十年架构 成长之路

SACC 第十届中国系统架构师大会

SYSTEM ARCHITECT CONFERENCE CHINA 2018

2018年10月17-10月21日 北京海淀永泰福朋喜来登酒店

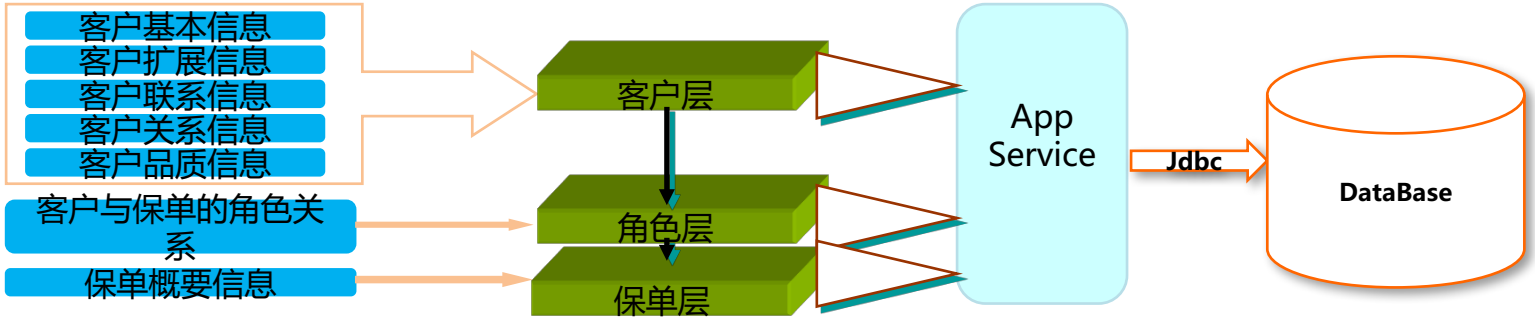
IT168.com ChinaUnix ITPUB

中国平安 PING AN
保险 · 银行 · 投资

平安人寿、产险开源架构分享

应用总体架构情况

应用总体架构



数据量

表名		数据量（单位亿）
ECIF_XXX	核心表	5.2
ECIF_XXX		1.6
ECIF_XXX		1.47
ECIF_XXX		2.3
…		…

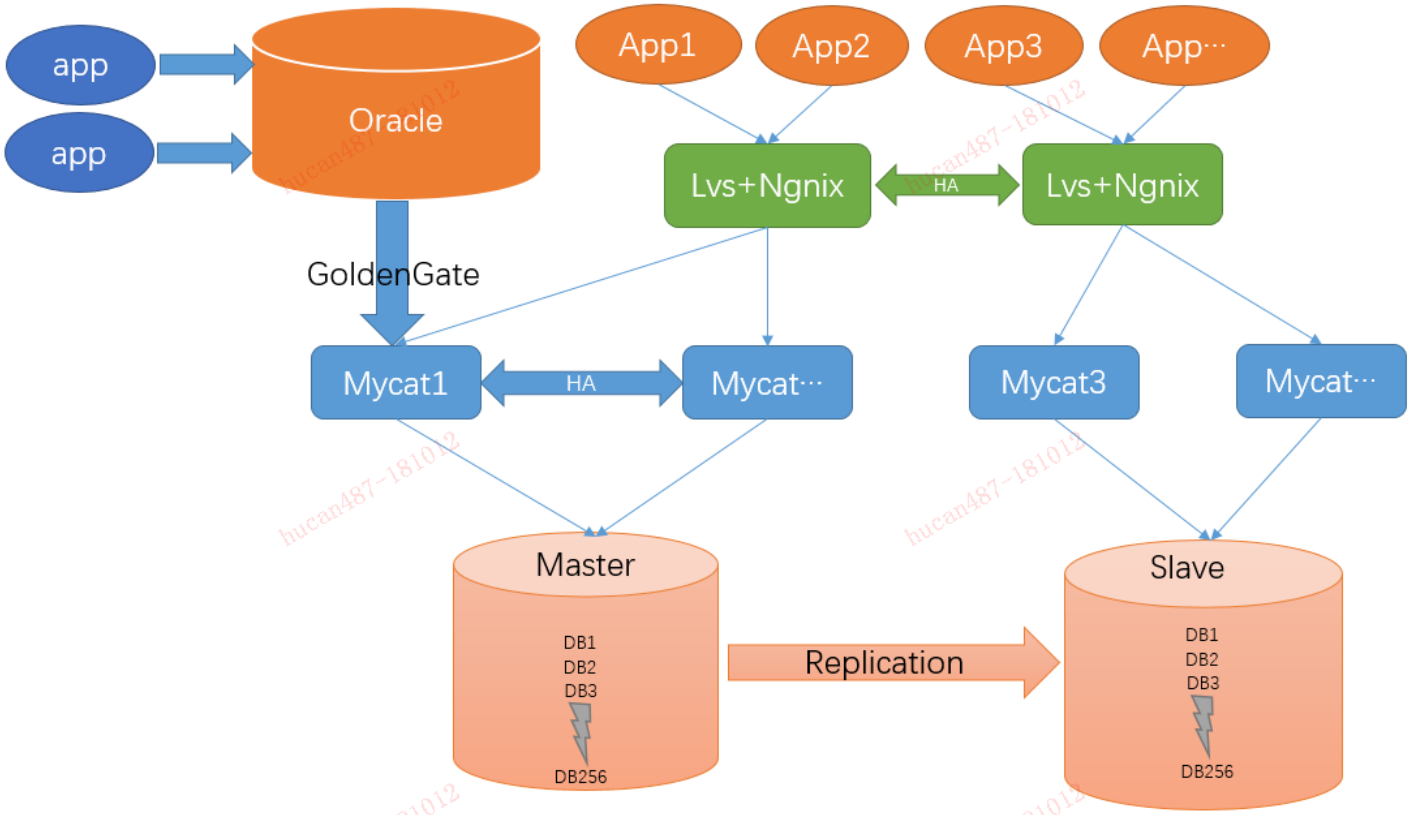
DB总体架构情况

Oracle支撑目前传统业务，后安排全部迁移新架构；

Ngnix+Lvs层用于负载均衡，读写分离；

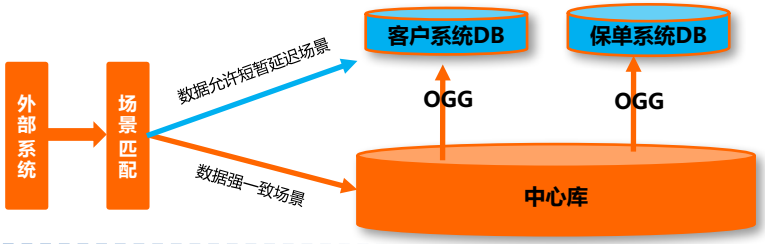
Mycat用于数据路由，DB分库配置；

底层使用Mysql作为数据持久化支撑；



Mysql分库设计方案

业务模型设计

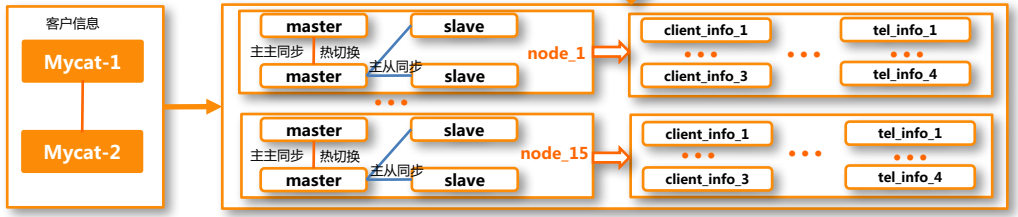


客户信息设计

◆ 查询条件
1：根据客户五项查询客户信息（其中有两项，三项，四项查询）
2：根据客户查询客户信息
3：根据保单号及角色查询客户信息
4：根据手机号码查询客户信息

表名（客户）	大小（单位G）	数据量（单位亿）	分片字段
ECIF_A	27.2	1.47	IDNO
ECIF_B	38.9	2.3	CLIENTNO
ECIF_C	44	1.8	CLIENTNO
ECIF_D	16.8	1.5	CLIENTNO
ECIF_TE	13.7	1.4	CLIENTNO
ECIF_F	16.5	1.5	CLIENTNO

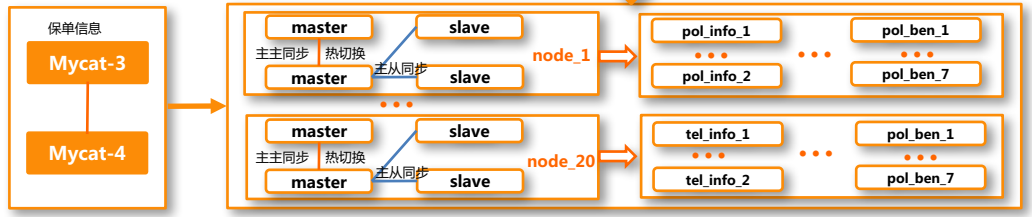
◆ 客户信息节点计算
1：每个表预期存储3百万条
2：初步预算需100个节点
3：1.6亿/15*300万≈ 3个分表存储客户基本信息
4：2.3亿/15*300万≈4个分表存电话信息



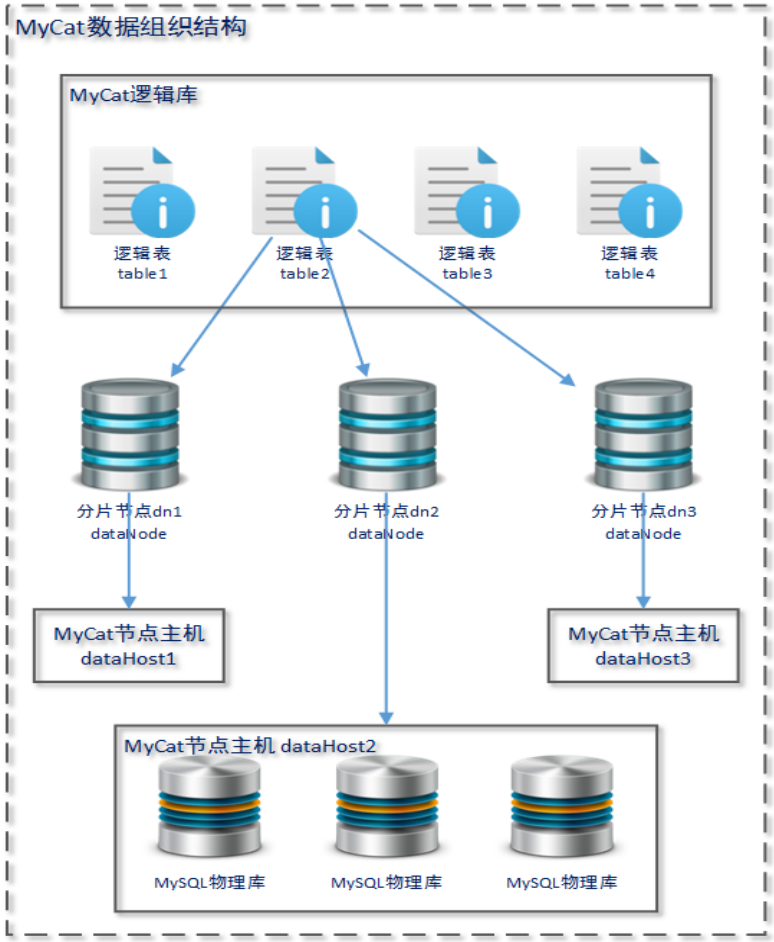
保单信息设计

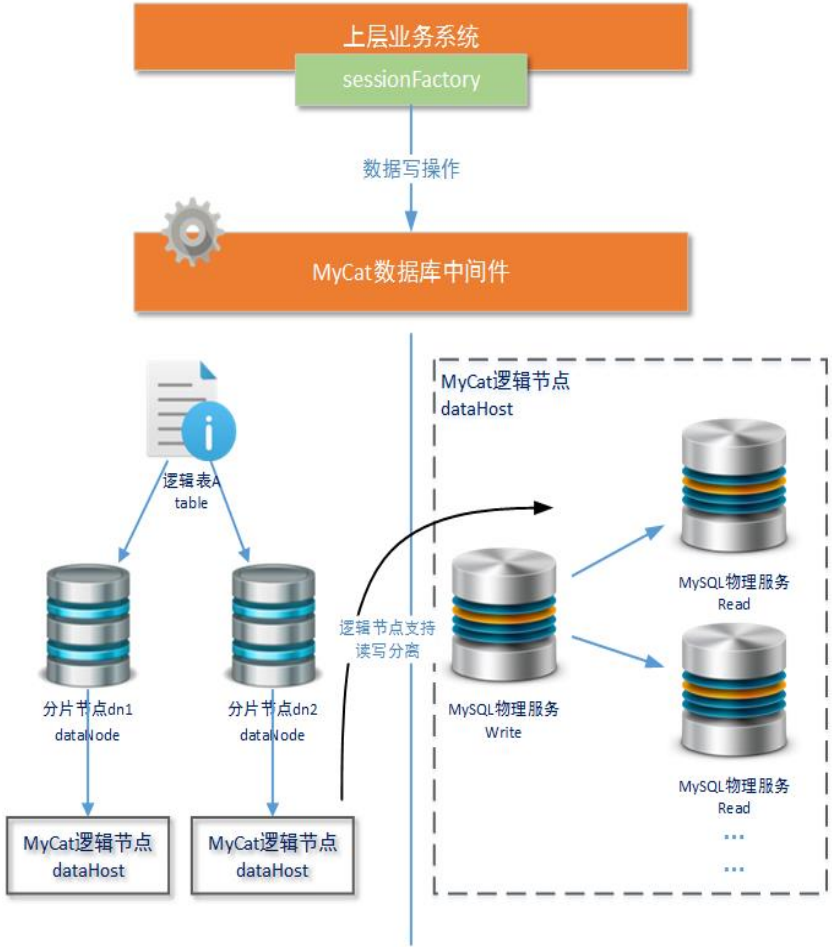
表名（保单）	大小（单位G）	数据量（单位亿）	分片字段
POL_A	96.4	5.2	INSNO
POL_B	36.2	1.6	APPLICANT
POL_C	6.2	0.6	APPLICANT
POL_D	32.3	1.7	APPLICANT
POL_E	79	5.2	INSNO

◆ 保单信息节点计算
1：每个表存储四百万条
2：初步预算需20个节点
3：1.6亿/20*400万≈2个分表存储保单信息
4：5.2亿/20*400万≈7个分表存储保单险种信息



MyCAT逻辑架构图


MyCat数据组织结构
MyCat逻辑库
逻辑表 table1
逻辑表 table2
逻辑表 table3
逻辑表 table4
分片节点dn1 dataNode
分片节点dn2 dataNode
分片节点dn3 dataNode
MyCat节点主机 dataHost1
MyCat节点主机 dataHost3
MyCat节点主机 dataHost2
MySQL物理库
MySQL物理库
MySQL物理库


上层业务系统
sessionFactory
数据写操作
MyCat数据库中间件
逻辑表A table
分片节点dn1 dataNode
分片节点dn2 dataNode
MyCat逻辑节点 dataHost
MyCat逻辑节点 dataHost
逻辑节点支持读写分离
MyCat逻辑节点 dataHost
MySQL物理服务 Write
MySQL物理服务 Read
MySQL物理服务 Read

逻辑表分库配置案例

schemas配置：

```
<table name=  "telephone_info" dataNode="myecif,myecif_$1-128" autoIncrement="false" rule="telephone_info"/>
<table name="address_info" dataNode="myecif,myecif_$1-128" autoIncrement="false" rule="address_info"/>
```

dataNode配置：

```
<dataNode name="myecif_1" dataHost="myecif0" database="myecif_0001"/>
 .....
<dataNode name="myecif_33" dataHost="myecif1" database="myecif_0033"/>
 .....
<dataNode name="myecif_65" dataHost="myecif2" database="myecif_0065"/>
 ....
```

rules配置:

```
<tableRule name=“address_info">
    <rule>
      <columns>clientno</columns>
      <algorithm>address_info</algorithm>
    </rule>
  </tableRule>
<function name=“address_info" class="io.mycat.route.function.PartitionByJumpConsistentHash">
    <property name="totalBuckets">256</property>
  </function>
```

支撑业务量情况

调用量

每天调用量8000W左右，查询客户信息6300W左右，查询保单信息1700W左右

系统	调用次数	平均响应时间(ms)	场景
A	3700W次/天	22.372	客户信息查询：80% 1. 根据5项、4项、3项查询客户信息（FREY+UWS） 2. 疑似客户判断 a) FREY：3项、2项 b) UWS：5项、4项、3项、2项
			客户信息新增\更新：20%（FREY+UWS）
B	1200W次/天	25.34	客户信息查询：60% 应用场景：登录，客户资料变更
			保单信息查询：40% 应用场景：查询名下保单
C	1100W次/天	35.526	客户信息查询：90% 应用场景：报案受理时核实客户信息
			保单信息查询：100% 应用场景： 查询客户保单情况
D	570W次/天	16.9107	中间系统，暂无法统计
E	370W次/天	26.9267	客户信息查询：100% 应用场景：查询客户
F	230W次/天	25.3258	客户信息查询：100% 应用场景：疑似客户（MIT调用）
G	180W次/天	24.1542	客户信息查询：100% 应用场景：查询客户
H	100W次/天	27.3332	客户信息查询：40% 应用场景：查询服务人员
			保单信息查询：60% 应用场景： 寿险保单信息

系统优点

支持高并发

- **新系统支持高并发**

新系统可以支持90000TPS以上。

TPS分布：

FREY：30000TPS
金管家：30000TPS
其他：30000TPS

支持高可用

- Mycat多活配置；
- Mysql主从同步并可以自动完成主从切换;
- Ngnix服务自动切换；

另：应用层可降级保持可用性

1. 核心数据库出现性能瓶颈，可以屏蔽非核心业务调用,保持核心业务正常运行。
2. 缓存出现问题，可以切换核心数据库，避免大面积调用异常。

保持数据一致性

- **统一客户信息出入口**

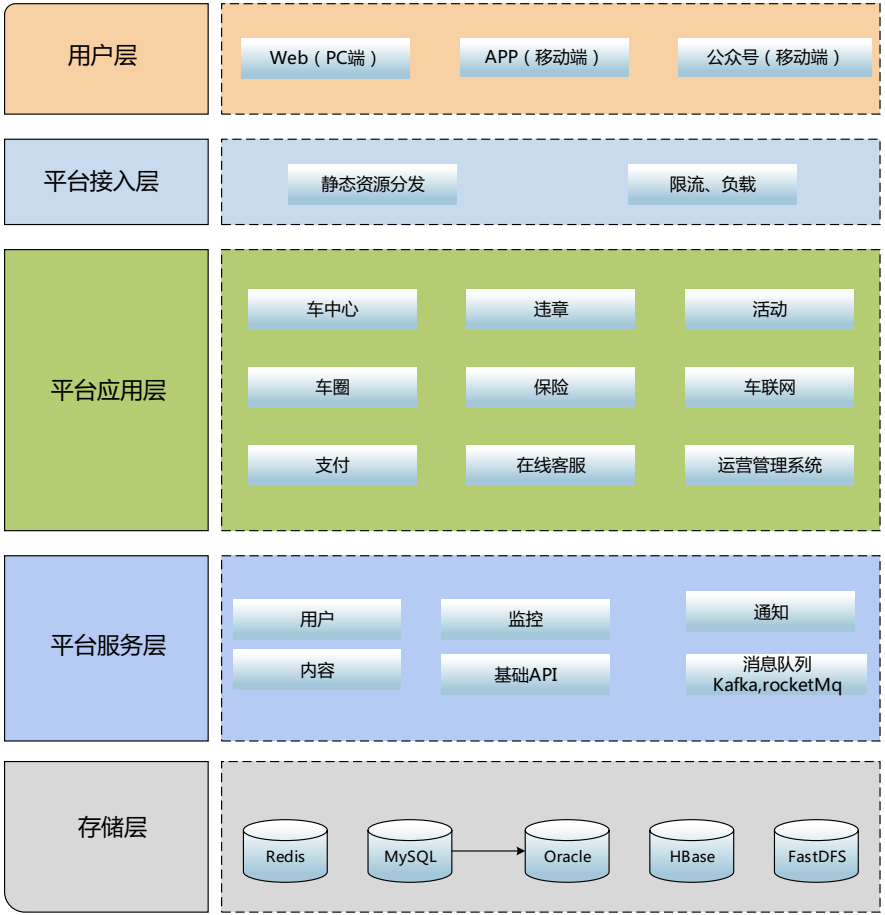
客户信息由客户信息系统持久化到核心中心库。

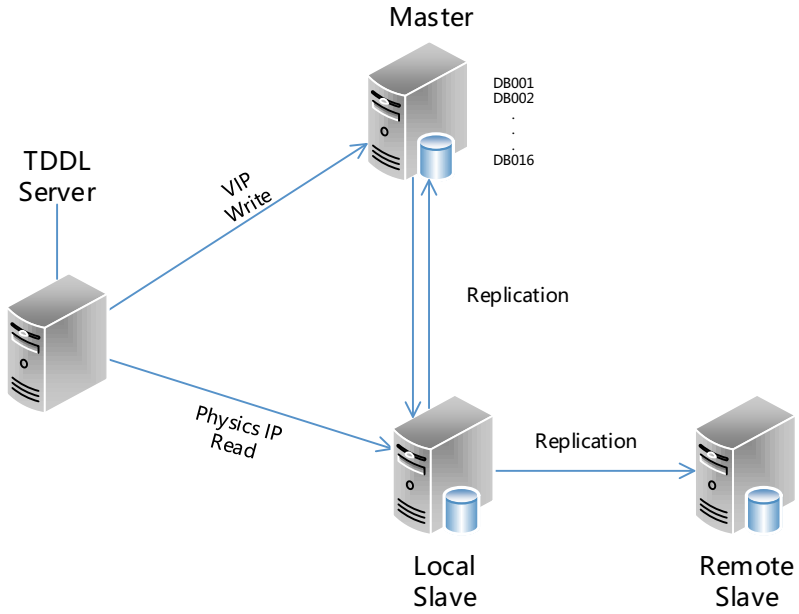
客户信息变更记录可追踪

支持横向扩展

- Mycat数据路由层支持横向扩展；
- Mysql DB层支持横向扩展；

平安好车主整体架构


用户层
Web（PC端）
APP（移动端）
公众号（移动端）
平台接入层
静态资源分发
限流、负载
平台应用层
车中心
违章
活动
车圈
保险
车联网
支付
在线客服
运营管理系统
平台服务层
用户
监控
通知
内容
基础API
消息队列
Kafka,rocketMq
存储层
Redis
MySQL
Oracle
HBase
FastDFS


Master
DB001
DB002
DB016
TDDL
Server
VIP
Write
Replication
Physics IP
Read
Replication
Local
Slave
Remote
Slave

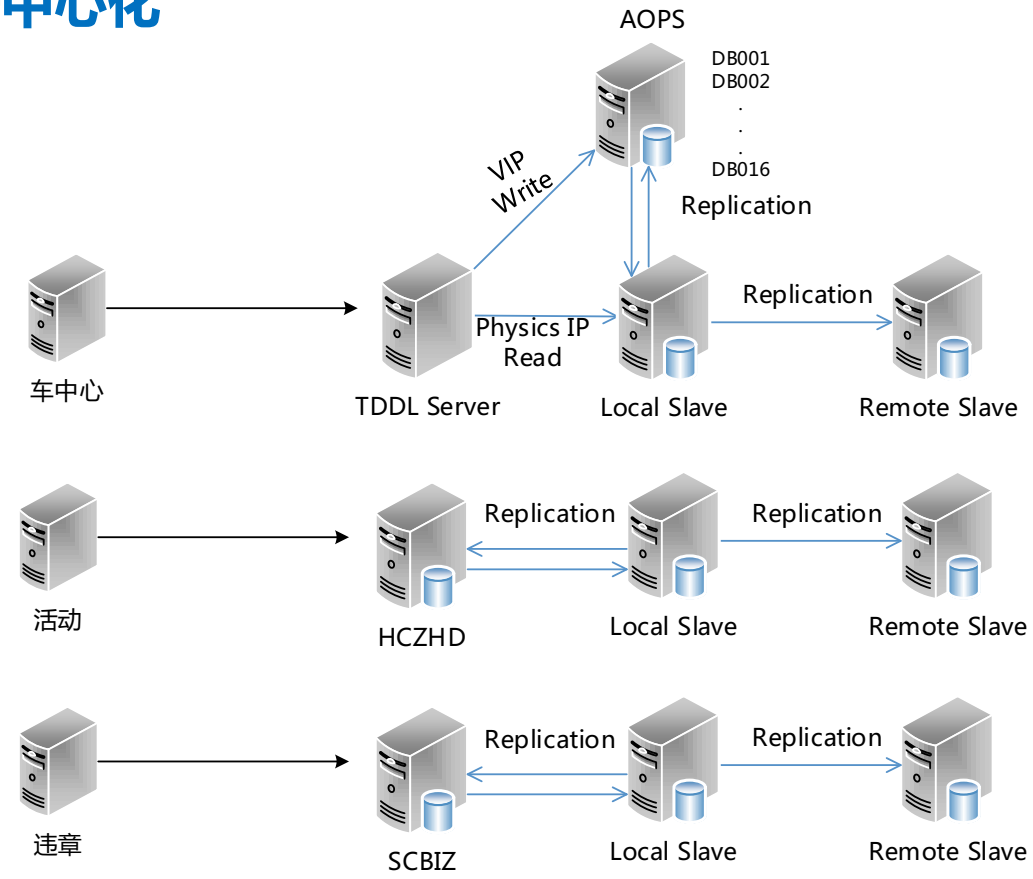
微服务架构—去中心化

优点：

将单体应用分解为一组服务，这些服务定义了明确的RPC或消息驱动的API边界。微服务开发的速度要快很多，更容易理解和维护。

其次，这种体系结构使得每个服务都可以由专注于此服务的团队独立开发。这就意味着开发人员可以采用新技术编写或重构服务，由于服务相对较小，所以这并不会对整体应用造成太大影响。

第三，微服务架构可以使每个微服务独立部署。开发人员无需协调对服务升级或更改的部署。这些更改可以在测试通过后立即部署。



缺点：：

开发人员需要基于RPC或者消息实现微服务之间的调用和通信，而这就使得服务之间的发现、服务调用链的跟踪和质量问题变得的相当棘手。

微服务的另一个挑战是分区的数据库体系和分布式事务。更新多个业务实体的业务交易相当普遍。在微服务架构下，不同服务可能拥有不同的数据库。

微服务架构对应用稳定性也带来了很大的挑战。传统的单体WEB应用只需启动单一的APP 服务即可，而对微服务需要启动它依赖的所有其他服务。这种复杂性不可低估。

配置案例

连接池信息

com.pingan.tddl.atom.app.db_core.db0　　----按照DB个数分配连接池

```
userName=db_user
minPoolSize=20
maxPoolSize=400
idleTimeout=1
blockingTimeout=10000
connectionProperties=rewriteBatchedStatements=true&characterEncoding=UTF8&connectTimeout=1000&autoReconnect=true&so
cketTimeout=12000
```

分表规则

```
<bean id=“db_car_invert" class="com.pingan.tddl.rule.TableRule">
        <property name="allowFullTableScan" value="false" />
        <property name="dbNamePattern" value=“db_core_db{000}" />
        <property name="dbRuleArray" value="(#car_id,1,1024# % 1024).intdiv(64)" />
        <property name="tbNamePattern" value=“db_car_invert{0000}" />
        <property name="tbRules" value="#car_id,1,1024# % 1024" />
</bean>
```

应用用户量

4900万注册用户

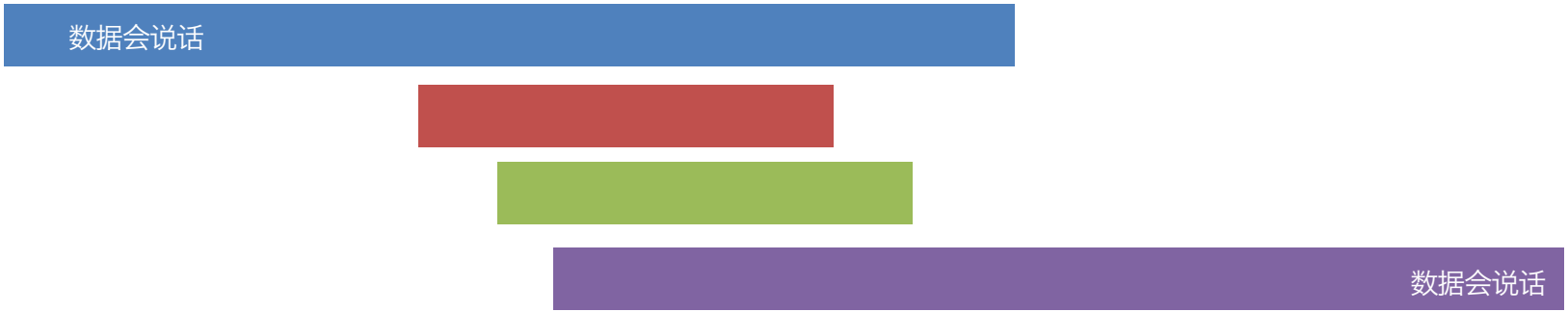
4年间用户增长迅速

3300万绑车用户

自主上传车辆信息

190万日活用户

精准推送，强化用户关系



950万月活用户

促活拉新，用户粘性高

590万社区月活

头条、圈子、问问，内容丰富有趣，形式多样

270万车后服务月活

保单查询、道路求援、违章处理等车后服务月活全国第

THANKS