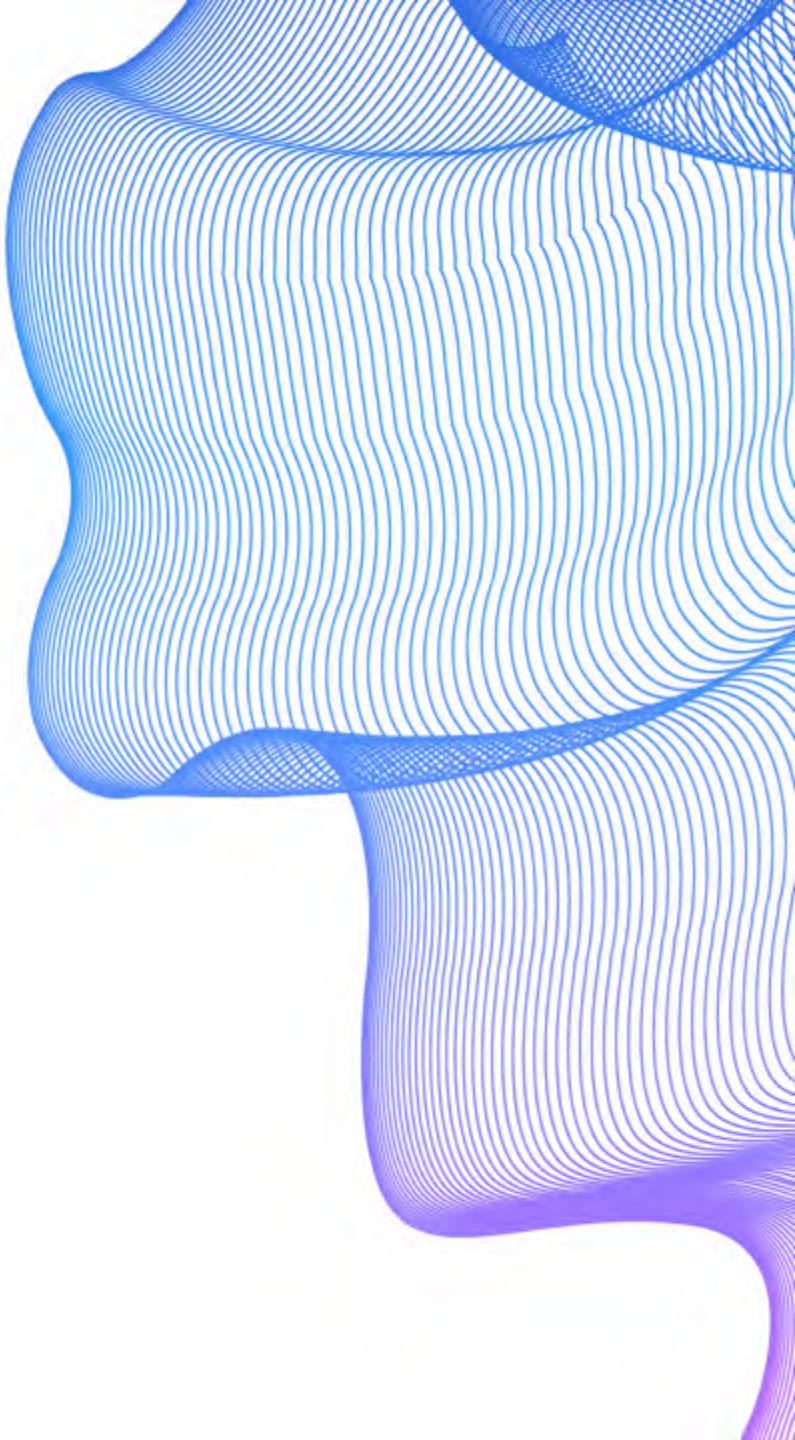
SimpleDB

高性能在线数据服务系统

杨嘉义 2017/05/12



Baidu 大数据

目录

应用背景

业务需求

超高并发、超低延迟的在线数据读写

多地域数据写入、数据完整一致

同时访问实时和批量数据，权限控制，高可用

复杂的业务计算逻辑，方便快速迭代

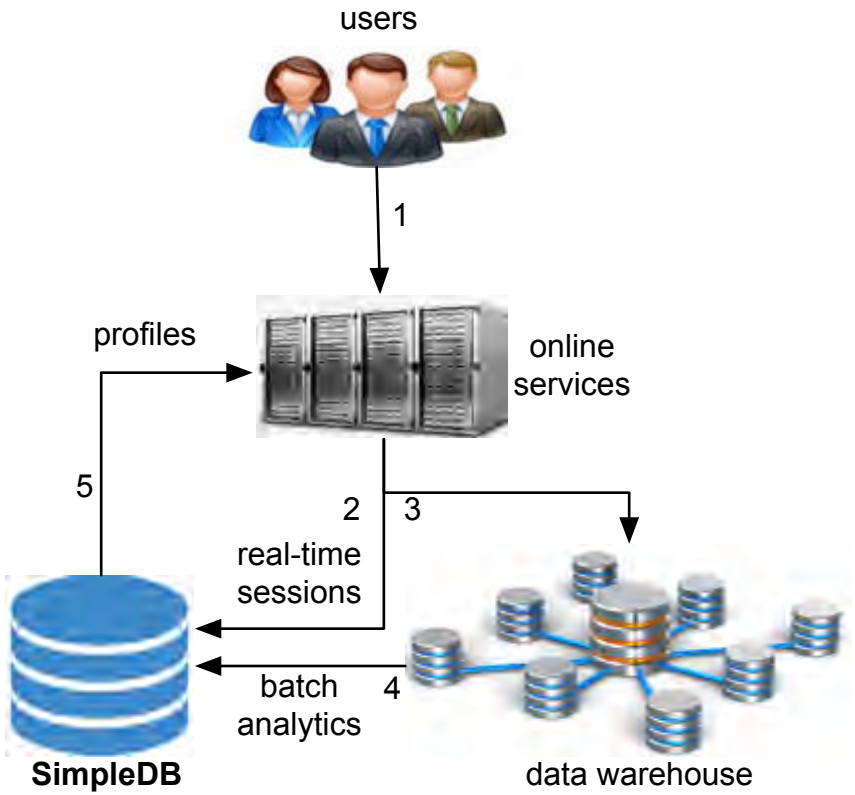
应用场景

Session、属性数据等存储场景

分布式cache、id-mapping服务场景

RTB、实时反作弊等实时辅助决策场景

其他同时需要低延迟、海量数据存储的在线场景



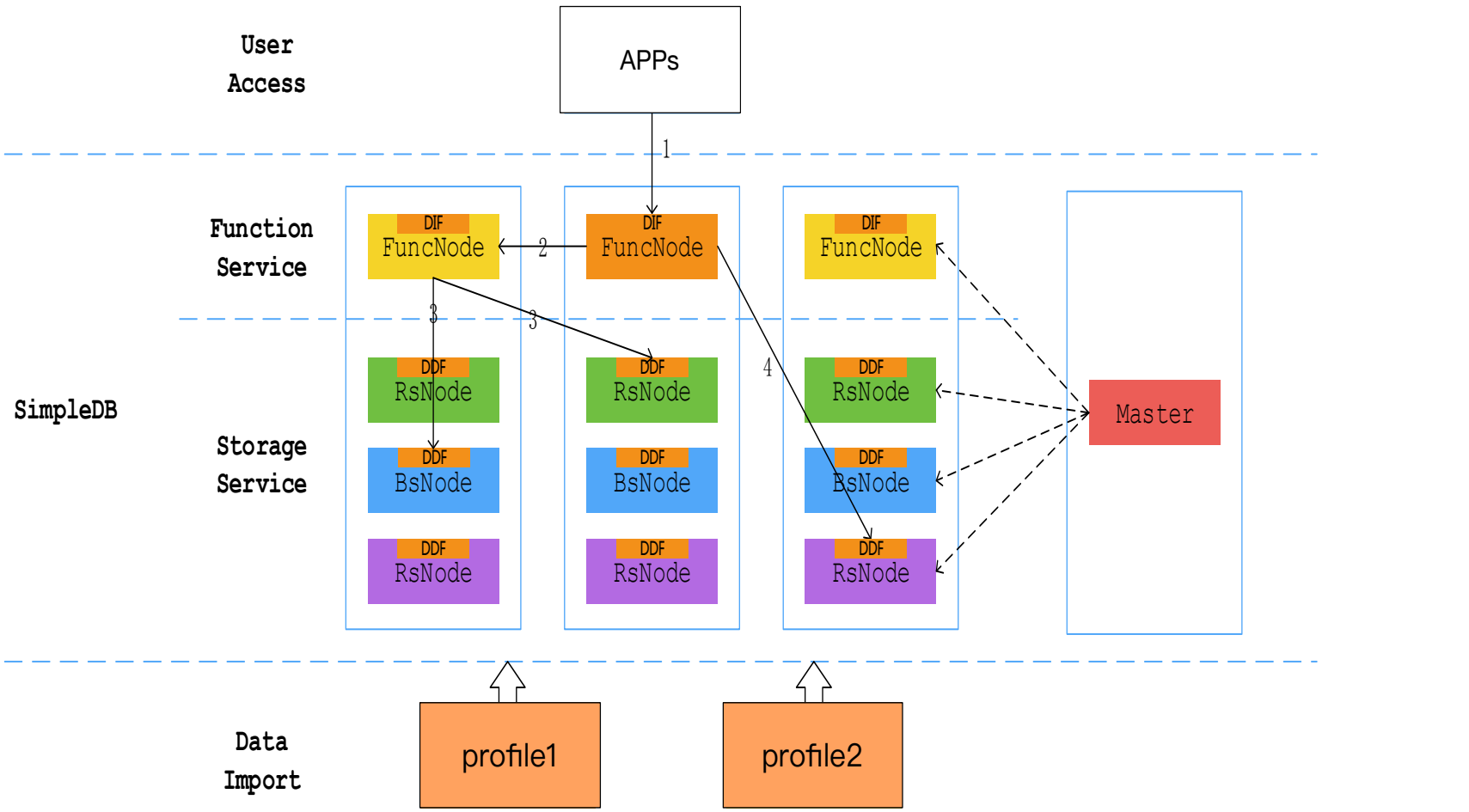
在线数据服务应用场景示例

设计目标

性能：数百万ops(ups&qps)，10ms级延迟，P级存储

功能：异地多活，多点写入，最终一致性，复杂UDF，过期淘汰

系统架构



在线数据服务分层架构图

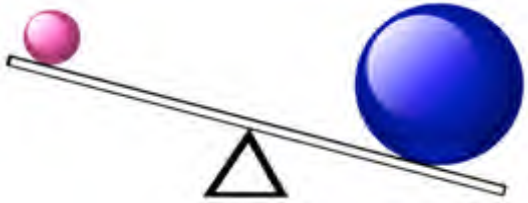
高性能

NoSQL DBs trade off traditional features to better support new and emerging use cases.

Andy Gross, Riak

事务ACID与隔离级别
支持复杂查询
数据绝对不可丢
复杂数据类型
强一致性

高性能



在高性能和传统特征之间权衡取舍

高性能——实时读写引擎

数据组织

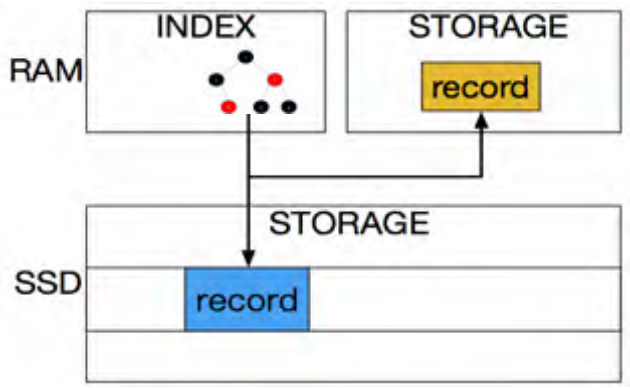
索引在内存，数据在RAM/SSD

索引：CAS, storage pointer

数据：schema-less, bytes

数据层级namespace. table. record

SET ns1.table1.key1 value1



数据组织及高性能读写

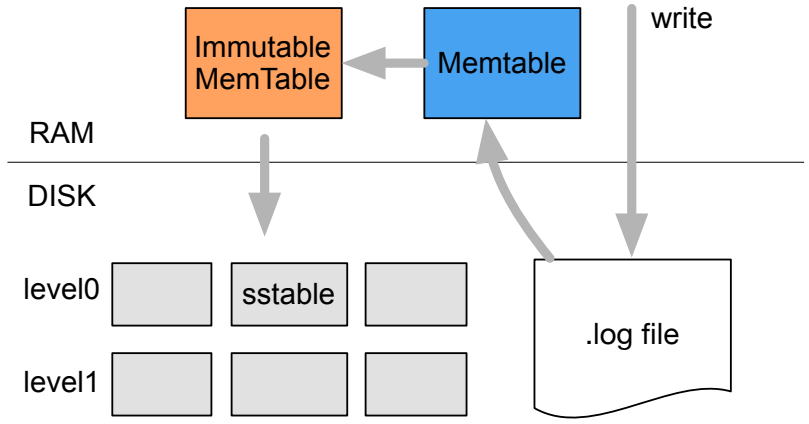
高性能读写

读数据最多一次落盘

先写write-buffer，异步flush SSD

更新索引指向新存储地址

Buffer write *vs.* write-ahead log



Write-ahead log示意图

高性能——批量只读引擎

批量服务

基于B+树只读KV引擎

从HDFS文件到在线服务全流程

数据累计时间长存储量巨大

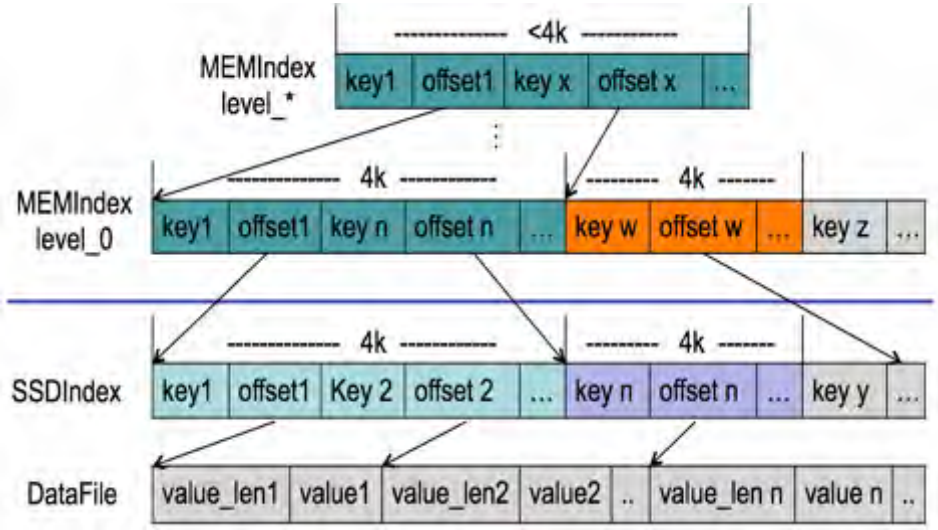
批量存储设计

MR产出数据文件，数据已分片且有序

按小时或天批量更新，只读，不加锁，mmap

索引一般放内存，但当value<1k时索引占空间很大

功能组件: importer, controller



只读引擎索引结构示意图

高性能——存储分配与访问

高效存储分配

内存使用Jemalloc分配，SSD按block分配

Record过期淘汰策略：TTL&过期数据 -> LRU删除 -> 停止写(MAX MEM)

定期回收与复用：追踪block有效数据量，回收较空闲block

SSD高效读写

大buffer写，direct IO

Block回收阈值60%，太高缩短磁盘寿命，太低降低资源利用率

Client访问

透明压缩、按长度压缩

本地缓存数据分布，single-hop直接访问底层数据

定期同步集群变化信息，失败重试机制

高可用——数据多副本

数据丢失概率

1DC 3副本降到2副本: 1.7万亿分一 增加到 8.2亿分之一（/年）

1DC 2副本 变成 2DC各2副本: 降到15亿亿分之一 (/年)

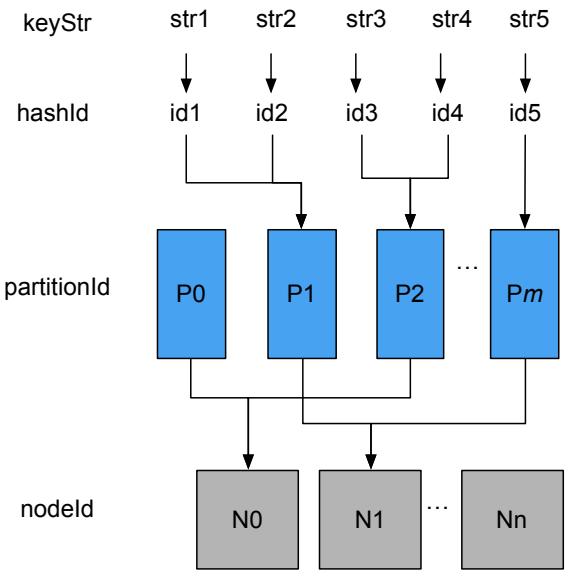
多DC场景下3副本可以降到2副本

数据分库要求

细分：分片数远大于机器数，固定分片数

热备：单节点同时有master和replica数据

强制分到同一库: key tag，减少rpc次数

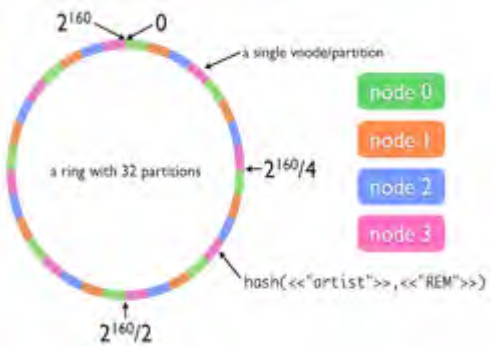


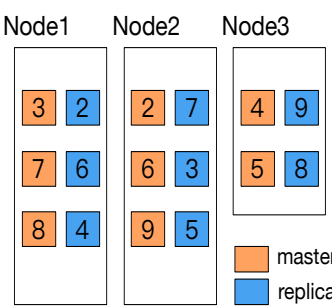
数据分库与分布

数据分布策略

虚拟节点一致性hash *vs.* 预计算分布表

心跳探活，同步集群状态





两种常见数据分布策略对比

高可用——异地多活

异地多活

用于灾备恢复、保证系统持续可用

每个DC都有数据读写，多点写入相互同步，也可单点写入

任意一个DC故障时，流量可以切到其他临近DC

挑战：延迟、数据同步、数据的一致性

数据异步同步

DC内部主从同步：

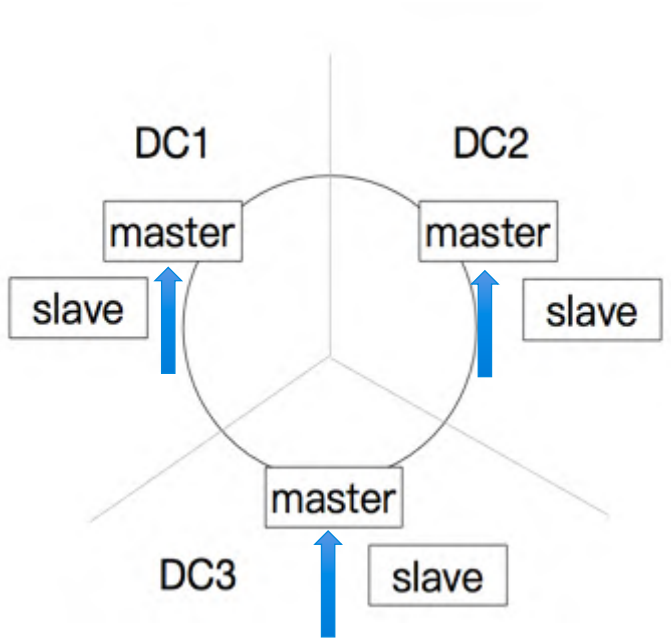
Read/write到master，异步同步replicas

故障时数据从分片可切换成主

跨DC数据同步：

Chain, active-active,active-passive, star等

选chain, A->B->C，减少跨地域网络带宽



多点写入示意图

高可用——元数据HA

元数据服务的功能

管理集群所有元数据

集群探活和rebalance、recovery

管理数据ns、table、partition操作及分布

管理Function存储、部署、版本及回滚

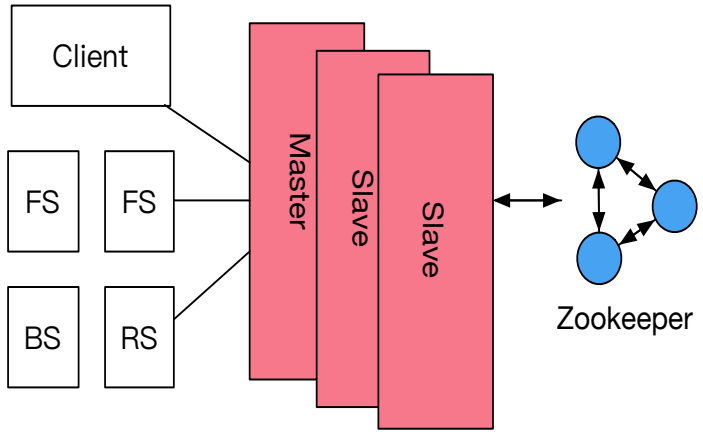
管理权限与审计

HA方案 ：Masters + Zookeeper

一个active master，多个slave，ZK负责选主

所有元数据信息都存在ZK

ZK: cluster coordinator, notification system



元数据高可用架构图

高可用——运维实践

故障处理方案

例行升级重启：重建索引，恢复数据，加速方案

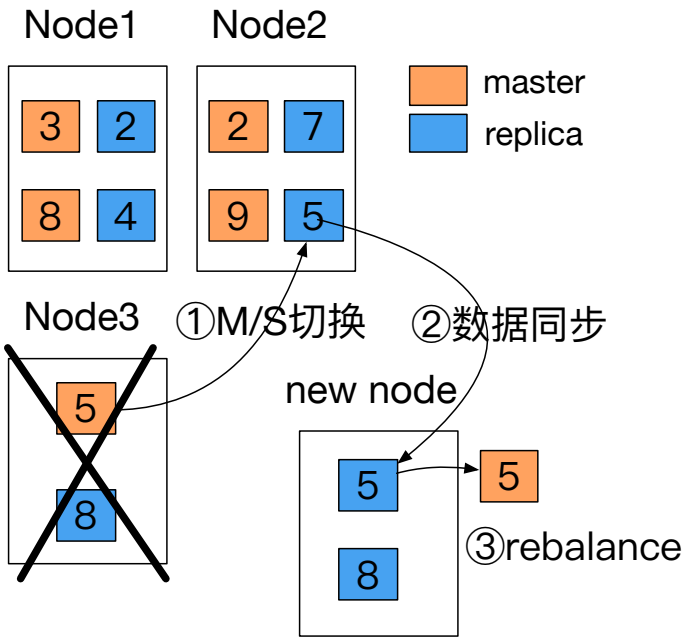
DC内节点磁盘故障：主从切换，同步数据，rebalance

DC内及DC间网络分割：继续写网内数据，日志堆积在本地，恢复后异步批量同步数据

高可用运维实践

数据分优先级，故障时裁剪非关键数据

Recovery时优先保证可读



单节点故障处理方案

最终一致性

数据操作持久化

用日志来记录分片的所有数据操作并持久化

只存key信息，value从存储中读取

跨地域及地域内数据同步

异步, batch merge

数据版本分配

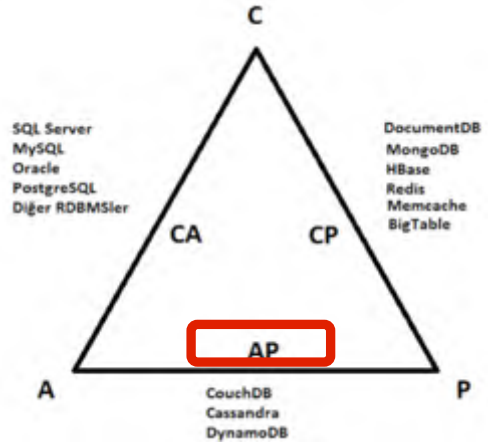
M/S切换和多写场景，使用新版本

最终一致性方案

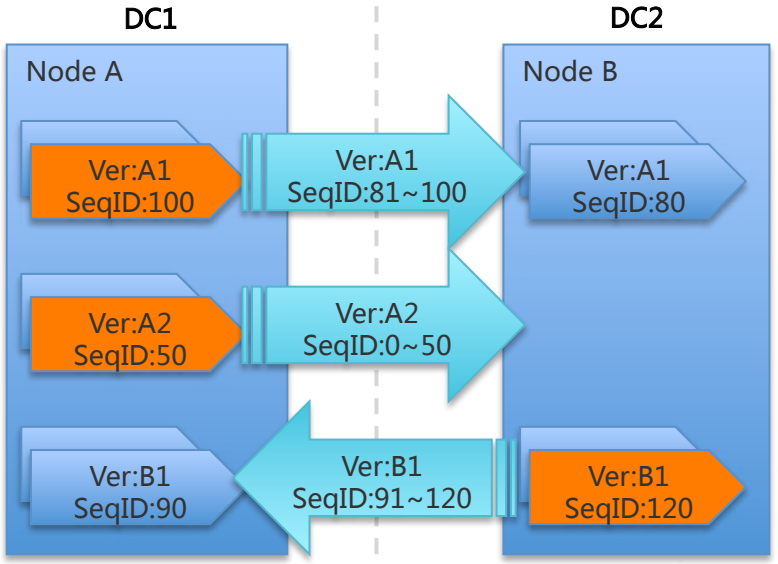
基于version ID和sequence ID

分片之间交换版本及序列信息，拉取diff

优化: 跳过历史版本数据



CAP原理选择AP



基于版本及序列号的最终一致性方案

实时批量数据融合

Lambda服务架构

batch + real-time

融合计算：合并，择优，筛选等

批量部分

DW + MR

经过精细化离线挖掘计算处理，按小时级或按天更新

实时部分

Kafka + ETL

简单计算，实时更新，端到端延迟低

辅助工具

Logging, debugging, monitor

复杂业务计算服务

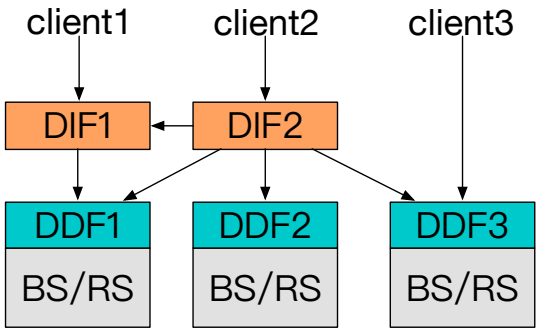
用户定义计算逻辑（UDF）

权限控制

复用通用计算逻辑

定制返回策略所需data view, 而非raw data

降低C/S之间网络带宽(1/10x ~ 1/100x)



Client端访问DIF&DDF示意图

两类UDF:

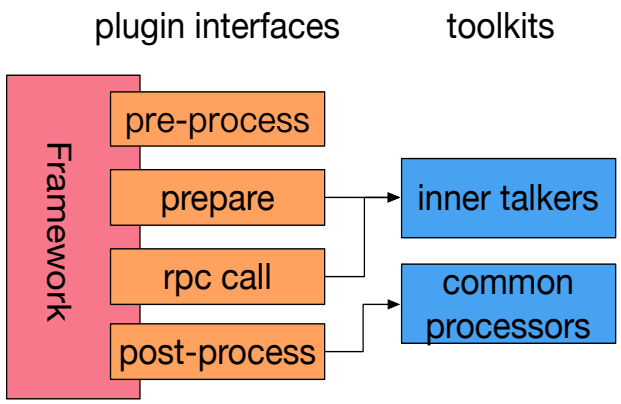
DIF综合处理底层数据，返回定制化的数据视图

DDF高效简单处理本地数据

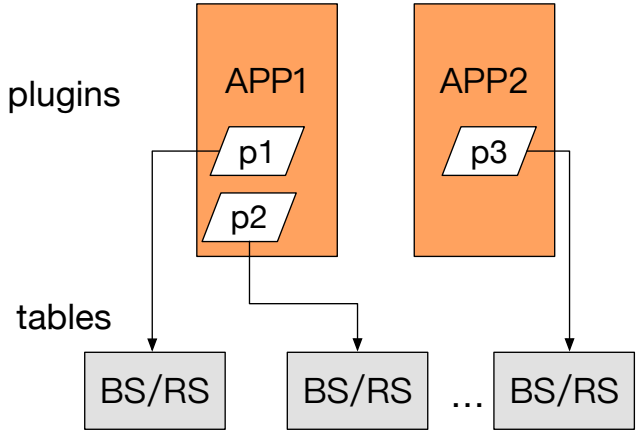
Data Independent Function	Data Dependent Function
A view of database	Local storage procedure
Created by function service	Created by storage service
Can access DIF&DDF&tables	Embedded, access current store
C++ .so/libs, ns isolation	LUA, sandbox isolation
High-performance complex calc	Simple Append/Join/Filter

两类计算DIF和DDF对比表

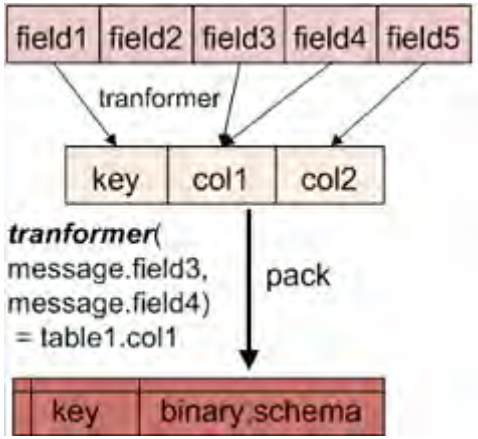
复杂业务计算服务——插件化和配置驱动



计算过程插件化



属性选择可灵活配置不需hard-code



日志更新table配置驱动自动填充

复杂业务计算服务——迭代实验

逻辑迭代需实验(A-B test)

例行软硬件升级

策略实验迭代需求

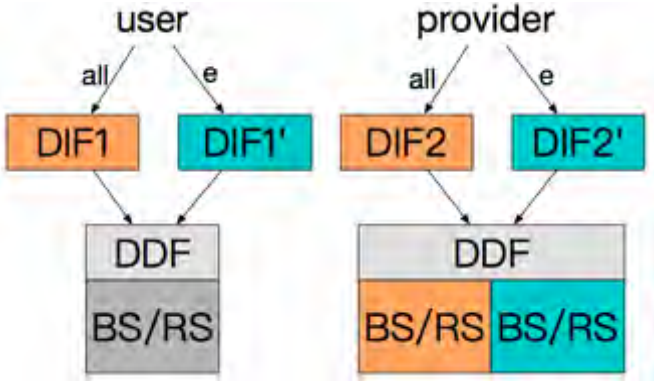
实验方案及种类

流量抽样结果：实验id列表

根据实验id访问不同的function、数据表

User侧实验：function计算逻辑升级、增减属性数据

Provider侧实验：新建实验数据表



Function及table迭代实验

权限控制

数据角色

admin, data provider, data user

两层访问控制

Table privilege

Function privilege

Grant命令

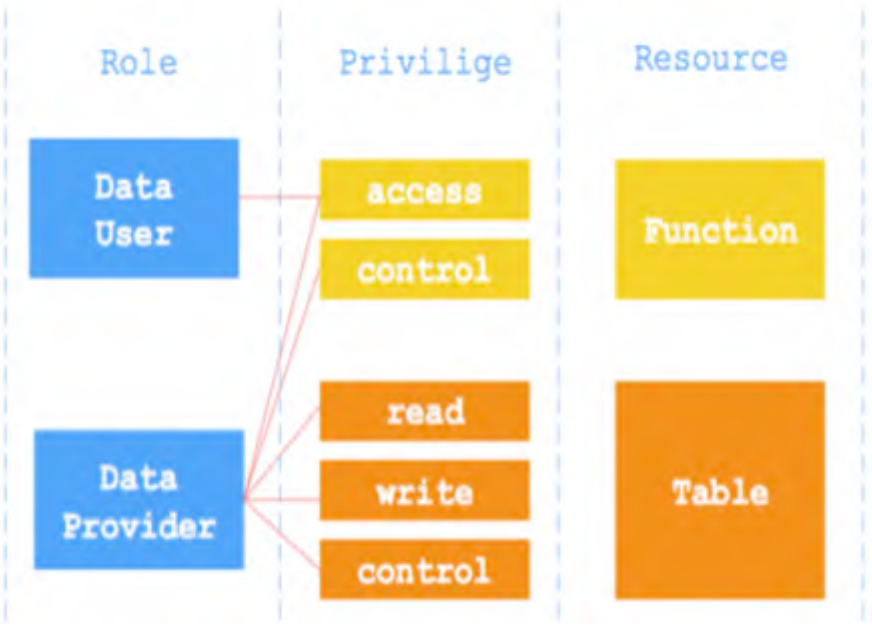
Admin grant c/r/w to providers

Providers grant r/w to users

隔离与审计

按应用方隔离部署

流量监控与控制，访问记录，审计存储、计算资源



两层权限控制保证数据访问安全

应用效果

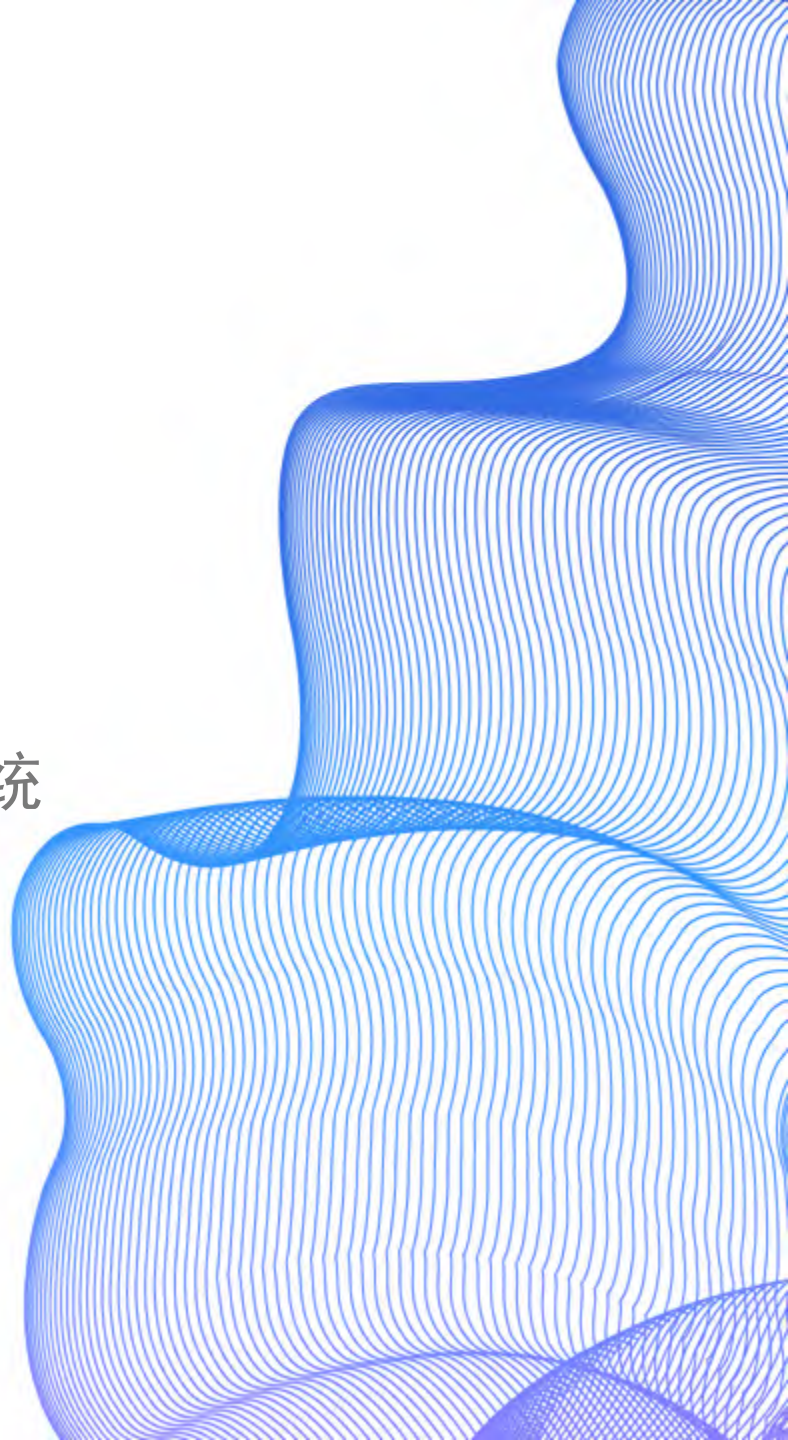
应用面、性能及安全性大幅提升

节约1/3机器资源

系统可用性保障

THANK YOU

SimpleDB: 高性能在线数据服务系统



Baidu 大数据