# MySQL cluster: un database ad alta affidabilità – parte 3

In questo terzo ed ultimo articolo completiamo il discorso sul cluster MySQL imparando a consultare i log, a capire le fasi di avvio ed i meccanismi di registrazione. Infine effettueremo il backup ed il restore delle basi dati registrate.

**di Raoul Scarazzini**

Prima di effettuare qualsiasi tipo di operazione per comprendere la gestione dei log, è necessario avviare il cluster.

Partendo dal presupposto che tutti i file sono configurati come illustrato negli scorsi numeri, dal management server è necessario lanciare il comando:

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

E su ciascuno dei nodi dati:

```
# ndbd -n
```

L'opzione "-n", come già spiegato, inizializza i nodi dati senza avviarli. L'avvio effettivo dei nodi dati andrà forzato attraverso la shell ndb_mgm e consentirà di controllare attraverso i log le fasi di avvio. Sempre attraverso questa shell, è possibile verificare lo stato attuale del cluster:

```
# 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, not started)
id=3    @192.168.0.3  (Version: 5.0.12, not started)

[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)
```

Quanto visualizzato, conferma che stiamo operando su di un cluster che comprende due nodi dati (in stato "not started"), un nodo di management ed un nodo client, al momento non connesso.

I nodi dati ed il nodo di management registrano i log nella cartella che è stata dichiarata nella sezione DataDir all'interno del file config.ini del management server.

Nel nostro caso, la DataDir dei nostri nodi è /mysql/. Al di sotto di questa, su ciascun nodo, si trovano file denominati ndb_ID_out.log, dove "ID" rappresenta l'identificativo del nodo, quindi ad esempio, il nodo di management loggerà all'interno del file ndb_1_out.log.

Visualizzando il contenuto di questi file, ci si può fare una chiara idea di ciò che i vari nodi stanno facendo.

Partendo dalla macchina di management, il contenuto del file sarà questo:

```
# tail /mysql/ndb_1_out.log
NDB Cluster Management Server. Version 5.0.12 (beta)
Id: 1, Command port: 1186
```

Questo conferma che il management è stato avviato ed è in ascolto sulla porta 1186.

Sul nodo due, il file conterrà:

```
# tail /mysql/ndb_2_out.log
2005-10-28 10:10:54 [NDB] INFO     -- Angel pid:
11026 ndb pid: 11027
2005-10-28 10:10:54 [NDB] INFO     -- NDB Cluster -
- DB node 2
2005-10-28 10:10:54 [NDB] INFO     -- Version
5.0.12 (beta) --
2005-10-28 10:10:54 [NDB] INFO     -- Configuration
fetched at management.mycluster.local port 1186
```

Mentre nel nodo tre, il file avrà questo contenuto:

```
# tail /mysql/ndb_3_out.log
2005-10-28 10:10:59 [NDB] INFO     -- Angel pid:
11061 ndb pid: 11062
2005-10-28 10:10:59 [NDB] INFO     -- NDB Cluster -
- DB node 3
2005-10-28 10:10:59 [NDB] INFO     -- Version
5.0.12 (beta) --
2005-10-28 10:10:59 [NDB] INFO     -- Configuration
fetched at management.mycluster.local port 1186
```

In entrambi i casi, abbiamo la conferma che i nodi dati si sono avviati nella maniera corretta e sono collegati al management server tramite la porta 1186.

Da notare come vengano indicati anche i pid (ossia i numeri identificativi) dei DUE processi relativi al nodo dati : quello angelo, che consente il controllo dei processi dalla console remota ndb_mgm, e quello effettivo, che si occupa della gestione dell'interscambio dati.

Le informazioni relative allo stato generale del cluster sono memorizzate sul management server nel file denominato ndb_1_cluster.log. In questo file vengono registrati tutti gli eventi che riguardano il cluster. Appena avviato ed una volta effettuate le connessioni da parte dei nodi dati e client, il suo contenuto dovrebbe essere simile al seguente:

```
# tail /mysql/ndb_1_cluster.log
2005-10-28 10:10:47 [MgmSrvr] INFO     -- NDB
Cluster Management Server. Version 5.0.12 (beta)
2005-10-28 10:10:47 [MgmSrvr] INFO     -- Id: 1,
Command port: 1186
2005-10-28 10:10:54 [MgmSrvr] INFO     -- Mgmt
server state: nodeid 2 reserved for ip
192.168.0.2, m_reserved_nodes 0000000000000006.
2005-10-28 10:10:54 [MgmSrvr] INFO     -- Node 1:
Node 2 Connected
2005-10-28 10:10:55 [MgmSrvr] INFO     -- Mgmt
server state: nodeid 2 freed, m_reserved_nodes
0000000000000002.
2005-10-28 10:10:59 [MgmSrvr] INFO     -- Mgmt
server state: nodeid 3 reserved for ip
192.168.0.3, m_reserved_nodes 000000000000000a.
2005-10-28 10:10:59 [MgmSrvr] INFO     -- Node 1:
Node 3 Connected
2005-10-28 10:11:00 [MgmSrvr] INFO     -- Mgmt
server state: nodeid 3 freed, m_reserved_nodes
0000000000000002.
```

I nodi 2 e 3 sono stati inizializzati e tramite la connessione al management node, hanno ottenuto il proprio "Node ID" e la configurazione del cluster. Questa fase preliminare dell'avvio dei nodi prevede l'allocazione delle porte che verranno usate per la comunicazione inter-nodo ed infine l'allocazione della memoria in relazione a quanto dichiarato nella configurazione.

### Log e Checkpoints

Per studiare le fasi di avvio del cluster attraverso i Log, è necessario comprendere i meccanismi con cui i dati vengono registrati su disco.

MySQL Cluster è un database distribuito su più nodi e le informazioni in esso contenute sono replicate in maniera sincrona. Questo significa che un'operazione di aggiornamento, definita "transazione", prima di essere registrata nel database, cioè prima di raggiungere lo stato "commited", deve essere effettuata sulla replica primaria e su ciascuna delle secondarie.

Anche se il database risiede completamente nella memoria del cluster, è ovvio che ciascun nodo possiede un filesystem nel quale vengono fisicamente memorizzati i dati. Sarebbe altrimenti impossibile recuperare i dati una volta spente le macchine, visto che il contenuto della RAM è svuotato ad ogni avvio del computer. Questo filesystem si trova su ogni nodo nella directory dichiarata come DataDir nel file di configurazione. Ad esempio, per il nodo con id 2, il filesystem verrà memorizzato all'interno della directory /mysql/ndb_2_fs.

Ciascun nodo, tiene traccia delle transazioni all'interno del "REDO log". La funzione del REDO log è quella di fornire un backup in caso l'intero cluster cada. I log sono controllati in maniera centralizzata, ma vengono effettuati localmente, sul filesystem di ciascun nodo.

Il REDO log è registrato in memoria e ad intervalli regolari (di default ogni 2 secondi) scritto su disco. Ogni volta che il REDO log viene scritto, viene effettuato un "Global Checkpoint" o "GCP".

La frequenza con cui i GCP vengono eseguiti può essere variata all'interno del file di configurazione del management server, indicando per l'opzione TimeBetweenGlobalCheckpoints un valore in millisecondi fra 10 e 32000.

Per alleggerire il carico di informazioni registrate nei REDO log (che è un file incrementale e che quindi continua ad ingrandirsi) e consentire una rapida ricostruzione in caso di totale caduta del cluster, MySQL effettua, sempre ad intervalli regolari, dei "Local Checkpoints" o "LCP".

Durante un LCP viene creata una copia su disco, definita "Snapshot", di tutto il database e viene alleggerito il Redo Log al quale viene rimossa la coda delle operazioni svolte sino a quel momento.

Un LCP impiega parecchio tempo a completarsi e viene effettuato mentre il database viene aggiornato. Quanto viene scritto quindi, è da considerarsi inconsistente, in quanto potrebbero esserci delle transazioni non concluse, delle tabelle su cui persiste un lock e così via. Per questo motivo, al momento dell'avvio di un LCP, viene creato un UNDO log, ossia un registro delle operazioni effettuate dal database durante il LCP.

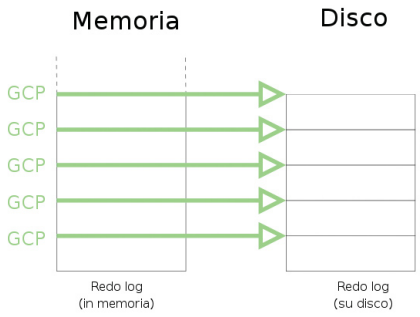


Figura 1 - ad intervalli regolari, definiti Global Checkpoints (GCP) il REDO log viene scritto sul disco

L'UNDO log consente di rendere lo snapshot del database, una volta terminato il LCP, di nuovo consistente.

Anche in questo caso la frequenza degli LCP può variare in funzione alle esigenze, settando nel file di configurazione l'opzione TimeBetweenLocalCheckpoints.

Riassumendo, se un solo nodo per qualsiasi ragione cade, si ricostruirà seguendo queste fasi:

**a)** Riconnessione al server di management e segnalazione della propria presenza agli altri membri;
**b)** Copia dei metadata, ossia tabelle, indici ed informazioni del cluster dal management server;
**c)** Copia dei dati dal nodo primario;
**d)** Riassunzione dello stato di nodo primario;

Se invece sarà tutto il sistema a cadere, allora il database verrà ricostruito partendo dall'ultimo LCP e dalle parti di REDO Log che sono state registrate su disco dall'ultimo GCP. Le transazioni registrate DOPO l'ultimo GCP verranno perse. Questa serie di operazioni è definita "System recovery".

### Le fasi di avvio del Cluster

A questo punto possiamo seguire le fasi di avvio del cluster direttamente dal file di log della macchina di management, lanciando il comando

```
tail -f /mysql/ndb_1_cluster.log
```

e forzando da una nuova shell della macchina di management l'avvio dei nodi dati tramite il programma ndb_mgm:

```
# ndb_mgm
-- NDB Cluster -- Management Client --
ndb_mgm> all start
Connected to Management Server at:
management.mycluster.local:1186
NDB Cluster is being started.
NDB Cluster is being started.
```

Nel file di ndb_1_cluster.log si potranno seguire le operazioni preliminari di avvio del cluster:

```
2005-10-28 11:06:59 [MgmSrvr] INFO     -- Node 2:
Start initiated (version 5.0.12)
2005-10-28 11:06:59 [MgmSrvr] INFO     -- Node 3:
Start initiated (version 5.0.12)
2005-10-28 11:06:59 [MgmSrvr] INFO     -- Node 2:
Start phase 0 completed
2005-10-28 11:06:59 [MgmSrvr] INFO     -- Node 2:
Communication to Node 3 opened
2005-10-28 11:06:59 [MgmSrvr] INFO     -- Node 3:
Start phase 0 completed
2005-10-28 11:06:59 [MgmSrvr] INFO     -- Node 3:
Communication to Node 2 opened
2005-10-28 11:06:59 [MgmSrvr] INFO     -- Node 2:
Node 3 Connected
2005-10-28 11:06:59 [MgmSrvr] INFO     -- Node 3:
Node 2 Connected
```

Da questo punto in poi, i nodi attraverseranno le varie fasi che porteranno all'avvio effettivo del cluster. Queste, definite stage, sono 12 e partono da 0:

**Stage 0**: Se il processo ndbd è stato avviato tramite un "initial start", cioè con l'opzione --initial (si veda l'articolo precedente), viene pulito il filesystem del nodo relativo.
**Stage 1**: È la prima vera fase di avvio in cui vengono stabilite le connessioni tra i nodi ed avviati gli heartbeat (letteralmente "battito cardiaco"), ossia i processi di controllo esistenza di ciascun nodo verso gli altri. Una comunicazione heartbeat prevede uno scambio di messaggi come "Io ci sono. Tu ci sei?" che consentono ad ogni nodo di decidere come agire in caso di problemi. Ad esempio, una mancata risposta ad un heartbeat da parte di un nodo primario imporrebbe al nodo contenente la replica secondaria di far diventare la sua copia di dati primaria.
**Stage 2**: In questa fase viene eletto il nodo bilanciatore, che

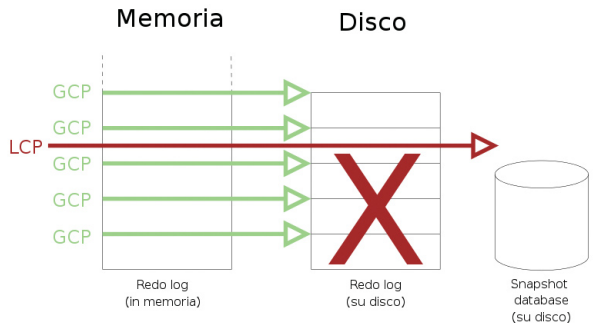


Figura 2 - durante il Local Checkpoint (LCP) viene scritta su disco un'immagine definita "Snapshot" e vengono eliminate dal Redo log le operazioni precedenti

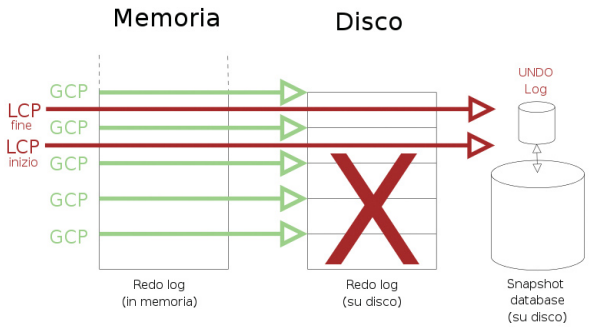


Figura 3 - quando il Local Checkpoint termina, viene utilizzato l'UNDO Log per rendere nuovamente consistente lo Snapshot registrato

può essere o il management node o uno dei nodi sql client. La funzione del nodo bilanciatore sarà quella di decidere come distribuire il carico delle richieste sui nodi dati.
**Stage 3**: Vengono inizializzate alcune variabili interne del cluster.
**Stage 4**: Se viene forzata la pulitura del filesystem relativo a tutto il cluster o al singolo nodo, viene creato il "REDO log".
**Stage 5**: In questa fase, un nodo che è stato riavviato (e deve quindi reinserirsi nel cluster) viene sincronizzato con il nodo master ed incluso nelle transazioni.
**Stage 6 e 7**: Viene effettuato l'aggiornamento di variabili interne.
**Stage 8**: Vengono ricostruiti tutti gli indici.
**Stage 9**: Viene effettuato l'aggiornamento di variabili interne.
**Stage 10**: Le applicazioni possono connettersi ai nodi ed incominciare ad effettuare interscambio dati.
**Stage 11**: Un nodo immesso all'interno del cluster assume il ruolo di replica primaria ed effettua l'invio della replica allo slave associato.

Il passaggio da una fase all'altra è segnalato all'interno dei log con il messaggio "Start phase <numero> completed":

```
...
2005-10-28 11:06:59 [MgmSrvr] INFO     -- Node 2:
Start phase 1 completed
...
```

Chiaramente, non avendo ancora collegato un client e non avendo effettuato alcun riavvio, al termine della fase 9 il nostro cluster è avviato:

```
...
2005-10-28 11:07:05 [MgmSrvr] INFO     -- Node 2:
Started (version 5.0.12)
2005-10-28 11:07:05 [MgmSrvr] INFO     -- Node 3:
Communication to Node 4 opened
2005-10-28 11:07:05 [MgmSrvr] INFO     -- Node 3:
Communication to Node 5 opened
2005-10-28 11:07:05 [MgmSrvr] INFO     -- Node 3:
Communication to Node 0 opened
2005-10-28 11:07:05 [MgmSrvr] INFO     -- Node 3:
Start phase 8 completed (initial start)
2005-10-28 11:07:05 [MgmSrvr] INFO     -- Node 3:
Start phase 9 completed (initial start)
2005-10-28 11:07:05 [MgmSrvr] INFO     -- Node 3:
Started (version 5.0.12)
2005-10-28 11:07:06 [MgmSrvr] INFO     -- Node 3:
Node 1: API version 5.0.12
2005-10-28 11:07:06 [MgmSrvr] INFO     -- Node 2:
Node 1: API version 5.0.12
2005-10-28 11:07:06 [MgmSrvr] INFO     -- Node 3:
Prepare arbitrator node 1 [ticket=399a0001367af1d4]
2005-10-28 11:07:06 [MgmSrvr] INFO     -- Node 2:
Started arbitrator node 1 [ticket=399a0001367af1d4]
```

È da notare come l'ultimo messaggio indichi chi sia l'arbitrator node (il bilanciatore del carico), cioè il management server.

A questo punto, per completare l'avvio del Cluster ed iniziare ad accedere alle tabelle, possiamo avviare il server mysqld che si collegherà al cluster

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

ed osservare quanto viene scritto nei log:

```
2005-10-28 15:57:26 [MgmSrvr] INFO     -- Mgmt ser-
ver state: nodeid 4 reserved for ip 192.168.0.4,
m_reserved_nodes 0000000000000012.
2005-10-28 15:57:27 [MgmSrvr] INFO     -- Node 3:
Node 4 Connected
2005-10-28 15:57:27 [MgmSrvr] INFO     -- Node 2:
Node 4 Connected
2005-10-28 15:57:27 [MgmSrvr] INFO     -- Node 2:
Node 4: API version 5.0.12
2005-10-28 15:57:27 [MgmSrvr] INFO     -- Node 3:
Node 4: API version 5.0.12
```

Il client mysqld si è collegato al server di management, ha scaricato la configurazione e si è collegato direttamente ai nodi dati del cluster con cui effettuerà interscambio dati.

Un'ultima nota riguardo ai log del cluster riguarda l'esecuzione dei Local Checkpoints che verrà segnalata all'interno del file /mysql/ndb_1_cluster.log sulla macchina di management con righe simili a quella che segue:

```
2005-10-28 11:07:05 [MgmSrvr] INFO     -- Node 2:
Local checkpoint 1 started. Keep GCI = 1 oldest
restorable GCI = 1
```

Questa riga indica che è stato avviato il LCP con identificativo 1 sul Nodo 2 e che l'ultimo GCP ripristinabile è quello con identificativo 1.

Ovviamente, mano a mano che passerà il tempo, altre indicazioni di questo tipo seguiranno:

```
2005-10-28 12:01:43 [MgmSrvr] INFO     -- Node 2:
Local checkpoint 2 started. Keep GCI = 1 oldest
restorable GCI = 2
2005-10-28 12:56:21 [MgmSrvr] INFO     -- Node 2:
Local checkpoint 3 started. Keep GCI = 1609 oldest
restorable GCI = 3
...
2005-11-01 04:16:15 [MgmSrvr] INFO     -- Node 2:
Local checkpoint 100 started. Keep GCI = 157530
oldest restorable GCI = 8890
```

```
2005-11-01 05:10:53 [MgmSrvr] INFO     -- Node 2:
Local checkpoint 101 started. Keep GCI = 159138
oldest restorable GCI = 8890
```

Come si può osservare, la frequenza tra un Local Checkpoint ed un altro è di circa un ora.

### Backup e Restore

Terminata la spiegazione relativa ai log, concludiamo il discorso relativo al Cluster MySQL con le operazioni di Backup e Restore.

### Backup

Effettuare backup periodici di una base dati è essenziale per svariate ragioni: riparare ad errori causati dalle applicazioni, effettuare un aggiornamento del cluster (che non è possibile eseguire in linea) e ripristinare l'intera base dati in seguito ad un totale crash del sistema.

I metodi di backup sono essenzialmente due: logico e fisico.
**Backup logici**: I backup logici riguardano l'esportazione della struttura logica del database in file sql. Tali backup possono essere effettuati attraverso gli strumenti canonici di mysql quali ad esempio il programma mysqldump. Questi si rivelano utili per il trasferimento dell'intera base dati ad un altro tipo di database, essendo scritti nel linguaggio sql standard, universalmente comprensibile. Un esempio di come effettuare un backup tramite mysqldump è il seguente:

```
# mysqldump  ndb_test > ndb_test_BACKUP.sql
```

Questo comando creerà un file denominato ndb_test_BACKUP.sql nella directory corrente che conterrà tutti gli statement sql necessari alla totale ricostruzione del database ndb_test che abbiamo creato nel precedente articolo.

Questo metodo di backup risulta comodo se si ha la necessità di esportare database dalle dimensioni ridotte o semplicemente singole tabelle, ma su basi dati dalle dimensioni sostenute, non risulta una scelta vincente.
**Backup fisici**: I backup fisici riguardano le copie fisiche dei filesystem dei nodi. Questo tipo di backup è realizzabile unicamente attraverso l'utilizzo degli strumenti offerti dalla suite mysql-max e proprio per questo, su basi dati di grandezza notevole, è più performante di un backup logico.

Per avviare un backup è sufficiente avviare la console di management e digitare il comando "start backup":

```
ndb_mgm> start backup
Waiting for completed, this may take several minutes
Node 2: Backup 1 started from node 1
Node 2: Backup 1 started from node 1 completed
StartGCP: 300415 StopGCP: 300418
 #Records: 4098 #LogRecords: 0
 Data: 65688 bytes Log: 0 bytes
```

Quanto visualizzato, conferma la corretta esecuzione del backup : nella directory /mysql di ciascun nodo, è stata creata una sottodirectory denominata BACKUP che conterrà a sua volta tante sottodirectory quanti saranno i backup che effettueremo. Per il nodo 2 ad esempio, questa sarà la struttura delle directory:

```
# pwd
/mysql/BACKUP/BACKUP-1
# ls
BACKUP-1-0.2.Data   BACKUP-1.2.ctl   BACKUP-1.2.log
```

I file creati sono tre. Questo perché la struttura dei backup MySQL Cluster è composta da tre parti : i metadata (ossia le informazioni sulla struttura del database), i dati delle tabelle ed il log relativo alle transazioni.

I tre file creati hanno quindi le seguenti corrispondenze:

- BACKUP-<id del backup>.<id del nodo>.ctl contenente i metadata;
- BACKUP-<id del backup>-<id del file>.<id del nodo>.Data contenente i metadata;
- BACKUP-<id del backup>.<id del nodo>.log contenente i log delle transazioni;

### Restore

Per effettuare il restore di backup logici è sufficiente eseguire gli statement sql presenti nel file creato con il comando mysqldump. Discorso a parte va fatto per i backup fisici, il cui restore si può effettuare solo tramite l'utilizzo del programma ndb_restore.

Anche se il backup è stato effettuato in maniera centralizzata, ossia attraverso la console di management, i file di backup effettivi risiedono su ciascun nodo, pertanto il restore va lanciato tante volte quanti sono i nodi dati presenti nel cluster.

Per prima cosa quindi, considerando come le tabelle che verranno ripristinate non dovranno esistere, è necessario rimuoverle dal database di test creato nello scorso articolo:

```
# mysql
Welcome to the MySQL monitor.  Commands end with ;
or \g.
Your MySQL connection id is 1 to server version:
5.0.12-beta-max

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

mysql> drop table ndb_test.test;
Query OK, 1 row affected (0.22 sec)

mysql> quit
Bye
```

Successivamente, va considerato che il programma ndb_restore agisce come un nodo sql e per connettersi al server di management necessita di un node id disponibile.

La configurazione del cluster a cui il management node fa riferimento prevede la connessione di un solo nodo client, che quindi non deve essere occupato dal server mysqld. Prima di lanciare le operazioni di backup è necessario stoppare dalla macchina di management il server mysqld, ad esempio in questo modo:

```
# pkill mysqld
```

a questo punto è possibile avviare il test di restore recandosi sul nodo 2 e lanciando questo comando:

```
# ndb_restore --
connect="nodeid=4;host=management.mycluster.local:11
86" --nodeid=2 --backupid=1 --restore_meta --resto-
re_data /mysql/BACKUP/BACKUP-1/
Ndb version in backup files: Version 5.0.12
Connected to ndb!!
Successfully restored table ndb_test/def/test
_____________________________________________________
Processing data in table: ndb_test/def/test(2)
fragment 0
_____________________________________________________
Processing data in table: sys/def/NDB$EVENTS_0(1)
fragment 0
_____________________________________________________
Processing data in table: sys/def/SYSTAB_0(0)
fragment 0
Restored 0 tuples and 0 log entries
```

Il comando eseguito è autoesplicativo, vengono passati i parametri di connessione (--connect), l'identificativo del nodo a cui appartiene il backup (--nodeid), l'identificativo del backup (--backupid), cosa ripristinare (--restore_meta e –restore_date) ed infine la directory in cui risiede il backup.

Successivamente, la stessa operazione andrà lanciata dal nodo 3 con le dovute modifiche:

```
# ndb_restore --
connect="nodeid=4;host=management.mycluster.local:11
86" --nodeid=3 --backupid=1 --restore_data
/mysql/BACKUP/BACKUP-1/
Ndb version in backup files: Version 5.0.12
Connected to ndb!!
_____________________________________________________
Processing data in table: ndb_test/def/test(2)
fragment 1
_____________________________________________________
Processing data in table: sys/def/NDB$EVENTS_0(1)
fragment 1
_____________________________________________________
Processing data in table: sys/def/SYSTAB_0(0)
fragment 1
Restored 0 tuples and 0 log entries
```

Da notare come non sia presente l'opzione --restore-meta in quanto essendo i metadati già stati ripristinati nel precedente restore, tale operazione non è più necessaria. Una volta riavviato dalla macchina di management il server mysqld:

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

Potremo verificare il corretto ripristino della tabella test attraverso la shell del comando mysql:

```
mysql> use ndb_test;
Reading table information for completion of table
and column names
You can turn off this feature to get a quicker
startup with -A

Database changed
mysql> show tables;
+--------------------+
| Tables_in_ndb_test |
+--------------------+
| test               |
+--------------------+
1 row in set (0.01 sec)
```

## Conclusioni

Ovviamente la tabella di test considerata, non contenendo alcun record, risulta poco attendibile per un test significativo, ma è più che sufficiente per comprendere le meccaniche del funzionamento di questo prodotto. Ciò che si è voluto illustrare è come MySQL Cluster possa rappresentare, una volta debitamente configurato, un'alternativa gratuita, valida e soprattutto ad alta affidabilità rispetto ad altri costosi concorrenti commerciali.

Segnalo alcuni link da cui ho tratto parte delle informazioni presentate in questi articoli:

Il manuale ufficiale di MySQL Cluster: http://dev.mysql.com/doc/refman/5.0/en/ndbcluster.html

Un esempio su come realizzare un cluster a due nodi: http://www.davz.net/static/howto/mysqlcluster

Dell'altra ottima documentazione: http://www.browardphp.com/mysql_manual_en/manual_NDBCluster.html

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