厦门大学计算机科学系研究生课程

《大数据技术基础》

第7章 HBase

（2013年新版）

林子雨

厦门大学计算机科学系

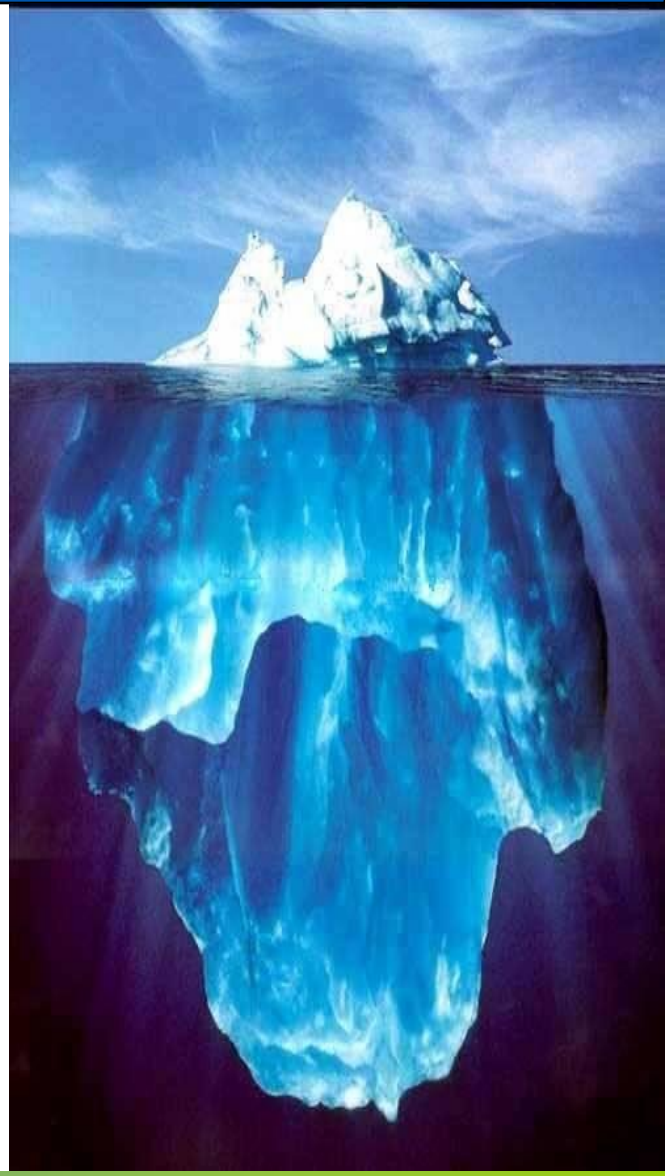
E-mail: ziyulin@xmu.edu.cn

主页：http://www.cs.xmu.edu.cn/linziyu

课程提要

- Hbase简介
- HBase使用场景和成功案例
- HBase和传统关系数据库的对比分析
- HBase访问接口
- HBase数据模型
- HBase的实现
- HBase系统架构
- HBase存储格式
- 读写数据
- MapReduce on HBase

本讲义PPT存在配套教材，由林子雨通过大量阅读、收集、整理各种资料后编写而成，下载配套教材请访问《大数据技术基础》2013班级网站：http://dblab.xmu.edu.cn/node/423



Hbase简介

- HBase（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，模仿并提供了基于Google文件系统的BigTable数据库的所有功能。两者间的对比如下：

	文件存储系统	数据处理方式	协同服务
BigTable	GFS	MapReduce	Chubby
Hbase	HDFS	MapReduce	Zookeeper

- HBase可以直接使用本地文件系统或者Hadoop作为数据存储方式，不过为了提高数据可靠性和系统的健壮性，发挥HBase处理大数据量等功能，需要使用Hadoop作为文件系统。与Hadoop一样，HBase主要依靠横向扩展，通过不断增加廉价的商用服务器，来增加计算和存储能力。

Hbase简介

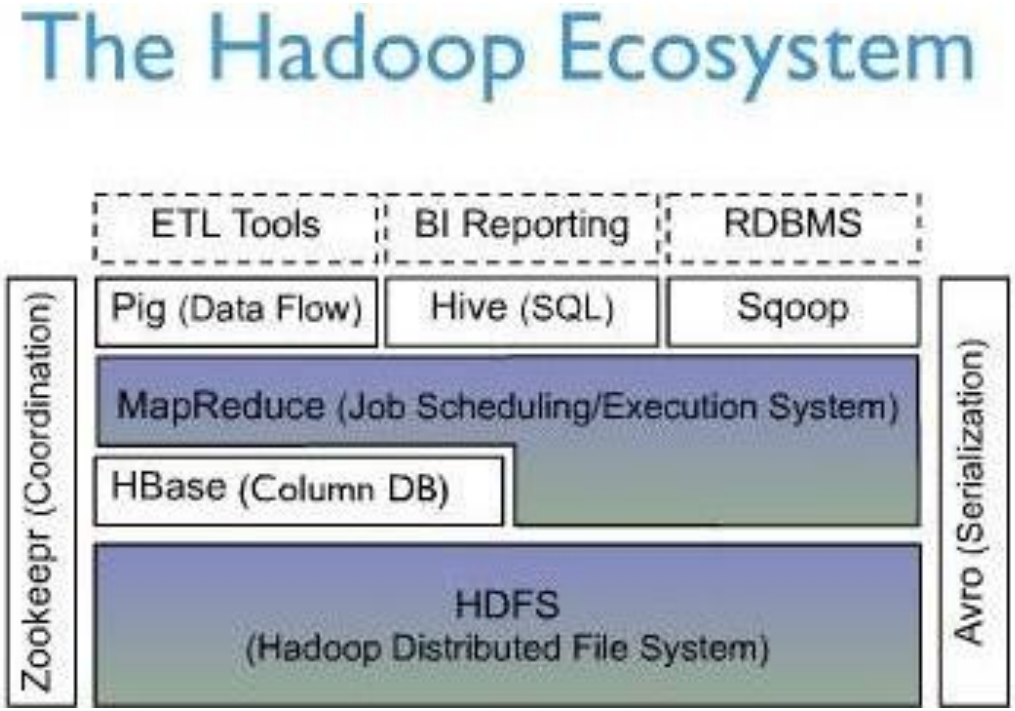
- HBase仅能通过主键(row key)和主键的range来检索数据，仅支持单行事务(可通过Hive支持来实现多表连接等复杂操作)。主要用来存储非结构化和半结构化的松散数据。
- HBase的目标是处理非常庞大的表，可以用普通的计算机处理超过10亿行数据，并且有数百万列元素组成的数据表。

HBase中的表一般有这样的特点：

- **大**：一个表可以有上亿行，上百万列；
- **面向列**：面向列(族)的存储和权限控制，列(族)独立检索；
- **稀疏**：对于为空(null)的列，并不占用存储空间，因此，表可以设计的非常稀疏。

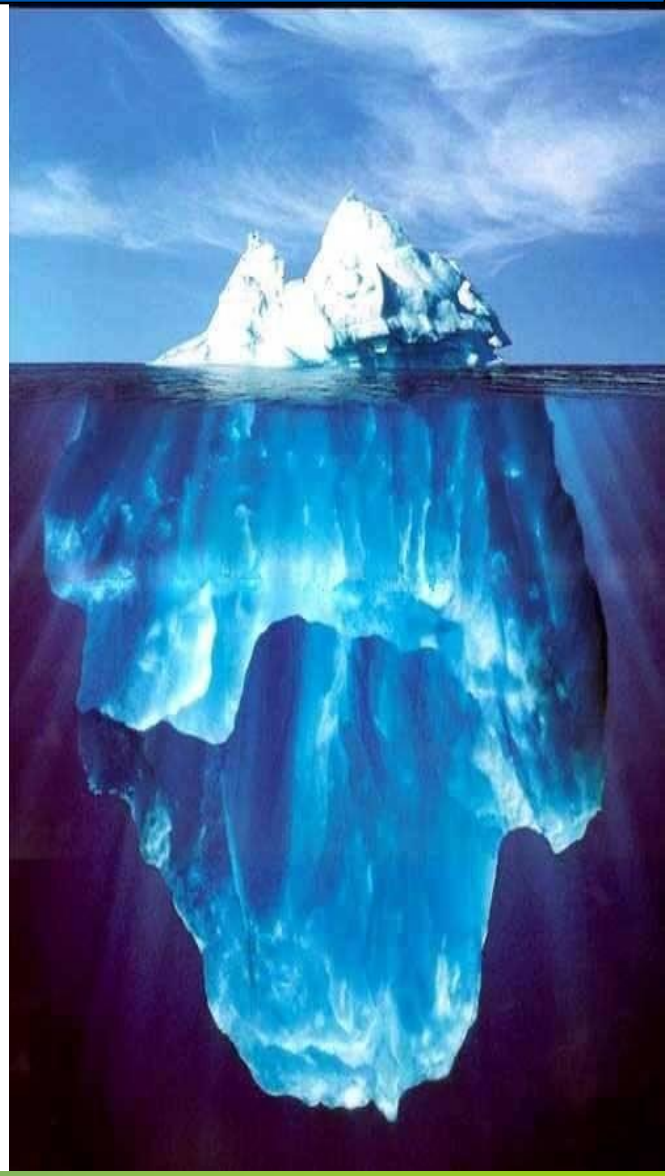
Hbase简介

下图是Hadoop生态系统中的各层系统，其中HBase位于结构化存储层，HDFS为HBase提供了高可靠性的底层存储支持，MapReduce为HBase提供了高性能的计算能力，Zookeeper为HBase提供了稳定服务和失败恢复机制。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，方便数据迁移。



课程提要

- Hbase简介
- HBase使用场景和成功案例
- HBase和传统关系数据库的对比分析
- HBase访问接口
- HBase数据模型
- HBase的实现
- HBase系统架构
- HBase存储格式
- 读写数据
- MapReduce on HBase



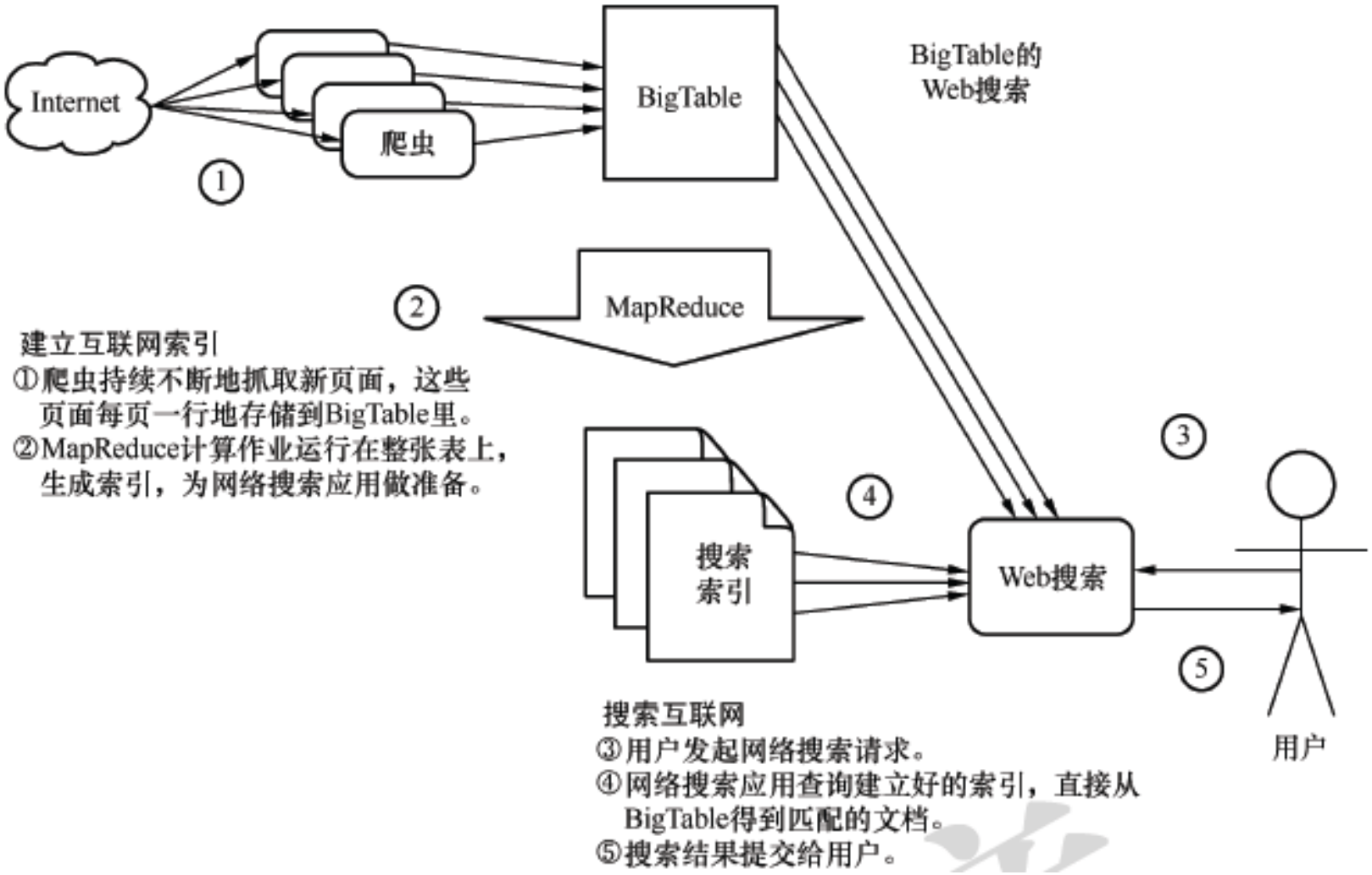
Hbase使用场景和成功案例

HBase被证实是一个强大的工具，尤其是在已经使用Hadoop的场合。如今HBase已经是Apache顶级项目，有着众多的开发人员和兴旺的用户社区。它成为一个核心的基础架构部件，运行在世界上许多公司（如Facebook、Twitter、Adobe等）的大规模生产环境中。

HBase 模仿了Google 的BigTable，而最初BigTable被发明的原因是为了解决互联网搜索问题：存储互联网。

搜索含有特定词语的文档，需要查找索引，该索引提供了特定词语和包含该词语的所有文档的映射。BigTable和模仿出来的HBase，为这种文档库提供存储，BigTable提供行级访问，所以爬虫可以插入和更新单个文档。搜索索引可以基于BigTable通过MapReduce计算高效生成。如果结果是单个文档，可以直接从BigTable取出。支持各种访问模式是影响BigTable设计的关键因素。

Hbase使用场景和成功案例



使用BigTable提供网络搜索结果

Hbase使用场景和成功案例

HBase设计初衷是用来存储互联网持续更新的网页副本，但用在互联网相关的其他方面也是很合适的。例如，HBase在社交网络公司内部和周围各种各样的需求中找到了用武之地。从存储个人之间的通信信息，到通信信息分析，HBase成为了Facebook、Twitter等公司的关键基础设施。

在这个领域，HBase有3种主要使用场景（但不限于这3 种）：

1.抓取增量数据

使用HBase 作为数据存储，抓取来自各种数据源的增量数据，如抓取用户交互数据，以备之后进行分析、处理。

2.内容服务

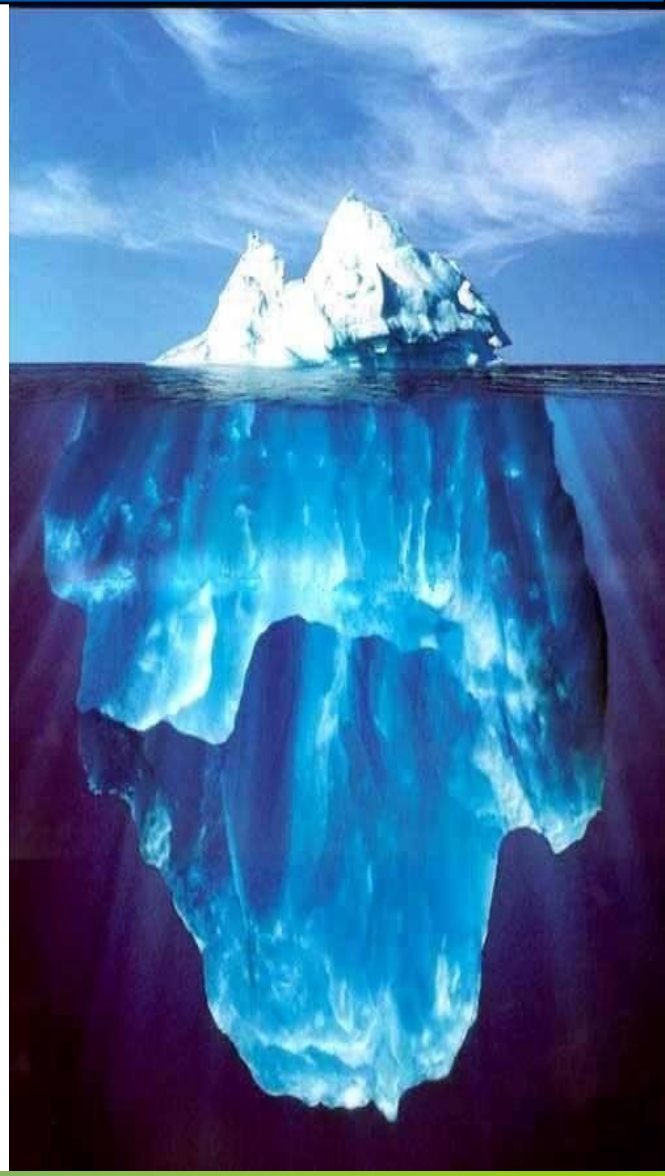
传统数据库最主要的使用场合之一是为用户提供内容服务，如URL短链接服务，可以HBase为基础，存储大量的短链接以及和原始长链接的映射关系。

3.信息交换

Facebook的短信平台每天交换数十亿条短信，HBase可以很好的满足该平台的需求：高的写吞吐量，极大的表，数据中心的强一直性。

课程提要

- HBase简介
- HBase使用场景和成功案例
- HBase和传统关系数据库的对比分析
- HBase访问接口
- HBase数据模型
- HBase的实现
- HBase系统架构
- HBase存储格式
- 读写数据
- MapReduce on HBase



HBase和传统关系数据库的对比分析

HBase与以前的关系数据库存在很大的区别，它是按照BigTable开发的，是一个稀疏的、分布的、持续多维度的排序映射数组。

HBase基于列模式的映射数据库，它只能表示很简单的键-数据的映射关系，它大大简化了传统的关系数据库。两者区别如下：

- **数据类型**：HBase只有简单的字符串类型，所有类型都由用户自己处理，它只保存字符串。而关系数据库有丰富的类型选择和存储方式。
- **数据操作**：HBase操作只有很简单的插入、查询、删除、清空等，表和表之间是分离的，没有复杂的表和表之间的关系，所以也不能也没有必要实现表和表之间的关联等操作。而传统的关系数据通常有各种各样的函数、连接操作。
- **存储模式**：HBase是基于列存储的，每个列族都有几个文件保存，不同列族的文件是分离的。传统的关系数据库是基于表格结构和行模式保存的。
- **数据维护**：HBase的更新正确来说应该不叫更新，而且一个主键或者列对应的新的版本，而它旧有的版本仍然会保留，所以它实际上是插入了新的数据，而不是传统关系数据库里面的替换修改。
- **可伸缩性**：HBase和BigTable这类分布式数据库就是直接为了这个目的开发出来的，能够轻易的增加或者减少（在硬件错误的时候）硬件数量，而且对错误的兼容性比较高。而传统的关系数据库通常需要增加中间层才能实现类似的功能。

HBase和传统关系数据库的对比分析

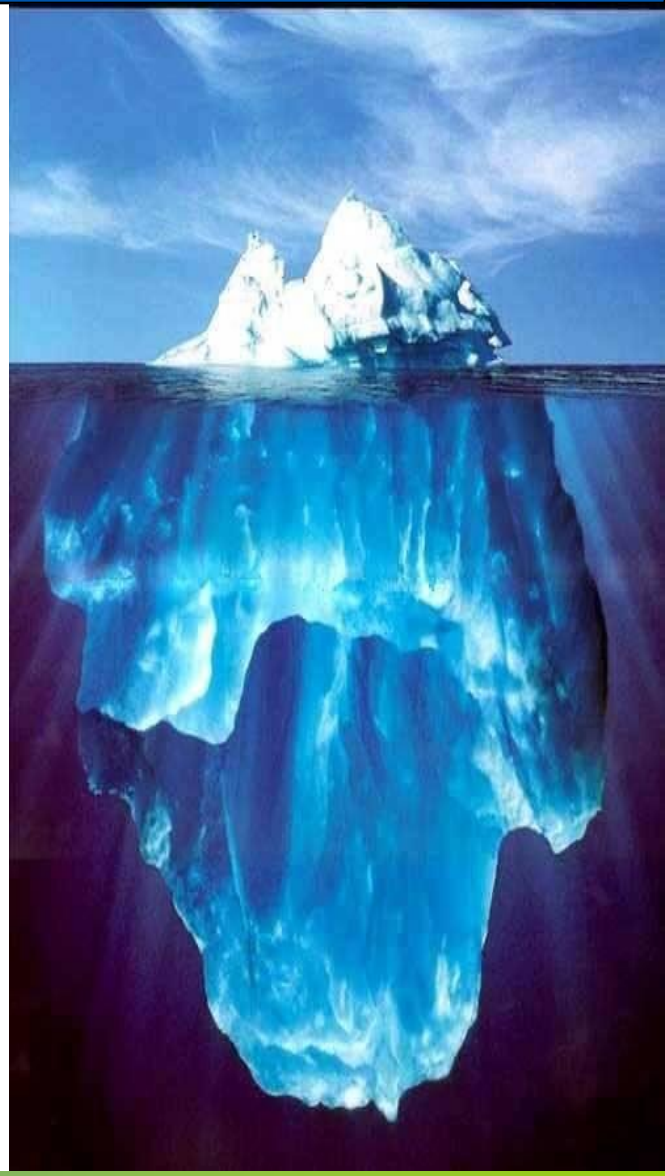
当前的关系数据库基本都是从上世纪70年代发展而来的，它们基本都有以下的体系特点：

- 面向磁盘存储和索引结构；
- 多线程访问；
- 基于锁的同步访问机制；
- 基于log记录的恢复机制。

而BigTable和HBase之类基于列模式的分布式数据库，更适应海量存储和互联网应用的需求，灵活的分布式架构可以使其利用廉价的硬件设备就组建一个大的数据仓库，而互联网应用就是以字符为基础的，BigTable和HBase就针对这些应用而开发出来的数据库。由于其中的时间戳特性，BigTable和HBase与生俱来就特别适合于开发wiki、archiveorg之类的服务，而HBase直接就是作为一个搜索引擎的一部分被开发出来的。

课程提要

- HBase简介
- HBase使用场景和成功案例
- HBase和传统关系数据库的对比分析
- HBase访问接口
- HBase数据模型
- HBase的实现
- HBase系统架构
- HBase存储格式
- 读写数据
- MapReduce on HBase

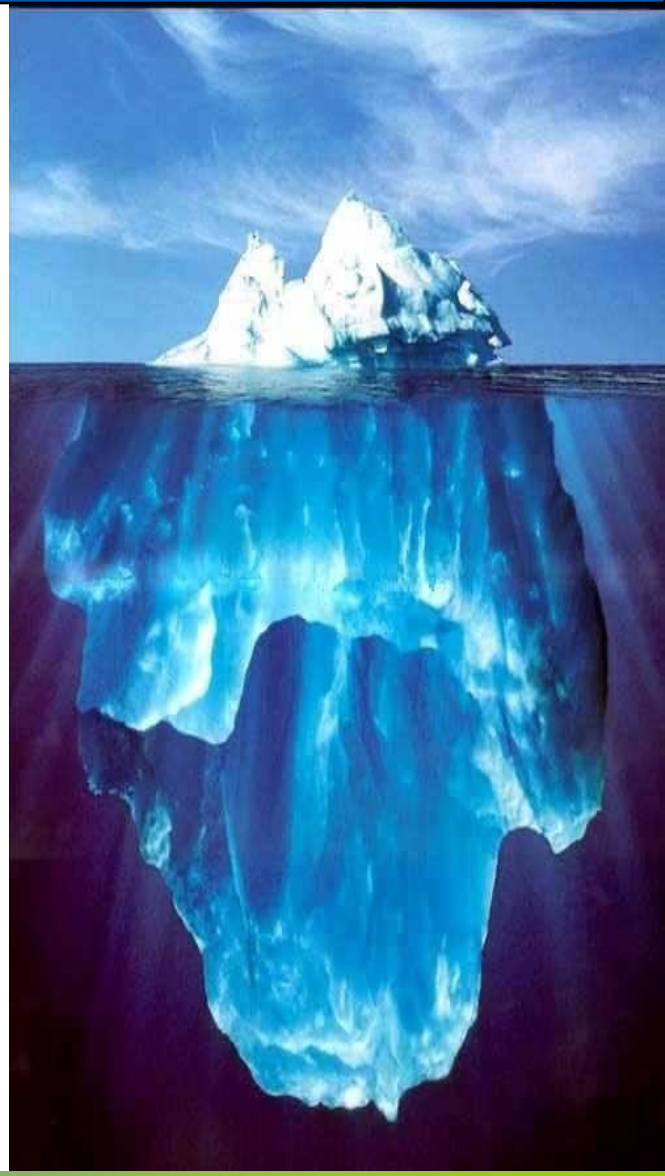


HBase访问接口

- Native Java API：最常规和高效的访问方式，适合Hadoop MapReduce作业并行批处理HBase表数据；
- HBase Shell：HBase的命令行工具，最简单的接口，适合HBase管理使用；
- 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；
- REST Gateway：支持REST风格的Http API访问HBase，解除了语言限制；
- 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；
- Hive：可以使用类似SQL语言来访问HBase。

课程提要

- HBase简介
- HBase使用场景和成功案例
- HBase和传统关系数据库的对比分析
- HBase访问接口
- HBase数据模型
- HBase的实现
- HBase系统架构
- HBase存储格式
- 读写数据
- MapReduce on HBase



HBase数据模型

HBase的索引是行关键字、列关键字和时间戳。每个值是一个不解释的字符数组，数据都是字符串，没有类型。用户在表格中存储数据，每一行都有一个可排序的主键和任意多的列。由于是稀疏存储的，所以，同一张表里面的每一行数据都可以有截然不同的列。

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	

HBase数据模型

- 列名字的格式是"<family>:<label>"，都是由字符串组成，每一张表有一个family集合，这个集合是固定不变的，相当于表的结构，只能通过改变表结构来改变。但是label值相对于每一行来说都是可以改变的。
- HBase把同一个family里面的数据存储在同一个目录底下，而HBase的写操作是锁行的，每一行都是一个原子元素，都可以加锁。
- 所有数据库的更新都有一个时间戳标记，每个更新都是一个新的版本，而HBase会保留一定数量的版本，这个值是可以设定的。客户端可以选择获取距离某个时间最近的版本，或者一次获取所有版本。

HBase数据模型

在HBase数据模型中，包括如下三个重要概念：

- **行键（Row Key）**：HBase表的主键，表中的记录按照行键排序。行键用来检索记录的主键。
- **时间戳（Timestamp）**：每次数据操作对应的时间戳，可看作是数据的版本号，不同的版本是通过时间戳来进行索引的。
- **列族（Column Family）**：表在水平方向有一个或者多个列族组成，一个列族中可以由任意多个列组成，即列族支持动态扩展，无需预先定义列的数量以及类型，所有列均以二进制格式存储，用户需要自行进行类型转换。

概念视图

一个表可以想象成一个大的映射关系，通过主键，或者主键+时间戳，可以定位一行数据（由于是稀疏数据，所以某些列可以是空白）。

Row Key	Time Stamp	Column "contents:"	Column "anchor:"		Column "mime:"
"com.cnn.www"	t9		"anchor:cnnsi.com"	"CNN"	
	t8		"anchor:my.look.ca"	"CNN.com"	
	t6	"<html>..."			"text/html"
	t5	"<html>..."			
	t5	"<html>..."			

上表是一个存储Web网页的范例列表片段。行名是一反向URL。contents列族用来存放网页内容，anchor列族存放引用该网页的锚链接文本。该主页被两个页面引用，因此包含了“anchor:cnnsi.com”和“anchhor:my.look.ca”的列。每个锚链接只有一个版本（由时间戳标识，如t9，t8）；contents列有三个版本，分别由时间戳t3，t5和t6标识。

物理视图

虽然从概念视图来看，HBase中的每个表格是由很多行组成的，但是，在物理存储上面，它是按照列来保存的。上述的概念视图应按如下形式存储。

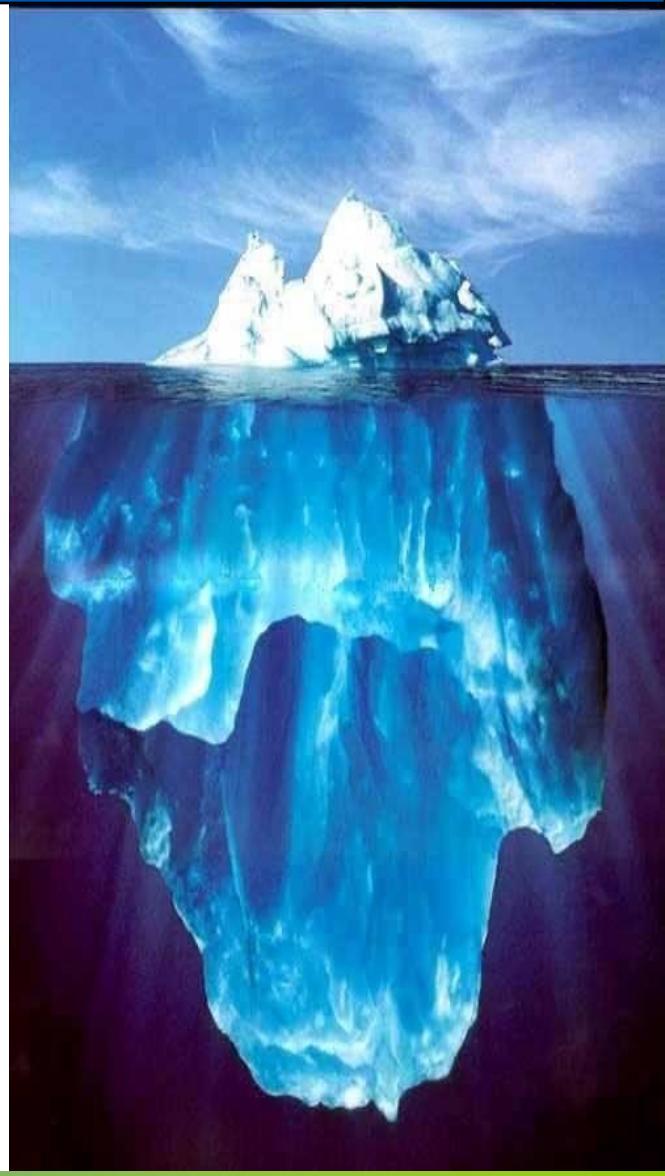
Row Key	Time Stamp	Column "contents:"
"com.cnn.www"	t6	"<html>..."
	t5	"<html>..."
	t3	"<html>..."

Row Key	Time Stamp	Column "anchor:"	
"com.cnn.www"	t9	"anchor:cnnsi.com"	"CNN"
	t8	"anchor:my.look.ca"	"CNN.com"

- 概念视图中空白的列实际不会被存储，当请求这些空白的单元格的时候，会返回null值。
- 如果在查询的时候不提供时间戳，那么会返回距离现在最近的那一个版本的数据（因为在存储的时候，数据会按照时间戳排序）。

课程提要

- Hbase简介
- HBase使用场景和成功案例
- HBase和传统关系数据库的对比分析
- HBase访问接口
- HBase数据模型
- HBase的实现
- HBase系统架构
- HBase存储格式
- 读写数据
- MapReduce on HBase



HBase的实现

HBase的实现包括三个主要的功能组件：

1. 库函数：链接到每个客户端
2. 一个HMaster主服务器
3. 许多个HRegion服务器

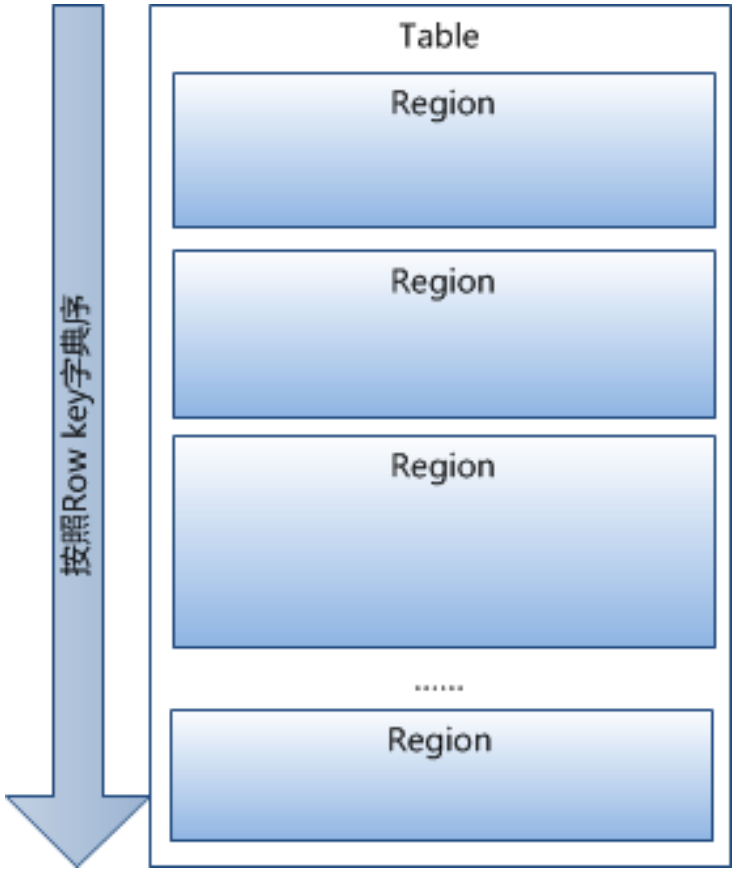
- HRegion服务器可以根据工作负载的变化，从一个簇中动态地增加或删除。主服务器HMaster负责把Hregion分配到HRegion服务器，进行HRegion服务器的负载均衡。
- 每个HRegion服务器管理一个HRegion集合，通常在每个HRegion服务器上，会放置10到1000个HRegion。HRegion服务器处理针对那些已经加载的HRegion而提出的读写请求，并且会对过大的HRegion进行划分。
- 客户端直接从HRegion服务器上读取数据，并不依赖于主服务器HMaster来获得HRegion的位置信息，所以在实际应用中，主服务器负载很小。

表和HRegion

一个HBase中存储了许多表。每个表都是一个HRegion集合，每个HRegion包含了位于某个域区间内的所有数据。

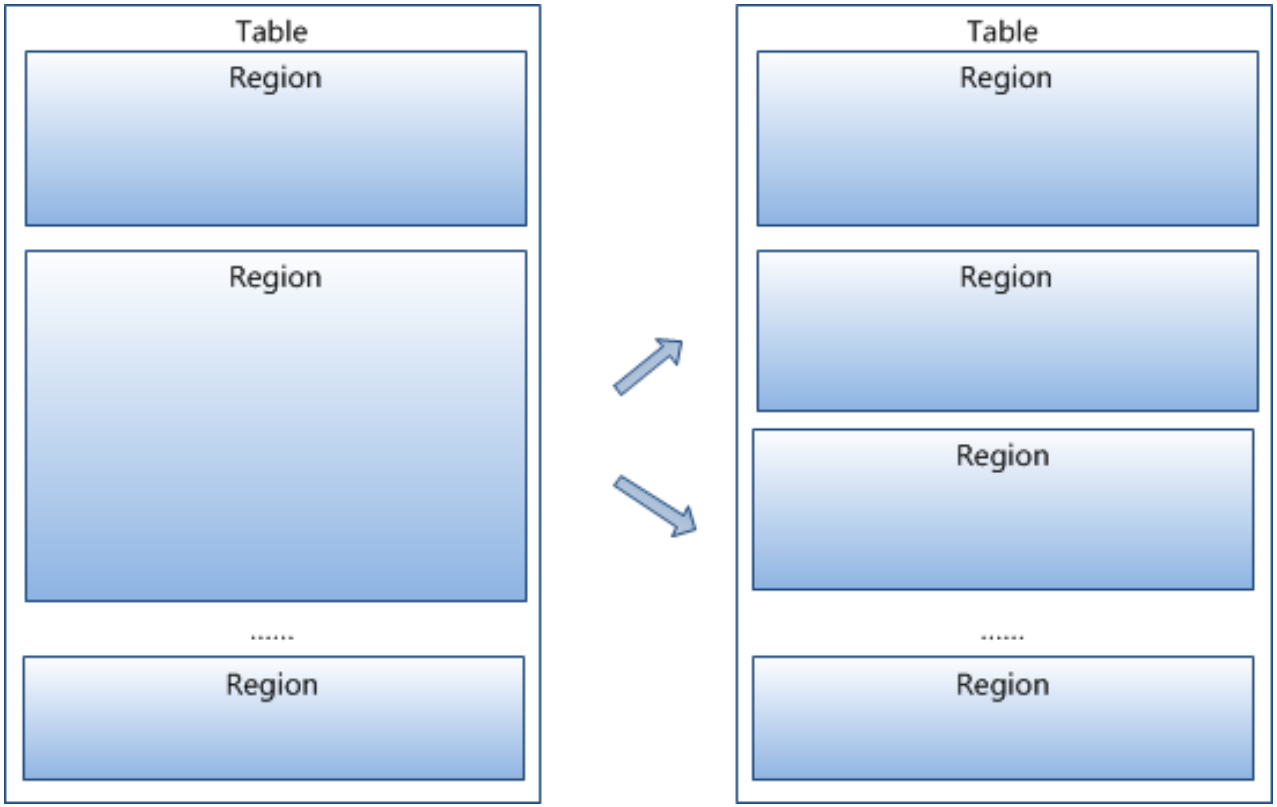
HBase中的表和HRegion的概念总结：

- 表中的所有行都按照行键的字典序排列。
- 表在行的方向上分割为多个HRegion（如右图所示）。



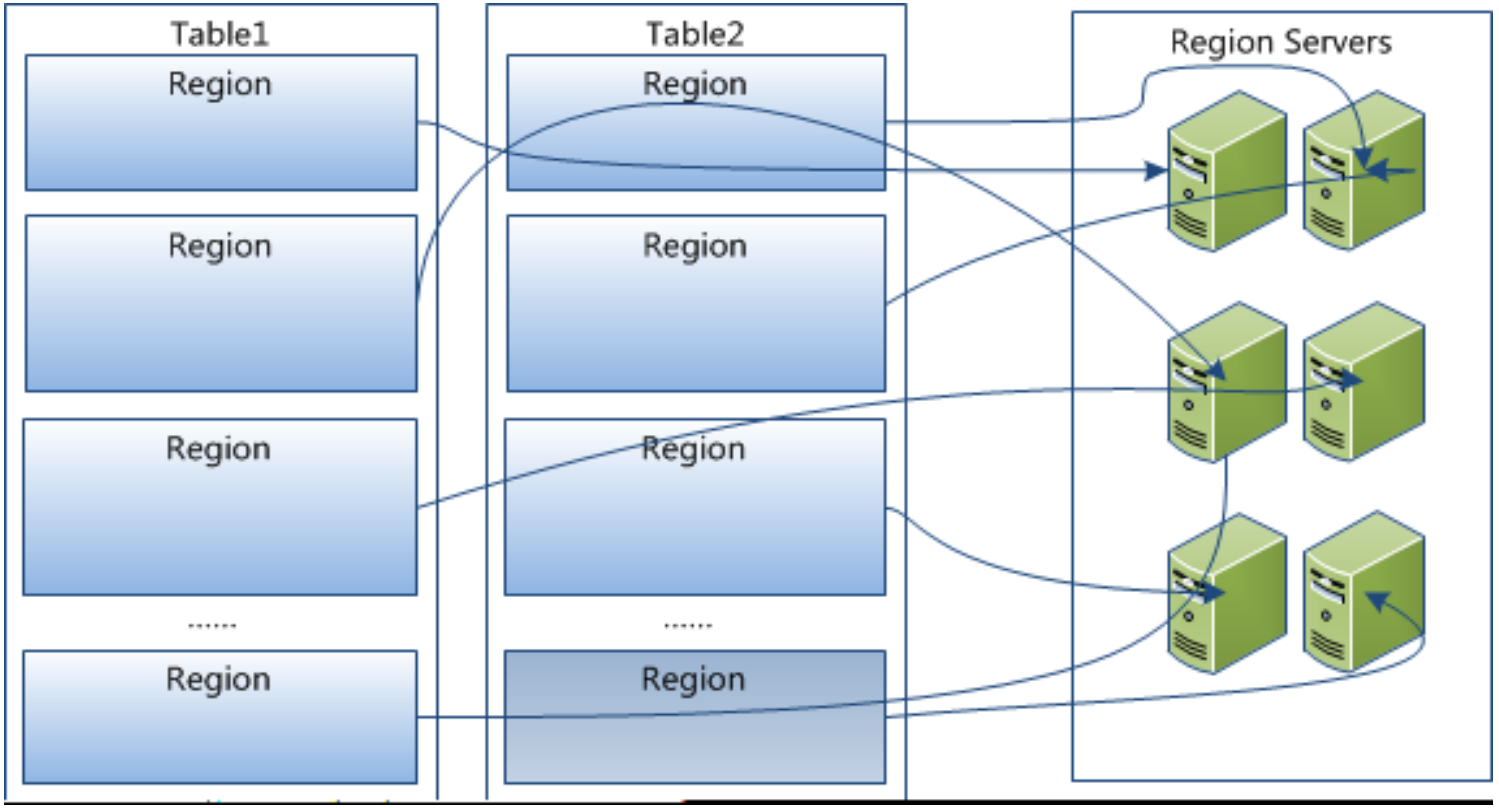
表和HRegion

- HRegion会按照大小进行分割，每个表一开始只有一个HRegion，随着数据不断插入表，HRegion不断增大，当增大到一个阀值的时候，HRegion就会等分成两个新的HRegion。不同的HRegion会被HMaster分配给相应的HRegionServer进行管理。



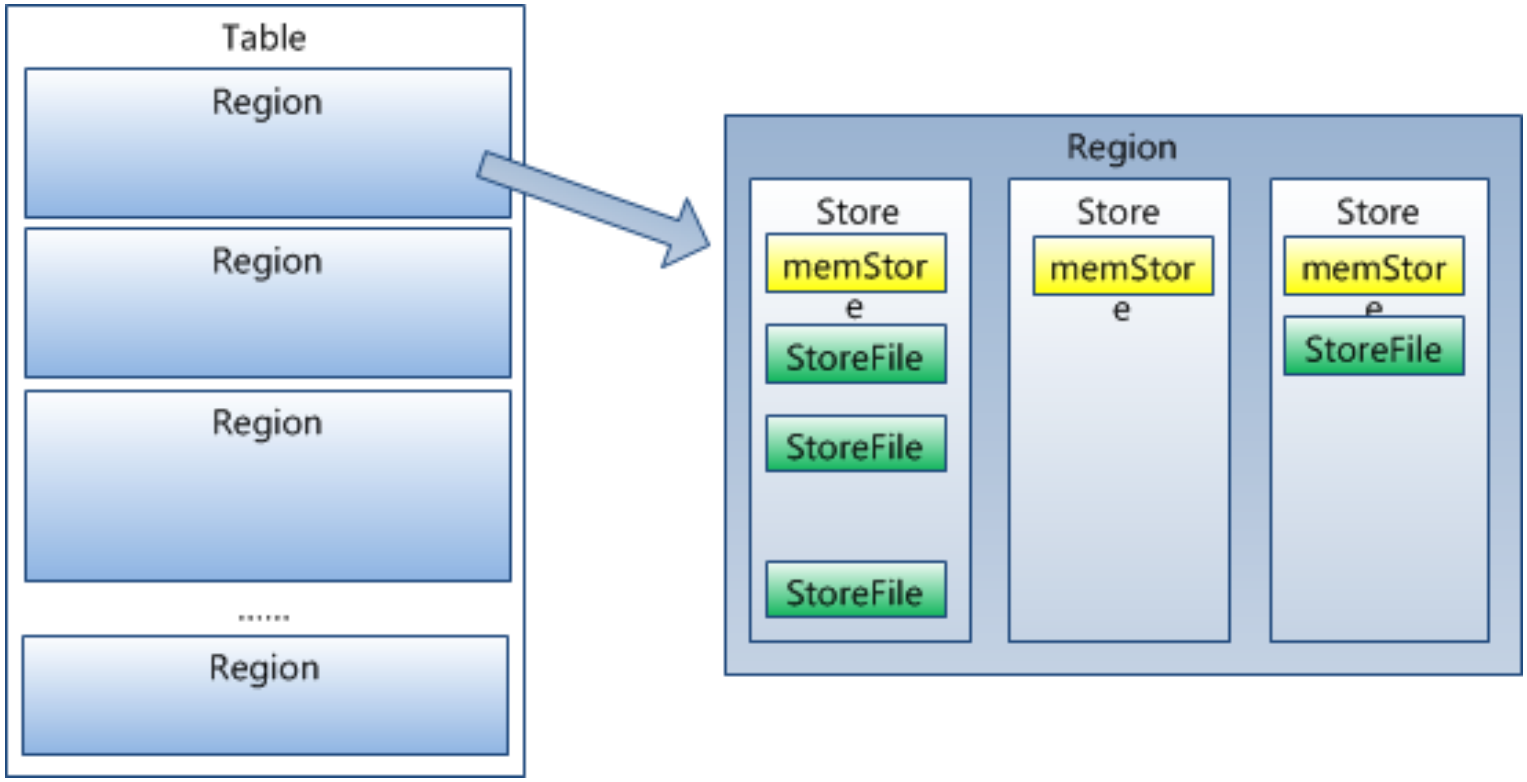
表和HRegion

- HRegion是HBase中分布式存储和负载均衡的最小单元。最小单元就表示不同的HRegion可以分布在不同的HRegionServer上，但同一个HRegion是不会拆分到多个HRegionServer上的。



表和HRegion

- HRegion虽然是分布式存储的最小单元，但并不是底层存储的最小单元。事实上，HRegion由一个或者多个HStore组成，每个HStore保存一个列族。每个HStrore又由一个memStore和0至多个HStoreFile组成。HStoreFile以HFile格式保存在HDFS上。



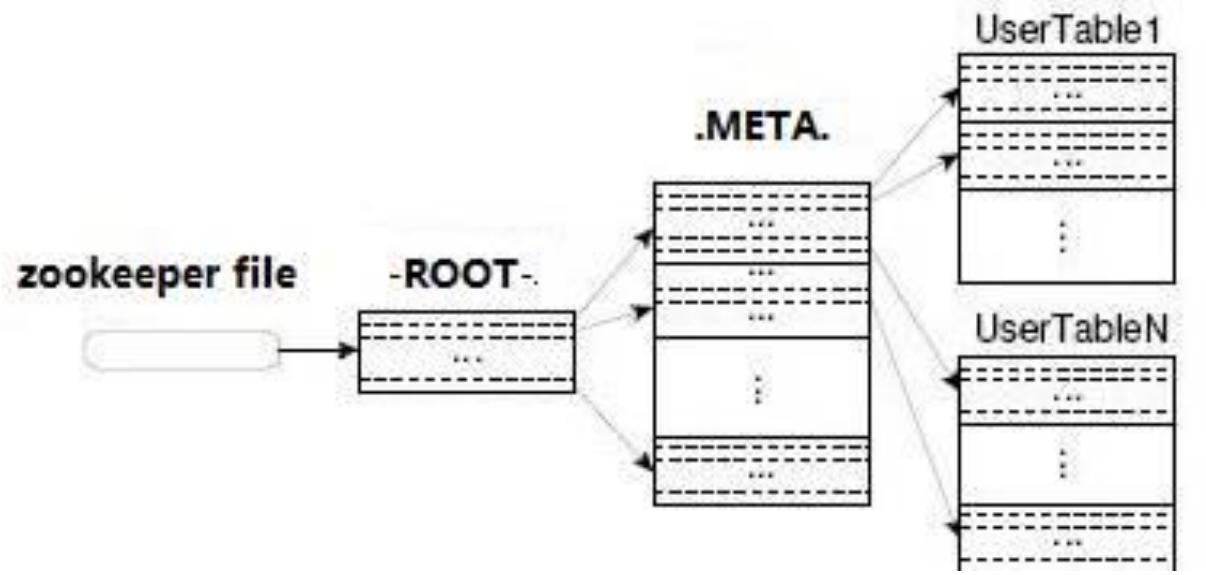
HBase三层结构

HBase使用三层类似B+树的结构来保存HRegion位置信息：

1.Zookeeper文件：它记录了-ROOT-表的位置信息，即root region的位置信息；

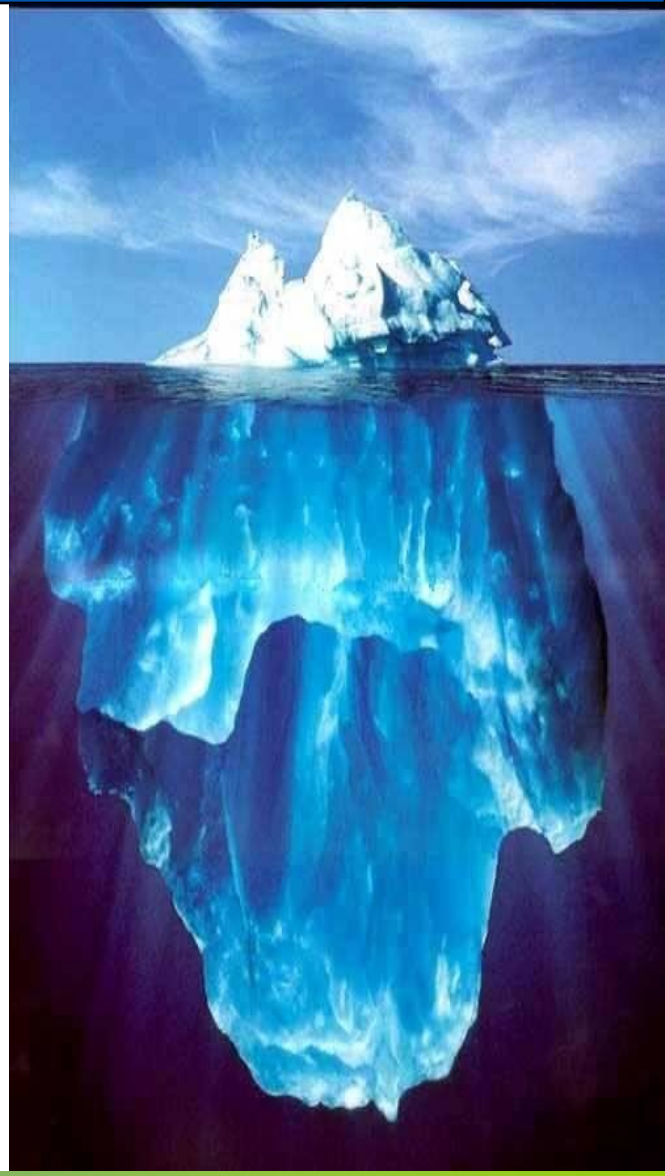
2.-ROOT-表：只包含一个root region，记录了.META.表中的region信息。通过root region，我们就可以访问.META.表的数据。

3..META.表：记录了用户表的HRegion信息，.META.表可以有多个HRegion，保存了HBase中所有数据表的HRegion位置信息。



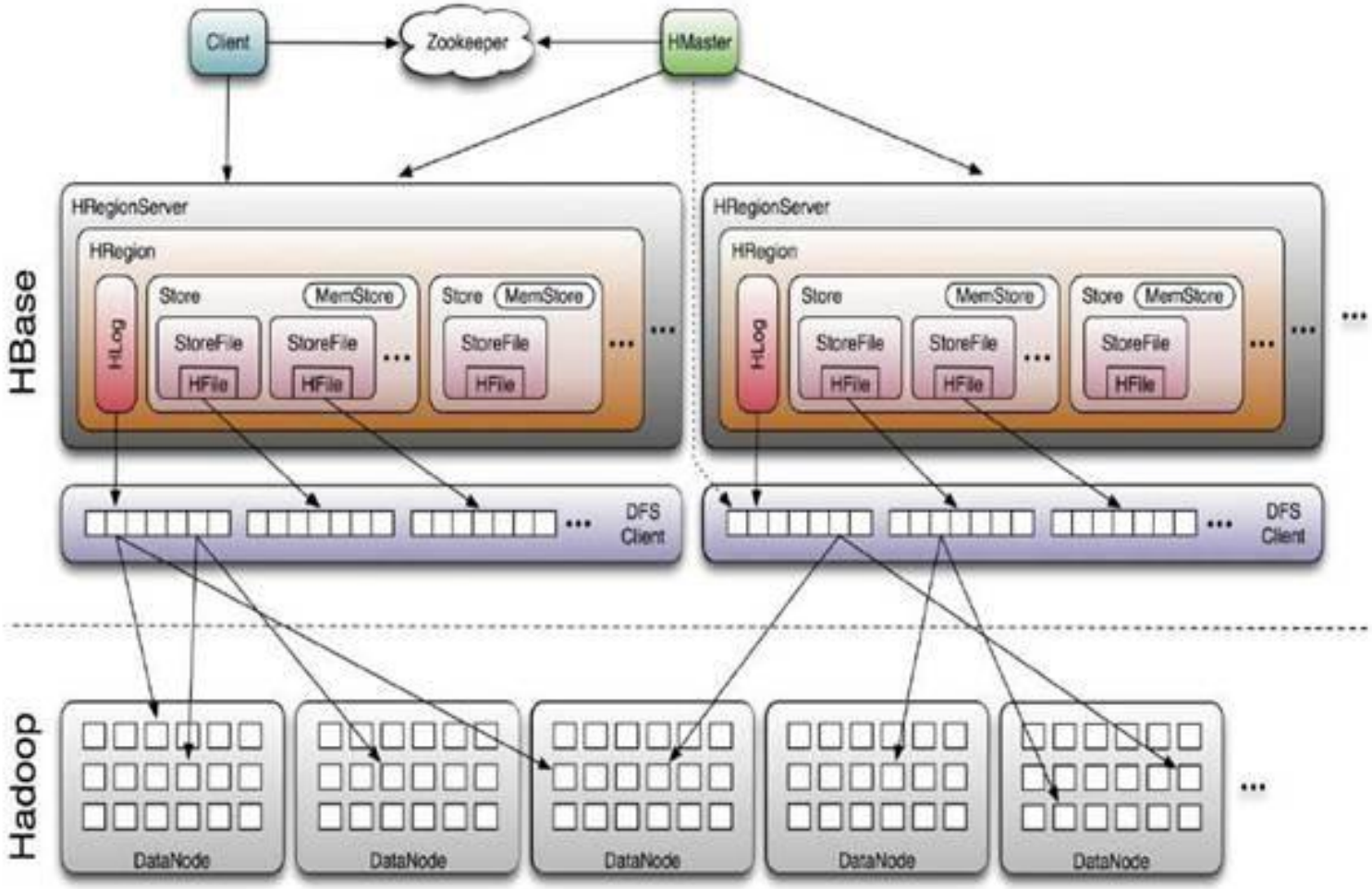
课程提要

- Hbase简介
- HBase使用场景和成功案例
- HBase和传统关系数据库的对比分析
- HBase访问接口
- HBase数据模型
- HBase的实现
- HBase系统架构
- HBase存储格式
- 读写数据
- MapReduce on HBase



HBase系统架构

- 下图是HBase的系统架构。



Client

- 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。
- Client包含访问HBase的接口，Client维护着一些缓存来加快对HBase的访问，比如HRegione的位置信息。
- 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中除了存储-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。

Zookeeper的作用，总结如下：

- 保证任何时候，集群中只有一个HMaster；
- 存储所有HRegion的寻址入口；
- 实时监控HRegionServer的状态，将HRegionServer的上线和下线信息实时通知给HMaster；
- 存储HBase的schema，包括有哪些表，每个表有哪些列族。

HMaster

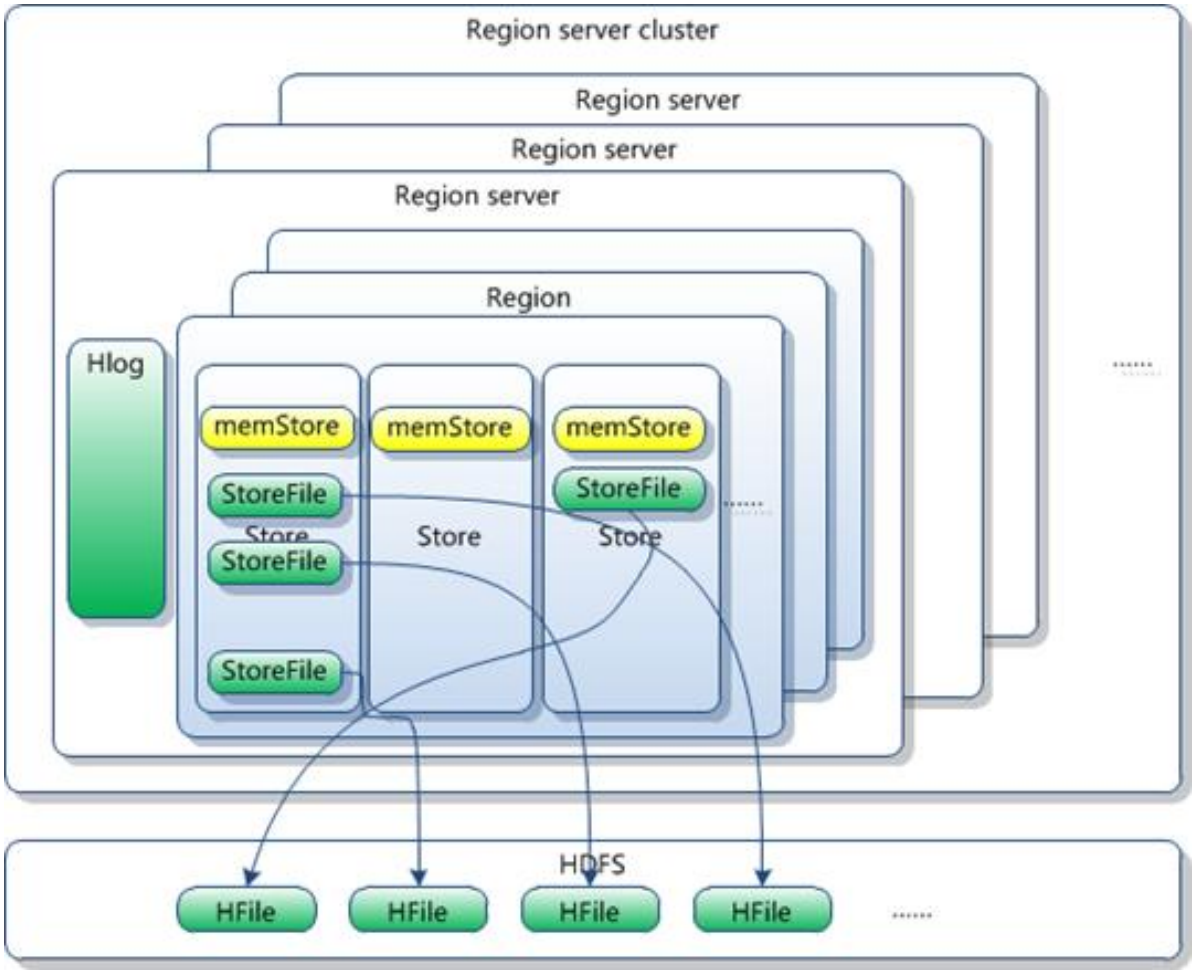
HMaster没有单点问题，HBase中可以启动多个HMaster，通过Zookeeper的Master Election机制保证总有一个HMaster运行。

HMaster在功能上主要负责表和HRegion的管理工作：

- 管理用户对表的增、删、改、查操作；
- 管理HRegionServer的负载均衡，调整HRegion分布；
- 在HRegion分裂后，负责新HRegion的分配；
- 在HRegionServer停机后，负责失效HRegionServer上的HRegion的迁移。

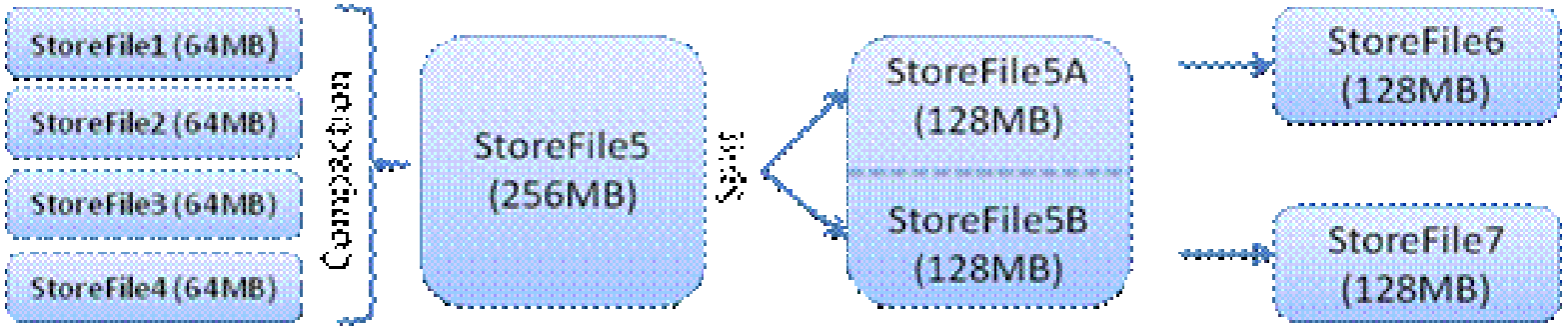
HRegionServer

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。



HStore

- HStore存储是HBase存储的核心了，由两部分组成，一部分是HMemStore，一部分是HStoreFile。
- 用户写入的数据首先会放入HMemStore，当HMemStore满了以后会Flush成一个HStoreFile。当HStoreFile文件数量增长到一定阈值，会触发合并操作，将多个HStoreFile合并成一个HStoreFile。
- 合并会形成越来越大的HStoreFile，当单个HStoreFile大小超过一定阈值后，会触发分裂操作，同时把当前HRegion 分裂成2个Hregion。
- 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，HMemStore中的内存数据将会丢失。这就需要引入HLog，当HRegionServer重启后进行数据恢复。



HRegion分配

- 任何时刻，一个HRegion只能分配给一个HRegionServer。
- HMaster记录了当前有哪些可用的HRegionServer，以及当前哪些HRegion分配给了哪些HRegionServer，哪些HRegion还没有分配。
- 当存在未分配的HRegion时，并且有一个HRegionServer上有可用空间时，HMaster就给这个HRegionServer发送一个装载请求，把HRegion分配给这个HRegionServer。HRegionServer得到请求后，就开始对此HRegion提供服务。

HRegionServer上线

- Master使用Zookeeper来跟踪HRegionServer的状态。
- 当某个HRegionServer启动时，会首先在Zookeeper上的server目录下建立代表自己的文件，并获得该文件的独占锁。由于HMaster订阅了server目录上的变更消息，当server目录下的文件出现新增或删除操作时，HMaster可以得到来自Zookeeper的实时通知。因此，一旦HRegionServer上线，HMaster能马上得到消息。

HRegionServer下线

- 当HRegionServer下线时，它和Zookeeper的会话断开，Zookeeper而自动释放代表这台server的文件上的独占锁。而HMaster不断轮询server目录下文件的锁状态。如果HMaster发现某个HRegionServer丢失了它自己的独占锁(或者HMaster连续几次和HRegionServer通信都无法成功)，HMaster就尝试去获取代表这个HRegionServer的读写锁，一旦获取成功，则表明HRegionServer和Zookeeper之间的网络断开了或是HRegionServer挂了。
- 上述情况发生时，HRegionServer都无法继续为它的HRegion提供服务了，此时HMaster会删除server目录下代表这台HRegionServer的文件，并将这台HRegionServer的HRegion分配给其它还活着的HRegionServer。
- 如果网络短暂出现问题导致HRegionServer丢失了它的锁，那么HRegionServer重新连接到Zookeeper之后，只要代表它的文件还在，它就会不断尝试获取这个文件上的锁，一旦获取到了，就可以继续提供服务。

HMaster上线

HMaster启动时，需要执行以下步骤：

1.从Zookeeper上获取唯一一个代表该HMaster的锁，用来阻止其它HMaster成为主服务器；

2.扫描Zookeeper上的server目录，获得当前可用的HRegionServer列表；

3.和第二步中的每个HRegionServer通信，获得当前已分配的HRegion和HRegionServer的对应关系；

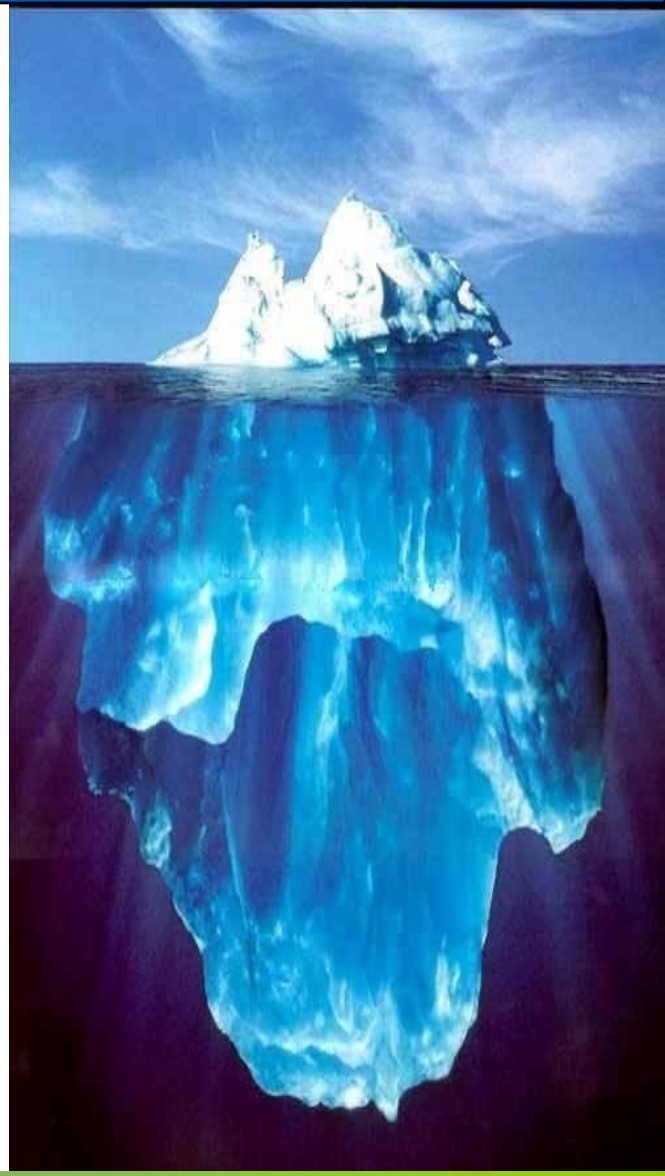
4.扫描.META.中HRegion的集合，计算得到当前还未分配的HRegion，将他们放入待分配HRegion列表。

HMaster下线

- 由于HMaster只维护表和HRegion的元数据，而不参与表数据IO的过程，HMaster下线，仅导致所有元数据的修改被冻结(无法创建删除表，无法修改表的schema，无法进行HRegion的负载均衡，无法处理HRegion上下线，无法进行HRegion的合并，唯一例外的是HRegion的分裂可以正常进行，因为只有HRegionServer参与)，表的数据读写还可以正常进行。
- 因此，HMaster下线短时间内对整个HBase集群没有影响。
- 从上线过程可以看到，HMaster保存的信息全是可以冗余信息（都可以从系统其它地方收集到或者计算出来），因此，一般HBase集群中总是有一个HMaster在提供服务，还有一个以上的HMaster在等待时机抢占它的位置。

课程提要

- HBase简介
- HBase使用场景和成功案例
- HBase和传统关系数据库的对比分析
- HBase访问接口
- HBase数据模型
- HBase的实现
- HBase系统架构
- HBase存储格式
- 读写数据
- MapReduce on HBase



HBase存储格式

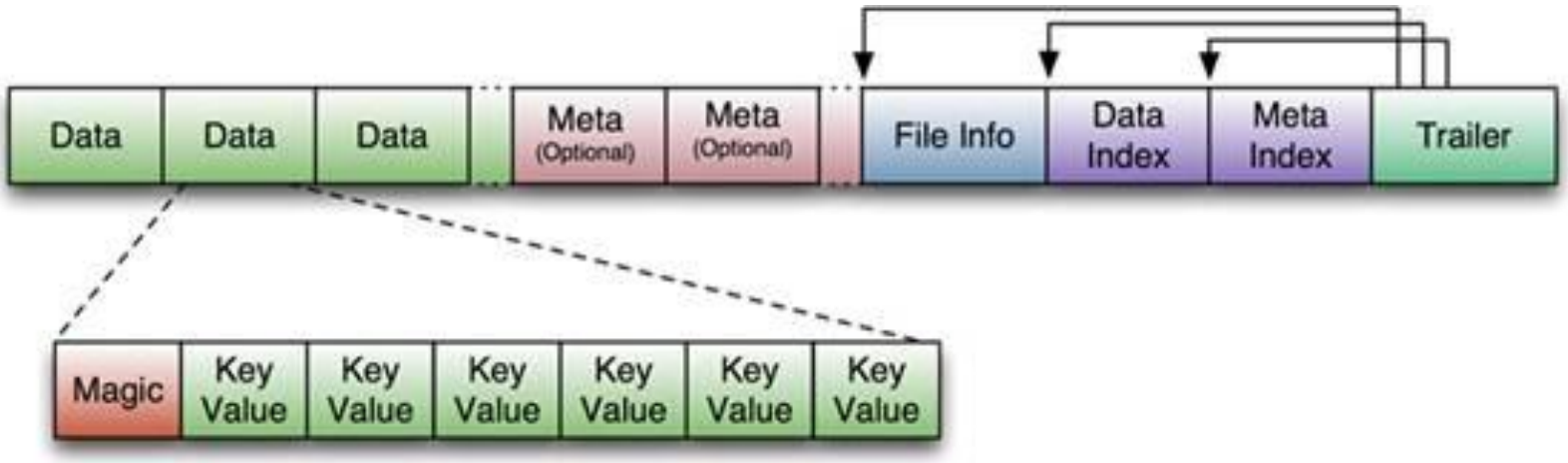
HBase中的所有数据文件都存储在Hadoop分布式文件系统HDFS上，主要包括上述提出的两种文件类型：

- **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HStoreFile就是对HFile做了轻量级包装，即HStoreFile底层就是HFile。
- **HLog File**：HBase中WAL（Write Ahead Log）的存储格式，物理上是Hadoop的顺序文件。

HFile

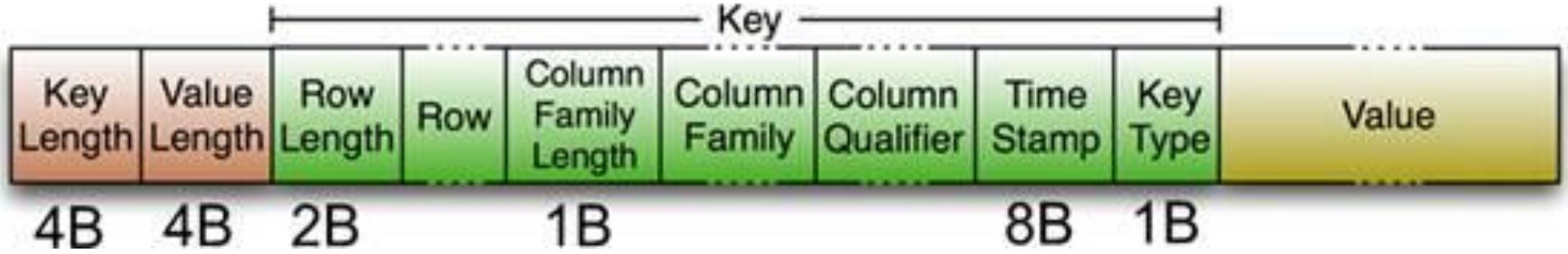
HFile分为六个部分：

- **Data Block**：保存表中的数据，这部分可以被压缩。
- **Meta Block** (可选的)：保存用户自定义的key/value对，可被压缩。
- **File Info**：HFile的元信息，不被压缩，用户可在这部分添加自己的元信息。
- **Data Block Index**：Data Block的索引。
- **Meta Block Index**(可选的)：Meta Block的索引。
- **Trailer**：读取HFile时会首先读取Trailer，Trailer保存了每个段的起始位置(Magic Number用来做安全check)，然后，DataBlock Index会被读取到内存中，这样当检索某个key时，不需要扫描整个HFile，只需从内存中找到key所在的block，通过一次磁盘io将整个block读取到内存中，再找到需要的key。



HFile

HFile里面的每个Key/Value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。如下图所示：



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。

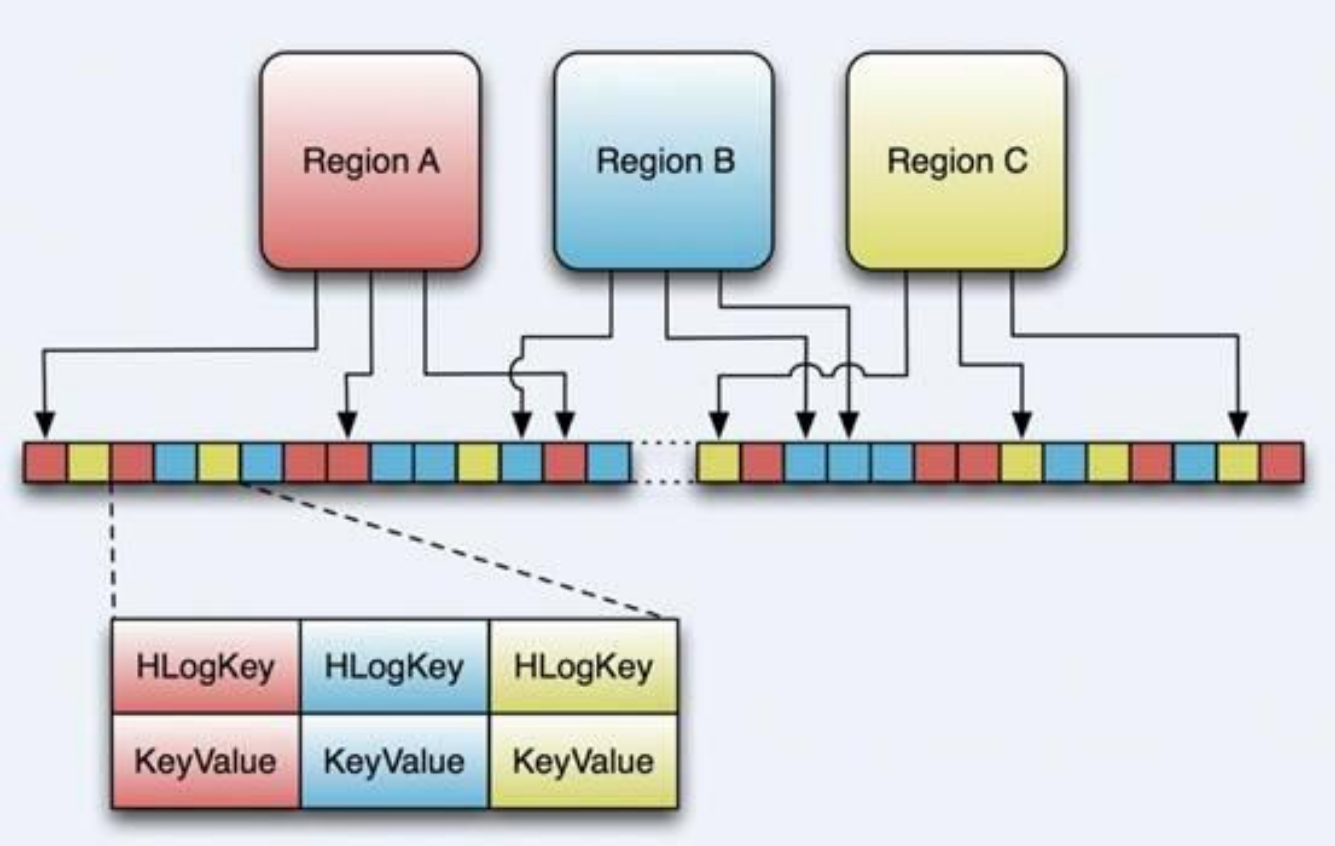
HLogFile

HLog又称WAL。WAL意为Write Ahead Log，类似Mysql中的binlog，用来做灾难恢复，HLog记录数据的所有变更，一旦数据修改，就可以从HLog中进行恢复。

每个HRegion Server维护一个HLog，而不是每个HRegion一个。这样不同HRegion(来自不同表)的日志会混在一起，这样做的目的是，不断追加单个文件相对于同时写多个文件而言，可以减少磁盘寻址次数，因此，可以提高对表的写性能。带来的麻烦是，如果一台HRegionServer下线，为了恢复其上的HRegion，需要将HRegionServer上的HLog进行拆分，然后分发到其它HRegionServer上进行恢复。

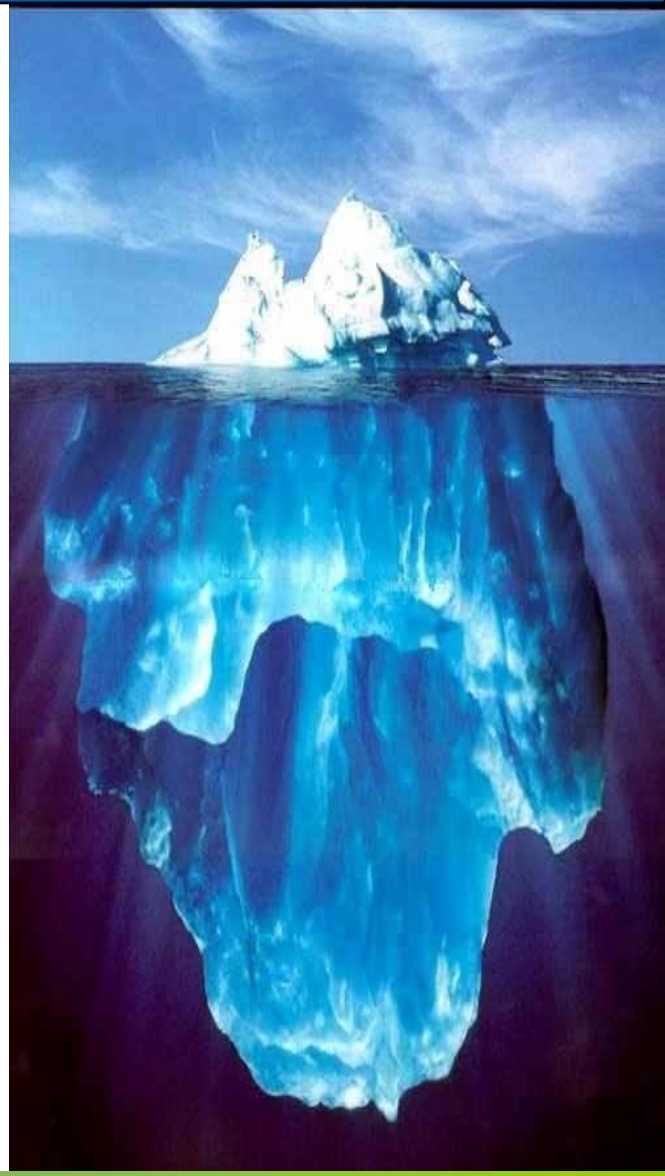
HLogFile

HLog文件就是一个普通的Hadoop顺序文件（Sequence File），顺序文件的Key是HLogKey对象，HLogKey中记录了写入数据的归属信息，除了表和HRegion名字外，同时还包括顺序号和时间戳。HLog 顺序文件的Value是HBase的Key/Value对象，即对应HFile中的Key/Value。



课程提要

- HBase简介
- HBase使用场景和成功案例
- HBase和传统关系数据库的对比分析
- HBase访问接口
- HBase数据模型
- HBase系统架构
- HBase存储格式
- 读写数据
- MapReduce on HBase



读写数据

- HBase使用HMemStore和HStoreFile存储对表的更新。
- 数据在更新时，首先写入HLog和内存(HMemStore)中，HMemStore中的数据是排序的，当HMemStore累计到一定阈值时，就会创建一个新的HMemStore，并且将老的HMemStore添加到flush队列，由单独的线程flush到磁盘上，成为一个HStoreFile。与此同时，系统会在Zookeeper中记录一个检查点，表示这个时刻前的变更已持久化了。
- 当系统出现意外时，可能导致内存(HMemStore)中的数据丢失，此时使用HLog来恢复检查点之后的数据。
- HStoreFile是只读的，一旦创建后就不可以再修改。因此HBase的更新其实是不断追加的操作。当一个HStore中的HStoreFile达到一定的阈值后，就会进行一次合并，将对同一个key的修改合并到一起，形成一个大的HStoreFile，当HStoreFile的大小达到一定阈值后，又会对HStoreFile进行分裂，等分为两个HStoreFile。

读写数据

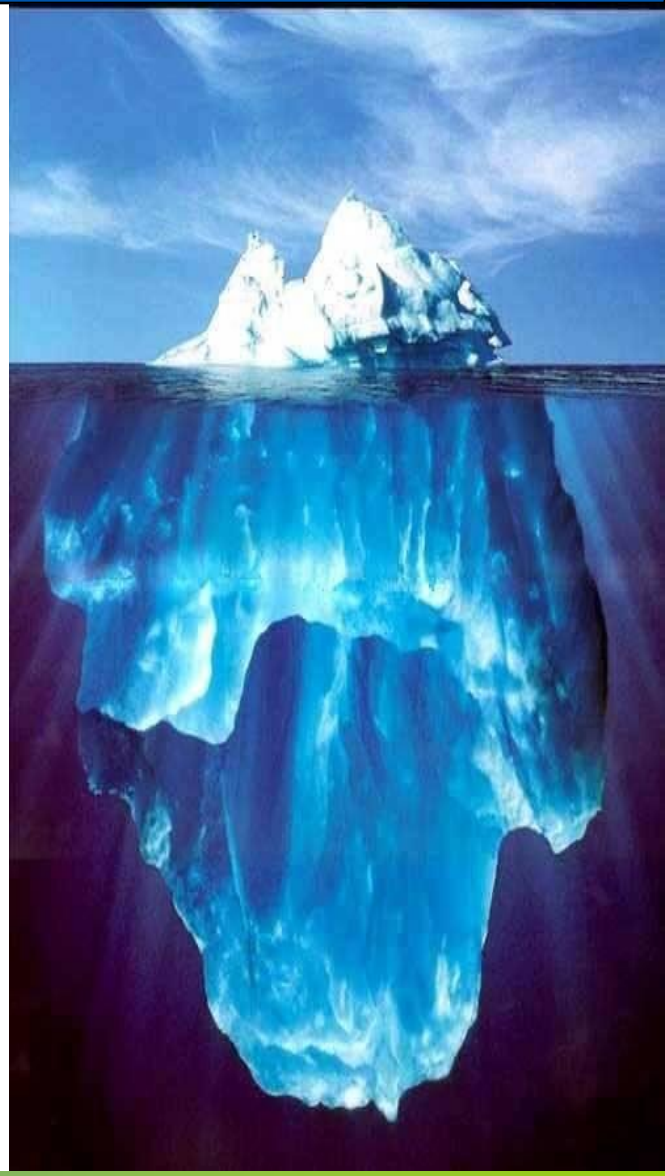
由于对表的更新是不断追加的，处理读请求时，需要访问HStore中全部的HStoreFile和HMemStore，将他们的按照行键进行合并，由于HStoreFile和HMemStore都是经过排序的，并且HStoreFile带有内存中索引，合并的过程还是比较快的。

写请求处理过程具体如下：

- client向HRegionServer提交写请求；
- HRegionServer找到目标HRegion；
- HRegion检查数据是否与schema一致；
- 如果客户端没有指定版本，则获取当前系统时间作为数据版本；
- 将更新写入HLog；
- 将更新写入HMemstore；
- 判断HMemStore的是否需要flush为HStore文件。

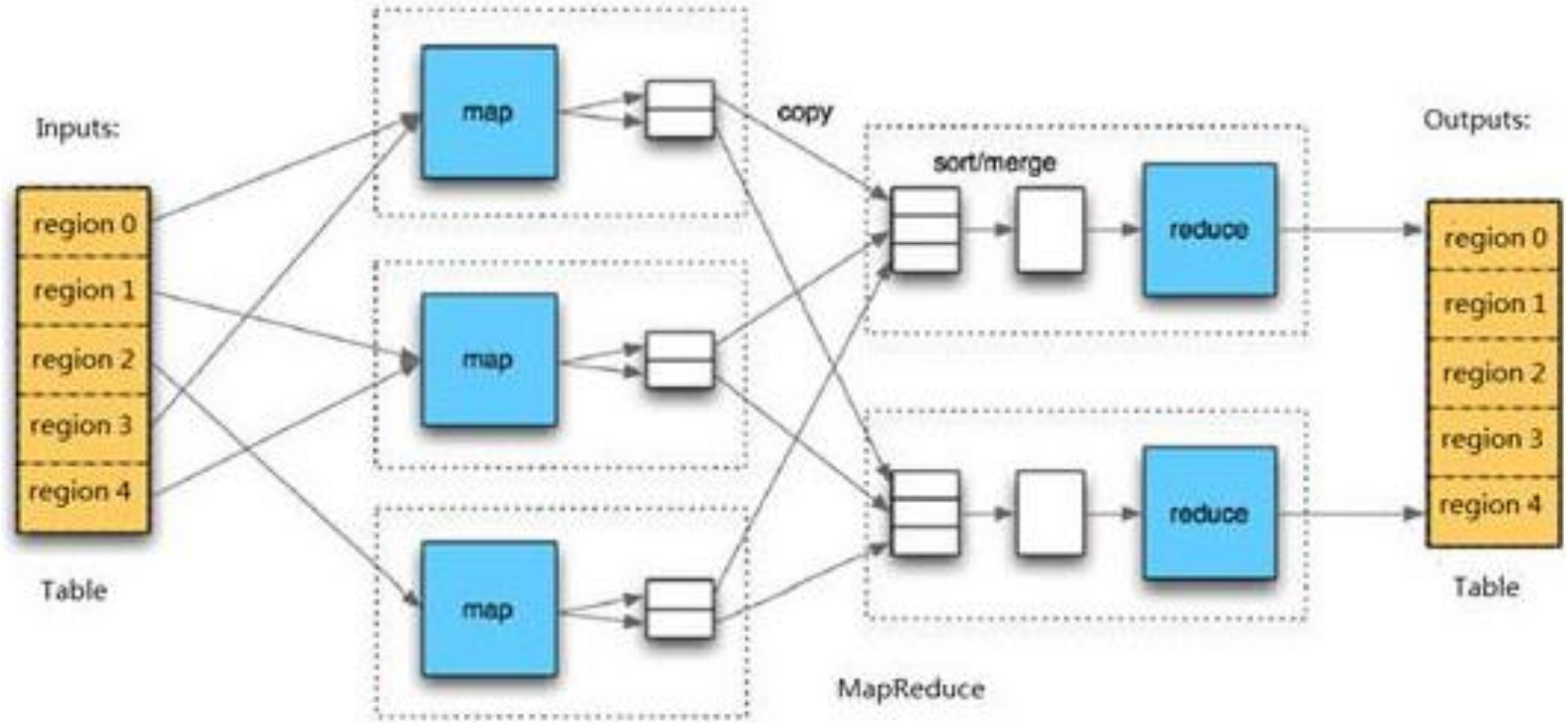
课程提要

- HBase简介
- HBase使用场景和成功案例
- HBase和传统关系数据库的对比分析
- HBase访问接口
- HBase数据模型
- HBase系统架构
- HBase存储格式
- 读写数据
- MapReduce on HBase



读写数据

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。HBase提供了配套的TableInputFormat和TableOutputFormat API，可以方便地将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。



主讲教师和助教

主讲教师：林子雨

单位：厦门大学计算机科学系
E-mail: ziyulin@xmu.edu.cn
个人网页：http://www.cs.xmu.edu.cn/linziyu
数据库实验室网站：http://dblab.xmu.edu.cn

助教：赖明星

单位：厦门大学计算机科学系数据库实验室2011级硕士研究生（导师：林子雨）
E-mail: mingxinglai@gmail.com
个人主页：http://mingxinglai.com

欢迎访问《大数据技术基础》2013班级网站：http://dblab.xmu.edu.cn/node/423
本讲义PPT存在配套教材《大数据技术基础》，请到上面网站下载。

Department of Computer Science, Xiamen University, September, 2013