大众点评数据库访问层的架构设计

自我介绍

- 美团点评 朱浩
- 目前是基础架构存储团队负责人，主要工作围绕MySQL和Redis打造稳定可靠的访问层中间件、存储，以及建设相应的运维体系。
- 曾在点评做过分布式调度系统、AB测试平台Gemini，点评的监控系统CAT以及点评的长连接等系统。

目录

- 为什么需要数据库访问层
- 架构和主要功能
- 运维体系的建设

在没有数据库访问层时

开发

- 连接池的配置太多，容易配错
- 数据库配置一旦变更，需要重启业务机器
- 一旦发生故障，不能迅速定位故障原因
- 一旦发生故障，不能迅速恢复
- 读写分离对业务有侵入
- 不支持分库分表

DBA

- 无法高效迁移数据库
- 无法动态的调整数据库流量
- 无法对有问题的SQL进行限制
- 一旦发生故障，不能迅速发现故障数据库

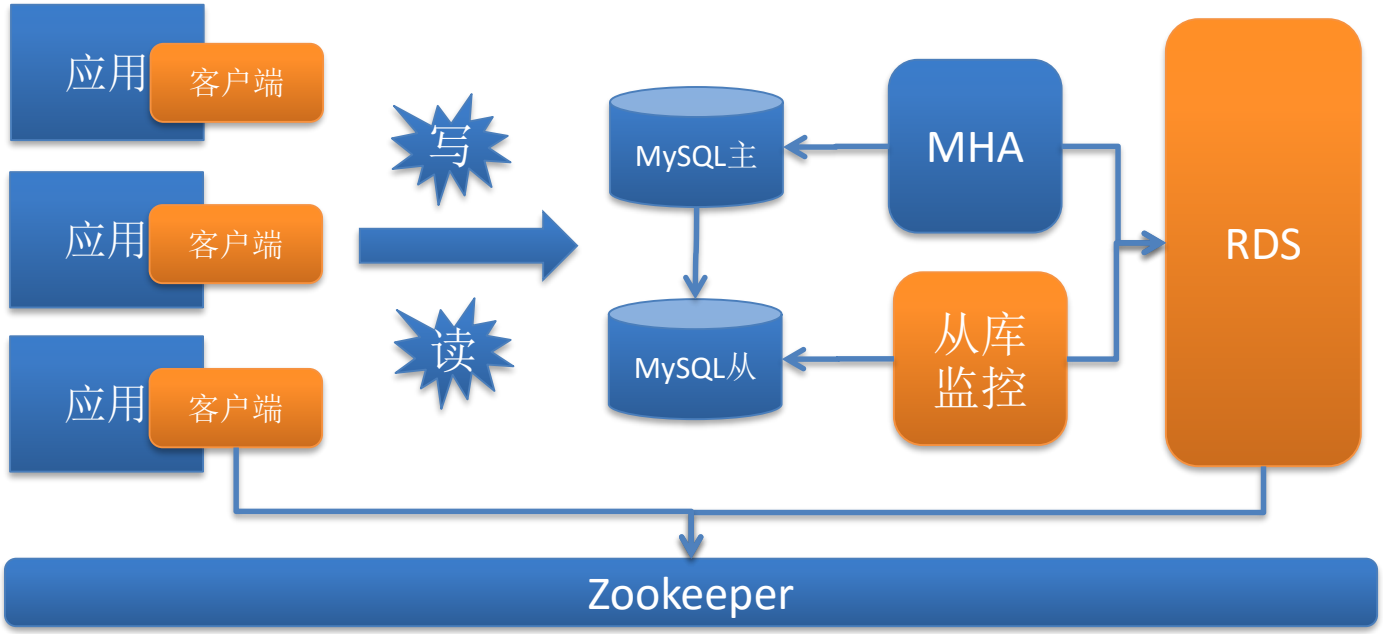
面临的架构选择

实现方案	业界组件	优点	缺点
proxy-based	mycat, 阿里cobar, Atlas	多语言支持	风险大 实现难度大
客户端-jdbc实现	阿里tddl	实现难度适中 支持各种ORM框架	目前仅支持java

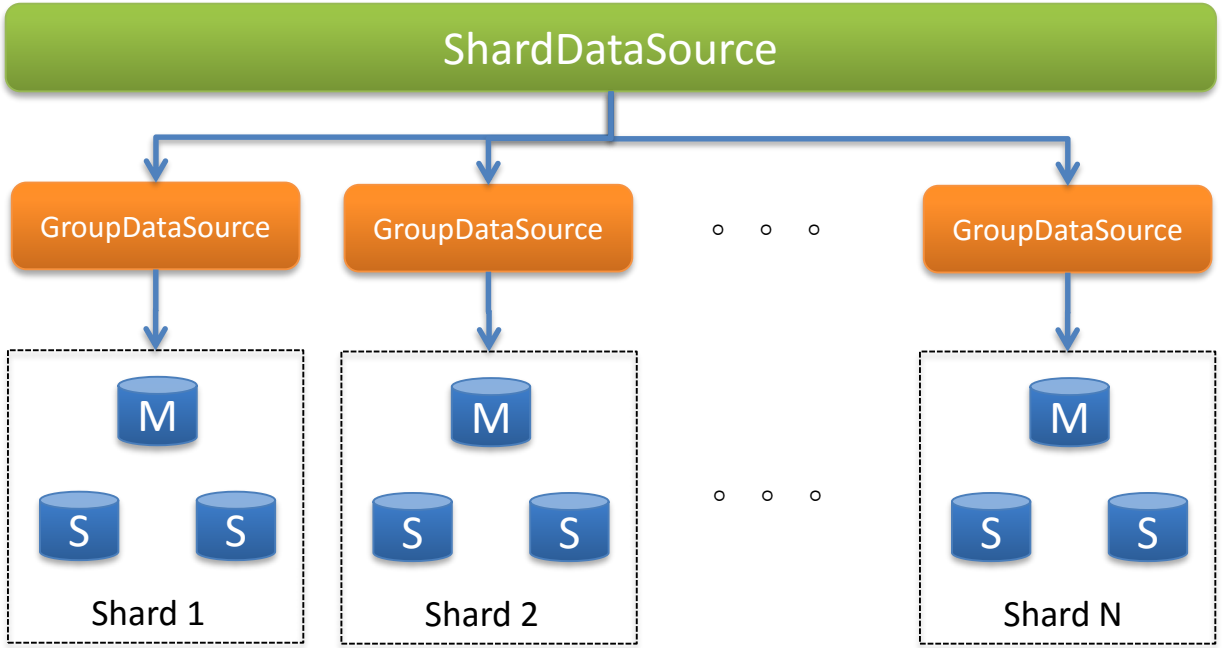
目录

- 为什么需要数据库访问层
- 架构和主要功能
- 运维体系的建设

整体架构



客户端架构


ShardDataSource
GroupDataSource
GroupDataSource
GroupDataSource
M
S
S
Shard 1
M
S
S
Shard 2
M
S
S
Shard N

主要功能

- 统一的源数据配置管理
- 动态的数据源
- 支持读写分离和分库分表
- 统一的监控方案
- 统一的高可用方案

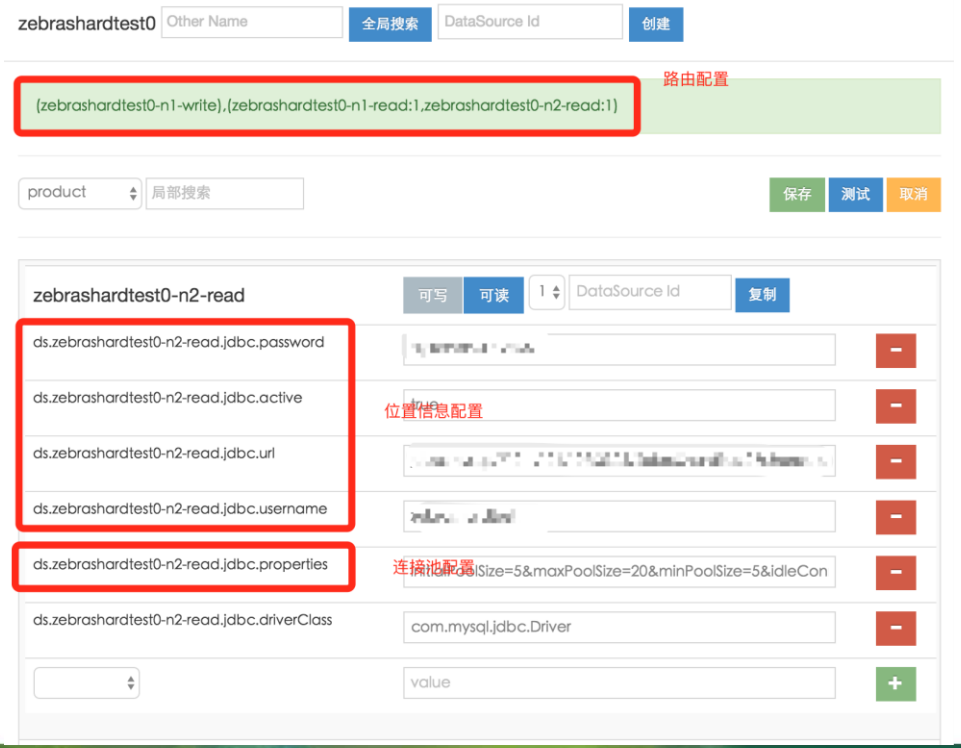
接入方式

```
<!-- 读写分离的数据源配置 -->
<bean id="groupDs" class="com.dianping.zebra.group.jdbc.GroupDataSource" init-method="init">
    <property name="jdbcRef" value="test"/>
</bean>

<!-- 分库分表的数据源配置 -->
<bean id="shardDs" class="com.dianping.zebra.group.jdbc.ShardDataSource" init-method="init">
    <property name="ruleName" value="test"/>
</bean>
```

统一的数据源配置管理

- 接入只需最简配置
- 屏蔽数据库地址
- 屏蔽连接池配置
- 屏蔽路由配置
- DBA管理配置



动态性

调整数据库地址、用户名密码	调整路由权重	调整连接池配置
• 支持DBA在线迁移数据库 • 开发无需重启应用服务器	• DBA在线调整数据库流量，分担数据库压力 • 如果主从延迟严重，可以禁用某个从库	• 业务开发在线调整连接池配置，应对流量压力，服务器无需重启

读写分离

- 按照权重路由
- 主从延迟的挑战
  - Hint方式支持单SQL走主库
  - 整个请求内的SQL走主库
  - 整个调用链上的SQL走主库
- 事务
  - 事务内的SQL全部走主库

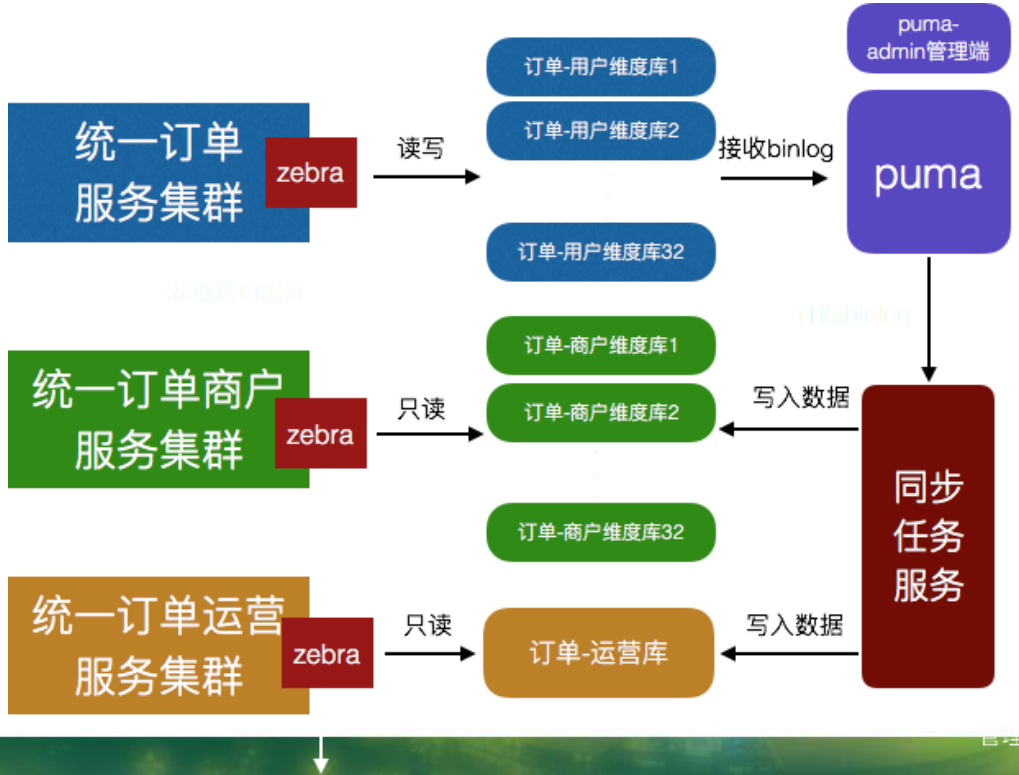
分库分表

- 分库分表规则须灵活
  - 以groovy语法实现
- 路由的SQL执行须高效
  - 并发执行
- 结果合并
  - limit, group by, order by, distinct等
- 事务支持
  - 支持单库事务，不支持分布式事务

分库分表的多维度太多？？

- 通过共享信息的方式
  - 比如订单库的两个维度：UserID和OrderID
  - UserID.subString(0,6) == OrderID.subString(0,6)
- 通过增加一张映射表
  - 比如验券业务有两个维度：UserID（维度）和ReceiptID
  - 增加了一张ReceiptID到UserID的映射表
- 通过冗余数据进行
  - 业务双写
  - binlog增量同步

大众点评优惠订单分库分表架构



实时监控

- 客户端进行数据上报至CAT(点评统一的监控系统)
- CAT实时监控的指标
  - SQL的总数、平均响应时间、95线和99线
  - SQL的访问情况分布
  - SQL的大小和返回值的监控：防止请求太大或者返回值太多造成oldGc
  - 如果是读写分离，请求在数据库实例的分布
  - SQL类型
  - 连接池的大小的监控
- CAT数据库大盘
- 支持DBA进行5分钟内发现和处理故障

实时监控

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59		

09:29	09:28	09:27	09:26	09:25	09:24	09:23	09:22
数据库访问正常	数据库访问正常	数据库访问正常	数据库访问正常	[illegible]	数据库访问正常	数据库访问正常	数据库访问正常

时间	数据库	主机名	IP	标题	内容	状态
09:26:18	[illegible]	[illegible]	[illegible]	ZEBRA	延迟:180s达到阈值,MARKDOWN	✖
09:26:18	[illegible]	[illegible]	[illegible]	ZEBRA	延迟:180s达到阈值,MARKDOWN	✖
09:26:18	[illegible]	[illegible]	[illegible]	ZEBRA	延迟:180s达到阈值,MARKDOWN	✖
09:26:18	[illegible]	[illegible]	[illegible]	ZEBRA	延迟:180s达到阈值,MARKDOWN	✖

高可用方案

- 主库高可用使用MHA
- 从库高可用自研服务进行探测
  - 探测从库不可用
  - 探测从库主从复制断开
  - 一旦以上情况发生，就将从库进行markdown
- SQL黑名单
  - zebra给每个SQL生成唯一ID以注释的方式加到每个SQL前面
  - DBA找到危险的SQL的ID，添加到黑名单，进行限流

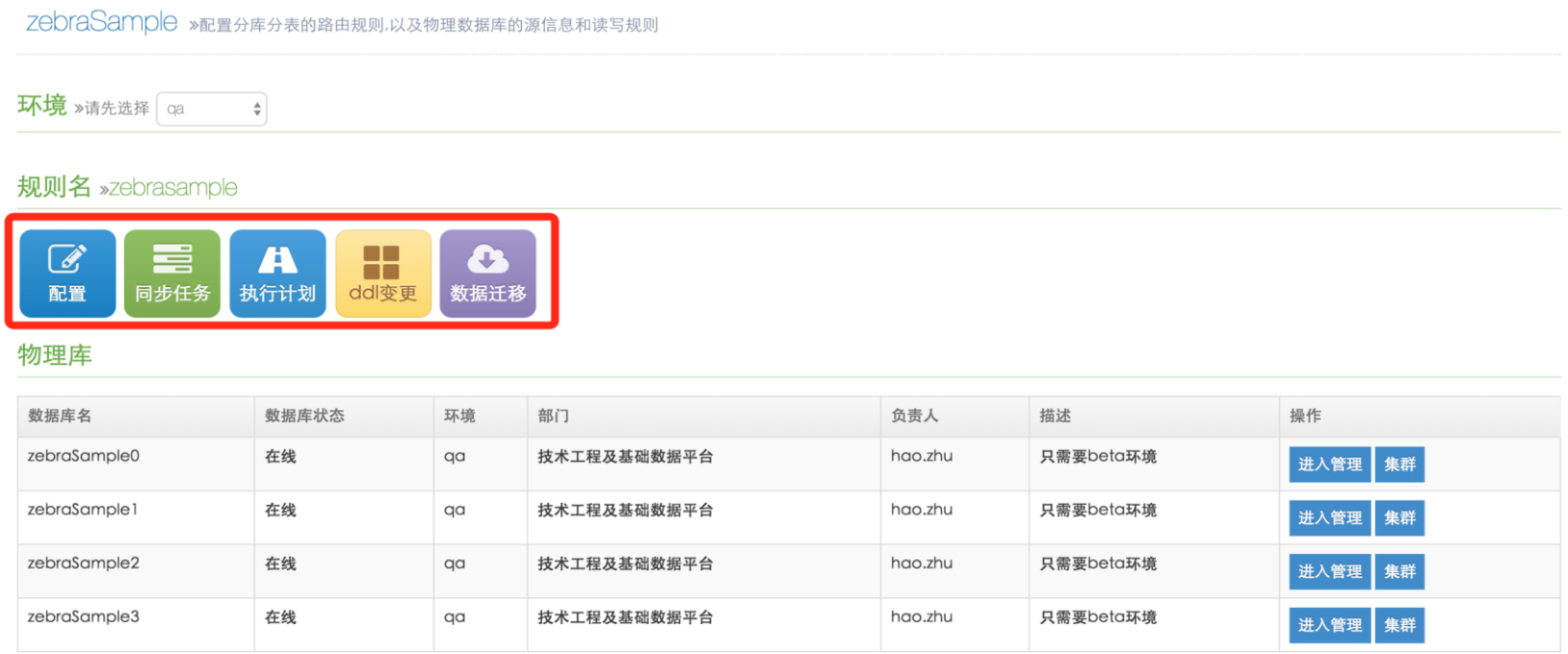
目录

- 为什么需要数据库访问层
- 架构和主要功能
- 运维体系的建设

运维体系建设

- 一键进行数据库创建，业务直接使用jdbcRef，提高研发效率
- 一键进行迁移数据库，业务无感知
- 提供统一的web的方式访问数据库，和客户端做到一致的体验
  - 集成数据查询、展示、SQL、历史等功能
  - 支持一键DDL变更
- 分库分表的运维体系
  - 一键进行分库分表的数据库搭建和DDL变更
  - 一键对分库分表的多维度数据进行同步
  - 支持在线查询SQL路由结果和数据结果
  - 大表数据一键进行分库分表的迁移

SQL Editor&分库分表体系建设



Q&A

代码地址：https://github.com/dianping/zebra