Big Table

目标

广泛的适用性（url、网页、图像等结构化数据的存储系统）

可扩展性

高性能

高可用（GFS保证）

数据模型

稀疏的，分布式的，持久的多维有序映射（Map）

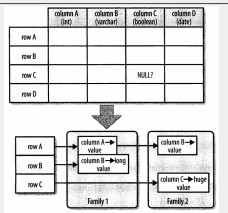
Map由行键，列键和时间戳索引

Map中的每个值都是一个未解释的字节数组

(row:string, column:string, time:int64) → string

行

- 行键，任意字符串（最大64KB）
- 单行原子读写一行
- 按字典序排列（网站url相近的key也相近）
- 一段范围的行叫tablet，、调度和负载均衡单位



列族

- 列键：family:qualifier
- 列键被（按类型）分组的集合
- 列族中的列类型相似，可压缩
- 列族的数量少，列族中列键的数量有无数

数据库反范式设计

SSTable文件格式

- 键到值的持久的，有序的不可变映射
- 读取操作
  - 按key读
  - 按key range读
- 由一系列块构成，文件末尾存在一个块索引，用于只扫描磁盘特定的块
- 依赖chubby提供分布锁服务，对文件进行原子的读写（对应开源实现是zoopKeeper）

API

建/删表，增加/删除列族，修改表/列族元数据(访问权限等)

按行键增/删，查找、遍历

单行事务

MapReduce

架构

client

master

tablet服务器

- 增量数据memtable
- 只读基线数据SSTable

相同的行键有序的结构，便于读取时合并数据

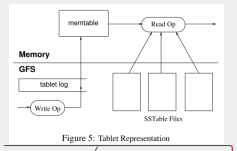


Figure 5: Tablet Representation

WAL（成组提交）+SStable的索引恢复memtable

合并

- memtable->SStable 小合并（minor compaction）
- 一些SStable+memtable->SStable 合并压缩（merging compaction）
- 将所有的SStable重写为一个SStable，并删除delete的行 大合并（major compaction）

保留delete标记的行

优化

局部组内的列族的数据单独存放在一个SStable，隔离其他组和列族

局部组（Locality groups）

- 设置属性，如长期cache在内存

压缩

- 自定义适合局部组数据类型的压缩方案（甚至不压缩）

缓存

- 扫描缓存（适用重复读）
- 块缓存（同行不同列）

局部组的布隆过滤器 减少磁盘扫描

日志

- 以tablet服务器为单位写日志文件
- 利于成组提交
- 并行化排序（行键、日志序列号）

日志恢复

- 排序后，按需分发给需要恢复的tablet服务器，避免重复扫描日志文件

日志写（解决抖动，保证延迟）

- 双日志写线程（备用线程）
- 日志序列号，消除重复日志

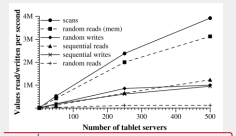
tablet迁移

1. 小合并，减少日志回放
2. tablet停止服务
3. 小合并（极快），合并第一次合并时产生的增量
4. 移动到另一个tablet服务器，无需回放日志

利用不变性

- SSTable并发读取
- memtable的每行使用写时拷贝，并发读写
- SStable过期删除
- 子tablet共享父SStable文件

实验



性能

- 顺序读
- 顺序写
- 扫描
- 随机读
- 随机读（cache到内存）

随机读取比所有其他操作慢一个数量级或更多。每次随机读取都涉及通过网络将64KB SSTable块从GFS传输到tablet服务器，其中只使用一个1000字节的值。（等价1200次读tablet，缓解方法：调整块大小为8KB）随机和顺序写比读更好，随机和顺序写没有明显差异，随机写每个tablet服务器只写一个日志文件，并且可成组提交，顺序读由于块缓存，比随机读更好

适合在随机、顺序写和顺序读场景

扩展性

非线性扩展

- 负载不均衡
- 负载是动态的，无法妥善调整，调整tablet需要一定的不可服务时间（1s）
- 网络饱和（随机读最快到达）

经验教训

故障

- 标准网络分区和故障停止
- 内存和网络损坏，大时钟偏差，挂机，扩展和非对称网络分区，我们正在使用的其他系统中的错误（例如Chubby），GFS溢出配额，计划内和计划外硬件维护。

解决方法：协议

- 明确功能使用方式前，延迟添加新功能
- 正确的系统级监控
- 简单设计（代码和设计清晰度）

参考文献

《Hbase权威指南》