我们再来看一下Hbase运行机制，主要内容包括HBase系统架构、Region服务器工作原理、Store工作原理、HLog工作原理。

这个图就是Hbase的系统架构，我们主要来看上面HBASE架构。

客户端：

使用HBase RPC机制与HMaster和HRegionServer进行通信

Client与HMaster进行通信进行管理类操作

Client与HRegionServer进行数据读写类操作

Zookeeper：

Zookeeper Quorum存储-ROOT-表地址、Master地址

Region服务器把自己以Ephedral方式注册到Zookeeper中，Master随时感知各个Region服务器的健康状况

Zookeeper避免Master单点问题 。

Master：

Master没有单点问题，HBase中可以启动多个Master，通过Zookeeper的Master Election机制保证总有一个Master在运行

主要负责Table和Region的管理工作：

1 管理用户对表的增删改查操作

2 管理RegionServer的负载均衡，调整Region分布

3 Region Split后，负责新Region的分布

4 在RegionServer停机后，负责失效RegionServer上Region迁移

RegionServer：

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

RegionServer管理一些列Region对象；

每个Region对应Table中一个Region，Region由多个Store组成；

每个Store对应Table中一个Column Family的存储；

Column Family就是一个集中的存储单元，故将具有相同IO特性的Column放在一个Column Family会更高效。

Store：

HBase存储的核心。由MemStore和StoreFile组成。

MemStore是Sorted Memory Buffer。

用户写入数据的流程：Client写入 -> 存入MemStore，一直到MemStore满 -> Flush成一个StoreFile，直至增长到一定阈值 -> 触发Compact合并操作 -> 多个StoreFile合并成一个StoreFile，同时进行版本合并和数据删除 -> 当StoreFiles Compact后，逐步形成越来越大的StoreFile -> 单个StoreFile大小超过一定阈值后，触发Split操作，把当前Region Split成2个Region，Region会下线，新Split出的2个孩子Region会被Master分配到相应的RegionServer上，使得原先1个Region的压力得以分流到2个Region上。

由此过程可知，HBase只是增加数据，有所得更新和删除操作，都是在Compact阶段做的，所以，用户写操作只需要进入到内存即可立即返回，从而保证I/O高性能。当用户读取数据时，Region服务器会首先访问MemStore缓存，如果找不到，再去磁盘上面的StoreFile中寻找。

对于缓存的刷新问题，系统会采用周期性把MemStore缓存内容写到磁盘StoreFile文件中，同时清空缓存并写入Hlog里面一个标记，这种设计可以极大地提升HBase的写性能。MemStore对于读性能也至关重要，假如没有MemStore，读取刚写入的数据就需要从文件中通过IO查找，这种代价显然是昂贵的！所以，在一个Store包含多个StoreFile文件。每个Region服务器都有一个自己的HLog 文件，每次都检查这个文件是否发生新一写操作，如果发现，先写MemStore，再刷写到StoreFile，最后删除旧的Hlog文件，从而为用户提供服务。

StoreFile的合并，当storeFile数量达到一定程度时，会影响查找速度，所以系统可以调用store.compact()把多个合并成一个，不过在合并过程中，耗费资源比较大，因此要设置一个阈值，当达到这个值时才合并操作。

对于store,它是region服务器的核心部分，如图4－11所示，一般多个StoreFile合并成一个StoreFile,合并成的StoreFile又超过了一定阈值后又会解发分裂操作，随后一个父Region被分裂成两个子Region。

再来看一下Hlog的工作原理，在分布式环境中，系统肯定有出错的情况，一旦出错用什么来恢复系统呢？Hbase就是依靠Hlog来进行恢复系统。在Hbase中每个Region都会配一个Hlog文件的，也是一种预写式日志，用户更新数据时必须先写日志，才能写MemStore进行缓存，而且，MemStore缓存内容对应的日志要全部写入磁盘后，缓存内容才能写到磁盘。

对于系统中region服务器发生故障时，系统中有一个Zookeeper会负责监测这些故障，直接通知Master,这时Master接到通知后会首先处理故障Region服务器上面遗留的HLog文件，这个文件中包含了多个Region对象的日志记录。系统会根据每条日志记录的Region对象，进行拆分HLog数据，分别放到相应Region对象的目录下，同时把这个对象相关的Hlog日志记录发送给相应的region服务器，Region服务器领取到分配给自己的Region对象以及与之相关的HLog日志记录以后，会按日志记录重做一遍操作，把日志记录写入MemStore缓存中，最后刷写到磁盘StoreFile文件中，实现数据恢复。Hlog共用日志有一个缺点就是恢复时需要分拆日志，到相应Region对象的目录下，有些繁琐，比较耗费资源。但有一个优点就是能提高写操作性能。