# 背景

随着网站业务的扩展、数据不断增加、用户也越来越多，单台数据库的压力也就越来越大，只通过数据库参数调整或者SQL优化基本已无法满足要求，这时可以采用读/写分离的策略来改变现状。

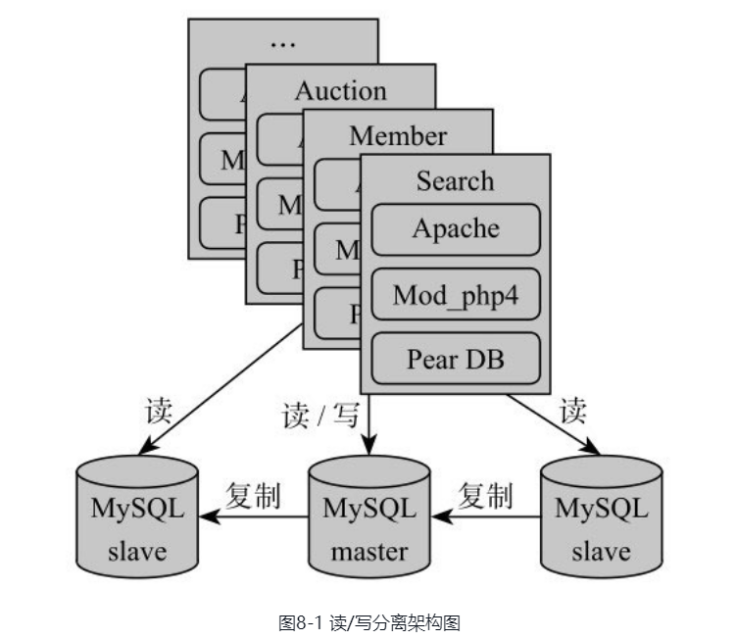
数据库层面通常采用的读/写分离技术为一个master数据库（以下简称master库），多个slave数据库（以下简称slave库）。master库负责数据更新和实时数据查询，slave库负责非实时数据查询。在实际的应用中，因为数据库都是读多写少（读取数据的频率高，更新数据的频率相对较少），而读取数据通常耗时比较长，占用的数据库服务器CPU较多，因此也会影响用户体验。对此通常的解决办法就是把查询从主库中抽取出来，采用多个从库，使用负载均衡，减轻每个从库的查询压力。

读写分离指的是，通过增加一些节点，扩展读的能力。这些节点可以是主节点的全部内容的副本或部分内容的副本，也可以是缓存产品。读写分离一般配合负载均衡产品一起使用。

注：要根据实际的业务需求，采取不同的读写分离策略，我们采用添加hint信息的方式，将需要实时查询的SQL发送到主节点，其余的根据代价模型计算分配到从节点。

# 原理

读/写分离的基本原理是：让master库处理事务增、删、改操作（INSERT、DELETE、UPDATE），而让slave库处理SELECT查询操作，replication数据库负责把数据变更同步到集群的slave库中。



采用读/写分离技术的目标是：既有效减轻master库的压力，又可以把用户查询数据的请求分发到不同的slave库，从而保证系统的健壮性。

# 实现方式

实现读/写分离有两种方式，第一种是通过客户端方式实现，比如PHP Yii框架，第二种是通过Proxy解析SQL的方式实现。通过Yii框架实现读/写分离非常简单，只需要在配置文件中写几个配置参数即可。

## Yii框架

首先，配置db.php文件，如图8-2所示。



在PHP实现读/写分离的过程中，要考虑如下几个问题。

❑ 在主从架构中，如果主从延时、主从数据不一致，怎么办？如果有延迟，能否不把延迟（N秒）的请求转发给这台slave？

❑ 如果是一主多从，那么从库的load balance负载均衡如何实现？当某台slave宕机时，能否不把请求转发给这台slave？当所有的slave不可用时，如何把请求转发给master？

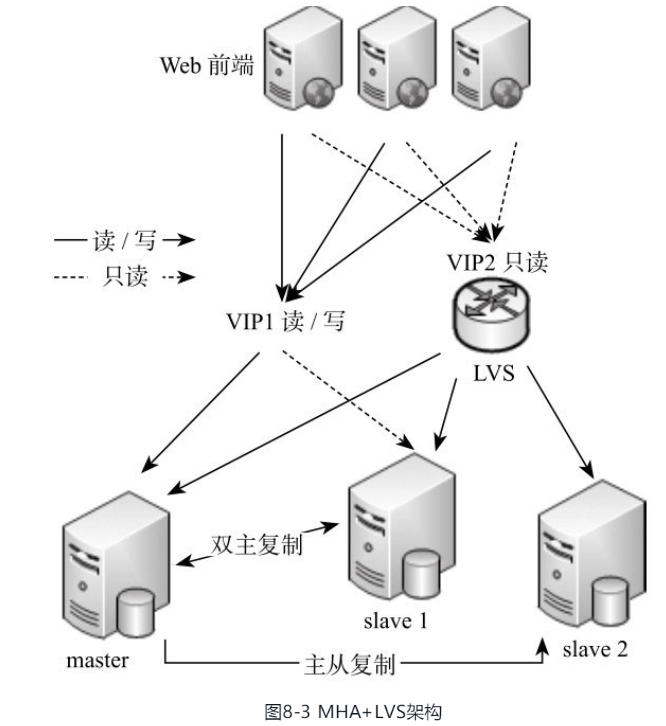
采用PHP Yii框架实现读/写分离时，是需要考虑以上问题的，且需要借助第三方负载均衡软件HAProxy解决这些问题。HAProxy提供了高可用性、负载均衡以及基于TCP和HTTP应用的代理，它支持虚拟主机，是一种免费、快速并且可靠的解决方案。对于那些负载特大的Web站点来说，HAProxy特别适用。这些站点通常需要实现会话保持或七层处理，HAProxy运行在当前的硬件上，完全可以支持数以万计的并发连接。

HAProxy具体以下特性：

1）免费开源，稳定性非常好。

2）根据官方文档，HAProxy使用Myricom厂商的万兆网卡可以将10Gb/s的网络带宽跑满，这个数值作为软件级负载均衡器是相当惊人的。

**大多数公司的架构是通过客户端方式实现的：一个主库，多个从库。主库负责写，从库负责查询，主库的高可用性通过MHA（Master High Availability）实现，从库读的负载均衡通过LVS或者HAProxy实现**。



通过HAProxy代理，基于connect方式，以及自定义脚本，可以解决slave延迟或宕机故障转移。

## Proxy

### MySQL Proxy

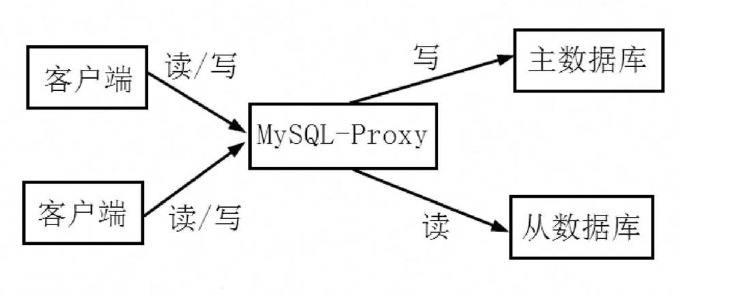
#### 概述

早前，甲骨文公司官方提供了MySQL Proxy，MySQL Proxy是一个位于客户端和MySQL服务器端之间的程序，通过它可以实现监听和管理客户端与MySQL服务器端之间的通信，最大的作用是实现数据库的读写分离，从而达到负载均衡的目的。

#### 原理

MySQL Proxy的常用用途包括负载平衡、故障分析、查询分析、查询过滤和修改等。作为一个中间层代理，通俗地说，它就是**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。对于应用程序来说，MySQL Proxy是完全透明的，应用程序只需要连接到MySQL Proxy的监听端口即可。

MySQL Proxy最强大的功能是实现“读写分离”，基本原理是让主数据库处理事务性查询，让从数据库处理SELECT查询，最后通过数据库的复制功能把事务性查询导致的数据变更同步到集群中的从数据库中。MySQL Proxy实现读写分离的过程：



提示：

如果此时只有一个MySQL Proxy机器，可能会出现单点失效的问题，解决办法是使用多个proxy机器作为冗余，在应用服务器的连接池配置中配置多个proxy的连接参数即可。

MySQL Proxy通过mysql-proxy来指定配置的参数。下面了解一下该命令常用参数的含义：

●　--proxy-backend-addresses：该参数用来指定MySQL服务器的IP地址和端口号，如果代理多个服务器，可以用逗号分隔。

●　--proxy-read-only-backend-addresses：该参数用来指定只读服务器的IP地址和端口号，如果代理多个服务器，可以用逗号分隔。

●　--proxy-skip-profiling：该参数用来设置是否禁用查询性能分析。

●　--proxy-lua-script：该参数用来指定lua脚本文件。

●　--daemon：采用daemon方式启动。

●　--admin-address：指定MySQL Proxy的管理端口。

●　--proxy-address=：指定mMySQL Proxy的监听端口。也可以通过mysql-proxy --help-all查看完整的参数含义。

### MaxScale

#### 概述

但由于近几年MySQL Proxy一直没有正式版本，所以无法用在生产上。然而，MariaDB于2015年1月14日宣布其旗下的MaxScale发布GA版本。

#### 原理

MaxScale使用C语言开发，利用Linux下的异步I/O功能，使用epoll作为事件驱动框架。它是MariaDB开发的一个数据库智能代理服务，允许根据数据库SQL语句将请求路由到多个服务器，且可设定各种复杂的转向规则。MaxScale可用于透明地提供数据库的负载均衡和高可用性，同时也可提供高度可伸缩和灵活的架构，支持不同的协议和路由决策。

MaxScale有两种方式实现读/写分离。一种是基于**connect**的，类似于HAProxy，不解析SQL语句，可以通过PHP Yii框架或Java Mybatis框架实现。在此方式中，用MaxScale做多台slave的负载均衡，并且支持主从同步延迟检测功能。

另一种是基于**statement**的，要解析SQL语句。在这种方式里，前端程序不需要修改，通过MaxScale对SQL语句进行解析，把读/写请求自动路由到后端数据库节点上，从而实现读/写分离。

**商业软件OneProxy中间件也是基于statement方式实现读/写分离的**。这种方式的好处是不修改程序代码，减少了复杂度，可平滑迁移，无感知；缺点是解析SQL势必会增加CPU的性能损耗，性能没有基于connect的方式好。

#### 测试

##### 基于connect方式的测试

##### 基于statement方式（SQL解析）的测试

# Oneproxy

## 背景

假设你的业务突飞猛进，现在每天有上千万的页面访问量或会员登录，这时若还只用一台机器来提供会员信息的服务可能会顶不住或运行极不稳定。由于会员信息具有以读为主的业务特征，因此可以考虑复制多份会员信息，将读流量分担出去，从而进行架构的横向扩展，以保证业务的稳定。如果希望对应用完全透明，那么后端的读节点可以根据需要进行透明伸缩。

OneProxy可在自动单点切换的功能基础上进行扩展，实现对应用透明的读/写分离功能。它所具有的内置故障检测机制可以在1秒钟内发现后端故障，并主动踢除出问题的数据库，无须向应用推送后端数据库的运行状态，以实现轻松运行维护。目前已在Zabbix、PHP、PHPWind、Discuz和JavaJDBC上验证过，越来越多的用户正在选择用OneProxy来保障以读为主的互联网业务。

## 概述

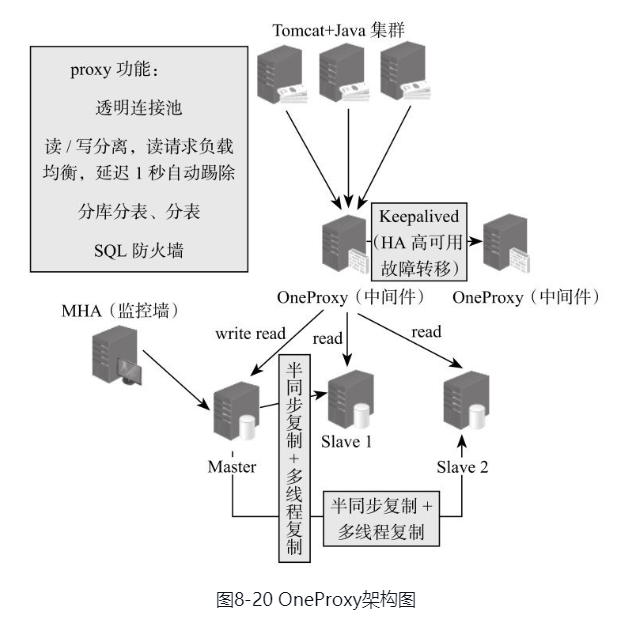
OneProxy平民软件是完全自主开发的分布式数据访问层，可帮助用户在MySQL/PostgreSQL集群上快速搭建支持分库分表的分布式数据库中间件，也是一款具有SQL白名单（防SQL注入）及IP白名单功能的SQL防火墙软件。它采用与MySQL Proxy一致的反向协议输出模式，应用非常简单且透明，这让用户畏惧的分库分表（Horizontal Partitioning）工作变得极其简单可控！它基于Libevent机制实现，单个实例可以实现25万个的SQL转发能力，用一个OneProxy节点可以带动整个MySQL集群。

## 功能

从信息服务的角度来看，数据库层对上的服务有三种行为：读取、存入和计算，比如页面显示主要是读取，交易处理主要是写入，而报表汇总则是计算。基本的信息业务都可以分解为由这三种行为的一定比例构成。对任何一种行为来讲，都需要考虑服务能力的扩展（Capacity）及服务的可靠性（Availability），比如在京东、淘宝、支付宝等网站上购买东西时，可能需要先登录，登录过程其实是一个信息的读取行为，考虑到他们的用户量，一定要求做到系统可以随时扩容，并且当一台物理机器发生故障时，服务能不受影响。对于以读取信息为主的服务，可以考虑将数据复制多份，并且当用户登录时，可以从任何一份复制的数据里进行检索，这样既扩展了系统的信息读取能力，也不会让任何一份数据复制失效从而产生大的影响。

数据库本身都提供了数据复制的便捷手段，比如Oracle的Active DataGuard、 MySQL的Replication等，为了简化事务管理，数据库的复制节点通常是只读的，称为从节点或备节点。而用于写入的那个原始节点则称为主节点，主节点和从节点之间构成一套主从节点集群或主备节点集群。由于从节点或备节点只能读取不能写入，所以要扩展读能力，则先要做读/写分离，要实现该功能首先要梳理和修改应用代码，进行详尽的线下测试，再进行仔细的应用发布，然后修改应用程序对读取操作的数据源管理和选择方式，这可能是一个非常复杂的软件项目。

当后台用的是MySQL数据库或兼容MySQL协议的数据库时，就可以不用修改应用程序，使用OneProxy透明地实现读/写分离功能，因为OneProxy可以识别MySQL协议，可以清晰地识别协议包中的SQL语句，可对不同类型的语句进行透明地转发处理，自然也就可以实现对应用透明的读/写分离了，完全不需要启动前面所说的复杂的软件项目，架构如图8-20所示。



## 环境搭建

# Atlas

参考：《跟老男孩学Linux运维：MySQL入门与提高实践》

# 应用

对于读多写少（非更新查询为主）的负载，特别适合做读写分离。需要留意的是，要保证用户感知到自己所做的变更有效即可，用户在很多情况下并不需要知道其他用户的改变。如果用户对于数据的一致性要求在某个时刻很高，那么这部分数据，建议不要使用读写分离，MySQL的复制可能会出现延时，无法满足业务的需要。你可以采取变通的方式，比如，在用户修改了内容之后，临时强制用户访问主节点，以获取一致性的数据，在过一段时间之后，再让用户访问副本的数据，一般在此时，副本的数据已经同步到最新状态了。

读写分离往往和负载均衡技术配合使用。负载均衡软硬件产品有F5、Haproxy及一些自己设计的MySQL Proxy代理等，负载均衡可以更高效地利用硬件，你可以设置权重，分配更多的流量给性能更好的机器，负载均衡产品一般还有故障检测、自动冗余切换等功能，这可以大大提高机器的可用性。

读写分离技术的一个难点在于延时的影响，你需要有一个手段来确保你没有读取到太旧的数据，写操作和一些不能容忍延时的查询，需要指向主库。对于数据延时敏感度不高的数据，你需要定义延时的阈值，通过自动或手工的方式处理延时数据对于用户体验的影响。你可以通过监控SHOW SLAVE STATUS里的输出Seconds\_behind\_master的方式判断是否有延时，但这种方式不太可靠。监控复制滞后（replication lag）更稳健的方式是通过心跳表的方式。我们很难确保MySQL的延时，因为网络波动、复制异常、性能问题等都可能导致复制中断，而往往需要人工来进行干预，毕竟有能力开发专用的Proxy代理的公司很少，所以，不建议使用读写分离，采用读写分离一般是基于一个前提，主库已经出现了读瓶颈，如果出现了读瓶颈，那么使用缓存一般是更有效、更成熟的解决方案。

由于没有好的读写分离的方案，如果你一定要使用读写分离，那么推荐应用程序自身实现读写分离，把读的流量指向负载均衡产品或Proxy代理。