新浪

SACC 2010系统架构师大会

基于MySQL的可扩展架构设计

杨海朝

E-mail: jackbillow@gmail.com

You are the One

个人简介

新浪高级DBA

整个新浪数据库运维工作...

超过700台数据库机器...

每天查询量超过50亿...

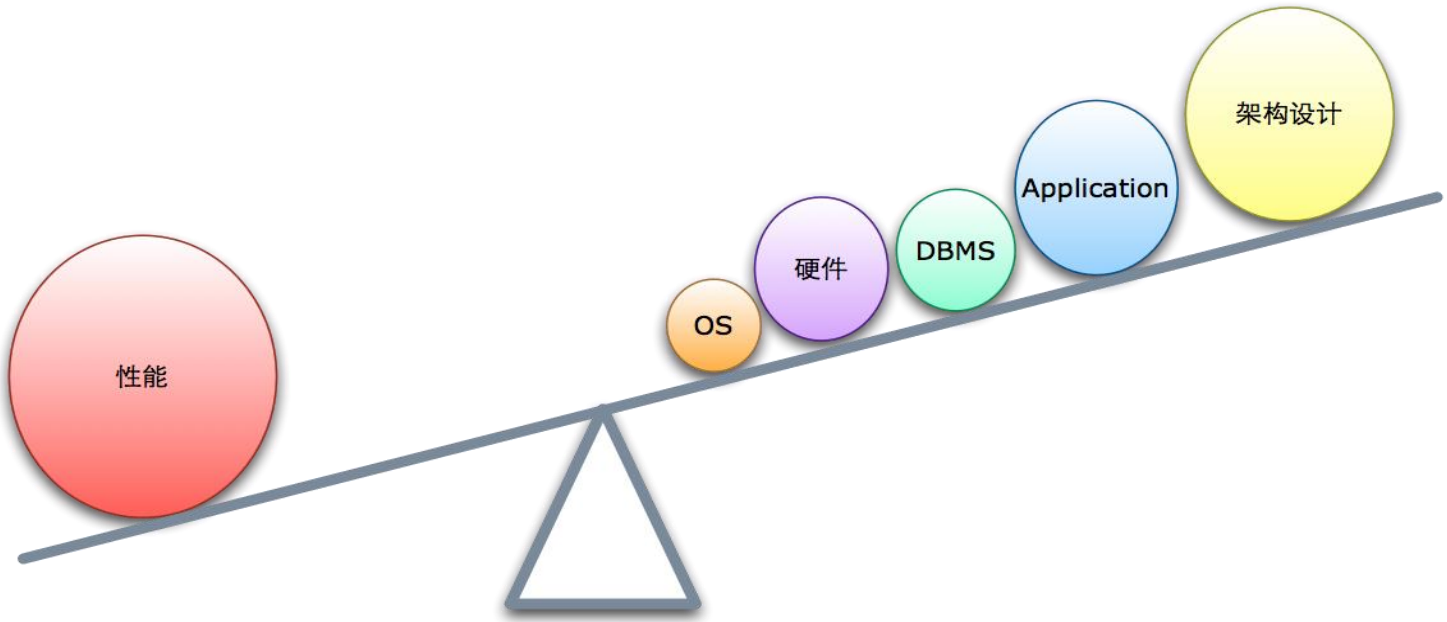
单个项目每天的查询量超过39亿...

鱼缸的启示

可扩展的重要性

1. 非功能性需求0级别
2. scale-up & scale-out
3. 后期改造成本高
4. 价格性能曲线

影响性能因素



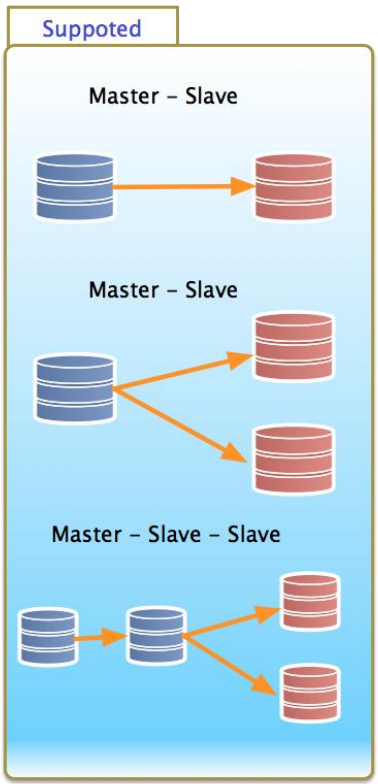
议题

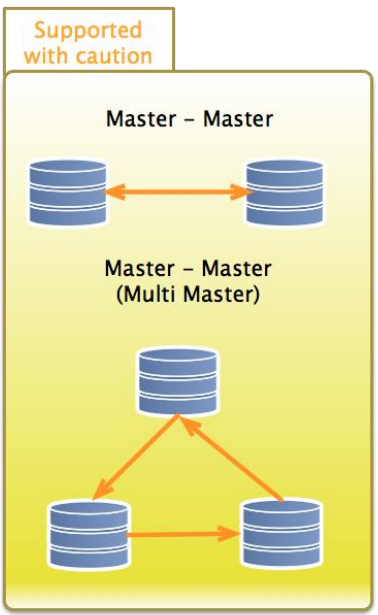
1. 复制结构
2. 基于Sharding策略的结构设计
3. 基于HA的scale-out设计
4. 跨IDC的scale-out设计实例
5. Q&A

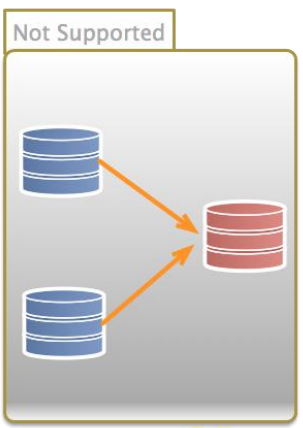
议题

1. 复制结构
2. 基于Sharding策略的结构设计
3. 基于HA的scale-out设计
4. 跨IDC的scale-out设计实例
5. Q&A

MySQL Replication结构



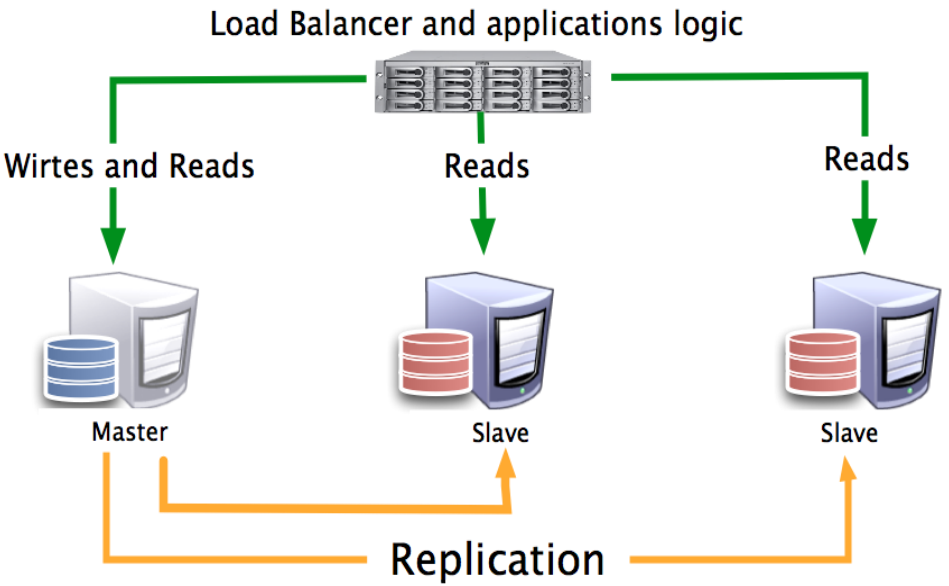




replication

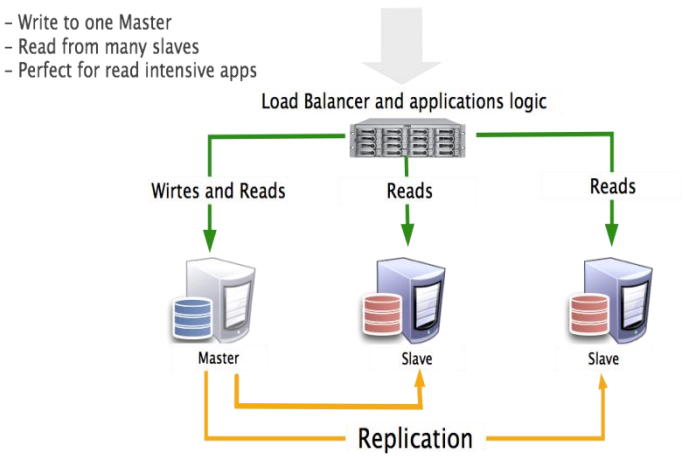
1 Master--N Slave

- Write to one Master
- Read from many slaves
- Perfect for read intensive apps

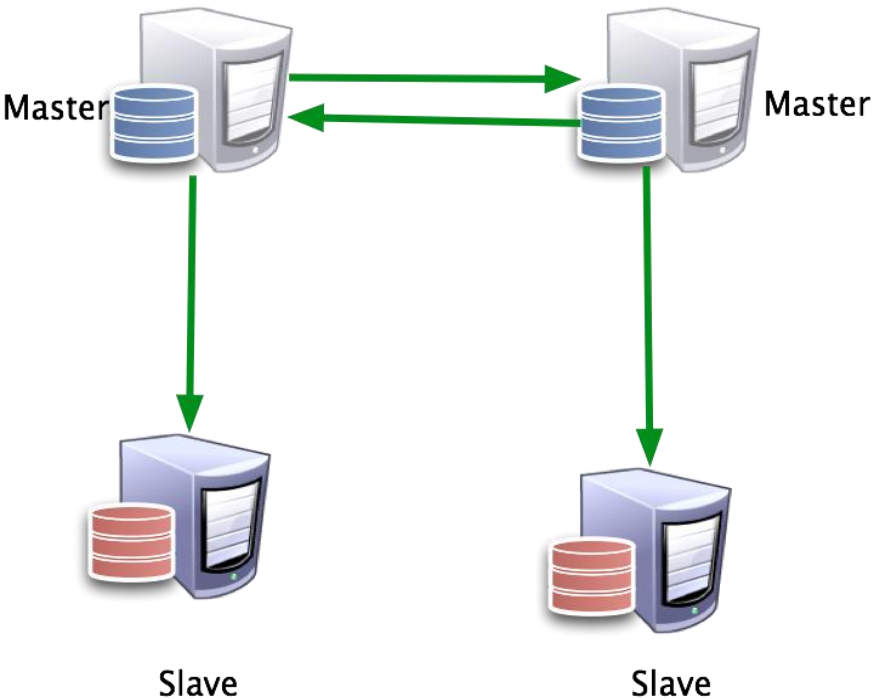


1 Master--N Slave

1. 最常用的方案
2. web2.0中小型站点选择
3. 读密集型应用

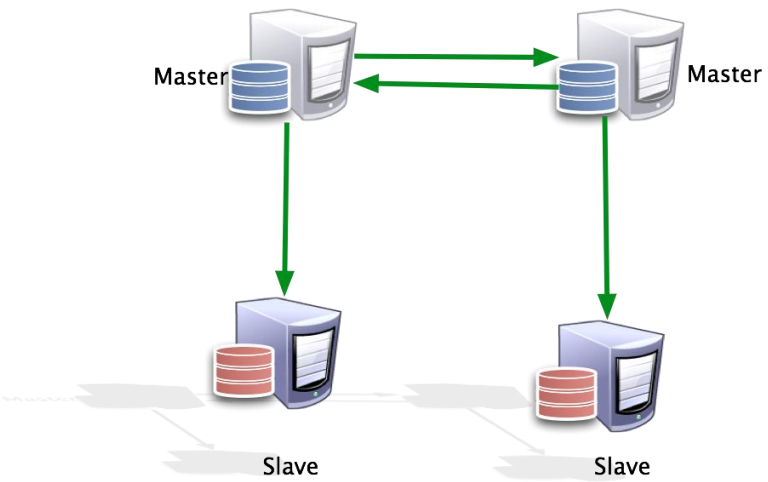


2 Master--N Slave

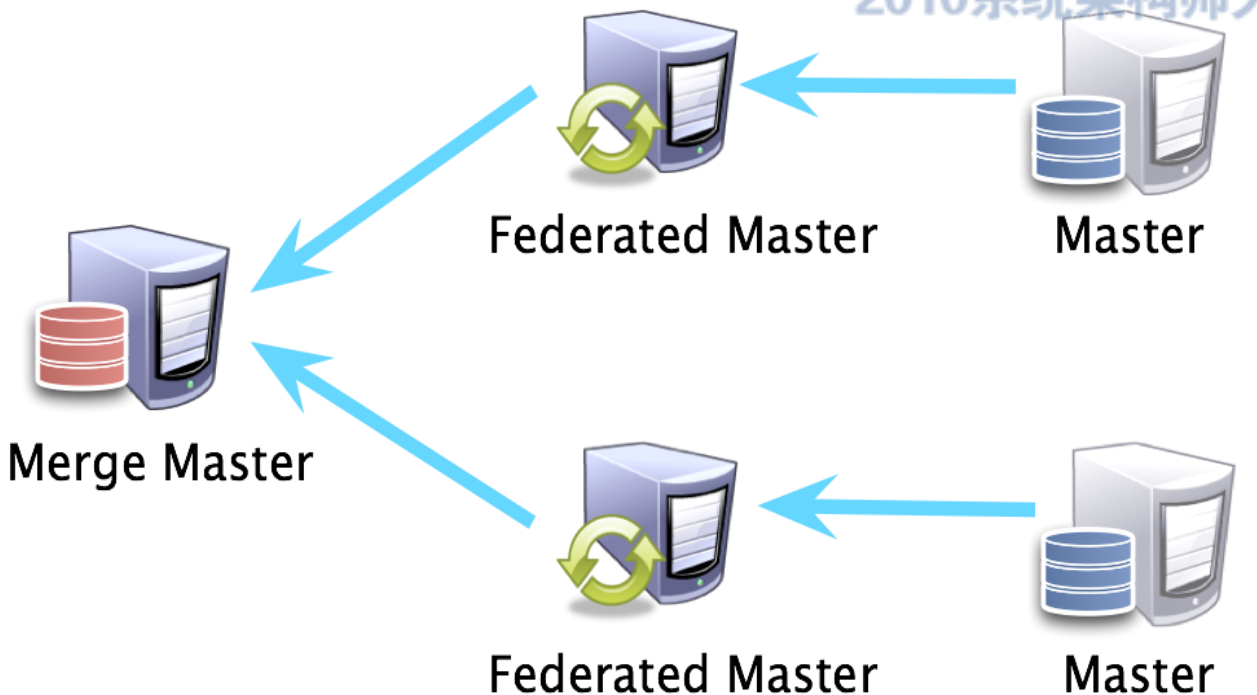


2 Master--N Slave

1. AP or AA
2. 自增问题
3. 更新丢失
4. HA功能

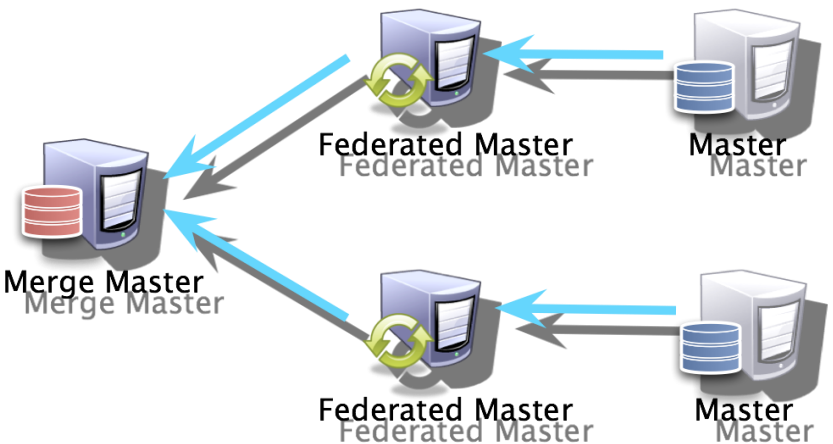


1 Slave--2 Master

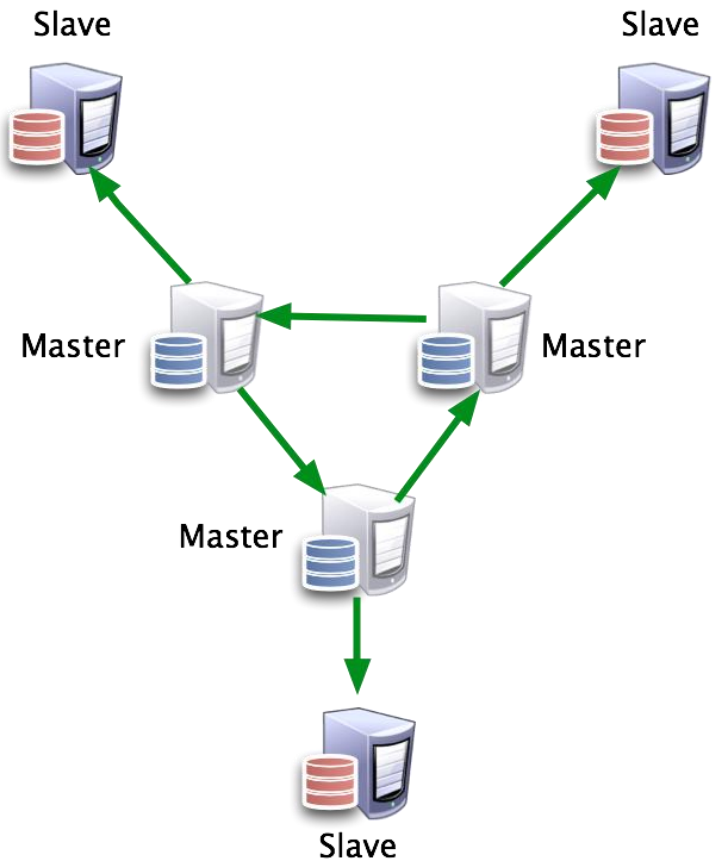


1 Slave--2 Master

1. Slave可能成为瓶颈
2. 不支持事务

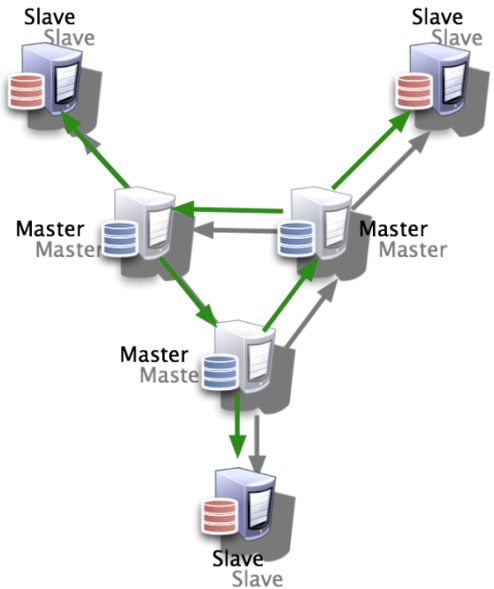


N Master--N Slave



N Master--N Slave

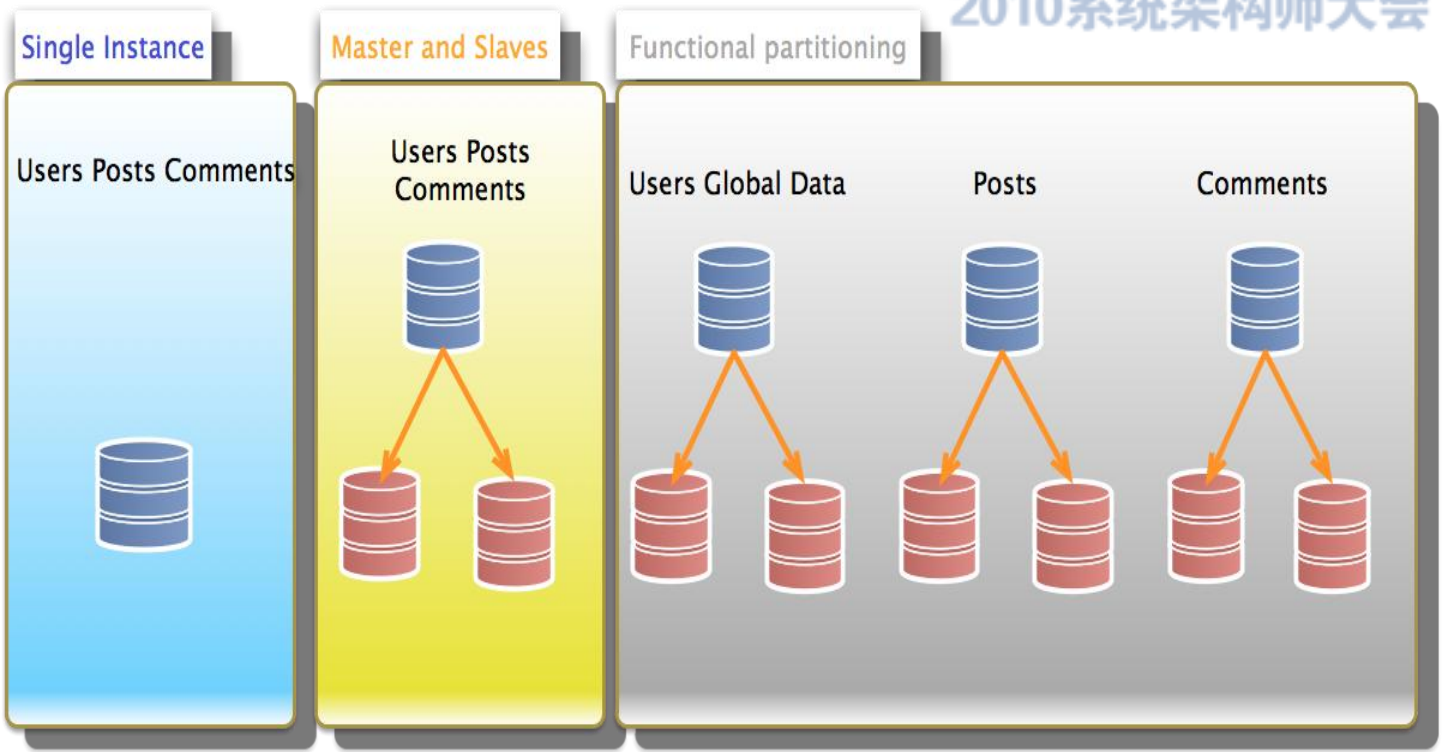
1. 自增问题
2. 更新丢失
3. 故障恢复



议题

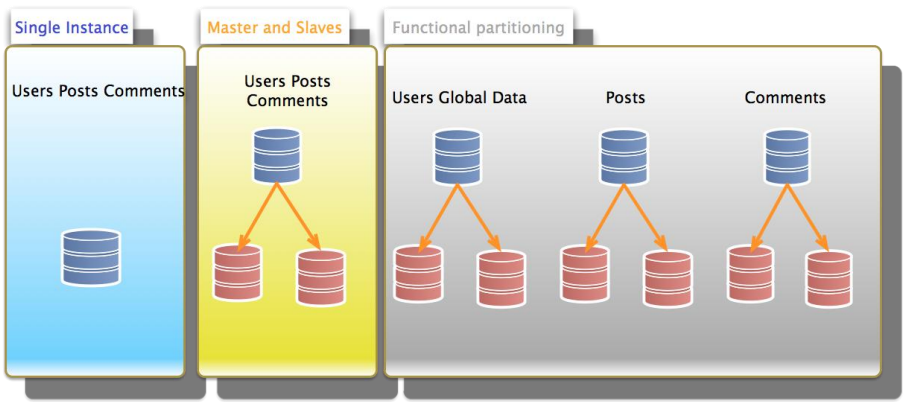
1. 复制结构
2. **基于Sharding策略的结构设计**
3. 基于HA的scale-out设计
4. 跨IDC的scale-out设计实例
5. Q&A

垂直拆分（功能拆分）

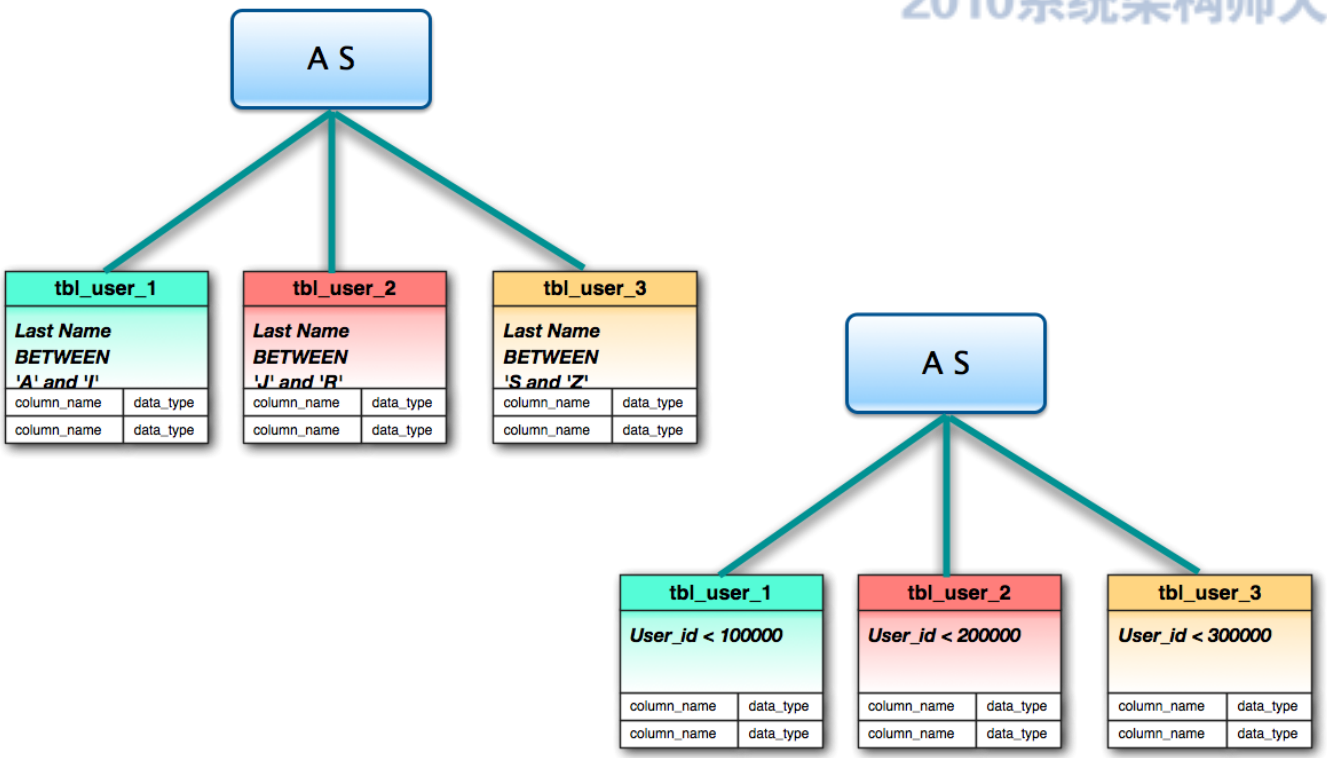


垂直拆分（功能拆分）

1. 明确业务的功能需求
2. 逐步去拆分
3. 需要和水平拆分结合

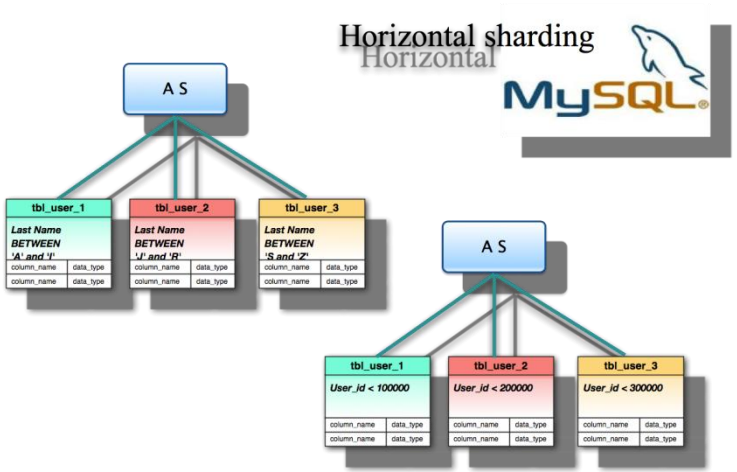


水平拆分--1



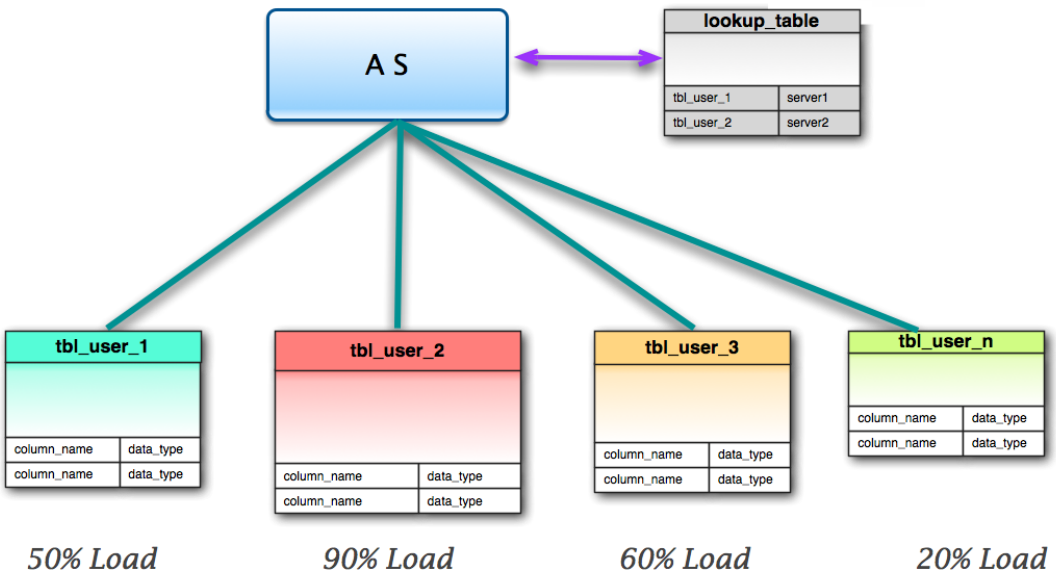
水平拆分--1

1.选择partitioning key至关重要
2. 拆分粒度大小指定
3. 考虑后期搬移策略



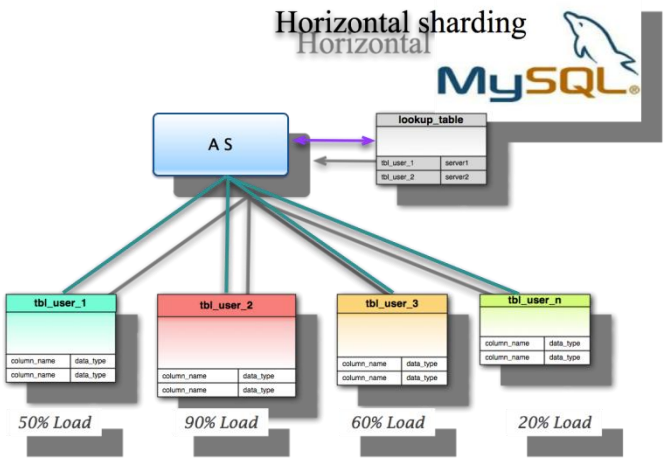
水平拆分--2

Horizontal sharding



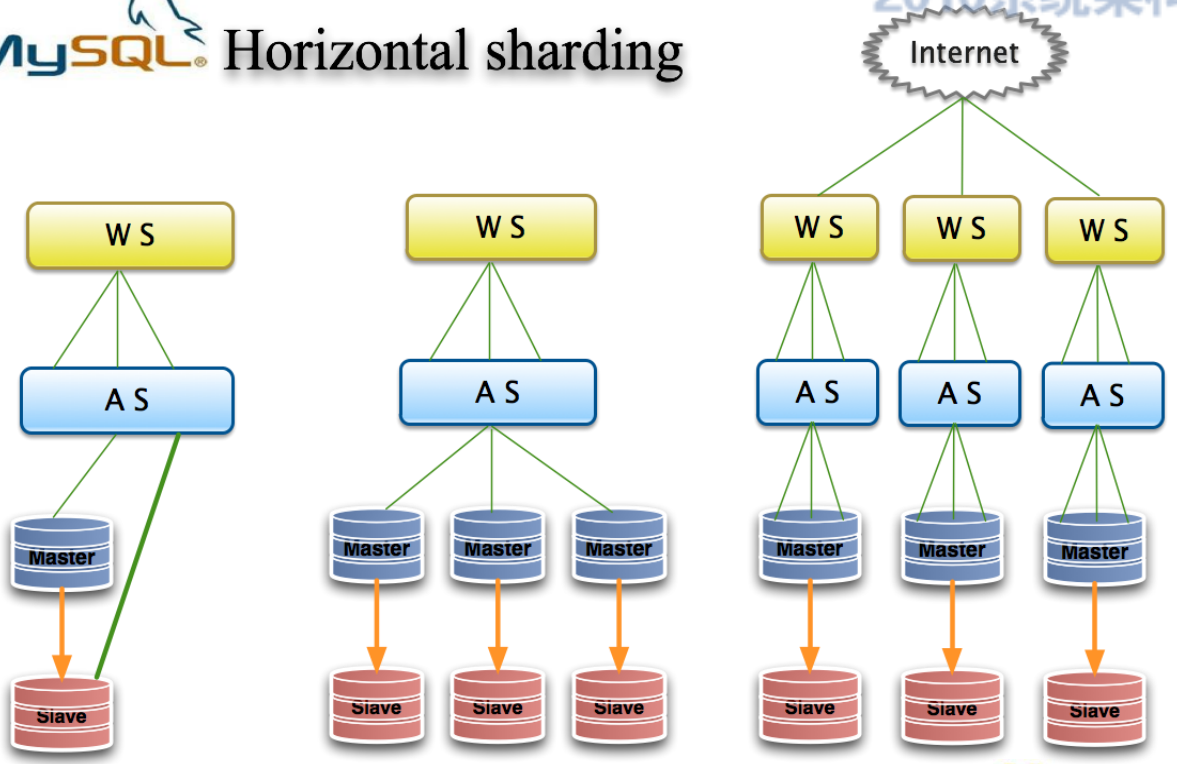
水平拆分--2

1. 选择partitioning key至关重要
2. 多一次查询消耗，索引可能成为瓶颈
3. 弹性的LB



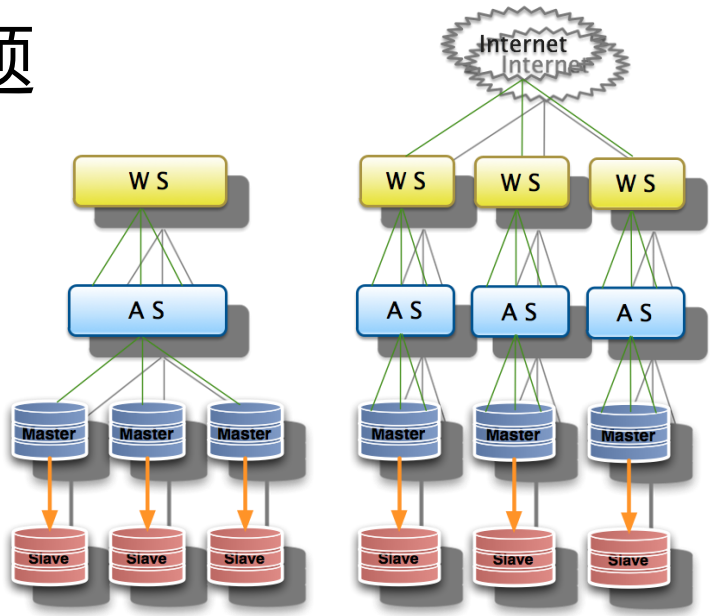
水平拆分--3

MySQL Horizontal sharding



水平拆分--3

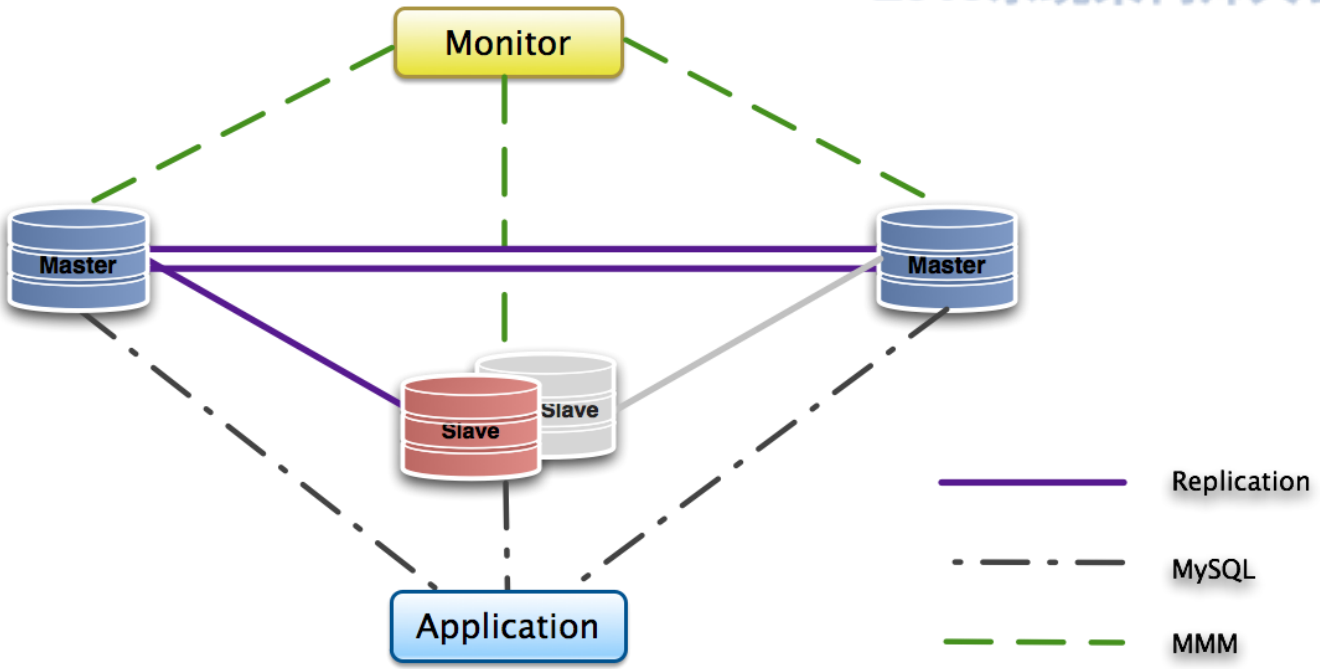
1. 选择partitioning key至关重要
2. 冷热数据区分不好
3. 存在进一步扩展问题



议题

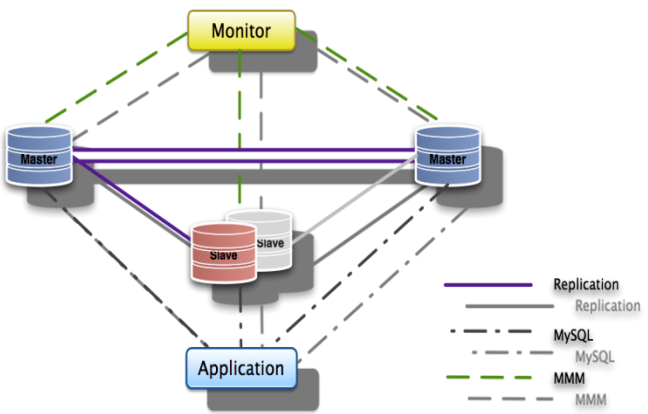
1. 复制结构
2. 基于Sharding策略的结构设计
3. **基于HA的scale-out设计**
4. 跨IDC的scale-out设计实例
5. Q&A

MMM方式

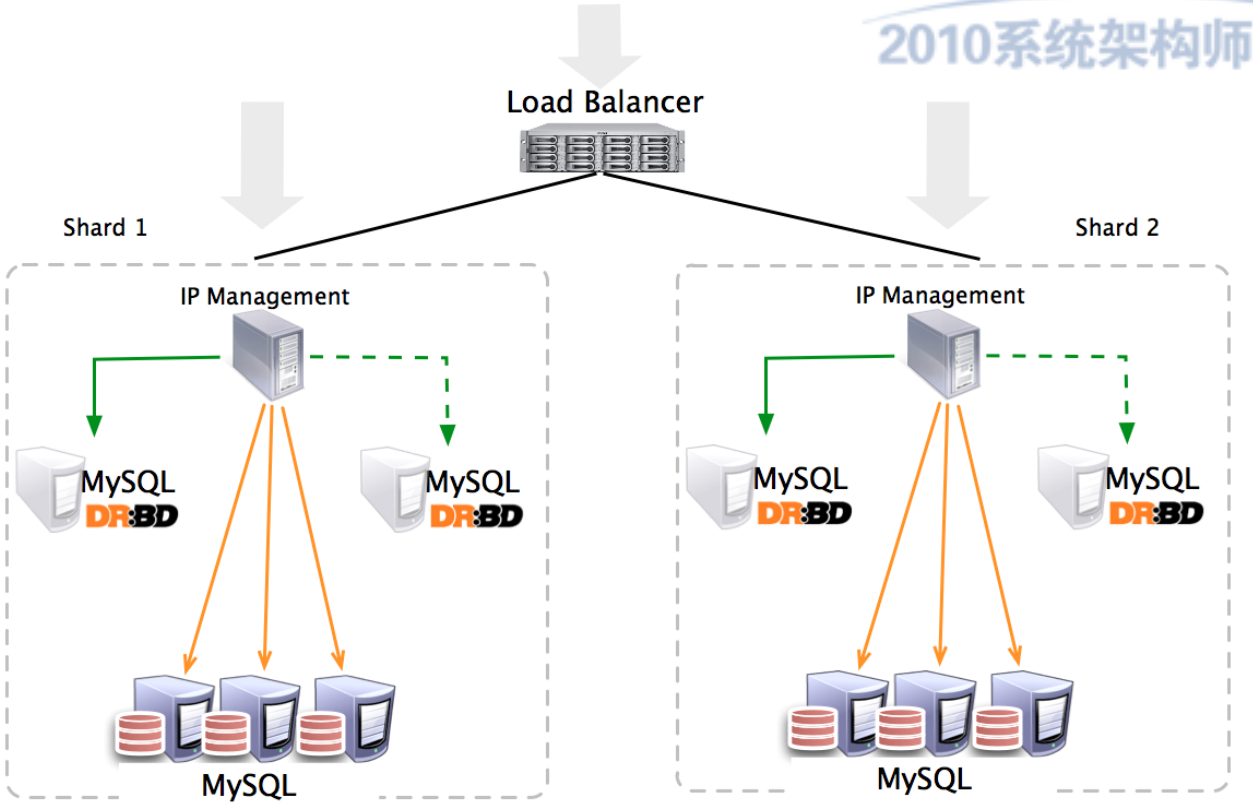


MMM方式

1. 实现不了多IDC容灾
2. 监控是OSPF
3. 不是真正的LB
4. 默认不支持自动切换
5. 浪费IP资源
6. 非无缝的切换方式

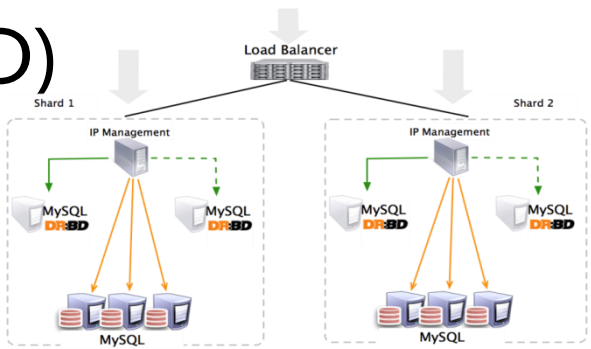


Heartbeat+DRBD方式

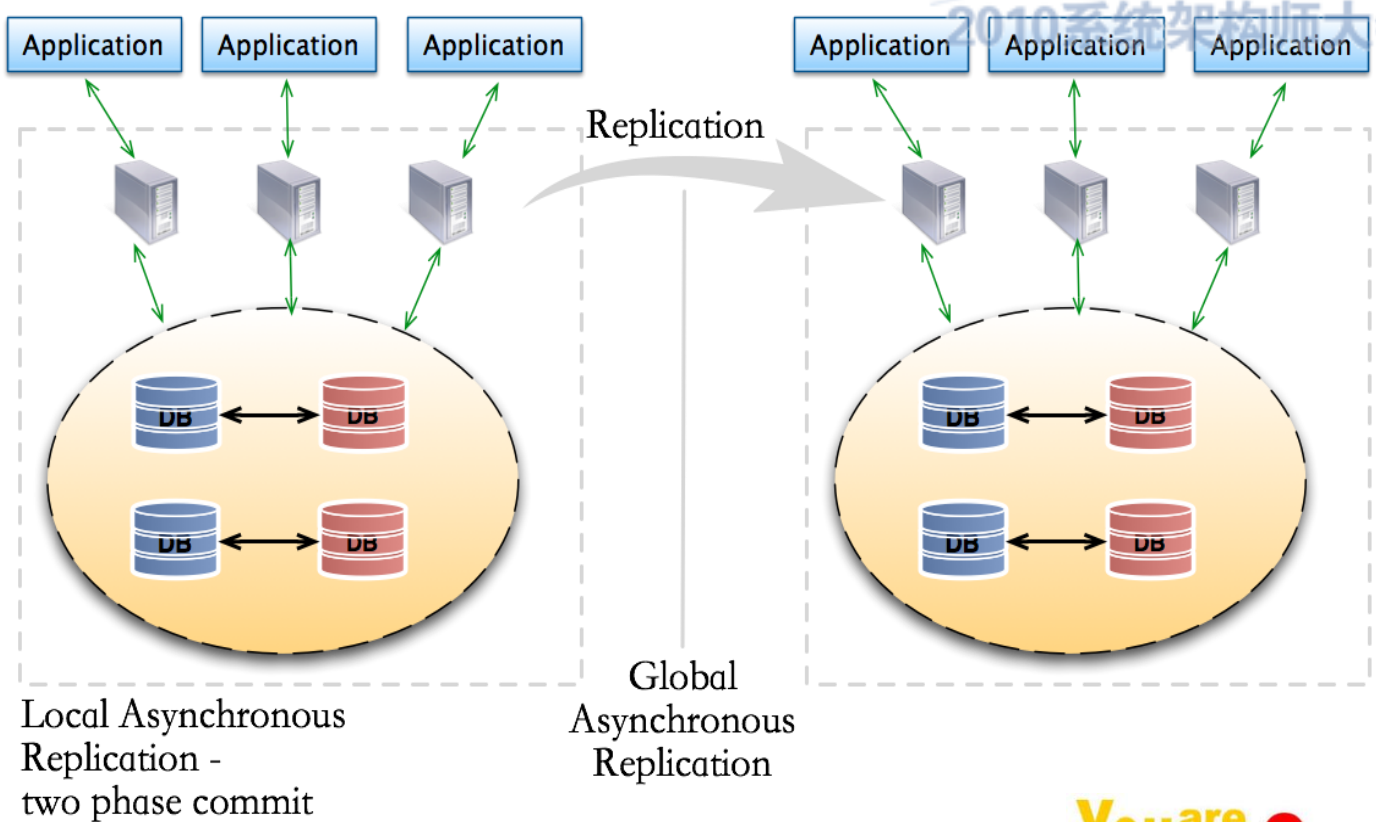


Heartbeat+DRBD方式

1. 多IDC容灾问题
2. 仅仅是主库的HA
3. 性能还有一定的损失
4. 切换时间无法达到毫秒级
5. 资源浪费
6. 维护复杂(裂脑，DRBD)

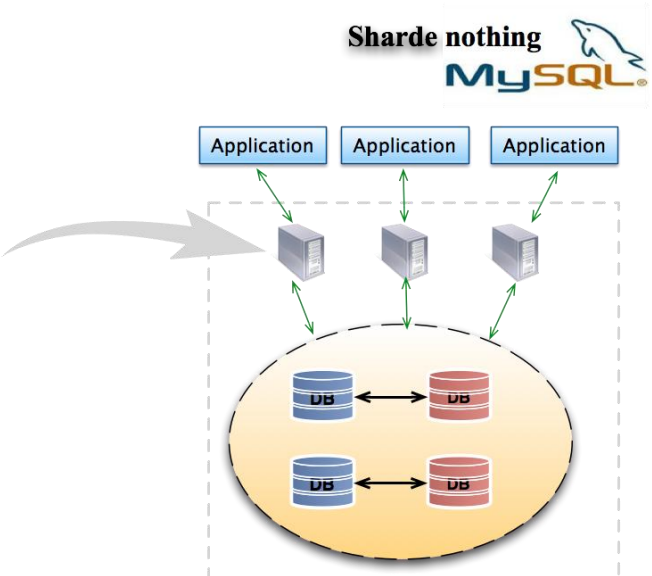


Sharde Nothing方式



Sharde Nothing方式

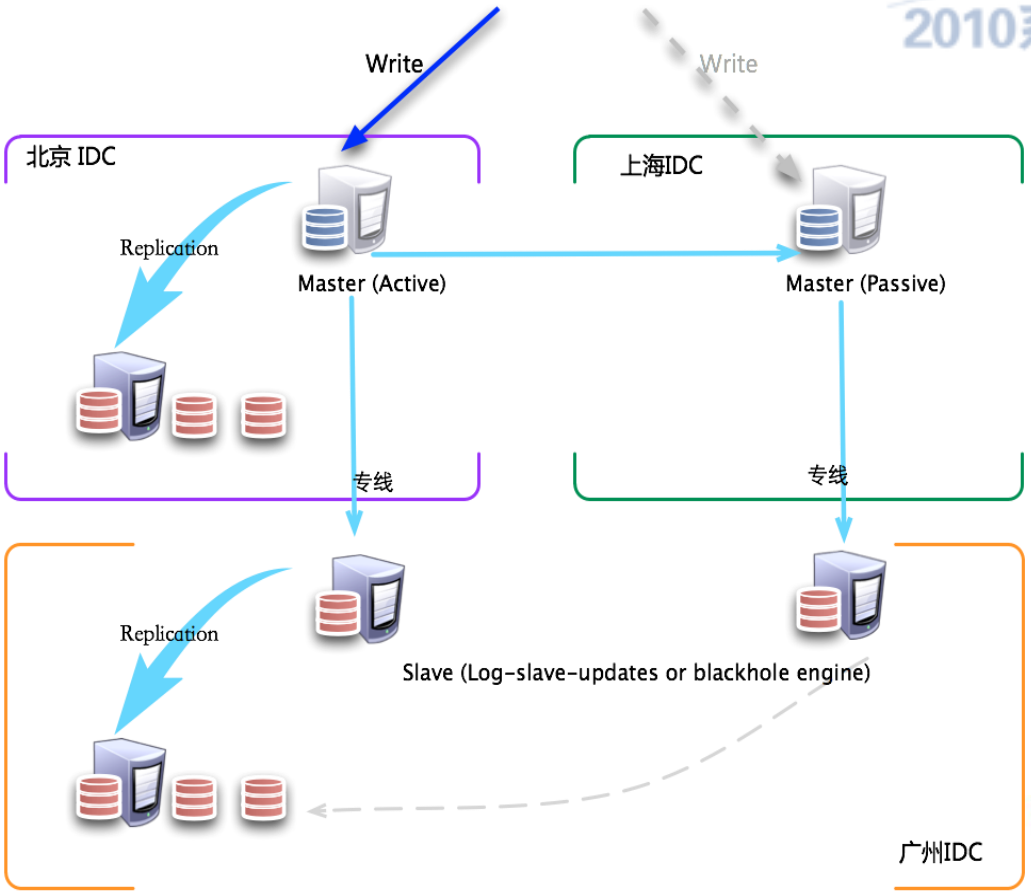
1. IDC容灾问题
2. 网络带宽要求高
3. 安全问题
4. 应用设计方面
   （FK，join,fulltext）
5. 性能受限于硬件



议题

1. 复制结构
2. 基于Sharding策略的结构设计
3. 基于HA的scale-out设计
4. **跨IDC的scale-out设计实例**
5. Q&A

M(a)/M(p)结构应用实例



M(a)/M(p)结构应用实例

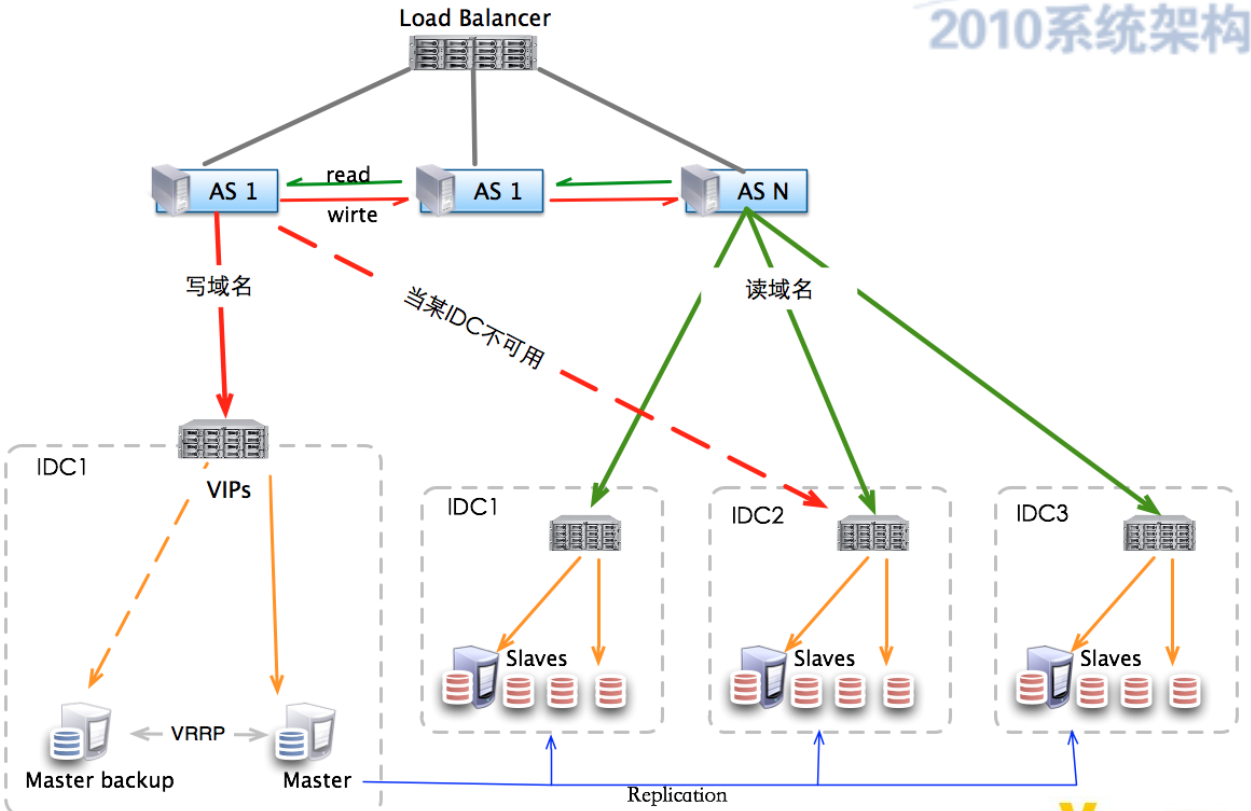
优点：

1. 能实现多IDC的高可用性和容灾
2. 减少网络流量成本
3. 减少主库的同步压力

缺点：

1. lag增加
2. 资源成本增加
3. 维护成本增加
4. 监控复杂

结合LVS结构应用实例



结合LVS结构应用实例

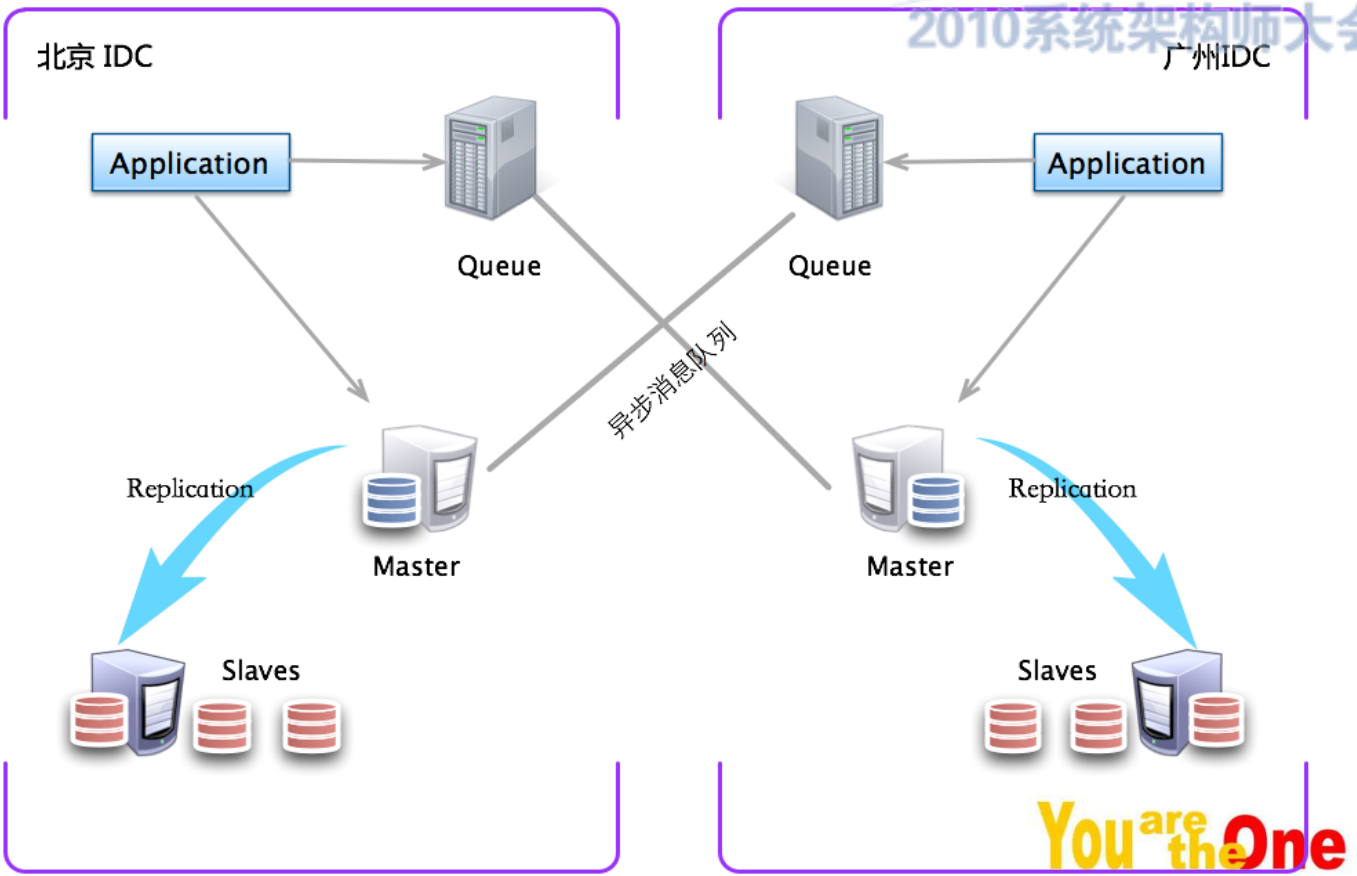
优点：

1. 能实现多IDC的高可用性和容灾
2. 好的读负载均衡
3. 主库能实现比较好的HA

缺点：

1. 资源利用容易不均衡
3. 维护成本增加
4. 监控复杂
5. LVS可能成为瓶颈

M(a)/M(a)结构应用实例


北京 IDC
广州IDC
Application
Application
Queue
Queue
异步消息队列
Replication
Replication
Master
Master
Slaves
Slaves

M(a)/M(a)结构应用实例

优点：

1. 能做到IDC的容灾
2. 基于BASE思想设计
3. 减少业务高峰对资源的需求
4. 不依赖于网络环境

缺点：

1. 要求业务的耦合性满足
2. 开发需要做一些工作
3. 监控和维护成本增加

议题

1. 复制结构
2. 基于Sharding策略的结构设计
3. 基于HA的scale-out设计
4. 跨IDC的scale-out设计实例
5. Q&A

新浪微博平台开发者大会即将举行:

可扩展分布式数据存储

微博技术架构

微博cache设计

即时搜索

社会化应用与新浪云

more...

谢谢

Q&A