NEWTON PAIVA
Centro Universitário

NOMES:

Leonardo Claro

Diego Lage

Charles Tancredo

Márcio Castro

O MySQL Cluster é versão do MySQL adaptada para um ambiente de computação distribuída, provendo alta disponibilidade e alta redundância utilizando a NDBCLUSTER storage engine. Tal ação permite a criação de um cluster ou conjunto de nós com vários computadores.

Este cluster não deve ser confundido com a replicação tradicional, onde um servidor MySQL denominado Mestre atualiza um ou mais escravos. Desta forma, as transações são comitadas sequencialmente, e um atrazo nestas resulta em um banco escravo desatualizado. Caso o Mestre falhe, o escravo pode não ter todas as atualizações gravadas.

No Cluster, todos os nós são colocados em sincronia, e uma transação comitada em qualquer nó é comitada em todos os demais nós. Caso um dos nós falhe, os demais permanecerão em um estado consistente. O gerenciamento é efetuado via um servidor com o software Management Node, onde são configurados todos os parâmetros, logs, controle dos nós e adição de mais nós no Cluster.

Outra vantagem do Cluster é que não é compartilhada nenhuma informação sobre configuração entre os nós, ou seja, não é preciso mudar a configuração de todos os nós atuais para adicionar mais um. Com isso, fica fácil adicionar novas máquinas, sendo possível aumentar o poder de processamento do seu cluster sem a necessidade de reconfigurar todo o cluster.

O Cluster é composto por três tipos diferentes de nós:

a - Data Node (nó de dados): onde são armazenados seus dados;

b - Management Node (nó de gerenciamento): que controla o cluster;

c- SQL Node (nó SQL): onde são executadas as queries.

E dois tipos diferentes de clientes:

a - MySQL Client (cliente do banco): que administra os nós SQL (e executa queries também);

b - Management Client (nó de dados): que administra o restante do cluster de banco de dados.

São necessários então 4 computadores para a configuração do Cluster, de acordo com a configuração recomendada, onde cada um executa os nós de gerenciamento

e SQL, e dois servem como data nodes.
Esses nós compartilham os dados entre si em uma arquitetura que possui dois conceitos básicos: réplicas e fragmentos.
Réplicas são o número de vezes que o dado aparecerá no cluster como um todo e Fragmentos são o número de pedaços em que essa réplica será repartida e distribuída nos nós de dados, ou seja, se tivermos duas réplicas, teremos cada dado armazenado no banco duas vezes, e se tivermos dois fragmentos, precisaremos de dois nós de dados diferentes para armazená-las.
Podemos criar mais de uma réplica (ou seja, duas ou mais cópias do dado original), mas além de não ser recomendado pelo time do MySQL, causa um stress desnecessário ao mecanismo síncrono de persistência do Cluster e acaba gerando uma demora na escrita.
O que devemos focar na verdade é em aumentar o número de fragmentos! Neste caso, quanto mais, melhor (até um limite gerenciável, que depende das suas máquinas e da qualidade da rede entre elas). O Cluster trabalha com números pares de nós de dados que obedece a razão Réplicas x Fragmentos, ou seja, com duas réplicas e dois fragmentos temos quatro nós, com duas replicas e três fragmentos temos seis nós e assim por diante, ou seja, sempre que adicionamos mais máquinas temos que adicionar 2 de por vez. Entretanto o cluster não compartilha nenhuma informação de configuração, portanto não tem necessidade de que se adicione maquinas idênticas às que já estão em produção, porém tem que tomar cuidado para não adicionar uma máquina inferior as que já estão em produção, pois todos os nós são tratados de forma igual, caso coloque uma máquina inferior, a performance da aplicação que será usado no balanceamento dos dados a da máquina inferior, prejudicando então o resultado do cluster.

Uma ótima pratica de cluster é colocar um cluster para dados de alta escalabilidade e outro com replicação simples dos dados mais volumosos que não são tão requeridos, gerando assim uma alta disponibilidade para os dados mais críticos das nossas aplicações.
O Cluster garante a disponibilidade dos dados em uma memória volátil, porém todas as transações são logadas em um arquivo, e quando esse arquivo fica grande é salvo em disco, com isso caso ocorra alguma falha, os dados serão recuperados e

inseridos na base em disco.
O Cluster de nós de dados é interpretado como sendo um grande servidor MySQL, rodando em uma base de dados do tipo NDB(Network DataBase), porem dificilmente ele interage com um cliente MySQL normal. Também existem outras tarefas como LOGs binários, conection manager, preparo das queries e manipulação dos resultados. Por isso existem os nós SQL (SQL Node). SQL Node é onde as aplicações devem realizar suas queries. Você pode ter quantos nós SQL precisar.
Uma boa prática é ter mais de um SQL Node, pois caso tenha somente um representante será um ponto único de falha no caso de uma queda da máquina.

Existe uma técnica chamada IP TakeOver, que cria um grupo de máquinas que respondem por apenas um IP, sendo que se não houver falha, a máquina MASTER responde pelo IP definido, caso a máquina MASTER caia, a segunda máquina(SLAVE) troca o seu IP assumindo, para continuar respondendo a requisição.

Outra técnica menos elegante seria usar um roteador especifico para realizar o balanceamento de carga, agrupando todos os SQL Node, e por segurança preventiva ser dois desses roteadores de alta disponibilidade, assim não tendo falhas nos SQL Node.

É possível escalar separadamente o cluster de nós de dados e de nós de SQL, ou seja, caso a carga maior está na manipulação de dados, adicone mais dois servidores como nós de dados, senão se for no gerenciamento de conexão ou preparos das queries, adicione mais um nó SQL.

No MySQL existe um gerenciador de cluster chamado Management Node. Ele habilita e desabilita os nós, analisando os status dos nós e ainda realizando backups on-line dos dados, também ele consegue fazer verificações dos status das transações e queries, analisando e exibindo estatísticas do cluster, e logs.

Management Node também pode colocar o cluster em modo “Single User”, que fica disponível apenas para os usuários, e nomear um nó SQL como sendo o único

capaz de acessar o cluster. Isso é bom quando se faz necessário uma manutenção mais pesada do cluster, depois de uma quebra geral ou durante uma manutenção de rede prolongada.

Grande cuidado com o comando para entrar em single user mode, as transações que estão sendo executadas serão finalizadas sem nenhum aviso! Não é aconselhável executar este comando em bases de produção.

Instalando e configurando o MySQL Cluster

O processo de instalação do MySQL Cluster é muito simples. Se você utiliza Linux, Solaris ou MacOS, e tem o pacote de versão for a Max 4.1.12 ou posterior, o cluster já está instalado em sua máquina.
Se você não tenha o pacote descrito à cima instalado, basta entrar no site da MySQL, baixar o pacote Max-4.1.12 (ou posterior) e instale nas máquinas que farão parte do cluster.

1. Realize o download do MySQL Max for Unix.

```
$ wget http://dev.mysql.com/get/Downloads/MySQL-
4.1/mysql-max-4.1.13-pc-linux-gnu-i686.tar.gz/from/
http://www.linorg.usp.br/mysql/
$ tar zxvf mysql-max-4.1.13-pc-linux-gnu-i686.tar.gz
$ mv mysql-max-4.1.13-pc-linux-gnu-i686 mysql
$ cd mysql
```

Na demonstração será utilizado o diretório /home para configurar o cluster. No Windows você tem que apontar para o diretório onde esta a base de dados. Descompactando o arquivo e altere o nome para ‘mysql’.
Agora iremos configurar o cluster do mysql. Devemos alterar dois arquivos de configuração, o ‘my.cnf’ e o ‘config.inc’ para o cluster. Se o arquivo 'my.cnf' não existir no diretório dos arquivos de configuração, copie-o do diretório do mysql denominado de ‘support-files’. Neste diretório existes versões de exemplo do 'my.cnf'. O diretório de configuração padrão no Unix é o /etc, já no Windows

encontrasse no banco os arquivos de configuração do Mysql.
Se você realizou o download do mysql no site descrito a cima, Deveremos criar um usuário do MySQL, veja a seguir (2).

2. Grupo e o usuário mysql no Unix.
$ addgroup mysql
$ adduser -g mysql mysql

Para criar usuários no Unix o comando varia dependendo da versão do sistema. Os mais comuns são: ‘adduser’ e ‘useradd’, ‘addgroup’ e ‘groupadd’.
Execute o script 'configure' que esta localizada no diretorio raiz do Mysql. Obs.: execute como root.
Verifique se os diretórios de logs e de dados estão com permissão de escrita para o user 'mysql'.

3. Removendo as tabelas InnoDB e BDB para simplificar o
exemplo.
skip-innodb
skip-bdb

Assim iremos concentrar no cluster. Pode se utilizar uma configuração mias simples para efeito didático.
A estrutura é composta de apenas duas máquinas: maquinaA e maquinaB. Na maquinaA colocaremos os nós de gerenciamento e SQL e também um nó de dados, e na maquinaB colocaremos apenas um nó de dados.
Para que o cluster funcione, precisamos adicionar as seguintes linhas no ‘my.cnf ’ de todas as máquinas que vão rodar nós de dados, no nosso caso, as duas máquinas (4).

4. Configuração comum.
[mysqld]
(...)
ndbcluster

```
ndb-connectstring=maquinaA  #  endereço do nó de gerenciamento
[mysql_cluster]
ndb-connectstring=maquinaA  #  endereco do nó de gerenciamento
```

Precisamos criar a configuração básica do cluster agora. Que conterá as listas de todos os nós, chamado ‘config.ini’ (ver 5).

5. Configuração do cluster.

```
# config padrão do cluster
[NDBD DEFAULT]
NoOfReplicas=2
DataMemory=10M
IndexMemory=10M
[TCP DEFAULT]
portnumber=2202
# config do no de gerenciamento
[NDB_MGMD]
hostname=maquinaA
datadir=~/mysql/data
# nó de dados A
[NDBD]
hostname=maquinaA
datadir=~/mysql/data
# nó de dados B
[NDBD]
hostname=maquinaB
datadir=~/mysql/data
# nó SQL
[MYSQLD]
```

Na seção MYSQLD não esta preenchida para que seja possível conectar o tanto de nós que forem necessários. O ‘datadir’ esta diretório home no subdiretório ‘mysql’ criado anteriormente neste exemplo.

Mova o arquivo ‘config.ini’ no diretório /etc. Estamos prontos para iniciar!

Iniciando o cluster

Para observar todas as etapas da inicialização, vamos começar pelo nó de gerenciamento que monitora os nós do cluster. Para isso, inicie o Manager (6).

6. Iniciando o nó de gerenciamento.

```
$ ~/mysql/bin/ndb_mgmd
```

Abra o cliente de gerenciamento (7), para verificar que o manager está rodando. Deverá aparecer o texto “-- NDB Cluster -- Management Client --” na tela e aparecer o prompt “ndb_mgm>” para que você digite os comandos de gerenciamento. Para testar, digite show. Deverão aparecer informações como as mostradas na (7).

7. Tela principal do gerenciamento do cluster.

```
$ ~/mysql/bin/ndb_mgm
ndb_mgm> show
Cluster Configuration
---------------------
[ndbd(NDB)] 2 node(s)
id=2 (not connected, accepting connect from maquinaA)
id=3 (not connected, accepting connect from maquinaB)
[ndb_mgmd(MGM)] 1 node(s)
id=1 @maquinaA (Version: 4.1.13)
[mysqld(API)] 1 node(s)
id=4 (not connected, accepting connect from maquinaA)
```

Para conectar a segunda máquina do nó do cluster deverá ser executado o mesmo comando executado para subir a primeira máquina. Esse comando deve ser executado conectado na máquina que contém o segundo nó.

Com o comando "Show" no console é possível verificar o nó conectado.

Posteriormente é necessário iniciar o nó SQL.

Rodando o comando "Show" será possível verificar ambas as máquina conectadas.

Para verificar que o cluster realmente funciona crie uma tabela é insira dados.
Obs.: é necessário definir a engine do cluster, só assim os dados serão replicados para os todos os nós.

Agora que todos os nós contém dados, deve-se retirar nó do ar e inserir dados novamente na tabela criada.
Então os dados foram inseridos somente em um nó, pois um deles está parado.

Se rodar um select na tabela é possível verificar que os dados estão realmente lá.

Agora deve-se religar o nó que foi desligado e desligar o que estava ligado.

Fazendo um novo select percebe-se que todos os dados inseridos estão também neste nós que está ativo. Isso acontece porque quando se ativa o primeiro nó que foi desligado o MYSQl sincroniza os dados entre os nós.
Obs.: Não é possível parar um nó antes que a sincronização dos dados seja totalmente realizada o que garante a qualidade dos dados.

Quando se tenta parar o ultimo nó ativo (o único em funcionamento) ele apresenta um erro salientando que isso causaria uma falha de sistema, o que garante que os dados estejam sempre disponíveis.

A diferença básica entre o MySQL cluster e a replicação simples é na parte da disponibilidade automática.

Um cenário de replicação master-slave deve ser encarada mais como segurança de dados, backup e disponibilidade a frio do que um mecanismo de alta disponibilidade.

Já o cluster consegue prover, por meio de uma arquitetura consistente, a certeza de que se uma máquina falhar os dados ainda estarão disponíveis como se nada tivesse acontecido.