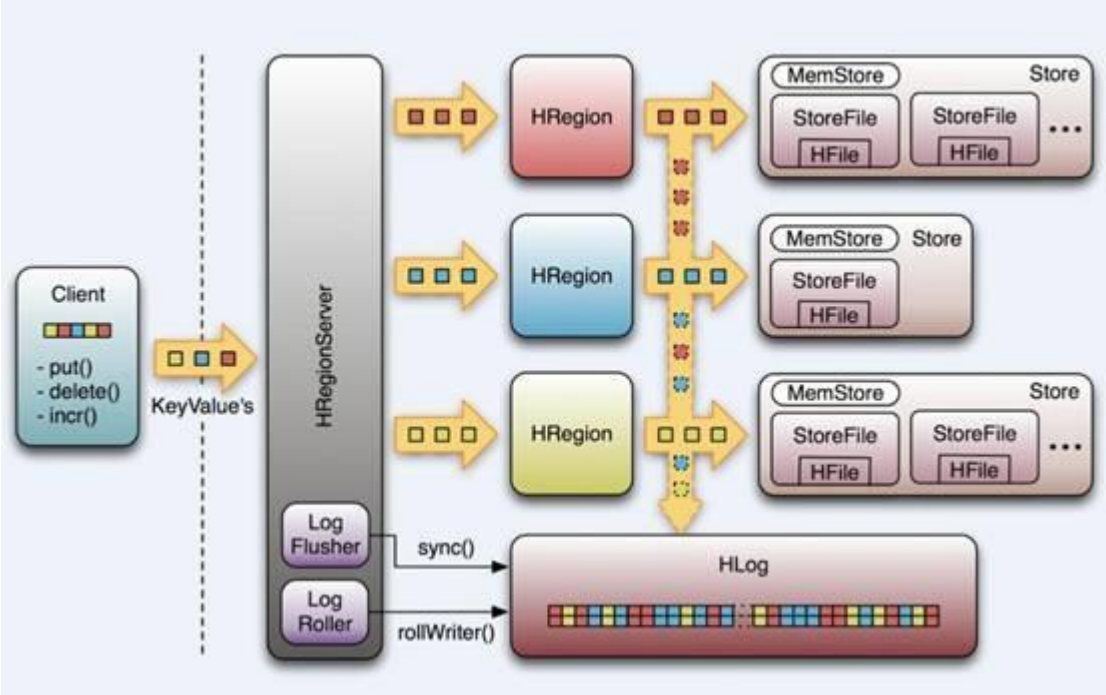


Hadoop数据分析平台 第6周

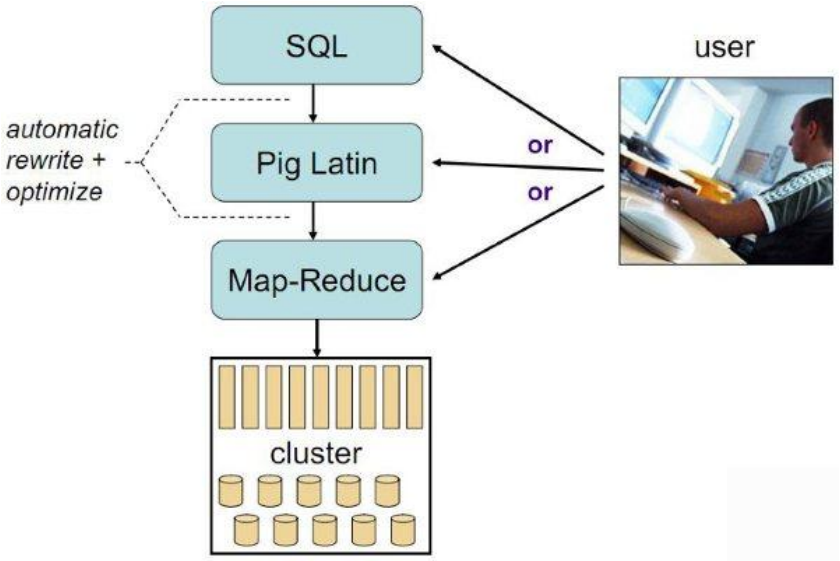
2012.10.14

Hadoop家族

- Pig
- Zookeeper
- Hbase
- Hive
- Sqoop
- Avro
- Chukwa
- Cassandra

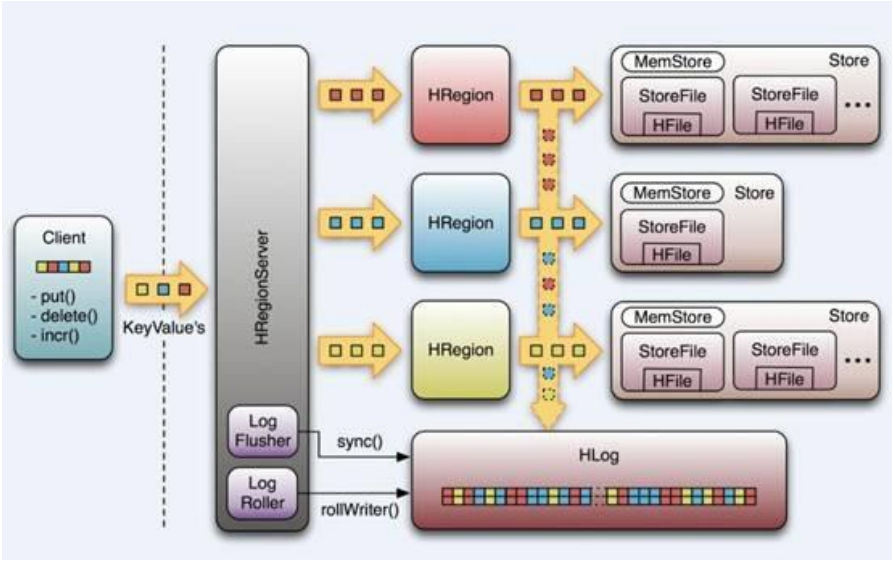
Pig

- Hadoop客户端
- 使用类似于SQL的面向数据流的语言Pig Latin
- Pig Latin可以完成排序，过滤，求和，聚组，关联等操作，可以支持自定义函数
- Pig自动把Pig Latin映射为Map-Reduce作业上传到集群运行，减少用户编写Java程序的苦恼
- 三种运行方式：Grunt shell，脚本方式，嵌入式



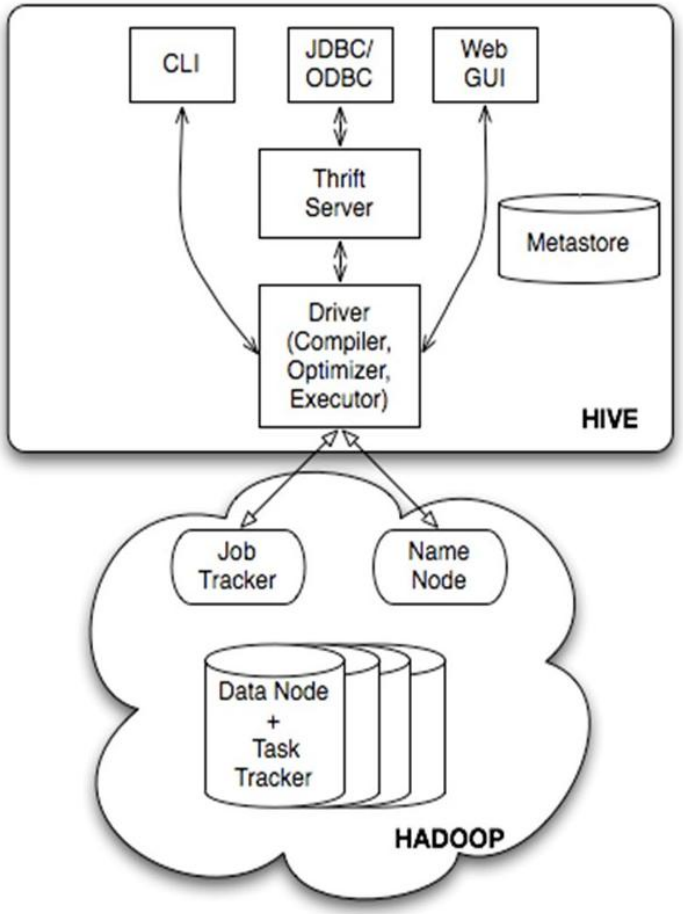
Hbase

- Google Bigtable的开源实现
- 列式数据库
- 可集群化
- 可以使用shell、web、api等多种方式访问
- 适合高读写（insert）的场景
- HQL查询语言
- NoSQL的典型代表产品



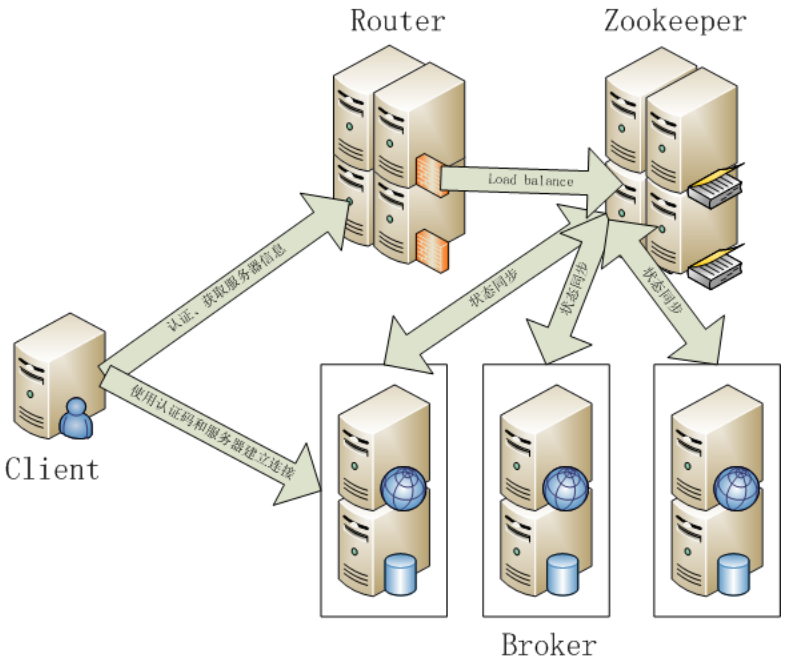
Hive

- 数据仓库工具。可以把Hadoop下的原始结构化数据变成Hive中的表
- 支持一种与SQL几乎完全相同的语言HiveQL。除了不支持更新、索引和事务，几乎SQL的其它特征都能支持
- 可以看成是从SQL到Map-Reduce的映射器
- 提供shell、JDBC/ODBC、Thrift、Web等接口



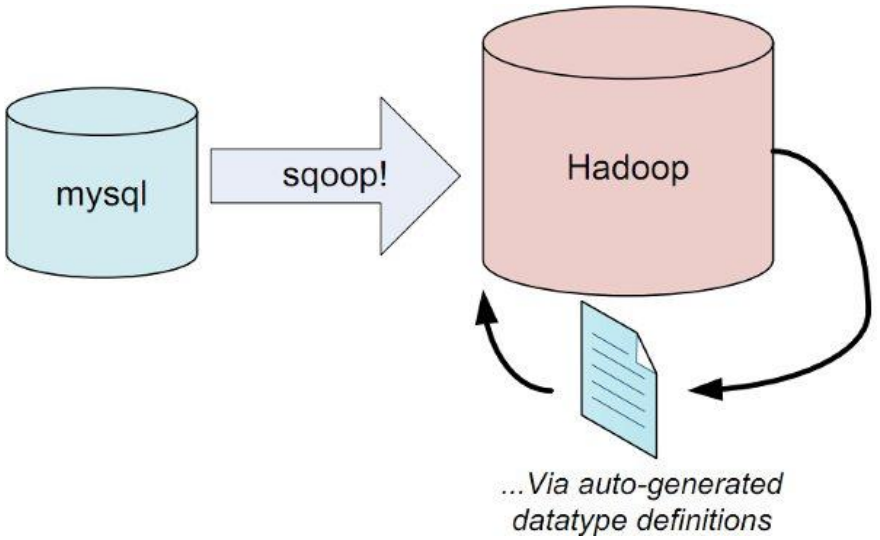
Zookeeper

- Google Chubby的开源实现
- 用于协调分布式系统上的各种服务。例如确认消息是否准确到达，防止单点失效，处理负载均衡等
- 应用场景：Hbase，实现Namenode自动切换
- 工作原理：领导者，跟随者以及选举过程



Sqoop

- 用于在Hadoop和关系型数据库之间交换数据
- 通过JDBC接口连入关系型数据库



Sqoop

Avro

- 数据序列化工具，由Hadoop的创始人Doug Cutting主持开发
- 用于支持大批量数据交换的应用。支持二进制序列化方式，可以便捷，快速地处理大量数据
- 动态语言友好，Avro提供的机制使动态语言可以方便地处理 Avro数据。
- Thrift接口

Chukwa

- 架构在Hadoop之上的数据采集与分析框架
- 主要进行日志采集和分析
- 通过安装在收集节点的“代理”采集最原始的日志数据
- 代理将数据发给收集器
- 收集器定时将数据写入Hadoop集群
- 指定定时启动的Map-Reduce作业队数据进行加工处理和分析
- Hadoop基础管理中心（HICC）最终展示数据

Cassandra

- NoSQL，分布式的Key-Value型数据库，由Facebook贡献
- 与Hbase类似，也是借鉴Google Bigtable的思想体系
- 只有顺序写，没有随机写的设计，满足高负荷情形的性能需求

Hbase简介

- HBase是一个分布式的、面向列的开源数据库，该技术来源于Chang et al所撰写的Google论文“Bigtable：一个结构化数据的分布式存储系统”。
- 就像Bigtable利用了Google文件系统（File System）所提供的分布式数据存储一样，HBase在Hadoop之上提供了类似于Bigtable的能力。
- HBase是Apache的Hadoop 项目的子项目。
- HBase不同于一般的关系数据库,它是一个适合于非结构化数据存储的数据库.另一个不同的是HBase基于列的而不是基于行的模式

Big Table的想法

- 学生表的例子S(s#,sn,sd,sa)
- 存放为关系的学生表
- 以bigtable方式存放学生表
- Bigtable: 无所不包的大表

Hbase逻辑模型

- 以表的形式存放数据
- 表由行与列组成，每个列属于某个列族，由行和列确定的存储单元称为元素
- 每个元素保存了同一份数据的多个版本，由时间戳来标识区分

表 4-1　数据存储逻辑视图

行　键	时间戳	列族 contents	列族 anchor	列族 mime
"com.cnn.www"	t9		anchor:cnnsi.com="CNN"	
	t8		anchor:my.look.ca= "CNN.com"	
	t6	contents:html="<html>..."		mime:type="text/html"
	t5	contents:html="<html>..."		
	t3	contents:html="<html>..."		

行键

- 行键是数据行在表里的唯一标识，并作为检索记录的主键
- 访问表里的行只有三种方式

1 通过单个行键访问

2 给定行键的范围访问

3 全表扫描

行键可以是最大长度不超过64KB的任意字符串，并按照字典序存储

对于经常要一起读取的行，要对行键值精心设计，以便它们能放在一起存储

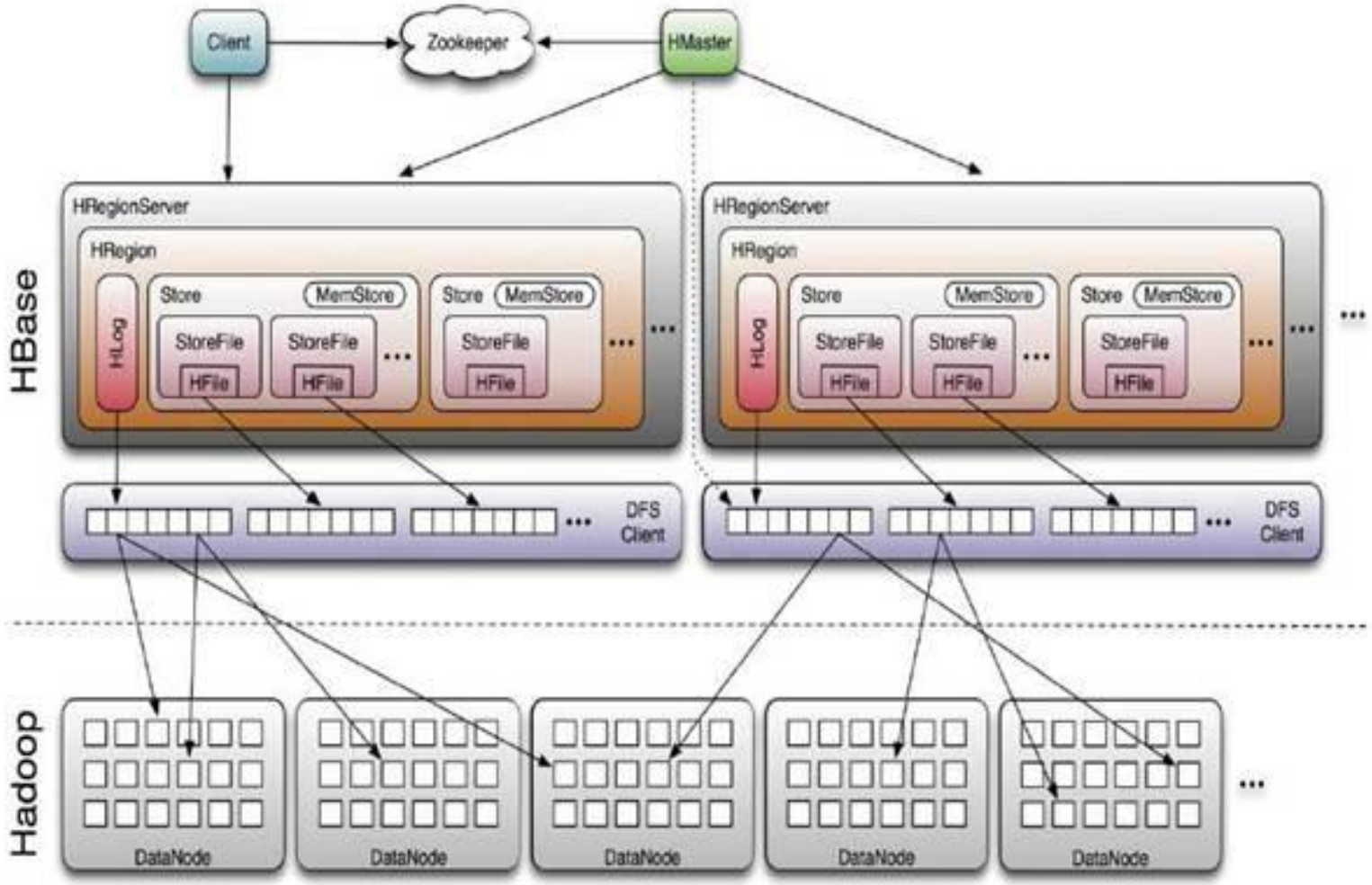
列族与列

- 列表示为<列族>:<限定符>
- Hbase在磁盘上按照列族存储数据，这种列式数据库的设计非常适合于数据分析的情形
- 列族里的元素最好具有相同的读写方式（例如等长的字符串），以提高性能

时间戳

- 对应每次数据操作的时间，可由系统自动生成，也可以由用户显式的赋值
- Hbase支持两种数据版本回收方式：1 每个数据单元，只存储指定个数的最新版本 2 保存指定时间长度的版本（例如7天）
- 常见的客户端时间查询：“某个时刻起的最新数据”或“给我全部版本的数据”
- 元素由 行键，列族:限定符，时间戳唯一决定
- 元素以字节码形式存放，没有类型之分

Hbase物理模型


Client
Zookeeper
HMaster
HBase
HRegionServer
HRegion
HLog
Store
MemStore
StoreFile
HFile
DFS Client
Hadoop
DataNode

Region和Region服务器

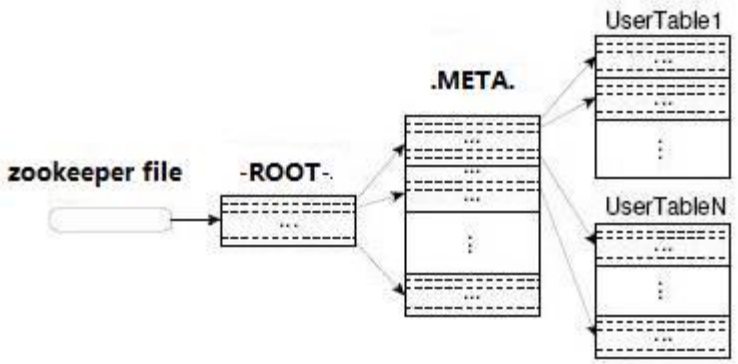
- 表在行方向上，按照行键范围划分成若干的Region
- 每个表最初只有一个region，当记录数增加到超过某个阈值时，开始分裂成两个region
- 物理上所有数据存放在HDFS，由Region服务器提供region的管理
- 一台物理节点只能跑一个HRegionServer
- 一个Hregionserver可以管理多个Region实例
- 一个Region实例包括Hlog日志和存放数据的Store
- Hmaster作为总控节点
- Zookeeper负责调度

HLog

- 用于灾难恢复
- 预写式日志，记录所有更新操作，操作先记录进日志，数据才会写入

-ROOT- 和 .META. 表

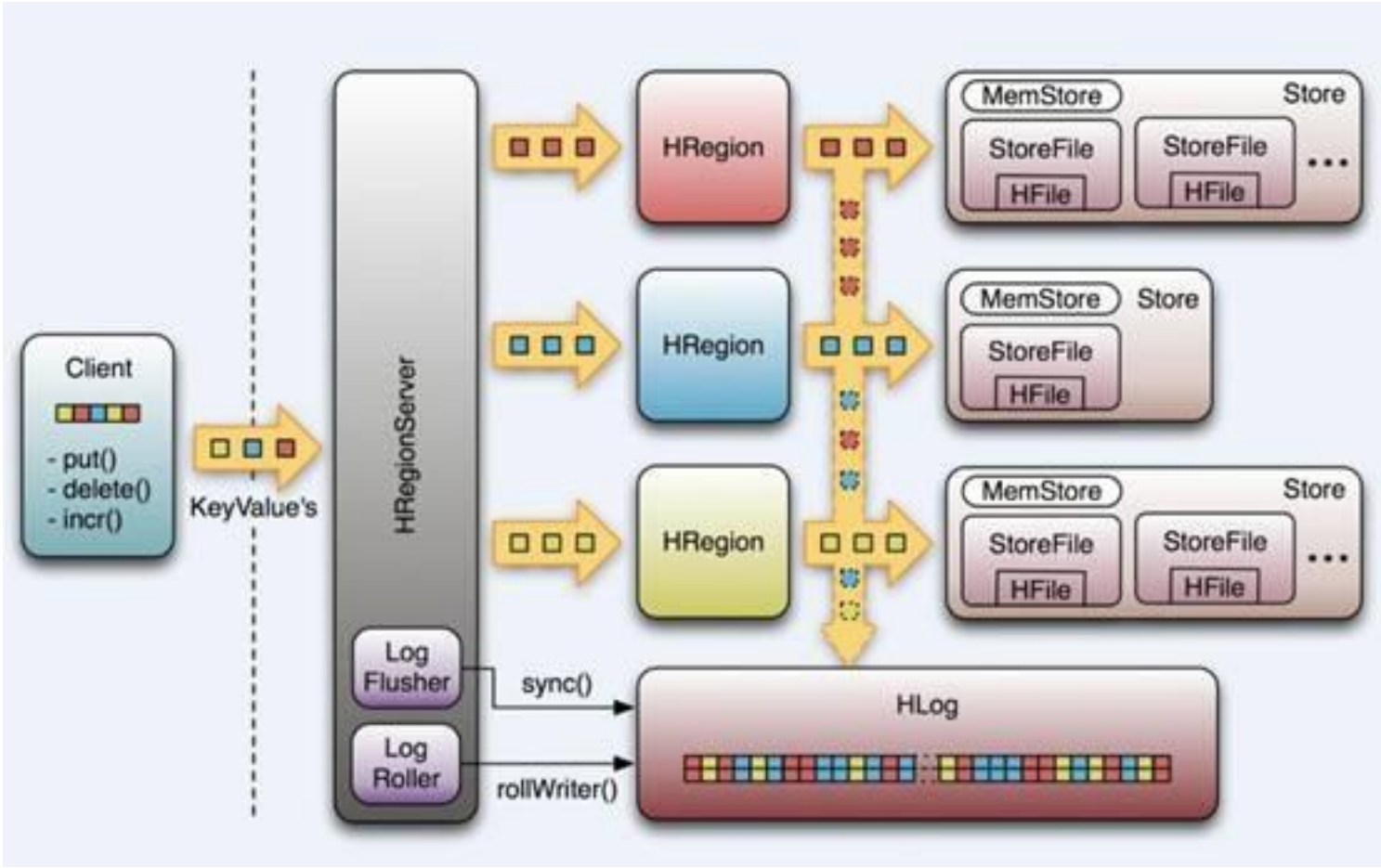
- HBase中有两张特殊的Table，-ROOT-和.META.
- Ø .META.：记录了用户表的Region信息，.META.可以有多个regoin
- Ø -ROOT-：记录了.META.表的Region信息，-ROOT-只有一个region
- Ø Zookeeper中记录了-ROOT-表的location



Memstore与storefile

- 一个region由多个store组成，每个store包含一个列族的所有数据
- Store包括位于把内存的memstore和位于硬盘的storefile
- 写操作先写入memstore，当memstore中的数据量达到某个阈值，Hregionserver会启动flashcache进程写入storefile，每次写入形成单独一个storefile
- 当storefile文件的数量增长到一定阈值后，系统会进行合并，在合并过程中会进行版本合并和删除工作，形成更大的storefile
- 当storefile大小超过一定阈值后，会把当前的region分割为两个，并由Hmaster分配到相应的region服务器，实现负载均衡
- 客户端检索数据时，先在memstore找，找不到再找storefile

图解

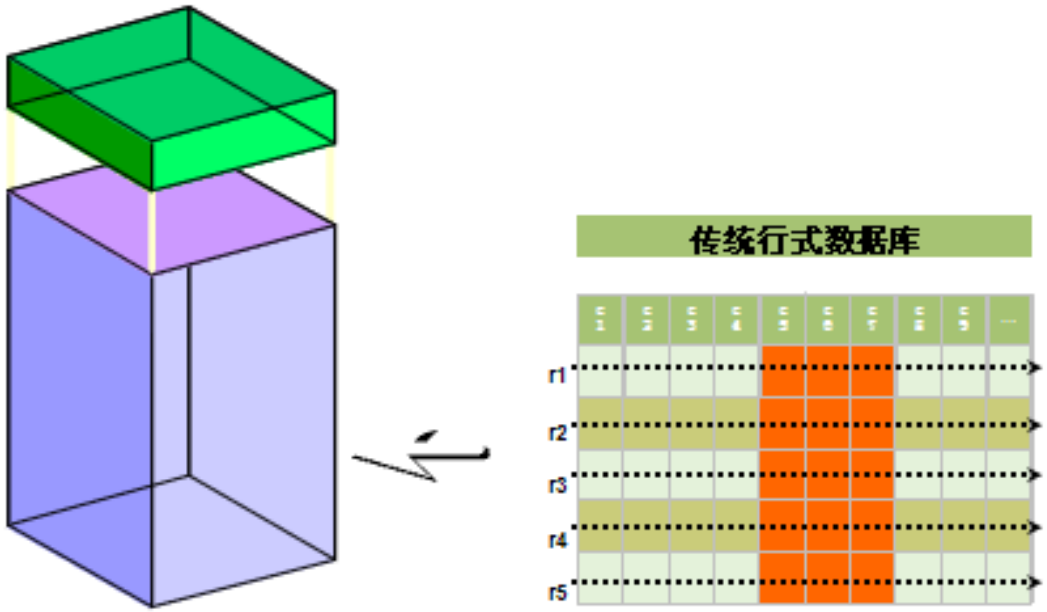

Client
- put()
- delete()
- incr()
KeyValue's
HRegionServer
Log Flusher
Log Roller
sync()
rollWriter()
HRegion
HRegion
HRegion
MemStore
Store
StoreFile
HFile
StoreFile
HFile
MemStore
Store
StoreFile
HFile
MemStore
Store
StoreFile
HFile
StoreFile
HFile
HLog

Hbase vs Oracle

- 索引不同造成行为的差异
- Hbase适合大量插入同时又有读的情况
- Hbase的瓶颈是硬盘传输速度，Oracle的瓶颈是硬盘寻道时间
- Hbase很适合寻找按照时间排序top n的场景

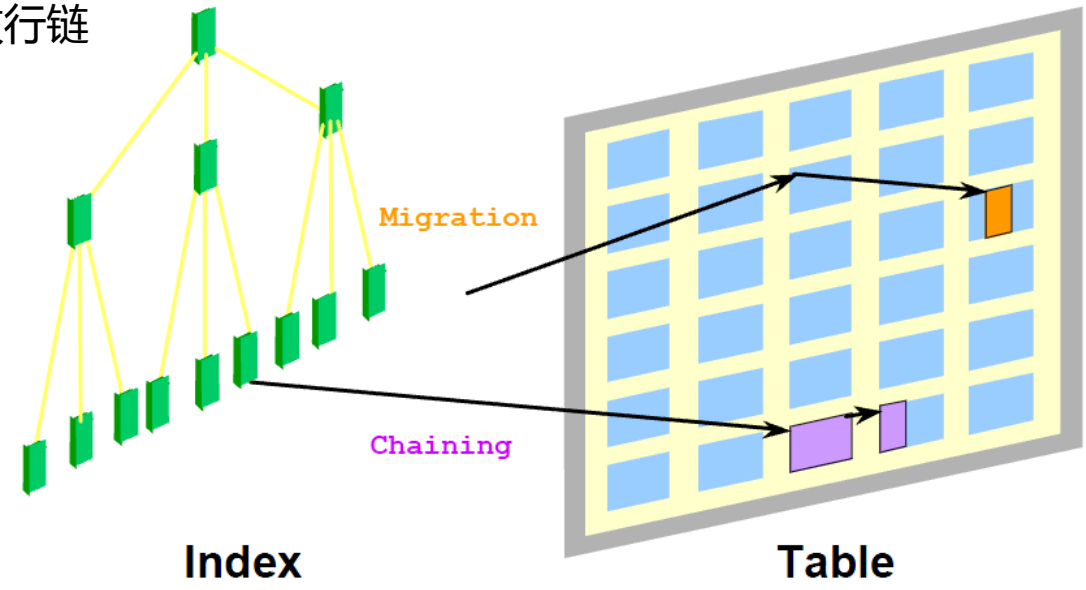
传统数据库的行式存储

- 数据存放在数据文件内
- 数据文件的基本组成单位：块/页
- 块内结构：块头、数据区



行式存储的问题

- 读某个列必须读入整行
- 行不等长，修改数据可能导致行迁移
- 行数据较多时可能导致行链



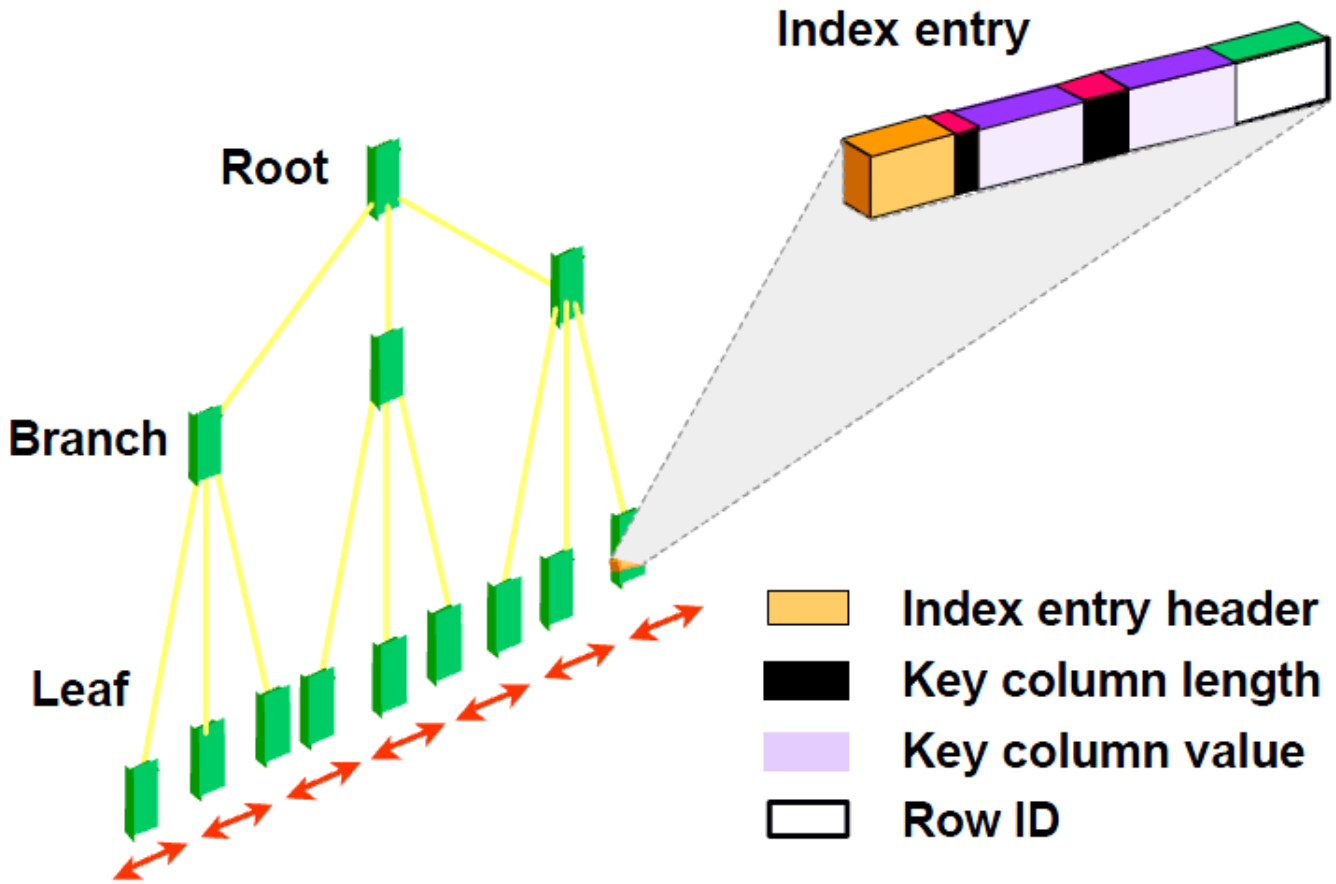
Oracle行式存储的访问路径

- 全表扫描
- 行标识访问

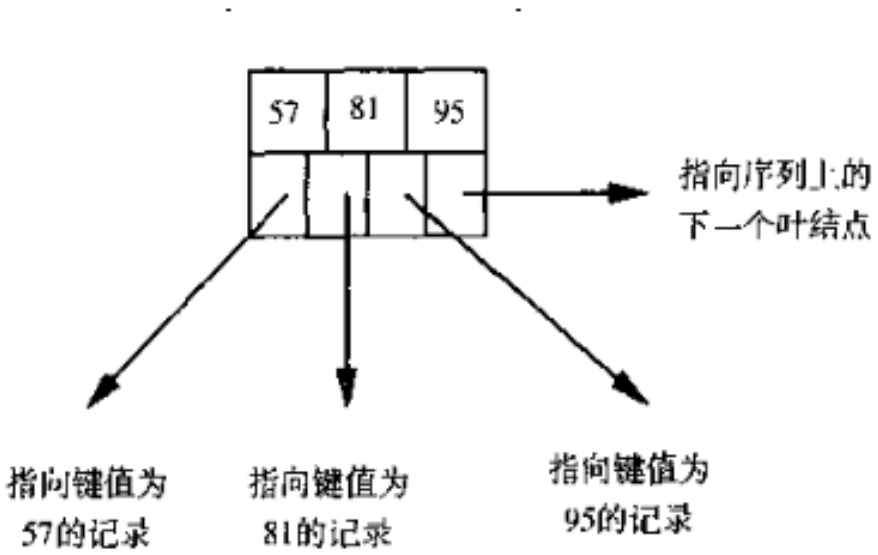
```
SQL> select rowid,empno,ename,job,sal,deptno from emp;

ROWID                  EMPNO ENAME      JOB              SAL     DEPTNO
------------------ ---------- ---------- --------- ---------- ----------
AAAM9KAAEAAAAJdAAA       7369 SMITH      CLERK            800         20
AAAM9KAAEAAAAJdAAB       7499 ALLEN      SALESMAN        1600         30
AAAM9KAAEAAAAJdAAC       7521 WARD       SALESMAN        1250         30
AAAM9KAAEAAAAJdAAD       7566 JONES      MANAGER         2975         20
AAAM9KAAEAAAAJdAAE       7654 MARTIN     SALESMAN        1250         30
AAAM9KAAEAAAAJdAAF       7698 BLAKE      MANAGER         2850         30
AAAM9KAAEAAAAJdAAG       7782 CLARK      MANAGER         2550         10
AAAM9KAAEAAAAJdAAH       7839 KING       PRESIDENT       8000         10
AAAM9KAAEAAAAJdAAI       7844 TURNER     SALESMAN        1500         30
AAAM9KAAEAAAAJdAAJ       7900 JAMES      CLERK            950         30
AAAM9KAAEAAAAJdAAK       7902 FORD       ANALYST         3000         20
AAAM9KAAEAAAAJdAAL       7934 MILLER     CLERK           1400         10
```

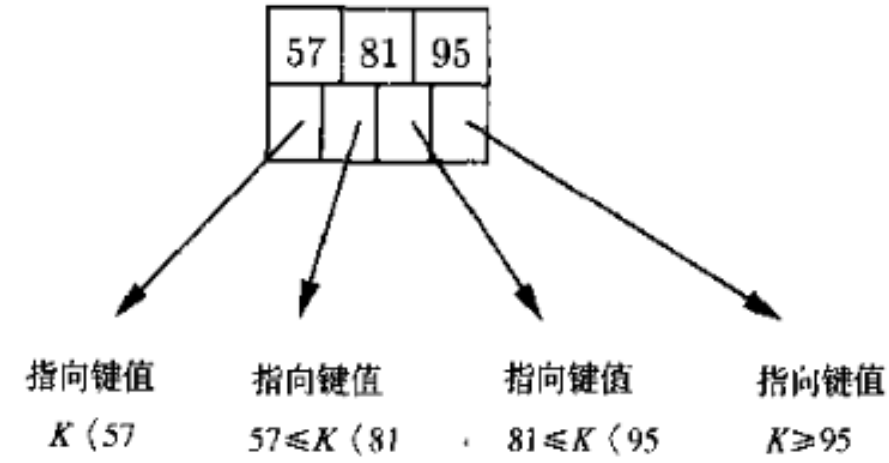
行标识访问：B树索引


Index entry
Root
Branch
Leaf
Index entry header
Key column length
Key column value
Row ID

B树索引原理：结点



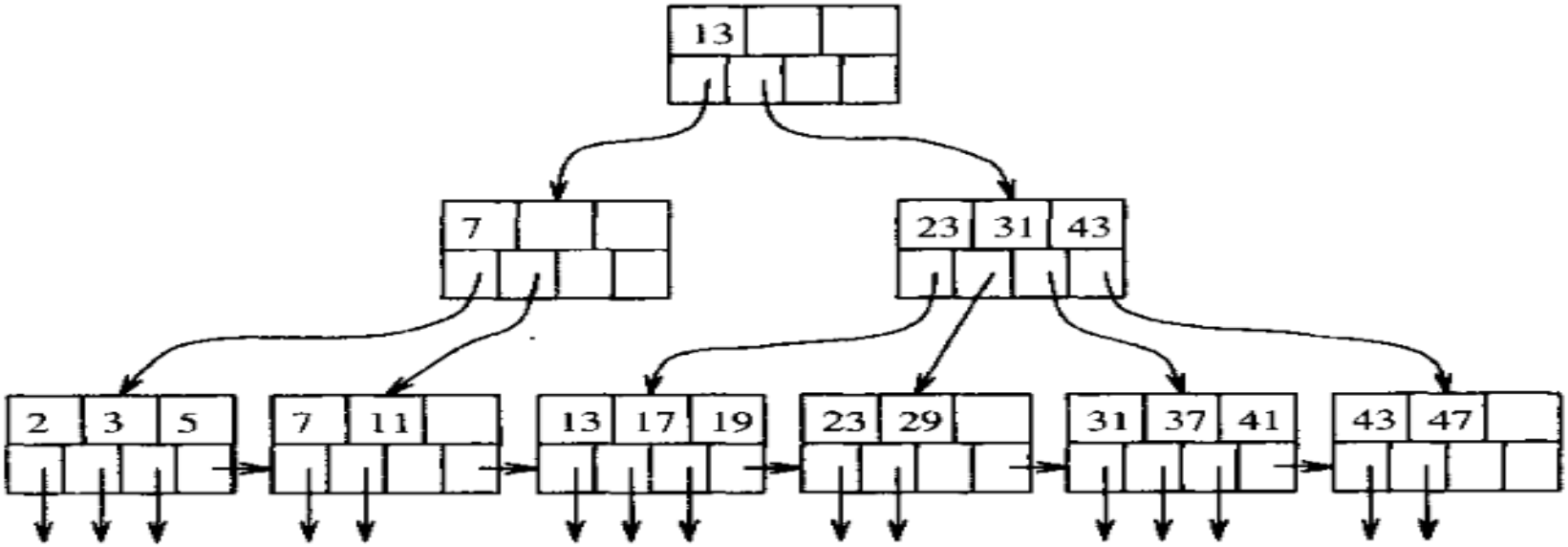
典型的B树叶结点



典型的B树内部结点

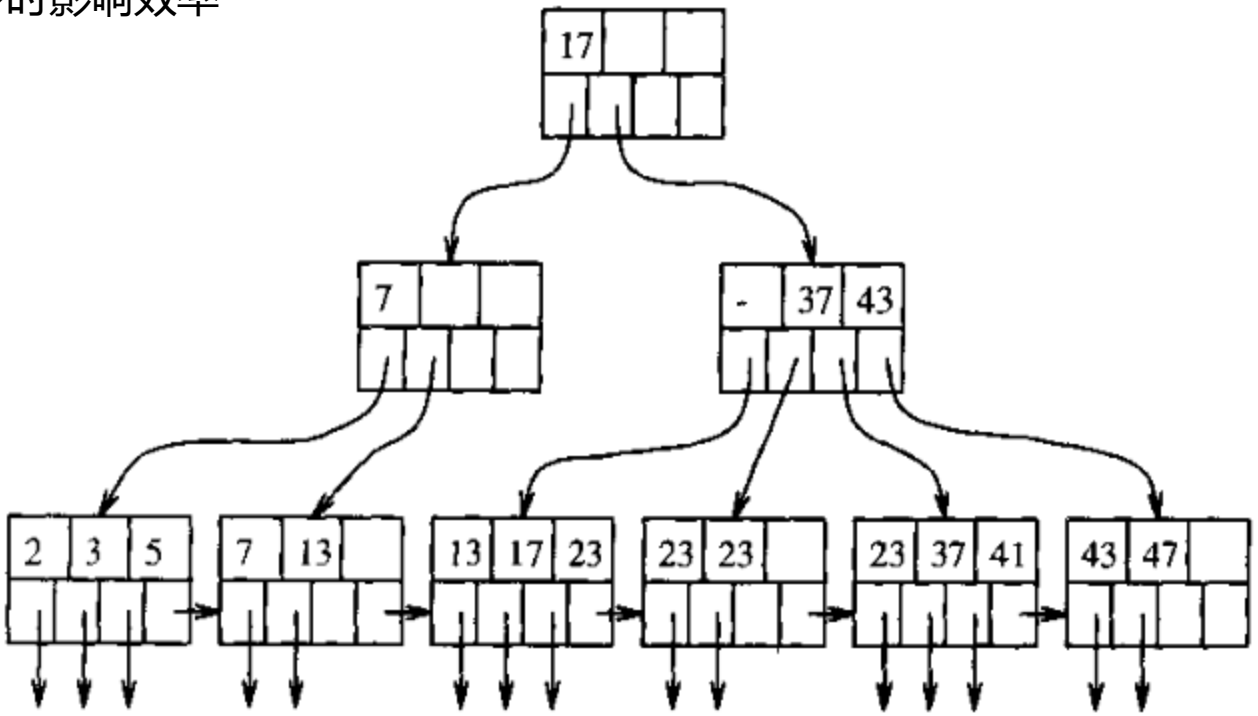
B树索引原理：树形

- 利用B树进行查询——access path
- B树插入——分裂结点
- B树删除——合并结点



B树索引的弱点

- 空间代价，创建时间代价，维护代价
- 重复值多时影响效率



一棵带重复键的B树

BigTable的LSM索引

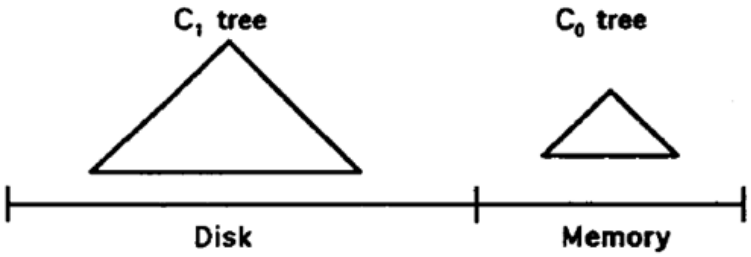


Fig. 1. Schematic picture of an LSM-tree of two components

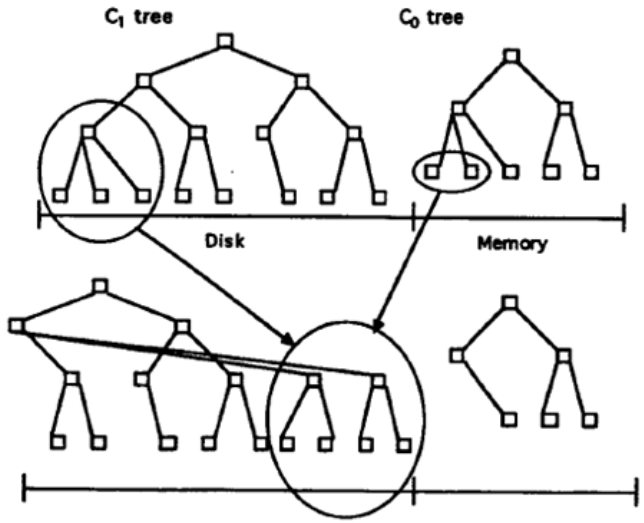


Fig. 2. Conceptual picture of rolling merge steps, with result written back to disk

Thanks

FAQ时间