数据库系统概论新技术篇

键值对数据库

陈跃国

中国人民大学信息学院

2017年4月

概述

- 本节课讲述键值对数据库系统的概念、应用场景、基本原理、和典型实现案例
- 通过学习加深其与传统数据库应用定位的不同，以及在关键技术上如何应对大数据的数据服务需求

键值对数据库

1 数据服务与键值对数据库

2 键值对数据模型

3 键值对数据库原理

数据服务

❖数据服务(data serving)

服务：指程序化的解决众多个体提出个性化的简单请求。

数据服务：数据的简单读写

数据库事务：也算一种数据服务，但属于mission-critical并可以较为复杂的数据服务

数据服务起源

- 门户网站页面优化布局，需要读取用户众多的属性
- 日志分析，用户画像，写用户属性
- 高并发简单数据读写
- 数据规模大：亿*千
- 属性动态增减

键值对数据库

- 键值对数据库/键值对系统的定义：一个针对关联数组（字典或Hash表）提供高吞吐数据服务（存储、读取和管理）的数据库系统
- 关联数组包含了很多记录，每个记录又有不同的属性
- 系统通过唯一标识记录的键（key）来迅速存储和读取单行记录中的数据，实现对关联数组的高并发读写服务

数据库 v.s. 键值对数据库

❖ **Bigtable: 需求带来新的技术突破**

指标	Database	KV-stores
数据量	TB	PB
吞吐量	<100K TPS	1M TPS
列数	100（固定schema）	1000-10000（可扩展）
复杂性	较为复杂	极为简单
一致性	Mission critical（严格一致性）	马马虎虎（最终一致性）
接口	SQL	API （是NoSQL的一种）
成本	高	低

键值对数据库

1 数据服务与键值对数据库

2 键值对数据模型

3 键值对数据库原理

键值对数据库设计

- ❖为什么不直接使用**Hashtable**？
  - 多列、动态列增减、列族
- ❖为什么不直接使用关系数据库表？
  - 大量空值、高并发
  - 动态列增减、列族

Users	a_1	a_2	…	a_n
u_1				
u_2				
…			c_{ij}	
u_n				

键值对数据模型

- 数据模型
  - 持久化的、分布式的、多维**Hashtable**
  - 键值排序（实现按键的快速数据检索）
  - 列族、**Tablet**（分片）、**Timestamp**

	A	B	C	D	E	F
1					Column Family: 地址	
2	Tablet	Row Name	名字	性别	家庭地址	工作地址
3	Tablet1	310101	小吴	男	SH1	BJ1
4		310102	小朱	女	SH2	BJ2
5	Tablet2	310103	小华	男	SH2	BJ2

数据模型示例

ColumnFamily: Rockets

Key	Value
1	**Name** / **Value** name: Rocket-Powered Roller Skates toon: Ready, Set, Zoom inventoryQty: 5 brakes: false
2	**Name** / **Value** name: Little Giant Do-It-Yourself Rocket-Sled Kit toon: Beep Prepared inventoryQty: 4 brakes: false
3	**Name** / **Value** name: Acme Jet Propelled Unicycle toon: Hot Rod and Reel inventoryQty: 1 wheels: 1

键值对系统操作接口

- API操作接口
  - get(key), put(key, value), delete(key)
  - execute(key, operation, parameters)
  - 指定列的数据读取
  - 有的允许在一定范围内的键值读取
  - rowkey与columnkey

键值对数据库

1 数据服务与键值对数据库

2 键值对数据模型

3 键值对数据库原理

CAP理论

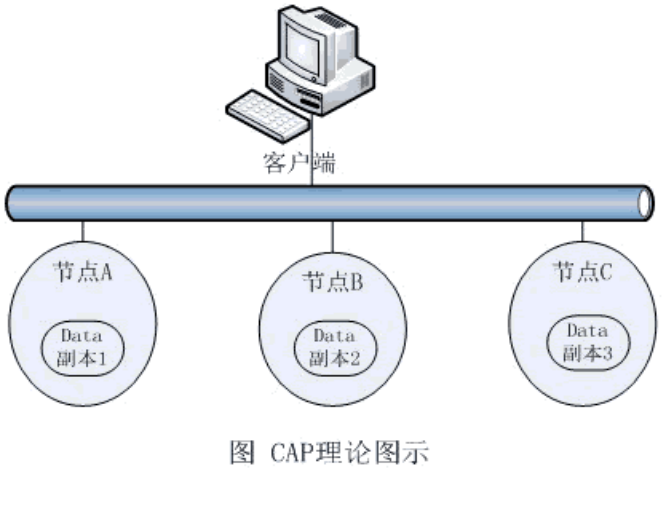
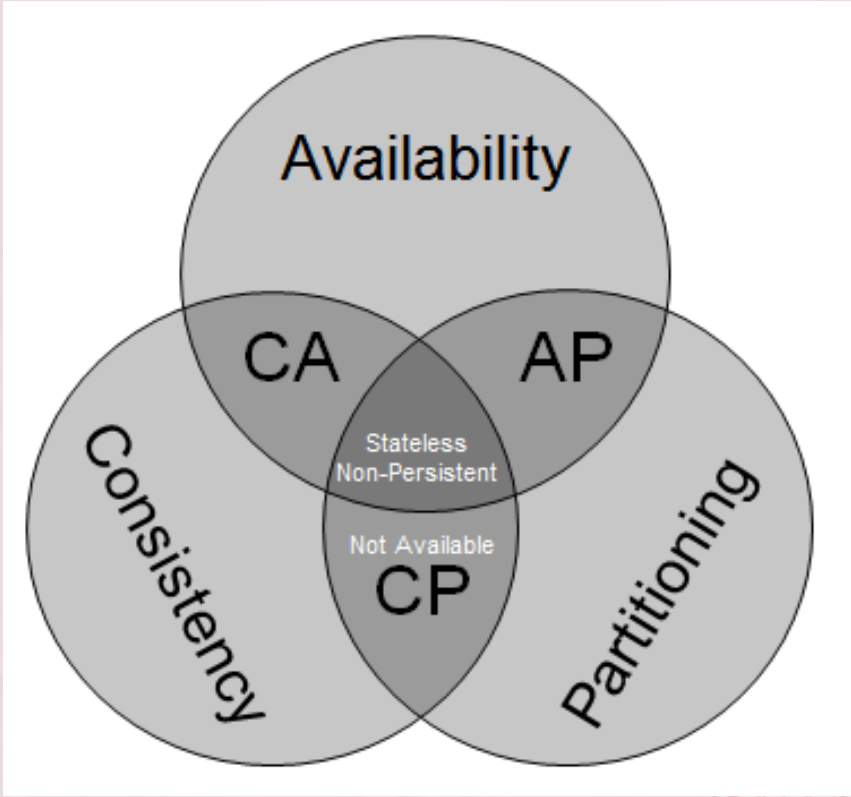


图 CAP理论图示

分布式系统的3个属性
只能满足2个，要舍弃1个



最终一致性

- 很长时间没有数据更新后，最终所有更新会传播到整个系统，所有节点的数据保持一致
- 如果一个节点没有离开服务，最终系统接受的和这个节点数据相关的更新操作会作用在该节点上
- **BASE协议 (Basically Available, Soft state, Eventual consistency), 区别于ACID**
  - 数据的副本之间可能不一致
  - 该数据没有更新，最终副本之间会一致

键值对系统设计考虑

- 牺牲掉以下数据库特性
  - 链接操作
  - group by
  - order by
  - ACID事务处理
  - SQL

换取高性能、高吞吐、可扩展和高容错的键值对数据读写解决方案

HBase系统架构

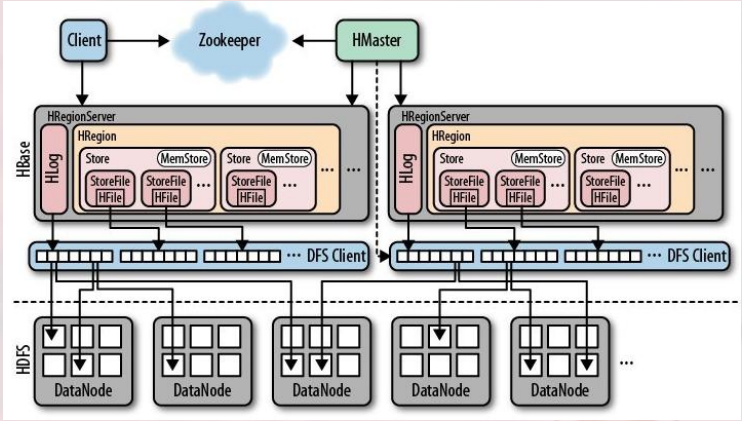
- 当Table随着不断变大后，会逐渐分裂成多份splits，成为regions，一个region由[startkey,endkey)表示，不同的region会被Master分配给相应的RegionServer进行管理

- **Client**：HBase Client使用HBase的RPC机制与HMaster和HRegionServer进行通信；对于数据读写类操作，Client与HRegionServer进行RPC

- **HRegionServer**：用户I/O请求，向HDFS文件系统中读写数据，是HBase中最核心的模块。HRegionServer内部管理了一系列HRegion，每个HRegion对应了Table中的一个 Region，HRegion中由多个HStore组成。每个HStore对应了Table中的一个Column Family的存储

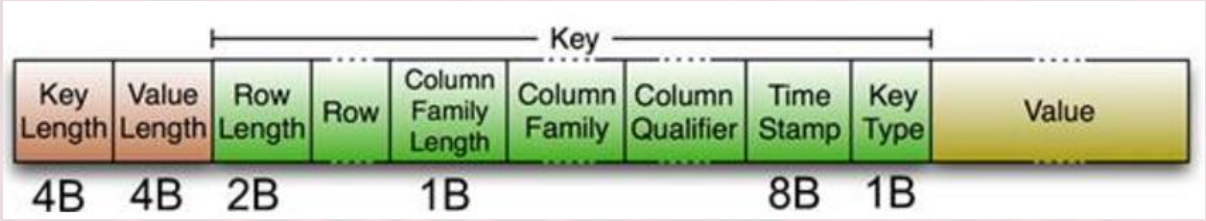
- **HMaster**：HBase中可以启动多个HMaster，通过Zookeeper的Master Election机制保证总有一个Master运行，HMaster在功能上负责Table和Region的管理工作

HMaster
1. 管理用户对Table的增、删、改、查操作
2. 管理HRegionServer的负载均衡，调整Region分布
3. 在Region Split后，负责新Region的分配
4. 在HRegionServer停机后，负责失效HRegionServer 上的Regions迁移



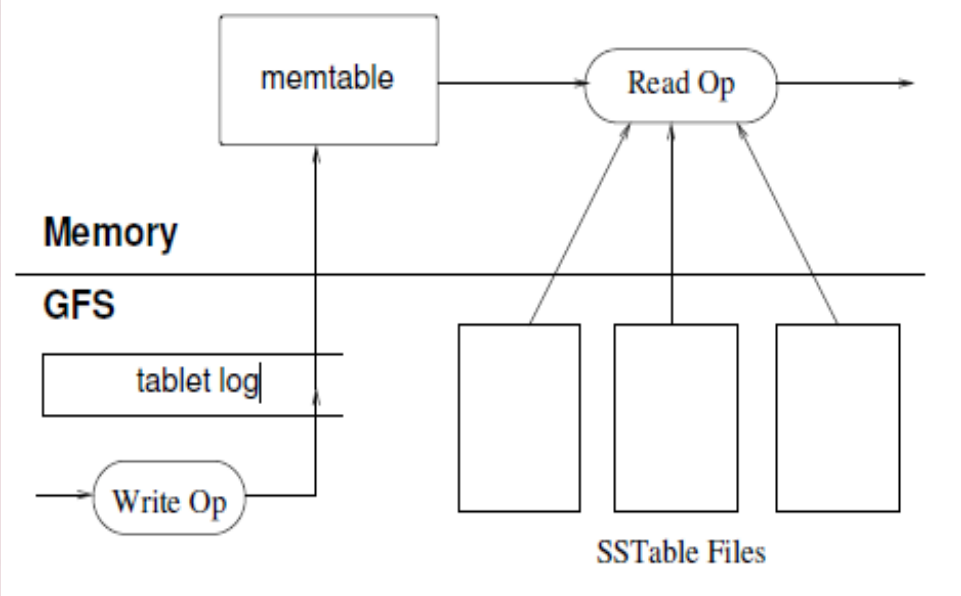
HBase存储格式

- HBase中的所有数据文件都存储在Hadoop HDFS文件系统上
- HFile，HBase中KeyValue数据的存储格式，是Hadoop的二进制格式文件
- HFile里面的每个KeyValue对就是一个简单的byte数组。但是这个byte数组里面包含了很多项，并且有固定的结构
- 开始是两个固定长度的数值，分别表示Key的长度和Value的长度。紧接着是Key，开始是固定长度的数值，表示RowKey 的长度，紧接着是RowKey，然后是固定长度的数值，表示Family的长度，然后是Family，接着是Qualifier，然后是两个固定长度的数值，表示TimeStamp和Key Type（Put/Delete）。Value部分是纯粹的二进制数据



存储模型与实现原理

- 写操作
  - 写日志、同时写入memtable
  - memtable批量持久化
- 读操作
  - 内存和磁盘都要读
  - 键值排序索引支持
- 内部Compaction操作
  - 涉及memtable和次SSTable的整合，回收修改/删除记录所占的空间



典型键值对数据库

- 大Hashtable
  - Amazon S3 (Dynamo), Voldemort, Scalaris
- 列族
  - Cassandra, BigTable, HBase, Sherpa/Pnuts, Hypertable
- 内存型
  - Redis, Memcached…

NoSQL数据库

- NoSQL： Not Only SQL
  - 键值对数据库
  - 图数据库
  - 文档数据库
  - MapReduce（不是数据库）
- NewSQL
  - 保证ACID，扩展性好

小结

- 大数据带来新的数据服务需求
- 追求更好的数据服务吞吐率就需要牺牲一些数据库系统的特性，在特定应用下，有些特性是可以牺牲的
- 典型应用
  - 用户画像管理
  - 非结构化数据的元数据管理
  - 多版本数据管理
  - 键值对数据管理