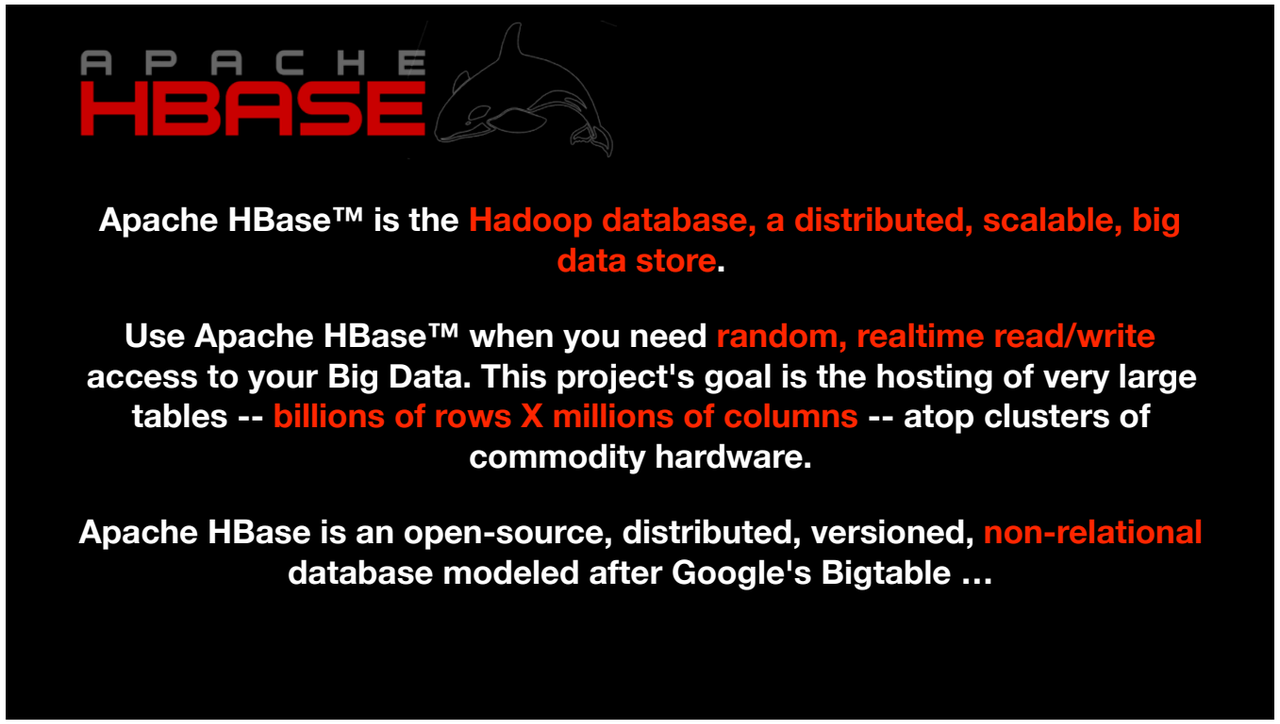
HBase基本介绍

田志鹏 20190714

上次分位点估算当时没解决的两个问题已更新ppt.

今天讲的内容比较基础, 而且偏理论, 因为我个人也没有太多实际使用经验, 纸上谈兵.



先来一段HBase官网的自我介绍. blabla翻译一下

重点看其中的红字, 什么hadoop数据库, 分布式的, 可伸缩的, 随机实时读写 十亿级行, 百万级列

每次看一个项目介绍完自己, 还是不知道他是干嘛的, 希望今天我介绍完, 大家能知道他是干嘛的

回顾:

稀疏的, 行和列比较随意, 不需要固定的schema, 没有值的位置不占空间

分布式的, 本身hdfs的是分布式的容错的, 在借助region和cf的水平垂直分表, 整个数据可以很好的分散

持久化的, 大部分数据都是基于hdfs的持久化,(btw 顺序写磁盘, 速度不慢)

Sorted map. 整个数据模型就是一个按key排序的大Map,

Agenda

- ⋆ Data Model
- ⋆ Architecture & Component
- ⋆ Schema design

1. Data Model

存什么样的数据

Row Key	Time Stamp	ColumnFamily contents	ColumnFamily anchor	ColumnFamily people
"com.cnn.www"	t9		anchor:cnnsi.com = "CNN"	
"com.cnn.www"	t8		anchor:my.look.ca = "CNN.com"	
"com.cnn.www"	t6	contents:html = "<html>… "		
"com.cnn.www"	t5	contents:html = "<html>… "		
"com.cnn.www"	t3	contents:html = "<html>… "		
"com.example.www"	t5	contents:html = "<html>… "		people:author = "John Doe"

既然是一个数据库, 一个非关系型的数据库, 我们首先要了解他的数据模型, 就是说他能存什么样的数据.

像redis是存kv结构的数据, MongoDB是存储文档型数据, 那HBase存什么样的数据?

数据模型 逻辑视图

- ’表/行/列’
- Row Key
- ColumnFamily列族 ：ColumnQualifier列限定名
  分数:语文
- Version/Timestamp

Row Key	ColumnFamily 分数			ColumFamily 图片
name	语文	数学	英语	头像
Baron	38.02669	46.88364	49.23896	01001010111011011010101010101101
Bernardo	6.662968	82.81097	1.353296	01001010111011011010101010101101
Bernete	51.80541	57.27945	48.85917	01001010111011011010101010101101
Brent	59.31392	11.02464	32.99529	01001010111011011010101010101101
Callida	60.1045	69.73448	56.50936	01001010111011011010101010101101
Carmelia	87.41105	19.16582	74.00486	01001010111011011010101010101101
Cyrus	18.97029	29.61708	39.21837	01001010111011011010101010101101
Darelle	8.404259	38.36477	37.49422	01001010111011011010101010101101
Geralda	30.15843	18.99907	66.69951	01001010111011011010101010101101
Hilarius	81.8834	13.86951	40.84456	01001010111011011010101010101101
Hirsch	9.053185	52.44312	26.59205	01001010111011011010101010101101
Ilyse	10.56388	29.18008	27.29948	01001010111011011010101010101101
Jakob	79.34119	43.90385	33.48238	01001010111011011010101010101101
Jenilee	69.06755	71.35634	89.43621	01001010111011011010101010101101
Jonathan	62.80531	22.72737	62.44845	01001010111011011010101010101101
Kate	51.7571	0.480606	33.72056	01001010111011011010101010101101
Kath	28.42197	37.67409	60.66452	01001010111011011010101010101101

整个HBase和关系数据库很像, 但又要时时注意两者的区别. 右面我继续以一次考试学生分数距离.

首先也有表/行/列这三个概念. hbase最小就是一个table.

然后要为每行数据选择row key, row key和关系数据库的主键类似, 但不是自增的, 要我们自定义, 这里我用人名当row key

然后是列, 这里hbase要求我们把列做个分类, 叫列族. 这里我分了两个ColumnFamily, 分数和图片

每个列族名+列限定名用冒号连起来组成列名, 比如 分数:语文. 然后我又有图片列族, 只有一列头像, 存的是二进制的图片内容

然后hbase又引入了版本概念, 就是图上这个 行 交 列 的每一个格子 都可以有多个版本的值, 我没有画出来.

版本用时间戳表示.

数据模型 RowKey

- 整个table都是按rowkey排序存储的
- 按rowkey用LSM树做了索引, 没有其他二级索引
- 所以只有按rowkey查询比较快, 其他全表扫
- 除了列族要求是可打印字符, 别的都可以是任意二进制数据

Row Key	ColumnFamily 分数			ColumFamily 图片
name	语文	数学	英语	头像
Baron	38.02669	46.88364	49.23896	01001010111011011010101010101101
Bernardo	6.662968	82.81097	1.353296	01001010111011011010101010101101
Bernete	51.80541	57.27945	48.85917	01001010111011011010101010101101
Brent	59.31392	11.02464	32.99529	01001010111011011010101010101101
Callida	60.1045	69.73448	56.50936	01001010111011011010101010101101
Carmelia	87.41105	19.16582	74.00486	01001010111011011010101010101101
Cyrus	18.97029	29.61708	39.21837	01001010111011011010101010101101
Darelle	8.404259	38.36477	37.49422	01001010111011011010101010101101
Geralda	30.15843	18.99907	66.69951	01001010111011011010101010101101
Hilarius	81.8834	13.86951	40.84456	01001010111011011010101010101101
Hirsch	9.053185	52.44312	26.59205	01001010111011011010101010101101
Ilyse	10.56388	29.18008	27.29948	01001010111011011010101010101101
Jakob	79.34119	43.90385	33.48238	01001010111011011010101010101101
Jenilee	69.06755	71.35634	89.43621	01001010111011011010101010101101
Jonathan	62.80531	22.72737	62.44845	01001010111011011010101010101101
Kate	51.7571	0.480606	33.72056	01001010111011011010101010101101
Kath	28.42197	37.67409	60.66452	01001010111011011010101010101101

数据模型 Column

- ColumnFamily的设计是要求把相关的列放在同族
- 所有的各项配置, 都是指定到列族上, 不是列, 同列族数据, 物理上存在同文件
- ColumnFamily在建表时确定, 具体有哪些列是数据随意添加的

Row Key	ColumnFamily 分数			ColumFamily 图片
name	语文	数学	英语	头像
Baron	38.02669	46.88364	49.23896	01001010111011011010101010101101
Bernardo	6.662968	82.81097	1.353296	01001010111011011010101010101101
Bernete	51.80541	57.27945	48.85917	01001010111011011010101010101101
Brent	59.31392	11.02464	32.99529	01001010111011011010101010101101
Callida	60.1045	69.73448	56.50936	01001010111011011010101010101101
Carmelia	87.41105	19.16582	74.00486	01001010111011011010101010101101
Cyrus	18.97029	29.61708	39.21837	01001010111011011010101010101101
Darelle	8.404259	38.36477	37.49422	01001010111011011010101010101101
Geralda	30.15843	18.99907	66.69951	01001010111011011010101010101101
Hilarius	81.8834	13.86951	40.84456	01001010111011011010101010101101
Hirsch	9.053185	52.44312	26.59205	01001010111011011010101010101101
Ilyse	10.56388	29.18008	27.29948	01001010111011011010101010101101
Jakob	79.34119	43.90385	33.48238	01001010111011011010101010101101
Jenilee	69.06755	71.35634	89.43621	01001010111011011010101010101101
Jonathan	62.80531	22.72737	62.44845	01001010111011011010101010101101
Kate	51.7571	0.480606	33.72056	01001010111011011010101010101101
Kath	28.42197	37.67409	60.66452	01001010111011011010101010101101

数据模型 物理视图

- 逻辑视图 不等于 物理视图
- 伪装起来的KV存储

接下来我们说一下数据模型的物理视图, 什么叫物理视图, 就是逻辑上HBase是刚才的表行列的结构, 使用的时候也可以这么想, 但是实际HBase存储数据的时候, 不是这么存的! 用一句话总结物理视图, 就是伪装起来的KV存储

这里我想了好久如何给大家讲HBase数据模型的物理视图

数据模型 物理视图

Row Key	ColumnFamily 分数			ColumnFamily 图片
name	语文	数学	英语	头像
Baron	38.02669	46.88364	49.23896	01001010111011010101010101101
Bernardo	6.662968	82.81097		01001010111011010101010101101
Bernete	51.80541		48.85917	01001010111011010101010101101
Brent	59.31392	11.02464	32.99529	01001010111011010101010101101
Callida	60.1045	69.73448	56.50936	01001010111011010101010101101
Carmelia	87.41105	19.16582	74.00486	01001010111011010101010101101
Cyrus	18.97029	29.61708	39.21837	01001010111011010101010101101
Darelle	8.404259	38.36477	37.49422	01001010111011010101010101101
Geralda	30.15843	18.99907	66.69951	01001010111011010101010101101
Hilarius	81.8834	13.86951	40.84456	01001010111011010101010101101
Hirsch	9.053185	52.44312	26.59205	01001010111011010101010101101
Ilyse	10.56388	29.18008	27.29948	01001010111011010101010101101
Jakob	79.34119	43.90385	33.48238	01001010111011010101010101101
Jenilee	69.06755	71.35634	89.43621	01001010111011010101010101101
Jonathan	62.80531	22.72737	62.44845	01001010111011010101010101101
Kate	51.7571	0.480606	33.72056	01001010111011010101010101101

key		value
Baron	分数: 语文 : 20190607	31.28453536
Baron	分数: 数学 : 20190608	93.77157638
Baron	分数: 英语 : 20190609	87.79503988
Baron	分数: 英语 : 20190608	60.54470649
Bernardo	分数: 语文 : 20190607	30.06661704
Bernardo	分数: 数学 : 20190608	93.76191086
Bernardo	分数: 英语 : 20190607	57.783734
Bernete	分数: 语文 : 20190607	52.62751332
Bernete	分数: 数学 : 20190607	11.81483627
Bernete	分数: 英语 : 20190607	19.98519842
...	...	...

key		value
Baron	图片:头像:20190607	01001010111011010101010101101

数据模型 物理视图

```
{
    "Baron": {
        "分数": {
            "语文": {
                "20190607": 31.28
            },
            "数学": {
                "20190608": 93.77
            },
            "英语": {
                "20190609": 87.79,
                "20190608": 60.54
            }
        },
        "图片": {
            "头像": "0100010101010101"
        }
    }
}
//多级map转一级map
{
    "Baron 分数:语文 20190607": 31.28,
    "Baron 分数:数学 20190607": 93.77,
    "Baron 分数:英语 20190607": 87.79,
    "Baron 分数:英语 20190607": 60.54,
    "Baron 图片:头像 20190607": "0100010101010101",
}
```

或者换一个方式来看

寻迹数据 场景

- 设备id作为rowkey
- 日志时间作为version时间
- 其余作为10列

name	类型	描述
id	String	具体的id
id_type	String	取值：raw_idfa, md5_imei, md5_mac
device_model	String	设备机型
ip	String	原始的ip地址
gcj_lon	Double	高德地图的经度信息
gcj_lat	Double	高德地图的纬度信息
radius	Double	覆盖半径
scene	String	场景
media_name	String	媒体名称
location	String	中文位置
timestamp	Long	时间戳

1个ID, 100条行为(100个版本), 10列 = 1000个KV对:

id存了1000次, 时间存了1000次, 所有列名(location这个字符串)存了100次

1个ID, 100条行为(100个版本), 1列 = 100个KV对

千亿级PV, 万亿个KV对

寻迹数据 对比

	格式	表名	20190701	说明
试验1	10列, 什么也不开	daas:test_tzp1	2.2T	/user/mz_supertool/test/tzp/hbase_test3/2
试验5	1列, 什么都不开	daas:test_tzp1	608.9G	256个初始分区
试验4	1列, 开encoding	daas:test_tzp22	237.0G	COMPRESSION => 'SNAPPY'
试验2	10列, 开encoding和压缩	daas:test_tzp11	227.3G	split有点问题
试验3	1列, 开encoding和压缩	daas:test_tzp2	124.3G	DATA_BLOCK_ENCODING => 'FAST_DIFF'
原始	orc, 12列	-	42.1G	/user/mz_supertool/test/panfeng/id-geolc

这里先对比 前两行. 一个10列一个1列.

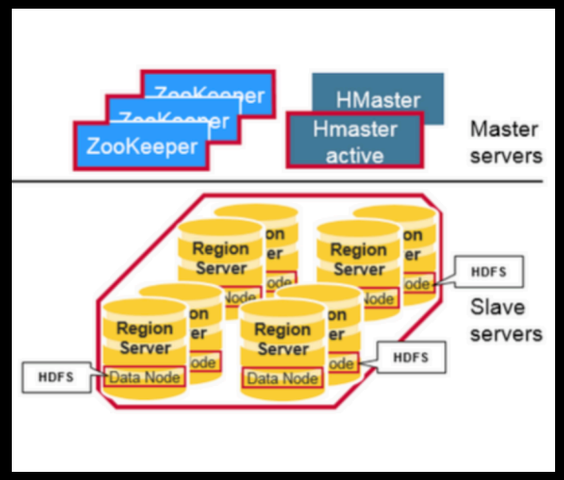
数据模型 why

- 一开始想吐槽这种模型设计, 后来看google论文说人家参考了很多模型最终才决定这样设计的....
- 简单高效, 可以作为其他应用的基础

- OpenTSDB: 时序数据库, 主要是监控数据这类的
- JanusGraph: 图数据库, 知识图谱
- GeoMesa: 时空位置数据库
- Kylin: OLAP, 用HBase存cube
- Phoenix: Sql on HBase

2. Architecture

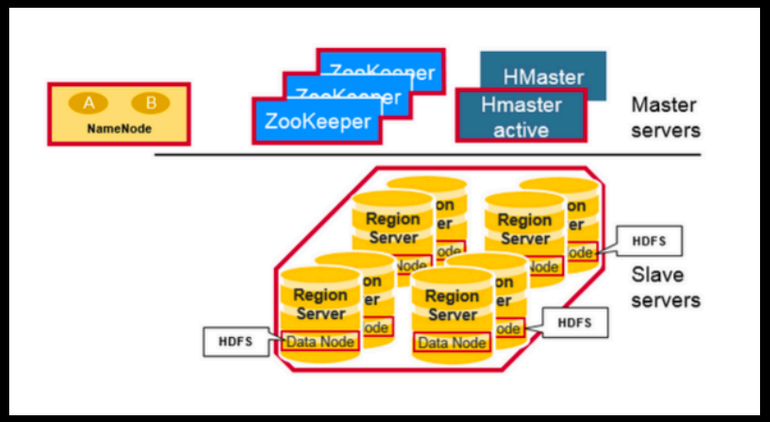
系统模块架构啥样, 如何存取数据



进入第二部分, 架构组成. 知道了存的数据样子, 我们接下来看点实际的

系统组成

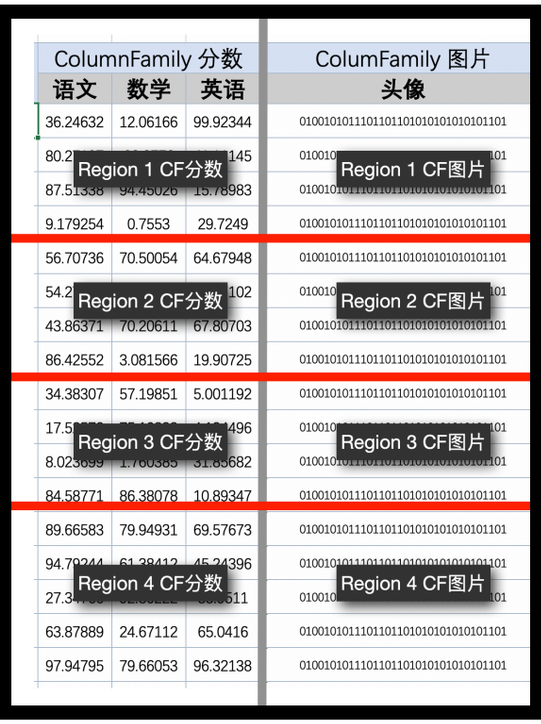
- HMaster
- RegionServer
- Zookeeper
- NameNode/DataNode



如图有这么几个组成部分, 前两个是HBase的
Master是负责管理的, RegionServer是实际干活的
zookeeper作为协调信息存储的地方, 比如节点健康状态
然后HBase的数据都要存放在hdfs上, 就要有node. 如图可以看出RegionServer和Datanode尽量在同一台机器上.

系统组成 Region

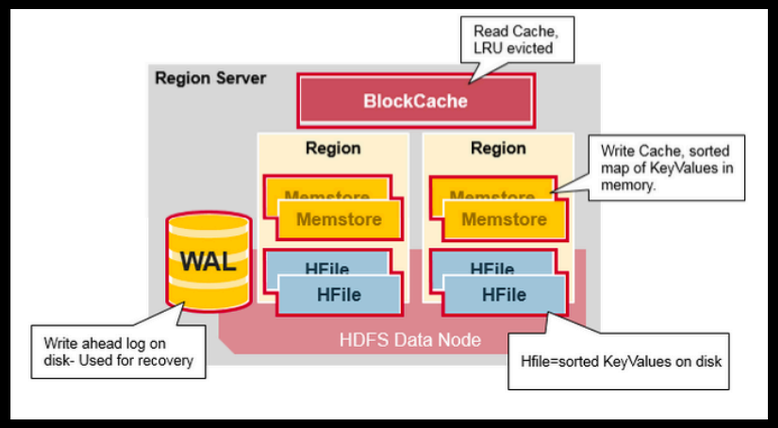
- 水平 按rowkey分开 region
  - Pre-split: 0-5 6-10
  - Auto-split: size
- 垂直 按CF分开



水平按rowkey分. 这个分两步, 第一个是在建表的时候指定分的方式. 比如两个split, 0-5 6-10
自动分区是指一个region大小超了
region的概念. 这个很类似关系数据库里我们说水平/垂直分表的意思.

系统组成 RegionServer

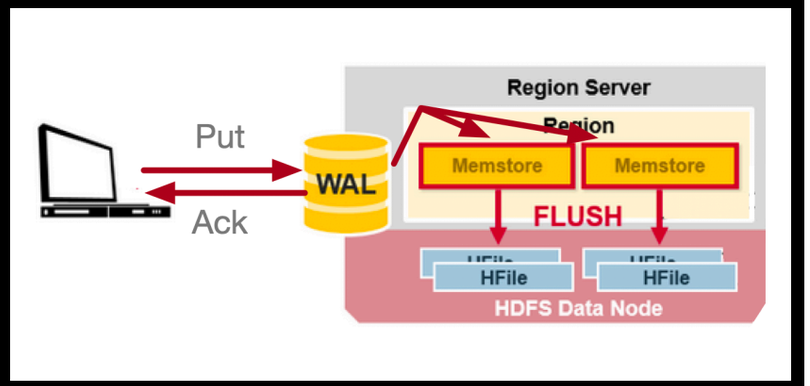
- 读缓存: BlockCache
- 写缓存: Memstore
- 写操作日志: WAL
- 数据文件: HFile



深入RegionServer内部. 有两个Cache和两种文件

系统组成 RegionServer写操作

- 先写WAL做故障恢复用
- 写到Memstore中
- 足够大时flush到HFile
- Compaction (Minor/Major)

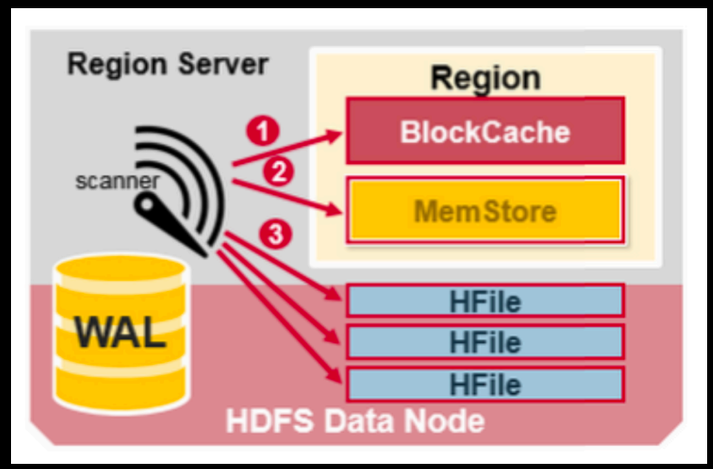


一个Region多个CF就多个Memstore
在Memstore里已经处理好格式, 排序
有一个Memstore满了, 就整个Region flush

这里每隔一会就flush, 会生成很多小的HFile, HBase会执行两种compaction,
minor com只是将小的何为一些大的. major更狠一些, 合成一个文件 (all the HFiles in a region to one HFile per column family).
归并排序.

系统组成 RegionServer读操作

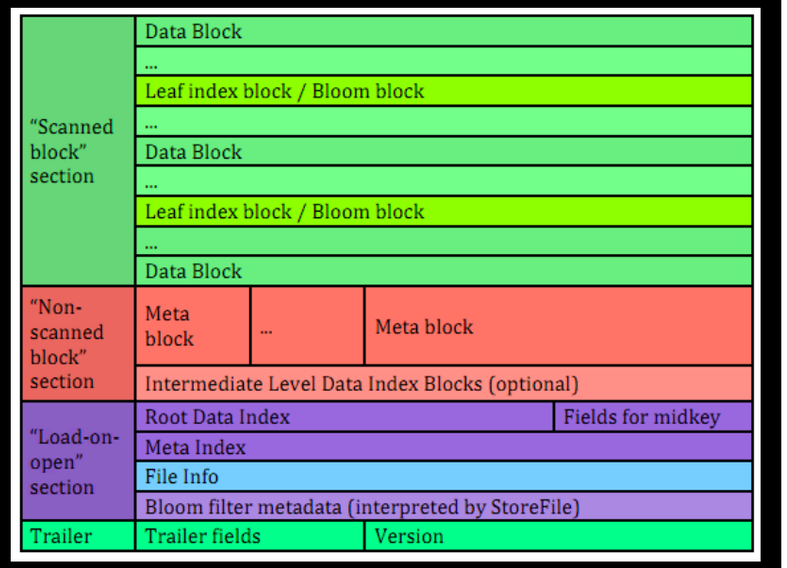
- 先找BlockCache
- 再找Memstore, 写入还没flush
- 根据index去HFile里找
- HBase如何在hdfs这种append-only文件系统上实现, 修改/删除操作的



这里需要提的一点是, BlockCache里不光对数据做了缓存, 其实在RegionServer启动的时候, 会把所有region的索引信息加载进去.

系统组成 HFile

- 整个HFile分为4大部分
- 所有东西都被组织为大小相同的block
- Index block / bloom block



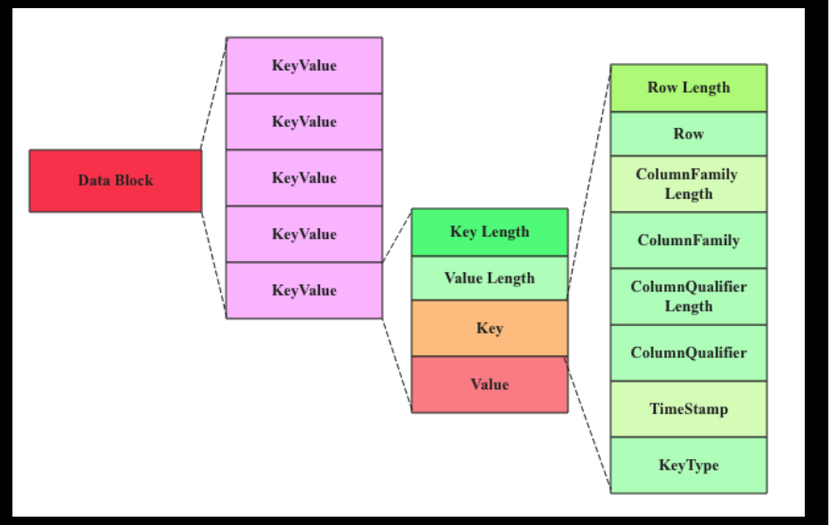
这个图是整个HFile文件的结构.

最下是文件尾, 根据文件尾上的信息, 就能读取到 load-on-open部分的各种索引信息

最上是数据块所有实际的数据, 数据块的叶节点索引, 布隆索引都在这

系统组成 HFile DataBlock

- 排序好的KeyValue



这里重点看一下Datablock内部结构

系统组成 DataBlockEncoding

Key Len	Val Len	Key	Value
24	…	RowKey:Family:Qualifier0	…
24	…	RowKey:Family:Qualifier1	…
25	…	RowKey:Family:QualifierN	…
25	…	RowKey2:Family:Qualifier1	…
25	…	RowKey2:Family:Qualifier2	…
…	…	…	…

系统组成 Prefix Encoding

Key Len	Val Len	Prefix Len	Key	Value
24	…	0	RowKey:Family:Qualifier0	…
1	…	23	1	…
1	…	23	N	…
19	…	6	2:Family:Qualifier1	…
1	…	24	2	…
…	…	…	…	…

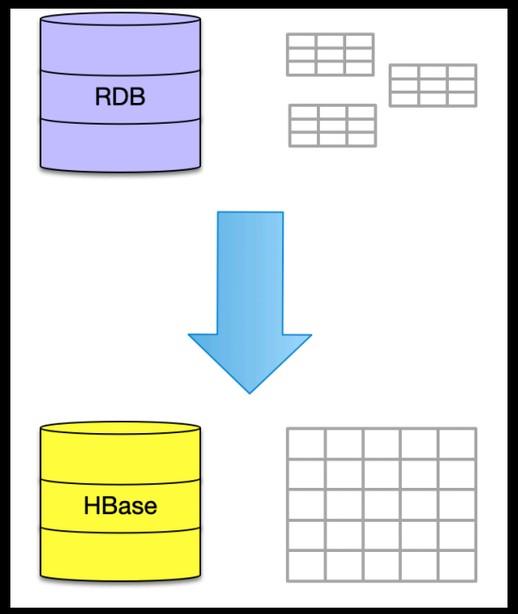
系统组成 Diff Encoding

Flags	Key Len	Val Len	Prefix Len	Key	Timestamp	Type	Value
0	24	512	0	RowKey:Family:Qualifier0	1340466835163	4	…
5		320	23	1	0		…
3			23	N	120	8	…
0	25	576	6	2:Family:Qualifier1	25	4	…
5		384	24	2	1124		…
…	…	…	…	…	…	…	…

借助全局排序的优势, 相近的KeyValue在一起,
用encoding能压缩存储key用的空间

3. Schema Design

如何设计表结构



Schema设计

- 合理设计RowKey 和 Column
- 根据查询需求决定schema
- 单一大宽表, 避免跨表查询
- 必要的时候进行冗余, 反范式
- 相关的数据要存放在一起
- 不要有热点
- 将列限定符视为数据通常是一个合理选择

在关于数据库里我们有各种范式来帮助设计, 在hbase里也要合理设计一下

https://link.springer.com/article/10.1186/s40537-017-0071-x
https://ieeexplore.ieee.org/abstract/document/7216979
https://ieeexplore.ieee.org/abstract/document/7214031

Schema设计

- 关于各种size的合理设计:
  - region数 50-100
  - 列族数量 1-3
  - 每个region大小 10G-50G
  - 每个cell大小 10M

Schema设计 OpenTSDB

存监控指标的时序数据库

RowKey:

[salt]<metric_uid><time bucket><tagk1><tagv1>[...

内存使用率　　20190725 16　　机器号　　2294

Column Qualifier:

<time offset><format flag>

指定指标的所有数据
指定指标指定时间的所有数据
指定一个服务器所有指标?

类似关系数据中建索引的时候, 最左前缀的原则

Schema设计 OpenTSDB

另一种存指标数据的设计

RowKey:

<host name><timestamp>

Column Qualifier:

<metric>

鱼和熊掌不可兼得: 冗余

Schema设计 有趣的操作

- 倒序域名: com.google.cloud 利于压缩空间
- 加盐salt: 防止出现热点
- 倒序时间戳 Long.Max - timestamp

参考

- 官方文档
- MapR HBase architecture
- HFile结构解析
- Google BigTable/SSTable

Q & A

Thanks