# 数据库规划

## 一期数据库架构

数据库架构图：



数据库环境：

服务器：linux

数据库软件：mysql

数据库读写分离软件：mysql proxy

问题：

主从复制也带来其他一系列性能瓶颈问题：

1. 写入无法扩展

2. 写入无法缓存

3. 复制延时

4. 锁表率上升

5. 表变大，缓存率下降

配置目的：

读写分离。写操作涉及到锁的问题，不管是行锁还是表锁还是块锁，都是比较降低系统执行效率的事情。我们这样的分离是把写操作集中在一个节点上，而读操作其其他的N个节点上进行，从另一个方面有效的提高了读的效率，保证了系统的高可用性。

当数据库宕机的时候可以自动将应用连接切换到从库上。从上层的java程序来讲，不需要知道主服务器和从服务器的来源，即主从数据库服务器对于上层来讲是透明的。

目前系统中针对数据量做了不同的分区，在查询数据时，使用上分区字段，可以达到数据行分片。不同分片存储在各自的数据文件中，查询不同的数据可以避免读统一数据文件的，造成热点的问题。

已解决：

1. 主从复制
2. 增量备份、全量备份和异地备份
3. 数据库监控（zabbix）
4. 业务分库

未解决问题：

1. 主从复制延迟验证——目前有监控
2. 主从复制数据一致性验证
3. 主备库自动切换
4. 读写分离
5. 慢日志的解析及监控

## 二期数据库架构

数据库架构图：



数据库环境：

服务器：linux

数据库软件：mysql

数据库读写分离软件：mysql proxy

问题：

1. 单个库数据量太大，单个DBServer存储空间不够
2. 单个库表太多，查询的时候，代开表操作也消耗系统资源
3. 单个表容量太大，查询的时候，扫描行数过多，磁盘IO大，查询缓慢
4. 单个库表能承载的访问量有限，再高的访问量也只能通过分库分表实现

配置目的：

数据量达到亿万条时，做数据库分库分表。采用

## 三期数据库架构

随着数据量的增加，业务复杂度的增加，可以采用集群方式存储数据。

# 数据库规划——工具

## MySQL Proxy(oracle)

### 功能介绍

MySQL Proxy是一个处于你的client端和MySQL server端之间的简单程序，它可以监测、分析或改变它们的通信。它使用灵活，没有限制，常见的用途包括：负载平衡，故障、查询分析，查询过滤和修改等等。

MySQL Proxy就是这么一个中间层代理，简单的说，MySQL Proxy就是一个连接池，负责将前台应用的连接请求转发给后台的数据库，并且通过使用lua脚本，可以实现复杂的连接控制和过滤，从而实现读写分离和负载平衡。对于应用来说，MySQL Proxy是完全透明的，应用则只需要连接到MySQL Proxy的监听端口即可。当然，这样proxy机器可能成为单点失效，但完全可以使用多个proxy机器做为冗余，在应用服务器的连接池配置中配置到多个proxy的连接参数即可。

MySQL Proxy更强大的一项功能是实现“读写分离”，基本原理是让主数据库处理事务性查询，让从库处理SELECT查询。数据库复制被用来把事务性查询导致的变更同步到集群中的从库。

在Proxy启动时可以指定Proxy所需要使用的lua脚本，在lua脚本中预先实现6个方法：

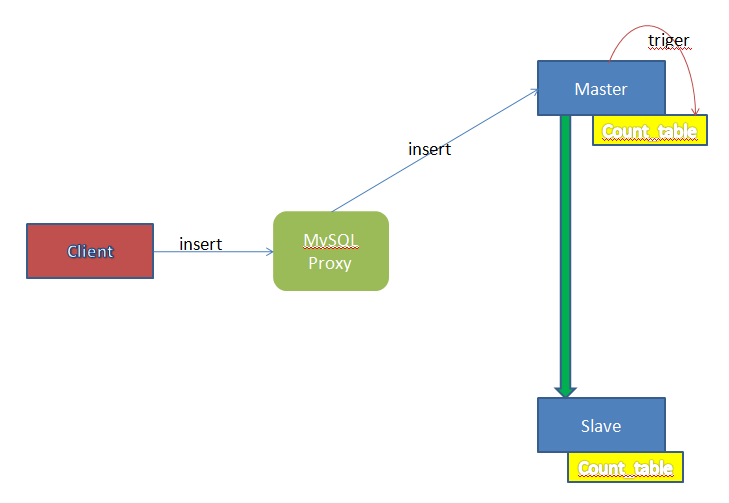
* connect\_server()     // 接收到Client的连接请求时调用
* ead\_handshake()   //
* read\_auth()               // 读取Client的认证信息时调用
* read\_auth\_result() // 读取认证结果时调用
* read\_query()            // 读取Client的query请求时调用
* read\_query\_result()   //读取query结果时调用

解决读写分离：

通过在read\_query()中加入代码，我们可以截取出当前的请求是insert、update还是select，然后把 insert和update请求发送到Master中，把select请求发送到Slave中，这样就解决了读写分离的问题。

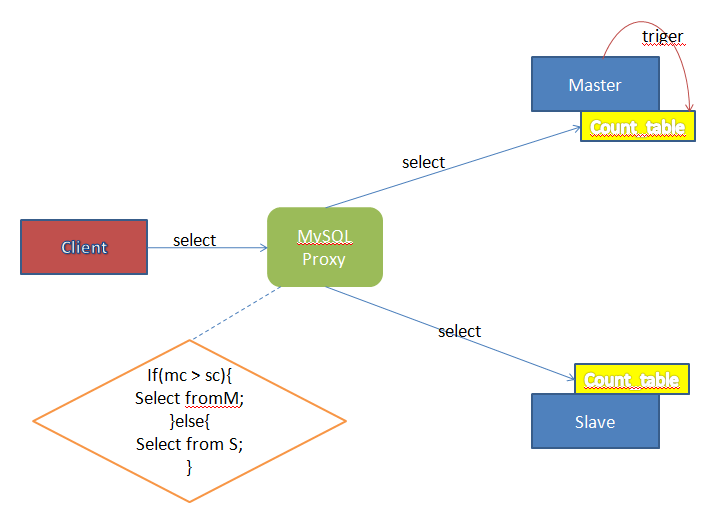
解决同步延迟:

在Master上增加一个自增表，这个表仅含有1个的字段。当Master接收到任何数据更新的请求时，均会触发这个触发器，该触发器更新自增表中的记录。如下图所示：



mysql\_proxy\_write

由于Count\_table也参与Mysq的主从同步，因此在Master上作的 Update更新也会同步到Slave上。当Client通过Proxy进行数据读取时，Proxy可以先向Master和Slave的 Count\_table表发送查询请求，当二者的数据相同时，Proxy可以认定 Master和Slave的数据状态是一致的，然后把select请求发送到Slave服务器上，否则就发送到Master上。如下图所示：



mysql\_proxy\_read

通过这种方式，就可以比较完美的结果MySQL的同步延迟不可控问题。之所以所“比较完美”，是因为这种方案double了查询请求，对 Master和Slave构成了额外的压力。不过由于Proxy与真实的Mysql Server采用连接池的方式连接，因此额外的压力还是可以接受的。

解决failover：

利用mysql\_proxy实现failover，默认连接A，如果A宕掉则连接B，A启动后再连接到A，但是mysql\_proxy如果单点故障就不能够正常连接数据库。

Mysql proxy+Keepalived

### 产品优点

MySQL Proxy具有很大的灵活性，我们可以最大限度的使用它。

### 产品缺点

* MySQL Proxy实际上并不直接提供相关功能，这些功能都要依靠自行编写LUA脚本实现。
* MySQL-Proxy子进程异常退出。
* 不支持大数据量的分库分表且性能较差。

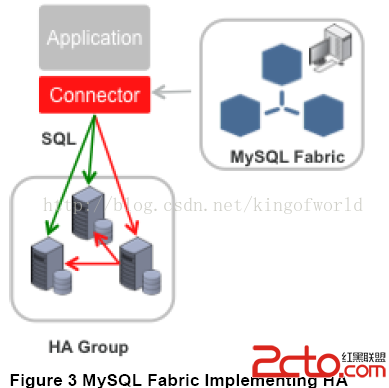
## MySQL Fabric(oracle)

### 功能介绍

为了实现和方便管理MySQL 分片以及实现高可用部署，Oracle在2014年5月推出了一套为各方寄予厚望的MySQL产品 -- MySQL Fabric, 用来管理MySQL 服务，提供扩展性和容易使用的系统，Fabric当前实现了两个特性：高可用和使用数据分片实现可扩展性和负载均衡，这两个特性能单独使用或结合使用。

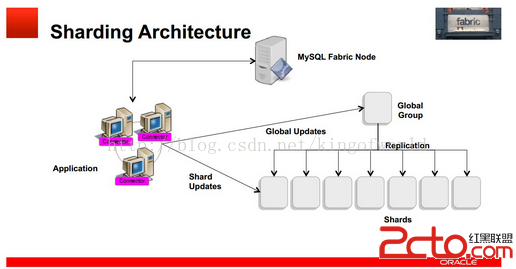
MySQL Fabric 使用了一系列的python脚本实现。

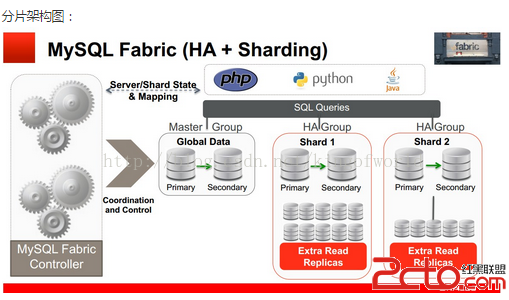
Fabric支持实现高可用性的架构图如下：



Fabric使用HA组实现高可用性，其中一台是主服务器，其他是备份服务器， 备份服务器通过同步复制实现数据冗余。应用程序使用特定的驱动，连接到Fabric 的Connector组件，当主服务器发生故障后，Connector自动升级其中一个备份服务器为主服务器，应用程序无需修改。

Fabric支持可扩展性及负载均衡的架构如下：





使用多个HA 组实现分片，每个组之间分担不同的分片数据（组内的数据是冗余的，这个在高可用性中已经提到）

应用程序只需向connector发送query和insert等语句，Connector通过MasterGroup自动分配这些数据到各个组，或从各个组中组合符合条件的数据，返回给应用程序。

### 产品优点

与MySQL复制功能共同使用实现高可用性时，MySQLFabric能够提供自动故障检测和故障转移功能。具体包括：

* 监控主数据库，如果服务器出现故障，MySQL Fabric将选出一个从数据库，并将其升级为新的主数据库。
* 提供事务处理的自动路径选择以连接到当前主数据库，及从数据库间的查询负载均衡。拓扑及服务器状态对应用透明。

自动数据分片和再分片功能简化了开发运营团队的流程管理工作。这帮助用户能够：

* 对表分片，实现读写的横向扩展；
* 选择对哪些表分片，并指定分片键的字段，包括使用基于哈希映射还是基于区间映射；
* 将现有数据片转移到新的服务器上，或者将这些数据片进一步分成更小的数据片。

Conector帮助读写分离：

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。

### 产品缺点

* 自增长键不能作为分片的键。
* 事务及查询只支持在同一个分片内，事务中更新的数据不能跨分片，查询语句返回的数据也不能跨分片。

## Amoeba

### 功能介绍

Amoeba for MySQL 位于Client、Database Server(s)之间，具有负载均衡、高可用性、sql过滤、可承受高并发、读写分离、Query Route（解析sql query语句，并且根据条件与预先设定的规则，请求到指定的目标数据库。可并发请求多台数据库合并结果）、对客户端透明，能降低数据切分带来的复杂多数据库结构、数据切分规则给应用带来的影响。适用mysql 4.1或者以上版本（mysql 协议版本：10）暂时不支持事务、DDL语句目前只会分配给默认的数据库执行，运行环境至少需要运行 mysql 4.1以上服务, Java 1.5或以上版本。

Amoeba与mysql proxy存在一些区别。在mysql proxy上面如果想要读写分离并且读集群、写集群机器比较多情况下，用mysql proxy 需要自己写一个LUA脚本，目前mysql proxy没有现成的比较好的lua脚本。amoeba只需要进行相关的配置就可以满足需求。

### 产品优点

基于XML的配置文件，用SQLJEP语法编写规则，配置比较简单。

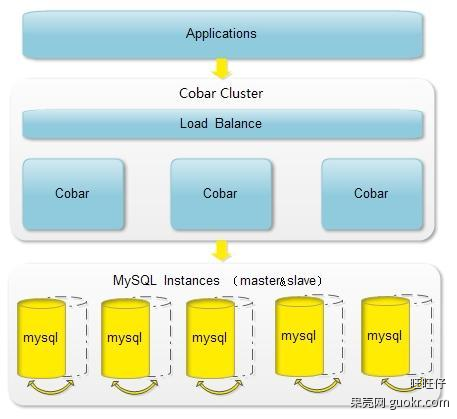
### 产品缺点

* 目前还不支持事务。
* 对返回大数量的查询并不合适。
* 不支持分库分表，只能做到分数据库实例。
* Amoeba需要解析和转发请求的SQL，这期间会造成一定的性能损耗。
* Amoeba不支持跨库join和排序
* Insert语句必须指定表的列名
* Amoeba需要更严格的SQL语句规范

## cobar(阿里巴巴)

### 功能介绍

Cobar是阿里巴巴（B2B）部门开发的一种关系型数据的分布式处理系统，它可以在分布式的环境下看上去像传统数据库一样为您提供海量数据服务。cobar中间件以proxy的形式位于前台应用和实际数据库之间，对前台的开放的接口是mysql通信协议。将前台SQL语句变更并按照数据分布规则转发到合适的后台数据分库，再合并返回结果，模拟单库下的数据库行为。



### 产品优点

1、数据和访问从集中式改变为分布：

（1）Cobar支持将一张表水平拆分成多份分别放入不同的库来实现表的水平拆分

（2）Cobar也支持将不同的表放入不同的库

（3） 多数情况下，用户会将以上两种方式混合使用

2、解决连接数过大的问题。

3、对业务代码侵入性少。

4、提供数据节点的failover，HA：

(1)Cobar的主备切换有两种触发方式，一种是用户手动触发，一种是Cobar的心跳语句检测到异常后自动触发。那么，当心跳检测到主机异常，切换到备机，如果主机恢复了，需要用户手动切回主机工作，Cobar不会在主机恢复时自动切换回主机，除非备机的心跳也返回异常。

(2)Cobar只检查MySQL主备异常，不关心主备之间的数据同步，因此用户需要在使用Cobar之前在MySQL主备上配置双向同步。

### 产品缺点

1) 不支持跨库情况下的join、分页、排序、子查询操作。

2) SET语句执行会被忽略，事务和字符集设置除外。

3) 分库情况下，insert语句必须包含拆分字段列名。

4) 分库情况下，update语句不能更新拆分字段的值。

5) 不支持SAVEPOINT操作。

6) 暂时只支持MySQL数据节点。

7) 使用JDBC时，不支持rewriteBatchedStatements=true参数设置(默认为false)。

8) 使用JDBC时，不支持useServerPrepStmts=true参数设置(默认为false)。

9) 使用JDBC时，BLOB, BINARY, VARBINARY字段不能使用setBlob()或setBinaryStream()方法设置参数。

10) 支持弱一致性事务（多库并行执行/提交）。

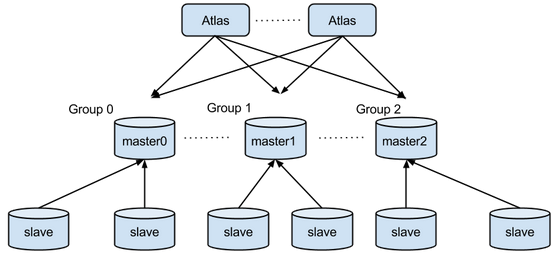
11)主异常时切换到从后主恢复，没有 failback，只能手动切换回主

## Atlas(360)

### 功能介绍

Atlas是由 Qihoo 360, Web平台部基础架构团队开发维护的一个基于MySQL协议的数据中间层项目。它是在mysql-proxy 0.8.2版本的基础上，对其进行了优化，增加了一些新的功能特性。360内部使用Atlas运行的mysql业务，每天承载的读写请求数达几十亿条。

Atlas是一个位于应用程序与MySQL之间，它实现了MySQL的客户端与服务端协议，作为服务端与应用程序通讯，同时作为客户端与MySQL通讯。它对应用程序屏蔽了DB的细节，同时为了降低MySQL负担，它还维护了连接池。



Atlas+Keepalived

### 产品优点

1、主库宕机不影响读

主库宕机，Atlas自动将宕机的主库摘除，写操作会失败，读操作不受影响。从库宕机，Atlas自动将宕机的从库摘除，对应用没有影响。在mysql官方的proxy中主库宕机，从库亦不可用。

2、通过管理接口，简化管理工作，DB的上下线对应用完全透明，同时可以手动上下线。

3、实现读写分离

（1）为了解决读写分离存在写完马上就想读而这时可能存在主从同步延迟的情况，Altas中可以在SQL语句前增加 /\*master\*/ 就可以将读请求强制发往主库。

（2）主库可设置多项，用逗号分隔，从库可设置多项和权重，达到负载均衡。

4.、实现分表

（1）需带有分表字段。

（2）支持SELECT、INSERT、UPDATE、DELETE、REPLACE语句。

（3）支持多个子表查询结果的合并和排序。

5、之前官方主要功能逻辑由使用lua脚本编写，效率低，Atlas用C改写，QPS提高，latency降低。

6、安全方面的提升

（1）通过配置文件中的pwds参数进行连接Atlas的用户的权限控制。

（2）通过client-ips参数对有权限连接Atlas的ip进行过滤。

（3）日志中记录所有通过Altas处理的SQL语句，包括客户端IP、实际执行该语句的DB、执行成功与否、执行所耗费的时间。

7、平滑重启

通过配置文件中设置lvs-ips参数实现平滑重启功能，否则重启Altas的瞬间那些SQL请求都会失败。该参数前面挂接的lvs的物理网卡的ip，注意不是虚ip。平滑重启的条件是至少有两台配置相同的Atlas且挂在lvs之后。

### 产品缺点

Atlas的分表功能，不能实现分布式分表，所有的子表必须在同一台DB的同一个database里且所有的子表必须事先建好，Atlas没有自动建表的功能。

Atlas sharding只对sql语句提供有限的支持, 目前支持基本的Select, insert/replace, delete, update语句, 支持全部的Where语法(SQL-92标准), 不支持DDL(create drop alter)以及一些管理语句, DDL请直连MYSQL执行, 请只在Atlas上执行Select, insert, delete, update(CRUD)语句, 对于以下语句, 如果语句命中了多台dbgroup, Atlas均未做支持(如果语句只命中了一个dbgroup, 如select count(\*) from test where id < 1000, 其中dbgroup0范围是0 - 1000, 那么这些特性都是支持的)

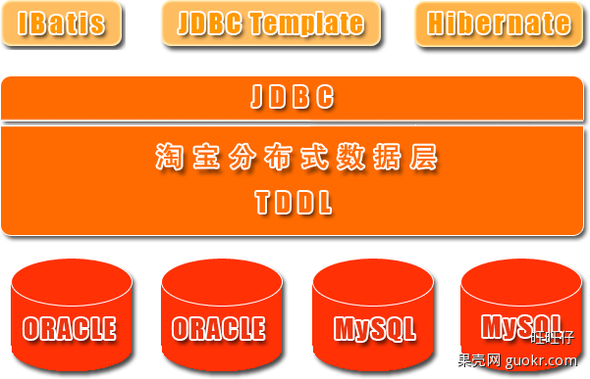
* Limit Offset(支持Limit)
* Order by
* Group by
* Join
* ON
* Count, Max, Min等函数

请不要在Sharding的表上使用这些特性, 如果对这种特性有需求请不要让此表sharding。

事务在Atlas的非sharding的表是完全支持的, 但是对于sharding的表, Atlas只能提供部分的支持(不支持跨dbgroup的事务)。

## TDDL(阿里巴巴)

淘宝根据自身业务需求研发了 TDDL （ Taobao Distributed Data Layer ）框架， 主要用于解决 分库分表场景下的访问路由（持久层与数据访问层的配合）以及异构数据库之间的数据同步 ，它是一个基于集中式配置的 JDBC DataSource 实现，具有分库分表、 Master/Salve 、动态数据源配置等功能。



淘宝人自定的 TDDL 优点：

**1 、数据库主备和动态切换；**

**2 、带权重的读写分离；**

**3 、单线程读重试；**

**4 、集中式数据源信息管理和动态变更；**

**5 、剥离的稳定 jboss 数据源；**

**6 、支持 mysql 和 oracle 数据库；**

**7 、基于 jdbc 规范，很容易扩展支持实现 jdbc 规范的数据源；**

**8 、无 server,client-jar 形式存在，应用直连数据库；**

**9 、读写次数 , 并发度流程控制，动态变更；**

**10 、可分析的日志打印 , 日志流控，动态变更；**

**注意**：

### 产品优点

* **数据库主备和动态切换；**
* **带权重的读写分离；**
* **单线程读重试；**
* **集中式数据源信息管理和动态变更；**
* **剥离的稳定 jboss 数据源；**
* **支持 mysql 和 oracle 数据库；**
* **基于 jdbc 规范，很容易扩展支持实现 jdbc 规范的数据源；**
* **无 server,client-jar 形式存在，应用直连数据库；**
* **读写次数 , 并发度流程控制，动态变更；**
* **可分析的日志打印 , 日志流控，动态变更；**

### 产品缺点

* TDDL 必须要依赖 diamond 配置中心（ diamond 是淘宝内部使用的一个管理持久配置的系统，目前淘宝内部绝大多数系统的配置）。
* 淘宝开源网站源码不是全部开放，后续支持很慢。需要专门的团队开发和维护代码。