# Hbase架构和工作原理

Hbase全称为Hadoop Database，即Hbase是Hadoop的数据库，是一个高可靠性、高性能、面向列、可伸缩、 实时读写的分布式数据库。Hbase利用Hadoop的HDFS作为其文件存储系统，利用Hadoop的MapReduce来处理Hbase中的海量数据，利用zookeeper作为其协调工具。   
本篇文章将重点介绍Hbase三个方面的内容：Hbase体系结构（架构）的介绍、Hbase shell的操作、Hbase的Java api的客户端操作。   
(一)Hbase的架构介绍   
   
Hbase的体系结构是一个主从式的结构，主节点Hmaster在整个集群当中只有一个在运行，从节点HRegionServer有很 多个在运行，主节点Hmaster与从节点HRegionServer实际上指的是不同的物理机器，即有一个机器上面跑的进程是Hmaster，很多机器上面跑的进程是HRegionServer，Hmaster没有单点问题，Hbase集群当中可以启动多个Hmaster，但是通过zookeeper的事件处理机制保证整个集群当中只有一个Hmaster在运行。   
既然Hbase是数据库，那么数据库从根本上来说就是存储表table的，但是我们在这里必须注意一下，Hbase并非是传统的关系型数据库(例如：Mysql、Oracle)，而是非关系型数据库，因为Hbase是一个面向列的分布式存储系统。下面我们有必要介绍一下Hbase中表table的数据模型，先给大家呈现一个hbase数据库的表users：   
   
特别注意：表中的空白单元并不表示有这个单元存在，在传统的数据库中，空白单元表示该单元存在其值为空(null,这是因为传统数据库总是结构化的)。但在Hbase中，画成二维表只是在逻辑上便于理解，其本质完全是非结构化的。   
我在下面举例子的过程中用的也是这张users表。   
上面这个表就是一个典型的hbase table，与传统的关系型数据库具有很大的差别，下面我们详细介绍有关table的相关概念：   
RowKey(行健)：table的主键，table中的记录默认按照RowKey升序排序。   
列族(Column Family):即表中的address、info。table在水平方向上有一个或者多个Column Family组成，一个Column Family中可以由任意多个Column(例如address中的province、city、country、town)组成，即列族支持动态扩展，无需预先定义Column的数量以及类型，所有Column均已二进制格式进行存储，用户需要自行进行类型转换。   
TimeStamp(时间戳)：每次用户对数据进行操作对应的时间，可以看做是数据的Version number。例如在上面的表中，xiaoming所对应的company有两个数据信息(alibaba、baidu)，而这两个单元格信息实际上是对应操作时间的，如下图所示：   
   
既然Hbase可以将表中的数据进行分布式存储，那么它到底是以怎样的形式进行分布式存储的呢？我们自然而然想到了HDFS这个分布式文件管理系统是将海量数据切分成若干个block块进行存储的，同理Hbase也采取了类似的存储机制， 将一个table切分成若干个region进行存储，下面我们就介绍Region的相关概念：   
当Table随着记录数不断增加而变大后，Table在行的方向上会被切分成多个Region，一个Region由[startkey,endkey) 表示，每个Region会被Master分散到不同的HRegionServer上面进行存储，类似于我的block块会被分散到不同的DataNode节点上面进行存储。下面是Hbase表中的数据与HRegionServer的分布关系，如图所示：   
   
接下来介绍在Hbase的体系结构中，Hmaster、HRegionServer、Zookeeper集群这三个角色的作用：   
   
Hmaster节点的作用：   
①不负责存储表数据，负责管理RegionServer的负载均衡（即防止某些RegionServer存储数据量大，有些   
RegionServer存储数据量小），调整RegionServer上面Region的分布   
②管理RegionServer的状态，例如在HRegionServer宕机后，负责失效HRegionServer上Regions的迁移   
③在Region Split后，负责新Region的分配   
HRegionServer节点的作用：   
HRegionServer主要负责响应用户的I/O请求，即负责响应用户向表中的读写操作，是Hbase体系结构中最核心的模块。HRegionServer内部存储了很多的HRegion，就像DataNode节点中存储了很多的Block块一样，从上图Hbase完整的体系结构中我们可以看到，HRegion实际上是由很多个HStore组成的，所谓HStore就是表中的一个Column Family，可以看出每个Column Family其实就是一个集中的存储单元，这恰恰也帮助我们理解了为什么Hbase是NoSql系列的数据库，为什么是面向列的数据库，在Hbase的表设计中，我们最好将具备共同I/O特性的Column放在同一个列族中，这样读写才最高效，为了让大家更好的理解HRegionServer、HRegion、HStore、ColumnFamily四者之间的关系，我结合之前提到的users表画了一个四者关系的示意图：   
   
简单来说：就是HRegionServer服务器中存储了很多的HRegion，每个HRegion是由很过个HStore组成的，每个ColumnFamily就是一个HSore。   
在此还要简单介绍一下HLog与MemStore这两个角色的作用：   
Hlog：Hlog中存储了用户对表数据的最新的一些操作日志记录。   
MemSore：HRegion会将大量的热数据、访问频次最高的数据存储到MemStore中，这样用户在读写数据的时候不需要从磁盘中进行操作，直接在内存中既可以读取到数据，正因为MemStore这个重要角色的存在，Hbase才能支持随机，高速读取的功能。   
Zookeeper集群的作用：   
①通过zk集群的事件处理机制，可以保证集群中只有一个运行的Hmater   
②Zookeeper集群中记录了-ROOT-表的位置   
在这里顺便介绍一下Hbase中两张特殊的表：-ROOT-表与.META.表   
-ROOT-表：记录了所有.META.表的元数据信息，-ROOT-表只有一个Region   
.META.表：记录了Hbase中所有用户表的HRegion的元数据信息，.META.表可以有多个Region   
③Zookeeper集群实时监控着HRegionServer这些服务器的状态，将HRegionServer的上线和下线信息实时通知给Hmaster节点，使得Hmaster节点可以随时感知各个HRegionServer的健康状态。   
在上面我们依次介绍了HMaster、HRegionServer、zookeeper集群的作用，Client使用Hbase的RPC机制与Hmaster与HRegionServer进行通信，对于管理类操作，Client与Hmaster进行RPC进行通信，对于数据(表)的读写类操作，Client与HRegionServer进行通信。注意：在用户对数据表的读写过程中，与Hmaster是没有任何关系的，Hmaster在这一点上不同于我们的NameNode节点，可以看出由于zookeeper集群的存在，Hmaster节点的作用被大大弱化了。   
在Hbase的架构介绍中在介绍最后一点：Hbase的寻址机制，即Hbase在海量的表数据中，是如何找到用户所需要的表数据的呢？——Hbase是通过索引的机制解决了这个问题。   
   
Client访问用户数据之前需要首先访问zookeeper集群，通过zookeeper集群首先确定-ROOT-表在的位置，然后在通过访问-ROOT- 表确定相应.META.表的位置，最后根据.META.中存储的相应元数据信息找到用户数据的位置去访问。通过 这种索引机制解决了复杂了寻址问题。

