# 背景

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。

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。

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

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？

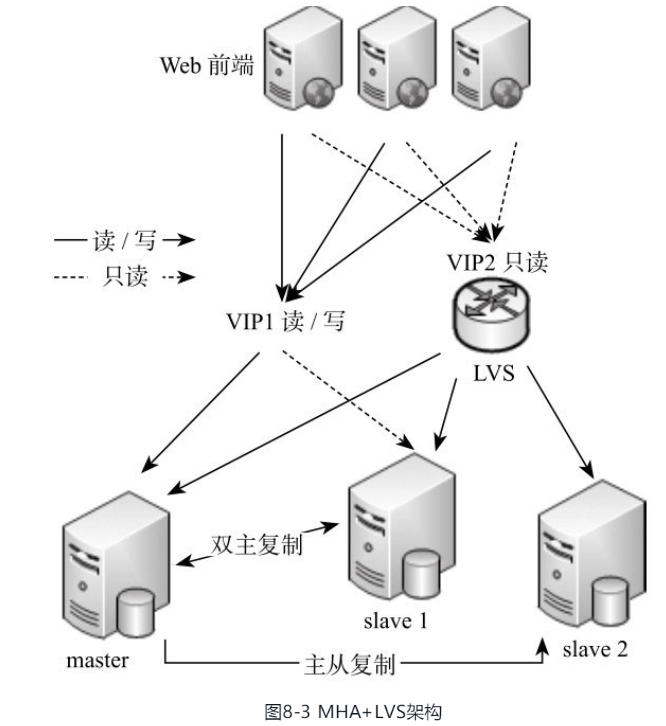
采用PHP Yii框架实现读/写分离时，是需要考虑以上问题的，且需要借助第三方负载均衡软件HAProxy解决这些问题。HAProxy提供了高可用性、负载均衡以及基于TCP和HTTP应用的代理，它支持虚拟主机，是一种免费、快速并且可靠的解决方案。对于那些负载特大的Web站点来说，HAProxy特别适用。这些站点通常需要实现会话保持或七层处理，HAProxy运行在当前的硬件上，完全可以支持数以万计的并发连接。

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

早前，甲骨文公司官方提供了MySQL Proxy，但由于近几年一直没有正式版本，所以无法用在生产上，如图8-4所示。然而，MariaDB于2015年1月14日宣布其旗下的MaxScale发布GA版本。

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。

# 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。