Two Node MySQL Cluster

1.0 EXECUTIVE SUMMARY

This white paper describes the challenges involved in deploying the 2 node High Available MySQL-Cluster with a proposed solution. For the sake of users reading this document it also describes in brief the main components of the MySQL Cluster which are necessary to understand the paper overall.

The solution relies on the Linux HA framework (Heartbeat/Pacemaker) so the white paper can be best understood with the knowledge of Linux HA framework.

2.0 BUSINESS CHALLENGES

The MySQL cluster demands at least 4 nodes to be present for deploying a High Available MySQL database cluster. The typical configuration of any enterprise application is a 2 Node solution (Active-Standby mode or Active-Active Mode). The challenge lies in fitting the MySQL Cluster Nodes in the 2 Nodes offering the application services and to make it work in that configuration with no single point of failure.

3.0 MYSQL CLUSTER

The intent of this section is to briefly mention the important actors and their roles in the overall MySQL Cluster. For more information the reader can refer to the MYSQL reference documents from its official site (http://dev.mysql.com/doc/index.html).

MySQL Cluster is a technology that enables clustering of in-memory databases in a “shared-nothing system”. By “shared-nothing” it is meant that each node has its own storage area. Note that the use of the shared storage mechanisms such as network file systems are not supported and recommended by MySQL. For High Availability there are at least 2 nodes on which the data is stored locally so that in case of one node failure the data is still accessible from the other available nodes. One of the data node acts as “Master” and the rest ones act as “Slaves”. These data nodes communicate with each other to replicate the data across the data nodes so that the change done on one node is also visible on rest of the nodes.

MySQL Cluster integrates the standard MySQL server with an in-memory clustered storage engine called NDBCluster (Network DataBase).
There are the following actors in a MySQL Cluster:

1. Clients/APIs
2. SQL Nodes
3. Data Nodes
4. NDB Management Node

The figure below depicts these actors with the corresponding interactions.

3.1 Clients/APIs

These are the clients/applications that actually use the MySQL Cluster. They connect through various methods such as MySQL C APIs, Java connectors. All of these clients connect to the SQL Nodes which in turn connects with the Data Nodes for accessing any data stored in the MySQL Cluster.

There is a special API provided by MySQL known as NDBAPI which directly connects with the Data Node bypassing the SQL Node. This increases the read/write operations performance significantly and is recommended to be used for high throughput applications which require very high rates for read/write.

3.2 SQL Node

This is a node that accesses the cluster data. In the case of MySQL Cluster, an SQL node is a traditional MySQL server that uses the NDBCLUSTER storage engine. An SQL node is a mysqld process started with the `--ndbcluster` and `--ndb-connectstring` options.

3.3 Data Node

This type of node stores cluster data. There are as many data nodes as there are replicas, times the number of fragments. For example, with two replicas, each having two fragments, you need four data nodes. One replica is sufficient for data storage, but provides no redundancy; therefore, it is recommended to have 2 (or more) replicas to provide redundancy, and thus high availability.
3.4 NDB Management Node

The role of a management node is to manage the other nodes (Data Nodes, SQL Nodes and Client Nodes using NDBAPI). The Management node is the first node to be started in a MySQL Cluster before starting any other node. The functions it performs are:

1. Providing configuration data to other nodes.
2. Starting and stopping nodes.
3. Starting the cluster.

3.5 Challenges

This section lists the challenges that were faced during the implementation:

1. MYSQL requires at least four nodes for the cluster to work ([http://www.mysql.com/products/cluster/faq.html#11](http://www.mysql.com/products/cluster/faq.html#11))

   **11. How many physical servers are needed to create a minimum Cluster configuration?**

   **A:** For evaluation and development purposes, you can run all nodes on a single host. For full redundancy and fault tolerance, you would need a minimum 6 x physical hosts:
   
   - 2 x data nodes
   - 2 x SQL/NoSQL Application Nodes
   - 2 x Management Nodes

   *Many users co-locate the Management and Application nodes which reduces the number of nodes to four.*

   This was a major limitation as the Design and Architecture demands for 2 nodes. The challenge was to accommodate all the MySQL Cluster components into 2 nodes in a High Available environment. The limitation is due to the three node architecture of Mysql Cluster. Running all the nodes (Data/SQL/Management) on one node and having one standby for them posses following potential problems:

   - The failure of one node will cause failure of all the three nodes at once causing the whole cluster to fail. What MySQL guarantees is that there is no single point of failure but does not guarantee if two nodes fail at the same time, for e.g. one Data Node and one Management Node failing precisely at the same time.

2. Correct set of configuration for all the MYSQL Cluster components in both the nodes so that the cluster can work properly without any issues. Note that there are always high chances of incorrect configuration as the IP Addresses were always be different as per the customer setups.

3. The failover of the Management Server. There was one Management Server that has to run in two nodes. This could cause a single point of failure. Although a cluster can still continue after it is started and the Management Server fails but still there are some important tasks that are done by the Management Server which are required at runtime also:
   
   a. Provide status information about the cluster and allow to use the ndb_mgm for various maintenance tasks like taking a hot backup
   b. Own the cluster config file (therefore it must be running to start a node)
   c. Arbitration in case of a potential split-brain
   d. Logging

4. Incorrect source IP Address chosen by the NDBD and MySQLD for connecting to the Management Server. The NDBD and the MySQLD components use ANY Address to bind to. This was a necessity but it caused issues when they try to connect to the Management Server. The Management server binds to a Virtual IP (REPNW_VIP) which is accessible from the other node also. But when the MySQLD and NDBD tries to connect with Management Server it chooses the source IP of the TCP packet as the REPNW_VIP itself on the node which has the REPNW_VIP
associated with it. So essentially the packet’s source and destination IPs are same (REPNW_VIP). The Management Server could not find the corresponding node in its configuration and replies back with error causing the NDBD and MySQLD to shutdown. Note that NDBD and MySQLD can bind to a specific IP address in case of multiple interface case which could have solved this issue but for running it on two node it was a necessity to bind them on ANY Address.

3.6 Solution

This section describes the solution that was implemented in Mexsat for the challenges mentioned above.

1. The configuration implemented was to run the MySQLD and NDBD on each node. The Management Server was run on the Active Node (Note that the 2 nodes run in Active-Standby configuration). The Active Node is the one which is externally reachable through a virtual IP (External VIP) through which the traffic for the applications will come. The REPNW_VIP is also collocated with this External VIP so that the management server can bind to it. These constraints were possible through the Pacemaker configuration. All the resources running in a node are configured as a resource for pacemaker where constraints on the resources are put through the pacemaker configuration.

Presented below is a sample configuration file (Note that the application resources are removed from this configuration).

```
configure
property no-quorum-policy=ignore
property stonith-enabled=true
rsc_defaults resource-stickiness=INFINITY
rsc_defaults migration-threshold=50
primitive fence_node1 stonith:external/ipmi params hostname="PttCP-Node1" ipaddr="10.1.127.17" userid="Administrator" passwd="YBCOKY4T" passwd_method="param" interface="lanplus" op monitor interval="40" timeout="200" on-fail="restart"
```
The MySQL cluster is highly dependent on the IP Addresses, host names and node ids configuration of all the nodes. That means that the configuration has to be correct across all the nodes and the management server has to have the information about the SQL, NDB, API nodes and the other nodes needs to know where the Management server is.

Since this information is not static (Note that the configuration can change in the deployment setup with respect to host names and IP Addresses) so it has to be configured during the installation process only.

The installation process implemented was a 2 step process.

a. Golden Disk Installation: This involves installation of the OS and the all the rpms necessary to run. During this step MySQL cluster rpms are also installed and the default configuration files are placed at relevant path.

b. Application Package Installation: This involves installation of the application and configuring the system. All the dynamic information is taken as an input from the user and the relevant configuration files with respect to MySQL are updated during this process.
Presented below is a sample configuration file for mysqld and Management Server.

```plaintext
[[mysqld]
ndbcluster                      # run NDB storage engine
ndb-connectstring=rpnetwork_host_vip
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
user=mysql
# Disabling symbolic-links is recommended to prevent assorted security risks
symbolic-links=0
pid-file=/var/lib/mysql/mysqld.pid
default-storage-engine=NBBCluster

[mysql_cluster]
# Options for ndbd process:
ndb-connectstring=rpnetwork_host_vip

[mysqld_safe]
log-error=/var/log/mysqld.log
pid-file=/var/lib/mysql/mysqld.pid

[client]
socket=/var/lib/mysql/mysql.sock

[ndbd]
# Options affecting ndbd processes on all data nodes:
NoOfReplicas=2    # Number of replicas
DataMemory=80M    # How much memory to allocate for data storage
IndexMemory=18M   # How much memory to allocate for index storage
    # For DataMemory and IndexMemory, we have used the
    # default values. Since the "world" database takes up
    # only about 500KB, this should be more than enough for
    # this example Cluster setup.
Arbitration=WaitExternal
ArbitrationTimeout=3000
StartPartialTimeout=5000
StartPartitionedTimeout=6000

[tcp]
# TCP/IP options:
portnumber=2205   # This the default; however, you can use any
    # port that is free for all the hosts in the cluster
    # Note: It is recommended that you do not specify the port
    # number at all and simply allow the default value to be used
    # instead

[ndb_mgmd]
NodeId=1
# Management process options:
hostname=rpnetwork_host_vip     # Hostname or IP address of MGM node
datadir=/var/lib/mysql-cluster  # Directory for MGM node log files
ArbitrationRank=0

[ndbd]
```
3. The failover of the management server is controlled through pacemaker. The Management Server needs to be run on the “Active” Node in a “Active/Standby” two node configuration. This is ensured by making Management Server a primitive resource in Pacemaker and adding a collocation rule in Pacemaker with the VIP. Note that the “Active” Node always hosts the “VIP”.

```
colocation ndb_vip -with-ndb_mgm inf: NDB_MGMT REPNETWORK_VIP
```

This rule ensures that the Management Server is run where the VIP resides. This also ensures the movement of the Management Server to the other node in case of Failover/Switchover of the Active Node when the VIP moves.

Note that all the other nodes contact the Management Server on the VIP only so for them they do not see this transition happening and the cluster keeps on running in case of the Active Node failures.

4. A simple IP tables rule solves this issue. Again since the IP Addresses can change during installation the IP table rule is also made during the installation process itself. This rule is run each time the machine reboots (entry put in rc.local).

```
num  target     prot opt source               desti
1    SNAT       tcp --  10.1.127.34         10.1.127.34         tcp dpt:1186 to:10.1.127.35
```

4.0 REFERENCES

1) [http://www.linux-ha.org/wiki/Main_Page](http://www.linux-ha.org/wiki/Main_Page)
HUGHES Systique Corporation (HSC), part of the HUGHES group of companies, is a leading Consulting and Software company focused on Communications and Automotive Telematics. HSC is headquartered in Rockville, Maryland USA with its development centre in Gurgaon, India.

SERVICES OFFERED:

Technology Consulting & Architecture: Leverage extensive knowledge and experience of our domain experts to define product requirements, validate technology plans, and provide network level consulting services and deployment of several successful products from conceptualization to market delivery.

Development & Maintenance Services: We can help you design, develop and maintain software for diverse areas in the communication industry. We have a well-defined software development process, comprising of complete SDLC from requirement analysis to the deployment and post production support.

Testing: We have extensive experience in testing methodologies and processes and offer Performance testing (with benchmarking metrics), Protocol testing, Conformance testing, Stress testing, White-box and black-box testing, Regression testing and Interoperability testing to our clients.

System Integration: As system integrators of choice HSC works with global names to architect, integrate, deploy and manage their suite of OSS, BSS, VAS and IN in wireless (VoIP & IMS), wireline and hybrid networks: NMS, Service Management & Provisioning.

DOMAIN EXPERTISE:

Terminals
- Terminal Platforms: iPhone, Android, Symbian, Windows CE/Mobile, BREW, PalmOS
- Middleware Experience & Applications: J2ME, IMS Client & OMA PoC

Access
- Wired Access: PON & DSL, IP-DSLAM

Core Network
- IMS/3GPP, IPTV, SBC, Interworking, Switching solutions, VoIP

Applications
- Technologies: C, Java/J2ME, C++, Flash/lite, SIP, Presence, Location, AJAX/Mash
- Middleware: GlassFish, BEA, JBOSS, WebSphere, Tomcat, Apache etc.

Management & Back Office:
- Billing & OSS, Knowledge of COTS products, Mediation, CRM
- Network Management: NM Protocols, Java technologies, Knowledge of COTS NM products, FCAPS, Security & Authentication

Platforms

Automotive Telematics
- In Car unit (ECU) software design with CAN B & CAN C
- Telematics Network Design (CDMA, GSM, GPRS/UMTS)

BENEFITS:
- Reduced Time to market: Complement your existing skills, Experience in development-to-deployment in complex communication systems, with key resources available at all times
- Stretch your R&D dollars: Best Shore” strategy to outsourcing, World class processes, Insulate from resource fluctuations