The FuzzyPhoto MySQL Cluster

Table of Contents

1. Synopsis.....1

2. Background.....1

3. Hardware Structure.....2

3. Software Structure.....3

4. Initiating Cluster.....3

5. Importing and Migrating MySQL databases from innoDB or MyISAM to NDB.....6

1. Synopsis

This document has been constructed to assist users of the MySQL cluster established for the purposes of the FuzzyPhoto project. The cluster is hosted in the Portland Building within De Montfort University. The document will assume some previous experience with Information Technology (I.T) and SQL commands. It will cover:

- Background to the purpose of the MySQL cluster.
- The hardware structure of the MySQL cluster.
- The software structure of the MySQL cluster.
- The process of initiating the cluster.
- The process of restarting the cluster.
- How to migrate a database onto the cluster from a standard MySQL export.

2. Background

The FuzzyPhoto MySQL cluster serves as the main data repository for the FuzzyPhoto project. Within the cluster a composition of data from the Exhibitions of the Royal Society 1870 – 1915 (stored at De Montfort University) and partner organisations is stored. The data is composed of meta-data relating to historical photographs within these collections. The cluster also serves to assist in the processing of links between these collections. The structure of the cluster was to maintain data isolation from external access along with allowing continued,

sustained expansion of core structure. Figure 1 shows how the cluster relates to other elements within the process. The main output from the cluster is a links database. Users can access this database via a web server (more detail is given in additional documents). The links database is periodically updated from the cluster through a defined link.

3. Hardware Structure

The FuzzyPhoto is constructed across three Dell PowerEdge R320 servers. Each server has the below specification:

- CPU – Xeon E5-2403 1.80GHz, 10M Cache.
- Network – Broadcom 5720 Dual port 1GB ethernet.
- Memory – 4GB UDIMM, 1333MHz.
- Hard Drive – 2 x 500GB SATA, RAID 1 configured.

Each server runs a separate installation of the Ubuntu 12.04 LTS Precise Pangolin server. The hardware structure is shown in Figure 1.

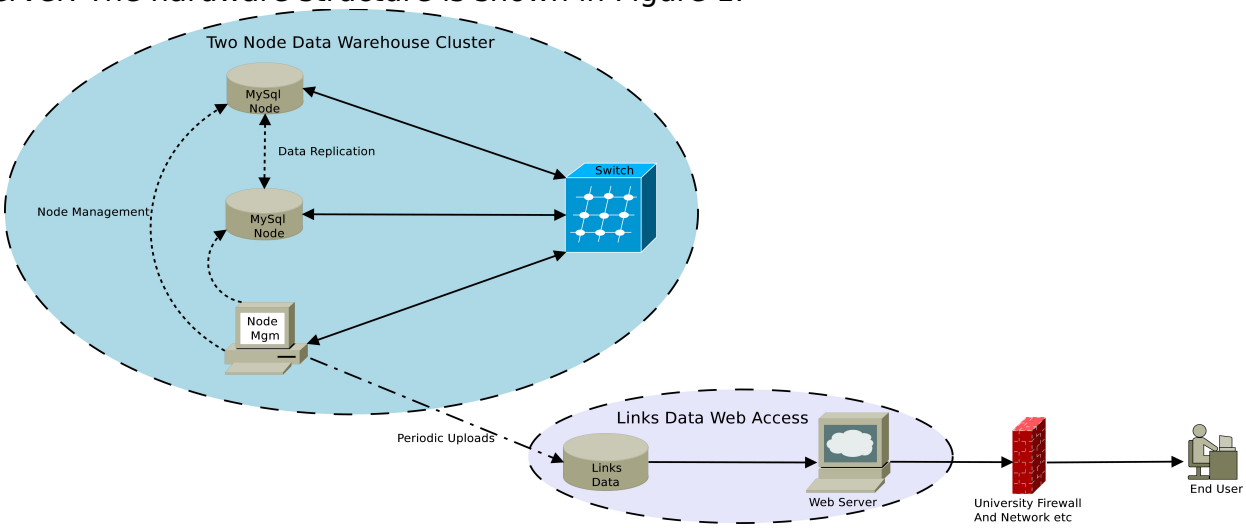

Figure 1. Structure of the FuzzyPhoto Cluster.

The cluster is composed of three physical devices. The three servers house separate components of the cluster structure alongside a web server used within the FuzzyPhoto project. The web server is isolated from the cluster through both software and hardware interfaces. Each of the three servers are linked via a gigabit switch. Although the web server is hosted on one of the servers, it is not physically linked to the cluster network.

The cluster is controlled via a KVM switch. This allows access from a single terminal. In position 1, the switch relates to the web server / manager, position 2 it relates to cluster slave 1 and position 3, it relates to cluster slave 2.

3. Software Structure

The cluster is composed of three separate elements: Management, Data and SQL Nodes.

3.1 Management Node

The MySQL management node resides on kmd4. Unlike the other devices it is a virtual machine hosted on the FuzzyPhoto web server. This allows isolation of the system from user access. The management node is also isolated through separate network access. In MySQL cluster, there is no single central server or process. All the agents that are involved in the process collaborate in managing the cluster as a whole. The management node is responsible for:

- Starting, stopping, and restarting cluster nodes.
- Cluster configuration.
- Cluster software upgrades.
- Host and node status reporting.
- Recovery of failed nodes.

3.2 Data Node

The data nodes are used to store the data within the cluster structure. Tables are automatically sharded (horizontal partitioning of data) across the data nodes which also manage load balancing, replication, failover and self-healing.

3.3 SQL Node

The SQL nodes act in a similar manner to standard SQL databases. They are MySQL servers that connect the data nodes in order to perform storage and retrieval of information.

4. Initiating Cluster

The process of initialising and starting the cluster needs to be carried out in a set order. This is:

1. Start the management node.
2. Initialise the data nodes.
3. Connect the SQL nodes.

This process will be described taking each step in turn highlighting the necessary command-line statements to use. The command-line statements will be given as

```
cd /usr/local/
```

To enter a command, first open a terminal. This can be accomplished by pressing *ctrl + alt + t.*

4.1 Start the Management Node

Switch the KVM to position 1. This will allow the keyboard and mouse to connect to KMD4. Within the KMD4 operating system is contained in a Virtual Machine (VM). A VM is a software based operating system that emulates the functions of a real world computer architecture. The VM is run using Virtual Box. The management node VM can be started from the command-line:

```
vboxmanage startvm clusterManager
```

where `clusterManager` is the VM being started.

4.1.1 Locate the Correct Directory

Using the command-line, navigate to the location of the management node.

```
cd /usr/local/mysql
```

4.1.2 Initialise the Management Server

In the terminal, run the following command:

```
./bin/ndb_mgmd -f configini --initial
--configdir=/var/lib/mysql-cluster #Note: all one line
```

The ndb_mgmd calls the cluster management server. The `-f` and `--configdir` options call a configuration file for the server from the defined location. Using the `--initial` option forces the server to read from the configuration file each time it is loaded rather than reading the global configuration.

4.1.3 Start the Management Client

In the terminal, run the following command:

```
sudo ./bin/ndb_mgm
```

This command must be run with root privileges. This will start the management client process. This is not required to mange the cluster but is used to monitor the processes.

To view the connections to the manager, enter

```
SHOW
```

At this point there will be no connections shown.

4.2 Start the Data and SQL Nodes

On each of the additional servers there resides both a data node and a SQL node. The following commands need to be carried out on both servers to start the nodes. Use KVM switch position 2 and 3 to operate the input devices on cluster slaves 1 and 2 that house both data and SQL nodes.

4.2.1 Start the Data Node

Using the command-line, navigate to the location of the data and SQL nodes.

```
cd /usr/local/mysql
```

In the terminal, run the following command:

```
sudo ./bin/ndbd --initial
```

This command must be run with root privileges. This will restart the data node using an initial restart process.

Following on, enter this command:

```
sudo ./bin/mysqld --user=root &
```

This command must be run with root privileges. This will restart the data node using an initial restart process.

4.3 Managing the Connections

The management client can be used to view and manage the connections to the cluster. This is started by running the commands in Section 4.1.3. Below are a set of commands that can be run to assist in managing the cluster.

- `HELP` *– displays information about all available commands.*
- `SHOW` *– Displays information on the clusters status. Use this after starting the data and SQL nodes to check the connections have been started successfully. Possible node statuses include* `UNKNOWN, NO_CONTACT, NOT_STARTED, STARTING, STARTED, SHUTTING_DOWN and RESTARTING.`
- `node_id START` *– This brings online a specified node.*
- `ALL START` *– Brings all nodes online except management nodes.*
- `node_id STOP` *– Stops the specified management or data nodes*
- `node_id RESTART` *– Restarts the specified node. Options can be used including* `-i` *for an initial restart.*
- `node_id STATUS` *– Gives the status of the specified node.*
- `MemoryUsage` *– Gives the memory usage and index memory being used by each data node*
- `SHUTDOWN` *– Shuts down all the cluster data nodes and management nodes.*
- `EXIT, QUIT` *– Terminates the management client.*

5. Importing and Migrating MySQL databases from innoDB or MyISAM to NDB

For replication to occur across the cluster, a different engine (type of table) to those often used on a desktop or server configuration needs to be used. For a table to be replicated the engine needs to be defined as `NDBCLUSTER` or `NDB`. A table defined in either `innoDB` or `MyISAM` can be added to the cluster but it will not be replicated. The following process will describe how an SQL dump can be imported and converted to the required engine format. A MySQL dump is a program that can be used to dump a database or a collection of databases for backup or transfer to another SQL server. The dump contains SQL statements to create and/or populate the required tables.

5.1 Import Database to SQL nodes

To import a SQL dump initially there maybe a requirement to uncompress the file. On each data node, uncompress the files in an accessible directory. Depending on the format, this can be achieved using the below statements.

*If the file is *.tar*

```
tar xvf archive_name.tar
```

*If the file is *.tar.gz*

```
tar xvfz archive_name.tar.gz
```

*If the file is *.tar.bz2*

```
tar xvfj archive_name.tar.bz2
```

Once the file(s) have been uncompressed, they can be imported directly onto the cluster SQL servers. The first step in the process is to create a database on the server to house the tables. Firstly, connect to the server using the below command.

```
./mysql - h 127.0.0.1 -P 3306
```

The h option is the host (local in this instance) and P is the port. This is the defined port for the server.

Next a database is created using the below command, followed by exiting the SQL command-line.

```
CREATE DATABASE database_name;
EXIT;
```

The tables from the uncompressed SQL dump can be imported using the following command where `database_name` is the database created and `tables.sql` is the uncompressed SQL dump.

```
sudo ./mysql – h 127.0.0.1 -P 3306 database_name < tables.sql
```

Once the database has been updated, the tables engine needs to be converted. This can be achieved by re-connecting to the SQL server.

```
./mysql - h 127.0.0.1 -P 3306
```

The following SQL command outputs a list of SQL commands that can be copied to the command-line to alter all of the tables within the database.

```
SELECT CONCAT ('ALTER TABLE', table_name, ' ENGINE=NDB;') AS
sql_statements FROM information_schema.tables AS tb WHERE
table_schema = database_name ORDER BY table_name DESC;
```

Within the command database_name needs to be replaced by the database being used. This will return a list of SQL statements. These can be copied into the SQL server command-line to convert all of the tables within the database. To assist in the copy and pasting of the commands into the command-line, copy the statements into the gedit text editor. Replace all "|" with "". This will remove all unneeded characters. The resulting statements can be copied directly into the command-line.

If a single table needs to be converted, this can be achieved using the following command.

```
ALTER some_table ENGINE=NDB;
```

This will change the engine of `some_table` to the required engine so that the cluster will replicate it across all nodes.