搜索技术博客－淘宝

关注技术 关注搜索 关注淘宝

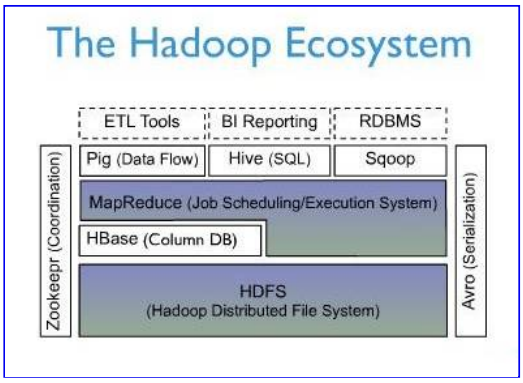
14
一

HBase技术介绍
莫问

HBase简介

HBase 一 Hadoop Database，是一个高可靠性、高性能、面向列、可伸缩的分布式存储系统，利用HBase技术可在廉价PC Server上搭建起大规模结构化存储集群。

HBase是Google Bigtable的开源实现，类似Google Bigtable利用GFS作为其文件存储系统，HBase利用Hadoop HDFS作为其文件存储系统；Google运行MapReduce来处理Bigtable中的海量数据，HBase同样利用Hadoop MapReduce来处理HBase中的海量数据；Google Bigtable利用 Chubby作为协同服务，HBase利用Zookeeper作为对应。



上图描述了Hadoop EcoSystem中的各层系统，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访问接口

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数据模型

Table & Column Family

Row Key	Timestamp	Column Family	
		URI	Parser
r1	t3	url=http://www.taobao.com	title=天天特价
	t2	host=taobao.com	
	t1		
r2	t5	url=http://www.alibaba.com	content=每天…
	t4	host=alibaba.com	

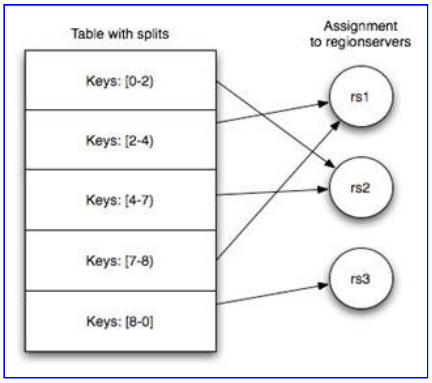
Ø Row Key: 行键，Table的主键，Table中的记录按照Row Key排序

Ø Timestamp: 时间戳，每次数据操作对应的时间戳，可以看作是数据的version number

Ø Column Family：列簇，Table在水平方向有一个或者多个Column Family组成，一个Column Family中可以由任意多个Column组成，即Column Family支持动态扩展，无需预先定义Column的数量以及类型，所有Column均以二进制格式存储，用户需要自行进行类型转换。

Table & Region

当Table随着记录数不断增加而变大后，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：



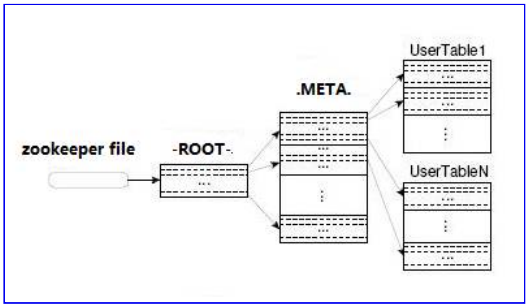
-ROOT- && .META. Table

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

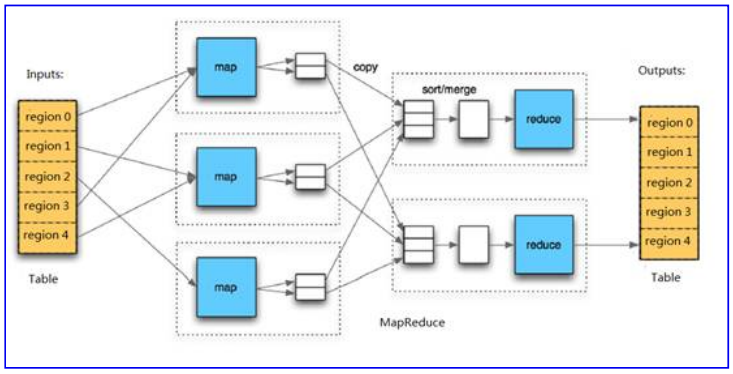
Ø Zookeeper中记录了-ROOT-表的location



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。

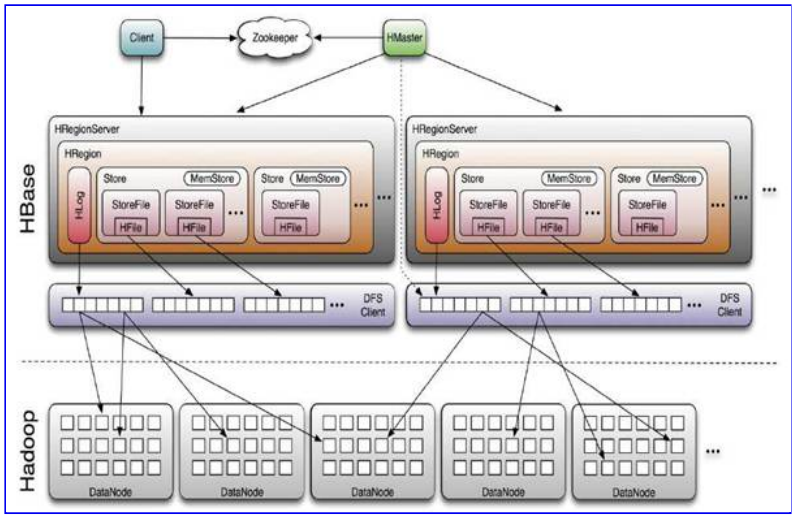
MapReduce on HBase

在HBase系统上运行批处理运算，最方便和实用的模型依然是MapReduce，如下图:



HBase Table和Region的关系，比较类似HDFS File和Block的关系，HBase提供了配套的TableInputFormat和TableOutputFormat API，可以方便的将HBase Table作为Hadoop MapReduce的Source和Sink，对于MapReduce Job应用开发人员来说，基本不需要关注HBase系统自身的细节。

HBase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

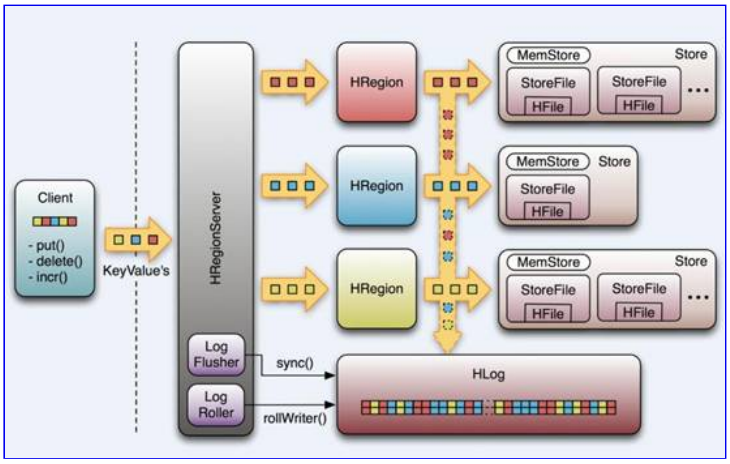
2. 管理HRegionServer的负载均衡，调整Region分布

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

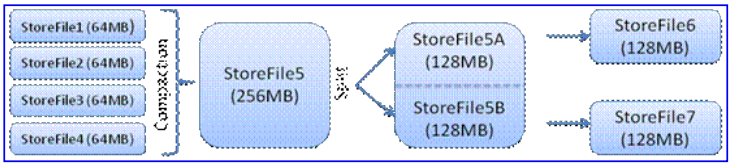
HRegionServer

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。



HRegionServer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

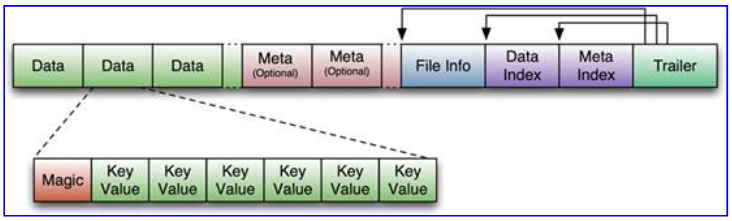
HBase中的所有数据文件都存储在Hadoop HDFS文件系统上，主要包括上述提出的两种文件类型：

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

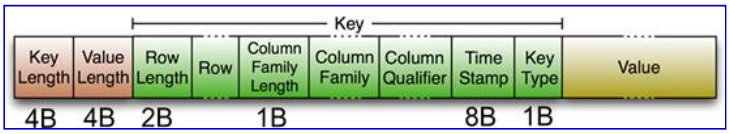
下图是HFile的存储格式：



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。

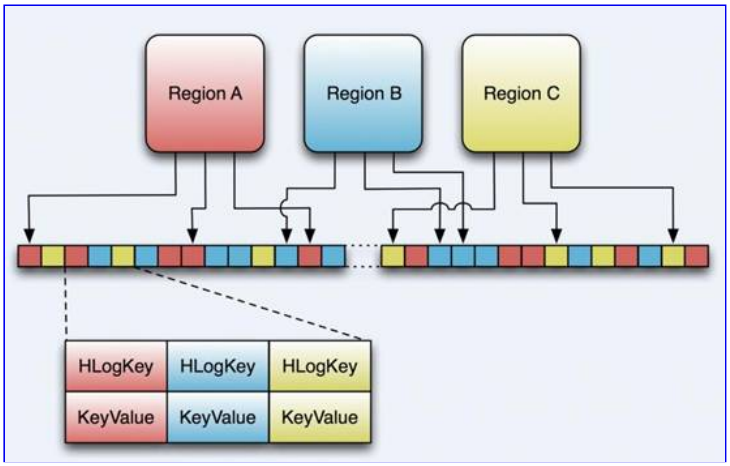
Data Block是HBase I/O的基本单元，为了提高效率，HRegionServer中有基于LRU的Block Cache机制。每个Data块的大小可以在创建一个Table的时候通过参数指定，大号的Block有利于顺序Scan，小号Block利于随机查询。每个Data块除了开头的Magic以外就是一个个KeyValue对拼接而成，Magic内容就是一些随机数字，目的是防止数据损坏。后面会详细介绍每个KeyValue对的内部构造。

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。

HLogFile



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，可参见上文描述。

结束

本文对HBase技术在功能和设计上进行了大致的介绍，由于篇幅有限，本文没有过多深入地描述HBase的一些细节技术。目前一淘的存储系统就是基于HBase技术搭建的，后续将介绍“一淘分布式存储系统”，通过实际案例来更多的介绍HBase应用。

Filed under – 导购搜索 11 Comments so far.

相关文章

- 几个随机算法
- 从“非诚勿扰”看淘宝算法效果测试
- 测试：淘宝网招聘平台现处测试阶段 正式上线在即

- 字符串匹配那些事（一）

11 Responses

1. chao on 14 一 2011

   哎哟，还不错噢！

   回复

2. shawny on 14 一 2011

   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.

   =>.META.表后还少一个访问USER表吧？也就是说要想知道key对应的regionServer需要查询4次才能确定其位置。

   回复

   - 莫问 on 17 一 2011

     是的，meta表中存了user表中各个region的信息，包括它所在的regionserver地址，然后再去访问user表的数据

     回复

3. layerJ on 21 一 2011

   对于这种key-value的DB系统，dao层该如何设计呢？是借鉴rdbms的orm思想？有没有最佳实践可以参考呢？

   多谢！

   回复

4. HFile存储格式 | 飛奔呱蝸牛 on 17 三 2011

   […] 参考：http://www.searchtb.com/2011/01/understanding-hbase.html […]

   回复

5. docete on 06 四 2011

   有没有测试过hbase+thrift（python or cpp)的随机读性能？

   回复

6. 莫问 on 08 四 2011

   淘宝内部有cpp的服务用thrift server访问hbase，相当于把thrift server做了个本地api转换代理，性能还不错

   回复

   - docete on 13 四 2011

     多大的数据量？单thrift qps是多少？我侧过 100M records，每个record 1k，单thrift只有100左右。这个数字相当不理想啊。

回复

7. 
pandonix on 13 四 2011

请教两个问题：RowKey的排序是在MemStore中完成的？另外，MemStore每次flush都对应一个StoreFile，换句话说，StoreFile是不可能再被插入修改的吧？

回复

8. 
莫问 on 03 五 2011

rowkey是keyvalue对象的一部分，可以理解为是最前面的一部分，keyvalue对象在memstore中是以sortedmap的形式排序的。memstore flush出来的hfile文件是不能被修改的，所有的修改操作都是通过合并hfile文件，新版本数据覆盖旧版本数据完成的

回复

9. 
黄兵 on 04 七 2011

是不是淘宝现在用把集群的MySQL向Hbase演进？？？

淘宝是不是在数据库架构层面转型？？？？ 采用Hbase作为数据的存储。 NoSQL真的可以成为主力1？？

回复

留言

Name (required) 名字

Email (required) 邮件地址

URL 博客地址

提交

Comment

留言

订阅

Subscribe to feed
get the latest updates!

搜索

- 最近更新
  - Doclist压缩方法简介

- 字符串匹配那些事（一）
- autoconf AC_ARG_WITH,AC_CACHE_CHECK,AC_TRY_LINK宏学习
- spinlock剖析与改进
- google group varint 无损压缩解压算法的高效实现
- 提升磁盘IO性能的几个技巧
- Redis内存存储结构分析
- BigPipe学习研究

标签云

A/B Test autoconf BTS Cassandra checkstyle CppUnit Google GPU Hadoop java JNI JUnit Leslie Lamport Linux mangodb MapReduce Multivariate Test NoSQL paoxs PHP Protocol Buffers redis Samba Scrapy spinlock Streaming zookeeper 一淘 分布式抓取 分布式锁 前端 单元测试 导购搜索 性能 搜索产品优化 搜索引擎 算法 网络爬虫

近期评论

- Redis内存存储结构分析 « 同一屋檐下 在 Redis内存存储结构分析 上的评论
- medal 在 Redis内存存储结构分析 上的评论
- BigPipe学习研究-搞前端的可以看一下，属于高级部分了 | haohtml's blog 在 BigPipe学习研究 上的评论
- cfanbo 在 BigPipe学习研究 上的评论
- Redis作者谈Redis应用场景 | haohtml's blog 在 Redis内存存储结构分析 上的评论
- 淘宝搜索：定向抓取网页技术漫谈 | 标点符 在 定向抓取漫谈 上的评论
- 字符串匹配那些事（一） | 听雨轩 在 字符串匹配那些事（一） 上的评论
- 不解 在 BigPipe学习研究 上的评论

友情链接

- 淘宝DBA团队
- 淘宝UED团队
- 淘宝招聘
- 淘宝数据平台团队
- 淘宝核心系统团队
- 淘宝质量保障团队
- 量子统计官方博客

功能

- 注册
- 登录
- 文章 RSS
- 评论 RSS
- WordPress.org