

MySQL Cluster Manager

降低复杂性和风险,同时提高管理器效率

MySQL 白皮书

2010.07

目 录

1 摘 要.....	3
2 MySQL Cluster 管理器介绍.....	3
2.1 MySQL Cluster 管理器自动化管理.....	4
2.2 用 MySQL Cluster 管理器自动监控和自动恢复.....	4
2.3 MySQL Cluster 管理器的高可用操作.....	5
3 MySQL Cluster 管理器—架构和用途.....	6
3.1 MySQL Cluster 管理器架构.....	6
3.1.1 改变了以前管理 MySQL Cluster 方式.....	7
3.2 MySQL Cluster 管理器模式和条件.....	9
3.3 使用 MySQL Cluster 管理器——一个工作实例.....	10
3.3.1 配置举例.....	10
3.3.2 安装、配置、运行和访问 MySQL Cluster 管理器.....	11
3.3.3 创建和开始一个 Cluster.....	12
3.3.4 检查 Cluster 状态.....	14
3.3.5 检查和设置 MySQL Cluster 参数.....	15
3.3.6 升级 MySQL Cluster.....	17
3.3.7 升级 MySQL Cluster 管理器.....	18
3.3.8 关闭 Cluster.....	18
3.3.9 MySQL Cluster 提供的 HA.....	19
3.3.10 在一个现存的 Cluster 中引进 MySQL.....	19
4 结论.....	19
5 关于 MySQL Cluster 数据库.....	20
6 其他资源.....	21

1 摘要

根据行业调查显示，人员编制和停机时间只占数据库所有权总成本的 50% 不到。与此同时，IT 预算和员工水平下降或持平，而高可用网站和融合通信服务需求持续增长，这就给 IT 部门带来巨大压力。

机构要求 IT 更多关注战略创新，并减少 IT 环境的日常管理工作。IT 部门需要通过快速部署新服务或重新配置现有的应用，来灵活满足市场的需求。

与此同时 SLAs（服务等级协议）变得更苛刻。服务中断将直接影响收益和客户满意度，业界一些观察员估计宕机 1 分钟将花费\$30,000，1 小时的话略低于 200 万美元。

由于这些压力，IT 部门必须少花钱多办事。他们必须增加生产率和效率的同时又不影响可用性。而智能的数据库管理工具是确保这类转变的关键。

MySQL Cluster 管理器自动执行常规管理任务，从而简化了创建和管理 MySQL Cluster 数据库的工作。因此，数据库管理员（DBAs）和系统管理员的生产力会更高，确保他们能够集中精力关注 IT 战略创新和更快速响应客户需求。

与此同时，之前由手动配置错误引起的数据库宕机风险就大大减小了。

2 MySQL Cluster 管理器介绍

随着更多用户采用集群化数据库来提高服务的可用性、扩展性和性能，人们日益认识到，集群架构固有的高复杂性会使得管理和经营费用不断增加。

为解决该问题，MySQL 已开发了 MySQL Cluster 管理器，它可以自动简化创建和管理 MySQL Cluster 数据库。

举一个例子，管理操作要求回滚重启一个 MySQL Cluster 数据库，以前需要 46 行手动命令并且消耗 DBA 两个半小时的时间，现在只需要一条单独命令，并且完全是由 MySQL Cluster 管理器完成。该服务减少了以下方面：

- 管理的复杂性和花费
- 自动配置以及变更管理进程的宕机风险。
- 用户管理命令脚本或开发和运维内部管理工具。

MySQL Cluster 管理器通过以下三个核心能力获得这些优势：

- 自动化管理
- 监控和自我愈合恢复
- 高可用性操作

下图显示的是 MySQL Cluster 管理器是如何结合到 MySQL Cluster 架构中去的。

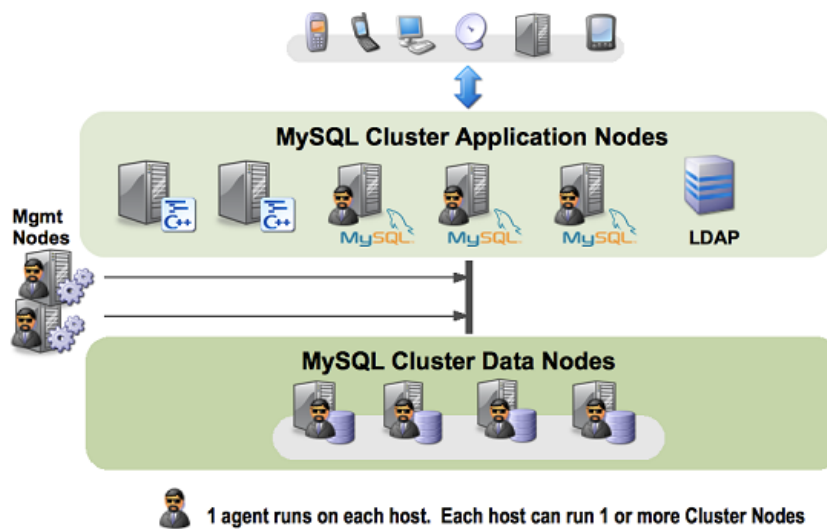


图 1 MySQL Cluster 管理器通过每个数据、SQL 和管理器节点上的分散代理执行

2.1 MySQL Cluster 管理器自动化管理

MySQL Cluster 管理器提供控制整个 Cluster 作为一个单一整体的能力,同时也支持很精细的控制 Cluster 本身内部个别的进程。管理员能够创建和删除整个 Cluster, 并且可以通过一条单独的命令来开始、停止和重启 Cluster。因此, 管理员不再需要手动依次按正确的顺序重启每个节点, 或者创建客户脚本来自动运行进程。

MySQL Cluster 管理器自动在线管理操作, 包括升级、降级和重新配置运行的 Cluster, 并且不会影响应用或服务访问数据库。管理员不再需要手动编辑配置文件并将其分散到 Cluster 节点中, 或者决定是否要回滚重启。MySQL Cluster 管理器完成所有这些工作, 因此实施了最佳实践并使得在线操作尤其简单、快速和更少出错。

2.2 用 MySQL Cluster 管理器自动监控和自动恢复

MySQL Cluster 管理器, 在操作系统和每进程水平上, 都能够监控 Cluster 的健康状态, 通过自动调查 Cluster 中的每个节点来进行监控。它可以侦查到一个进程或服务器主机是否是良好、死机或是危险, 并允许更快速的问题侦查、解决和恢复。

为了达到 99.999% 的可用性, MySQL Cluster 拥有自动解除故障的能力, 通过自动重启失败的数据节点, 无需手动控制就可以实现。MySQL Cluster 管理器扩展了该功能, 也可以监控和自动恢复 SQL 和管理节点。这就支持一个更完美和完整的 Cluster 自我愈合能力, 可

以完全恢复操作和应用。



图 2 MySQL Cluster 管理器监控和自动恢复失败的数据、SQL 和管理节点

2.3 MySQL Cluster 管理器的高可用操作

MySQL Cluster 数据库用于 Web、电信、政府和企业的核心业务应用。因此，MySQL Cluster 管理器不会影响到下层 Cluster 是至关重要的。

为了确保高可用性操作，MySQL Cluster 管理器从实际的数据库进程中分离，因此，如果一个管理代理停止或正在升级，它无论如何也不会影响正在运行的数据库。

为了确保 Cluster 配置的一致性，将会在现存环境中产生重要的管理费用。所有 MySQL Cluster 配置信息和进程标识被保存在磁盘上，当系统故障或重启 MySQL Cluster 管理器使其能幸免于难。当管理代理重启后，它们会自动重新同步其他运行的管理代理以确保配置整个 Cluster 的一致性，不需要管理员的介入。

MySQL Cluster 管理器协调每个管理代理之间的通信以精确传达重置请求。因此，要保持 Cluster 所有节点配置一致，只有当所有节点配置收到重置请求时，所有更改才可提交。如果一个或更多节点没有接收请求，那么就会生成一个错误报告返回给用户。通过自动化通信和同步重置请求，手动分发配置文件引起的错误就会减少。

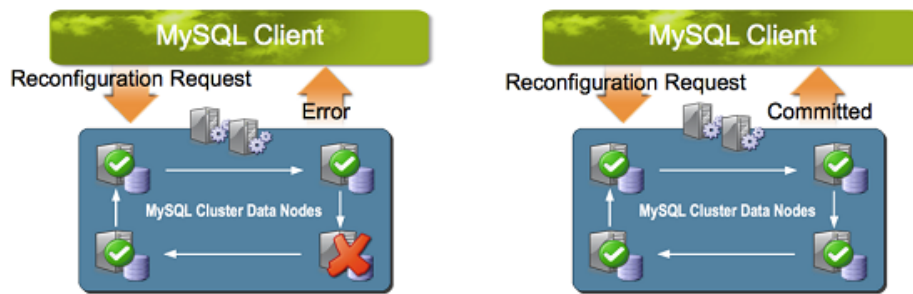


图 3 MySQL Cluster 管理器精确分配 Cluster 重置请求

以上功能描述显示，MySQL Cluster 管理器维持和加强了 MySQL Cluster 数据库的高可用特性。

3 MySQL Cluster 管理器—架构和用途

3.1 MySQL Cluster 管理器架构

MySQL Cluster 管理器作为一系列代理执行——运行在每个物理主机上，并包含管理的 MySQL Cluster 节点。管理员连接普通的 MySQL 用户到这些代理中的任何一个，并且代理之间可以互相通信。在默认情况下，1862 端口是用于连接代理的。4 个主机组成的与 MySQL 用户一起运行在一台笔记本上的例子如图 4 所示：

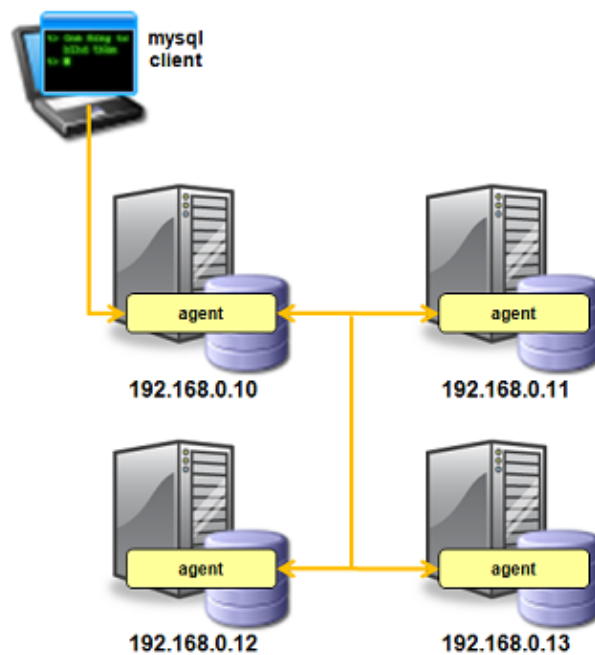


图 4 MySQL Cluster 管理器代理运行在每个主机上

代理共同工作，通过组成 Cluster 的节点执行操作，每个 Cluster 处理其本地节点（这些进程运行在相同的主机上）。例如，如果一个数据节点、管理节点或 MySQL Server 在 X 主机上失败了，那么该进程将会由运行在 X 主机上的代理重启。

在其他的例子中，在一个升级操作事件中代理协作配合以确保每个节点按照正确的顺序升级，但总是本地代理会精确执行升级一个具体的节点。

为了升级 MySQL Cluster 管理器，所有的代理被停止并且新的代理会开始（使用新的软件版本），并且不会影响 MySQL Cluster 数据库继续操作。

3.1.1 改变了以前管理 MySQL Cluster 方式

当使用 MySQL Cluster 管理器管理您的 MySQL Cluster 时，管理员不再编辑配置文件（例如 config.ini 和 my.cnf）；相反，这些文件由代理创建和维护。事实上，如果这些文件是手动编辑的，更改将会被代理商内部产生的配置信息覆盖。每个代理存储所有 Cluster 配置

数据，但是它仅创建运行在那台主机上的配置节点所要求的配置文件。图 5 显示的是该情况

的一个实例。

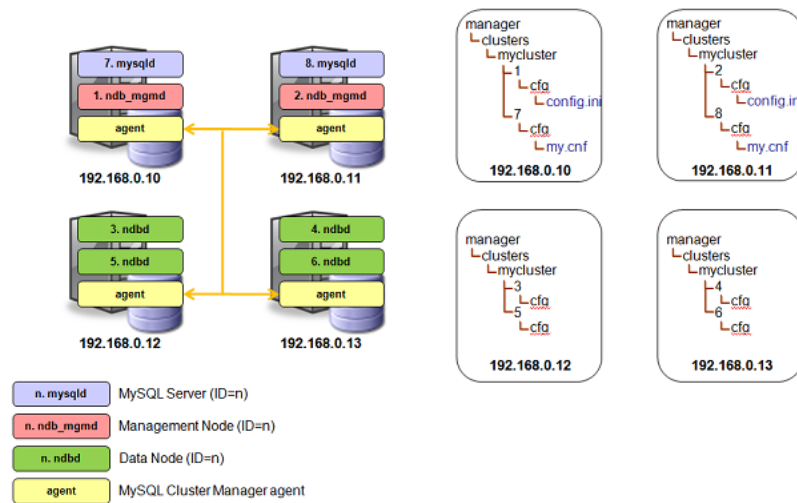


图 5 每个主机上创建的配置文件

同样的，当使用 MySQL Cluster 管理器的时候，管理员不得使用 `ndb_mgm` 命令（直接连接到管理节点也就意味着代理本身将没有与其执行任何操作的可见性）进行管理操作。

MySQL Cluster 管理器的引进不会消除管理节点的需求；特别的是它们还将继续发挥很多关键作用：

- 当数据节点开始（或重启），它们连接到管理节点以检索配置数据（管理节点依次抓取由代理创建的配置文件中的数据）。
- 当通过 MySQL Cluster 管理器停止或重启一个数据节点，状态更改实际上是由管理节点执行的。
- 管理节点能够继续扮演的是仲裁的角色（避免一个裂脑情形）。基于这个原因，它对于运行独立主机上数据节点的这些进程仍然是很重要的。
- 一些记录信息（例如，内存使用）在 Cluster 管理器中目前不可用，并且仍然使用 `ndb_mgm` 工具执行。

当使用 MySQL Cluster 管理器时，“代理”进程不再需要（或创建）数据节点，因为它成为代理侦查失败数据节点的责任以及如果需要的话重建的义务。此外，代理扩大了该功能包

含管理节点和 MySQL 服务器节点。

值得注意的是，代理本身没有**天使进程**，并且因此为了达到最高水平的可用性，管理员也许选择使用一个进程监控器来侦查一个代理故障并可以自动重启它。当一个代理不可用

时，MySQL Cluster 数据库继续以一个容错方式运行，但是管理变更没有执行直到它重启之后。当代理进程重启后，它重新确立了与其他代理的通信，并且已经获得所有当前配置数据，确保管理功能得到恢复。

3.2 MySQL Cluster 管理器模式和条件

之前查看如何安装、配置和使用 MySQL Cluster 管理器，有助于理解它们的若干组成及其之间关系（查看 MySQL Cluster 管理器）。图 6 说明的是该实体的数量并且每部分有一个简单的描述。

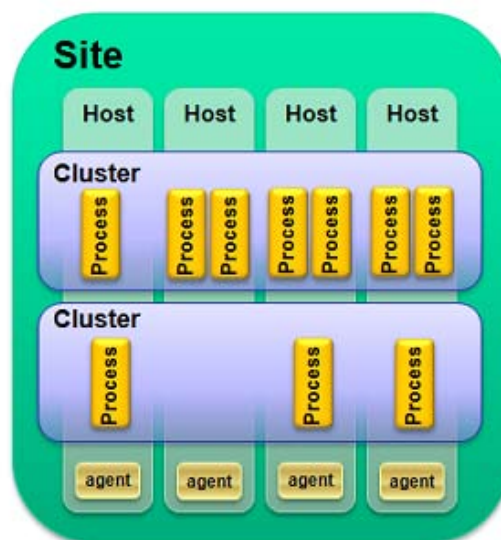


图 6 MySQL Cluster 管理器模式

站点：物理主机用来运行数据库 Cluster 进程，将会通过 MySQL Cluster 管理器管理。1 个站点可以包含 1 个或更多 Cluster⁵。

集群：显示的是一个 MySQL Cluster 部署。一个 Cluster 包含一个或更多进程运行在一

台或更多主机上。

主机：一个物理机器（例如一个服务器），运行在 MySQL Cluster 管理器代理上。

Host: a physical machine (i.e. a server), running the MySQL Cluster Manager agent.

代理：MySQL Cluster 管理器进程运行在每个主机上，负责所有执行进程的管理操作。

进程：一个单独的 MySQL Cluster 节点；ndb_mgmd, ndbd, ndbmtd, mysqld & ndbapi⁶ 中的其中一个。

Package：从 mysql.com 下载一份 MySQL Cluster 安装目录，存储在每个主机上。

3.3 使用 MySQL Cluster 管理器—一个工作实例

本节介绍了安装、配置和运行 MySQL Cluster 管理器的步骤。接着演示了如何使用它来创建和管理 MySQL Cluster 部署。

3.3.1 配置举例

纵观整个案例，MySQL Cluster 配置使用如图 7 所示。

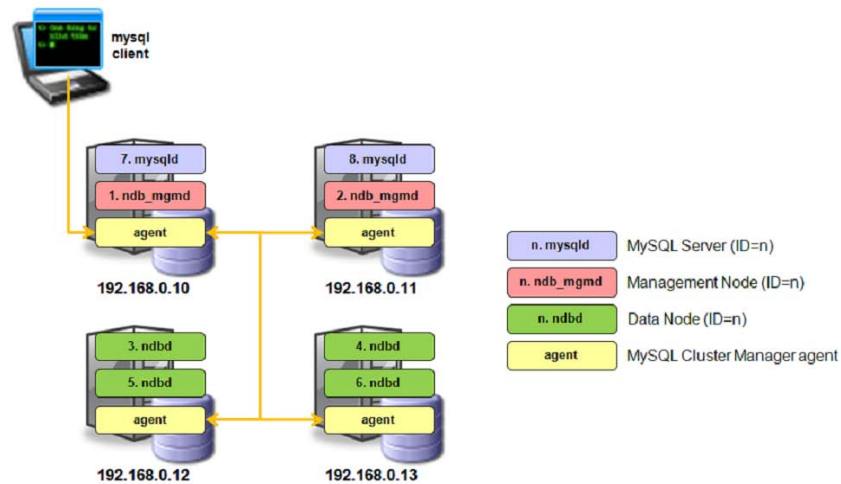


图 7 MySQL Cluster 配置使用举例

3.3.2 安装、配置、运行和访问 MySQL Cluster 管理器

在 Cluster 中代理必须安装和运行每个主机上。

1.将下载的 tar-ball 扩展到一个已知的目录（在本例中我们使用/usr/local/mcm）。

2.复制/usr/local/mcm/etc/mysql-cluster-manager.ini 到/home/billy/mcm/并且编辑它：

```
[mysql-proxy]
```

```
plugins=manager
```

```
log-file = mysql-manager-agent.log
```

```
log-level = message
```

```
manager-directory=/home/billy/mcm/
```

3.复制许可文件（user.dat 和 license.dat）到安装目录中。

4.运行代理进程

```
$ /usr/local/mcm/bin/mysql-cluster-manager --defaults-
```

```
file=/home/billy/mcm/mysql-cluster-manager.ini&
```

5.从任何机器（安装 MySQL 的客户端）访问代理

```
$ mysql -h 192.168.0.10 -P 1862 -u admin -p --prompt='mcm> '
```

Enter password: super

Welcome to the MySQL monitor. Commands end with ; or \g.

Your MySQL connection id is 1

Server version: 1.0.1-agent-manager MySQL Cluster Manager

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mcm>

每个包含管理的 MySQL Cluster 节点的物理主机，在重复 1 到 3 步骤时候，系统看起来如图 8 所示。

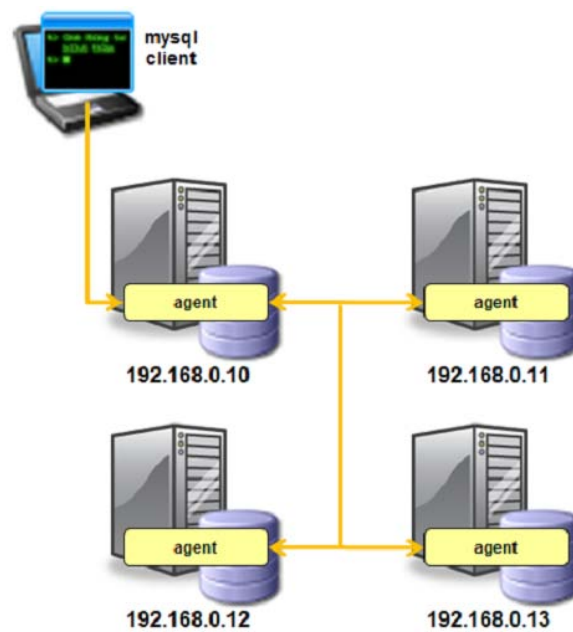


图 8 MySQL Cluster 管理器代理运行在每个主机上

3.3.3 创建和开始一个 Cluster

一旦代理安装和开始之后，下一步就是设置实体来管理 Cluster。作为一个前提条件，需要管理的 MySQL Cluster 版本的安装文件应该被存储在一个已知位置—在这个例子中，`/usr/local/mysql_X_Y_Z`。

如 3.3.2 部分中步骤 4 中，连接到代理进程运行在任何物理主机--从那创建所有需要 Cluster（站点、主机、**packages** 和 Cluster）的实体的整个进程以及开始的 Cluster 可被执行。

1. 定义站点—即运行 MySQL Cluster 进程的的主机设置需要管理。

```
mcm> create site --hosts=192.168.0.10,192.168.0.11,192.168.0.12,192.168.0.13 mysite;
```

2. 定义 **package (s)** 用于 Cluster；更多的 packages 能够晚点添加（例如当升级到最新的 MySQL Cluster 版本时）。在此例子中，每个主机的 MySQL Cluster 安装目录在相同的位置—如果不是此情形—主机选项能够用于辨别哪个主机目录名称适合于：

```
mcm> add package --basedir=/usr/local/mysql_6_3_27a 6.3.27a;
```

```
mcm> add package --basedir=/usr/local/mysql_7_0_9 7.0.9;
```

3. 创建 Cluster，指定 package（MySQL Cluster 版本）并且设置节点/进程以及哪个主机可能运行。

```
mcm> create cluster --package=6.3.26  
--processhosts=ndb_mgmd@192.168.0.10,ndb_mgmd@192.168.0.11,  
ndbd@192.168.0.12,ndbd@192.168.0.13,ndbd@192.168.0.12,  
ndbd@192.168.0.13,mysqld@192.168.0.10,mysqld@192.168.0.11  
mycluster;
```

4. Cluster 现在已经明确—所有剩下的开始的数据库进程变得可用。

```
mcm> start cluster mycluster;
```

```
mcm> start cluster mycluster;
```

需要注意的是，节点 ID 是自动分配的，提供的第一张进程列表由 1 开始，随后的每个

进程由 1 递增。

MySQL Cluster 现在启动和运行如以上图 7 所示。

3.3.4 检查 Cluster 状态

使用 MySQL Cluster 管理器，你可以检查一下若干层次的状态：

- Cluster 整体状态。
- 每个主机状态（特别是主机上的代理是否运行和可访问）
- 组成 Cluster 的每个节点（进程）的状态。

```
mcm> show status --cluster mycluster;
```

```
+-----+-----+
| Cluster | Status           |
```

```
+-----+-----+
| mycluster | fully-operational |
```

```
+-----+-----+
```

```
mcm> list hosts mysite;
```

```
+-----+-----+-----+
|      Host      | Status   | Version |
```

```
+-----+-----+-----+
| 192.168.0.10 | Available | 1.0.1   |
```

```
| 192.168.0.11 | Available | 1.0.1   |
```

```
| 192.168.0.12 | Available | 1.0.1   |
```

```
| 192.168.0.13 | Available | 1.0.1   |
```

```
+-----+-----+-----+
```

```
mcm> show status --process mycluster;
```

```
+-----+-----+-----+-----+-----+
| Id   | Process | Host           | Status | Nodegroup |
```

```
+-----+-----+-----+-----+-----+
| 1    | ndb_mgmd | 192.168.0.10 | running |           |
```

2	ndb_mgmd	192.168.0.11	running	
3	ndbd	192.168.0.12	running	0
4	ndbd	192.168.0.13	running	0
5	ndbd	192.168.0.12	running	1
6	ndbd	192.168.0.13	running	1
7	mysqld	192.168.0.10	running	
8	mysqld	192.168.0.11	running	

+-----+-----+-----+-----+-----+

3.3.5 检查和设置 MySQL Cluster 参数

当使用 MySQL Cluster 管理器时，管理员使用 `get` 和 `set` 命令，检查和改变所有配置参数，而不是直接编辑配置文件目录。通过 `get` 和 `set` 你可以控制属性被读或升级的范围。例如，你可以指明适合所有节点、某一特别类（像所有节点）的所有节点或者一个具体的节点的属性。此外，当读属性时，您可以提供一个具体的属性名称或所有属性适用要求，也可以标明是否是你希望看到的属性的默认值或者被覆盖的那些属性。

以下是 `getting` 和 `setting` 属性的一些例子。

1. 抓取所有合适的数据节点参数，包括默认情况下。

```
mcm> get -d :ndbd mycluster;
```

Name	Value	Process1	Id1	Process2	Id2	Level	Comment
__ndbmt_lqh_threads	NULL	ndbd	3			Default	
__ndbmt_lqh_workers	NULL	ndbd	3			Default	
Arbitration	NULL	ndbd	3			Default	
.....	:	:	:	:	:	:	:
__ndbmt_lqh_threads	NULL	ndbd	4			Default	
__ndbmt_lqh_workers	NULL	ndbd	4			Default	
Arbitration	NULL	ndbd	4			Default	
ArbitrationTimeout	3000	ndbd	4			Default	
.....	:	:	:	:	:	:	:
__ndbmt_lqh_threads	NULL	ndbd	5			Default	
.....	:	:	:	:	:	:	:
__ndbmt_lqh_threads	NULL	ndbd	6			Default	
.....	:	:	:	:	:	:	:

2. 获得 mysqld ID=7 的参数值（包括默认情况）:

```
mcm> get :mysqld:7 mycluster;
```

Name	Value	Process1	Id1	...
datadir	/usr/local/mcm/manager/clusters/mycluster/7/data	mysqld	7	...
HostName	ws1	mysqld	7	...
ndb-nodeid	7	mysqld	7	...
ndbcluster		mysqld	7	...
NodeId	7	mysqld	7	...

3. 获得连接到 mysqld ID=7 时的端口参数:

```
mcm> get -d port:mysqld:7 mycluster;
```

Name	Value	Process1	Id1	Process2	Id2	Level	Comment
port	3306	mysqld	7			Default	

4. 关闭检查所有 MySQL 服务器的特权，并改变连接 mysqld ID=8 到 3307 的端口。允许数据节点在失败时候自动重启:

```
mcm> set skip-grant-tables:mysqld=true,port:mysqld:8=3307,StopOnError:ndbd=false  
mycluster;
```

MySQL Cluster 管理器自动决定哪些节点（进程）需要重启以及以何种顺序更改有效但

又能避免服务丢失。在这个例子中，该结果在每个管理、数据和 MySQL 服务器节点中继续重启如图 9 所示：

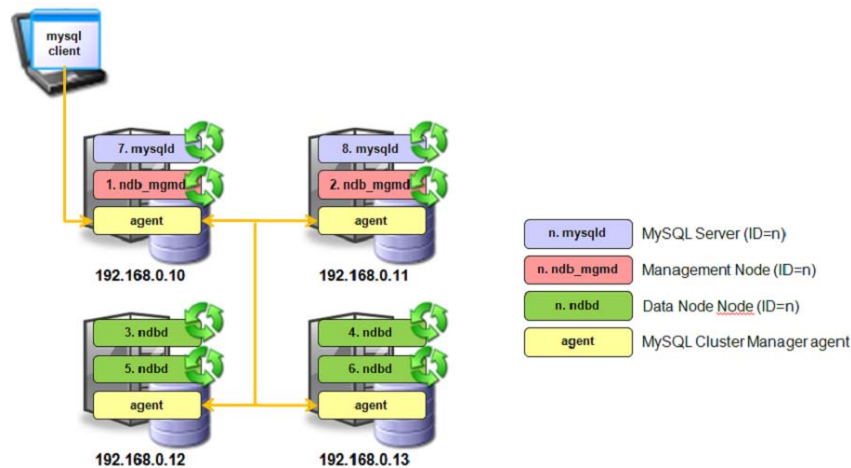


图 9 MySQL Cluster 在配置更改之后节点自动重启

3.3.6 升级 MySQL Cluster

使用 MySQL Cluster 管理器升级 MySQL Cluster 是非常简单的。管理员需要的时间和出现人为错误的机会大大减少。

在执行升级之前有三个准备工作：

1. 确保您打算执行的升级方法被支持，请参看：

<http://dev.mysql.com/doc/mysql-cluster-excerpt/5.1/en/mysql-cluster-upgrade-downgrade-compatibility.html>

如果不支持，您首先需要执行升级到一个中间的 MySQL Cluster 版本。

2. 确保您使用的 MySQL Cluster 管理器既支持原来的也支持最新的 MySQL Cluster 版本，在 <http://dev.mysql.com/doc/mysql-cluster-manager/1.0/en/> 查看参考文件。如果不是，您必须首先升级到 MySQL Cluster 管理器的一个最近的版本，在 3.3.7 中有描述。

3. 确保安装的最新版本 MySQL Cluster 的 tar-ball 已经备份到每个主机并且定义了相关的 packages。

升级本身是使用一条单一的命令来执行的，这会依次升级每个数据节点，这样数据库服务连接到应用和客户端不会被中断。

```
mcm> upgrade cluster --package=7.0.9 mycluster;
```

```
+-----+
```

```
| Command result |
```

```
+-----+
```

```
| Cluster upgraded successfully |
```

```
+-----+
```

```
1 row in set (1 min 41.54 sec)
```

3.3.7 升级 MySQL Cluster 管理器

升级 MySQL Cluster 管理器是非常简单的。简单的停止每个主机上的代理然后开始新版本代理就可以了。相同的 `mysql-cluster-manager.ini` 文件应该被用于新的版本；特别是 `manager-directory` 属性应该和之前使用的版本是相同的，因此配置信息不会丢失。

当 MySQL Cluster 管理器代理停止时而 MySQL Cluster 继续运行，因此升级的时候不会影响服务。如果一个 MySQL Cluster 节点在升级的过程中失败了，数据继续工作但是该节点没有被恢复直到新的代理开始—为此，管理员可能选择当数据库需求最小的时候进行升级。

3.3.8 关闭 Cluster

可以使用一条单独的命令来停止整个 Cluster。

```
mcm> stop cluster mycluster;
```

```
+-----+
```

```
| Command result |
```

```
+-----+
```

```
| Cluster stopped successfully |
```

```
+-----+
```

```
1 row in set (22.96 sec)
```

不像如果没使用 MySQL Cluster 管理器时使用 `ndb_mgm` 的 `shutdown` 命令，停止 Cluster 也将会安全的停止所有 MySQL 服务器进程。

3.3.9 MySQL Cluster 提供的 HA

如果 MySQL Cluster 侦查到一个 MySQL Cluster 进程失败了，那么它将会自动重启之（仅当节点 `StopOnError` 属性设置成 `FALSE` 时才发生）。对于数据节点，代理从现存的 MySQL Cluster 数据节点天使进程中接管该功能，但是对于管理 MySQL 服务器节点来说这是个新功能。

重启 MySQL Cluster 管理器代理失败了，这是管理员或应用的责任，例如创建一个脚本在 `/etc/init.d`。

3.3.10 在一个现存的 Cluster 中引进 MySQL

在 MySQL Cluster Cluster1.0 中，在 MySQL Cluster 管理器管理下没有自动引进 MySQL Cluster 部署的方法，该功能将会在未来版本中加入。与此同时，<http://dev.mysql.com/doc/mysql-cluster-manager/1.0/en/mcm-install-cluster-migrating.html> 会提供可用的文档程序。

4 结论

本文我们介绍了 MySQL Cluster 管理器通过自动处理常规的管理任务是如何简化创建和管理 MySQL Cluster 数据库的。

之前的管理操作需要管理员手动输入多条命令,才能够确保每个操作按照正确的顺序执行,而现在只要一条单独的命令就可以执行了,并且完全自动化运行。

MySQL Cluster 管理器拥有以下三个核心能力:

- 自动管理
- 监控和自我愈合恢复
- 高可用操作

拥有这些功能, MySQL Cluster 管理器使用户受益,减少以下方面:

- 管理的复杂性和经费
- 自动配置和更改管理进程引起的宕机风险
- 管理命令的用户脚本或内部开发和运维管理工具

因此,管理员生产力变得更高,确保他们可以集中精力在 IT 战略创新上并且快速响应客户不断变化的要求。与此同时,之前手动配置错误引起的数据库宕机风险显著减少了。

5 关于 MySQL Cluster 数据库

MySQL Cluster 是一个高可用、实时性关系型数据库,拥有一个“无共享”分布式架构,无单点故障,可达 99.999%的可用性,可以满足您最苛刻的关键任务应用的要求。

MySQL Cluster 实时性设计能够交付可预测的、毫秒级响应时间,拥有每秒处理数以万计的事物的能力。支持内存和磁盘的数据,自动数据分区实现负载均衡,并且拥有在线零宕机添加数据节点能力,允许线性扩展数据库来处理最不可预知的工作量。

MySQL Cluster 的优势已经被大多数最苛刻的数据库管理环境所认可,像电信、Web、

金融和政府部门,如 Alcatel-Lucent, Cisco, Ericsson, Juniper, Shopatron, Telenor, UTStarcom 和 US Navy.

学习更多 MySQL Cluster,参考“MySQL Cluster 架构和新特性白皮书”,在 MySQL Cluster 资源页面的技术白皮书部分 <http://www.mysql.com/products/database/cluster/resources.html>

6 其他资源

Web 上的 MySQL Cluster Manager:

引进 MySQL Cluster Manager 1.0 来运行一个 Cluster:

<http://dev.mysql.com/doc/mysql-cluster-manager/1.0/en/mcm-install-cluster-migrating.html>

MySQL Cluster Manager 文档:

<http://dev.mysql.com/doc/mysql-cluster-manager/1.0/en/>