



Does MySQL Rock ?

Gregory Steulet . Database consultant
26.11.2007

Does MySQL Cluster scale? Does MySQL Cluster rock? It's with these two questions in mind that we focused all our tests on MySQL Cluster 5.1.20. To reflect a real production environment several situations have been evaluated on physical machines. These tests are strongly experienced based which means that almost all features have been tested in order to answer to these two questions. But before trying to answer, let's have a look on the MySQL Cluster Architecture.

MySQL Cluster architecture

MySQL Cluster uses a "share nothing" architecture which means that there is no shared component such as a "shared disk" like in an Oracle Real Application Clusters (RAC) configuration. A MySQL Cluster configuration is made of three kinds of services which are called "Nodes" in MySQL Cluster vocabulary. All these nodes could run on the same physical host, of course such a configuration isn't advised at all for a production environment.

Table fragments are stored and spread over the storage nodes. The number of fragments is equal to the number of nodes, each fragment storing 1/N of data where N is the number of nodes. They are in charge of storing data and fetching them. The parsing and executing phases is devolved to the MySQL Server. Each storage node is encapsulated inside a node group and contains the same data as the other members of this group. The replicas parameter defines how many nodes are stored inside a same node group (number of copy for each fragment).

The management node acts as a central repository for the configuration information. It is only required during configuration and administration operation such as start, stop, backup and restore. It has also other roles like handling the split brain situations (i.e both data centers are cut off from each other) . The utility `ndb_mgm` provides an interface to the management node.

The last type of nodes are the MySQL Server, nothing new, just a normal MySQL Server that finds out how to connect to the cluster from a configuration file.

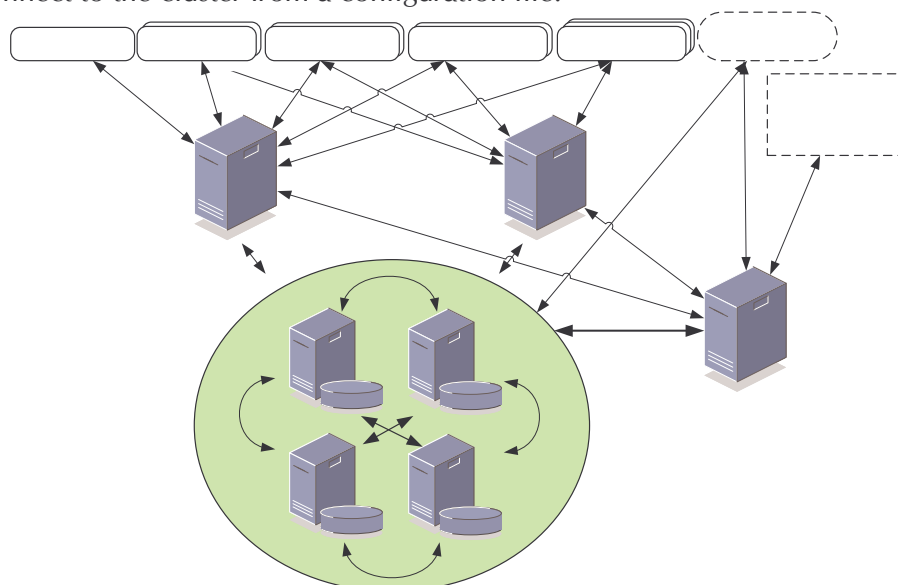


Figure 1: MySQL Cluster architecture



MySQL Cluster Scalability

The target of the scalability tests was to evaluate how a MySQL cluster performs in different configurations; it means several numbers of data nodes, replicas and MySQL server. It is important to notice that we used MySQL cluster as a “simple” customer and didn’t develop some specific application with the NDB API.

A standard test case with a number of performance tests has been defined. This test has been run against different cluster configurations. The measured metrics have been collected for comparison purpose. To complete the picture, the scenario has been run against a MySQL 5.0 cluster and against InnoDB on the same hardware. To perform all the tests the database part of the tool „sysbench“ has been used.

To simulate real life working situation we ran three different kinds of tests

- ETL test, performs a high load of data against table in order to provide results about the behaviour in a data warehouse
- OLTP test, performs a high number of read and write transaction
- Queries test, provides results regarding the speed of selects in a MySQL Cluster

For all tests previously described, we tried to see the impact of the number of data nodes, replicas and number of MySQL Server. We defined 5 setups (cf Figure 2)

Test case	# Data Nodes	# Replica	# MySQL Servers
Test case 1: Reference Case 5.0.37	8	2	2
Test case 2	8	2	2
Test case 3	4	2	6
Test case 4	2	2	8
Test case 5	8	4	2

Figure 2: Test cases that have been tested

We tuned the NDB nodes with several different parameters. During the tests we also used the Oracle recommendations regarding the network buffers (after we observed they offer some benefit also in the MySQL world ☺).

In order to compare the results, all test sceneries have been run against each setup. In each test the throughput has been measured and recorded. For comparison purpose only the result which performed the best has been considered. We observed that each time we get the best results was when we set up the number of thread to eight. The thread parameter allows simulating the number of concurrent users.

Depending on the general setup, the number of MySQL Server varies. Therefore for some tests it was possible to test higher parallelism than for others depending of the number of MySQL Servers still available. When we ran the complex OLTP tests against the cluster we encountered the error as described below (cf Figure 3) when using more than one transaction.

```
ALERT: failed to execute MySQL query: `BEGIN` :  
ALERT: Error 1296 Got error 4350 'Transaction already aborted' from  
NDBCLUSTER
```

Figure 3:Error during OLTP tests



The bug has been accepted by MySQL with number 27979. Therefore the OLTP test is based on very few result sets

Scalability results

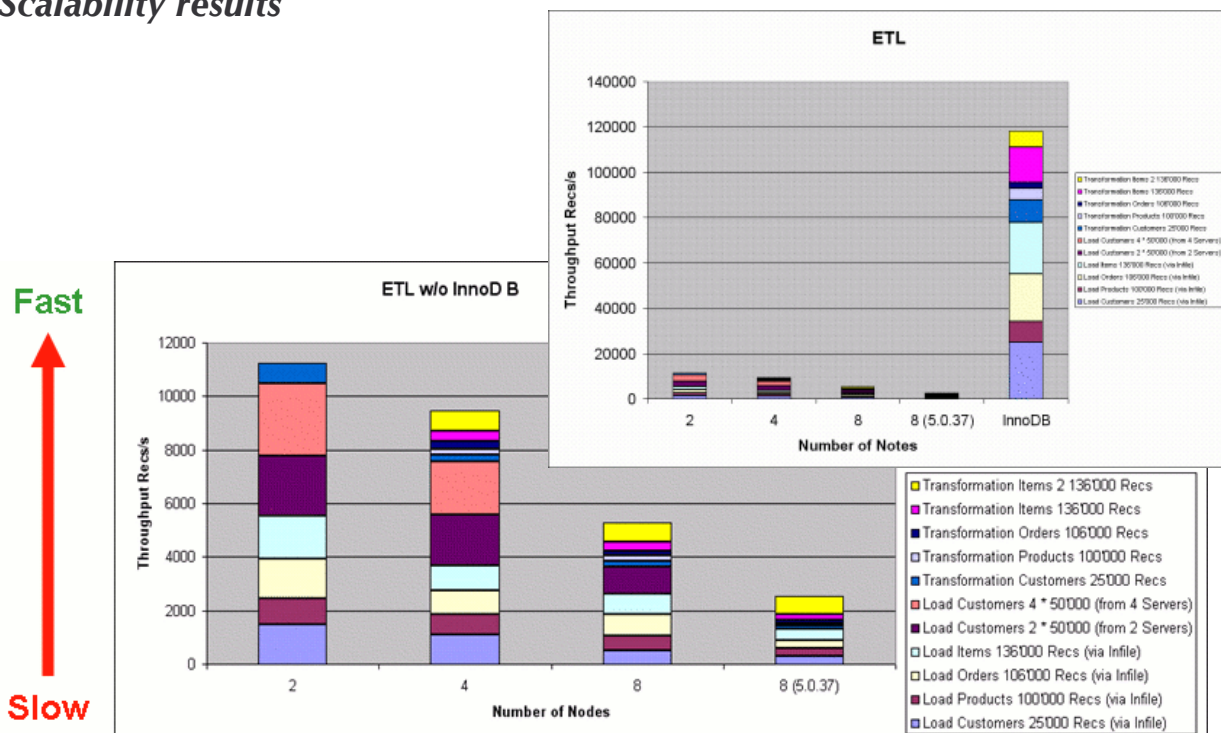


Figure 4: ETL scalability tests with tree kinds of configuration and with MySQL Cluster 5.0.37, InnoDB performs much better.

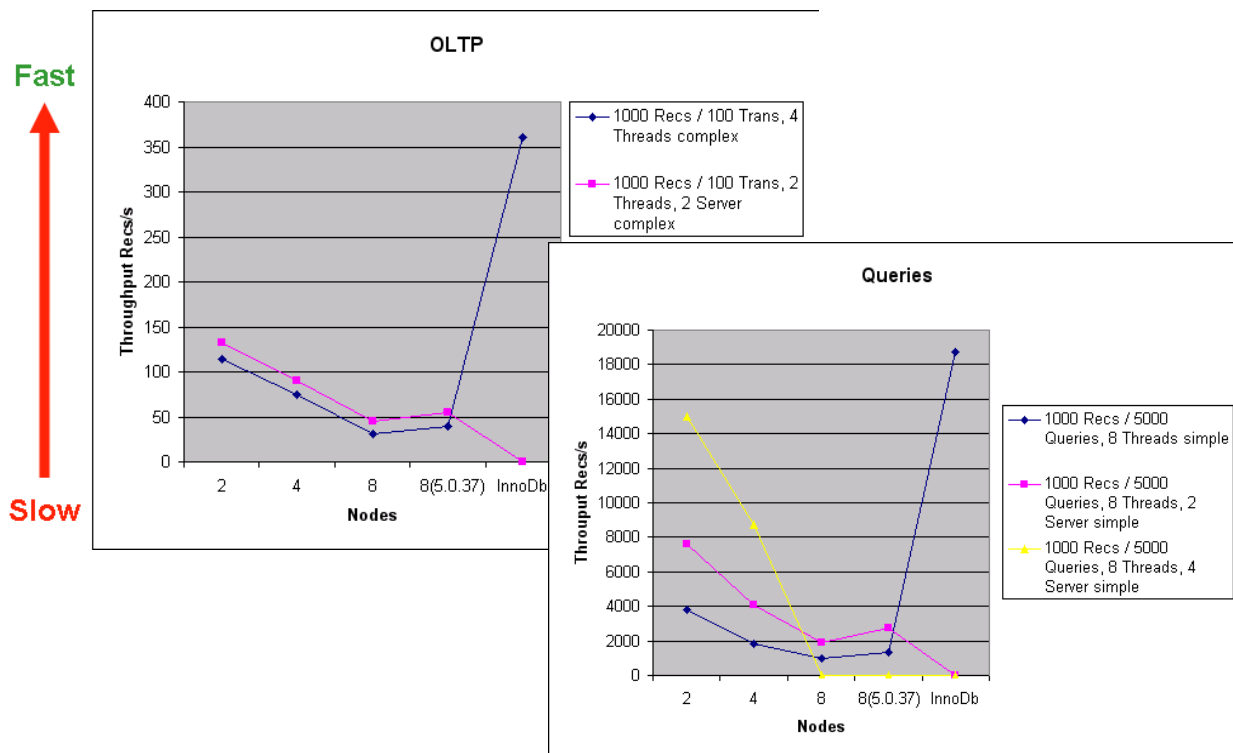


Figure 5: OLTP and Queries tests show quite similar results

Two things appear immediately if we look over those results. The first one is the more we add NDB nodes in a cluster, the worse the throughput will be. The cluster scales negatively. The



second one is even with the best configuration in terms of throughput (2 nodes) the throughput reaches barely the one of InnoDB. In addition this was only possible by using 4 MySQL servers and only for queries. Those bad results are due to the additional networking overheads especially communication between data nodes and between data nodes and mysql server. In addition when an insert happens to ensure that all NDB nodes have the correct data, NDB needs to execute a two "phase commit" procedure between data nodes. The number of messages during a two-phase commit is measured by :

$\text{Number of messages} = 2 \times \text{Number of fragments} \times (\text{Number of replicas} + 1)$
--

It proves that as either the number of fragments or the number of replicas increases, the transactions will be slower.

However we can also see that MySQL Cluster 5.1 performs twice better than MySQL 5.0 in the ETL tests which is promising for the next releases.

Summary

The best configuration we get was with a 2 NDB nodes setup. This configuration proves that NDB cluster is only slightly slower than InnoDB for queries. This might be a useful setup for a query based system that needs to achieve some high availability targets. In any cases InnoDB is faster than an NDB 5.1 cluster although NDB 5.1 seems to be faster than NDB 5.0 in ETL tests, some improvements have been achieved. Unfortunately at the moment Bug 27979 prevents the usage of NDB cluster for an OLTP system.

MySQL Cluster does not manage large transaction well, it's better to execute small transactions with few operations each rather than performing a single huge transaction with many operations.

During our tests we saw that increasing the number of replicas does not impact strongly the performances of the cluster (10-15 % observed). The only advantage of adding new data nodes is to allow more data to be loaded because MySQL Cluster 5.1 doesn't scale at all, in contrary it scales negatively.

MySQL Cluster Availability

Does MySQL rock? To answer to this question we have had to organize several tests. During those one we simulated situations which could happens in each day life and evaluated how the cluster reacts. These cases are the following:

- Lost of one NDB node in one node group
- Lost of two NDB nodes in two different nodes groups
- Lost of a NDB node group
- Lost of the active management server
- Lost of all management servers
- Lost of the MySQL server

Loosing one NDB or two NDB nodes in different node groups has no impact on the cluster availability but the running statements are impacted as well as the clients who launched the statements. The client still has access to the metadata but no more to the NDB engine, we have to restart our mysql client. If the master NDB node is impacted, another node will be automatically elected as master, based on which data node has been running the longest. When the NDB node comes back into the cluster, the recovery phase, which is statement-based, happens automatically



and won't impact the running statements. However the recovery phase seems to be much longer when transactions are running.

We can loose one node per node group without impacting the cluster availability, only clients who were executing statements will be impacted and will have to restart their MySQL Client. But now what happens when we loose all nodes in a same node group ? In theory, the cluster becomes inconsistent because we loose all copies of fragments and therefore data also. Practically, MySQL Cluster decides to shut down automatically all the NDB nodes (cf Figure 6) rather than staying up and providing wrong results.

```
ndb_mgm> Node 3: Forced node shutdown completed. Caused by error 2305: 'Node lost connection to other nodes and can not form a unpartitioned cluster, please investigate if there are error(s) on other node(s) (Arbitration error). Temporary error, restart node'. - Unknown error code: Unknown result: Unknown error code
...
Node 10: Forced node shutdown completed. Caused by error 2305: 'Node lost connection to other nodes and can not form a unpartitioned cluster, please investigate if there are error(s) on other node(s) (Arbitration error). Temporary error, restart node'
```

Figure 6: Log output of a complete crash after all node crash inside a node group

After such a crash, a format and restore need to be done on all NDB nodes. It is important to keep in mind that it is also mandatory to have a consistent cluster to restart MySQL Cluster, it means having at least one node per node group.

The results of these tests on NDB nodes provide us one of the MySQL Cluster best practices, separating each NDB node in a same node group in different data centers. In case of disaster on one site the cluster will still be consistent and therefore up and running.

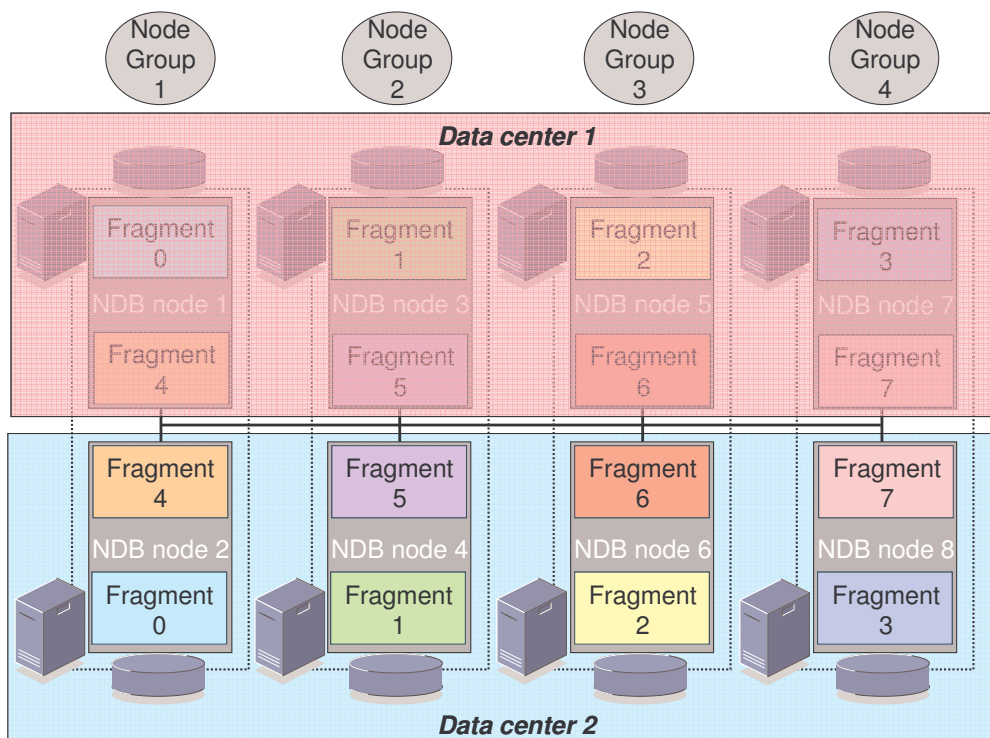


Figure 7: How the fragment and data nodes should be spread out in different data center

During the last sections we saw the consequences of loosing one or many NDB nodes in the same or different node groups. As you know a normal MySQL Cluster configuration is also made



of two other kinds of nodes: Management node and MySQL Server nodes. Therefore we also tested the impact of a crash on these kinds of nodes.

As we said in the introduction, the management server is used for both configuration and administration purpose. Many management nodes can be set up however only one will be active at time, it's a failover configuration. In case of crash of one or all management nodes the cluster will still be up and running, because this node is not needed for normal operations. However when we have to reconnect any kind of node, when we perform a backup, a shutdown or any administration or configuration task, it's needed to have at least one management node. When multiple management nodes are configured it is important to spread them out on different data center.

The last kind of node we put under fire was the MySQL server node. If we have a look on the MySQL Cluster architecture (cf Figure 1) we can see that the normal way to access to the data is through the MySQL server nodes. From there we can easily imagine what happens if the MySQL Server which provides access to the data to MySQL clients, applications or whatever is no more reachable: no more access to data!

From there we have three possibilities.

1. Parsing and changing all the connection strings into your application to go through another MySQL Server having access to the cluster. This approach could be quite time consuming...
2. Changing the impacted MySQL Server by a spare one with same IP and configuration.
3. Putting in place a high-availability solution, through the connection string, hardware or software solution or round-robin DNS.

In our tests we decided to implement a software high-availability solution through Heartbeat and MON which is really efficient.

Summary

MySQL Cluster ensures the high availability of the data not the high availability of the service. For the moment there is no MySQL solution to avoid the SPOF (Single Point of Failure) of the MySQL Server excepting direct access to NDB nodes. We have to use external components to ensure the service availability. Such components add complexity and more complexity could mean less availability. As far as you don't have a crash of all nodes in a same node group, the cluster availability is not impacted, only the running statements will fail. In case of NDB crash you can easily restart the impacted node without affecting the current activities. In addition MySQL Cluster provides an easy way to backup and restore data.

MySQL Cluster Replication could also be an interesting alternative (only since 5.1), see MySQL documentation chapter new features.

To some up, to the question does MySQL Rock? We can answer yes definitely!

Conclusion

MySQL Cluster provides nice features and rocks in many situations, for instance when a NDB node or management node(s) crash. There are efficient and quiet simple restore/recovery possibilities. MySQL Cluster 5.1 provides also interesting new features such as Cluster replication, disk storage and a more efficient space usage.



However disk storage needs to be improved, for the moment it's quite slow and space consuming.

MySQL Cluster doesn't seem to be a scalable solution. Adding new data nodes doesn't improve the performances, it's the opposite. For the moment Innodb appears still faster than NDB.

However MySQL Cluster could be used as a "low-cost" Highly Available data storage solution, compared to EMC solutions for instance.

Trivadis supported several customers for MySQL Cluster deployment. As for any High Available project (Oracle Dataguard or RAC, Veritas Storage Foundation, ...), the complexity of a cluster architecture shouldn't be underestimated. Several steps are very important in such a project: concept, documentation, testing/validating, and so on ...

Thanks to Yann Neuhaus, Thomas Jegen and Peter Welker without them this article wouldn't have been possible.

Gregory Steulet

Trivadis SA

Rue Marterey 5

CH-1005 Lausanne

Internet: www.trivadis.com

Tel: +41-21 321 47 00

Fax: +41-21 321 47 01

Mail: info@trivadis.com

Links and documentation

Trivadis - www.trivadis.com

MON - <http://sourceforge.net/projects/mon>

Heartbeat - <http://www.linux-ha.org>

sysbench - <http://sysbench.sourceforge.net>

MySQL Documentation – <http://dev.mysql.com/doc/refman/5.1/en/>

MySQL Clustering – MySQLPress - Alex Davies and Harrison Fisk