HBase技术原理

www.huawei.com

HUAWEI

目标

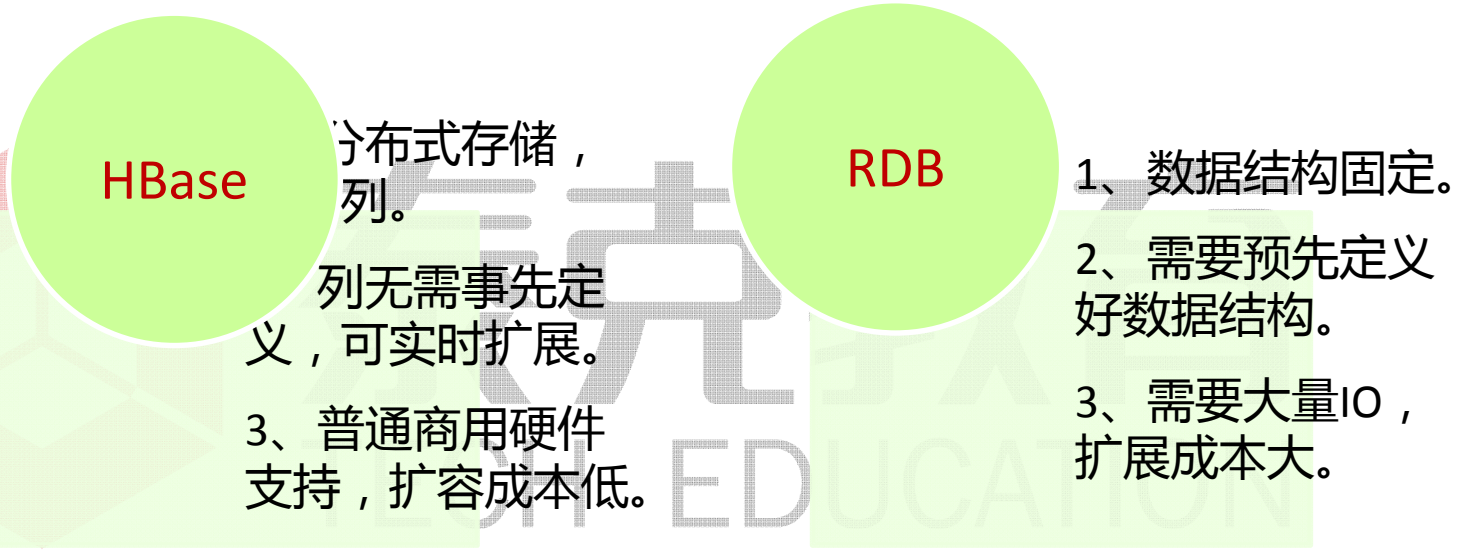
- 学完本课程后，您将能够：
  - 掌握HBase的系统架构
  - 掌握HBase的关键特性
  - 熟悉HBase的基本功能
  - 熟悉HBase华为增强特性

目录

HBase简介

- HBase是一个高可靠性、高性能、面向列、可伸缩的分布式存储系统。
  - 适合于存储大表数据（表的规模可以达到数十亿行以及数百万列），并且对大表数据的读、写访问可以达到实时级别。
  - 利用Hadoop HDFS（Hadoop Distributed File System）作为其文件存储系统，提供实时读写的分布式数据库系统。
  - 利用ZooKeeper作为协同服务。

HBase与RDB的对比

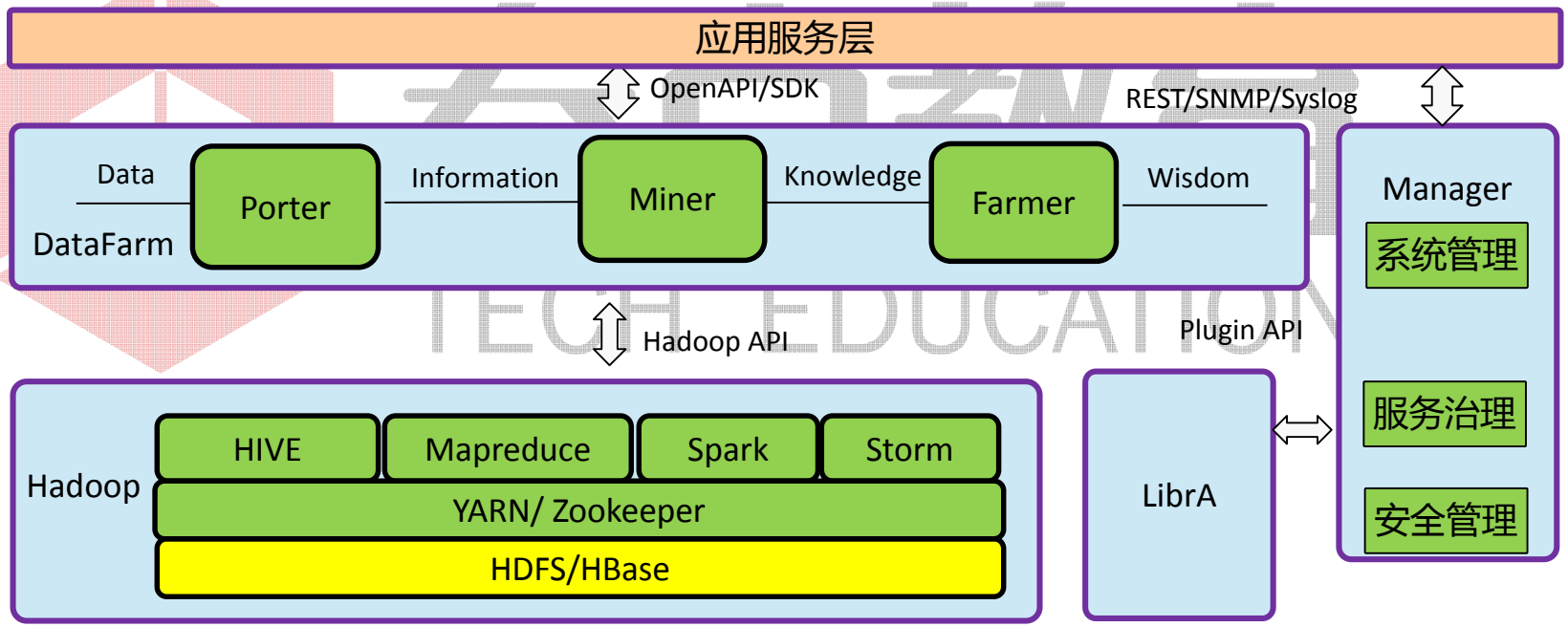


HBase应用场景

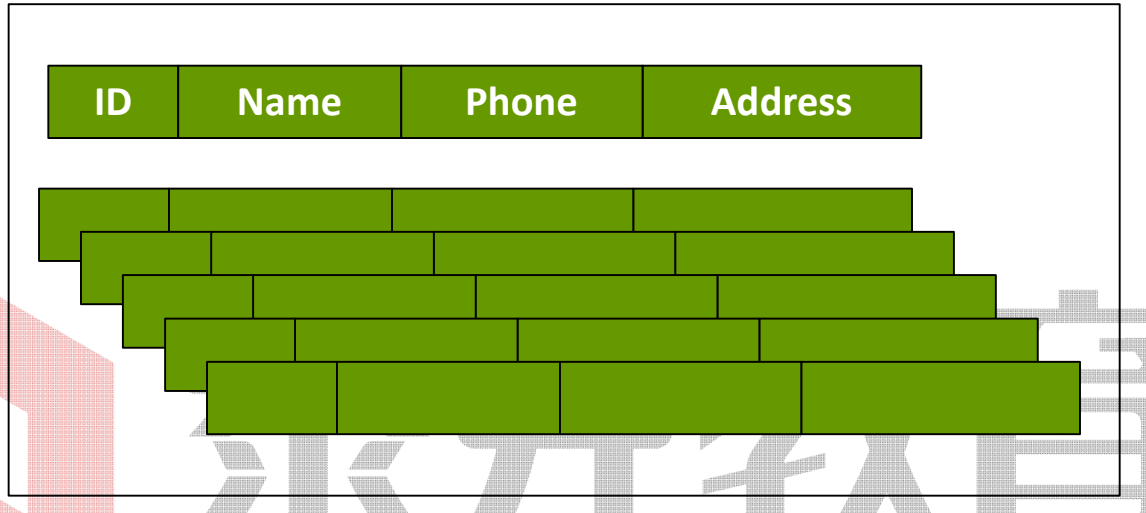
- HBase适合具有如下需求的应用：
  - 海量数据（TB、PB）。
  - 不需要完全拥有传统关系型数据库所具备的ACID特性。
  - 高吞吐量。
  - 需要在海量数据中实现高效的随机读取。
  - 需要很好的性能伸缩能力。
  - 能够同时处理结构化和非结构化的数据。

HBase在FusionInsight中的位置

- HBase作为一个高可靠性、高性能、面向列、可伸缩的分布式数据库，提供海量数据存储功能，用来解决关系型数据库在处理海量数据时的局限性。

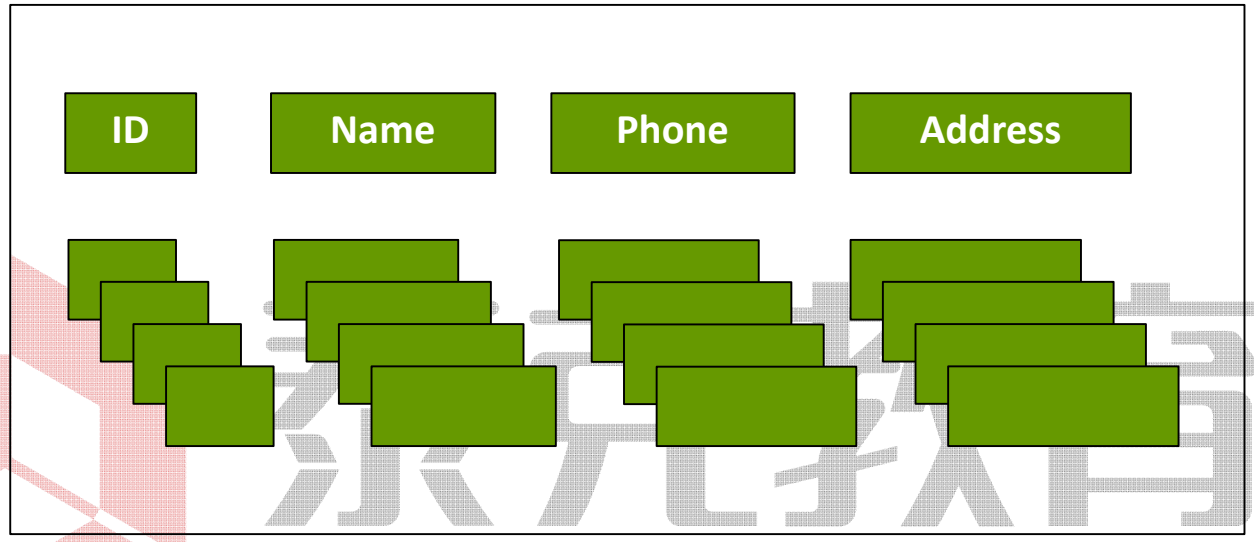


行存储



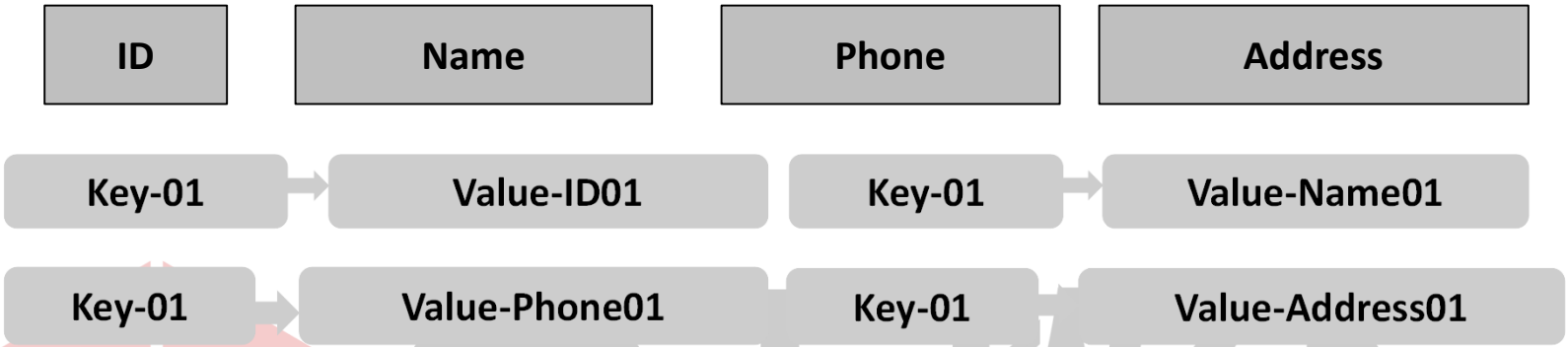
- 行存储，数据按行存储在底层文件系统中。通常，每一行会被分配固定的空间。
  - 优点：有利于增加/修改整行记录等操作；有利于整行数据的读取操作。
  - 缺点：单列查询时，会读取一些不必要的数据。

列存储



- 列存储，数据以列为单位，存储在底层文件系统中。
  - 优点：有利于面向单列数据的读取/统计等操作。
  - 缺点：整行读取时，可能需要多次I/O操作。

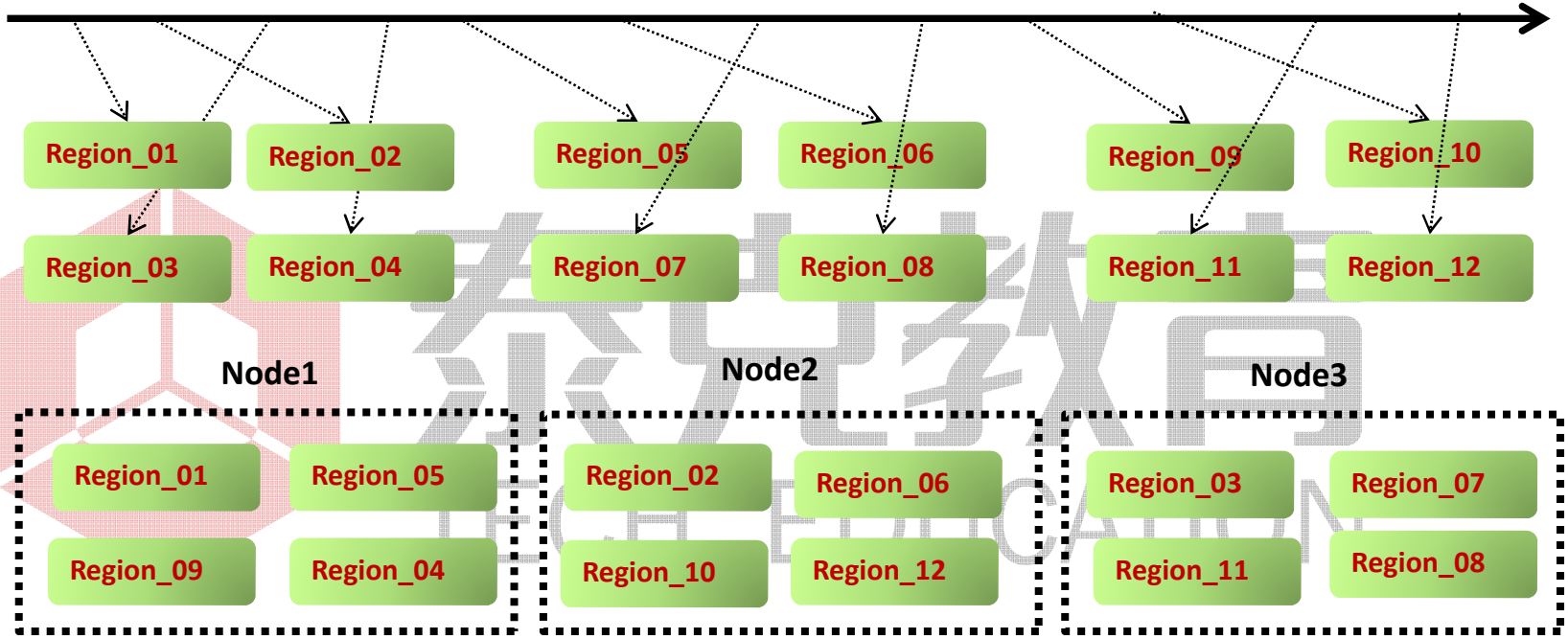
KeyValue存储模型（1）



- KeyValue具有特定的结构。Key部分被用来快速的检索一条数据记录，Value部分用来存储实际的用户数据信息。
- KeyValue作为承载用户数据的基本单元，需要保存一些对自身的描述信息，例如，时间戳，类型等等。那么，势必会有一定的结构化空间开销。
- 支持动态增加列，容易适应数据类型和结构的变化。以块为单元操作数据，列间、表间并无关联关系。

KeyValue存储模型（2）

- KeyValue型数据库数据分区方式--按Key值连续范围分区。



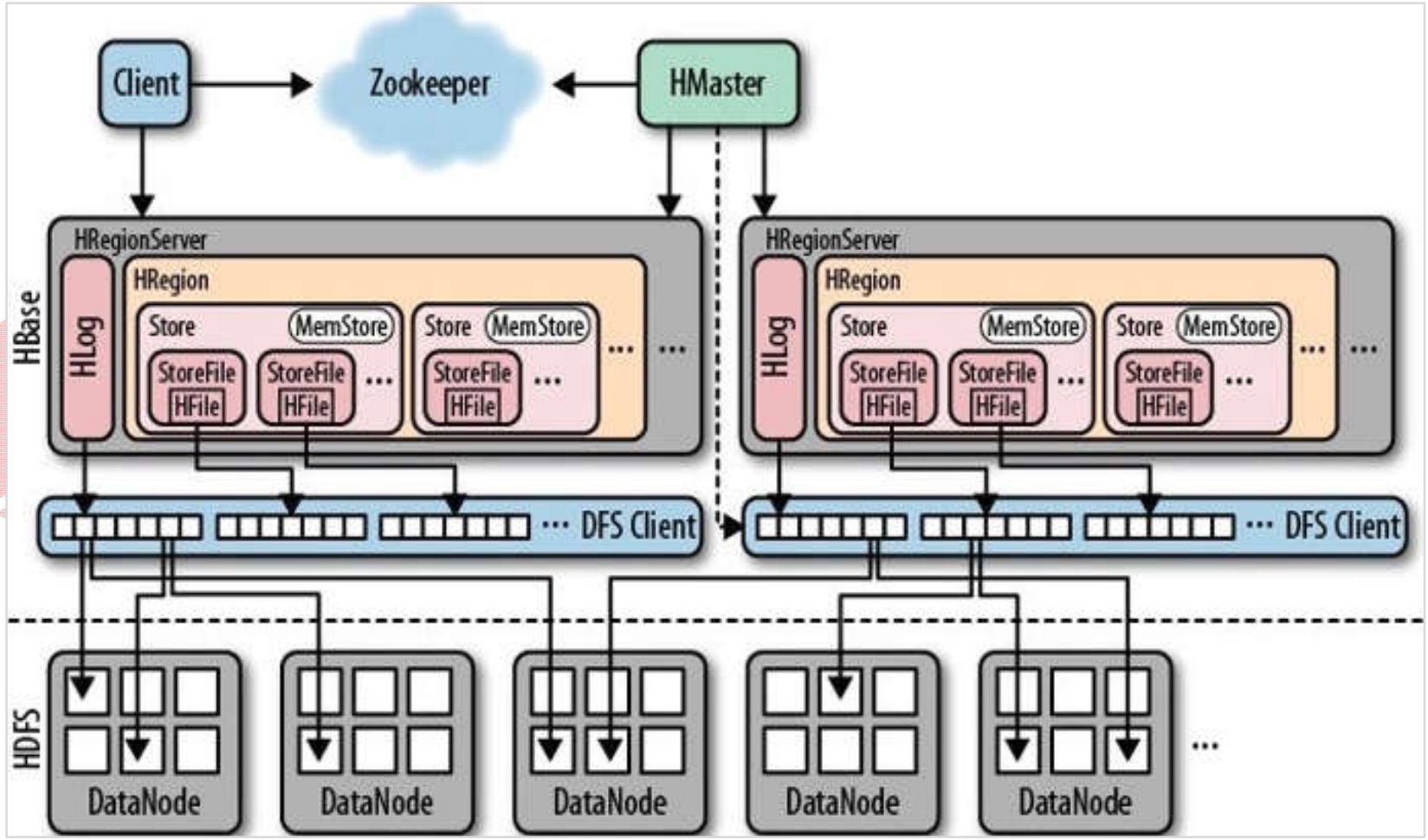
- 数据按照RowKey的范围(按RowKey的字典顺序)，划分为一个个的子区间。每一个子区间都是一个分布式存储的基本单元。

KeyValue存储模型（3）

- HBase的底层数据以KeyValue的形式存在，KeyValue具有特定的格式。
- KeyValue中拥有时间戳、类型等关键信息。
- 同一个Key值可以关联多个Value，每一个KeyValue都拥有一个Qualifier标识。
- 即使是Key值相同，Qualifier也相同的多个KeyValue，也可能有多个，此时使用时间戳来区分，这就是同一条数据记录的多版本。

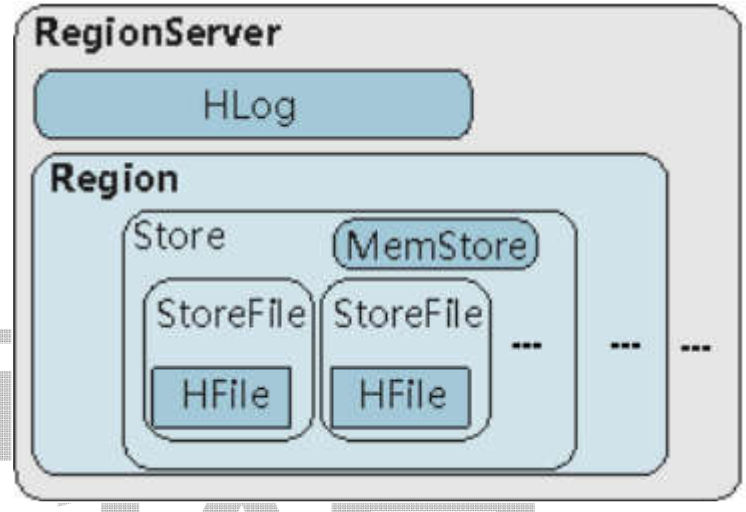
目录

HBase架构介绍（1）



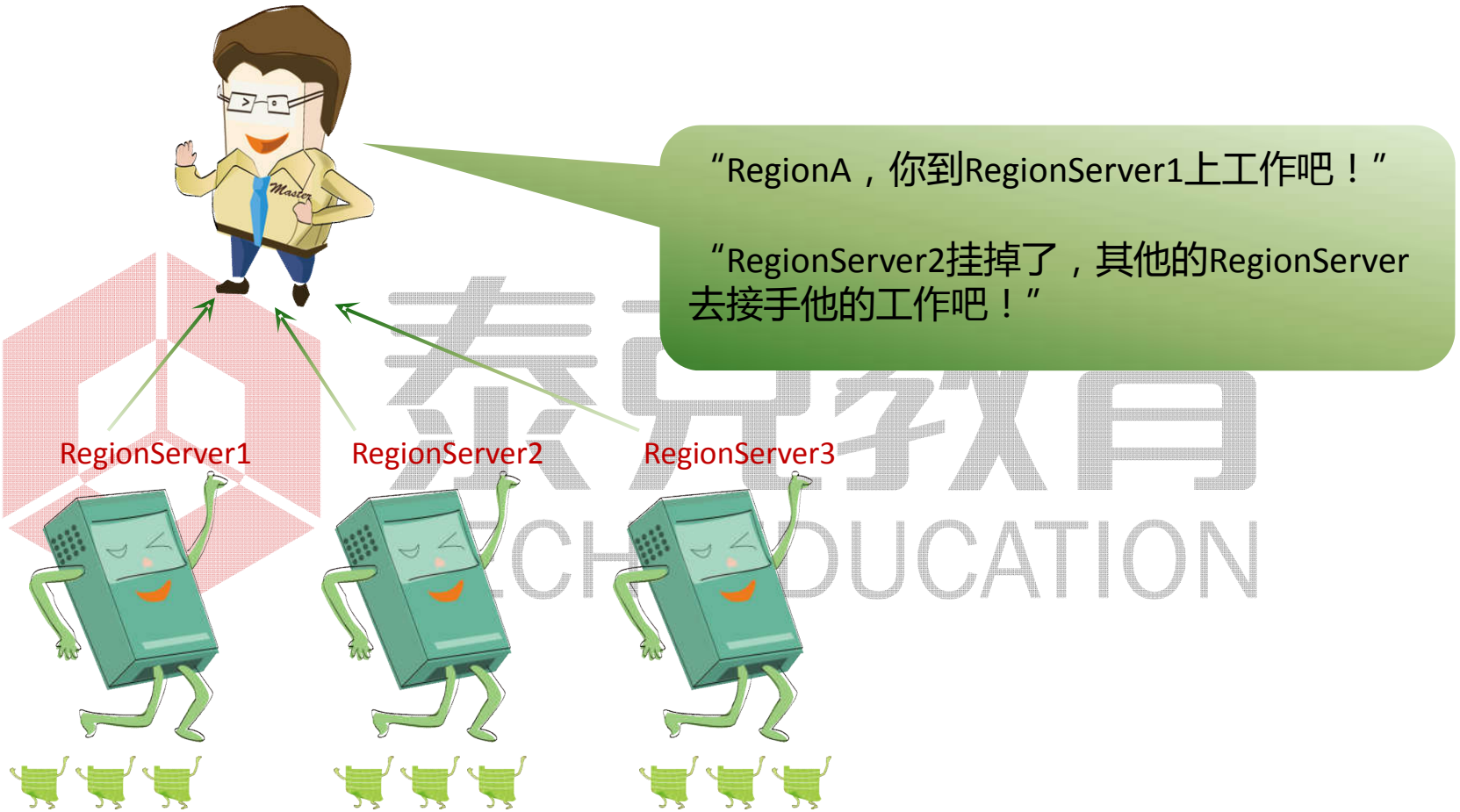
HBase架构介绍（2）

- Store：一个Region由一个或多个Store组成，每个Store对应图中的一个Column Family。
- MemStore：一个Store包含一个MemStore，MemStore缓存客户端向Region插入的数据。
- StoreFile：MemStore的数据flush到HDFS后成为StoreFile。



- Hfile：HFile定义了StoreFile在文件系统中的存储格式，它是当前HBase系统中StoreFile的具体实现。
- Hlog：HLog日志保证了当RegionServer故障的情况下用户写入的数据不丢失，RegionServer的多个Region共享一个相同的Hlog。

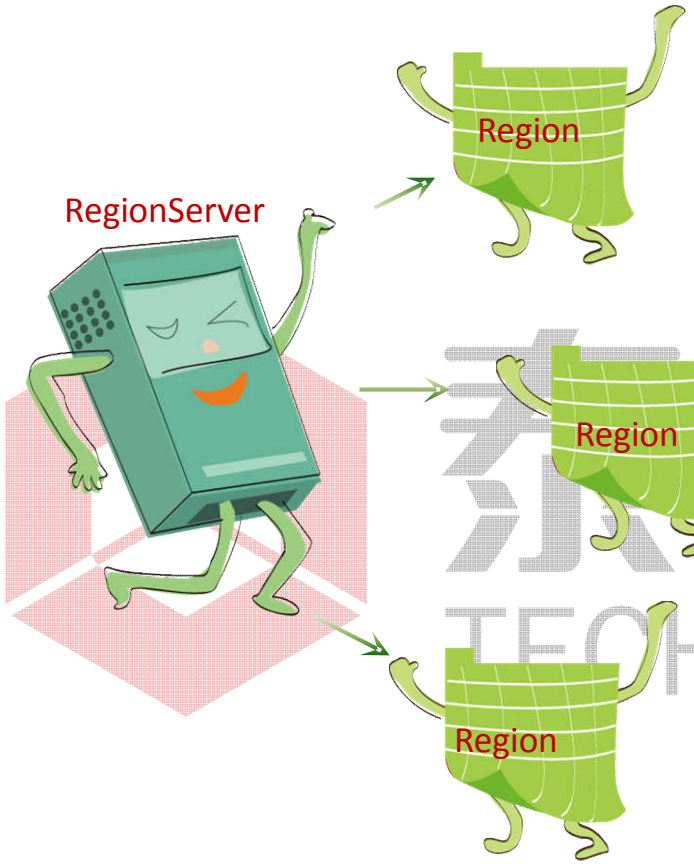
Hmaster (1)



HMaster (2)

- HMaster进程负责管理所有的RegionServer。
  - RegionServer Failover处理。
- 负责建表/修改表/删除表以及一些集群操作。
- HMaster进程负责所有Region的转移操作。
  - 新表创建时的Region分配。
  - 运行期间的负载均衡保障。
  - RegionServer Failover后的Region接管。

RegionServer

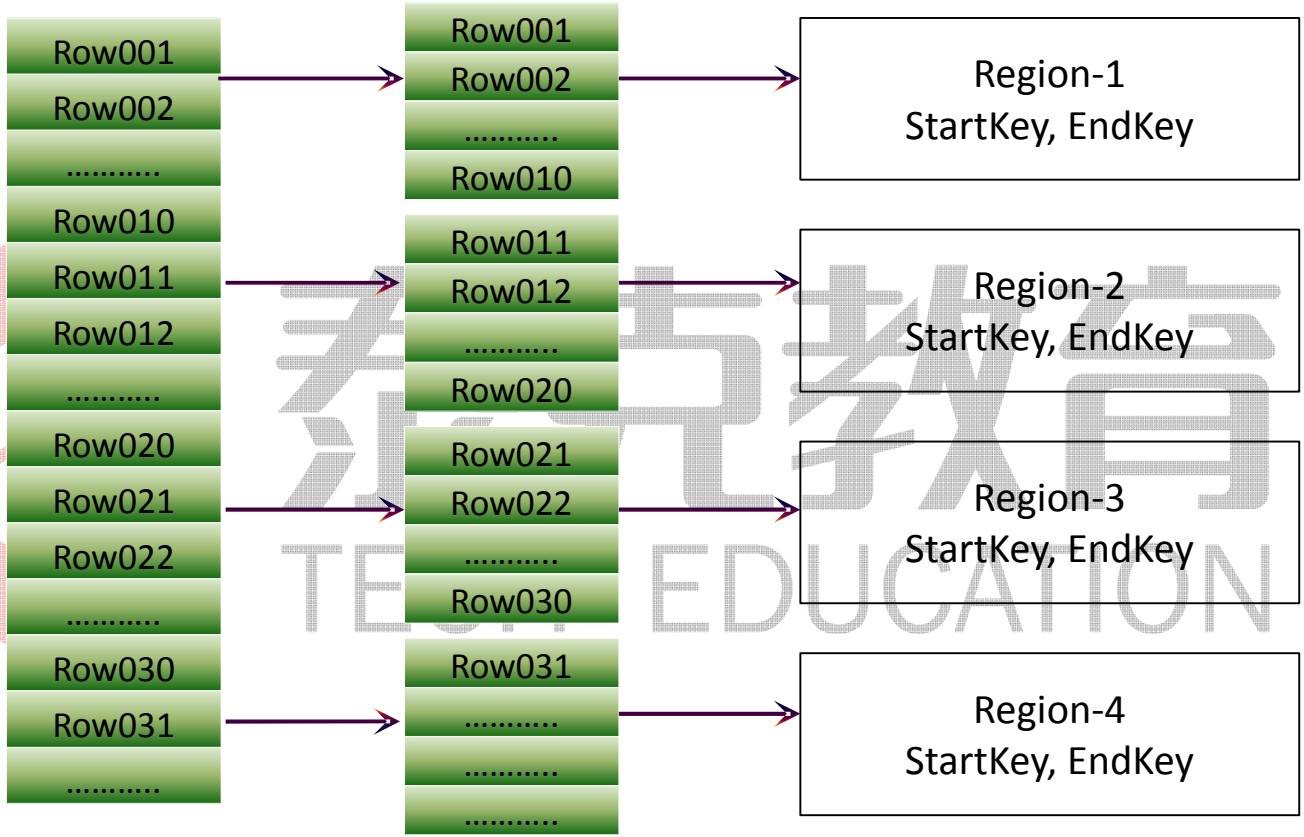


- RegionServer是HBase的数据服务进程。负责处理用户数据的读写请求。
- Region由RegionServer管理。所有用户数据的读写请求，都是和RegionServer上的Region进行交互。
- Region可以在RegionServer之间迁移。

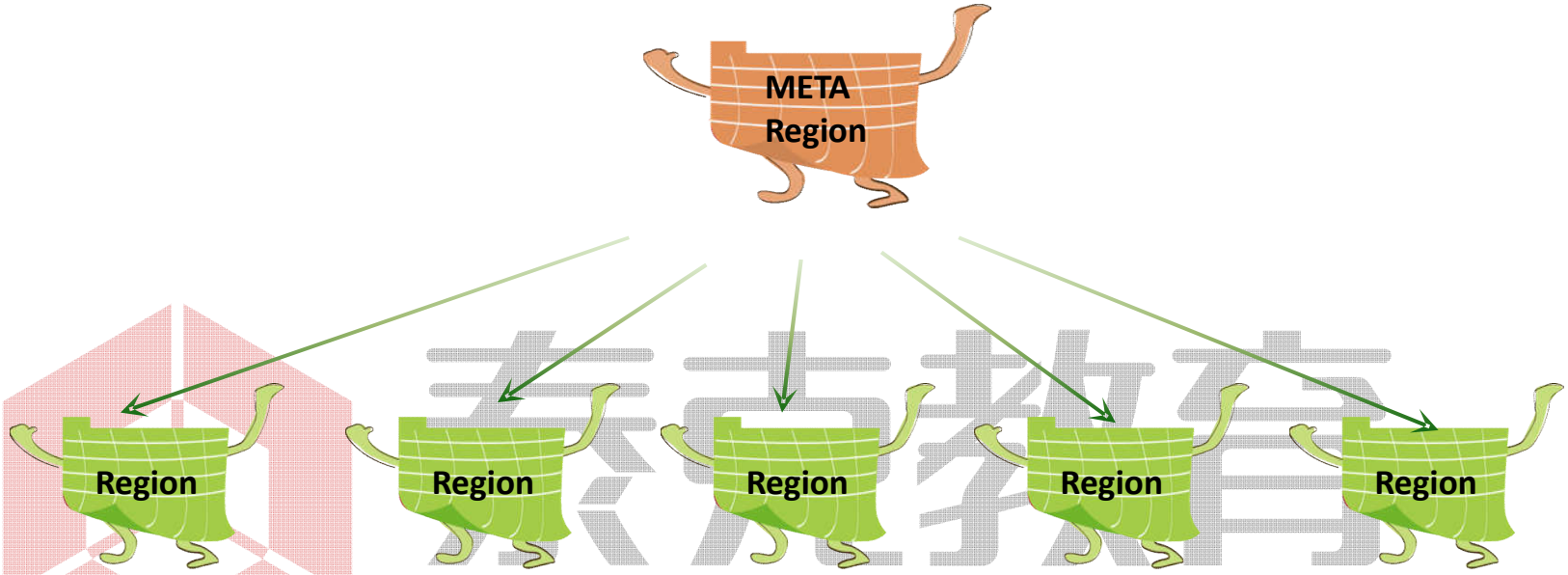
Region（1）

- 将一个数据表按Key值范围连续划分为多个的子表，这个子表，在HBase中被称作“Region”。
- 每一个Region都关联一个Key值范围，即一个使用StartKey和EndKey描述的区间。
  - 事实上，每一个Region仅仅记录StartKey就可以了，因为它的EndKey就是下一个Region的StartKey。
- Region是HBase分布式存储的最基本单元。

Region (2)

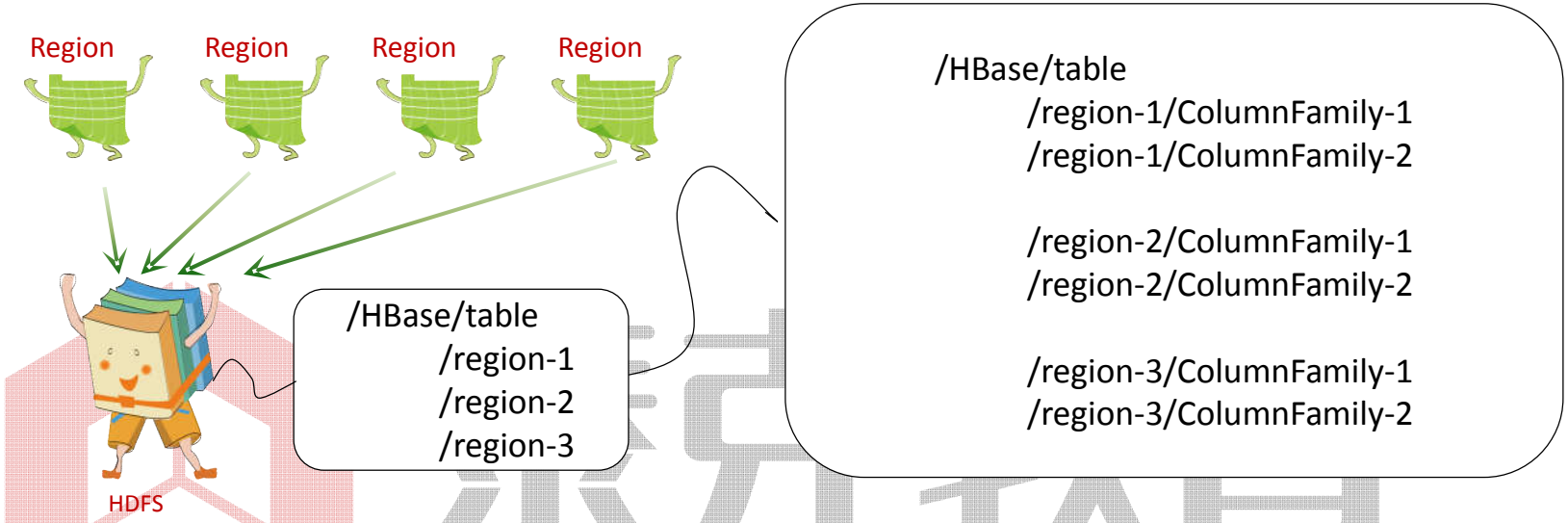


Region (3)



- Region分为元数据Region以及用户Region两类。
- Meta Region记录了每一个User Region的路由信息。
- 读写Region数据的路由，包括如下几步：
  - 找寻Meta Region地址。
  - 再由Meta Region找寻User Region地址。

Column Family



- ColumnFamily是Region的一个物理存储单元。同一个Region下面的多个ColumnFamily，位于不同的路径下面。
- ColumnFamily信息是表级别的配置。也就是说，同一个表的多个Region，都拥有相同的ColumnFamily信息（例如，都有两个ColumnFamily，且不同Region的同一个ColumnFamily配置信息相同）。

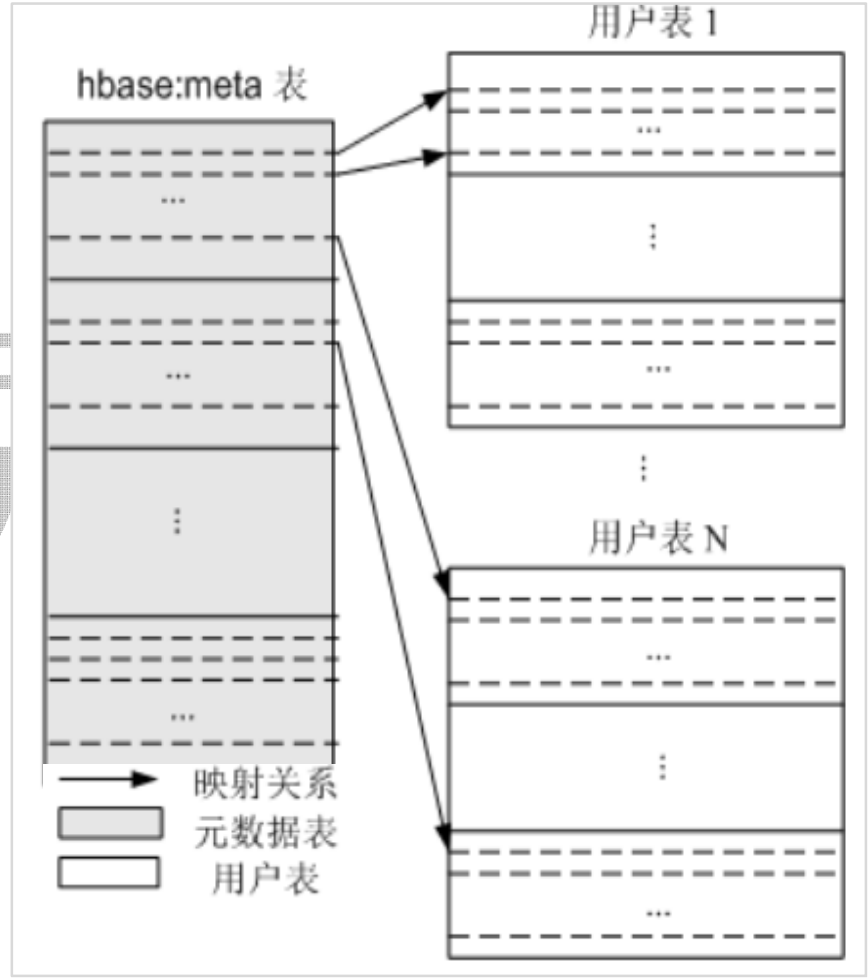
ZooKeeper

ZooKeeper为HBase 提供:

- 分布式锁的服务
  - 多个HMaster进程都尝试着去ZooKeeper中写入一个对应的节点，该节点只能被一个HMaster进程创建成功，创建成功的HMaster进程就是Active。
- 事件监听机制
  - 主HMaster进程宕掉之后，备HMaster在监听对应的ZooKeeper节点。主HMaster进程宕掉之后，该节点会被删除，其它的备HMaster就可以收到相应的消息。
- 微型数据库角色
  - ZooKeeper中存放了Region Server的地址，此时，可以将它理解成一个微型数据库。

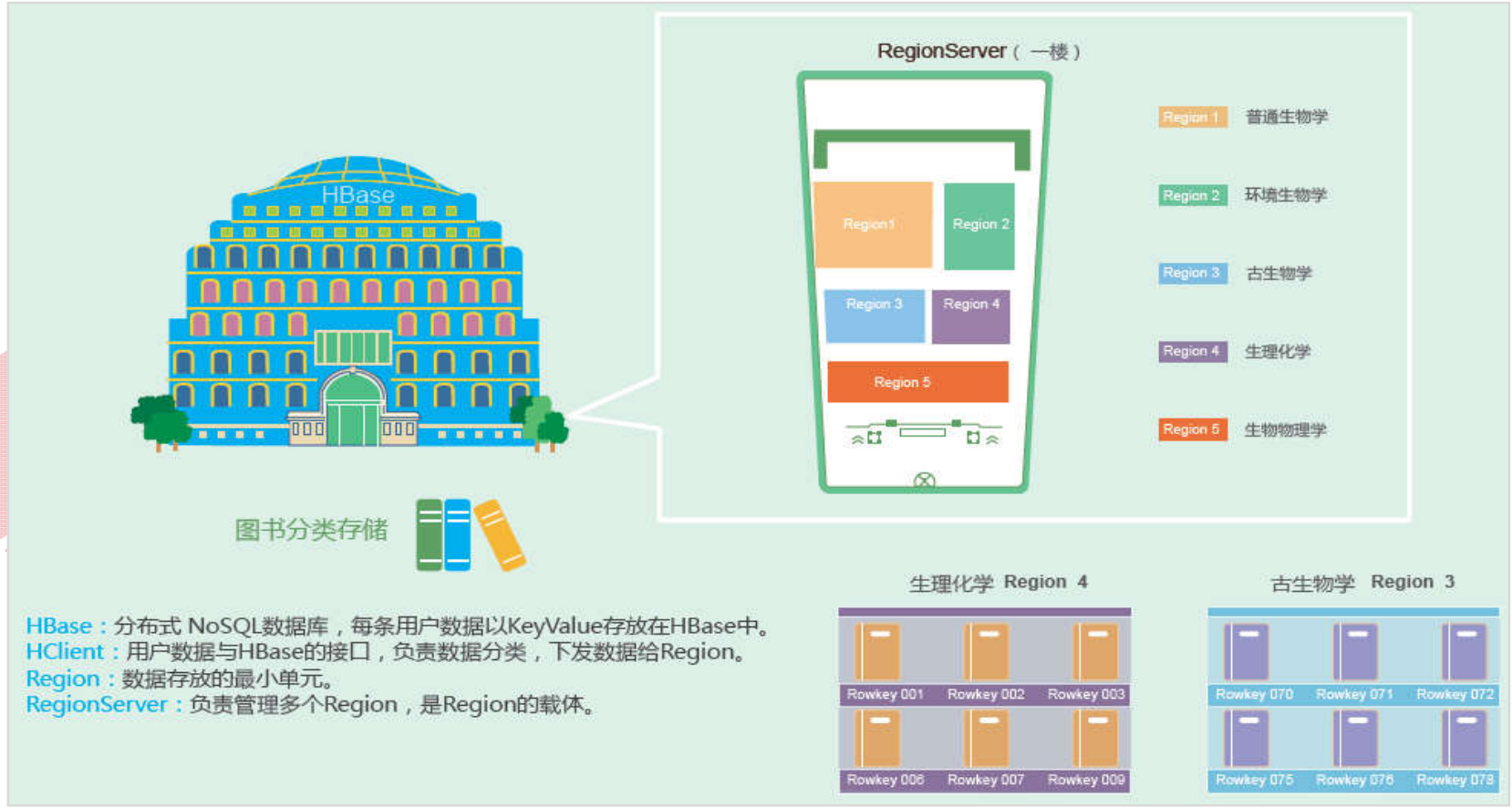
元数据表

- 元数据表HBase:Meta记录用户Region的信息，用来帮助Client定位到具体的Region。
- 元数据表也会被切分为多个Region，Region的元数据信息保存在ZooKeeper中。



目录

写流程

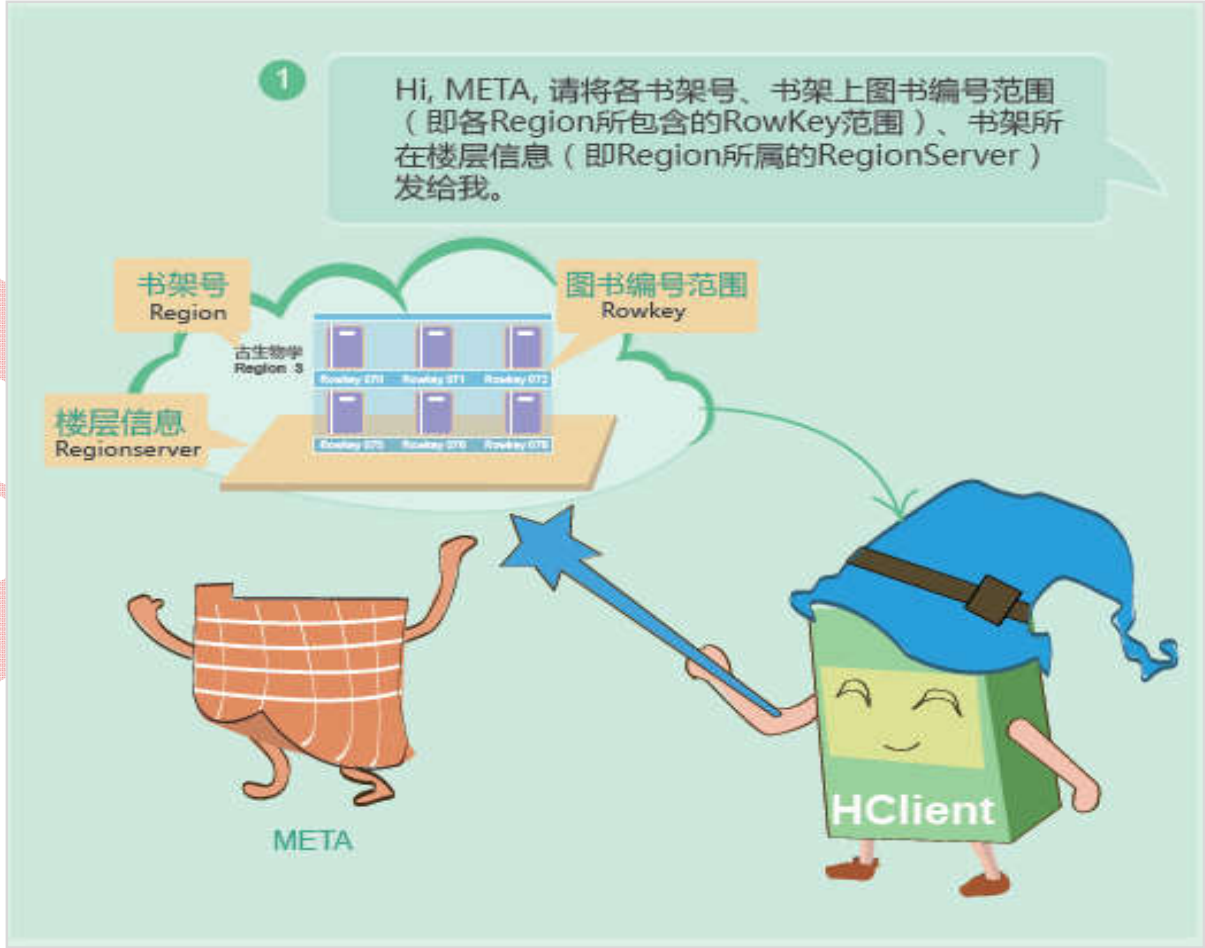


HUAWEI

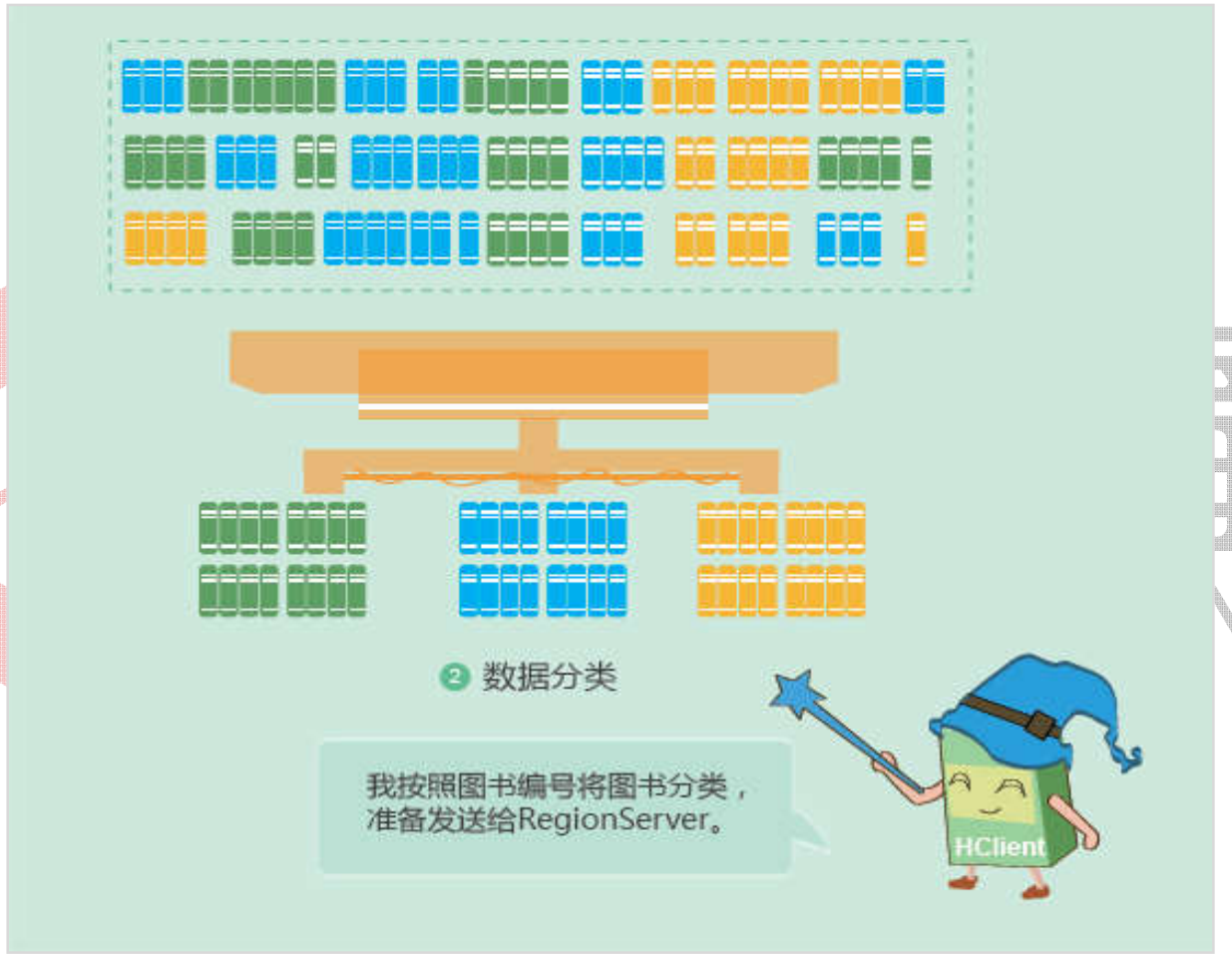
客户端发起写数据请求

- 客户端发起写请求，相当于图书供应商需要把图书发往到图书馆，但是这时候需要定位到哪些图书该发往到哪栋楼哪一层，也就是下一页描述的定位Region。

写流程 - 定位Region

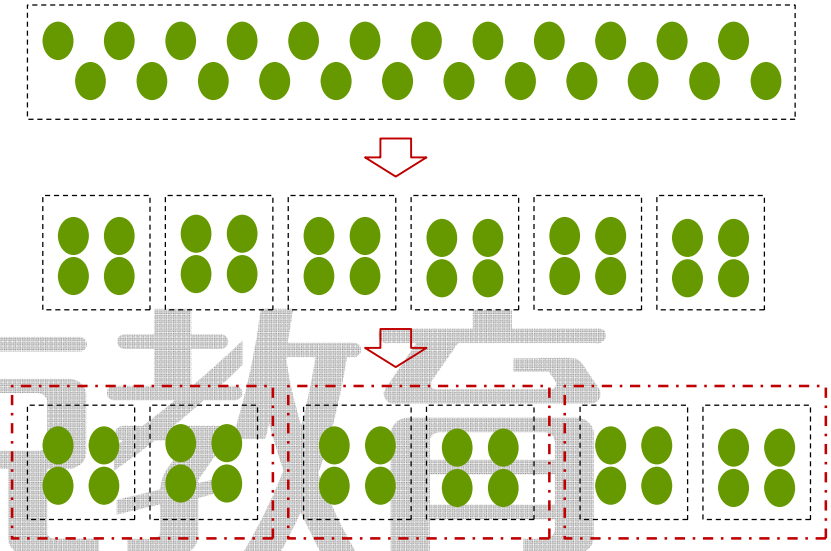


写流程 – 数据分组（1）



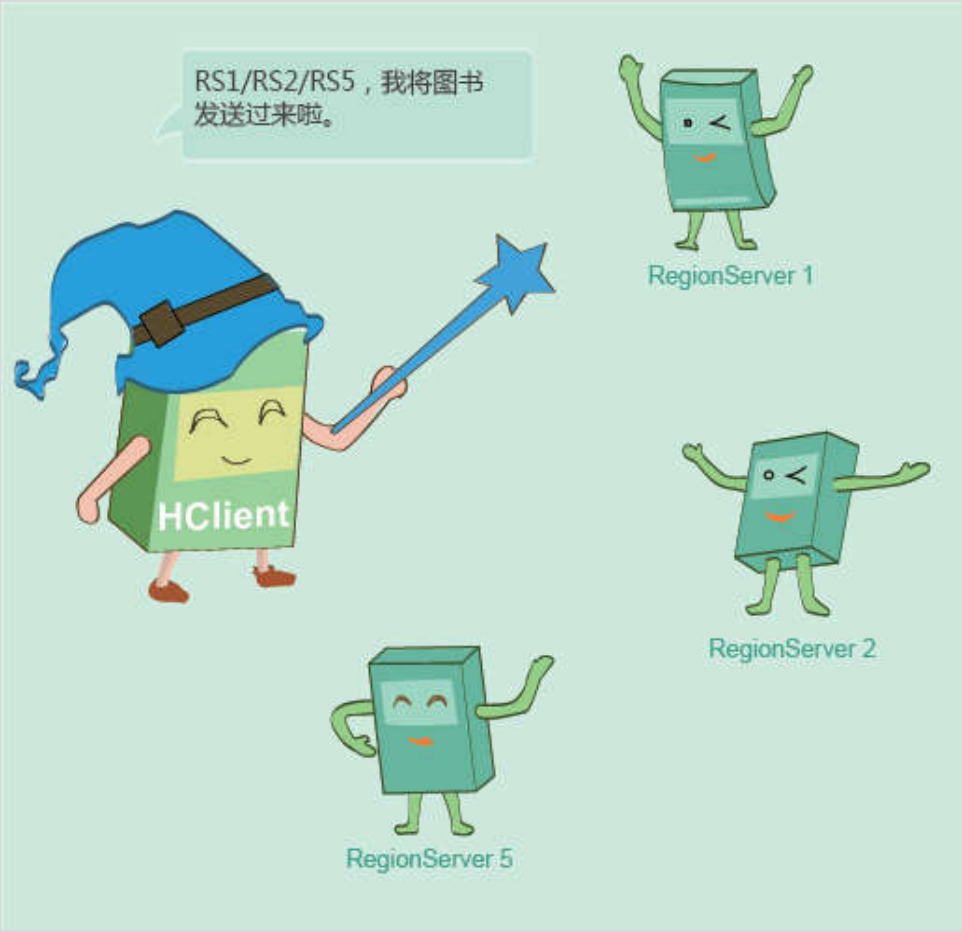
写流程 – 数据分组（2）

- 整个数据分组，涉及到两步
- “分篮子”操作：
  - 根据meta表找到表的region信息，此时也得到了对应的regionserver信息。
  - 根据rowkey，将数据到指定的region中。



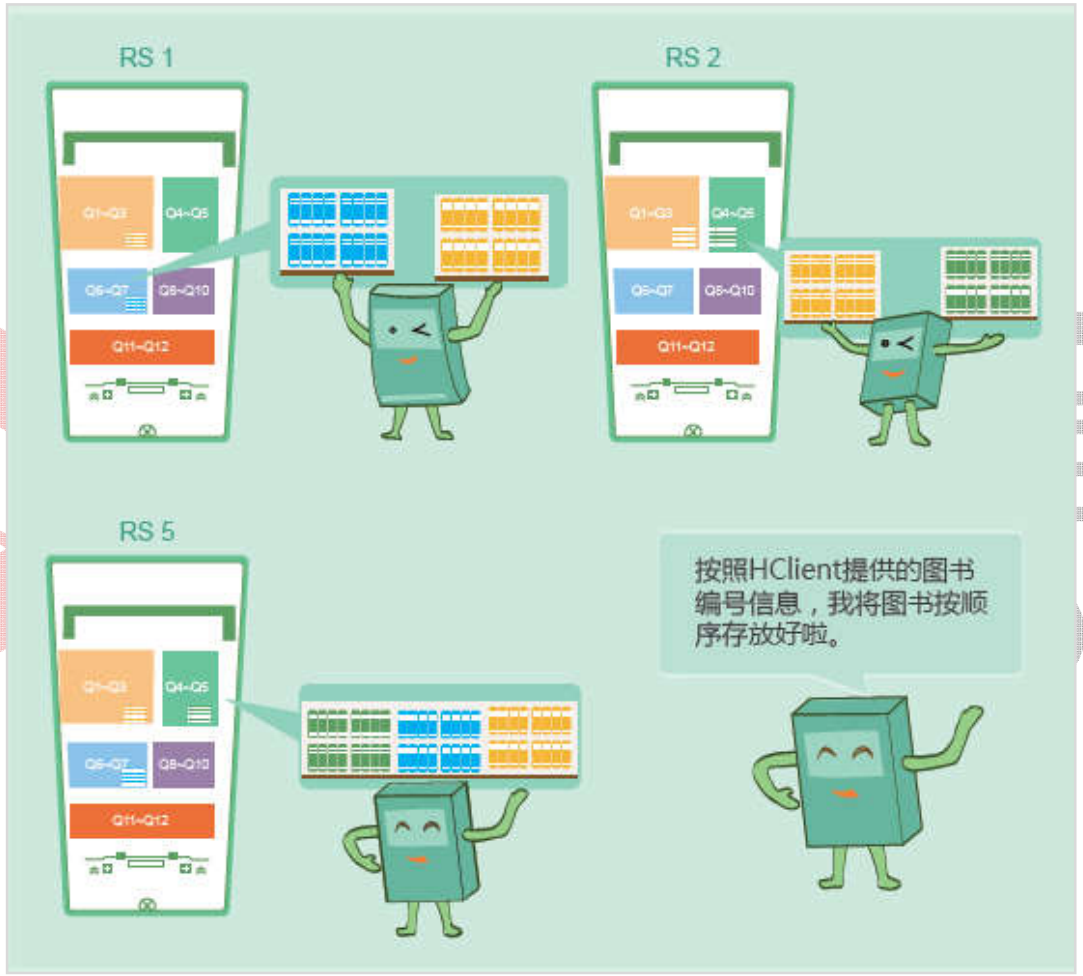
- 每个RegionServer上的数据会一起发送。发送数据中，都是已经按照Region分好组了。

写流程 – 往RegionServer发送请求

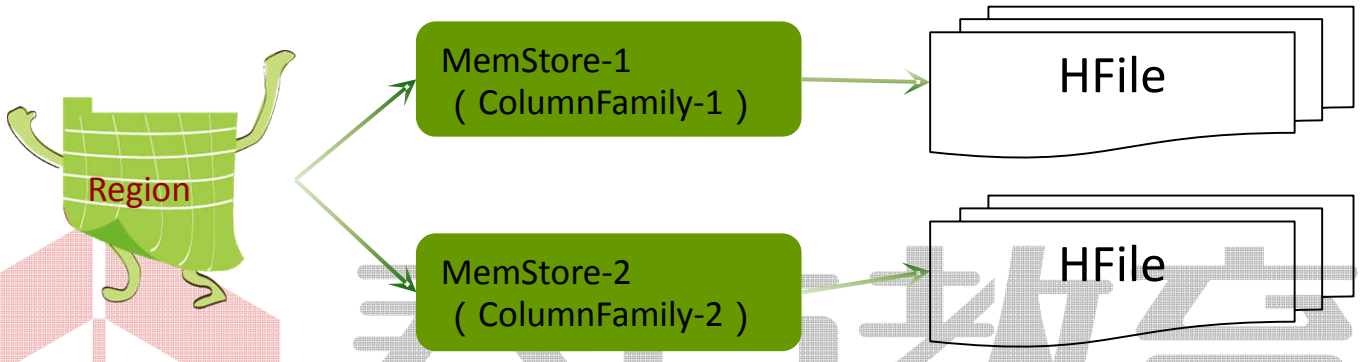


- 利用HBase自身封装的RPC框架，来完成数据发送操作。
- 往多个RegionServer发送请求是并行操作。
- 客户端发送完写数据请求后，会自动等待请求处理结果。
- 如果客户端没有捕获到任何的异常，则认为所有数据都已经被写入成功。如果全部写入失败，或者部分写入失败，客户端能够获知详细的失败Key值列表。

写流程 - Region写数据流程

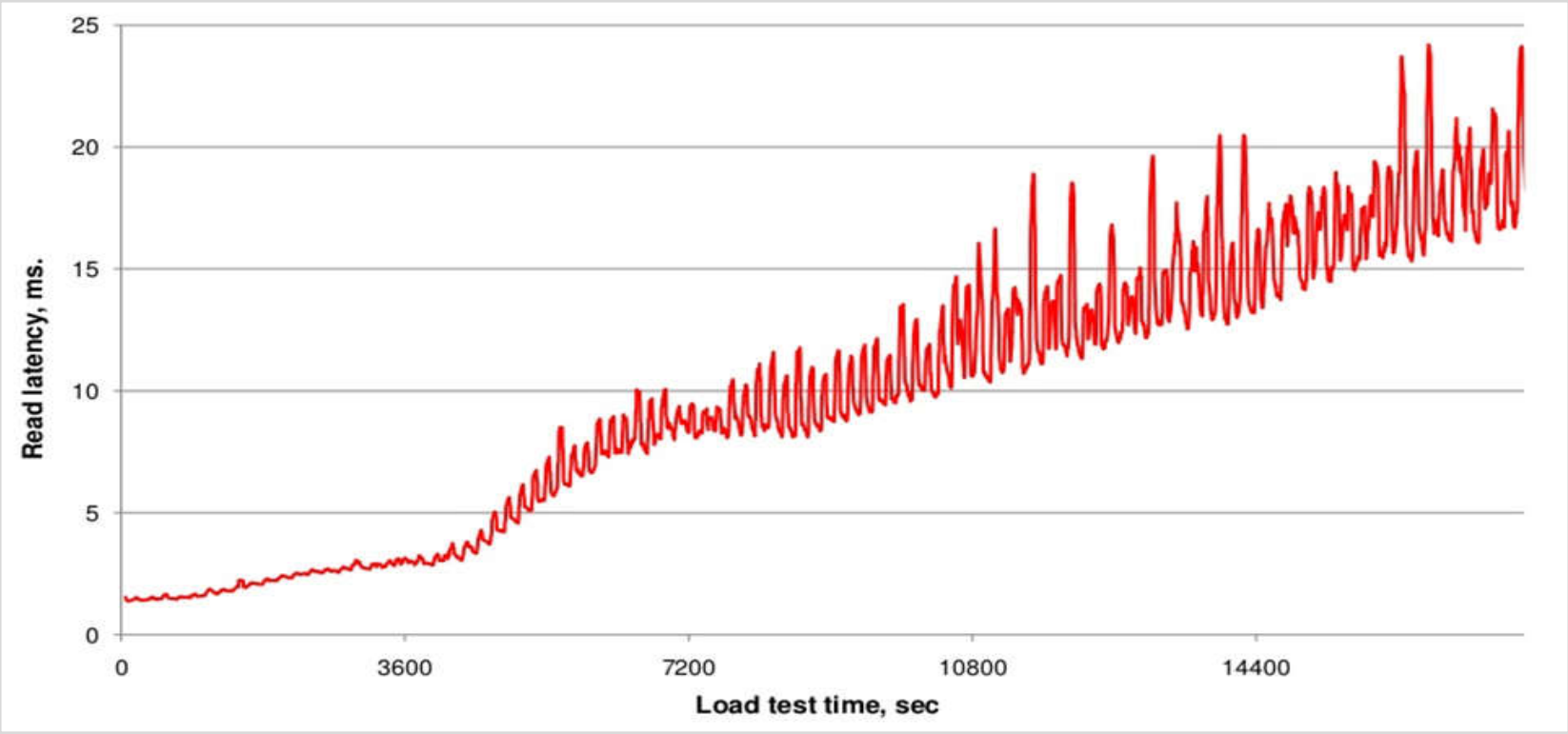


写流程 - Flush



- 以下多个种场景，会触发Memstore的Flush操作：
  - Region中MemStore的总大小，达到了预设的Flush Size阈值。
  - MemStore占用内存的总量和RegionServer总内存比值超出了预设的阈值大小。
  - 当WALs中文件数量达到阈值时。
  - HBase定期刷新Memstore，默认周期为1小时。
  - 用户可以通过shell命令分别对一个表或者一个Region进行flush。

多HFile的影响



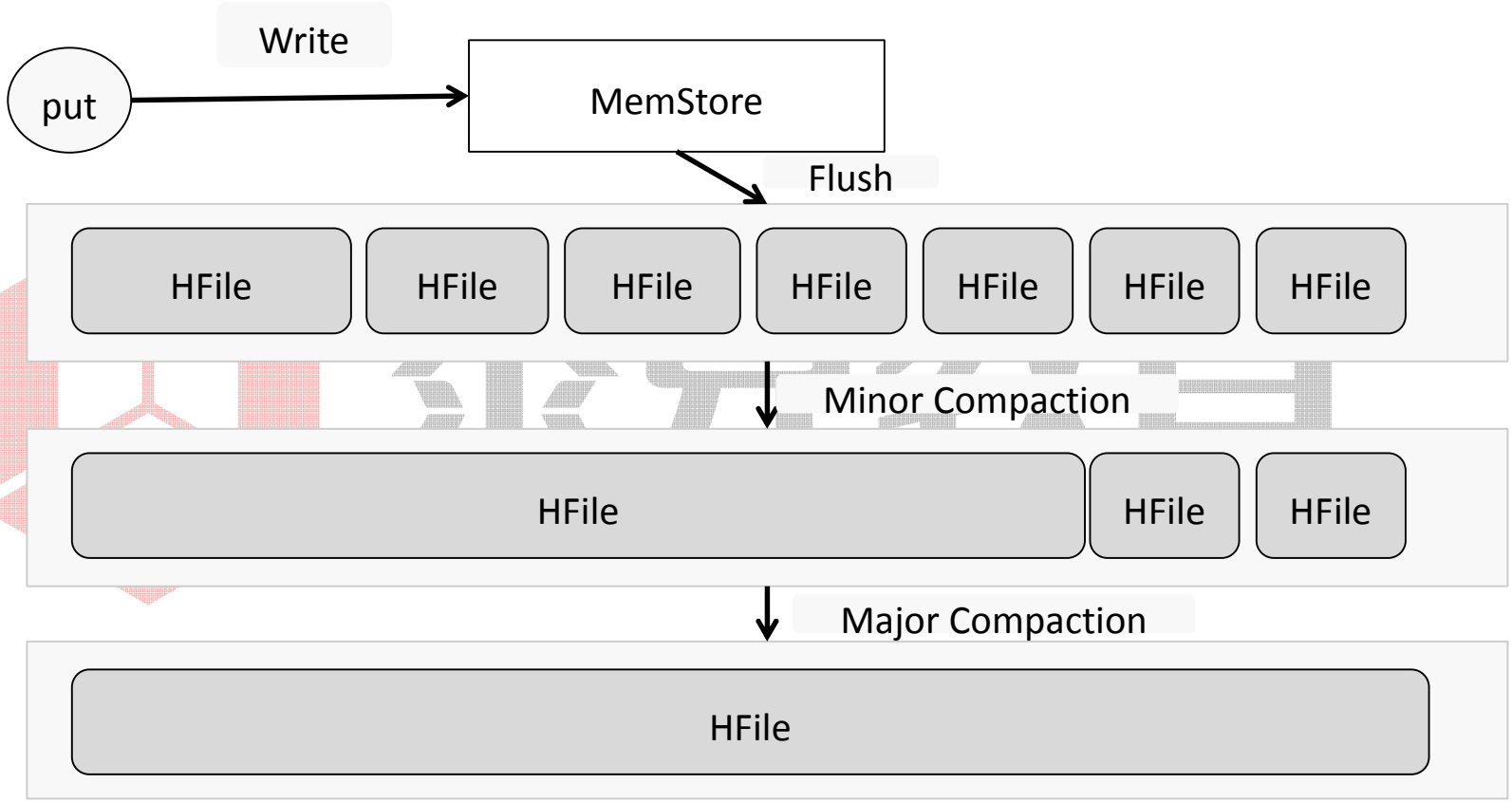
- HFile文件数目越来越多，读取时延也越来越大。

Compaction（1）

- Compaction的目的，是为了减少同一个Region中同一个ColumnFamily下面的小文件（HFile）数目，从而提升读取的性能。
- Compaction分为Minor、Major两类：
  - Minor:小范围的Compaction。有最少和最大文件数目限制。通常会选择一些连续时间范围的小文件进行合并。
  - Major:涉及该Region该ColumnFamily下面的所有的HFile文件。

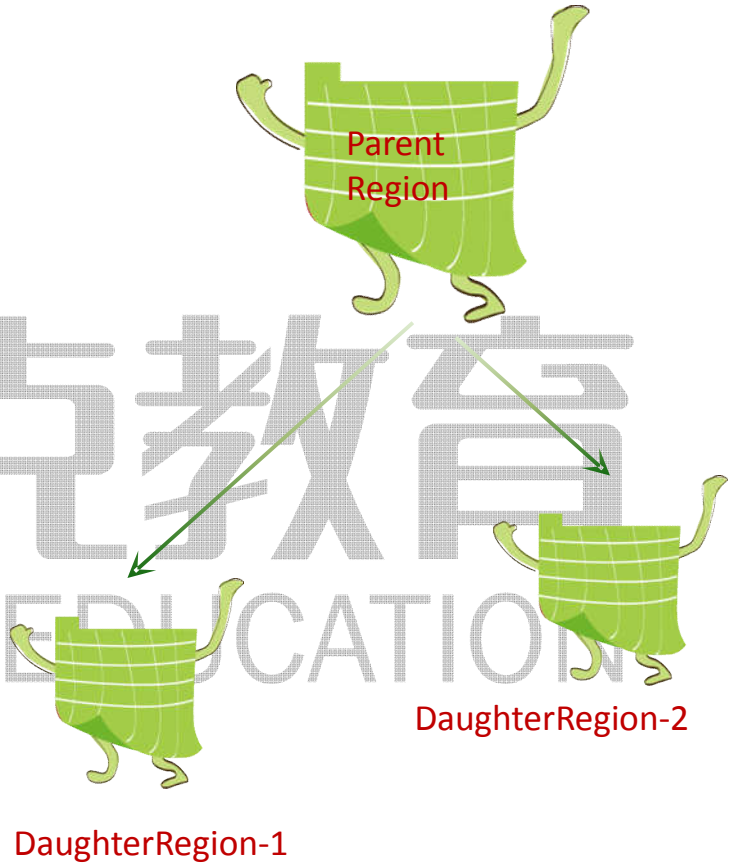
Minor Compaction选取文件时，遵循一定的算法。

Compaction (2)

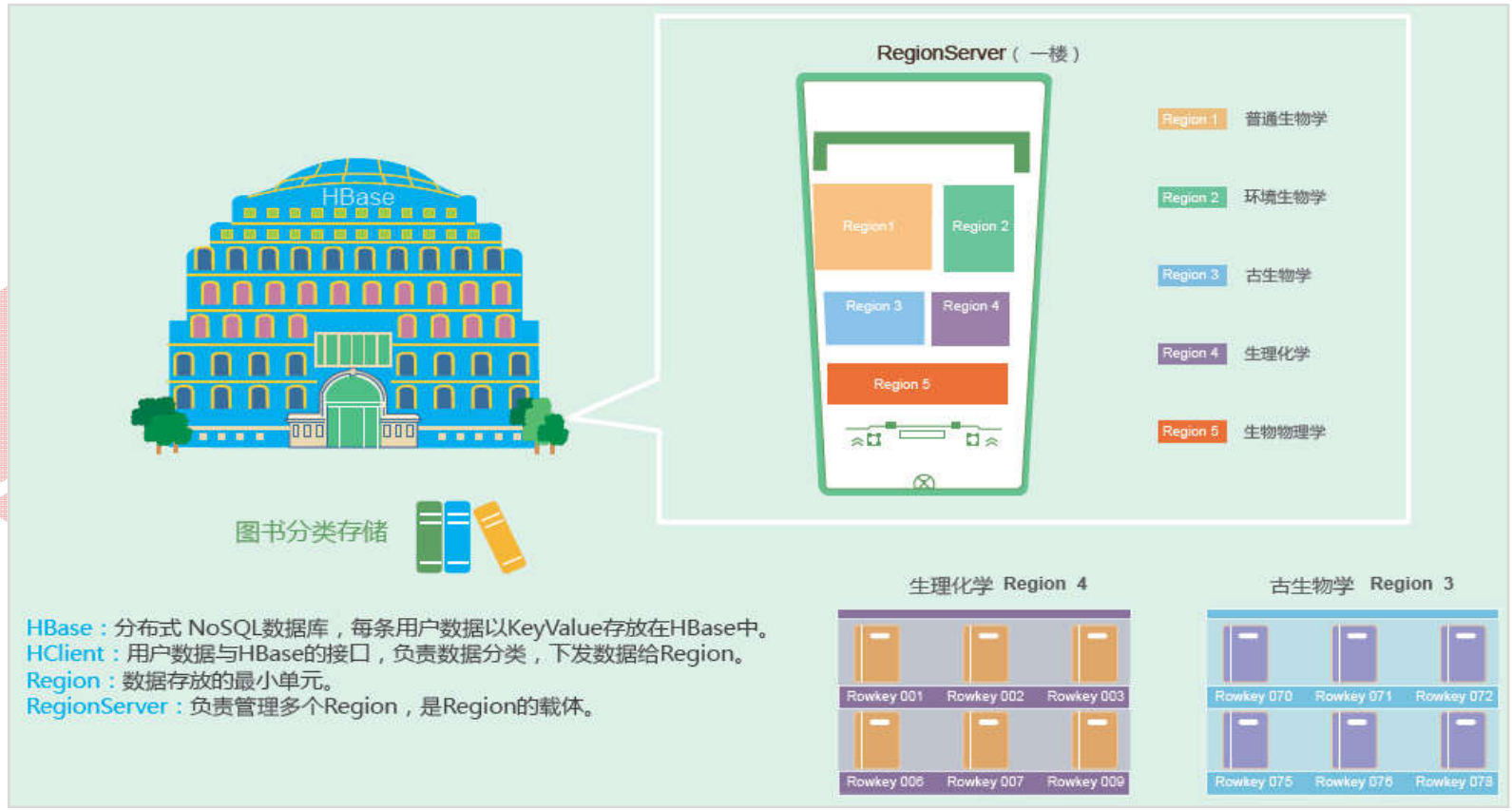


Region Split

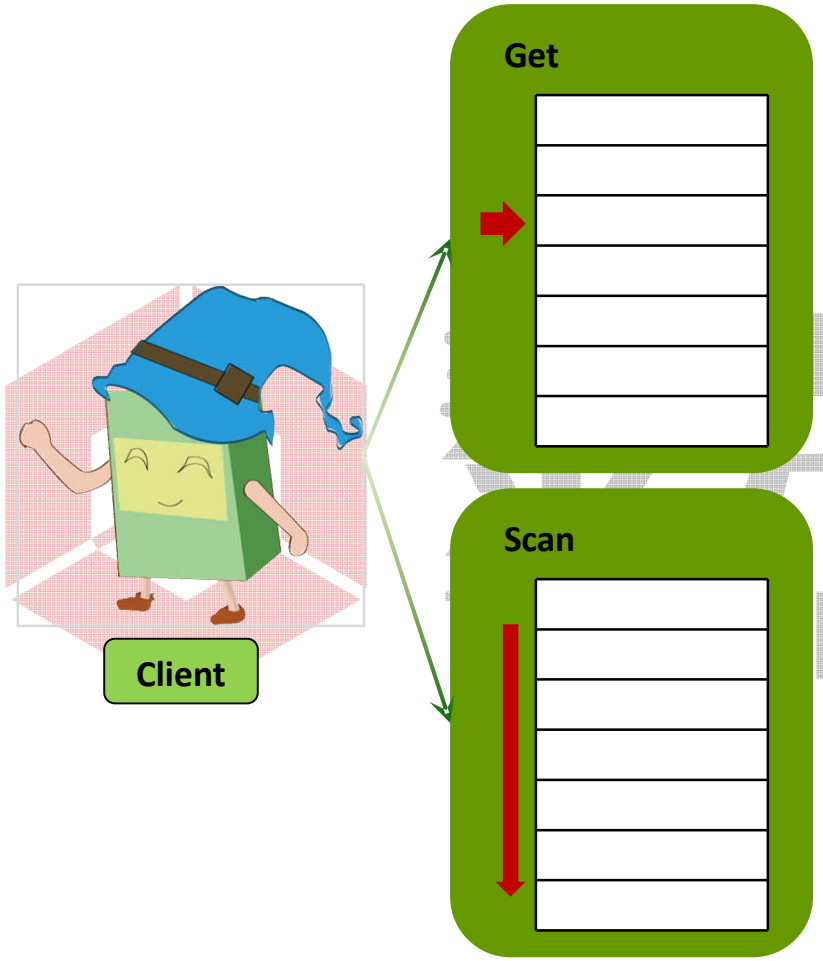
- Region Split是指集群运行期间，某一个Region的大小超出了预设的阈值，则需要将该Region自动分裂成为两个子Region。
- **分裂过程中，被分裂的Region会暂停读写服务。**由于分裂过程中，父Region的数据文件并不会真正的分裂，而是仅仅通过在新Region中创建引用文件的方式，来实现快速的分裂。因此，Region暂停服务的时间会比较短暂。
- 客户端侧所缓存的父Region的路由信息需要被更新。



读流程

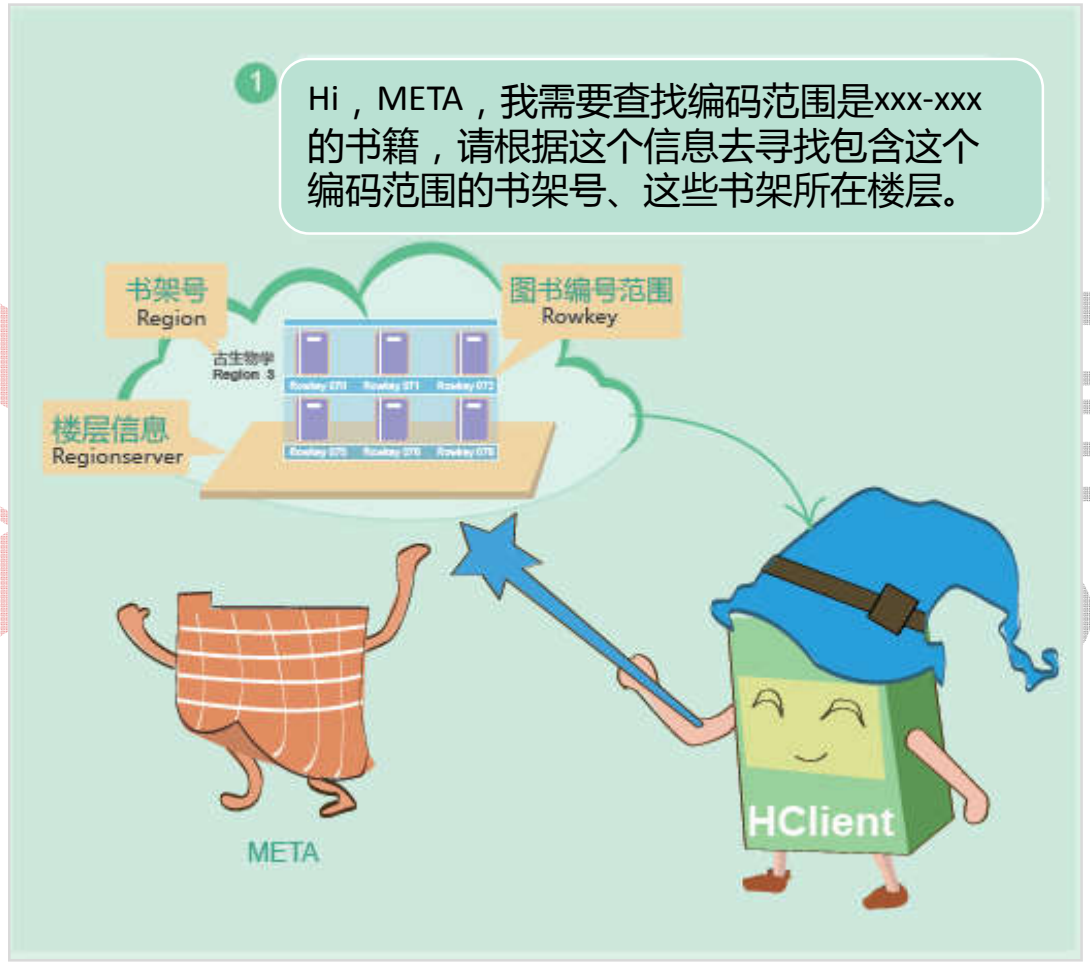


客户端发起读数据请求

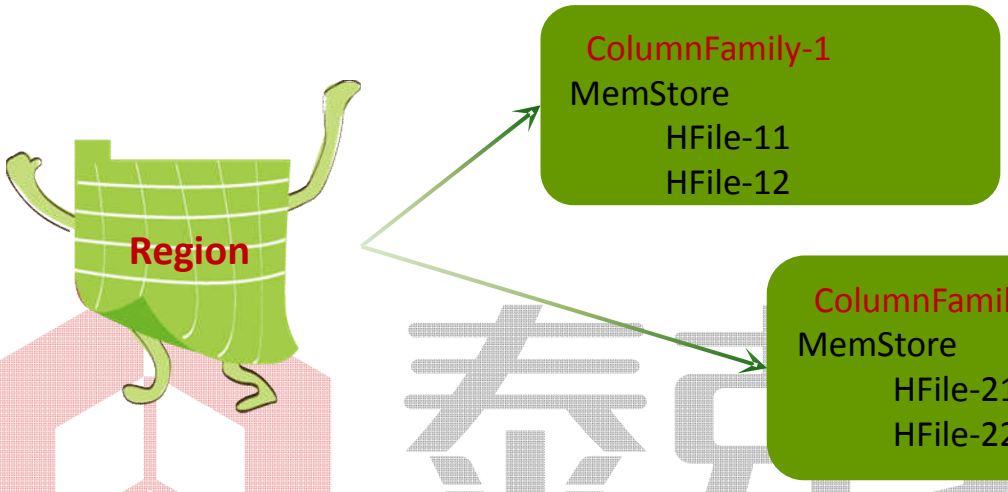


- Get操作在提供精确的Key值的情形下，读取单行用户数据。
- Scan操作是为了批量扫描限定Key值范围内的用户数据。

定位Region



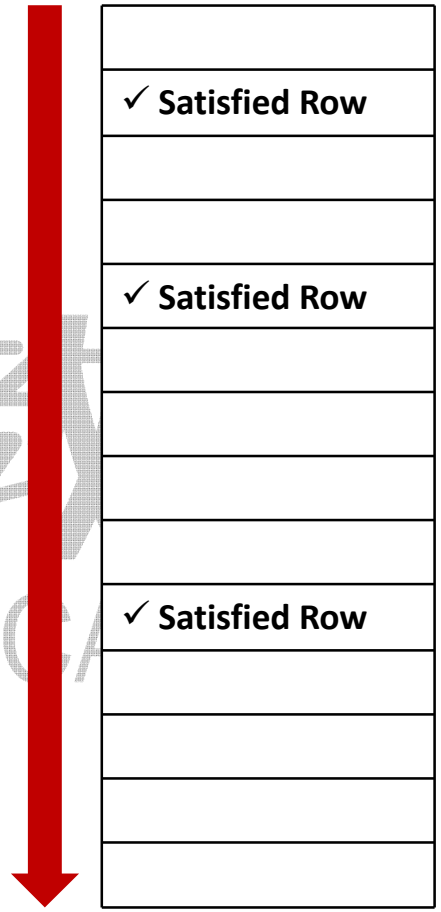
OpenScanner



- OpenScanner的过程中，会创建两种不同的Scanner来读取Hfile、MemStore的数据：
  - HFile对应的Scanner为StoreFileScanner。
  - MemStore对应的Scanner为MemStoreScanner。

Filter

- Filter允许在Scan过程中，设定一定的过滤条件。符合条件的用户数据才返回。
- 当前包含的一些典型的Filter有：
  - RowFilter
  - SingleColumnValueFilter
  - KeyOnlyFilter
  - FilterList



BloomFilter

- BloomFilter用来优化一些随机读取的场景，即Get场景。 它可以被用来快速的判断一条用户数据在一个大的数据集合（该数据集合的大部分数据都没法被加载到内存中）中是否存在。
- BloomFilter在判断一个数据是否存在时，拥有一定的误判率。但对于“用户数据 XXXX不存在” 的判断结果是可信的。
- HBase的BloomFilter的相关数据，被保存在HFile中。

目录

支持二级索引

- 二级索引为HBase提供了按照某些列的值进行索引的能力。

	Column Family A			Column Family B	
RowKey	A:Name	A:Addr.	A:Age	B:Mobile	B:Email
01	张三	北京	23	6875349	……
02	李二	杭州	43	6831475	……
03	王五	深圳	35	6809568	……
04	……	武汉	28	6812645	……
05	……	长沙	26	6889763	……
06	……	济南	35	6854912	……

没有二级索引时，查找手机号“68XXX的记录，必须按照RowKey做全表扫描，逐行匹配” “Mobile”字段，时延很大。

有二级索引时，先查索引表，再定位到数据表中的位置，不用全表扫描，时延小。

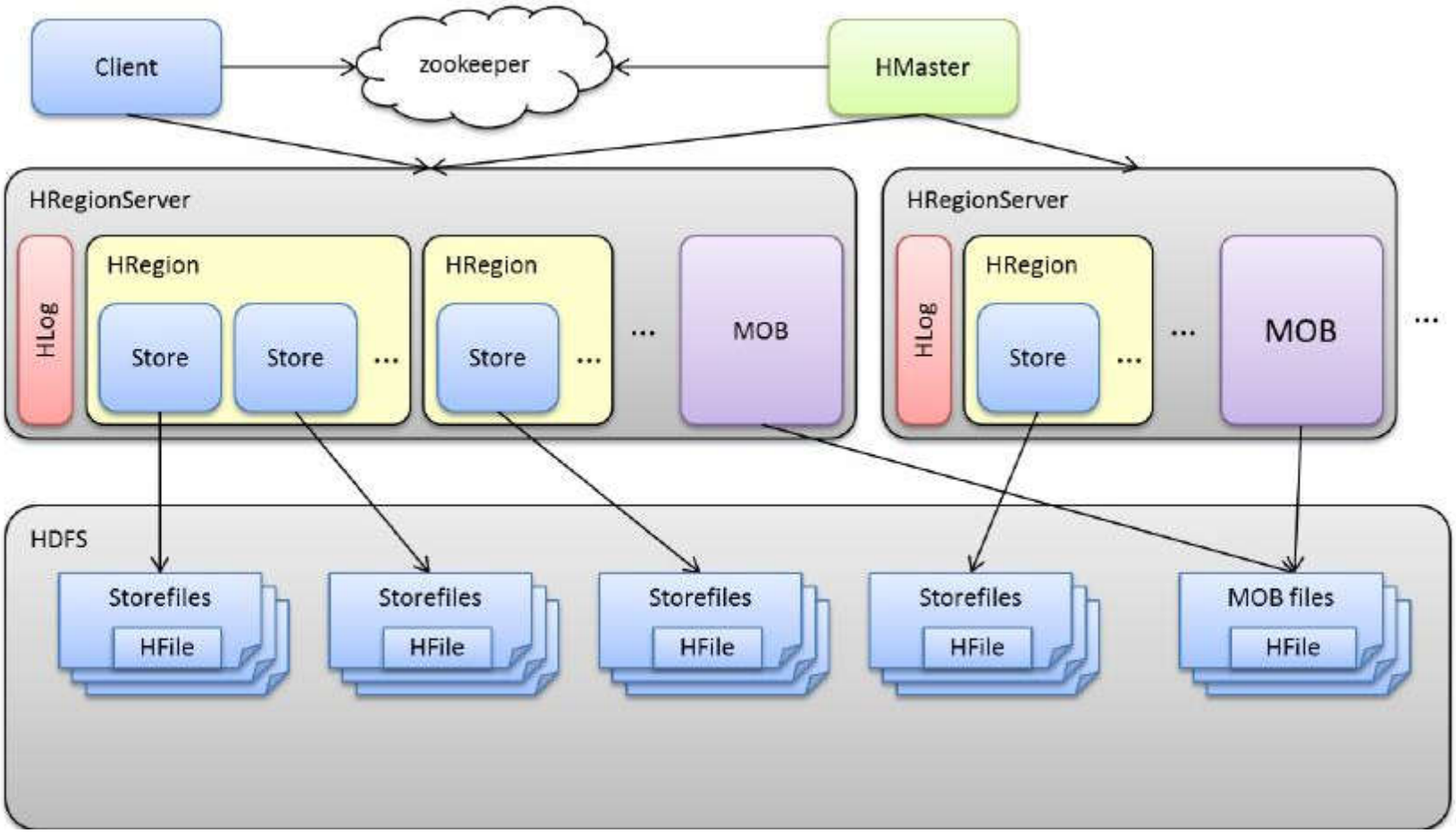
HFS

- HBase文件存储模块（HBase FileStream，简称HFS）是HBase的独立模块，它作为对HBase与HDFS接口的封装，应用在FusionInsight HD的上层应用，为上层应用提供文件的存储、读取、删除等功能。
- HFS的出现解决了需要在HDFS中存储海量小文件，同时也要存储一些大文件的混合的场景。简单来说，就是在HBase表中，需要存放大量的小文件（10MB以下），同时又需要存放一些比较大的文件（10MB以上）。

HBase MOB (1)

- MOB数据（即100KB到10MB大小的数据）直接以HFile的格式存储在文件系统上（例如HDFS文件系统），然后把这个文件的地址信息及大小信息作为value存储在普通HBase的store上，通过工具集中管理这些文件。这样就可以大大降低HBase的compation和split频率，提升性能。

HBase MOB (2)

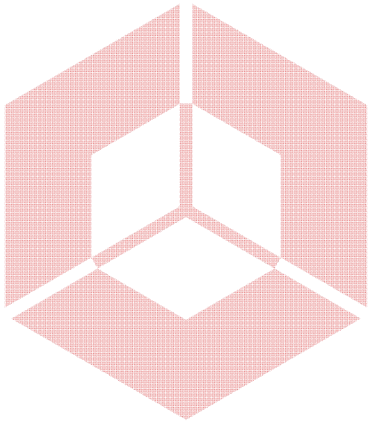


本章总结

- 本章描述了HBase的底层KeyValue存储模型、HBase的基本架构、HBase读写流程及FusionInsight HBase的增强特性。

思考题

1. HBase的Region在split时可以提供服务吗？
2. HBase的Region split有何好处？



思考题

1. Compaction的目的是什么？(　　)

   A. 减少同一个Region同一Column Family下的文件数目

   B. 提升数据读取性能

   C. 提示数据写入能力

   D. 增强HBase并发访性能

思考题

2. HBase的最小存储单元是什么？(　　)

A. Region

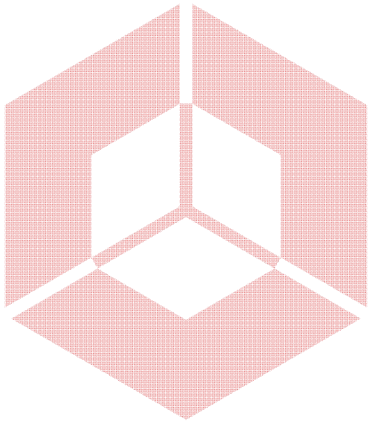
B. Column Family

C. Column

D. Cell

更多信息

- 下载培训资料：
  - http://support.huawei.com/learning/trainFaceDetailAction?lang=zh&pbiPath=term1000025185&courseId=Node1000009072
- eLearning课程：
  - http://support.huawei.com/learning/nodeQueryAction!loadTrainProjectInfo?lang=zh&pbiPath=term1000025185&courseId=Node1000009421&navId=MW000001
- 考试大纲：
  - http://support.huawei.com/learning/Certificate!toExamOutlineDetail?lang=zh&nodeId=Node1000003516
- 模拟考试：
  - http://support.huawei.com/learning/Certificate!toSimExamDetail?lang=zh&nodeId=Node1000004285
- 认证流程：
  - http://support.huawei.com/learning/NavigationAction!createNavi#navi[id]=_40



谢谢

www.huawei.com