



# MySQL Cluster Tutorial

MySQL Conference & Expo 2011

Max Mether <[max@skysql.com](mailto:max@skysql.com)>

Joffrey Michaie <[joffrey@skysql.com](mailto:joffrey@skysql.com)>

Johan Andersson <[johan@severalnines.com](mailto:johan@severalnines.com)>

# Who are we?



- Max Mether
  - Trainer and Consultant at MySQL from 2001
  - Curriculum Manager at MySQL
  - Training Manager at SkySQL from 2010
- Johan Andersson
  - Cluster Practice Manager at MySQL from 2003
  - Consultant at Severalnines from 2010
- Joffrey Michaie
  - Cluster Consultant at MySQL from 2009
  - Consultant at SkySQL from 2010



# Part 1

## Introduction



# Cluster Use Cases



- What is cluster used for?
  - Telecom applications
  - Online Gaming
  - Financial Applications
  - eCommerce
  - Session Management

# Cluster Usage



What are/will you using the cluster for??

# Features



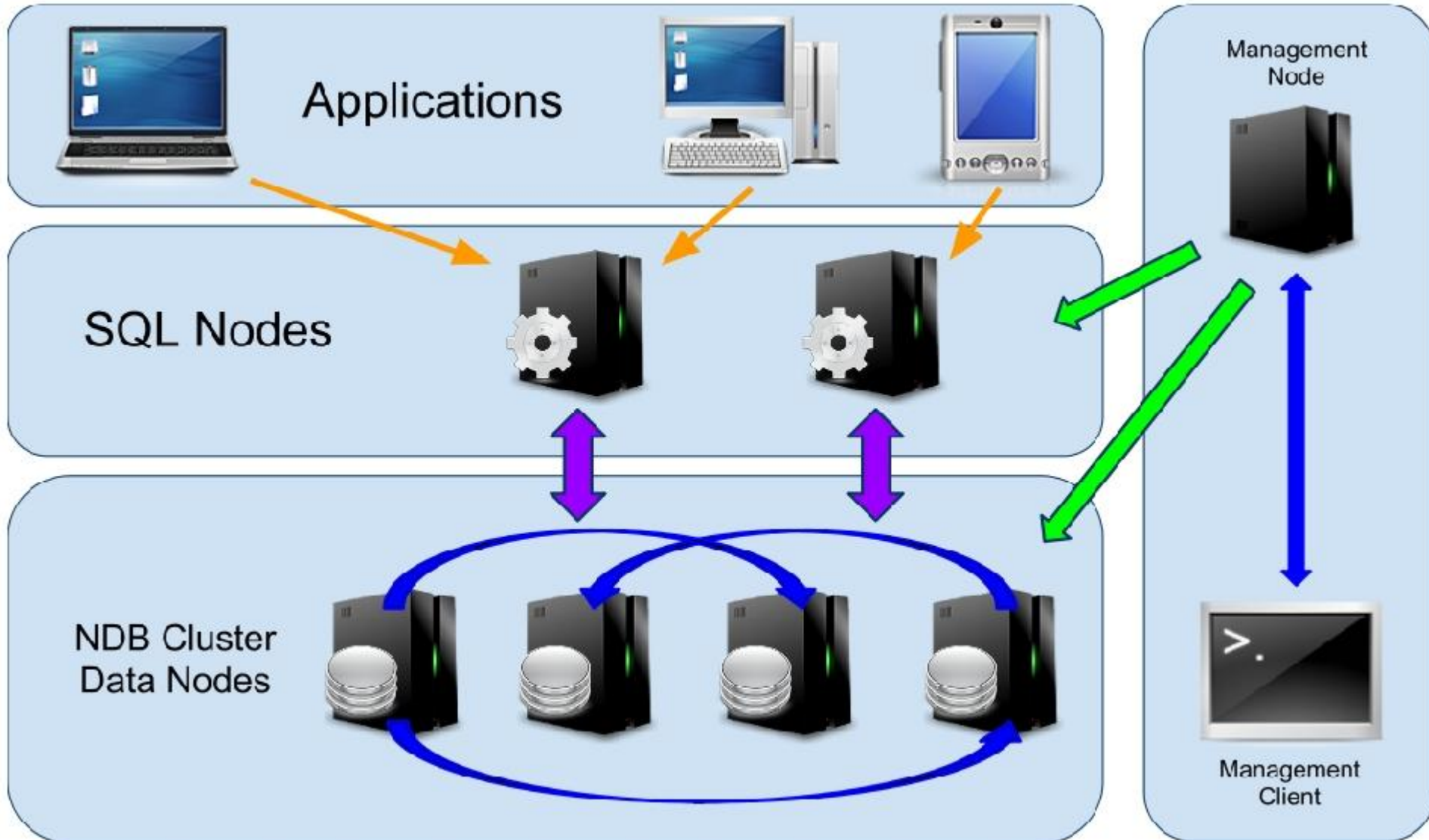
- Shared nothing architecture
  - No single point of failure
- Synchronous replication between nodes
- ACID transactions
- Row level locking

# Features



- In-memory storage
  - Some data can be stored on disk
  - Checkpointing to disk for durability
- Two types of indexes
  - Ordered T-trees
  - Unique hash indexes
- Online operations
  - Add node groups
  - Software upgrade
  - Some table alterations

# Architecture



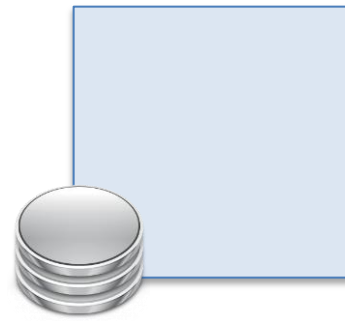
# Partitioning

Table


Node 3



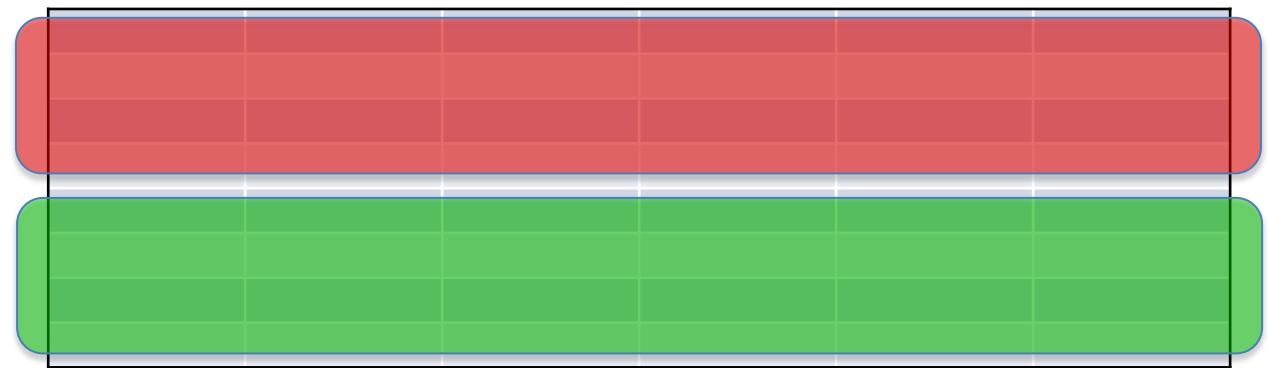
Node 4



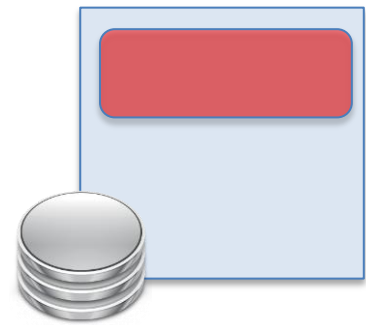
# Partitioning



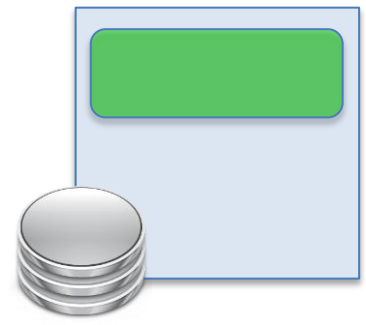
Table



Node 3

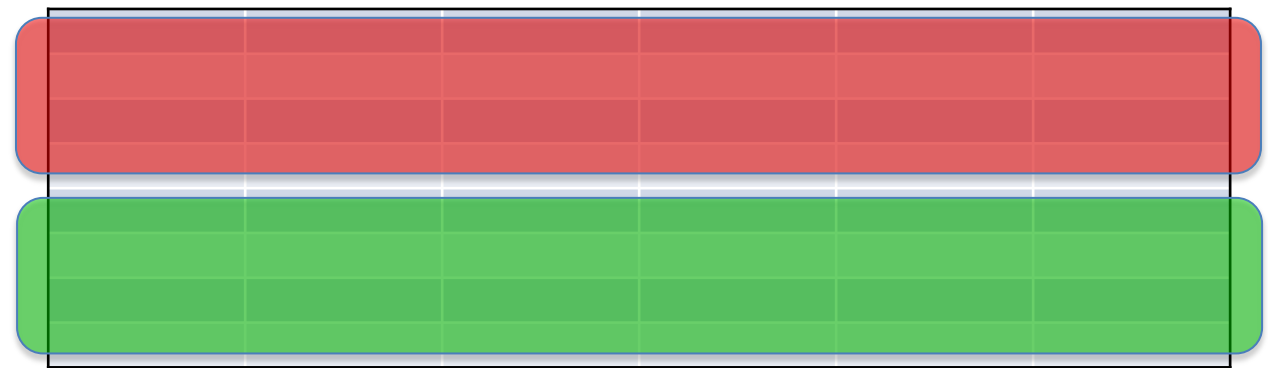


Node 4

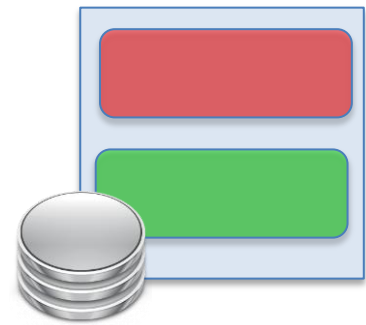


# Partitioning

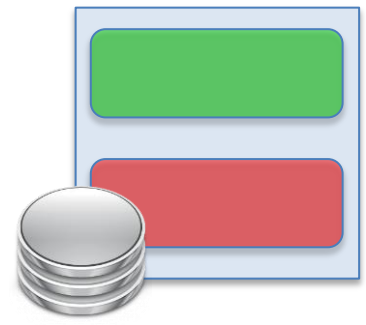
Table



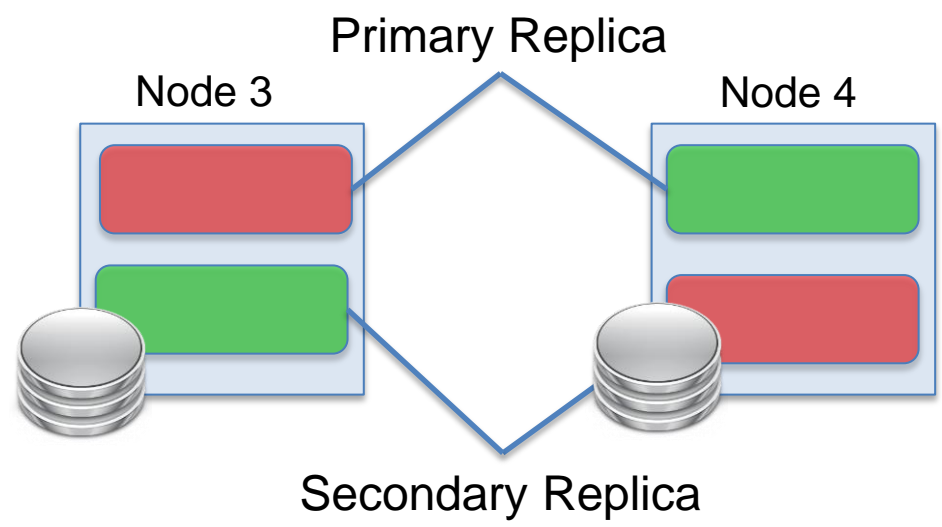
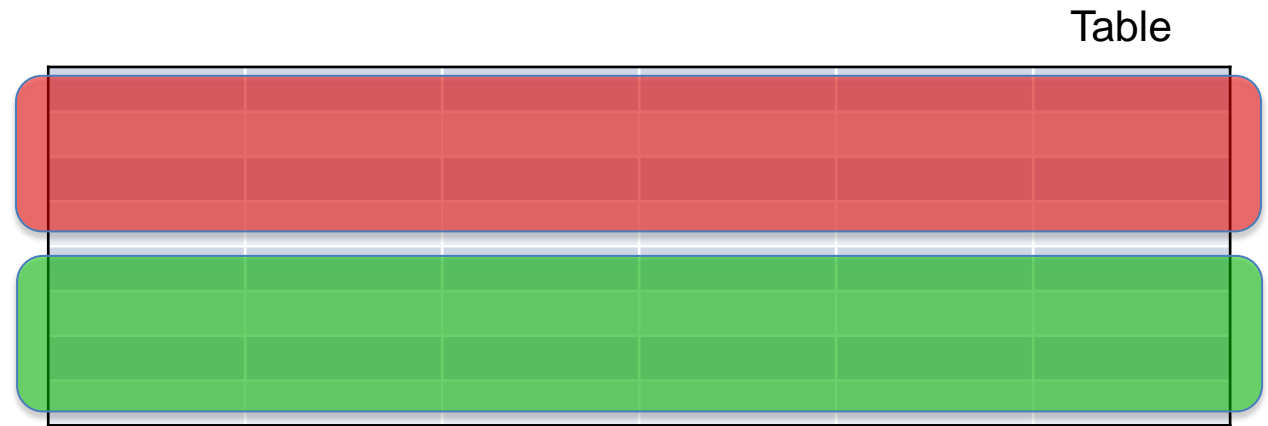
Node 3



Node 4



# Partitioning



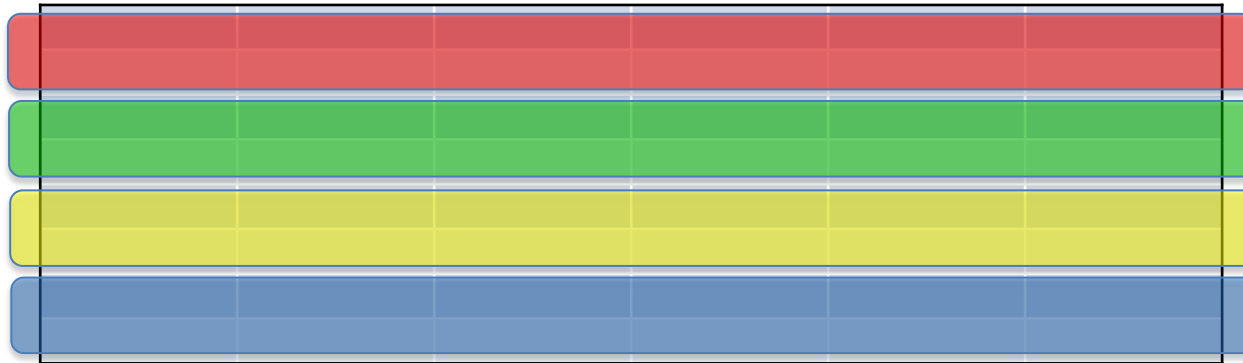
# Partitioning - 4 Data Nodes

Table




# Partitioning - 4 Data Nodes

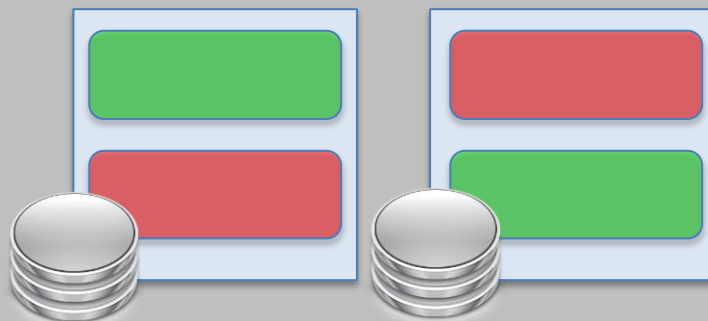
Table



## Node Group

Node 3

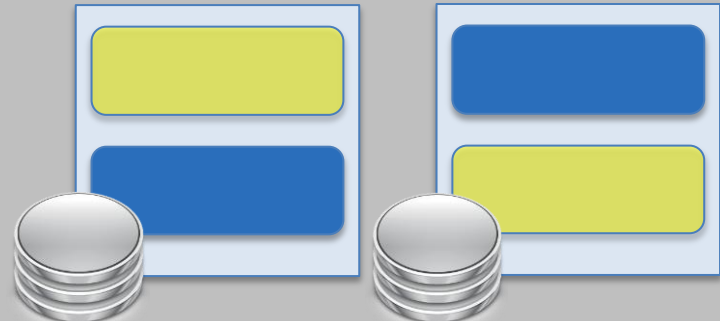
Node 4



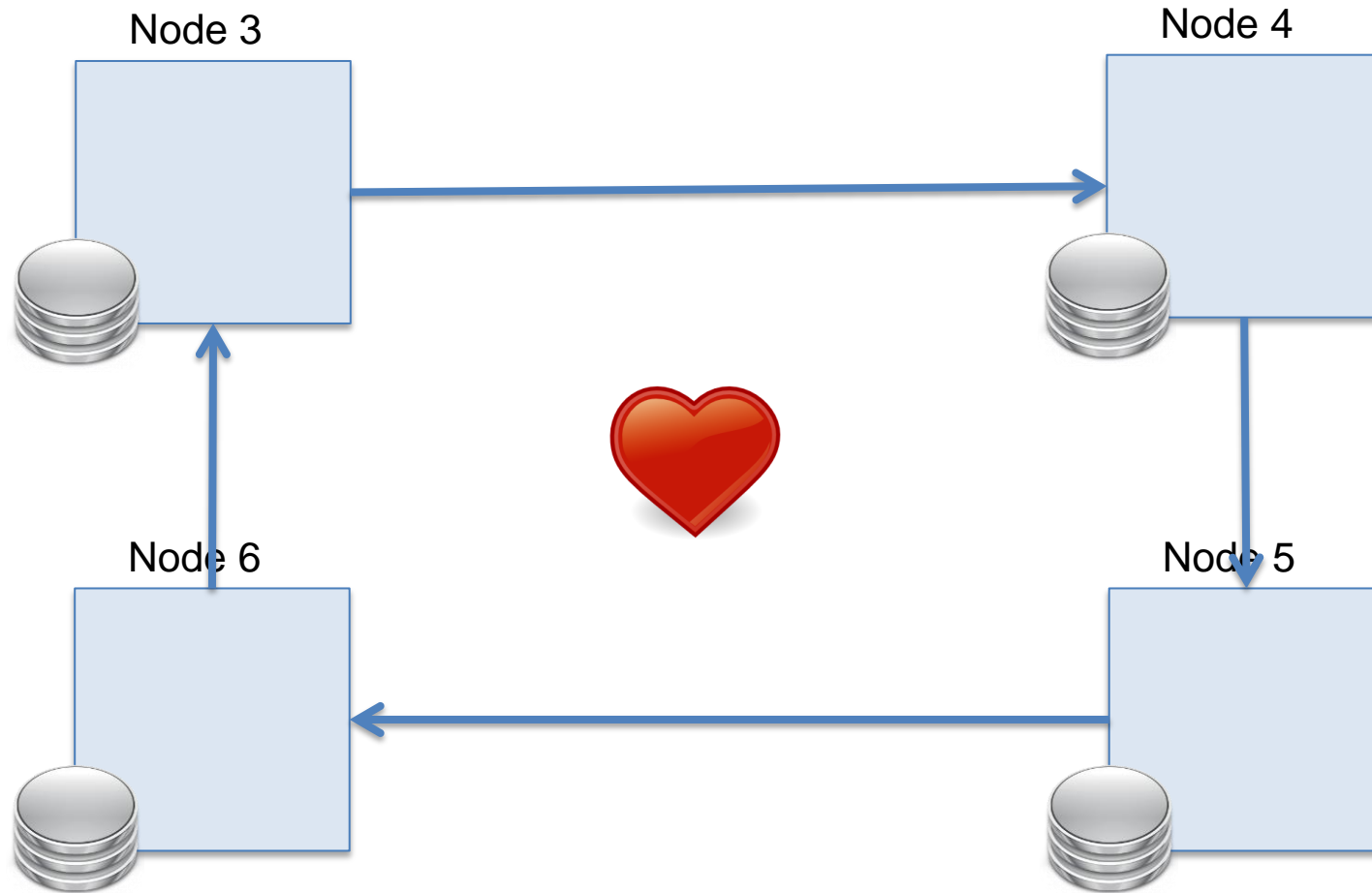
## Node Group

Node 5

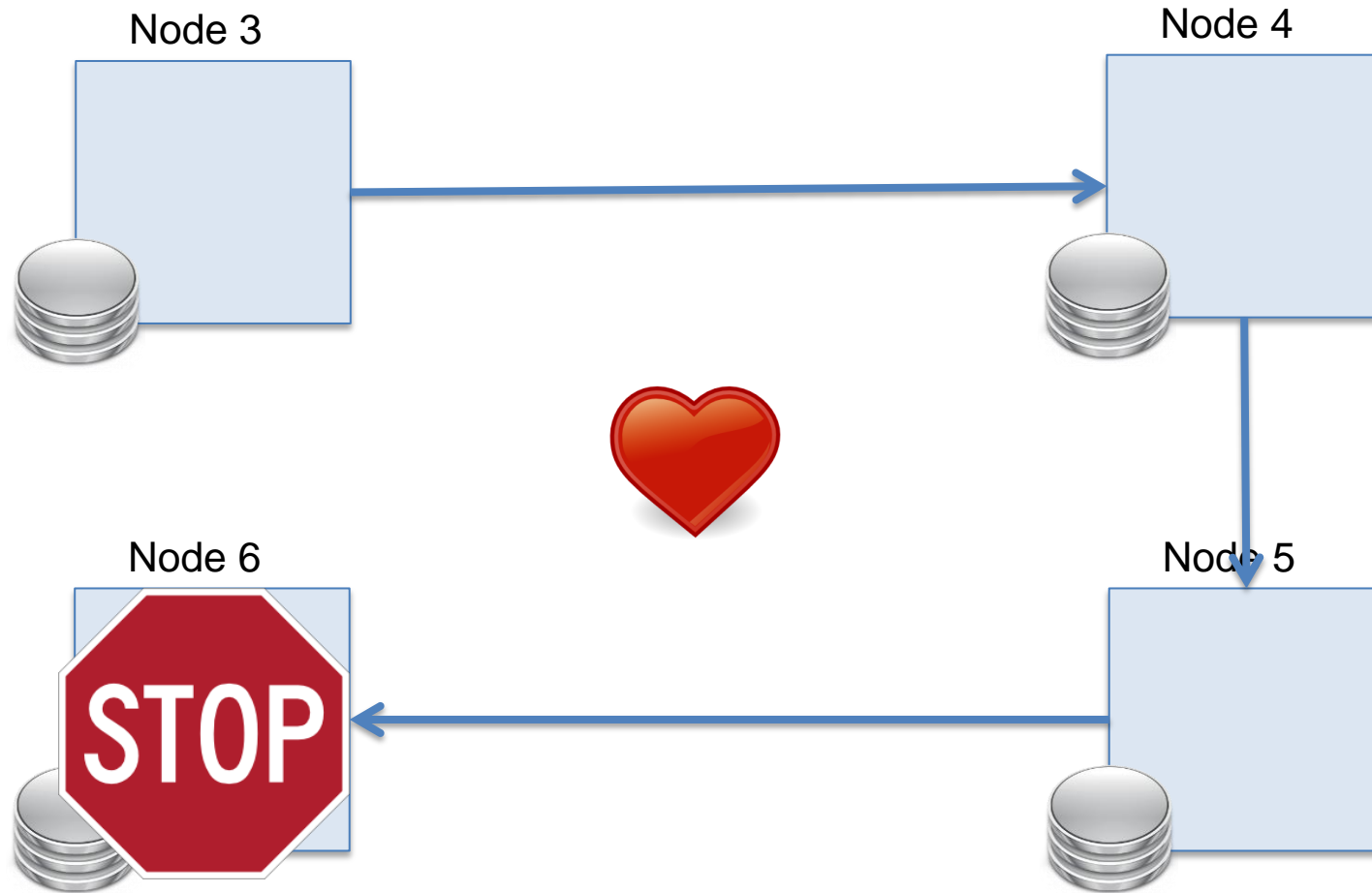
Node 6



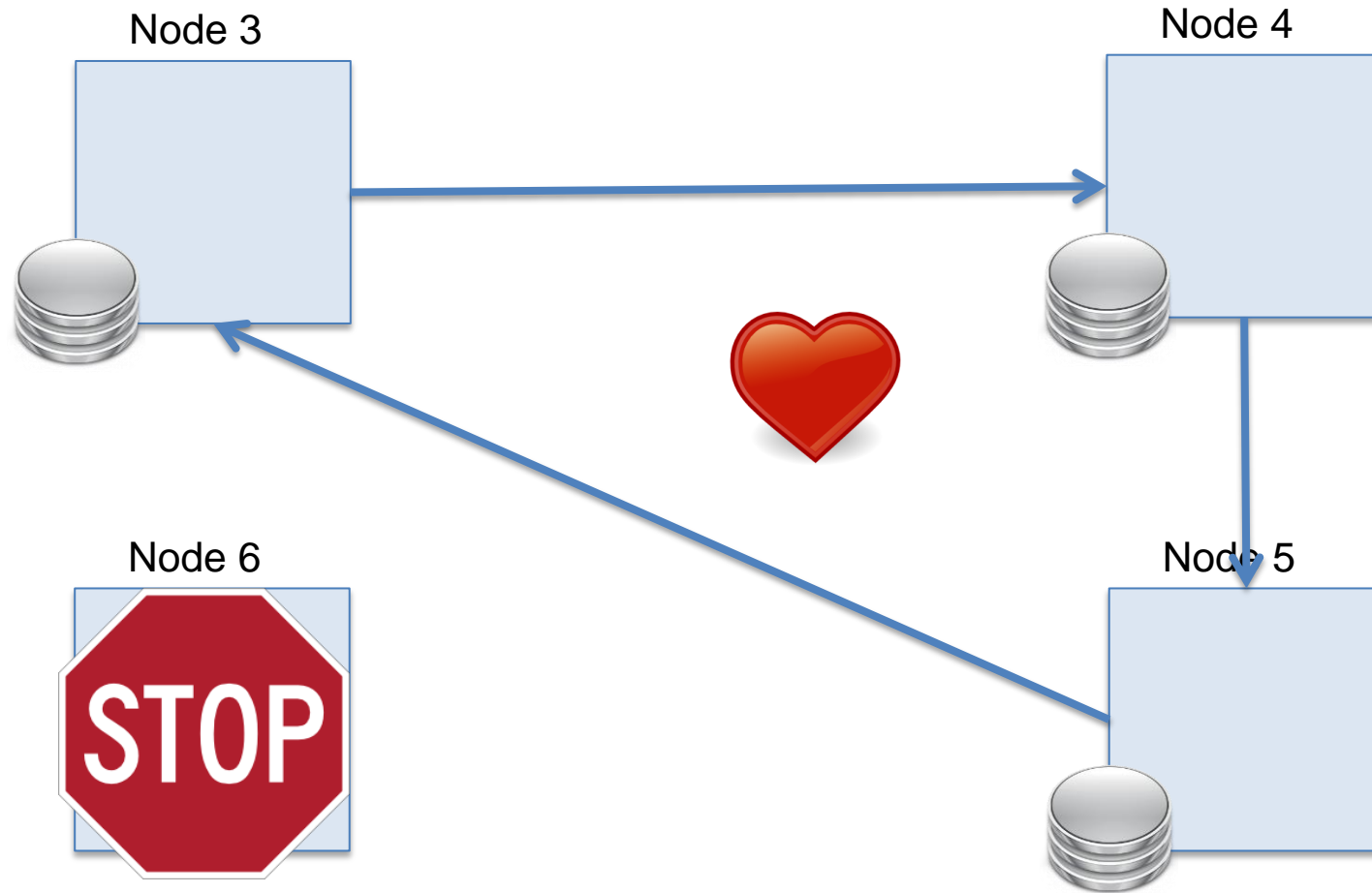
# Heartbeat Circle



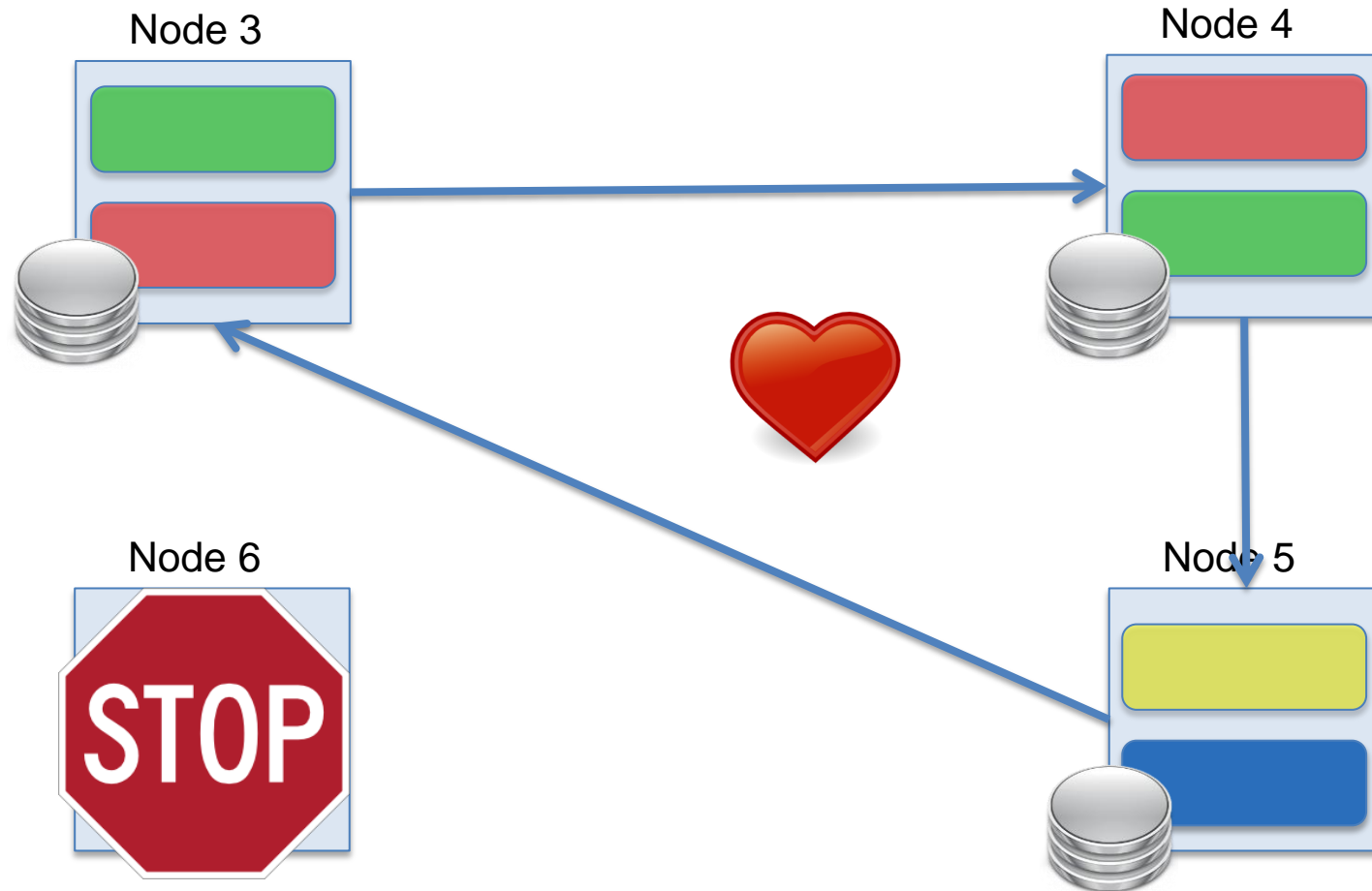
# Heartbeat Circle



# Heartbeat Circle



# Heartbeat Circle

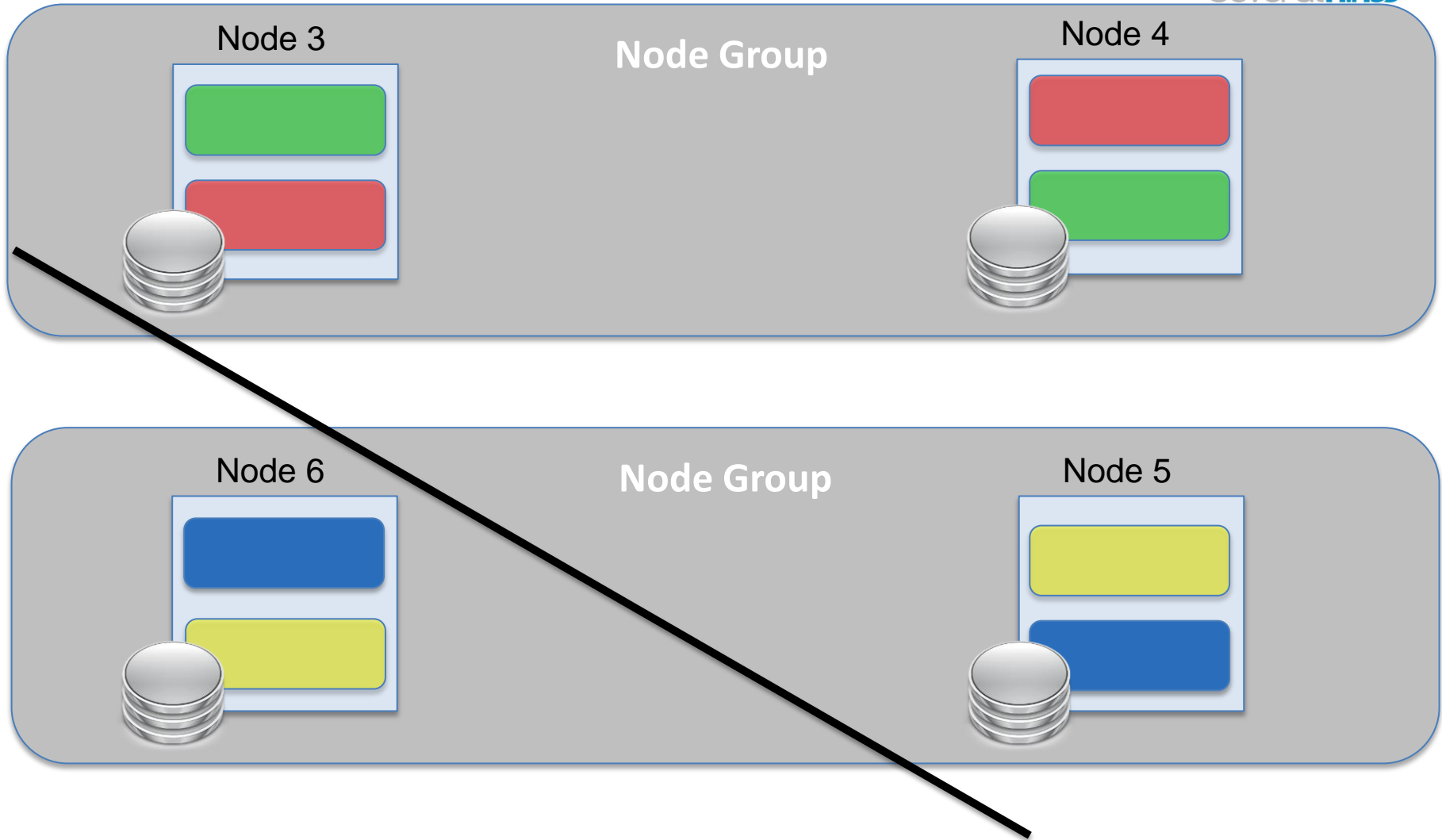


# Network Partitioning Protocol

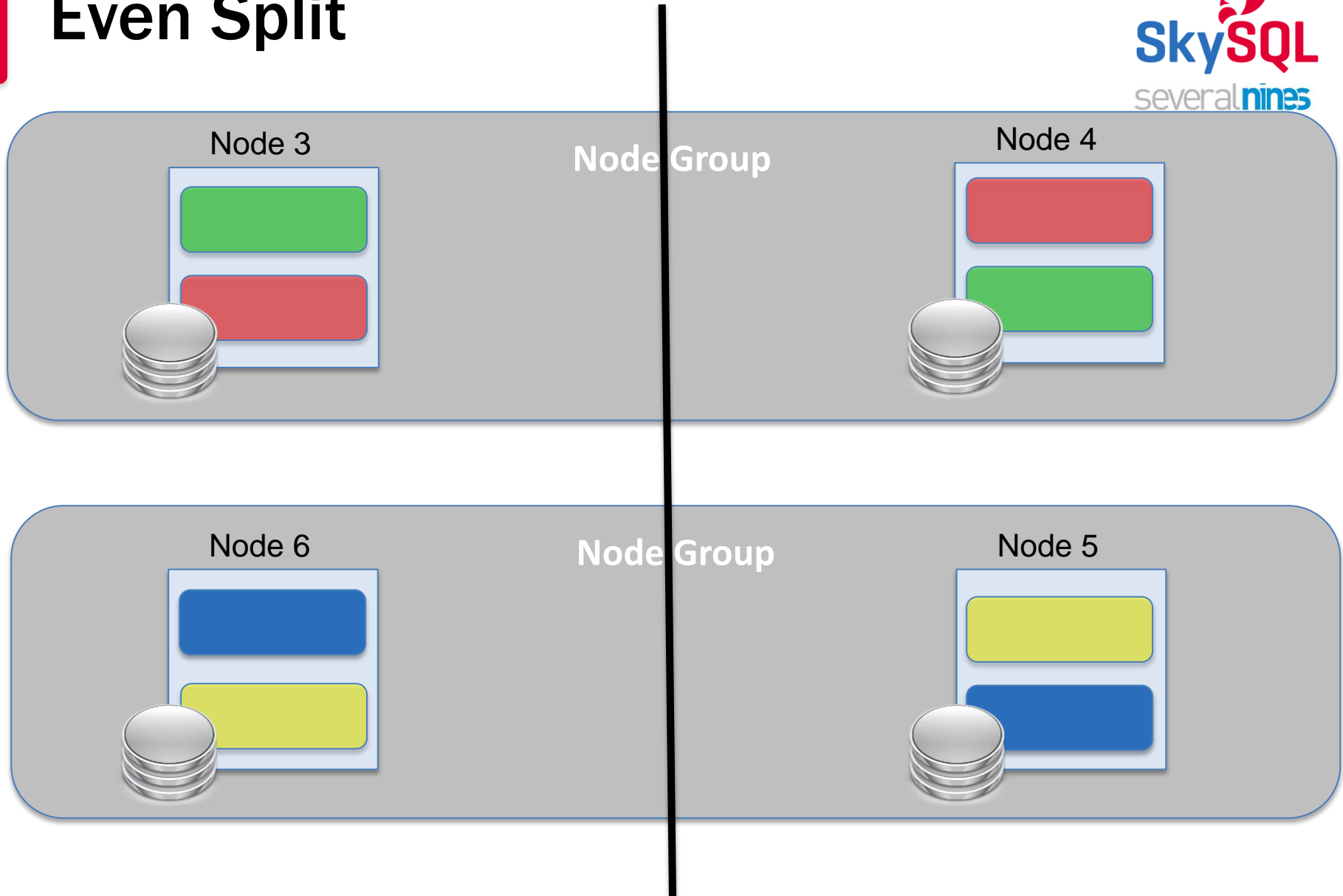


- The network partitioning protocol is designed to avoid a split brain scenario:
  1. Is there at least one node from each node group?
  2. Are all nodes present from any node group?
  3. Ask the arbitrator

# Uneven Split



# Even Split

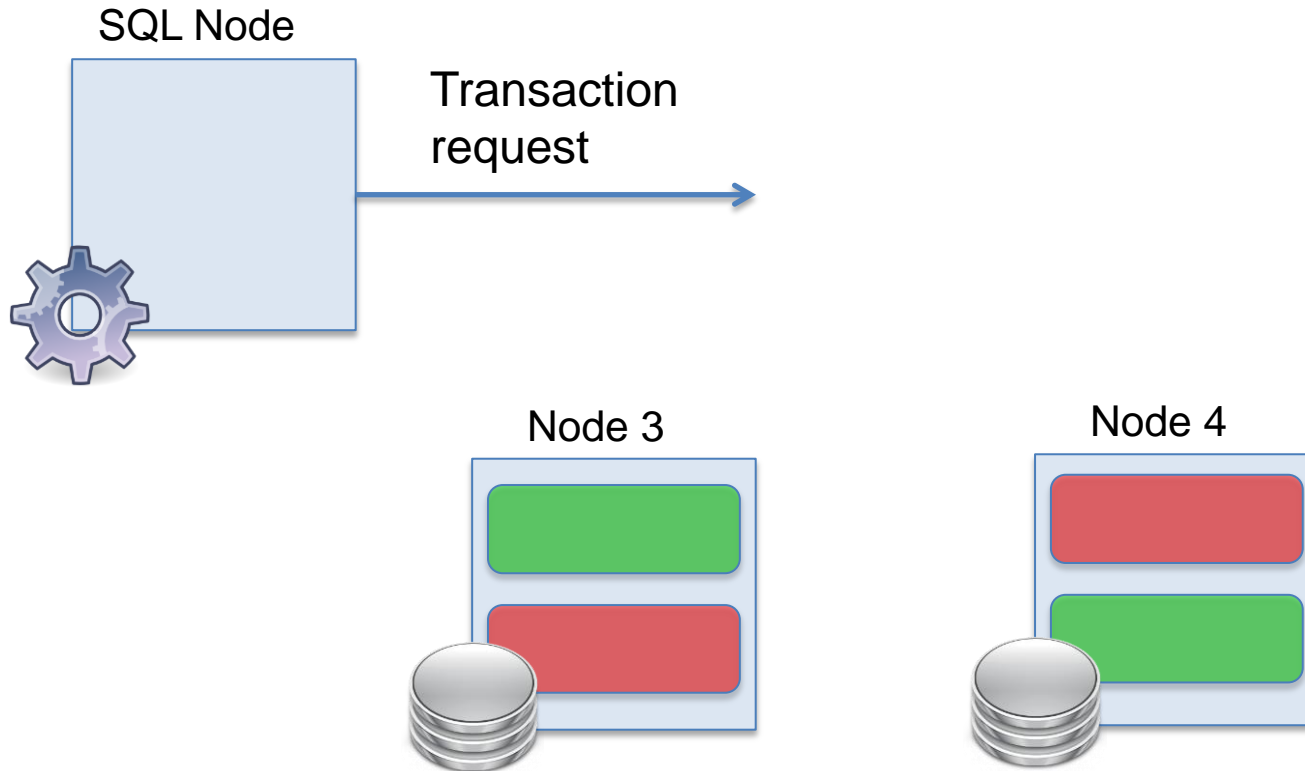


# Durability

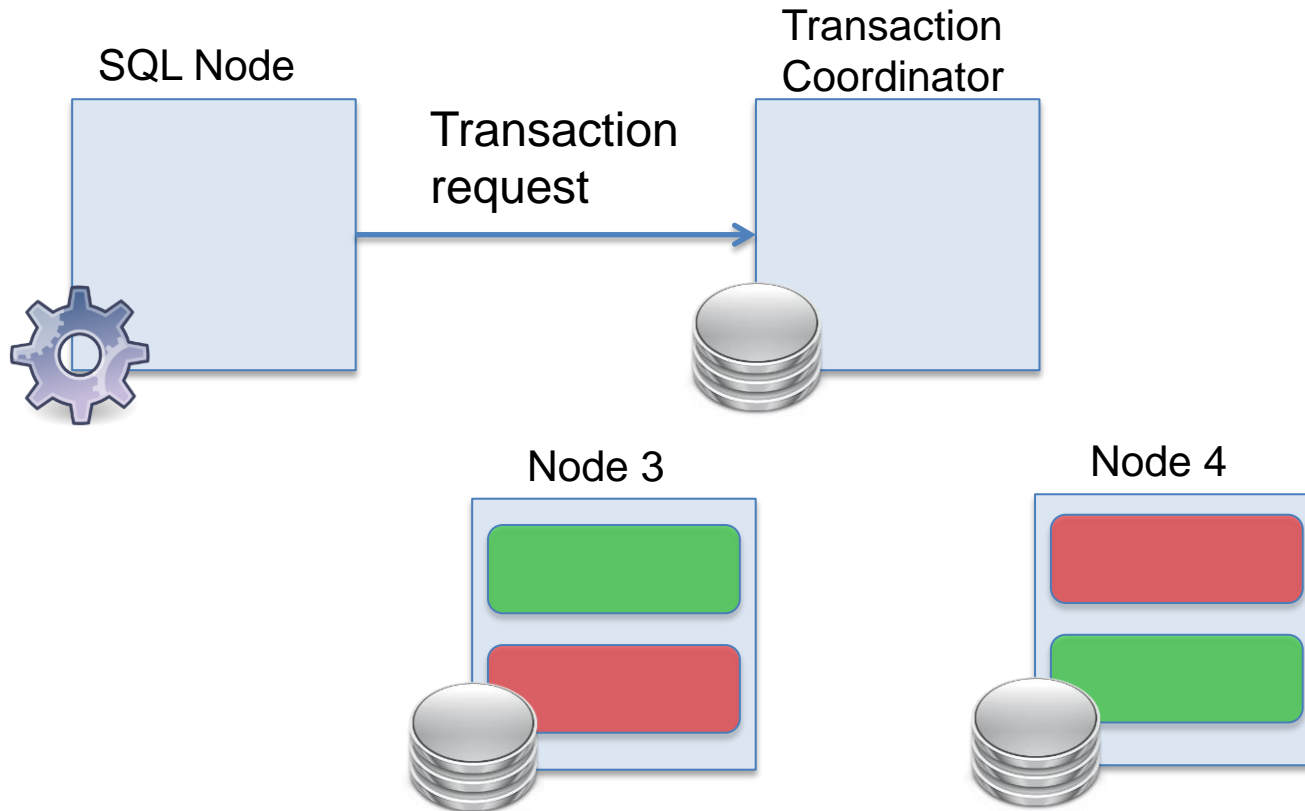


- In order for a node to recover faster some data is stored locally
  - The REDO log
    - Synchronized by global checkpoints (GCP)
  - The DataMemory
    - Synchronized by local checkpoints (LCP)
- These can also be used for system recovery

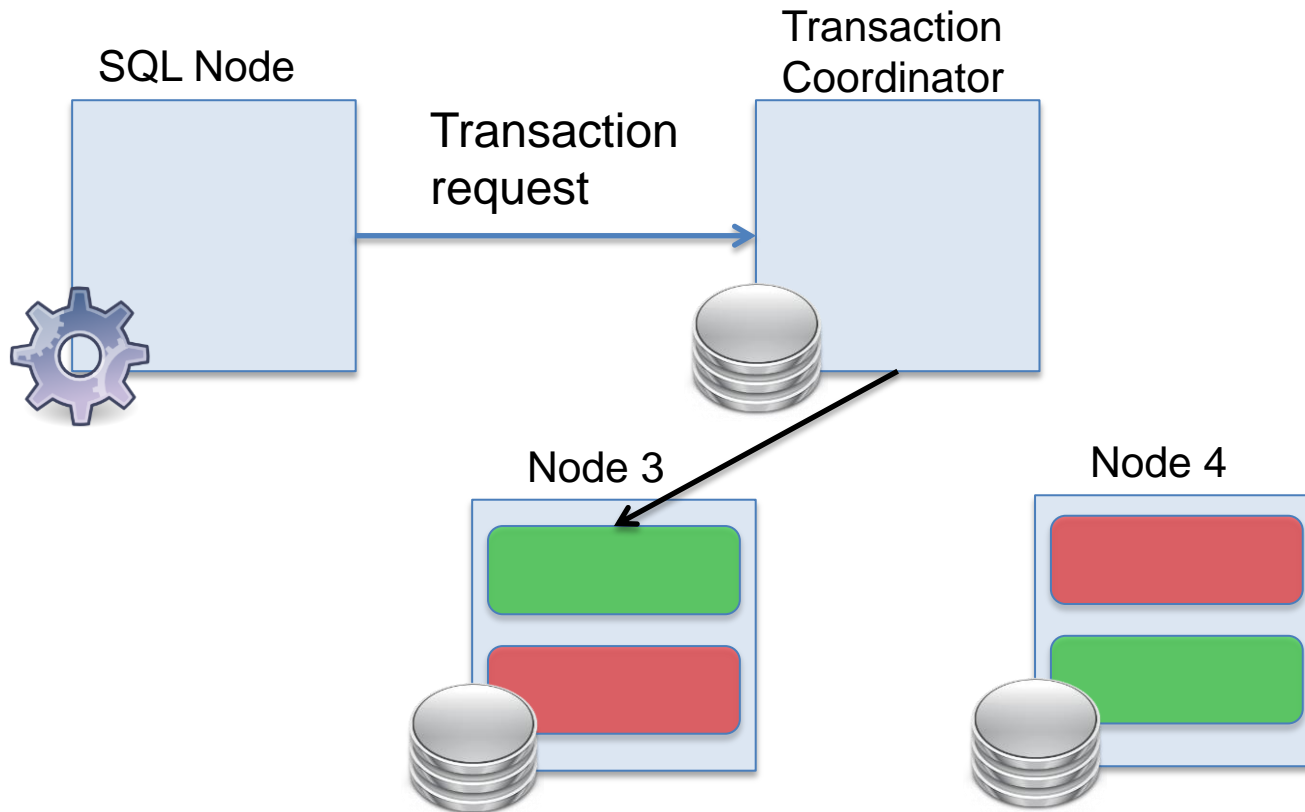
# Transactions



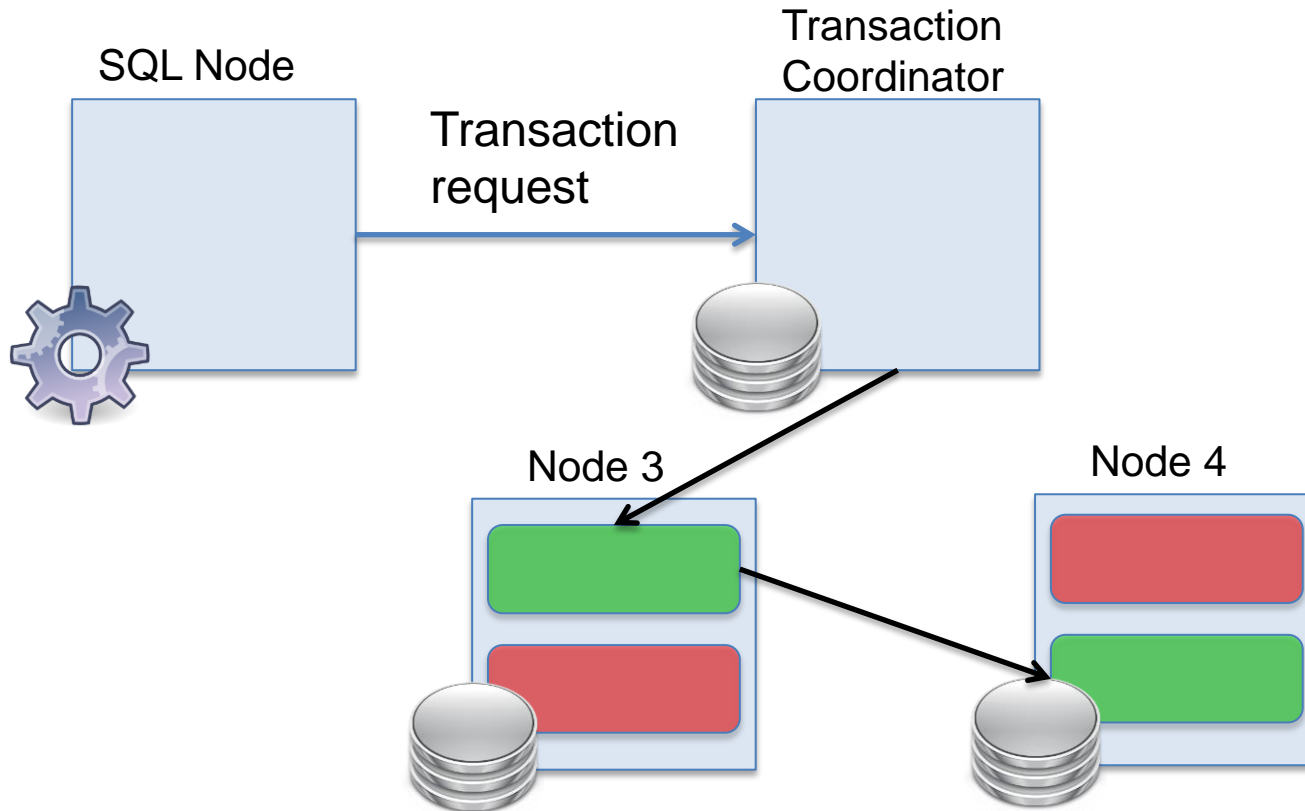
# Transactions – Two Phase Commit



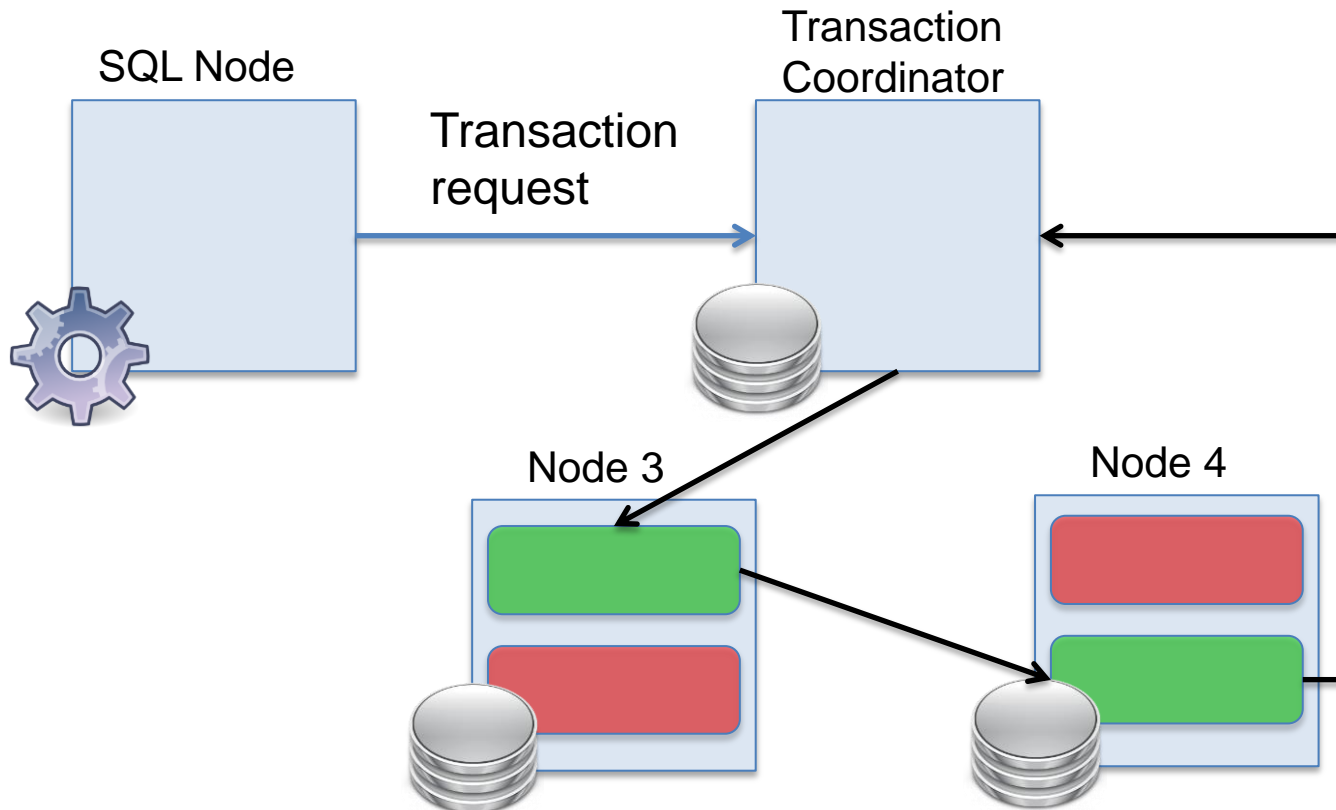
# Transactions – Prepare Phase



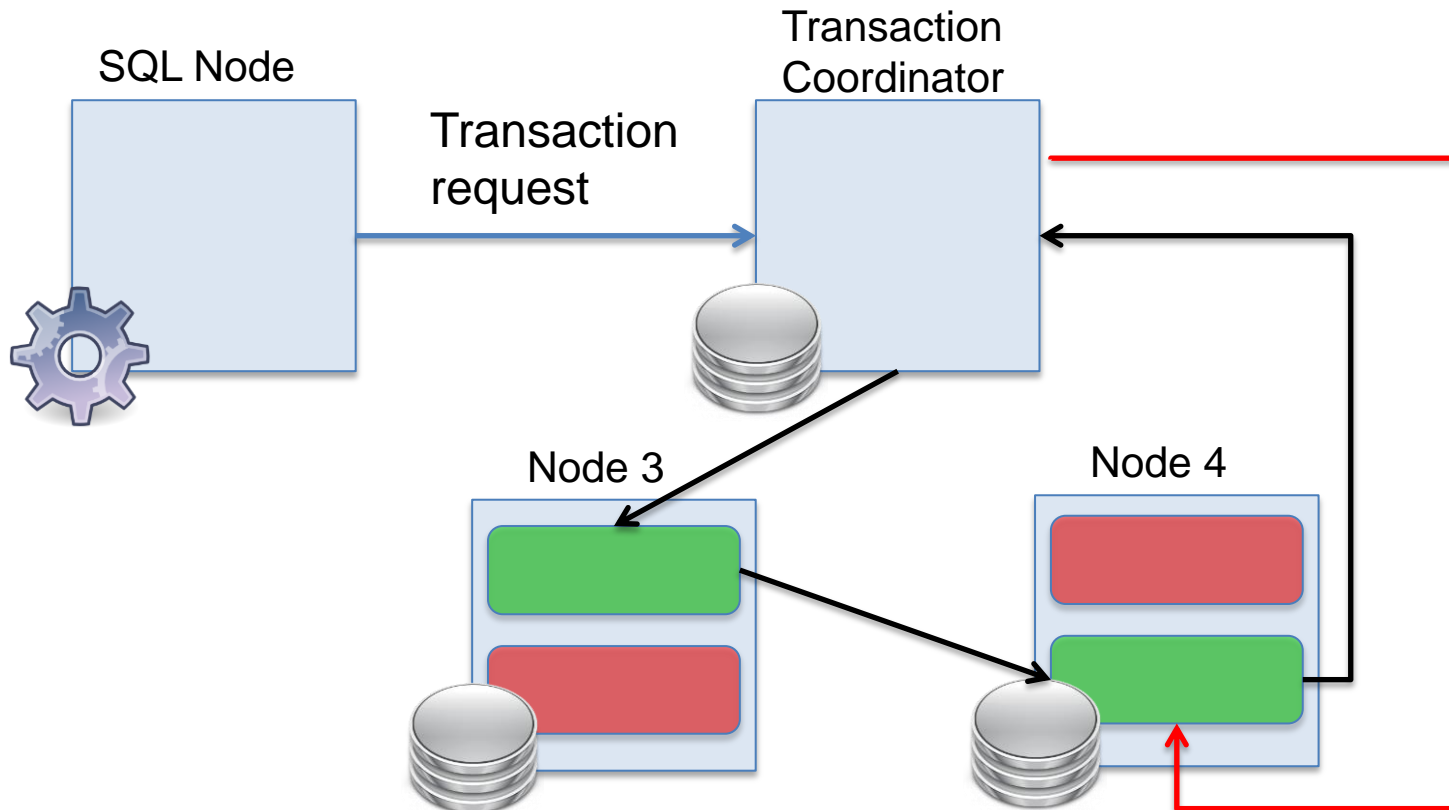
# Transactions – Prepare Phase



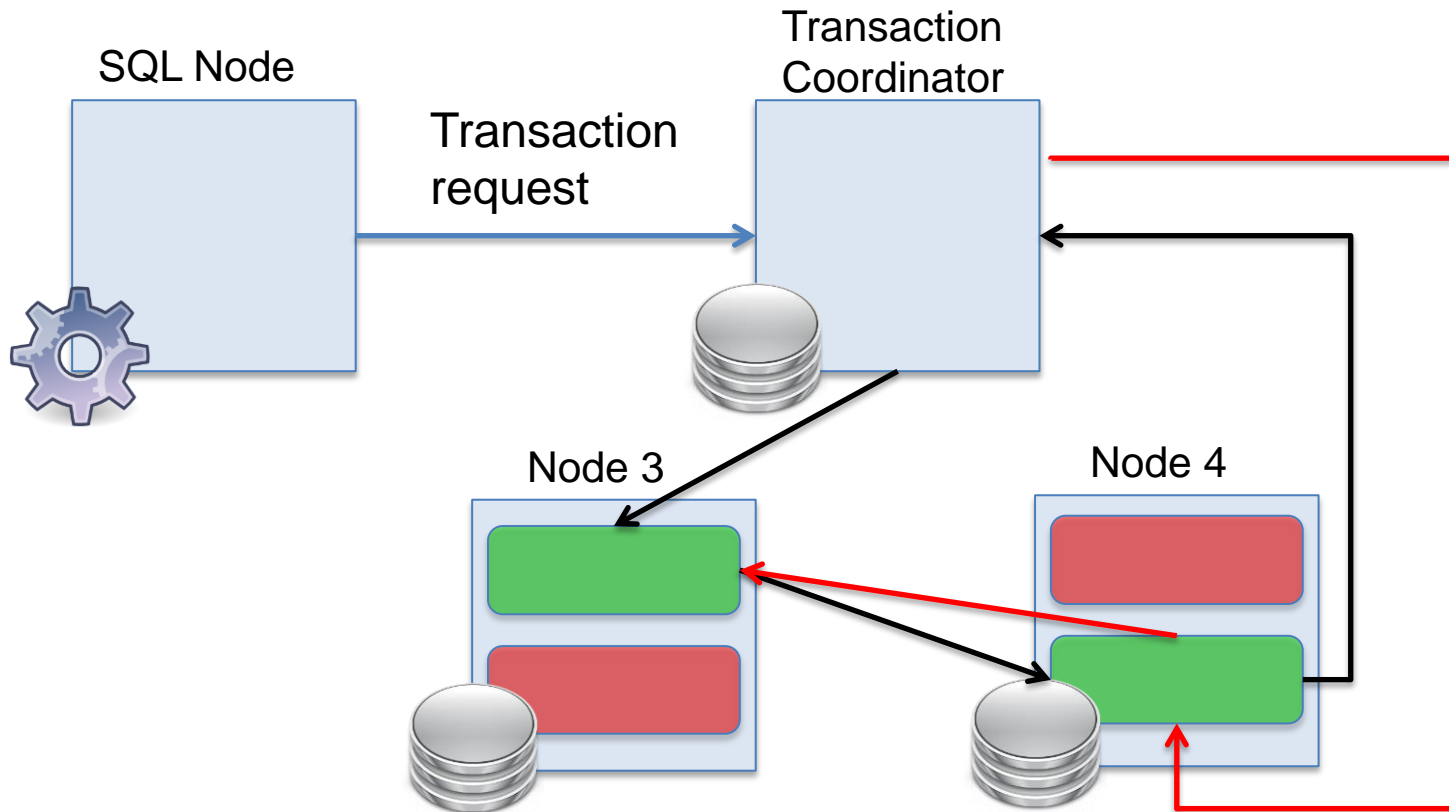
# Transactions – Prepare Phase



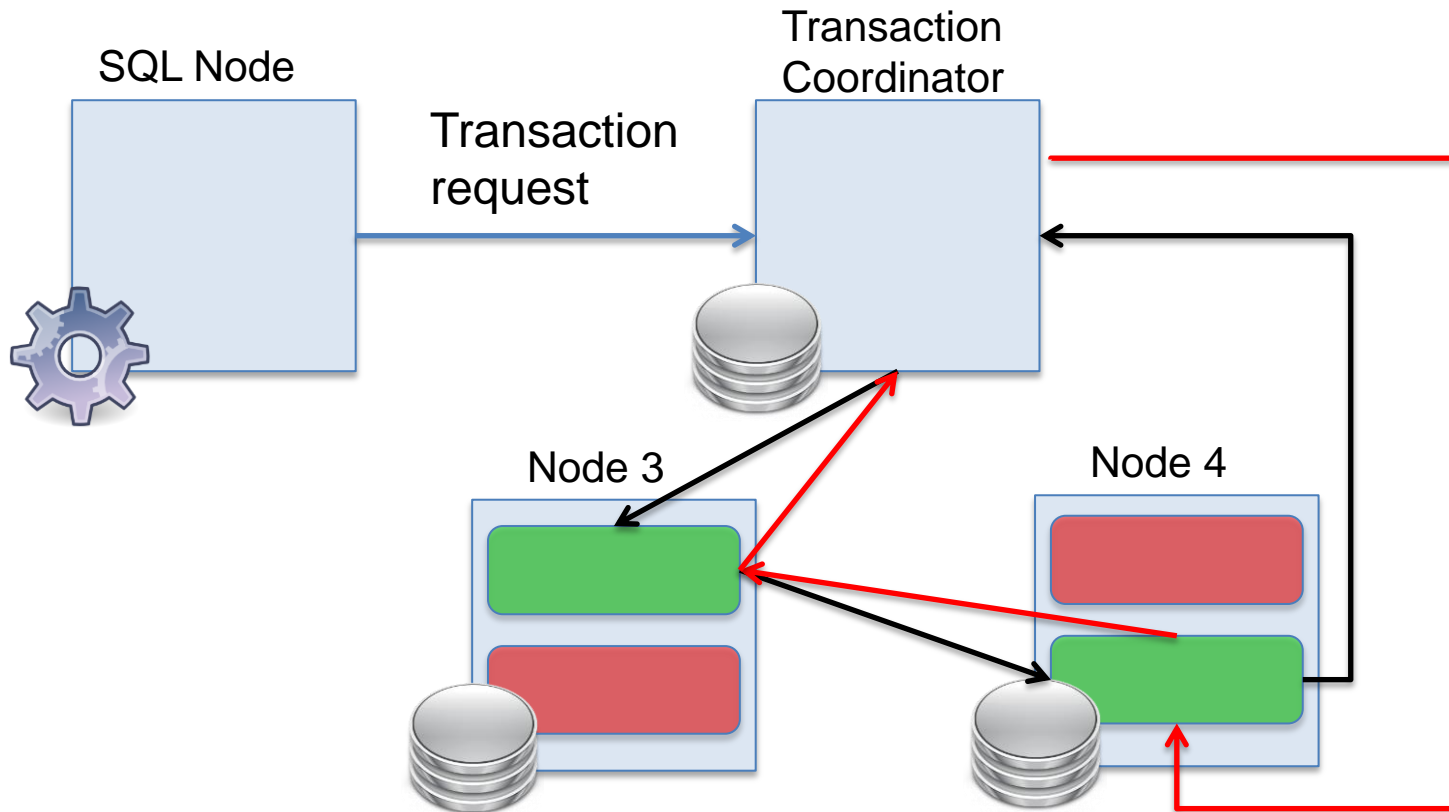
# Transactions – Commit Phase



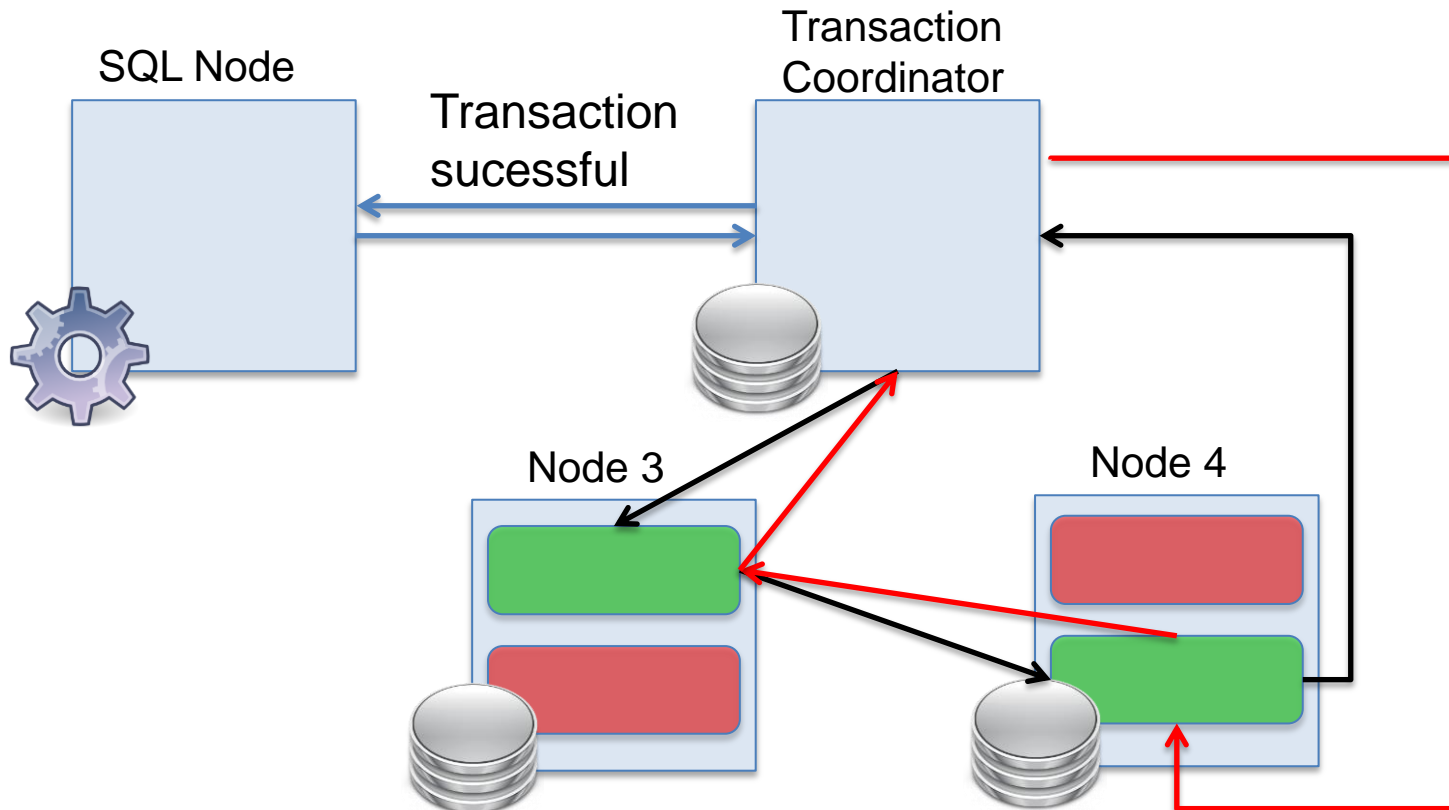
# Transactions – Commit Phase



# Transactions – Commit Phase



# Transactions – Commit Phase



# Indexes

- Unique Hash Indexes
  - Each table has a Primary Key hash index
  - Other unique hash indexes implemented by hidden tables
    - Partitioned like tables
- Ordered indexes
  - T-trees
  - Local for each node

# Part 2

## Practical Labs



# Preparations



1. Load the virtualbox and start the system
2. Examine the Cluster configuration file
3. Start the cluster
  - Start management node
  - Tail the cluster log
  - Start data nodes
  - Start the MySQL servers
4. Load the sakila database
5. Start the MySQL clients

# Exercise 1 – Initial test



1. Create a test table
2. Insert a row in the test table
3. Login to the other MySQL server and verify that the table is there too
4. Drop the table

# Exercise 2 – Backup and Restore



1. Take a backup
  - Use the `START BACKUP` command
2. Start a clean cluster with no data
3. Use the backup to restore the data
  - Use the `ndb_restore` utility

# Exercise 3 – Node Recovery



1. Examine the log during the process
2. Kill one of the data nodes with the `kill` command
3. Execute a query, is the cluster working?
4. Restart the node
5. Execute a query

# Exercise 4 – NDBINFO



1. Go to the ndbinfo schema
2. Examine the tables

# Exercise 5 – Resource Limits



1. Issue the statements that run into limits
2. Change the configuration file
  - Set `MaxNoOfConcurrentOperations` to 40000
3. Do a "rolling restart"
  - Restart each node one by one
4. Re-issue the failed statement

# Exercise 6 – Partial Restore



1. Restore one table from the backup
  - Restore the table directly or
  - Extract the contents to plainfile and import

# Exercise 7 – Query Optimization



1. Run the Query
  - Watch query time
2. Use EXPLAIN, show indexes
  - Watch the query execution plan and cardinality
3. Rewrite the query
  - Watch query time

# Part 3

## Best Practices

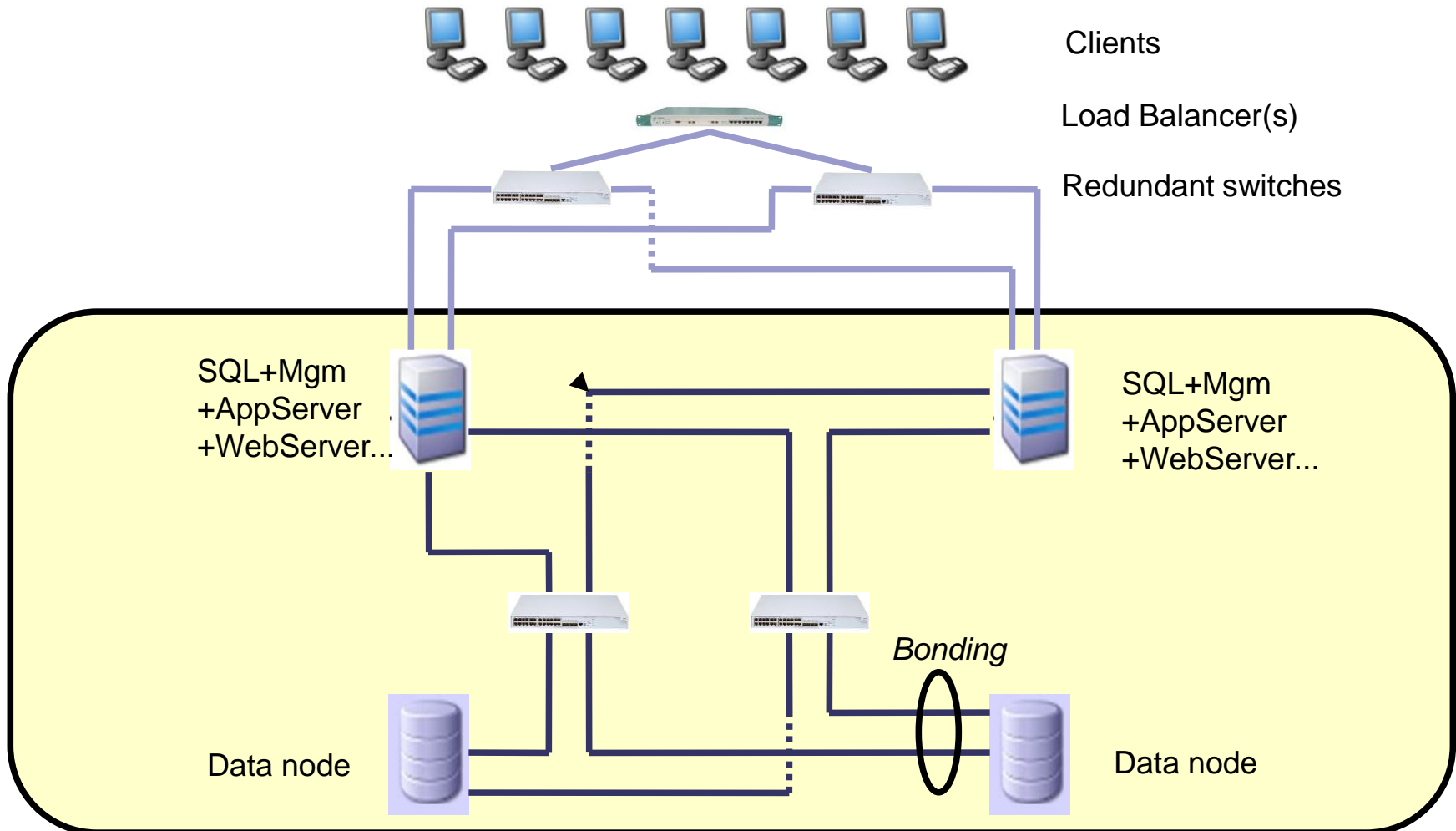


# Agenda



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

# Recommended Setup



# Networking



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

# Hardware – Data Nodes



- One data node can use 8 cores (Cluster 7.0+)
- **CPU:** 2 x 4 core (Nehalem works really well)
  - Fast CPU → fast processing of messages
- **RAM:** As much as you need
  - 10GB data set will require 20GB of RAM
  - Each node will then need  $2 \times 10 / \# \text{data nodes}$   
(For example 2 data nodes → 10GB → 16GB good)
- **Disk:**  $10 \times \text{DataMemory}$  + space for BACKUP + TableSpace (if disk data tables)

# Hardware – MySQL Servers



- **CPU:** 2 – 16 cores
- **RAM:** Not so important – 4GB enough (depends on connections and buffers)
- **Disks:** Used mainly for logging
  - Binary log needed for replication

# Disk Subsystem

## low-end



1 x SATA 7200RPM

- For a read-most, write not so much
- No redundancy (but other data node is the mirror)

## mid-end



1 x SAS 10KRPM

- Heavy duty (many MB/s)
- No redundancy (but other data node is the mirror)

## high-end



4 x SAS 10KRPM

- Heavy duty (many MB/s)
- Disk redundancy (RAID1+0) hot swap

- REDO, LCP, BACKUP – written sequentially 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

# 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 – easily 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 ]` (Cluster 7.0+)

# Configuration - General



- `MaxNoOfExecutionThreads<=#cores`
  - Contention can occur → unexpected behaviour
- `RedoBuffer=32-64M`
  - If you need to set it higher your disks are too slow
- `FragmentLogFileSize=256M`
- `NoOfFragmentLogFiles=`  
`6 x DataMemory (in MB) / (4x 256MB)`
  - Most common issue – redo log too small
- Try the configurator:  
[www.severalnines.com/config](http://www.severalnines.com/config)

# Application - Primary Keys



- Always define a primary key
  - Tables without primary keys are accepted
    - A hidden primary key is created
    - The hidden PK is not replicated
    - There are recovery issues with hidden PKs
    - Application behavior (KEY NOT FOUND.. etc)
- At least have a  
`id BIGINT AUTO_INCREMENT PRIMARY KEY`
  - Even if you don't need it for your applications

# Application - Query Cache



- Don't cache everything in the Query Cache
  - Expensive to invalidate over N mysql servers
  - A write on one server will force the others to purge their cache
- For tables that change seldom (or read-only)
  - Set `query_cache_type=2 (DEMAND)`  
`SELECT SQL_CACHE <cols> .. FROM table;`
  - This can be good for STATIC data

# Application – Transaction Size



- Transactions (large updates)
  - NDB designed for many and short transactions
    - Recommended to UPDATE / DELETE in small chunks
    - Use LIMIT 10000 until all records are UPDATED/DELETED
- MaxNoOfConcurrentOperations
  - Limit for how many records than can be modified simultaneously on one data node
  - MaxNoOfConcurrentOperations=1000000 will use 1GB of RAM
    - Use only if necessary

# Application – Table Locks



- Table lock commands are local only
  - FLUSH TABLE WITH READ LOCK;
  - LOCK TABLES <table> READ;
- 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

# REDO Log Optimizations



- Some tables account for a lot of writes, but do not need to be recovered (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;
```

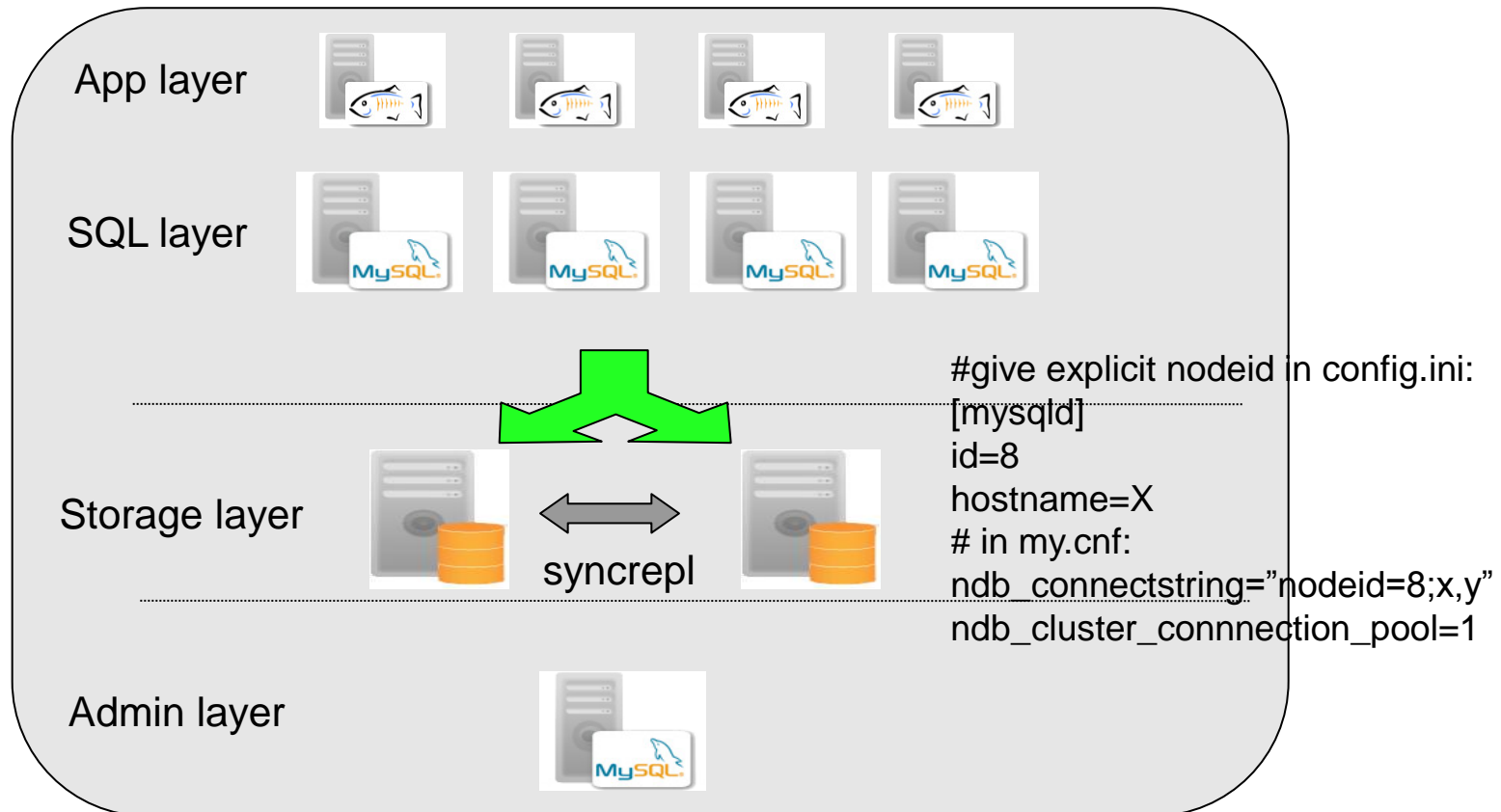
  - session\_table will not be REDO logged
    - No disk activity for this table!

# ALTER TABLE

- NOT online operations:
  - Rename a table
  - Change data type
  - Change storage size
  - Drop column
  - Rename column
  - Add/Drop a PRIMARY KEY
- Online operations:
  - Add column (ALTER ONLINE ...)
  - CREATE INDEX
  - Online add node (see my presentation from last year how to do it)
- Altering a 1GB table offline requires 1GB extra

# Administration Layer

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



# Online Upgrades



- OS, SW version (7.0.x → 7.1.x)
- Configuration
  - increase DM, IM, Buffers, redo log, [mysqld] slots
- Hardware (upgrade more RAM etc)
- Adding data nodes (from 7.0)
  - See Johan's presentation from the conference 2009

# Backup



- Backup of NDB tables
  - Online – can have ongoing transactions
  - Consistent – only committed data
  - `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

# Backup



- 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

# 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](http://www.severalnines.com/cmon)- monitors data nodes and MySQL servers