Introduction to MySQL Cluster: Architecture and Use

Arjen Lentz, MySQL AB (arjen@mysql.com)
(Based on an original paper by Stewart Smith, MySQL AB)

An overview of the MySQL Cluster architecture, what's different about it and what problems it can be used to solve. We'll be looking at how High Availability is achieved as well as setup considerations regarding performance. Basic use will also be covered, from setup (including schema considerations) to new (and exciting!) features in the 5.1 release.

Introduction

MySQL Cluster is a fault tolerant in-memory clustered database designed for 99.999% availability and fast automatic fail over all running on cost effective commodity hardware.

MySQL is an open source ACID (Atomicity Consistency Isolation Durability) compliant Relational Data Base Management System (RDBMS) aiming towards full SQL standards compliance. It has a reputation for ease of use, speed, quality and reliability and consequently is the world's most popular open source database with over eight million installations.

At time of writing (March 2005) 5.1 is in beta and 5.0 is the production release. By publication, a 5.1 production release may be available. MySQL Cluster has been included since 4.1.3 and in RPMs since 4.1.10a (there are separate cluster RPMs that need to be installed). It is included as part of the MySQL-Max distribution (this is different to the MaxDB product - which is a separate RDBMS by MySQL AB). MySQL AB officially supports MySQL Cluster on Linux, MacOS X and Solaris. Users have reported success with FreeBSD. Support for all MySQL platforms in the future.

The NDB Storage Engine

Storage engines are a unique architectural feature of MySQL. The VFS layer of your operating system allows applications to access files on different file systems through the one interface, MySQLs' storage engine architecture allows applications to access data stored in different ways all through the same SQL interface. Two commonly used storage engines are MyISAM (fast inserts and selects, full text indexes, GIS) and InnoDB (row-level locking, multi-version concurrency, ACID compliant).

MySQL Cluster provides a new storage engine for MySQL. The NDB (also known as ndbcluster) storage engine provides high availability in a shared-nothing architecture. Since there is no shared or special hardware (such as a SAN), MySQL cluster can easily be implemented on affordable commodity hardware. All data is synchronously replicated between nodes. The NoOfReplicas configuration parameter dictates how many copies of the data are kept in the cluster.

In the 4.1 and 5.0 releases, all data must be held in main memory on the nodes. A rough estimate of the memory needed in each node is

(SizeofDatabase * NumberOfReplicas * 1.1) / NumberOfDataNodes

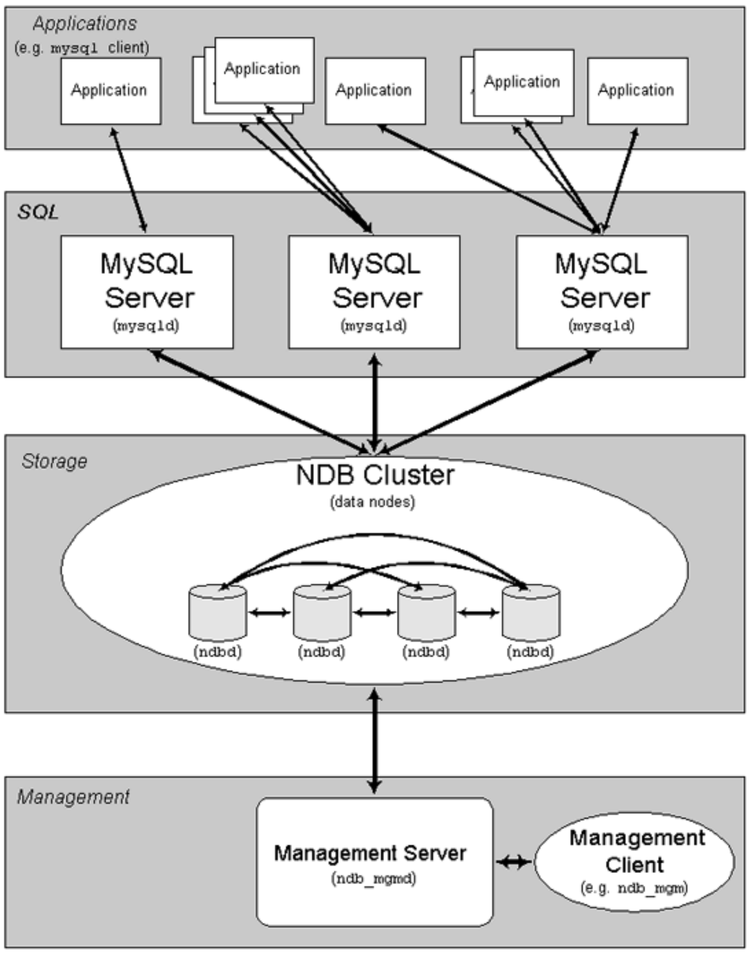
A transaction is committed when it is in memory of more than one node (i.e. it can survive a node crash). The in-memory architecture has the advantage of being very fast, with a single CPU core able to managing over 10,000 transactions per second.

Basic persistence is provided by periodically writing checkpoints to disk. The timing between checkpoints (among other things) is configurable. It is also possible to perform online backups of data in the cluster. After a system failure (where enough nodes have failed that the cluster no longer has a full data set) the system is restored to the last global checkpoint. When a single node is being restarted (node recovery) it will fetch the latest data from another node in the cluster.

The 5.1 release allows non-indexed fields to be stored on disk. In the future, it will be configurable if you want a disk or main memory based cluster. In current releases you can also configure to run in Diskless mode, where no data is ever written to disk (no checkpointing, no logging).

Components of a Cluster

A fairly basic cluster setup might look like this:



Applications

Applications connect to the cluster via a MySQL server exactly the same way as they would if they were using any other MySQL storage engine. Using the cluster is transparent to the end application.

It is up to the connector or application to load balance between the MySQL nodes. Some people use load balancing products, others use features of the connector (e.g. the MySQL JDBC driver) and others write their own small snippets of code to load balance and fail over.

In the event of node failure, all transactions that were using that node are aborted and it is up to the application to restart them.

MySQL Server nodes (mysqld)

Multiple MySQL servers can connect to the one cluster. This provides redundancy and increased performance due to parallelism. When an update is performed on one MySQL server it is immediately viewable from other MySQL servers attached to the cluster.

Data Nodes (ndbd)

All data is stored by the data nodes. This data is visible to all the MySQL servers connected to the cluster. Some MySQL special data such as the permissions and stored procedures are not stored in the cluster and must be updated on each MySQL server attached to the cluster.

Management Server Nodes (ndb_mgmd)

The management server provides configuration information to nodes joining the cluster. It is not a critical part of the cluster, only needing to be up for a node to join the cluster.

Management Client (ndb_mgm)

An end-user tool for administering and checking the status of the cluster. This can be used for starting and stopping nodes, getting status information and starting backups.

Node Interconnects

MySQL cluster supports several methods of transferring messages between the nodes. The most common is the TCP Transporter over 100Mbit or 1Gbit Ethernet. There can be significant performance gains in using gigabit Ethernet. You can also use SCI, which is a high-performance, low latency interconnect. There is also a shared memory transporter for nodes running on the same host, although it should currently be considered experimental.

Since communications between nodes are unencrypted, it is recommended to have a private network exclusively for cluster traffic. This also ensures that no other systems interfere with the performance of the cluster.

Sample Configuration

Below, additions to the my.cnf file are shown for connecting to a cluster. The connect string tells processes where management servers are so that they can join the cluster. In this case, we have one management server, so the connect string is just a host name.

```
# my.cnf
# example additions to my.cnf for MySQL Cluster
# (valid from 4.1.8)
# enable ndbcluster storage engine, and provide connectstring
for
# management server host (default port is 1186)
[mysqld]
ndbcluster
ndb-connectstring=ndb_mgmd.mysql.com
# provide connectstring for management server host (default
port: 1186)
[ndbd]
connect-string=ndb_mgmd.mysql.com
# provide connectstring for management server host (default
port: 1186)
```

```
[ndb_mgm]
connect-string=ndb_mgmd.mysql.com
# provide location of cluster configuration file
[ndb_mgmd]
config-file=/etc/config.ini
```

config.ini is the cluster configuration file (example below). In this setup we have two data nodes and two replicas. This means that each node holds a complete copy of the database. Here we allow up to three MySQL servers to connect to the cluster.

```
[NDBD DEFAULT]
NoOfReplicas= 2
DataMemory= 500M
IndexMemory= 100M
DataDir= /var/lib/mysql-cluster
[NDB_MGMD]
Hostname= ndb_mgmd.mysql.com
DataDir= /var/lib/mysql-cluster
[NDBD]
HostName= ndbd_2.mysql.com
[NDBD]
HostName= ndbd_3.mysql.com
[MYSQLD]
[MYSQLD]
[MYSQLD]
```

Failure Scenarios

If a MySQL Server node fails, it can be restarted which will reconnect it to the cluster. While the MySQL server is down, applications can connect to other MySQL servers connected to the cluster.

If a data node fails, other nodes discover the failure due to missed heartbeats. Since all data is synchronously replicated within the cluster, new transactions can use another storage node which has the same data as the failed one.

The continued operation of the cluster is not dependent on the Management Server. The management server only needs to be up when nodes are joining the cluster. You can however, have multiple management servers in the cluster.

Schema Considerations

There are several things you need to be aware of when designing (or adapting) database schemas to MySQL Cluster.

Indexes

There are three types of indexes in MySQL Cluster:

- Primary hash index

- Unique hash index
- Ordered tree index

Specifying a unique index from SQL will also create an ordered index unless USING HASH is specified.

Primary Keys

Every table has a primary key. If you declare a table in SQL without a primary key, NDB creates a hidden primary key. The primary key will have a primary hash index and an ordered index. You can specify USING HASH to skip the ordered index.

Space Usage

Mostly, the standard MySQL rules apply (as documented in the manual). However, a few things need to be considered:

- Prior to 5.1, only fixed size rows are supported, so a VARCHAR(255) column will use 260 bytes of storage no matter how much data is stored in it.
- BLOB and TEXT columns use 256 bytes of fixed space in the row. Additional space is allocated in 2kb chunks as needed.
- In 4.1, each primary key or hash index requires 25 bytes plus the size of the key per record of IndexMemory. The 5.0 release reduces this to 21 to 25 bytes per record. Each ordered index requires 10 bytes of DataMemory per record.
- Warnings are written to the cluster log when 80% of DataMemory or IndexMemory has been used. Warnings are again printed at each 5% increment of usage. A method to query this information from the management client will be implemented in the future.

New to 5.0

With condition pushdown enabled (via SQL: SET engine_condition_pushdown = 1; or through the mysqld configuration option –engine-condition-pushdown), conditions in the WHERE clause of an SQL statement can be evaluated on the data nodes instead of within the MySQL server. This can improve performance by five to ten times for such queries. This is due to savings in network traffic and the ability to execute the query in parallel.

Less IndexMemory is required as the primary key is no longer stored in the index memory.

The MySQL Query Cache now works with Cluster.

New to 5.1

With MySQL version 5.1, it becomes possible to use MySQL replication to replicate tables stored on the cluster. Replication is handled by one MySQL server connected to the cluster which records all updates to the cluster (from any of the MySQL servers) into its own binary log.

Non-indexed attributes may be stored on disk instead of RAM. Indexed fields still have to be in memory and also the primary key hash index but all other fields can be on disk.

Variable sized records will be supported. Currently, a VARCHAR(255) column would use 260 bytes of data no matter what the length of the actual record was. In MySQL 5.1 the minimum storage space will be used. This can reduce the average storage requirements by sometimes up to a factor of five.

Users will be able to define partitions based on the fields of the primary key. Partitioning based on KEY, HASH, RANGE and LIST handlers will be supported.

Finding out more

The MySQL Cluster product page on the mysql.com web site provides a good overview of the product including data sheets, white papers and customer success stories. Also mentioned here is the range of support and training options available from MySQL AB.

The MySQL Reference Manual provides detailed documentation on the MySQL server setup as well as NDB. The MySQL Cluster chapter is a good starting place if you are already familiar with MySQL. A HOWTO is also available as an article on the MySQL Developer Zone.

Two forums are frequented both by developers and users. There is a web based forum and a mailing list. As always, it's a good idea to read the FAQ in the manual and the list archives before asking questions - a lot of things have already been covered.

References

- MySQL Reference Manual (MySQL AB), http://dev.mysql.com/doc/mysql/
- HOWTO set up a MySQL Cluster for two servers (Alex Davies, 2005), http://dev.mysql.com/tech-resources/articles/mysql-cluster-for-two-servers.html
- MySQL Forums :: Cluster, http://forums.mysql.com/list.php?25
- MySQL Lists: cluster, http://lists.mysql.com/cluster
- MySQL Cluster product page, http://www.mysql.com/products/cluster/