接下来我们介绍第3节内容，HBase的实现原理，主要内容包括HBase的功能组件、表和Region以及Region的定位。

先来看第1个主要内容，HBase的功能组件。Hbase的三个主要的功能组件分别是库函数和一个master的主服务器，另外一个是多个Region的服务器。Master主要是负责管理HBase表的分区信息，同时呢，它也要维护Region服务器里面的列表，还要负责分配Region，以及多个Region之间的负载均衡问题。Region服务器负责存储和维护分配给自己的Region，处理来自客户端的读写请求。一个大的表会被它分成很多个region，就是分区的意思啊，region那这个region呢，就是由这个region服务器来负责具体的维护和管理，那我们客户端在访问数据的时候，也都是直接和这个region服务器进行数据的存取，所以我们说，客户端并不会直接从这个master上面去存取数据啊，一般都是获得整个它的region的位置信息，以后呢直接跑去和相应的region服务器去打交道，去那里去存数据去取数据，而且这是一个非常关键的地方，也是我们说HBase设计比较突出的优点地方，什么呢，就说客户端它并不直接依赖于master去获得这个位置信息，我们的客户端要想取数据，可以不通过master就可以获得相关的数据的，具体的存储的这个类似信息，然后呢，直接跑到数据所存的地方去把数据取出来，那它是通过谁呢？后面我们会讲的，通过Zookeeper来获得Region位置信息来获得更清晰的，后面我们会通过一个叫三级选址来告诉大家，他是怎么获得具体数据的存储位置的。

对于Hbase当中的表的存储会被分成多个region，它是按照行键字典的顺序来进行排列的。在一开始的时候只有一个表只会存在一个region中，由于数据在不断的增长，这个region就会越来越大，会超过region的大小范围，从而region就会开始拆分，一个变成二个，当数据量还在变大时，这个拆分会继续，但是这个操作速度非常快，只是改一下指向信息就可以了，在读取数据时还是读的原存储文件。后面后台会运行一个合并过程，把你拆分的项目数据要进行重新的这种相关的操作，最终会把它写到一个新的文件当中去，写完以后他才会把最新的认证的信息告诉你，后面来的应用都会去读取这个新的文件，所以要知道就是他虽然有拆分，但是拆分速度是非常快的。

每个Region默认大小是100MB到200MB，在2006年以前是这样的硬件配置。现在对于每个Region最好的大小建议可以达到1GB－2GB。不同的Region可以分布在不同的Region服务器上，但是同一个region是不会被分配到多个region服务器上的，而一个region服务器也可存储10－1000个region。

Region：就是要查找的数据所在的Region.META.表：是一张元数据表，记录了用户表的Region信息以及RegionServer的服务器地址，.META.可以有多个regoin。.META.表中的一行记录就是一个Region，记录了该Region的起始行、结束行和该Region的连接信息。-ROOT-：是一张存储.META.表的表，记录了.META.表的Region信息，-ROOT-只有一个region 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的三层结构的步骤：（1）用户通过查找zk（zookeeper）的/hbase/root-region-server节点来知道-ROOT-表在什么RegionServer上。（2）访问-ROOT-表，查看需要的数据在哪个.META.表上，这个.META.表在什么RegionServer上。（3）访问.META.表查看查询的行健在什么Region范围里面。（4）连接具体的数据所在的RegionServer，这回就真的开始用Scan来遍历row了。

HBase的三层结构中每个层次的作用如下表所示，第一层Zookeeper文件主要记录了-ROOT-表的位置信息，第二层-ROOT-表主要记录了.META.表的Region位置信息.-ROOT-表只能有一个Region。通过-ROOT-表，就可以访问.META.表中的数据，第三层.META.表记录了用户数据表的Region位置信息，.META.表可以有多个Region，保存了HBase中所有用户数据表的Region位置信息。

为了加快访问速度，.META.表的全部Region都会被保存在内存中

假设.META.表的每行（一个映射条目）在内存中大约占用1KB，并且每个Region限制为128MB，那么，上面的三层结构可以保存的用户数据表的Region数目的计算方法是：

（-ROOT-表能够寻址的.META.表的Region个数）×（每个.META.表的 Region可以寻址的用户数据表的Region个数）

一个-ROOT-表最多只能有一个Region，也就是最多只能有128MB，按照每行（一个映射条目）占用1KB内存计算，128MB空间可以容纳128MB/1KB=217行，也就是说，一个-ROOT-表可以寻址217个.META.表的Region。

同理，每个.META.表的 Region可以寻址的用户数据表的Region个数是128MB/1KB=217。

最终，三层结构可以保存的Region数目是(128MB/1KB) × (128MB/1KB) = 234个Region。

客户端访问数据时的“三级寻址”，寻址过程客户端只需要询问Zookeeper服务器，不需要连接Master服务器。