MySQL cluster: un database ad alta affidabilità – parte 2

In questo secondo articolo cercheremo di capire come si installa e come si configura il cluster MySQL sulla base dei concetti esposti nel precedente, che ne è l'ideale introduzione

di Raoul Scarazzini

Nel preparare l'ambiente per il primo test sul cluster MySQL, dobbiamo tenere in considerazione i discorsi affrontati nello scorso articolo sull'argomento Single Point Of Failure (o SPOF) e soprattutto, su come questi vadano evitati cercando di ridondare le risorse critiche il cui malfunzionamento potrebbe compromettere la stabilità del sistema.

È quindi impensabile, seppur possibile, effettuare dei test su una singola macchina. Il numero minimo di macchine su cui si dovrà lavorare sarà tre. Due nodi di storage ed uno di management.

Sui nodi storage faremo in modo di configurare delle repliche incrociate dei dati per testare, in caso di perdita di un nodo, come il sistema rimanga accessibile. Sul terzo nodo configureremo, oltre al demone per il management al quale i nodi di storage si collegheranno per ricevere le impostazioni di configurazione, anche un demone MySQL server in ascolto per poter manipolare i dati e permettere ad eventuali altre applicazioni di connettersi al cluster.

La struttura del nostro progetto è riassunta in figura1

Nella figura, fra parentesi viene indicato il nome host. Tale nome verrà indicato nelle configurazioni che effettueremo. Se però non si dispone di un DNS, indicare l'indirizzo IP sarà comunque corretto.

Una utile scappatoia potrebbe essere quella di aggiungere al file /etc/hosts presente in ogni macchina, le righe relative a ciascuna macchina facente parte del nostro cluster, in questa forma:

```
127.0.0.1    localhost.localdomain       localhost
192.168.0.1  management.mycluster.local  management
192.168.0.2  storage_a.mycluster.local   storage_a
192.168.0.3  storage_b.mycluster.local   storage_b
```

Certo il progetto non rispetta in pieno le rigide regole anti-SPOF elencate nello scorso articolo, ma trattandosi di un ambiente di test e non di produzione, risulta più che sufficiente.

Nel momento in cui si vorrà portare in produzione il progetto, allora ci si dovrà preoccupare di aumentare il numero di nodi di storage (sempre in numero pari), dividere management e MySQL server e ridondare ciascuno.

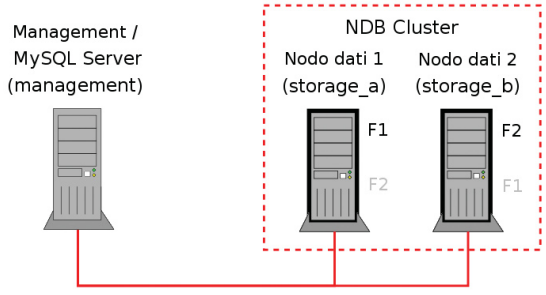

Figura 1 - La struttura del cluster con due nodi storage e due repliche. Ogni frammento è replicato su ciascun nodo

Scegliere il corretto software

Abbiamo già spiegato nel precedente articolo come i binari di MySQL che si trovano nelle comuni distribuzioni non abbiano le funzionalità HA. Per ottenere dei binari che soddisfino queste esigenze esistono due metodi : il primo sta nel ricompilare con apposite opzioni i sorgenti, il secondo nell'utilizzare il pacchetto binario mysql-max.

Salvo esigenze specifiche, utilizzare il pacchetto mysql-max-5.0.12-beta-linux-i686.tar.gz scaricabile da questo indirizzo http://dev.mysql.com/downloads/mysql/5.0.html risulta una "scelta vincente", mentre per sistemi con architetture differenti come ia64 o amd64, esistono le versioni di mysql-max relative.

Il link indicato porta non a caso alla versione 5.0 del pacchetto, che viene indicata come BETA. Infatti, anche se al momento la versione stabile è la 4.1, nel corso dell'articolo ci concentreremo sulla 5.0. Per prima cosa perché non manca

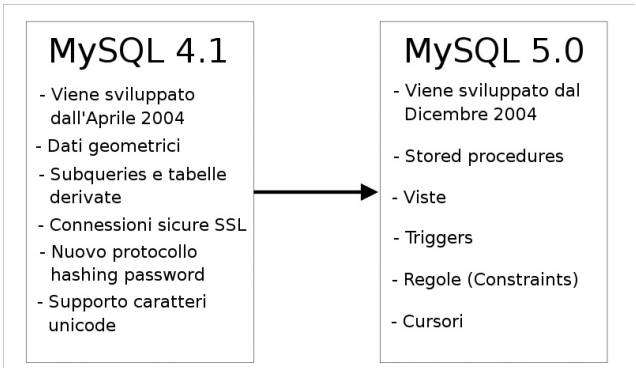

Figura 2 - Le funzionalità aggiuntive presenti nella versione 5.0 di MySQL rispetto alla 4.1

molto affinché diventi la release ufficiale ad essere distribuita ed inoltre presenta notevoli miglioramenti che vale la pena vedere in funzione.

Un elenco delle funzionalità aggiuntive presenti nella versione 5.0 lo si osserva in figura2.

Sempre all'indirizzo indicato per i download esiste la possibilità di scaricare anche i pacchetti rpm, convertibili senza problemi per le distribuzioni Debian in pacchetti deb tramite il software alien.

In tutto i pacchetti di binari sono quattro : MySQL-ndb-management, che contiene gli eseguibili per il nodo di management, MySQL-ndb-storage, contenente quelli per i nodi storage ed infine i pacchetti MySQL-ndb-tools e MySQL-ndb-extra contenenti programmi di utilità.

Esiste anche la possibilità di ricompilare manualmente i sorgenti scaricandosi il pacchetto .tar.gz ed indicando in fase di configurazione l'opzione --with-ndb-cluster. Tale operazione è però sconsigliata dagli stessi sviluppatori per ragioni di ottimizzazione e stabilità.

Nel corso dell'articolo utilizzeremo il pacchetto di binari tar.gz su tutte le macchine del cluster, poiché contiene tutti gli eseguibili necessari per far funzionare i nodi di storage, quello di management ed il server MySQL oltre a numerosi programmi di utilità e parecchi esempi.

Nulla vieta, una volta compresi i ruoli dei vari programmi, di utilizzare i pacchetti rpm per le installazioni in modo da ridurre il carico di spazio occupato dall'installazione.

Installazione

Utilizzando come stabilito i binari precompilati nel pacchetto .tar.gz, l'installazione è quanto di più semplice possa esservi. Basterà, su ciascuna macchina, scompattare il file in una directory stabilita, ad esempio /usr/local.

Essendo il nome della neo creata directory piuttosto lungo, converrà creare anche un link simbolico che ne faciliti l'accesso:

```
# cd /usr/local/
# tar -xzvf /<directory del file>/
  mysql-max-5.0.12-beta-linux-i686.tar.gz
# ln -s  mysql-max-5.0.12-beta-linux-i686 mysql
```

All'interno di questa directory si trova tutto il necessario per la costruzione del cluster.

Sarà utile creare, sempre su ciascuna macchina, anche una cartella /mysql sotto la quale verranno posti i file di configurazione ed i dati stessi.

Infine, per non dover lanciare i comandi ricorrendo tutte le volte ai path completi, possiamo aggiungere la directory dei binari /usr/local/mysql/bin/ alla variabile d'ambiente $PATH, inserendo questa riga all'interno del file /etc/profile:

```
export PATH=$PATH:/usr/local/mysql/bin/
```

In questo modo tutti gli eseguibili del pacchetto mysql-max saranno disponibili senza che sia necessario indicarne tutto il path.

Configurare il management server

La configurazione del server di Management risiede in un singolo file, denominato config.ini.

Tale file andrà creato nella directory /mysql del management server con questo contenuto:

```
[NDB_MGMD]
id=1
HostName=management

[NDBD DEFAULT]
NoOfReplicas=2
DataMemory=1950M
IndexMemory=450M

[NDBD]
id=2
HostName=storage_a
DataDir=/mysql/

[NDBD]
id=3
HostName=storage_b
DataDir=/mysql/

[MYSQLD]
```

Come si può notare, il file è diviso in sezioni, ciascuna delle quali inizia con una direttiva tra parentesi quadre. Ogni sezione termina quando inizia l'altra.

Prima di analizzare nel dettaglio ogni riga inserita, va detto che la configurazione indicata risulta ridotta, nel senso che esistono moltissime direttive per esigenze specifiche e sono tutte documentate da MySQL a questo indirizzo: http://dev.mysql.com/doc/mysql/en/mysql-cluster-config-file.html.

[NDB_MGMD]: Contiene le impostazioni relative al management server, nel nostro caso abbiamo dichiarato che tale nodo avrà identificativo 1 e che l'hostname relativo sarà "management". Ancora una volta va sottolineato che tale nome deve essere risolto dal DNS, altrimenti il cluster non potrà funzionare. In alternativa, si può inserire l'indirizzo IP.
[NDBD_DEFAULT]: Contiene le impostazioni comuni a tutti i nodi dati (NDBD). In questo caso sono stati indicati il numero di repliche (due secondo la struttura illustrata in figura1) e la quantità riservata alla memoria dati e indici.

Tale cifra è ottenibile con questa operazione

```
DataMemory = Node RAM * Total Nodes / Replicas
```

La quantità di memoria degli indici va stabilita in base alle

esigenze, un buon compromesso può essere circa il 20% della DataMemory.

I valori indicati nell'esempio sopra, sono quindi del tutto indicativi, poiché dipendono dalle macchine utilizzate per effettuare i test.

[NDBD]: Ciascuna di queste sezioni contiene le impostazioni del singolo nodo dati, per il quale vengono indicati l'identificativo numerico, il nome host (o l'indirizzo IP) e la directory nella quale verranno memorizzati i dati.

Le sezioni NDBD dovranno essere tante quante i nodi dati.

[MYSQLD]: Ciascuna di queste sezioni contiene le impostazioni dei singoli nodi sql. Per nodi sql si intendono server mysql, applicazioni che si basano sulle MySQL cluster API (di cui abbiamo parlato nello scorso articolo) o le stesse applicazioni della suite di mysql-max.

In questo caso è stato previsto quindi un solo client collegabile.

La configurazione del nodo di management termina qui, il file config.ini dovrà essere letto dal file eseguibile ndb_mgmd. Questo file eseguibile agirà da demone e rimarrà in ascolto per le connessioni dei nodi dati.

Configurare i nodi storage

Tutte le impostazioni dei nodi dati risiedono sul management server ed ognuno di essi prima di fare qualsiasi cosa vi si connette per capire come è composto il cluster e quale ruolo deve svolgere all'interno di questo.

Anche in questo caso esiste un file eseguibile che agisce da demone e che consente di effettuare il collegamento, "scaricare" la configurazione dal management server e comunicare con gli altri nodi dati per creare l'NDB Cluster, questo file è ndbd.

L'unica cosa da configurare è l'accesso al server di management, e ciò si ottiene creando un file denominato /etc/my.cnf che conterrà le impostazioni di configurazione lette da ndbd.

È comunque possibile passare all'eseguibile i dati di connessione direttamente a riga di comando.

Il file /etc/my.cnf conterrà quanto segue:

```
[MYSQL_CLUSTER]
ndbcluster
ndb-connectstring=management.mycluster.local
```

Chiaramente quanto scritto andrà ripetuto anche sul secondo nodo dati storage_b.

La configurazione dei nodi storage è così completata.

Configurare il server MySQL

L'ultima parte della configurazione riguarda il MySQL server che nel nostro caso abbiamo deciso di configurare sulla stessa macchina del management (ma che potrebbe risiedere su di una a se stante).

Per prima cosa, è necessario creare l'utenza mysql (gruppo mysql) con cui verrà eseguito il server:

```
# groupadd mysql
# useradd -g mysql mysql
```

Fatto questo, si può procedere all'installazione dei database di sistema, utilizzando il file mysql_install_db contenuto nella cartella scritps/ che si trova nell'alberatura creata dell'installazione del file tar.gz descritta poco sopra:

```
# cd /usr/local/mysql/
# scripts/mysql_install_db --user=mysql
```

Chiaramente si è forzata l'utenza mysql per l'esecuzione delle operazioni per ovvie ragioni di sicurezza.

Una volta completata questa operazione, non rimane che settare i permessi delle directory relative:

```
# chown -R root:mysql .
# chown -R mysql data/
```

Come ultima cosa, è necessario creare un file denominato my.cnf ad esempio sotto la cartella /mysql contenente, oltre ad alcune opzioni per il server stesso, le impostazioni di connessione al cluster:

```
[MYSQLD]
user=mysql
port=3306
socket=/tmp/mysql.sock
datadir=/usr/local/mysql/data
basedir=/usr/local/mysql

[MYSQL_CLUSTER]
ndbcluster
ndb-connectstring=management.mycluster.local
```

Le opzioni relative al server MySQL rispecchiano la nostra installazione : l'utente con cui verrà eseguito il server sarà "mysql", la porta su cui sarà in ascolto sarà quella standard "3306", il file di socket per le connessioni sarà /tmp/mysql.sock, le directory dei dati e del server saranno rispettivamente /usr/local/mysql/data e /usr/local/mysql.

Le altre opzioni presenti nell'area [MYSQL_CLUSTER] sono le stesse dei due nodi storage. Si riferiscono infatti al server di management.

La configurazione del MySQL server è quindi completa.

Avviare il nodo di management

La sequenza in cui le varie componenti del cluster verranno andranno sarà la seguente : server di management, nodi storage ed infine server MySQL.

Come osservato in fase di configurazione, il programma demone che rimarrà in ascolto come management server è ndb_mgmd, questo significa che sulla macchina management.mycluster.local andrà eseguito il seguente comando:

```
# ndb_mgmd --config-file=/mysql/config.ini
```

Il parametro passato è ovviamente il file di configurazione da noi creato.

La porta TCP su cui il demone rimane in ascolto è la 1186, per verificare quindi che questa sia aperta è possibile lanciare il seguente comando:

```
# netstat -natp | grep ndb
tcp        0      0 0.0.0.0:1186
0.0.0.0:*               LISTEN      10303/ndb_mgmd
tcp        0      0 192.168.0.1:35563
192.168.0.1:1186          ESTABLISHED10303/ndb_mgmd
tcp        0      0 192.168.0.1:1186
192.168.0.1:35565         ESTABLISHED10303/ndb_mgmd
tcp        0      0 192.168.0.1:1186
192.168.0.1:35563         ESTABLISHED10303/ndb_mgmd
```

Un output simile a quello riportato, indica che il processo è in esecuzione ed in ascolto.

Da questo punto in avanti, per gestire il cluster utilizzeremo il programma ndb_mgm:

```
# ndb_mgm
-- NDB Cluster -- Management Client --
ndb_mgm>
```

All'interno della shell ndb_mgm, è possibile osservare lo stato del cluster, avviare e stoppare i nodi, modificare i metodi di logging e così via. Una panoramica dei comandi disponibili si ottiene digitando help e premendo invio, per il momento, limitiamoci a digitare show e premere invio:

```
ndb_mgm> show
Connected to Management Server at:
management.mycluster.local:1186
Cluster Configuration
---------------------
[ndbd(NDB)]     2 node(s)
id=2 (not connected, accepting connect from storage_a)
id=3 (not connected, accepting connect from storage_b)

[ndb_mgmd(MGM)] 1 node(s)
id=1    @192.168.0.1  (Version: 5.0.12)

[mysqld(API)]   1 node(s)
id=4 (not connected, accepting connect from any host)
```

Si può osservare come sia stata visualizzata tutta la configurazione del cluster : Il management server ndb_mgmd(MGM) è in attesa di connessioni da parte dei due nodi dati ndbd(NDB) (ciascuno dei quali associato alla rispettiva macchina, storage a o b) e di un server MySQL mysqld(API).

Avviare i nodi dati

Per avviare i nodi dati che costituiranno l'NDB Cluster, dobbiamo recarci su ciascuna delle macchine storage ed avviare il programma ndbd:

```
# ndbd
```

Per verificare che il processo sia in esecuzione, possiamo filtrare l'output del comando ps come segue:

```
# ps aux | grep ndbd
root      3845  0.0  0.0  6476 2872 ?        S
12:41   0:00 ndbd
root      3846  3.5  6.1 1930968 251212 ?    S
12:41   0:00 ndbd
root      3878  0.0  0.0  3692  668 pts/0    S
12:41   0:00 grep ndbd
```

Come si noterà dall'output, pur avendo lanciato solo una volta il comando, nell'elenco dei processi si trovano due processi ndbd. Non c'è nulla di sbagliato in questo, infatti il demone ndbd prevede che esista un processo base che si occupi dei dati ed un processo "angelo custode" che consenta al nodo dati di essere gestito da remoto.

Il concetto si può capire pensando alle operazioni di restart di un nodo che prevedono il kill del processo ndbd ed il suo nuovo avvio. Se dal management server si volesse effettuare un'operazione come quella descritta ed esistesse un unico processo ndbd sul nodo dati, sarebbe impossibile farlo ripartire una volta stoppato. Vi è quindi la necessità che a guidare le operazioni del processo ndbd principale vi sia un angelo custode.

A questo punto se ritorniamo sul management server e sempre dalla shell ndb_mgm lanciamo il comando show, otterremo quanto segue:

```
# ndb_mgm
-- NDB Cluster -- Management Client --
ndb_mgm> show
Connected to Management Server at:
management.mycluster.local:1186
Cluster Configuration
---------------------
[ndbd(NDB)]     2 node(s)
id=2    @192.168.0.2  (Version: 5.0.12, Nodegroup: 0,
Master)
id=3    @192.168.0.3  (Version: 5.0.12, Nodegroup: 0)

[ndb_mgmd(MGM)] 1 node(s)
id=1    @192.18.0.1 (Version: 5.0.12)

[mysqld(API)]   1 node(s)
id=4 (not connected, accepting connect from any host)
```

Quanto visualizzato conferma che i due nodi dati sono cor-

rettamente avviati, appartengono ad un unico Nodegroup e che il nodo nodo con id 2 è Master.

Fra tutte le opzioni passabili da riga di comando all'eseguibile ndbd, due in particolare meritano una spiegazione "-n" e "--initial".

"-n" o "--nostart" sta per "Don't start ndbd immediately", questo significa che il nodo dati diverrà attivo solo quando riceverà il comando start dal management node. Il nodo dati sarà quindi in stato "not started". Supponiamo di aver avviato a questo punto il nodo dati 3 con il comando "ndbd -n". Un comando show dalla console di ndb_mgm produrrà quindi tra le altre questa riga:

```
[...]
id=3    @192.168.0.3 (Version: 5.0.12, not started)
[...]
```

Il nodo dati 3 rimarrà in stato "not started" sino a che dalla shell di ndb_mgm non verrà forzato lo start del nodo :

```
ndb_mgm> 3 start
Database node 3 is being started.
```

Fatto questo, il nodo dati 3 sarà "online":

```
[...]
id=3    @192.168.0.3 (Version: 5.0.12, Nodegroup: 0)
[...]
```

L'altra importante opzione passabile ad ndbd è "--initial". Passare questa opzione significa forzare la pulizia del filesystem relativo al nodo dati. Se l'initial start è effettuato su di un solo nodo, allora i dati verranno ricostruiti in base alle repliche presenti sull'altro, mentre se l'operazione viene effettuata su tutti i nodi, allora l'intera base dati viene ripulita.

Questa opzione torna utile quando vengono effettuate delle modifiche strutturali al cluster come ad esempio la modifica della memoria disponibile.

Vedremo nel prossimo articolo come sfruttare questo parametro.

Avviare il mysqld

Per avviare il mysqld basterà lanciare il comando mysqld a cui andrà passato il percorso del file di configurazione creato poco sopra tramite l'opzione "--defaults-file=":

```
# mysqld --defaults-file=/mysql/my.cnf &
```

Il carattere & alla fine del comando consente di mettere il processo in background per poter continuare a lavorare nella console. È possibile inoltre verificare che il processo mysqld sia in ascolto sulla porta configurata:

```
# netstat -natp | grep mysqld
tcp        0      0 0.0.0.0:3306
0.0.0.0:*                   LISTEN
15129/mysqld
tcp        0      0 192.168.0.1:34398
192.168.0.1:1186            ESTABLISHED
15129/mysqld
tcp        0      0 192.168.0.1:34402
192.168.0.2:32782           ESTABLISHED 15129/mysqld
tcp        0      0 192.168.0.1:34401
192.168.0.3:32779           ESTABLISHED 15129/mysqld
```

A questo punto l'avvio del cluster può dirsi completato, infatti ndb_mgm conferma che tutte le componenti sono operative:

```
# ndb_mgm
-- NDB Cluster -- Management Client --
ndb_mgm> show
Connected to Management Server at:
management.mycluster.local:1186
Cluster Configuration
---------------------
[ndbd(NDB)]     2 node(s)
id=2    @192.168.0.2  (Version: 5.0.12, Nodegroup: 0)
id=3    @192.168.0.3  (Version: 5.0.12, Nodegroup: 0,
Master)

[ndb_mgmd(MGM)] 1 node(s)
id=1    @192.168.0.1  (Version: 5.0.12)

[mysqld(API)]   1 node(s)
id=4    @192.168.0.1  (Version: 5.0.12)
```

Utilizzo del motore NDB

Per effettuare dei test di funzionamento del motore cluster, utilizzeremo il programma client MySQL dalla macchina di management:

```
# /usr/local/mysql/bin/mysql
Welcome to the MySQL monitor.  Commands end with ;
or \g.
Your MySQL connection id is 2 to server version:
5.0.12-beta-max

Type 'help;' or '\h' for help. Type '\c' to clear
the buffer.

mysql>
```

Per prima cosa verificheremo che nell'elenco dei motori MySQL disponibili sia presente anche quello del cluster NDB, con questo comando:

```
mysql> show engines;
+------------+---------+--------------------------+
| Engine     | Support | Comment                  |
+------------+---------+--------------------------+
```

```
...
|
| NDBCLUSTER | YES     | Clustered, fault-tolerant,
                          memory-based tables     |
| NDB        | YES     | Alias for NDBCLUSTER    |
...
+------------+---------+--------------------------+
18 rows in set (0.00 sec)
```

Se tra le righe risultanti appariranno anche quelle relative ad NDBCLUSTER, avremo la conferma del supporto al motore cluster da parte del server MySQL installato sulla macchina di management.

Occorre però un chiarimento. Tale server possiede le medesime funzionalità di un comune server MySQL ed anzi, se non si sfruttano direttive particolari in fase di creazione delle tabelle, queste verranno gestite dal motore di default, che si chiama MyISAM, e non dal motore NDB.

Per capire meglio questo concetto, procediamo col creare un database di test:

```
mysql> CREATE DATABASE ndb_test;
Query OK, 1 row affected (0.00 sec)
```

All'interno creeremo una semplice tabella da un campo di tipo integer contenente un singolo valore:

```
mysql> USE ndb_test;
Database changed
mysql> CREATE TABLE test (i INT);
Query OK, 0 rows affected (0.02 sec)

mysql> INSERT INTO test (i) VALUES (1);
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM test;
+------+
| i    |
+------+
|    1 |
+------+
1 row in set (0.00 sec)
```

Al momento la tabella NON è ancora gestita dal motore NDB, e ciò è confermato da questo comando:

```
mysql> SHOW CREATE TABLE test;
+-------+-----------------------------------------+
| Table | Create Table                            |
+-------+-----------------------------------------+
| test  | CREATE TABLE `test` (
  `i` int(11) default NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1           |
+-------+-----------------------------------------+
1 row in set (0.00 sec)
```

ENGINE=MyISAM indica infatti che la tabella viene gestita localmente dal motore MyISAM.

Per fare in modo di porre sotto motore NDB la tabella, è necessario forzare il motore da usare in fase di creazione, aggiungendo alla fine del comando CREATE TABLE la clausola ENGINE=NDBCLUSTER:

```
mysql> CREATE TABLE test (i INT) ENGINE=NDBCLUSTER;
```

oppure modificare la tabella appena creata come segue:

```
mysql> ALTER TABLE test ENGINE=NDBCLUSTER;
Query OK, 1 row affected (0.44 sec)
Records: 1  Duplicates: 0  Warnings: 0
```

A questo punto i dati della tabella (una sola, importantissima riga!) sono replicati sui nodi del cluster e sono raggiungibili attraverso il server mysqld da qualsiasi applicazione:

```
mysql> SHOW CREATE TABLE test;
+-------+-----------------------------------------+
| Table | Create Table                            |
+-------+-----------------------------------------+
| test  | CREATE TABLE `test` (
  `i` int(11) default NULL
) ENGINE=ndbcluster DEFAULT CHARSET=latin1       |
+-------+-----------------------------------------+
1 row in set (0.05 sec)
```

Un ultimo importante chiarimento riguarda eventuali altri server MySQL che potranno essere collegati al cluster NDB. Infatti il motore NDB agisce a livello di tabelle, non di database. Questo significa che non è il database ad essere replicato, ma solo le tabelle. Su ciascun ulteriore server MySQL collegato dovranno quindi esistere i database relativi. Nel nostro caso ad esempio, se volessimo accedere alla tabella test da un'altro server MySQL configurato nel management server, dovremmo prima creare il database ndb_test al cui interno, senza fare nient'altro, troveremo le tabelle del cluster.

Conclusioni

Abbiamo visto in questo articolo come configurare ed avviare il cluster MySQL creando al suo interno una tabella di test.

Nel prossimo impareremo a consultare i log, effettueremo dei test di funzionamento simulando dei crash sui singoli nodi per accertarci che non esistano SPOF ed effettueremo modifiche sulla struttura del database. Infine impareremo ad effettuare backup e restore dei dati in modo da poter approfondire tutte le funzionalità di questo interessante progetto.

Raoul Scarazzini - rascasoft@tiscali.it - http://web.tiscali.it/rascasoft - Raoul Scarazzini è responsabile del settore Linux presso Cutaway SAS, una società di consulenza IT a Milano