Day17_Hbase简介 & 架构 & 存储格式

大数据-张军锋 Day17 HBase 简介 架构 存储格式 Hbase

安装配置

- 解压配置环境变量
- 进入hbse的conf的目录配置hbase.env(环境,不启用自带zookeeper) ,hbase-

site.xml(节点,HMaster,data目录,tmp目录等),regionservers(配置节点名称)

- scp配置信息启动服务,必须先启动zookeeper

> 在某一台上解压hbase的压缩文件，如在192.168.15.5
> tar –zxvf hbase-1.2.0-bin.tar.gz
> 配置添加环境变量：

hbase

```
export HBASE_HOME=/usr/tools/hbase-1.2.0
export PATH=$PATH:$HBASE_HOME/bin
```

使环境变量生效
source /etc/profile

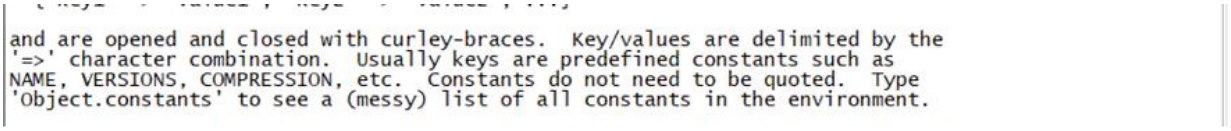
> 进入hbase的conf目录，需要修改三个文件：hbase-env.sh、hbase-site.xml和regionservers
> 其中hbase-env.sh中，在文档的十多行位置处添加：
> export JAVA_HOME=/usr/tools/jdk1.8.0_73
> 然后在后面添加：
> export HBASE_MANAGES_ZK=false
> regionservers文件中添加各个从属服务器的ip或者hostname：
> jokeros1
> jokeros2
> jokeros3
> hbase-site.xml中

```
<configuration>
        <property>
                <name>hbase.zookeeper.quorum</name>
                <value>jokeros1,jokeros2,jokeros3</value>
                <description>The directory shared by RegionServers.
</description>
        </property>
        <property>
                <name>hbase.zookeeper.property.dataDir</name>
                <value>/usr/tools/hbase-1.2.0/zookeeperdata</value>
                <description>Property from ZooKeeper config zoo.cf
g.
                The directory where the snapshot is stored.
                </description>
        </property>
        <property>
                <name>hbase.tmp.dir</name>
                <value>/usr/tools/hbase-1.2.0/tmpdata</value>
        </property>
        <property>
                <name>hbase.rootdir</name>
                <value>hdfs://jokeros1:9000/hbase</value>
                <description>The directory shared by RegionServers.
</description>
        </property>
        <property>
                <name>hbase.cluster.distributed</name>
                <value>true</value>
                <description>The mode the cluster will be in. Possi
ble values are
                false: standalone and pseudo-distributed setups wit
h managed Zookeeper
                true: fully-distributed with unmanaged Zookeeper Qu
orum (see hbase-env.sh)
                </description>
        </property>
</configuration>
```

保存后分别把hbase的整个文件夹拷贝到其他服务器：
scp /usr/tools/hbase-1.2.0 root@jokeros2: /usr/tools/
scp /usr/tools/hbase-1.2.0 root@jokeros3: /usr/tools/
在hadoop的namenode节点上启动hbase服务
start-hbase.sh
启动后：jps
HRegionServer
HMaster
子节点
HRegionServer
启动顺序
Hadoop-hdfs——-》hadoop-yarn——》zookeeper——》hbase

hbase的数据存储结构

kv的格式



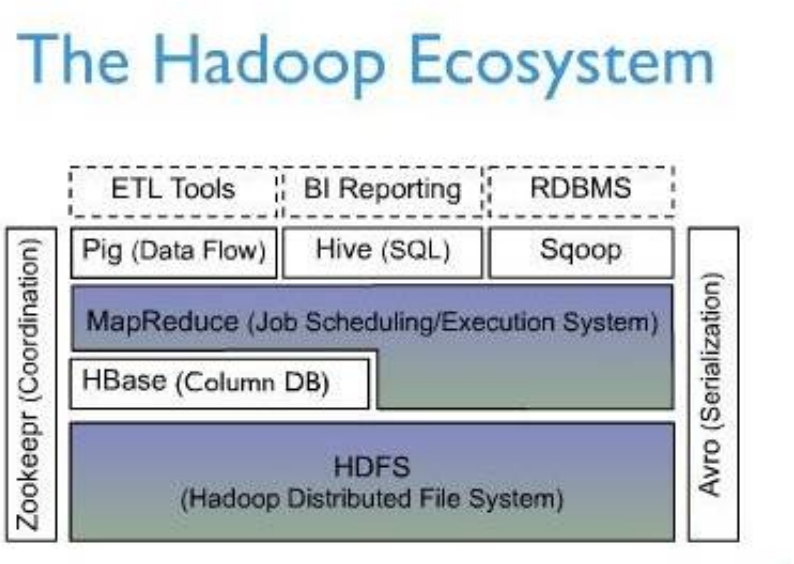
hbase的特点

> hbase是反模式的数据库,列式存储,用空间换性能,储存空间需要比较大,查询比较快,而数据库是模式,为了减少冗余数据
> hbase是稀疏的,有字段就存储没有的话就不在存储,而rdbms不管有没有值都会存在字段,所以hbase也节约了空间,灵活性比较高,但是规则低,所以由自行的约定规则,不在强加

命令

> hbase shell 命令
> create_namespace ‘bd14’ //创建数据库
> create ‘bd14:user’,’i’,’c’ //创建表 指定列簇
> list_namespace_tables ‘bd14’ //查看表 指定数据库
> list_namespace //查看当前数据库
> describe ‘bd14:user’ //查看表结构 指定表名字 数据库:表名 并且用单引号
> put ‘bd14:user’,’1’,’i:username’,’zhangsan’ 插入数据 ‘表名”rowkey”列名”时间戳’ 如果不写列名,那么也会加进去,只不过是列名为空
> get ‘bd14:user’,’1’ //查看数据 在value前边由一个时间戳
> ‘bd14:user’,’1’,’c:email’ +’时间戳’ 可以加上时间戳更精确,前边是必填 rpwkey+列名确定一条数据
> truncate ‘bd14:user’ //清空表数据
> disable ‘bd14:user’ drop ‘bd14:user’ //删除表结构,需要先disable,然后再执行drop命
> scan ‘bd14:user’ 查看表中数据,是看整张表

hbase系统



其中HBase位于结构化存储层，Hadoop HDFS为HBase提供了高可靠性的底层存储支持，Hadoop MapReduce为HBase提供了高性能的计算能力，Zookeeper为HBase提供了稳定服务和failover机制。此外，Pig和Hive还为HBase提供了高层语言支持，使得在HBase上进行数据统计处理变的非常简单。 Sqoop则为HBase提供了方便的RDBMS数据导入功能，使得传统数据库数据向HBase中迁移变的非常方便。

hbase的简介

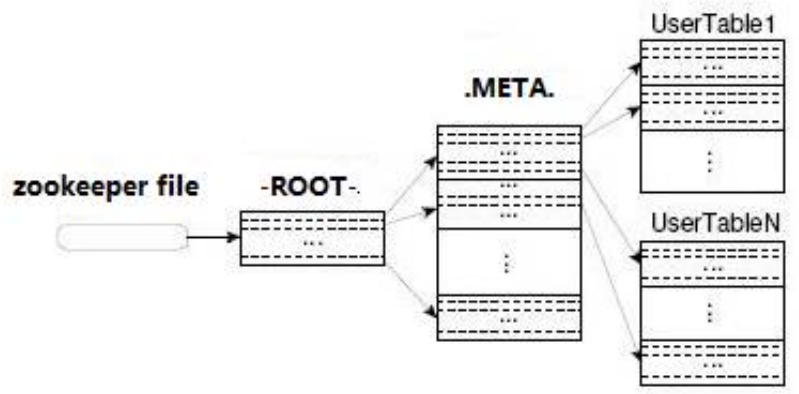
它的使用场景: 一次写入,多次读写,历时数据的查询,电台平台的订单查询(一次写入多次读写,话单信息等,日志信息)并且内部是以kv对的形式进行存储,它的索引即key的值,会进行排序,在存储之前会进行排序,这样就是有序的所以能快速查询,另外汇出吸纳冗余存储rowkey和列簇等,所以为了节省空间,尽可能的短一点,最好一个字母代表,在增删改的时候,请求过来会先发送大zookeeper上,得到meta表的位置,然后在访问meta表得知data的信息
发送请求—>zookeeper的hbase/meta-region-server找到hbase上的meta表的位置—>查询meta表,找到数据在哪个regionserver上—>查询rowkey等key来获取value

Hbase接口

1. Native Java API，最常规和高效的访问方式，适合Hadoop MapReduce Job并行批处理HBase表数据
2. HBase Shell，HBase的命令行工具，最简单的接口，适合HBase管理使用
3. Thrift Gateway，利用Thrift序列化技术，支持C++，PHP，Python等多种语言，适合其他异构系统在线访问HBase表数据
4. REST Gateway，支持REST 风格的Http API访问HBase, 解除了语言限制
5. Pig，可以使用Pig Latin流式编程语言来操作HBase中的数据，和Hive类似，本质最终也是编译成MapReduce Job来处理HBase表数据，适合做数据统计
6. Hive，当前Hive的Release版本尚没有加入对HBase的支持，但在下一个版本Hive 0.7.0中将会支持HBase，可以使用类似SQL语言来访问HBase

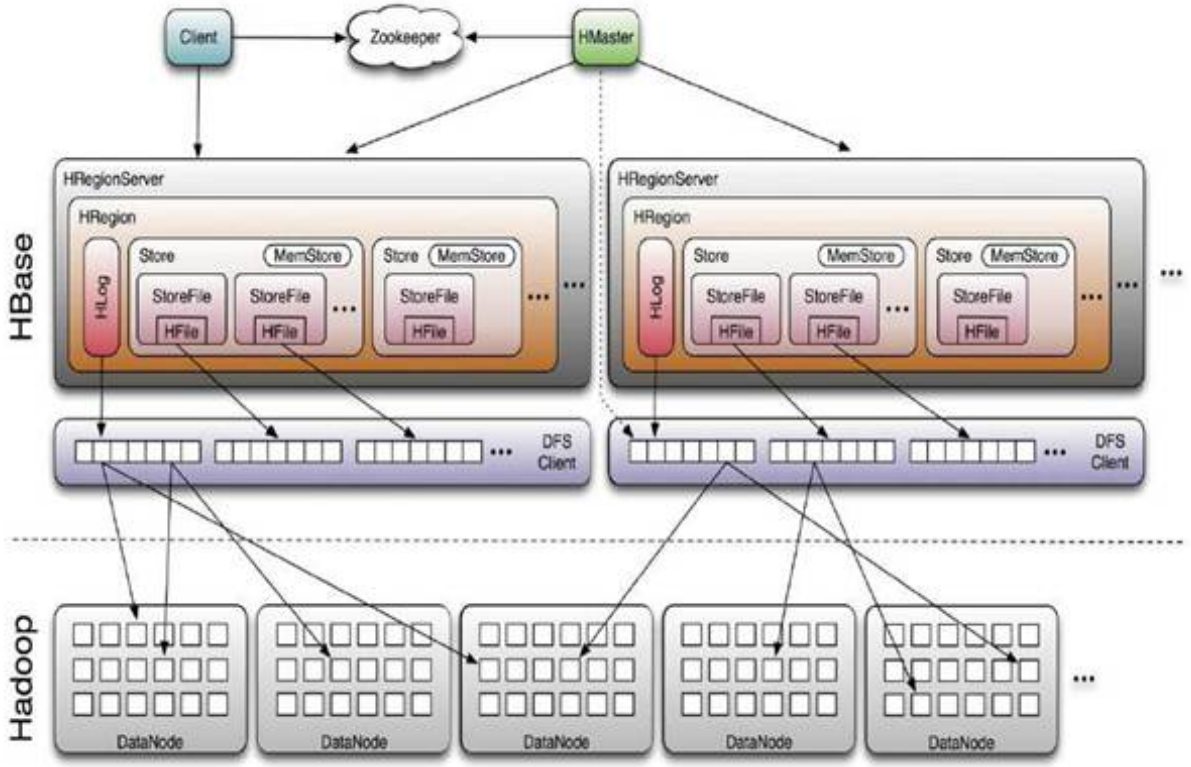
hbase的数据模型

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



Client访问用户数据之前需要首先访问zookeeper，然后访问-ROOT-表，接着访问.META.表，最后才能找到用户数据的位置去访问，中间需要多次网络操作，不过client端会做cache缓存。

hbase的系统架构



Client

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

Zookeeper

Zookeeper Quorum中除了存储了-ROOT-表的地址和HMaster的地址，HRegionServer也会把自己以Ephemeral方式注册到Zookeeper中，使得HMaster可以随时感知到各个HRegionServer的健康状态。此外，Zookeeper也避免了HMaster的单点问题，见下文描述

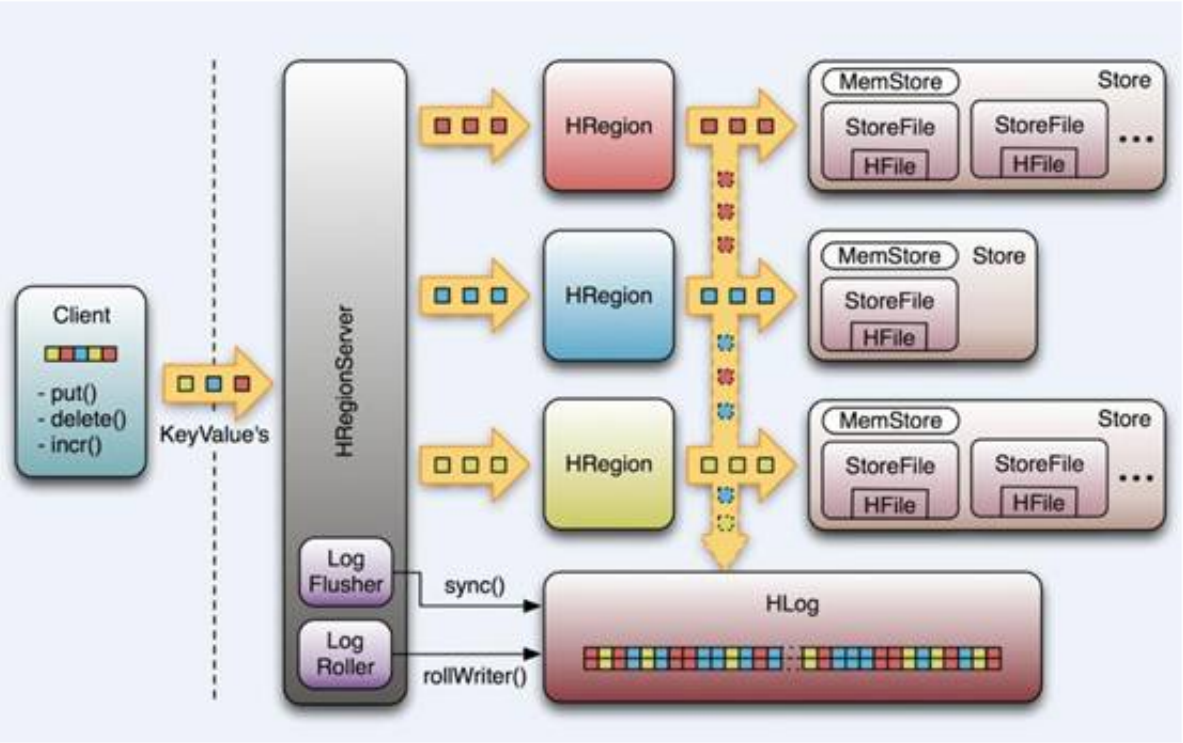
HMaster

HMaster没有单点问题，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在功能上主要负责Table和Region的管理工作：

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

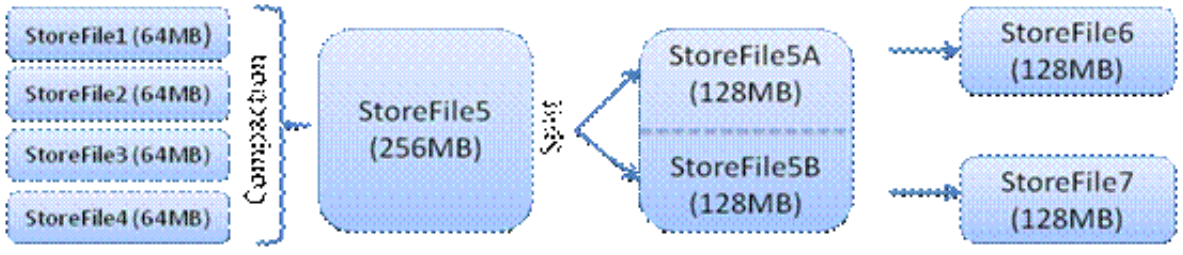
HRegionServer

HRegionServer主要负责响应用户I/O请求，向HDFS文件系统中读写数据，是HBase中最核心的模块。



注意:HRegin是表的一部分,它由rowkey划分范围,HR

RegionServer内部管理了一系列HRegion对象，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，可以看出每个Column Family其实就是一个集中的存储单元，因此最好将具备共同IO特性的column放在一个Column Family中，这样最高效。
HStore存储是HBase存储的核心了，其中由两部分组成，一部分是MemStore，一部分是StoreFiles。MemStore是Sorted Memory Buffer，用户写入的数据首先会放入MemStore，当MemStore满了以后会Flush成一个StoreFile（底层实现是HFile），当StoreFile文件数量增长到一定阈值，会触发Compact合并操作，将多个StoreFiles合并成一个StoreFile，合并过程中会进行版本合并和数据删除，因此可以看出HBase其实只有增加数据，所有的更新和删除操作都是在后续的compact过程中进行的，这使得用户的写操作只要进入内存中就可以立即返回，保证了HBase I/O的高性能。当StoreFiles Compact后，会逐步形成越来越大的StoreFile，当单个StoreFile大小超过一定阈值后，会触发Split操作，同时把当前Region Split成2个Region，父Region会下线，新Split出的2个孩子Region会被HMaster分配到相应的HRegionServer上，使得原先1个Region的压力得以分流到2个Region上。下图描述了Compaction和Split的过程：



在理解了上述HStore的基本原理后，还必须了解一下HLog的功能，因为上述的HStore在系统正常工作的前提下是没有问题的，但是在分布式系统环境中，无法避免系统出错或者宕机，因此一旦HRegionServer意外退出，MemStore中的内存数据将会丢失，这就需要引入HLog了。每个HRegionServer中都有一个HLog对象，HLog是一个实现Write Ahead Log的类，在每次用户操作写入MemStore的同时，也会写一份数据到HLog文件中（HLog文件格式见后续），HLog文件定期会滚动出新的，并删除旧的文件（已持久化到StoreFile中的数据）。当HRegionServer意外终止后，HMaster会通过Zookeeper感知到，HMaster首先会处理遗留的 HLog文件，将其中不同Region的Log数据进行拆分，分别放到相应region的目录下，然后再将失效的region重新分配，领取 到这些region的HRegionServer在Load Region的过程中，会发现有历史HLog需要处理，因此会Replay HLog中的数据到MemStore中，然后flush到StoreFiles，完成数据恢复。

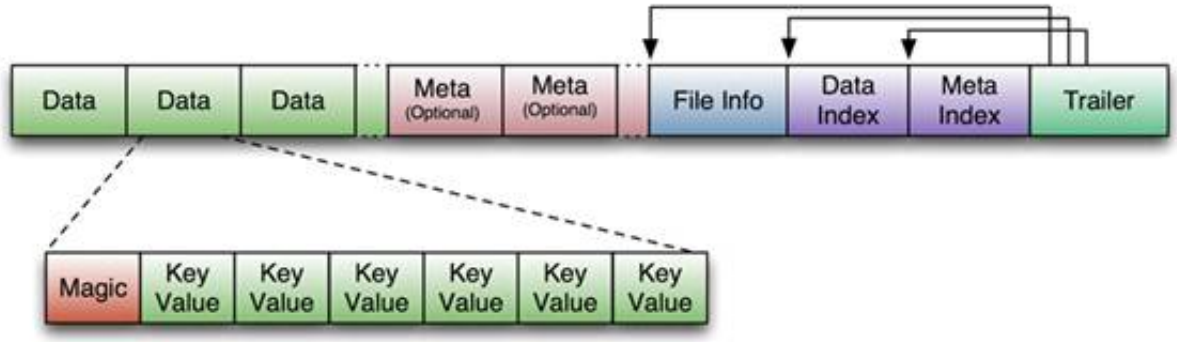
HBase存储格式

HBase中的所有数据文件都存储在Hadoop HDFS文件系统上，主要包括上述提出的两种文件类型：

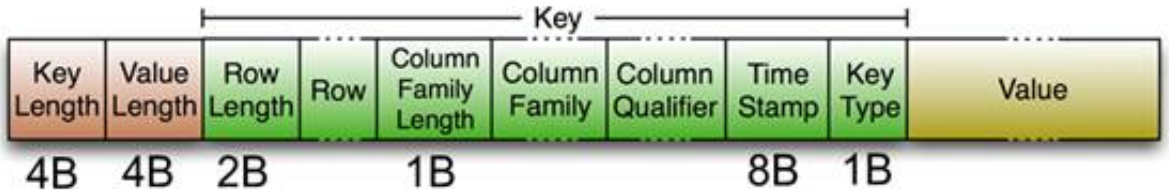
1. HFile， HBase中KeyValue数据的存储格式，HFile是Hadoop的二进制格式文件，实际上StoreFile就是对HFile做了轻量级包装，即StoreFile底层就是HFile
2. HLog File，HBase中WAL（Write Ahead Log） 的存储格式，物理上是Hadoop的Sequence File

HFile

首先HFile文件是不定长的，长度固定的只有其中的两块：Trailer和FileInfo。正如图中所示的，Trailer中有指针指向其他数据块的起始点。File Info中记录了文件的一些Meta信息，例如：AVG_KEY_LEN, AVG_VALUE_LEN, LAST_KEY, COMPARATOR, MAX_SEQ_ID_KEY等。Data Index和Meta Index块记录了每个Data块和Meta块的起始点。
Data Block是HBase I/O的基本单元，为了提高效率，HRegionServer中有基于LRU的Block Cache机制。每个Data块的大小可以在创建一个Table的时候通过参数指定，大号的Block有利于顺序Scan，小号Block利于随机查询。每个Data块除了开头的Magic以外就是一个个KeyValue对拼接而成, Magic内容就是一些随机数字，目的是防止数据损坏。后面会详细介绍每个KeyValue对的内部构造。

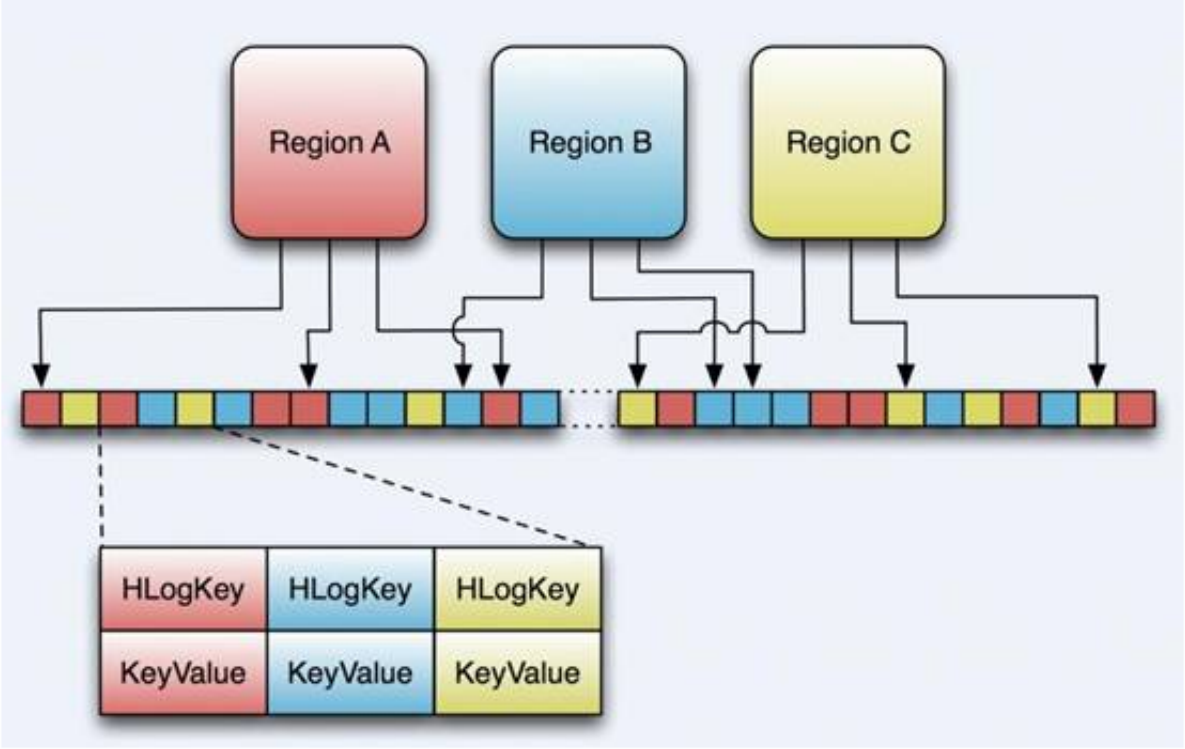


HFile里面的每个KeyValue对就是一个简单的byte数组。但是这个byte数组里面包含了很多项，并且有固定的结构。我们来看看里面的具体结构：



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Time Stamp和Key Type（Put/Delete）。Value部分没有这么复杂的结构，就是纯粹的二进制数据了。

HLog File



上图中示意了HLog文件的结构，其实HLog文件就是一个普通的Hadoop Sequence File，Sequence File 的Key是HLogKey对象，HLogKey中记录了写入数据的归属信息，除了table和region名字外，同时还包括 sequence number和timestamp，timestamp是“写入时间”，sequence number的起始值为0，或者是最近一次存入文件系统中sequence number。
HLog Sequece File的Value是HBase的KeyValue对象，即对应HFile中的KeyValue