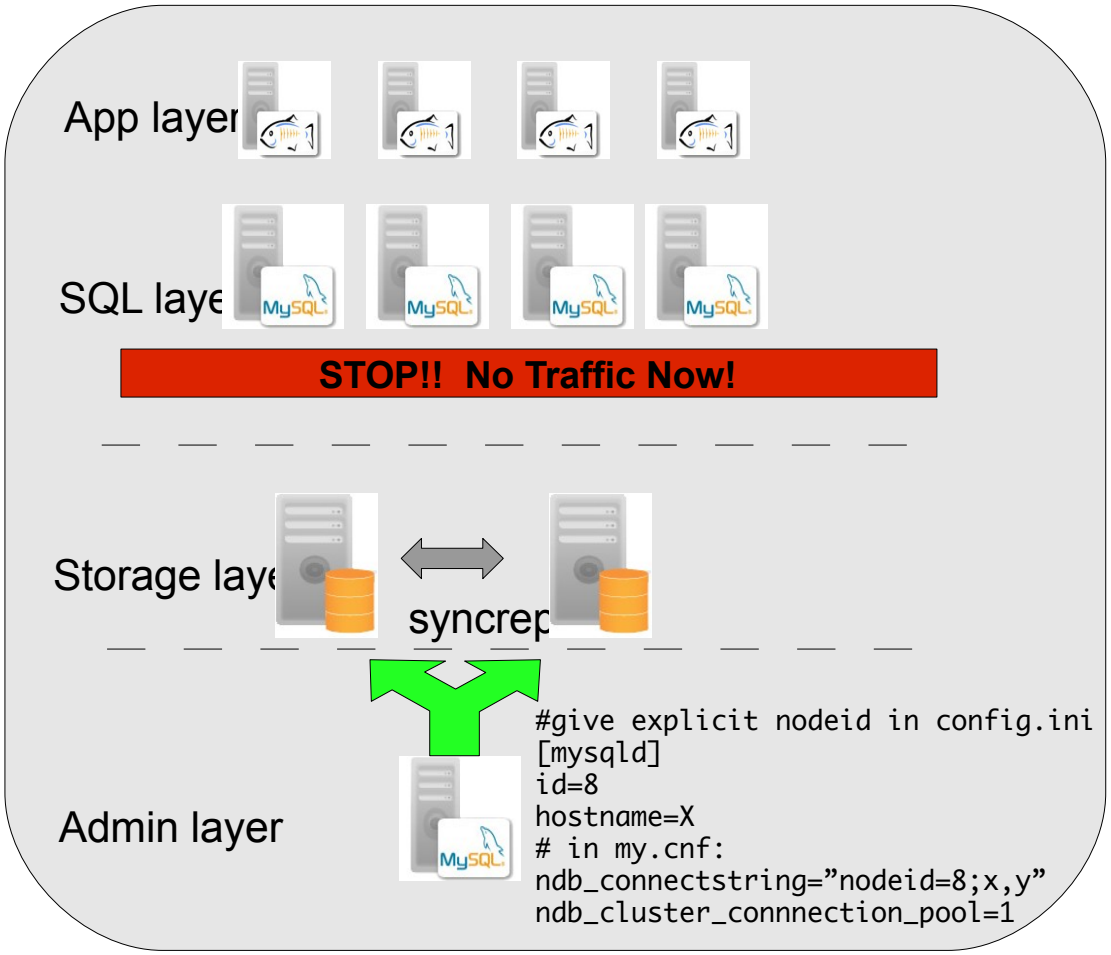


Administration Layer

- Modifying Schema is NOT online when you do:
  - Rename a table
  - Change data type
  - Change storage size
  - Drop column
  - Rename column
  - Add/Drop a PRIMARY KEY
- Altering a 1GB table requires 1GB of free DataMemory (copying)
- Online (and ok to do with transactions ongoing):
  - Add column (ALTER ONLINE …)
  - CREATE INDEX
  - Online add node (see my presentation from last year how to do it)

Admistration Layer

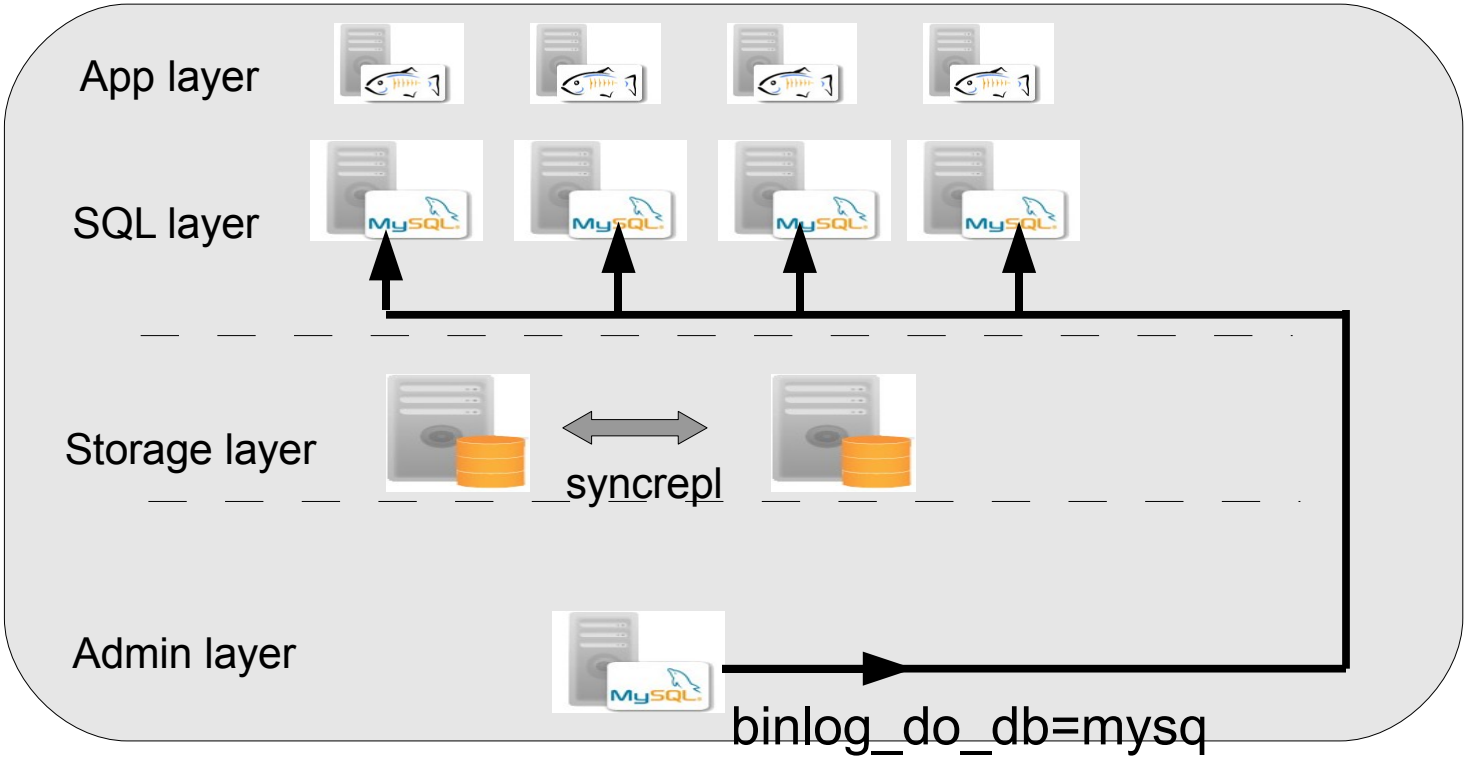
- ALTER TABLE etc (non-online DDL) performed on Admin Layer!



- 1. Block traffic from SQL layer to data nodes
  - ndb_mgm> ENTER SINGLE USER MODE 8
  - Only Admin mysqld is now connected to the data nodes
- 2. Perform heavy ALTER on admin layer
- 3. Allow traffic from SQL layer to data nodes
  - ndb_mgm> EXIT SINGLE USER MODE

Admistration Layer

- You can also set up MySQL Replication from Admin layer to the SQL layer
  - Replicate mysql database
    - GRANT, SPROCs etc will be replicated.
  - Keeps the SQL Layer aligned¨



Online Upgrades

- Change Online
  - OS, SW version (7.0.x → 7.1.x)
  - configuration
    - E.g, incease DM, IM, Buffers, redo log, [mysqld] slots etc
  - hardware (upgrade more RAM etc)
- These procedures requires a Rolling Restart
  - Change config.ini, copy it over to all ndb_mgmd
  - Stop BOTH ndb_mgmd, start BOTH ndb_mgmd
  - Restart one data node at a time
  - Restart one mysqld at a time
- Adding data nodes (from 7.0)
  - See my presentation from Last Year (UC 2009)
- Adding MySQL Servers
  - Make sure you have free [mysqld] slots
  - Start the new mysqld

Backup

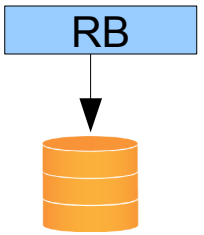
- Backup of NDB tables
  - Online – can have ongoing transactions
  - Consistent – only committed data and changes are backed up
  - `ndb_mgm -e "START BACKUP"`
  - Copy backup files from data nodes to safe location
- Non-NDB tables must be backed up separately
  - Mysql system tables are stored only in MYISAM.
  - You want to backup (for each mysql server)
    - mysql database
    - Triggers, SP ...
  - Use 'mysqldump'
    - `mysqldump mysql > mysql.sql`
    - `mysqldump --no-data --no-create-info -R > routines.sql`
  - Copy my.cnf & config.ini files

Restore

- ndb_restore is in many cases the MOST write intensive operation on Cluster
  - The problem is that ndb_restore produce REDO LOG
  - This is unnecessary but a fact for now
  - Restores many records in parallel, no throttling..
    - So 128 or more small records may be fine, but 128 BLOBs .

Temporary error: 410: REDO log buffers overloaded, consult online manual (increase RedoBuffer, and|or decrease TimeBetweenLocalCheckpoints, and|or increase NoOfFragmentLogFiles)

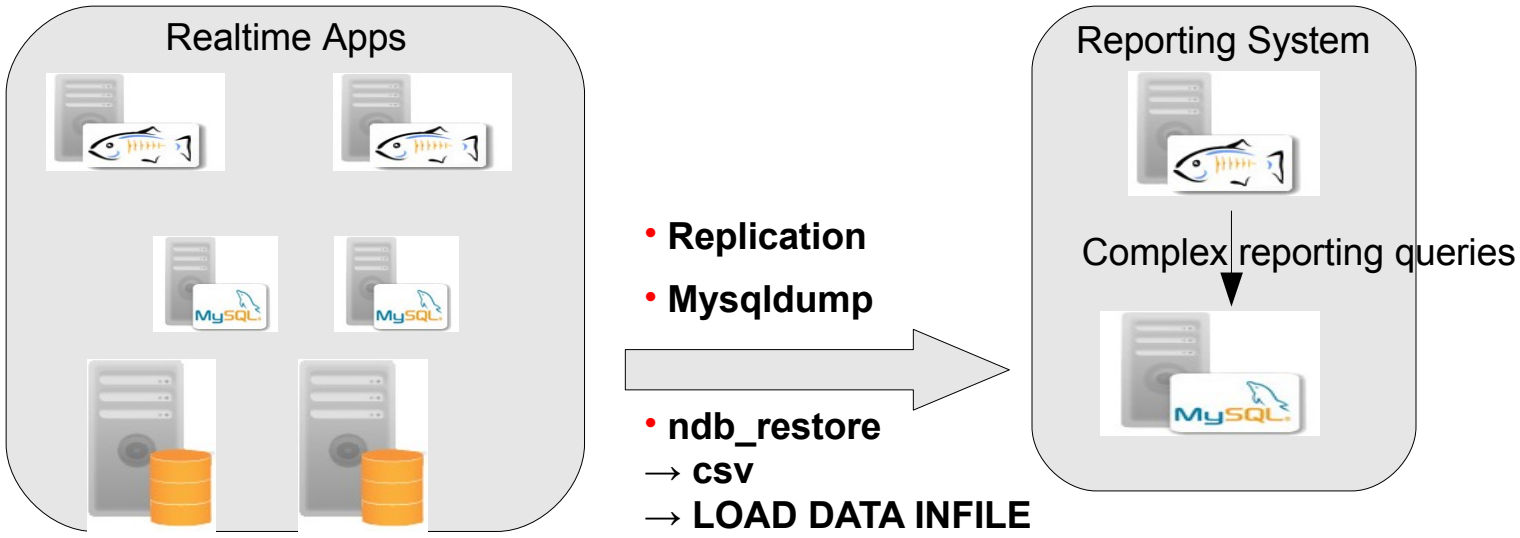
- If you run into this during restore
  - Try increase RedoBuffer (a value of higher than 64MB is seldom practical nor needed)
  - Run only one *instance* of ndb_restore
  - ndb_restore -p10 ....
    - Or even a lower value, e.g, -p1
  - If this does not help → faster disk(s) is/are needed



Synced every TBGCP

Separate Realtime and Reporting

- MySQL Cluster is really good for
  - Short, many parallel transactions
  - Is bad at most reporting type queries (complex JOINs).
- For reporting, replicate/copy out data to a reporting server more suitable for the job
  - MySQL Server running Innodb
  - For replication see http://johanandersson.blogspot.com/2009/05/ha-mysql-write-scaling-using-cluster-to.html



Scaling

- One data node can (7.0+) use up to 8 cores
  - CPU: Reaches bottleneck at about 370% CPU
  - DISK: iostat -kx 1 : Check util; await, svctime etc..
  - NETWORK: iftop (linux) - add another data node (spread load)
  - Add data node (online or offline)
- MySQL Server
  - CPU: About the same – 300-500%
  - DISK: Should not be a factor
  - NETWORK:
  - Add another MySQL Server to offload query processing

Monitoring

- Mandatory to monitor
  - CPU/Network/Memory usage
  - Disk capacity (I/O) usage
  - Network latency between nodes
  - Node status ...
  - Used Index/Data Memory
- www.severalnines.com/cmon- monitors data nodes and mysql servers
- New in 7.1 :
  - NDB$INFO Table in INFORMATION_SCHEMA
  - Check node status
  - Check buffer status etc
  - Statistics

Questions?

- Talk to us outside! We offer cheap consulting this week:
  - One question one beer.
- Go to the other sessions :

MySQL Cluster Performance Tuning Best Practices
2.00pm Wednesday 04/14/2010
Location: Ballroom D

MySQL Cluster and Pushdown-joins (In Pursuit of the Holy Grail)
Jonas Oreland
3.05pm Wednesday 04/14/2010
Location: Ballroom B


ORACLE®