大云NOSQL系统设计思考

--NOSQL在电信行业的应用

郭磊涛

guoleitao@chinamobile.com;

微博：*@ltguo*

2011-12-02

EXPO 2010 SHANGHAI CHINA

中国移动通信 CHINA MOBILE

中国移动技术创新引擎

提纲

1. NoSQL及其应用情况
2. 大云NoSQL系统研发需求
3. 大云NoSQL系统设计思路探讨

NoSQL⇔Not Only SQL / Non-relational

一种为满足特定应用需求而设计的结构化存储系统，其不保证关系数据库的ACID特性以及join操作，但是一般支持动态table schema定义并具有更好的扩展性。

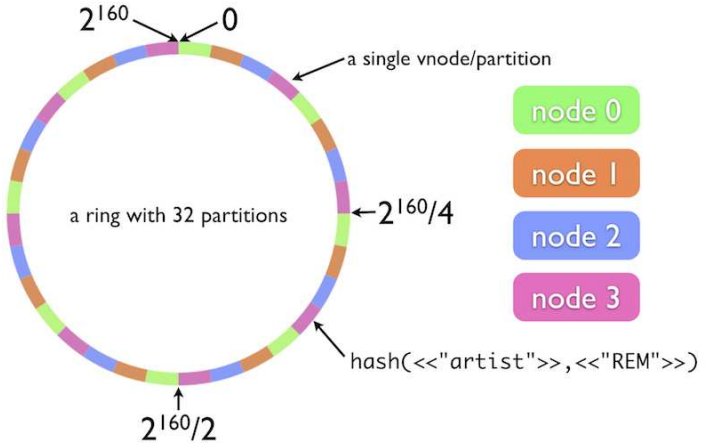
按照数据模型来分类

- 简单**Key –Value键值对**，通常只提供简单的get/set/delete操作
  - e.g. TC/TT, Dynamo, Voldemort, BDB, Riak, memBase, levelDB, memcacheDB, Tair ...
- **支持行列描述的Key-Value系统**，可以针对column做特定的操作
  - e.g. HBase, Cassandra, Hypertable ...
- **文档数据库**，一般以JSON等文档格式存储组织数据
  - e.g. MongoDB, CouchDB
- **图数据库**，图形关系的最佳存储，提供各种图算法API
  - e.g. Neo4j, FlockDB

➢针对自己具体的应用需求，大量互联网公司开发了NoSQL系统，并对外开源
➢ 开源NoSQL数据库需要经过更多的验证、评估和优化才可以应用到生产系统

NOSQL已经在互联网行业得到广泛应用

Facebook的Social Inbox系统集成了文本消息、IM、Email和SMS，每月需要存储***1350***亿条信息。附件和较大的msg存储在Haystack中，其余信息存储在**HBase**中，总存储量为***25TB/月***



Amazon有多种业务使用**Dynamo**系统，例如查询卖家排行、购物车、用户偏好查询、产品目录查询。同时，推出了**SimpleDB**的key-value服务

- 大量互联网公司均在其系统中采用开源或自研的NoSQL
  - 大部分系统提供数据最终一致性
  - 对数据读写实时性要求一般
  - 有的应用允许NoSQL存在短暂的不可访问的现象

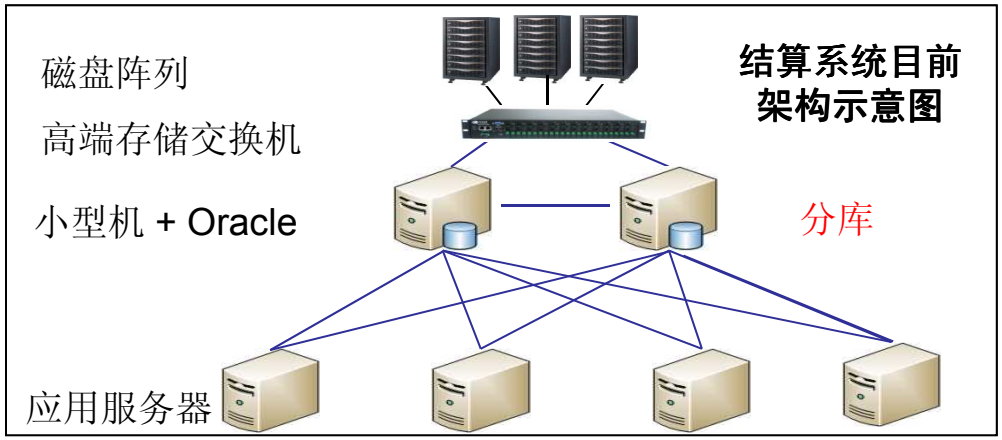
电信行业交易型结构化数据存储场景

- 电信行业交易型结构化数据存储的举例：
  - **话单结算系统**
  - 业务描述：省间结算
  - 系统架构：利用小型机+Oracle数据库存储和处理结算话单
  - 数据处理方式：*实时批处理*插入、查询和删除。高吞吐率，低延时
  - 数据库和集中存储系统IO压力大
  - **清帐单查询系统**
  - 业务描述： 用户清帐单的存储与查询
  - 系统架构：账单主要存储于Oracle数据库，清单以文件方式存储
  - 数据处理方式：*低延迟数据导入、查询*和导出
  - 需要提供大量并发查询，同时需要统计分析功能
  - **移动互联网应用**
  - 业务描述：图书按整本、章节、分册进行订购
  - 系统架构：小型机+Oracle数据库
  - 数据处理方式：*实时插入和查询*，高吞吐率，低延时
  - 订购关系永不删除，数据持续增长，数据库存储和处理压力大

电信业交易型结构化存储系统，仍然以传统RDBMS为主，但是存在性能和可伸缩性瓶颈

NoSQL的适用场景之一：分布式实时批处理

- 计费结算系统现状



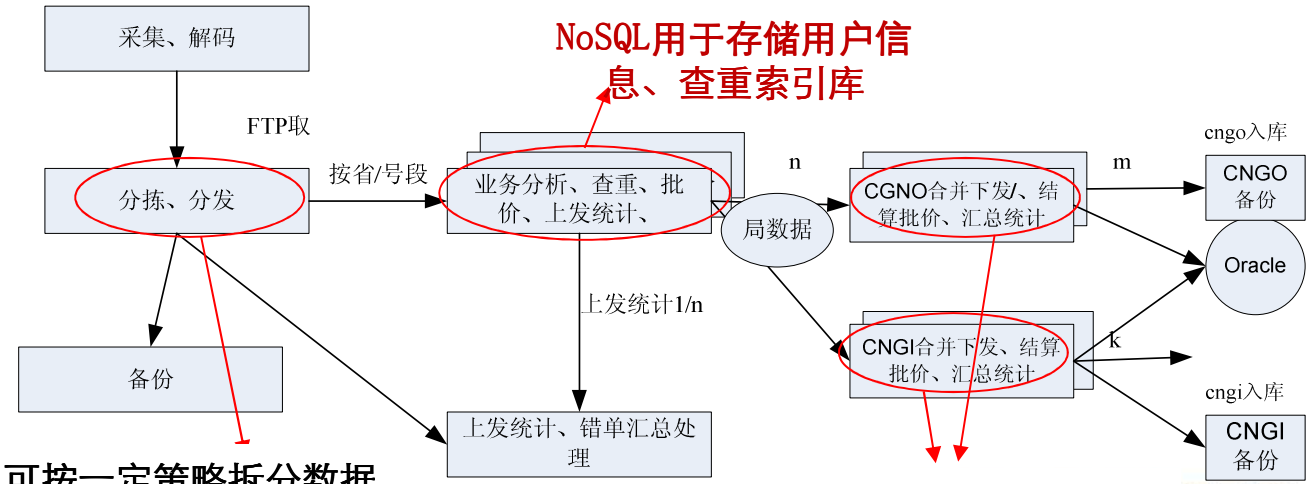
主要技术难题：数据库和存储系统I/O

1. 数据库：计费话单每月几千亿条，分库使扩容更困难。
2. 集中存储使I/O压力增加.

可以采用某种策略对话单数据进行拆分，多节点方式并行处理，计算和存储都实现水平扩展

对NoSQL的需求

- 实时批量数据加载、查询和删除能力
- 读写比率相当
- 数据强一致性



NoSQL的适用场景之二：订购关系存储

- 存储某业务的订购关系
  - 业务一旦订购，用户可永久使用。订购关系数据只增不删
  - 去年该业务订购关系已达20亿左右，每天新增1200万
    - 按当前增长速度计算，目前订购关系已达60+亿条。随着业务的增长，几年后订购关系将达到500亿条左右。
  - 数据操作
    - 90%为订购关系鉴权操作，9%为insert，1%为统计查询
    - 高并发，响应时间毫秒级
  - 目前状态：小型机+Oracle，数据存储量大，扩容成本高

对NoSQL的需求

- 海量数据存储
- 高并发低延时
- 读多写少
- 数据强一致性

电信领域对采用NoSQL仍然存在一些顾虑

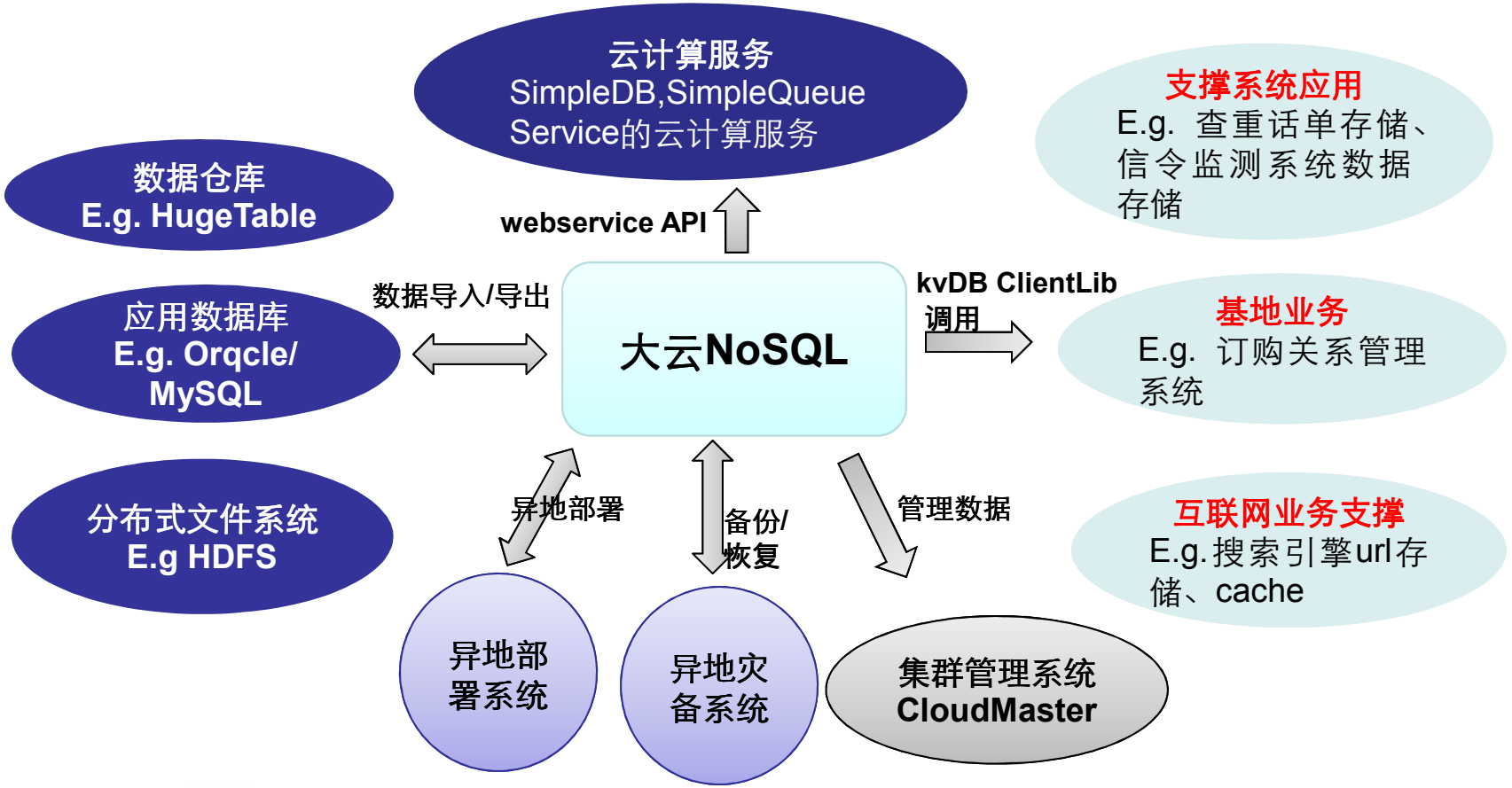
- 用户对采用NoSQL存在一些顾虑
  - 编程接口的改变
    - 现有电信行业交易型结构化数据存储应用基本全部采用SQL，需要修改现有应用才能使用NoSQL
  - 数据一致性
    - 大部分开源NoSQL系统提供数据最终一致性保证，而电信应用要求数据读写必须准确
  - 对事务的支持
    - 基本上所有NoSQL系统均不支持事务处理
  - 辅助功能
    - 导入/导出工具、快照、安装部署、性能监控等工具

提纲

1. NoSQL及其应用情况
2. 大云NoSQL系统研发需求
3. 大云NoSQL系统设计思路探讨

大云NoSQL系统需求

大云NoSQL是面向电信行业应用支持行列描述的Key-Value系统。它提供高并发、低延时、数据强一致性和可靠保证的交易型Key-Value存储与查询能力



（1）接口需求： 大云NoSQL系统需要提供数据导入/导出接口，与现有系统对接。同时，提供系统异地部署和异地灾备的能力，从而保证电信级的服务可靠性

大云NoSQL系统需求

(2) 系统性能

- 系统具备存储千亿条记录的能力
- 在高并发下，数据插入和查询响应时间为毫秒级
  - e.g. 订购关系鉴权应用中，需要在10ms内确认某用户是否订购某项业务

(3) 数据强一致性

- 要求在任何情况下均需保证数据的强一致性
  - e.g. 在计费和订购关系鉴权等应用场景中，不一致的数据会造成计费错误或用户的业务不可使用

(4) 系统扩展性好

- 系统具有scale up和scale out的能力
  - e.g. 提升节点硬件配置，系统性能提升。增加服务节点数，性能线性提升。

(5) 可靠性

- 系统可靠性要求更高

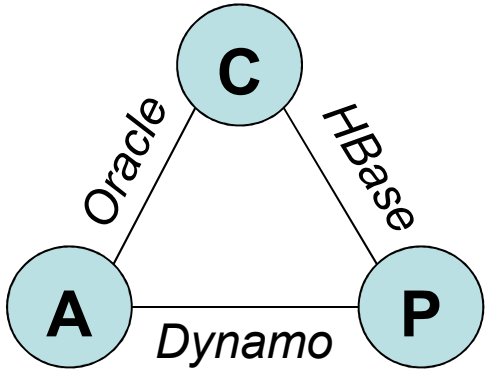
(6) 其他需求

- 具备基本的统计分析能力，以应对各种统计和查询的需求

提纲

1. NoSQL及其应用情况
2. 大云NoSQL系统研发需求
3. 大云NoSQL系统设计思路探讨

大云NoSQL系统设计：CAP理论



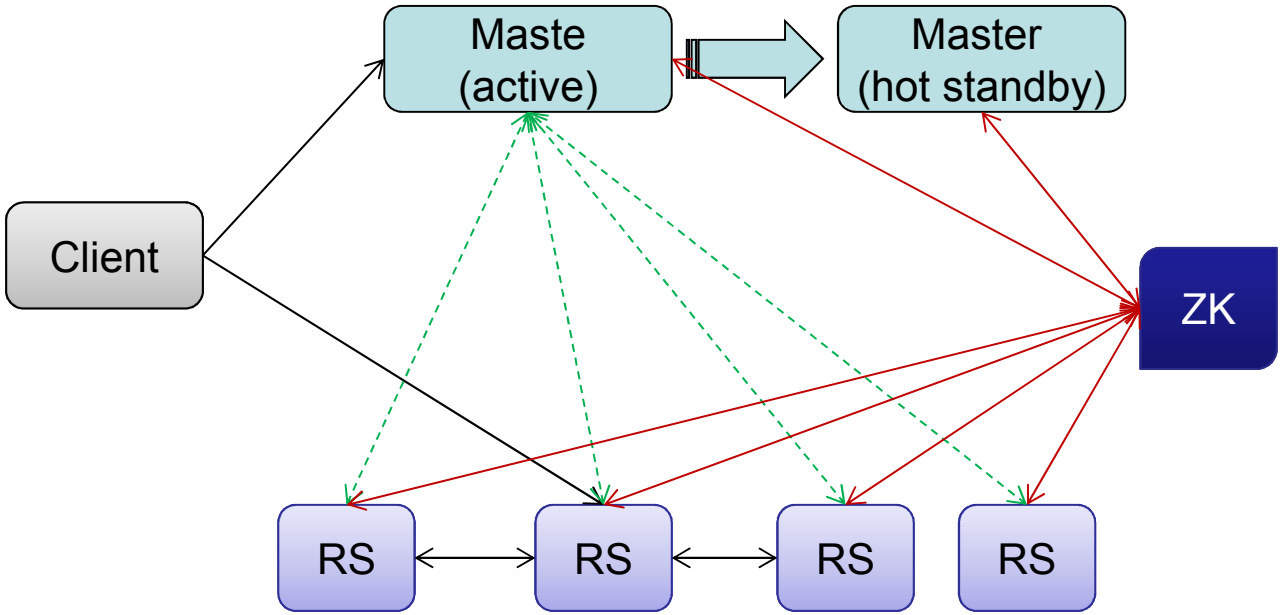
- CAP理论最早是在2000年由Berkeley的Eric Brewer教授提出。此后，MIT的Seth Gilbert和Nancy Lynch，理论上证明了Brewer猜想是正确的，CAP理论在学术上正式作为一个定理出现

- 一致性（C）：在分布式系统中的所有数据备份，在同一时刻是否同样的值
- 可用性（A）：在集群中一部分节点故障后，集群整体是否还能响应客户端的读写请求
- 分区容忍性（P）：集群中的某些节点在无法联系后，集群整体还能继续进行服务

CAP理论就是说在分布式存储系统中，最多只能实现上面的两点

我们的设计思路：无法避免P，但需要保证数据的强一致性(C)

大云NoSQL系统架构

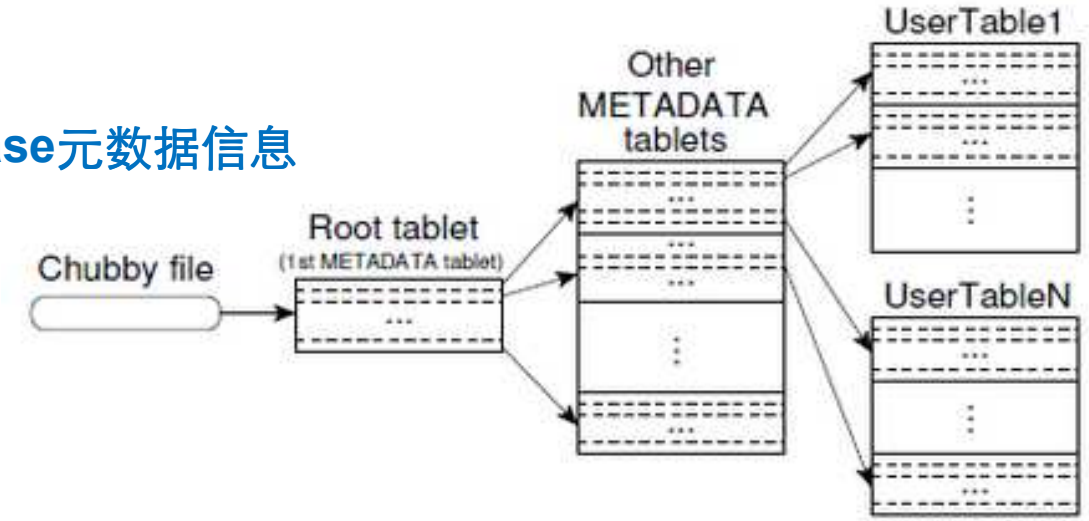


- Master: 中心控制节点，提供hot standby Master
- ZK：节点状态管理
- RS：RegionServer，存储和管理表的一部分数据
- Client：数据读写客户端

大云NoSQL系统设计：海量数据存储

- 如何解决海量数据存储？ 数据分区，分布式存储与管理
- 数据该如何分区？
  - *连续范围分区*: HBase, Hypertable,
  - *一致性哈希:* Dynamo, Voldemort ,Cassandra, Riak

HBase元数据信息



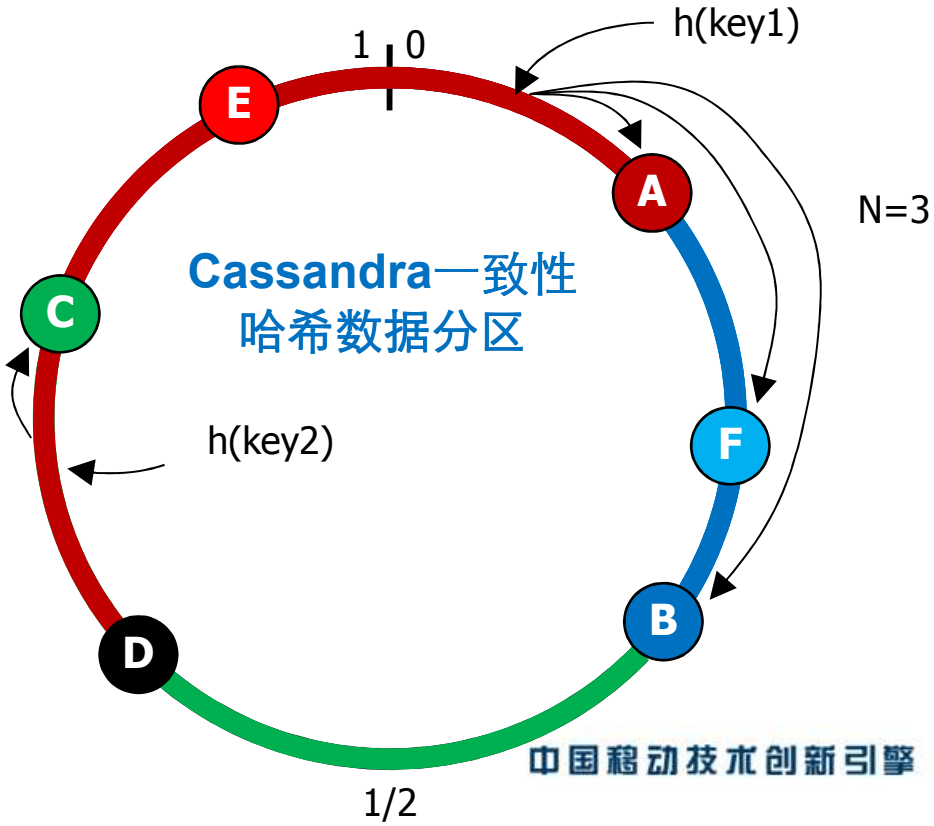
连续范围分区：

- key值按连续的范围分段，每段数据会被指定保存在某个节点上
- 优点：利于顺序扫描数据，易于负载均衡和节点控制管理
- 缺点：需要维护数据分段信息 以及 数据分段分配信息；
  管理节点存在SPOF问题

大云NoSQL系统设计：海量数据存储

一致性哈希：

- 为所有的Key计算一个Hash值，这些Hash值都分布在一个环(Ring)上。每个节点负责环上的一个区间。
- 优点：元数据信息少，依赖Hash函数即可查询到某个key所属的节点
- 缺点：
  - 只能根据给定key进行查询，不易实现数据顺序扫描
  - 不易负载均衡
  - 节点的加入和退出会立刻造成元数据和数据的更新/迁移，系统可用性存在风险



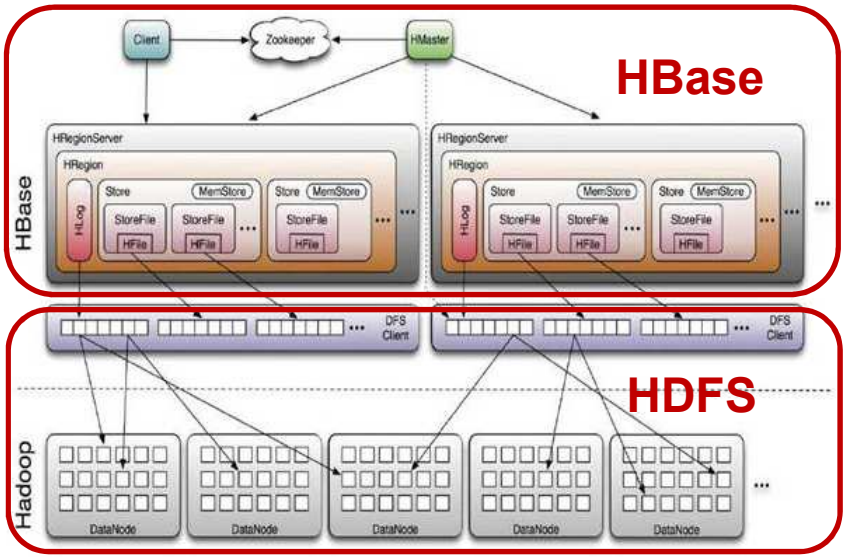
大云NoSQL的设计思路

•要求系统架构简单，对大量节点易于管理和控制

• 因此，采用数据*连续范围分区*

大云NoSQL系统设计：高性能数据读写

- 分解为两个问题
  - 如何更好的放置数据?
  - 如何更好的读写数据?
- 如何更好的放置数据
  - ***Shared-Disk***: HBase, Hypertable
  - ***Shared-Nothing***: Cassandra, memBase ...

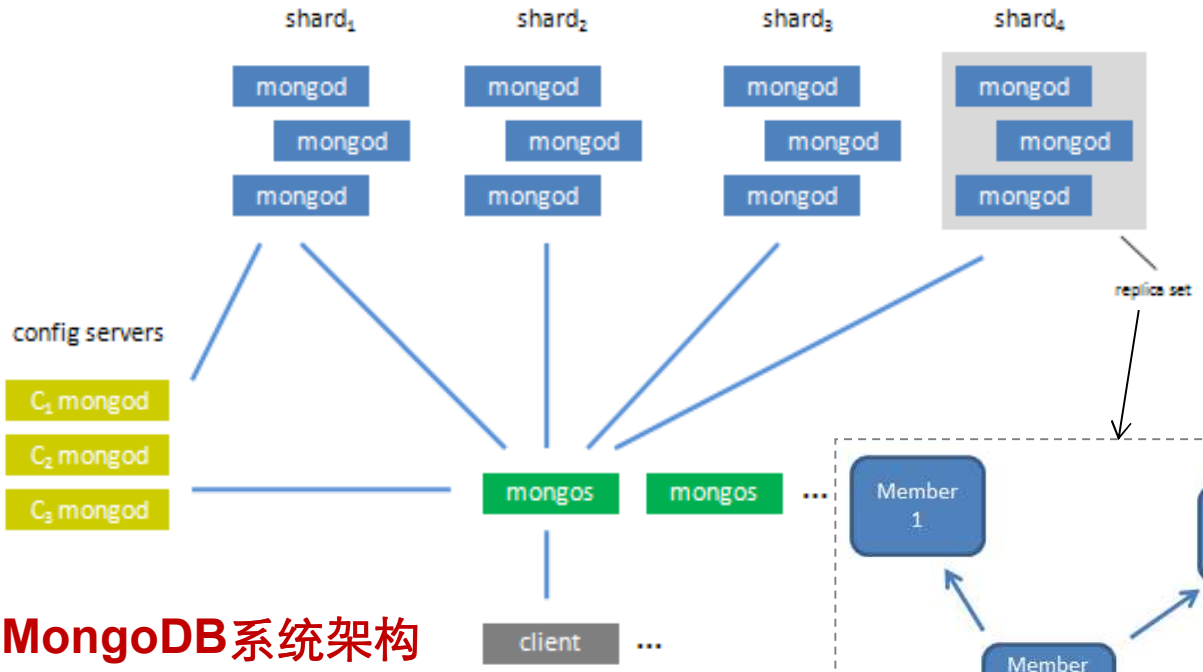


Shared-Disk：

- 节点之间共享存储，例如，HBase的所有RegionServer共享同一个HDFS并行文件系统
- **优点：**数据可靠性交由HDFS，HBase只需实现表管理，实现简单
- **缺点：**HDFS的数据放置策略对HBase透明，不易进行系统优化

大云NoSQL系统设计：高性能数据读写

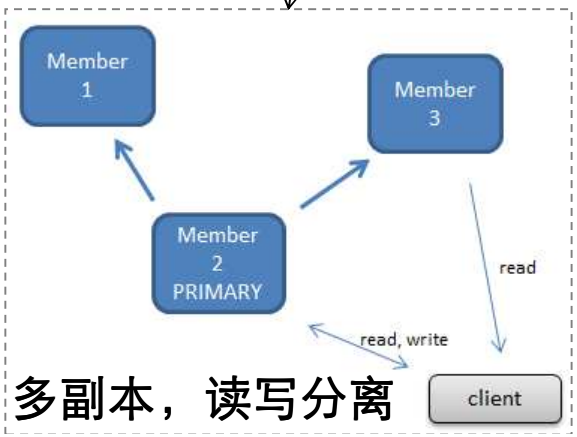
- 如何更好的放置数据：Shared-Nothing
  - 节点之间是独立的，不共享内存或磁盘
  - 优点：每个节点可以控制其管理的数据的位置和读写方式，易于性能优化
  - 缺点：数据可靠性（多副本）需要进行控制，较为复杂



MongoDB系统架构

大云NoSQL的设计思路

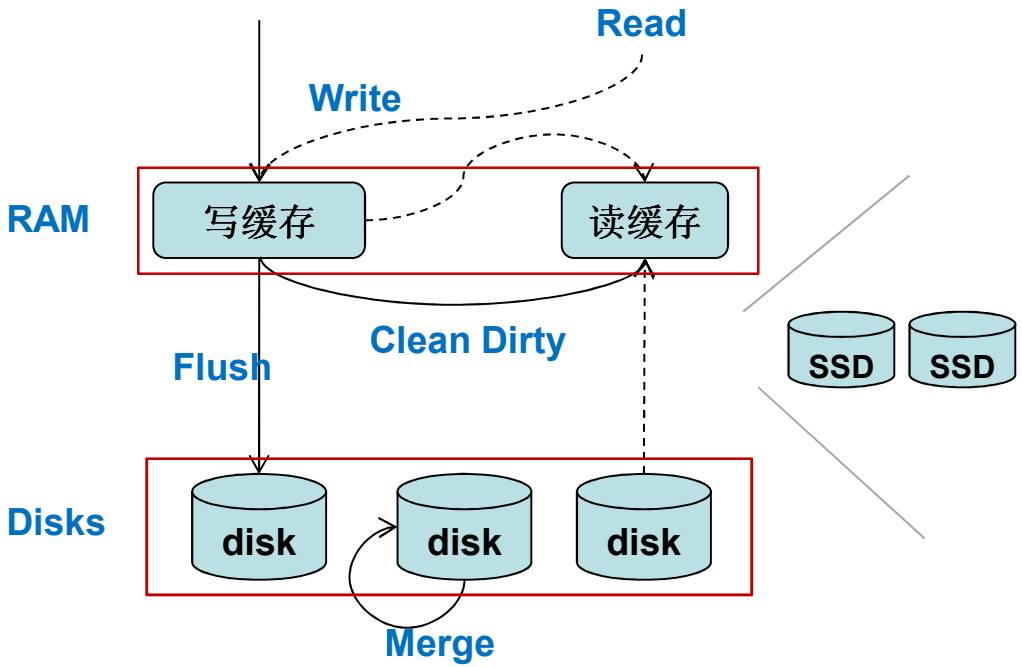
•系统需要提供高性能读写能力，因此需要采用***Shared-Nothing***的架构进行读写优化



多副本，读写分离

大云NoSQL系统设计：高性能数据读写

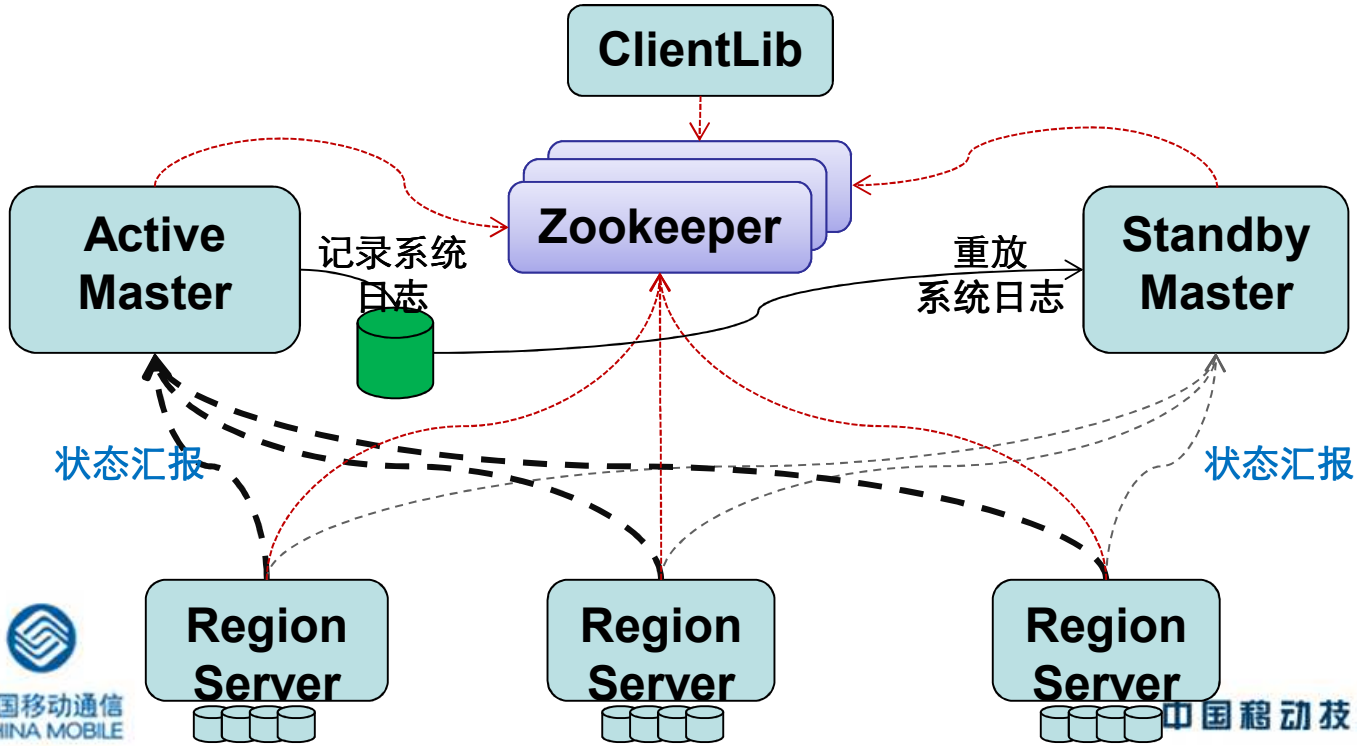
- 如何更好的读写数据？ ➔ 读写缓存和分级存储
  - 写缓存：将随机写转化为顺序写
  - 读缓存：减少磁盘seek/read时间
  - 分级存储：根据数据热度，分别存储在不同存储介质上，缩短查询时间



- 利用FlashCache将SSD作为磁盘的块缓存？
- 将SSD作为内存的扩展？

大云NoSQL系统架构：系统可靠性

- 可靠性包括：数据可靠性和服务可靠性
  - 数据可靠性：在写入数据时，利用2PC写数据副本，保证数据强一致性
  - RegionServer节点(RS)的可靠性：ZK监控RS节点的状态，当节点加入或退出时，Master进行数据的迁移
  - Master节点的可靠性：实现hot standby master，最小化master切换时间



总结

- 大云NoSQL的设计需求
  - 海量数据存储
    - 采用数据*连续范围分区*，进行分布式数据管理
  - 保证数据强一致性
  - 高性能数据插入和查询
    - 采用Shared-Nothing架构，每个节点管理自己的磁盘
    - 通过cache和SSD进行读写哟花
  - 高可靠
    - 借鉴Hadoop NameNode HA的思路，通过Hot Standby Master来最小化主节点切换时间
- 欢迎大家的意见和建议！

欢迎提出宝贵意见！

HTTP://LABS.CHINAMOBILE.COM/CLOUD