MySQLFabric概述

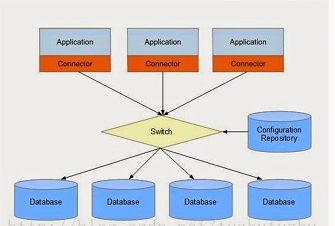
Oracle在2014年5月推出了一套为各方寄予厚望的MySQL产品 MySQL Fabric，从字面上不太能看出它是啥，但是从名称上还是有迹可循的。fabric是“织物”的意思，这意味着它是用来“织”起一片MySQL数据库。MySQL Fabric是一套数据库服务器场(Database Server Farm)的架构管理系统。

# MySQL Fabric是什么？

MySQL Fabric能“组织”多个MySQL数据库，是应用系统将大于几TB的表分散到多个数据库，即数据分片(Data Shard)。在同一个分片内又可以含有多个数据库，并且由Fabric自动挑选一个适合的作为主数据库，其他的数据库配置成从数据库，来做主从复制。在主数据库挂掉时，从各个从数据库中挑选一个提升为主数据库。之后，其他的从数据库转向新的主数据库复制新的数据。注意：这里说的“自动”是指由MySQL Fabric在后台完成，而不需要用户手动更改配置。最重要的是，MySQL Fabric是GPL的开源软件，也就是在符合GPL的规范下，你可以自由的使用和修改这个软件。

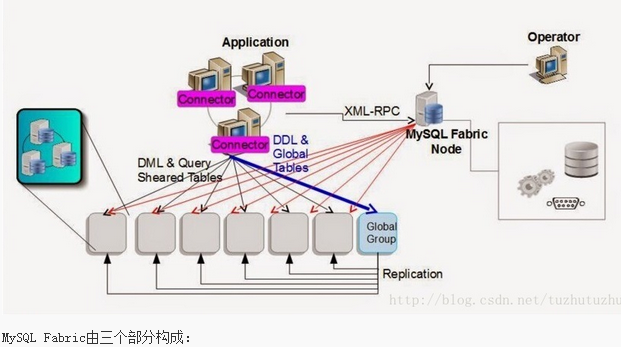
# MySQL Fabric要解决的问题

为什么做数据分片？当你的应用需要处理的表大于1TB的数据时，Data Shard常常是必须的。这么大的表，无论在查询、更新的效率上，或者是备份、更改结构所需要的时间上，都会造成很大的问题。然而当你将这么大的表分散到多个数据库服务器上，又会使每一台数据库服务器都有可能是单个故障点。只要有一台挂掉就会使整个系统的操作发生问题。另一方面，应用端的程序也会因为每个查询都要依其查询条件(where子句的内容)分别指向不同的数据库而变得更加复杂。再者，当数据分片的结构改变时(例如增加一个数据库)，会使应用端的所有程序都必须修改，从而导致维护变得极为复杂。为了解决应用程序复杂度增加的问题，有人在应用程序和数据库服务器之间增加一个代理(proxy)或者成为switch，应用程序所有对数据库的指令先送到proxy，再由proxy判断要转到哪个数据库。下图就是这个方案的示意图。这也许可以解决应用程序难以维护的问题，但是当应用端的数量增加，数据库分片增加，或者系统压力增加时，这个switch会成为容量及性能的瓶颈和单点故障(当它宕掉时，应用端找不到数据库)，而且所有的数据库指令均需要传两次(先到switch再到数据库)。每个查询都会造成额外的负荷。



# MySQL Fabric的架构

MySQL Fabric采用不用的做法，其架构如下图所示。主要的特点是把switch合并到各应用端的connector中，以解决单一switch的单点故障和性能瓶颈。



MySQL Fabric由三个部分构成：

## MySQL Fabric管理节点

是一个python脚本，是整个架构的核心。MySQL Fabric管理节点主要的功能是管理整个数据库服务器场(Database Server Farm),它启动时会找/etc/mysql/fabric.cnf这个配置文件，由它指定fabric背后当成存放Server Farm架构和配置之repository的MySQL数据库位置、端口和连接账号等信息。Fabric在初始化时(执行mysqlfabric manage setup命令)，会在MySQL数据库上开一个schema(通常是名称为fabric的database)，存放Server Farm的配置相关信息，如哪些服务器组由哪些数据库构成，各服务器组中的主从服务器分别是哪些，等等。MySQL Fabric节点在设置配置时，会对Server Farm中各数据库下达建立主从复制的命令(上图的红色线条)。在正常运行时定期ping各组的主服务器 ，当发现主数据库没有正常运行时，它会启动故障转移程序，在该server farm的从数据库中找一个合适的提升为主服务器。其他的从数据库则转向新的主数据库继续复制数据。

## 数据库服务器场(database server farm)

这是整个架构中的工作引擎，在传统的数据库应用中这是单一的MySQL数据库，MySQL Fabric则是以多个数据库支持大数据量表(TB级以上)和高可用性数据库的需求。这些数据库分成几个高可用组(HA Group)，每个组包含一个以上的数据库服务器，上图中最下面的几个灰色和浅蓝色的方块代表高可用组。如果高可用组中有多个数据库，MySQL Fabric会挑选(使用命令mysqlfabric group promote命令)一个提升为主数据库(Master)，其他数据库则成为从数据库(Slave)，从数据库复制主数据库的变化，完成设定同一高可用组内的主从复制。以后，Fabric会定期件事这个主数据库。当主数据宕掉之后，Fabric会从高可用组内挑选一个提升为主数据库，其他的数据库会转向这个新的主数据库继续复制。另一方面，MySQL Fabric也会只是应用端的conector对这些主从数据库做读写分离，当应用程序对数据库做读写兼有的操作时，connector会将该指令提交给主数据库。如果应用程序只会对数据库进行读操作，且连线的read\_only参数设置为“ON”，则所有的查询均轮流传送到这几个数据库。借助读写分离，应用系统的资料处理能力得以增加。此外，如前面所述，MySQL Fabric还能处理需要拆分到多个数据库服务器的表(sharding tables)，每一个高可用组都可能存放shard table的部分数据。应用端的connector会将对shard table的指令依MySQL Fabric的管理节点的设定送到不同的高可用组，这样可使数据库的容量随着高可用组的数量增加而增长。同时，对非拆分的表所下的指令和所有的DDL会由connector送到全局高可用组(global group)，全局高可用组的主数据库被MySQL Fabric设置为其他高可用组的主数据库。所有存拆分表的高可用组的主数据库复制global group的变化，这么一来其他高可用组都有一份非拆分表的资料。从而使得SQL中拆分表对非拆分表的JOIN操作变得更简单。

## Connector

应用系统在运行时，每个SQL指令都会经由connector发送到数据库。MySQL Fabric所搭配的connector和一般使用单机的MySQL数据库一样，只是在较新版的connector是fabric aware connector多了一些能处理数据库服务器场(database server farm)的功能。使他们能在建立数据库连接时，以XML-RPC协议检查MySQL Fabric的管理节点中server farm的配置，然后通过该连接下的查询可依fabric的指示送到适合的数据库。如此一来，常见的database shard方案中可能造成性能瓶颈的proxy放到connector中，从而解决了这个问题。目前MySQL Fabric支持的技术有java、python、PHP，即Connector/J、Connector/Python和Connector/PHP都是Fabric-aware。以java为例，JDBC driver必须是Connector/J 5.1.30以后的版本，Fabric的Java程序和一般对单机MySQL的查询的Java程序差不多，只是在建立database connection object时database connection URL不是指向数据库，改为指向MySQL Fabric管理节点(服务器的IP和端口通常是32274)。当查询的表时全局表(不做table shard)或DDL(例如建表或改表结构)时，建立connection object的要加上''fabricServerGroup="参数，之后通过这个connection object所下的SQL指令会送到Global Group的主数据库，再由数据库复制到其他的高可用组(shard)中。如果SQL命令所要操作的表是分区表(shard table)，则建立connection object时要在参数加上''fabricShardTable="参数，之后通过这个connection object所下的SQL命令会根据MySQL Fabric所设定的分表(shard)原则送到各分区(shard)的高可用组。这样一来，应用程序对这些shard table下的SQL指令时，不需要在SQL中判断要送到哪个数据库，完全由connector在建立数据库连接时向MySQL Fabric所查到的server farm的配置信息(哪个数据库属于哪个shard group，各shard table的拆分原则等)所决定。而且这个配置在建立主连接后就缓存在Connector所在的应用端。这样，在每次下SQL指令时就不需要重复查询MySQL Fabric管理节点，而依存于应用端的分表配置直接送到正确的数据库。而应用程序的效率不会因为做了表的拆分而有任何降低。