MySQL Cluster: An introduction

Geert Vanderkelen

2006-06-24

MySQL AB

Agenda

- Quick MySQL Introduction
- Overview
- Cluster components
- High Availability and Scalability
- Small example: Web Sessions
- New features
- Q&A

Who am I?

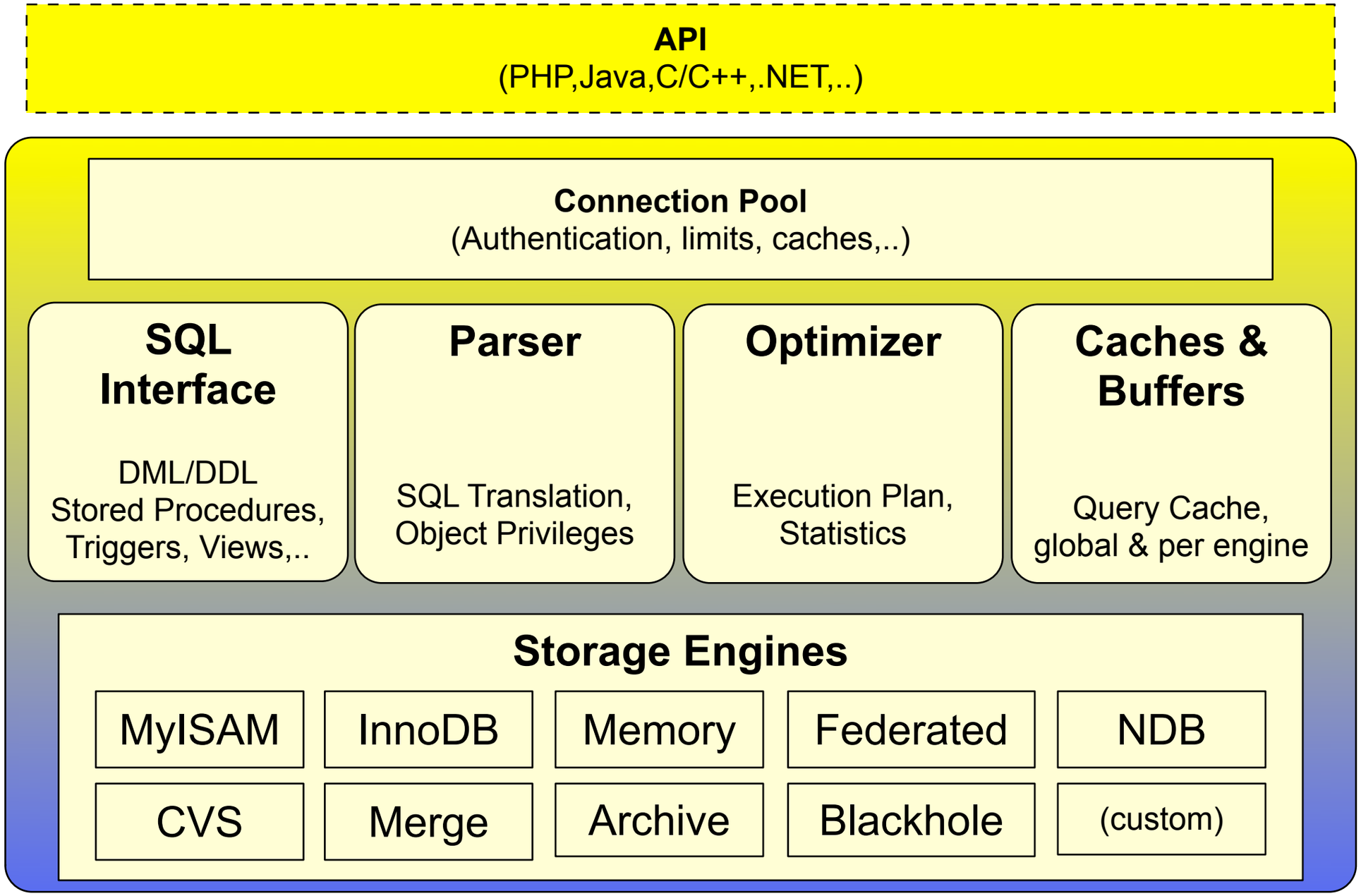
- Geert Vanderkelen
  - Support Engineer for MySQL AB
  - MySQL Cluster
  - Belgian, based in Germany

MySQL Quick Introduction

Quick intro to MySQL

- MySQL is a DBMS running on most OS
- Reputation for speed, quality, reliability and easy to use
- Storage Engines (MyISAM, InnoDB, ..)
- Current GA is 5.0:
  - SQL 2003 Stored procedures
  - Triggers, Updatable Views, Cursors
  - Precision math
  - Data dictionary (INFORMATION_SCHEMA database)
  - and more..
- Lots of Connectors and API available

MySQL Architecture



MySQL Cluster Overview

What is MySQL Cluster?

- In-memory storage
  - data and indices in-memory
  - check-pointed to disk
- Shared-Nothing architecture
- No single point of failure
  - Synchronous replication between nodes
  - Fail-over in case of node failure
- Supports transactions
  - READ COMMITTED only
- Row level locking
- Hot backups
- Online software upgrade

Background

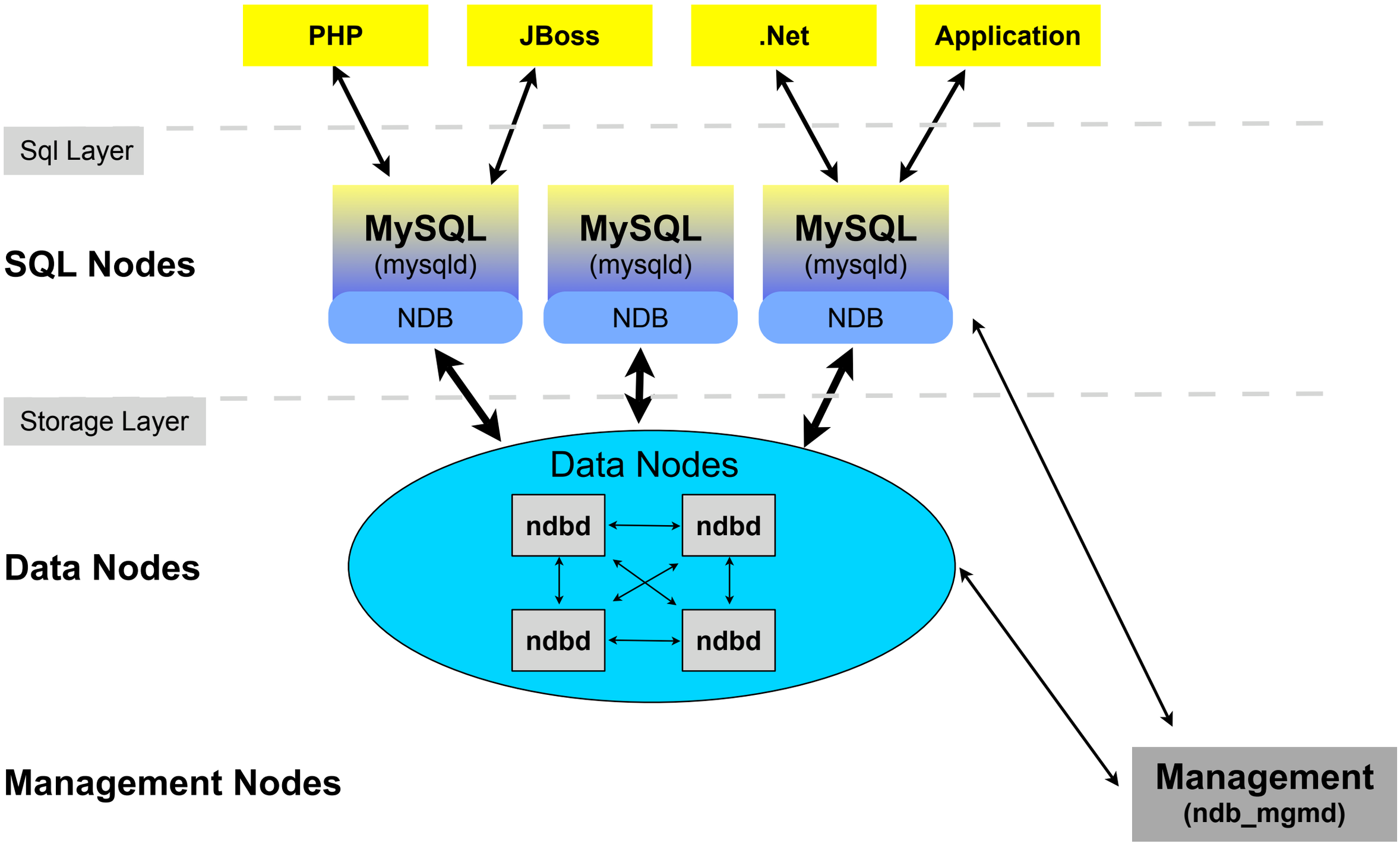
- Designed for telecom environment
- 99.999% Availability
- Performance
  - reponse time about 5-10ms
  - 10000 transactions/sec
- Scalability
  - many concurrent applications
  - load balancing

Cluster components

Cluster Nodes

- Participating processes are called ‘nodes’
  - Nodes can be on same computers
- Two tiers in Cluster:
  - SQL layer
    - SQL nodes (also called API nodes)
  - Storage layer
    - Data nodes
    - Management nodes
- Nodes communicate through transporters:
  - TCP (most common)
  - Shared memory
  - SCI (Scalable Coherent Interface)

Components of a Cluster



Data nodes

- Contain data and index
- Used for transaction coordination
- Each data node is connect to the others
- Shared-nothing architecture
- Up to 48 data nodes

SQL nodes

- Usually MySQL servers
- Also called API nodes
- Each SQL node is connected to all data nodes
- Applications access data using SQL
- Native NDB application (e.g. ndb_restore)
- Client application written using NDB API

Management nodes

- Controls setup and configuration
- Needed on startup for other nodes
- Cluster can run without
- Can act as arbitrator during network partitioning
- Cluster logs
- Accessible using ndb_mgm CLI

ndb_mgm CLI

```
[geert@ndbsup-1 5.1bk]$ ./bin/ndb_mgm
-- NDB Cluster -- Management Client --
ndb_mgm> show
Connected to Management Server at: ndbsup-priv-1:1406
Cluster Configuration
---------------------
[ndbd(NDB)]     2 node(s)
id=4    @10.100.9.8  (Version: 5.1.12, Nodegroup: 0, Master)
id=5    @10.100.9.9  (Version: 5.1.12, Nodegroup: 0)

[ndb_mgmd(MGM)] 2 node(s)
id=1    @10.100.9.6  (Version: 5.1.12)
id=20 (not connected, accepting connect from ndbsup-priv-2)

[mysqld(API)]   4 node(s)
id=10   @10.100.9.6  (Version: 5.1.12)
id=11 (not connected, accepting connect from ndbsup-priv-2)
id=12 (not connected, accepting connect from any host)
id=13 (not connected, accepting connect from any host)

ndb_mgm>
```

Physical Requirements

- Need at least 3 machines
- Data nodes
  - need lots of memory
  - ndb_size.pl (http://forge.mysql.com)
  - not CPU bound, single threaded
  - MySQL 5.1 brings disk based
- SQL/API nodes
  - usually more than data nodes
  - more CPU bound, multi threaded
- Dedicated network
  - TCP/IP communication used
  - separate data node traffic
  - protected from outside!

High Availability & Scalability

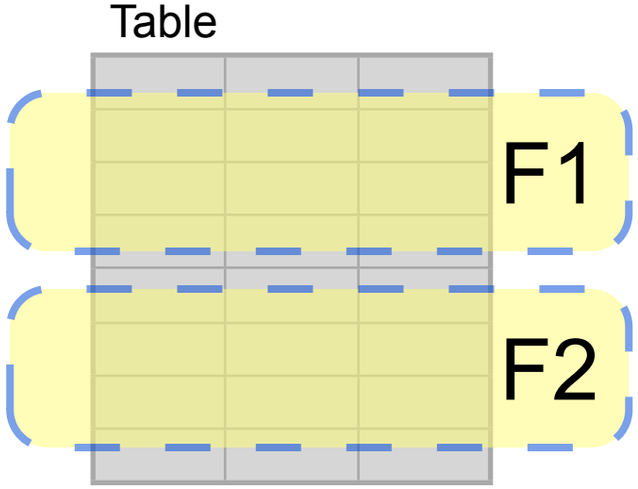
High Availability and Scalability

- Data Distribution
  - Fragmentation
  - Synchronous Replication
- Failure handling:
  - for MySQL nodes
  - for Data nodes
  - for Management nodes
- Backup
  - hot backups
  - restoring
- Cluster replication

Data Distribution

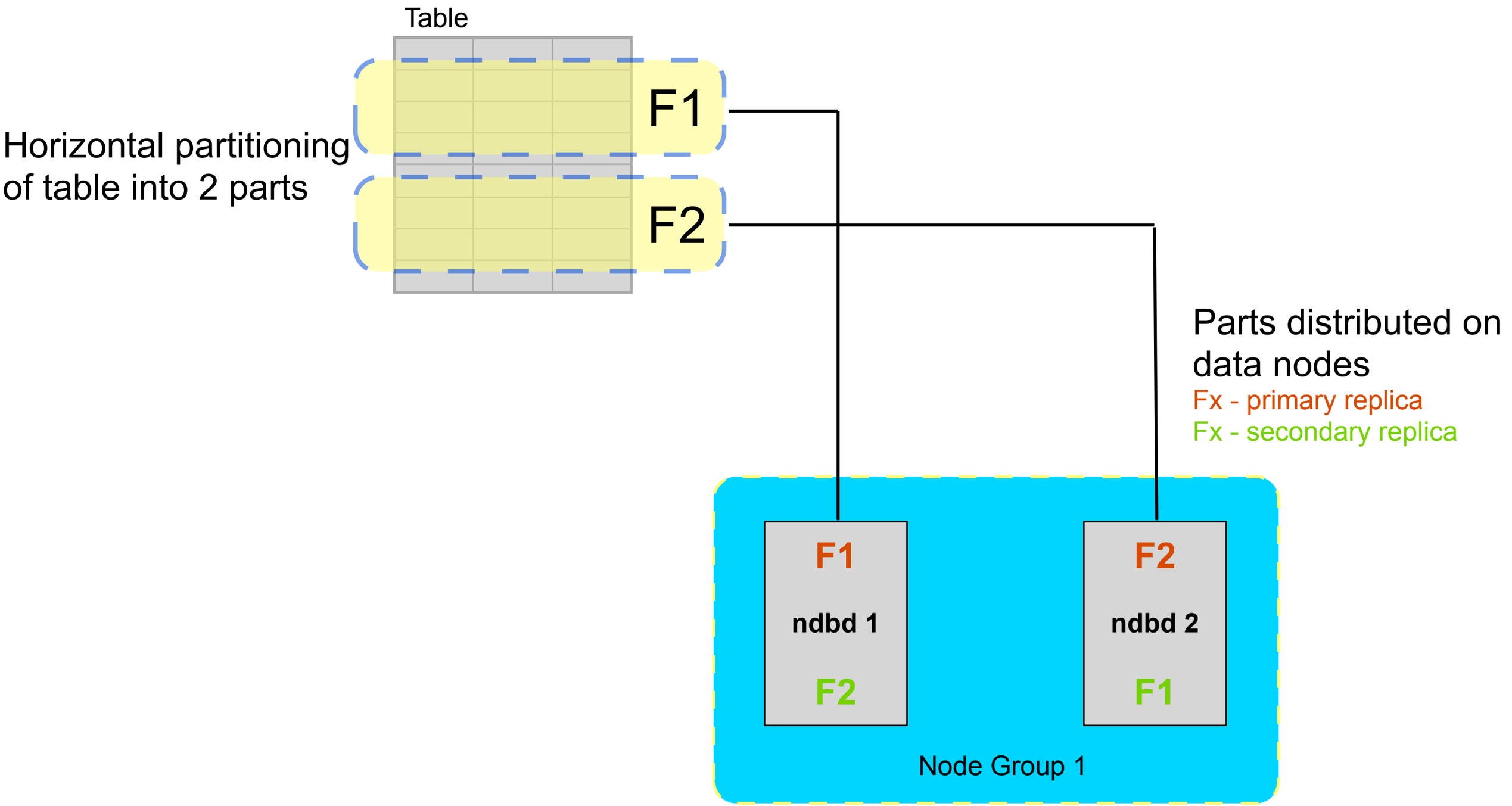
Fragmentation & Replication

- Fragmentation (or partitioning)
  - tables fragmented horizontally
  - if we have 4 data nodes, data fragmented in 4 parts
  - primary node for one fragment

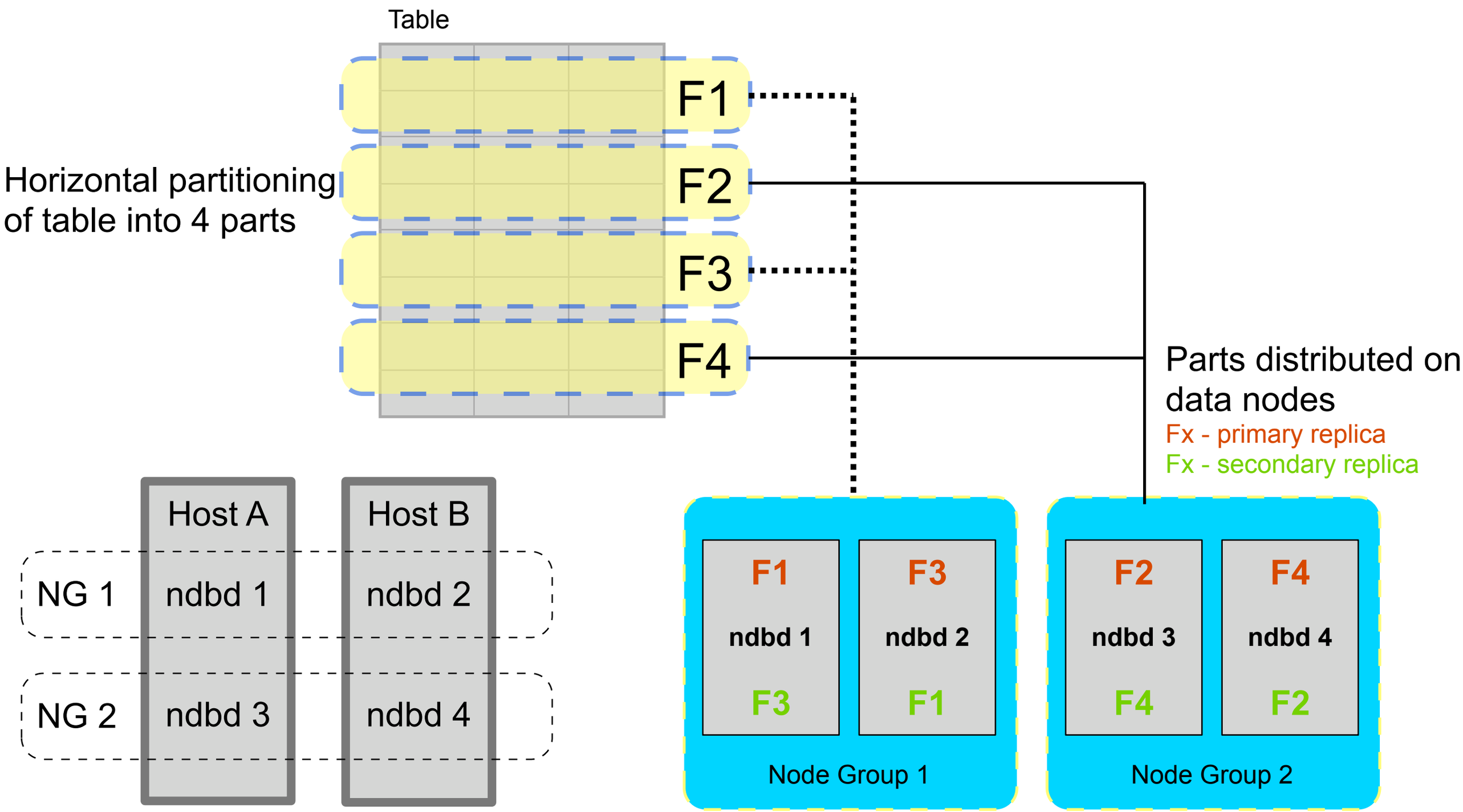


- Replication
  - each node has secondary fragment (replica)
  - synchronous: committed everywhere or not at all
  - nodes having same data are grouped

Data Fragmentation: 2 nodes, 2 replicas



Data Fragmentation: 4 nodes, 2 replicas



Failure Handling

A Configuration

[ndbd default]
NoOfReplicas = 2
DataMemory = 400M
IndexMemory = 32M
DataDir = /var/lib/mysql/cluster

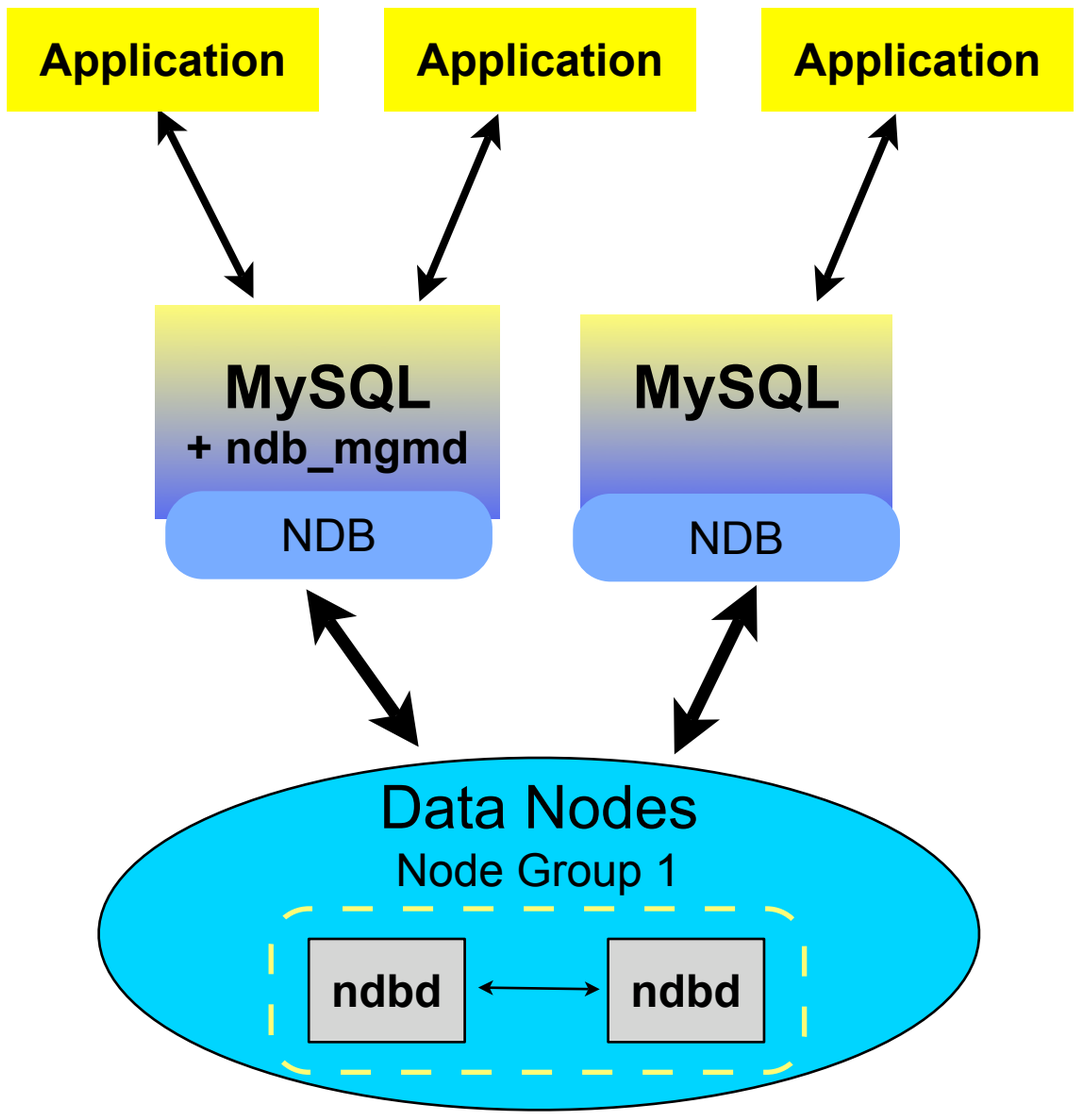
[ndb_mgmd]
DataDir = /var/lib/mysql/cluster
HostName = 192.168.0.42

[ndbd]
HostName = 192.168.0.40

[ndbd]
HostName = 192.168.0.41

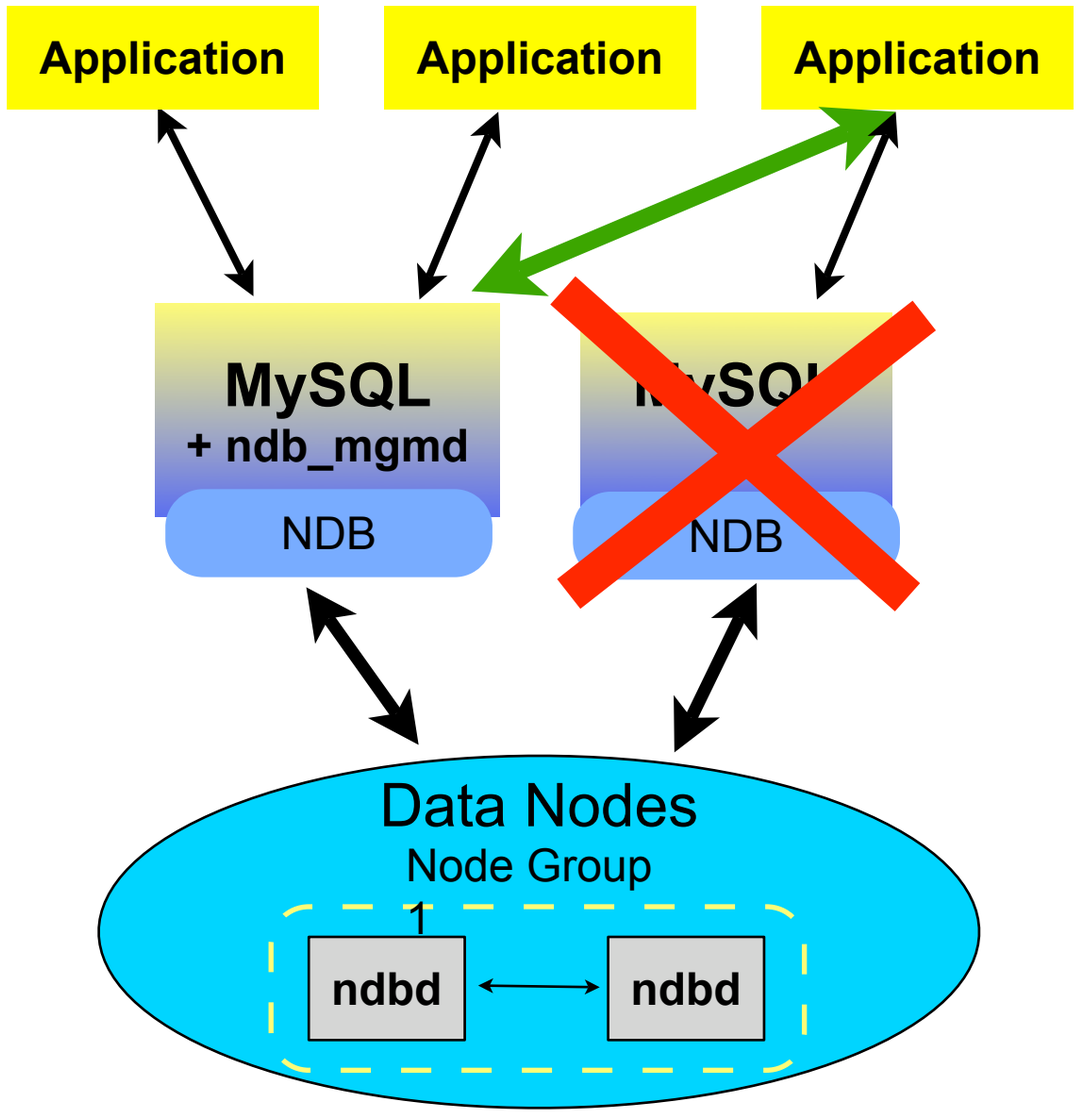
[mysqld]
HostName = 192.168.0.42

[mysqld]
[api]



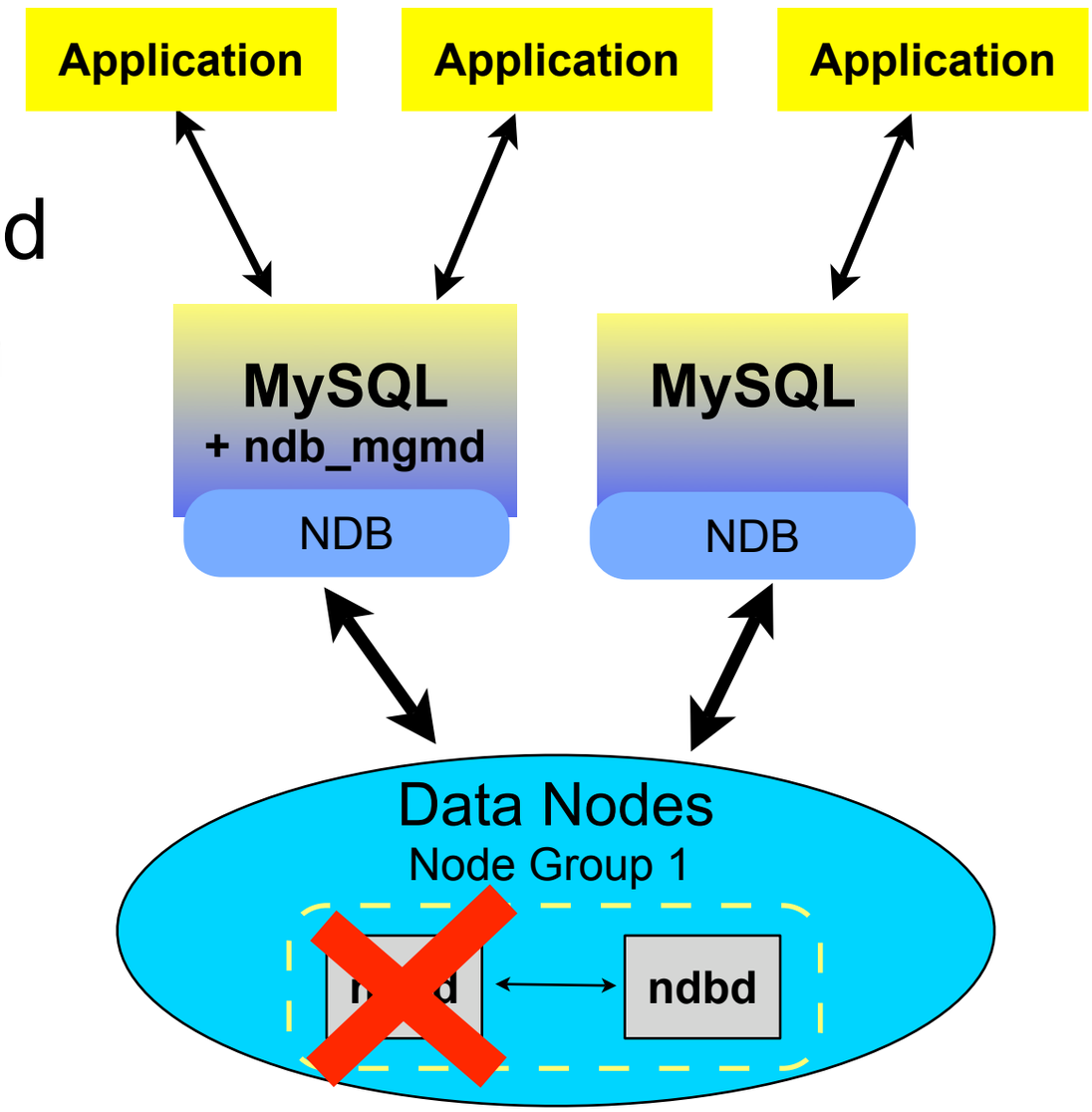
Failure: MySQL server

- Applications can use other
- mysqld reconnects



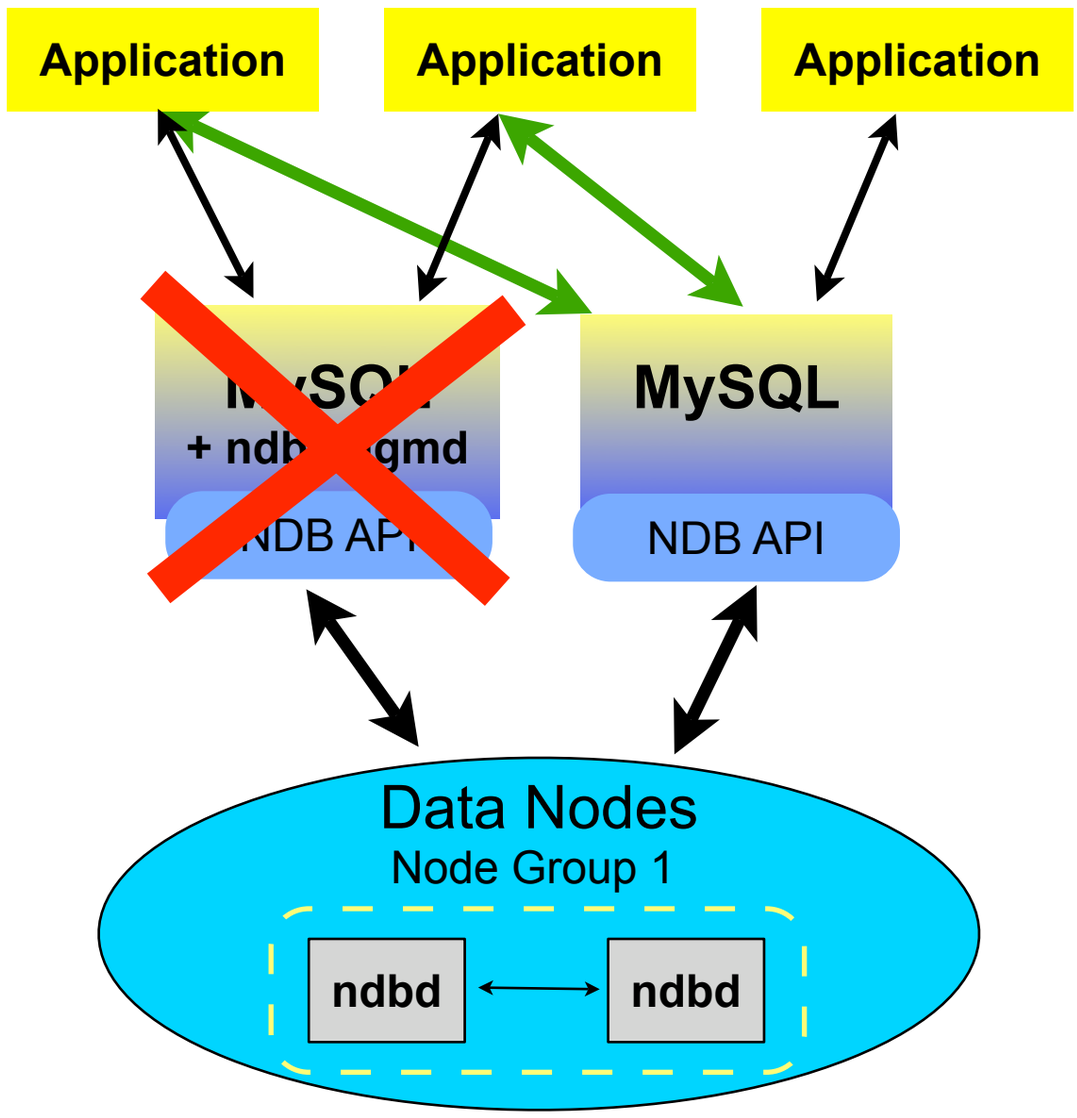
Failure: Data Node

- Other data nodes know
- Transaction aborted
- Min. 1 node per group needed
- 0 nodes in group = shutdown



Failure: Management Node

- Continued operations
- Needed for startup
- Best to have 2
- Can run on MySQL Nodes



Backups

Backups

- Hot Backups
  - no interruption
  - backups made at same point
  - locally on each data node
  - using ndb_mgm CLI
  - backups can be centralized
- Restoring
  - using ndb_restore application
  - can be done from any API node
  - first 1 time meta information, then data
  - needs single user mode
  - should never be needed

Scalability

Scalability

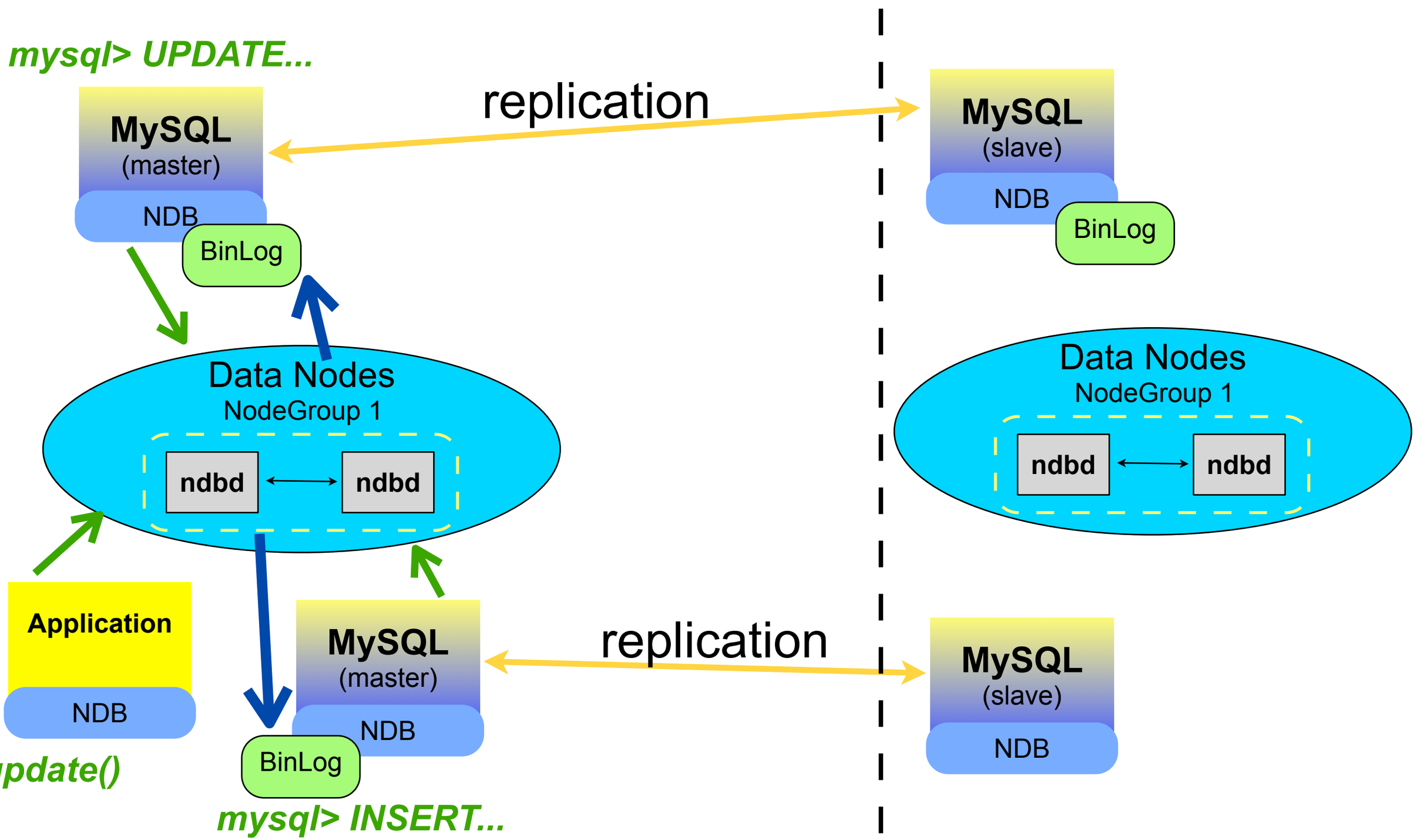
- Total of 64 nodes
- Max. 48 data nodes
- Typical setup:
  - 2 ndb_mgmd (management)
  - 4 data nodes
  - Up to 58 MySQL server (SQL nodes)
- Adding nodes need restart
  - adding data nodes can not be done online

Cluster Replication

Cluster Replication

- MySQL 4.1/5.0
  - statement based replication
  - all updates to data should go to 1 MySQL server
  - updates from NDBAPI application ignored
- MySQL 5.1
  - row based replication
  - updates can go to all MySQL servers
  - injector thread
  - NDBAPI applications not ignored
  - multiple replication channels

Cluster Replication in 5.1



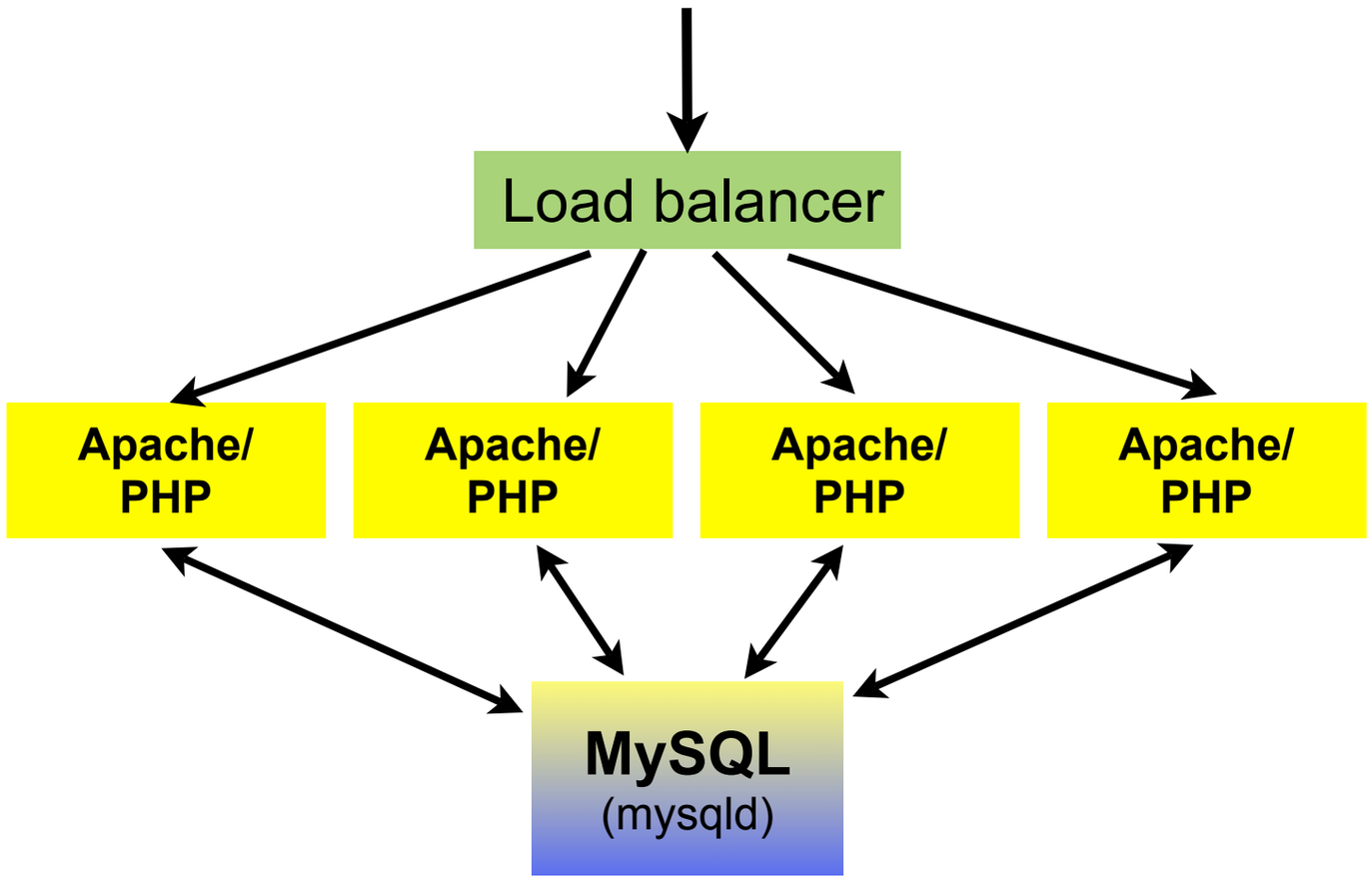
Cluster Usage Example

Web Sessions

- Use in web applications
  - shopping carts
  - personalized web pages
  - tracking of users
- Sample table for PHP sesssions

```
CREATE TABLE `store`.`sessions` (
      `id` CHAR(32) PRIMARY KEY,
      `data` TEXT,
      `usetime` TIMESTAMP ,
    ) ENGINE = InnoDB;
```

Web Sessions



- Without Cluster
  - One MySQL server holding data
  - Single point of failure

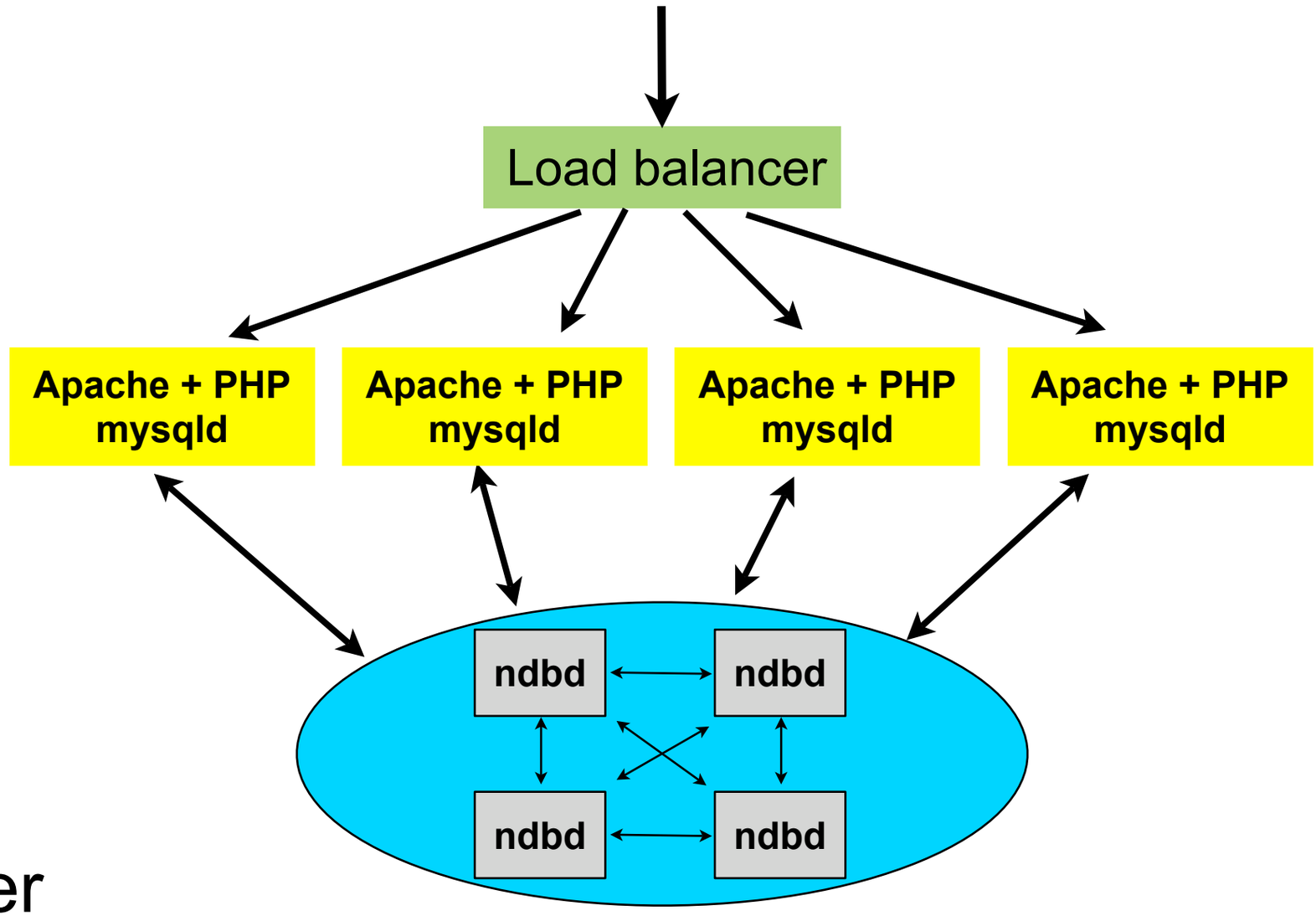
Web Sessions using Cluster

- Changing existing session table:

```
ALTER TABLE `store`.`sessions` ENGINE = NDBCLUSTER;
```

- No changes to the web application
- Each webserver runs a MySQL server
- Storing Gigabytes of session data
- Using inexpensive hardware

Web Sessions



- Wit Cluster
  - MySQL distributed
  - No Single point of failure
  - Shared storage, but redundant

Some New Features in 5.1

- Disk based storage (none indexed data)
- Auto-discovery tables and databases
- Cluster replication
- Variable-sized attributes (save space!)
- User defined partitioning
- Online add/drop index
- Better documentation

Get ready to Cluster!

- Docs: http://dev.mysql.com/manual/
- Mailinglist: cluster@lists.mysql.com
- Download: http://dev.downloads/mysql/5.0.html
  - or go latest MySQL 5.1 (beta currently)
- Standard binaries ‘mysql-max’ version
- RPM based:
  - Linux x86 generic RPM (dynamically linked)
  - MySQL-Max
  - The MySQL-ndb* packages
- From source:
  - ./configure --with-ndbcluster

Q&A

And Thanks!