www.qconferences.com
www.qconbeijing.com

伦敦 | 北京 | 东京 | 纽约 | 圣保罗 | 上海 | 旧金山

London · Beijing · Tokyo · New York · Sao Paulo · Shanghai · San Francisco

QCon全球软件开发大会

International Software Development Conference

在线业务数据框架的探讨

2013/4 金山云 杨钢

服务端软件开发的本质

- 数据的存储和处理
  - 结构化数据和非结构化数据
  - Scale Out / 大数据量
  - Caching / 读写分离
  - 数据一致性
  - 数据冗余和故障修复

数据分类

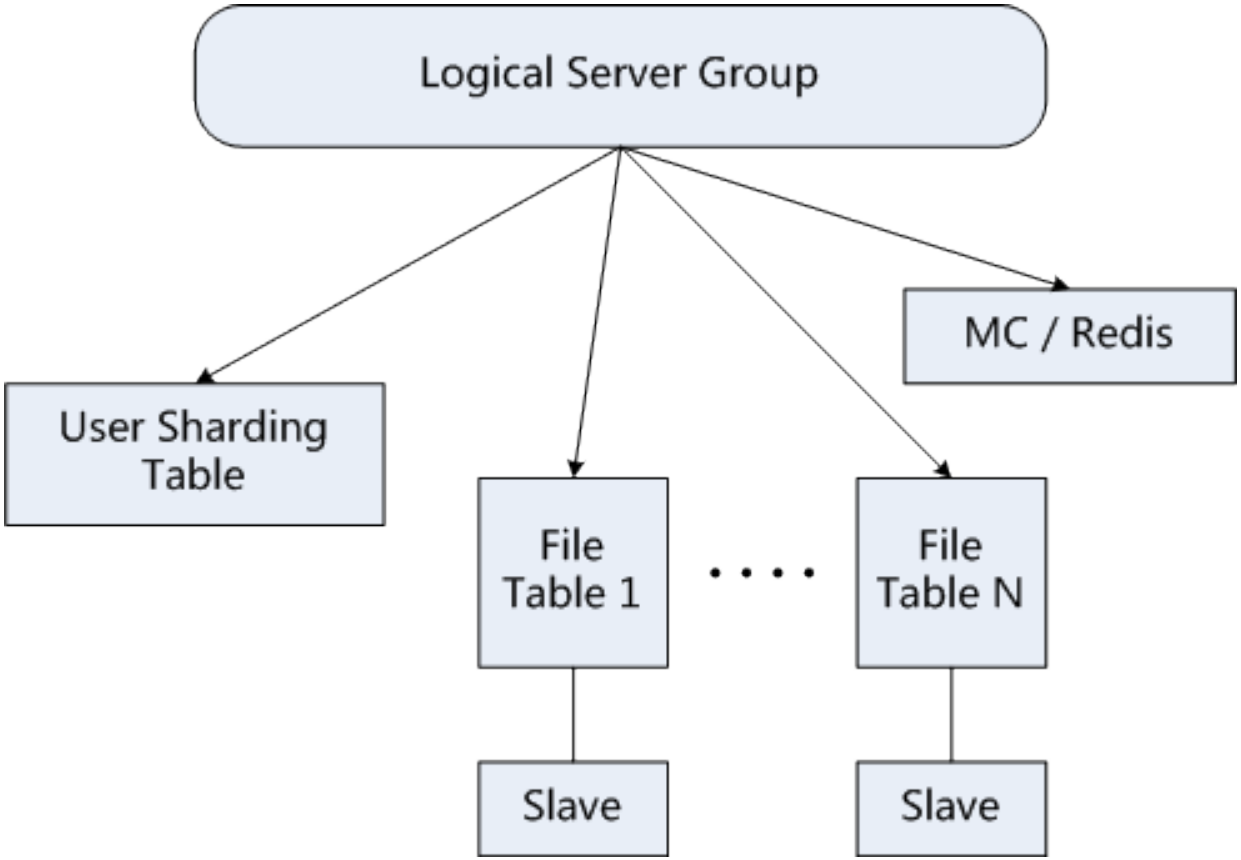
- 结构化数据
  - 在线数据：响应用户发起的操作，通常期望在最多1s内返回结果
  - 离线数据：响应批处理操作，通常数据量巨大，执行时间较长
- 非结构化数据
  - 业务数据
  - 载体数据：例如承载结构化数据库的分布式文件系统，云主机的块设备

结构化数据

业界通行做法

- 教科书方案，来自Flickr
  - Cache + MySQL
  - 分表（水平/垂直）
  - 读写分离
  - Master-Slave / Master-Master
  - ? MySQL Partition

金山快盘案例 V1



主流 NoSQL

- MongoDB
  - MongoDB核心贡献者：不是MongoDB不行，而是你不懂
- Cassandra
- 其他

NoSQL 简单比较

金山云IDF

- 技术思路
  - 定制设计相对于通用设计，通常有明显的性能（价格比）优势
  - 对数据的使用是有设计模式的
  - 采用渐进的步骤达到目标
- 设计目标
  - 为超大数据集合提供一个可定制和装配的数据框架，在实现服务稳定性、数据安全和可扩展性目标的基础上，提供远高于通用产品的运行时性能

金山云IDF

- 迁移路径
  - 新业务使用 Cache + MySQL 的经典模式，以降低风险和提高开发效率
  - 总结数据使用规律，先尝试 SQL 调优
  - 以表为单位实施 IDF 迁移计划（这里指单一表结构在所有集群上的数据量之和大到一个阈值）
  - 根据数据的使用模式选择 IDF 提供的模板，包含组件和迁移路径

IDF 简单架构

- 设备层
  - 提供为某种使用行为（Behavior）定制优化的物理设备
- 资源层
  - 提供统一的资源抽象和管理功能，特别是内存和外存
- 组件层
  - 提供可组装的 IDF 组件（软件）以定制优化细节
- 集成层
  - 用于拼接组件实现完整的数据仓库实现

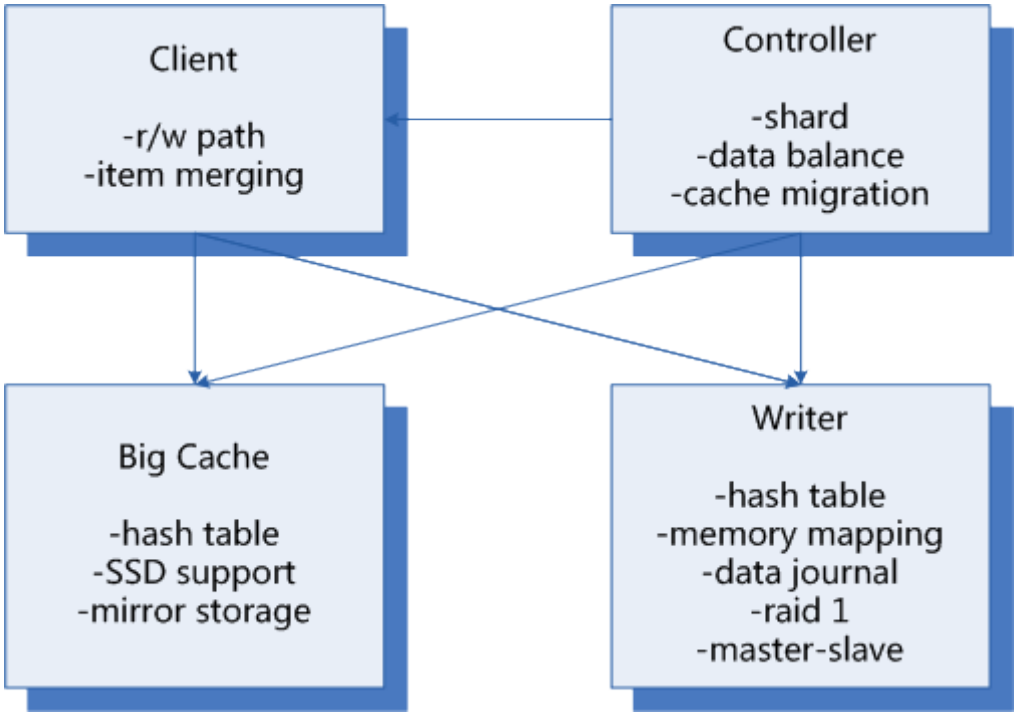
IDF 组件

- 容器组件
  - 提供一种特定数据结构，它经过针对设计场景的高度优化
- 查询组件
  - 在指定容器上实现较复杂的查询行为，以及查询路径的配置
- 存储组件
  - 提供用于容器数据安全的持久化存储组件，包含故障修复能力
- 调度组件
  - 完成数据的分区、查询调度和数据动态迁移、平衡等

金山快盘案例 V2

- 某 Key-Value 表
  - 读写均匀，不存在数据热点
  - 读写比例约为 5:1
  - 整体 QPS 在每秒10-20万次级别
  - 无事务要求，不与其他表构成强一致性要求
  - 对记录没有修改操作，删除操作由可控的任务完成
- 实施效果
  - 方案实施中，目前按照 MySQL 服务器 1/8 用量运行
  - 进一步方案经测算希望到达 1/20 用量

案例V2 IDF装配图



案例V2 IDF装配件

- Controller
  - 使用简单 Hash Sharding 策略，在扩容时采用 Batch Operation 一次性完成
- Client
  - 读路径 Big Cache -> Writer
  - 写路径 Writer；无锁
- Big Cache
  - 硬件使用存储服务器 + 桌面级SSD
  - 软件采用 Big Cache Hash Table 组件簇，安全性采用 Mirror
- Writer
  - 硬件使用带 RAID 1 的 SAS 硬盘，配置较大内存
  - 使用 Writer Hash Table 组件簇，持久化采用 Mapping + Log

金山快盘案例 V3

- 某 Key-Value 表
  - 读写均匀，不存在数据热点
  - 读写比例约为 5:1
  - 整体 QPS 在每秒5-20万次级别
  - 无事务要求，不与其他表构成强一致性要求
  - 对记录有修改和删除的操作

案例V3 IDF装配件

- Controller
  - 使用简单 Hash Sharding 策略，在扩容时采用 Batch Operation 一次性完成
- Client
  - 读路径 Writer -> Big Cache
  - 写路径 Writer；行级锁
- Big Cache
  - 硬件使用存储服务器 + 桌面级SSD
  - 软件采用 Big Cache Hash Table 组件簇，安全性采用 Mirror
- Writer
  - 硬件使用带 RAID 1 的 SAS 硬盘，配置较大内存
  - 使用 Writer Hash Table 组件簇，持久化采用 Mapping + Log

谢谢大家

www.infoq.com/cn

InfoQueue

@InfoQ

infoqchina

软件
正在改变世界！