HBase应用场景、原理与基本架构

讲师：董西成
博客：dongxicheng.org

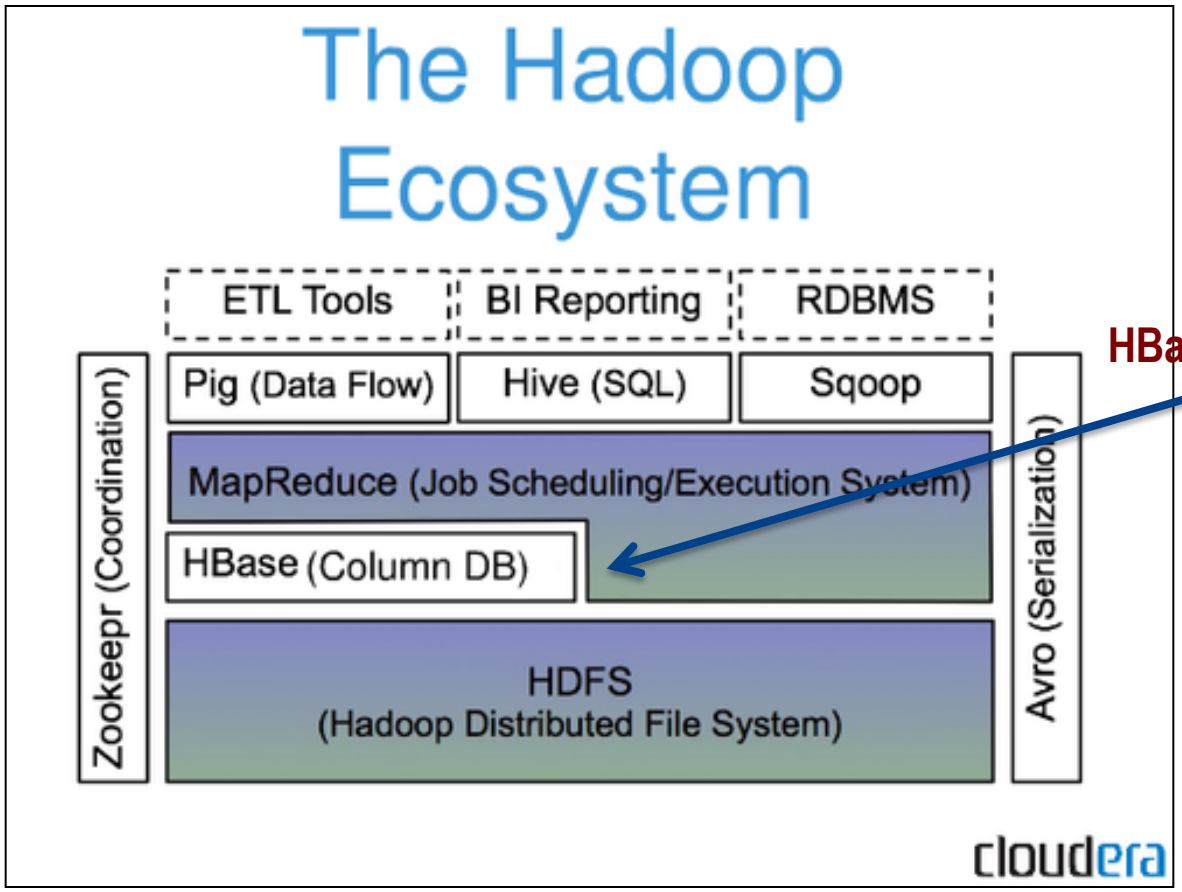
Open Passion Value

目录

HBase概述

- HBase是一个构建在HDFS上的分布式列存储系统；
- HBase是Apache Hadoop生态系统中的重要一员，主要用于海量结构化数据存储；
- 从逻辑上讲，HBase将数据按照表、行和列进行存储。

Hbase是Hadoop生态系统的一个组成部分



HBase 构建在HDFS之上

Hbase内部管理的文件全部存储在HDFS中

Hbase与HDFS对比

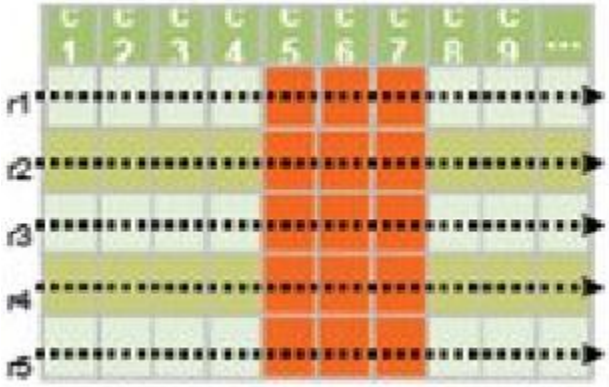
- 两者都具有良好的容错性和扩展性，都可以扩展到成百上千个节点；
- **HDFS适合批处理场景**
  - ✓ 不支持数据随机查找
  - ✓ 不适合增量数据处理
  - ✓ 不支持数据更新

Hbase表的特点

- 大：一个表可以有数十亿行，上百万列；
- 无模式：每行都有一个可排序的主键和任意多的列，列可以根据需要动态的增加，同一张表中不同的行可以有截然不同的列；
- 面向列：面向列（族）的存储和权限控制，列（族）独立检索；
- 稀疏：对于空（null）的列，并不占用存储空间，表可以设计的非常稀疏；
- 数据多版本：每个单元中的数据可以有多个版本，默认情况下版本号自动分配，是单元格插入时的时间戳；
- 数据类型单一：Hbase中的数据都是字符串，没有类型

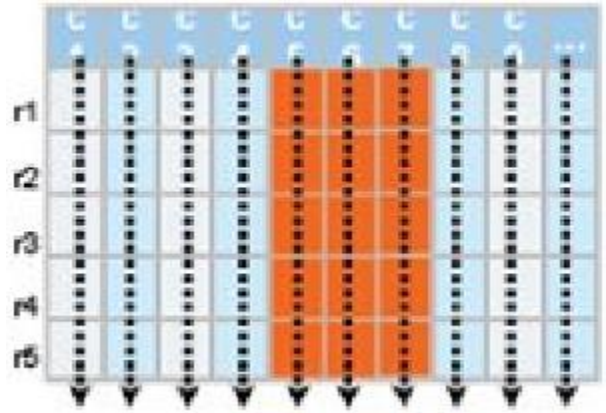
行存储与列存储

传统行式数据库



- 数据是按行存储的
- 没有索引的查询使用大量I/O
- 建立索引和物化视图需要花费大量时间和资源
- 面向查询的需求，数据库必须被大量膨胀才能满足性能要求

列式数据库

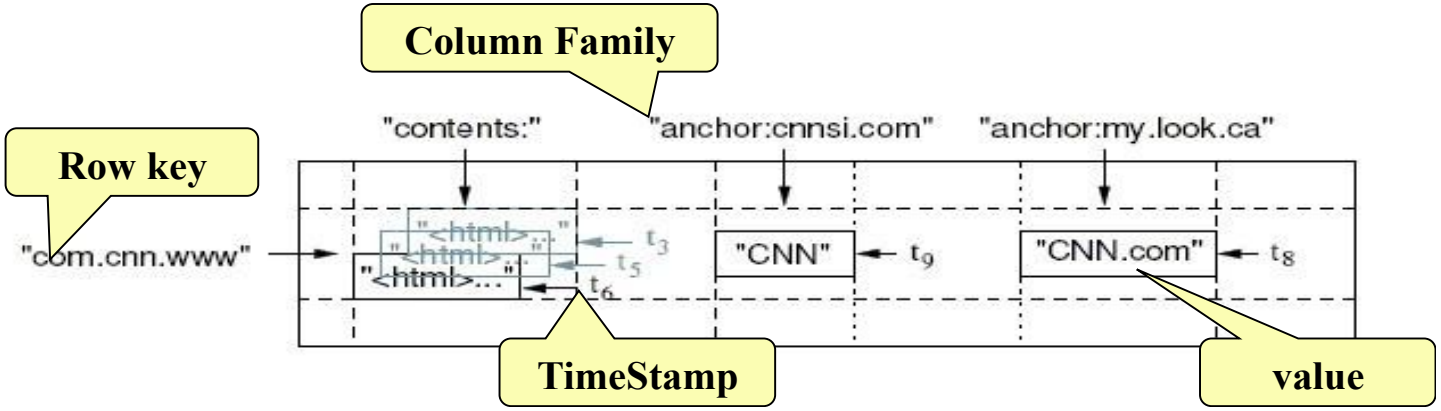


- 数据是按列存储-每一列单独存放
- 数据即是索引
- 指访问查询涉及的列-大量降低系统I/O
- 每一列由一个线索来处理-查询的并发处理
- 数据类型一致，数据特征相似-高效压缩

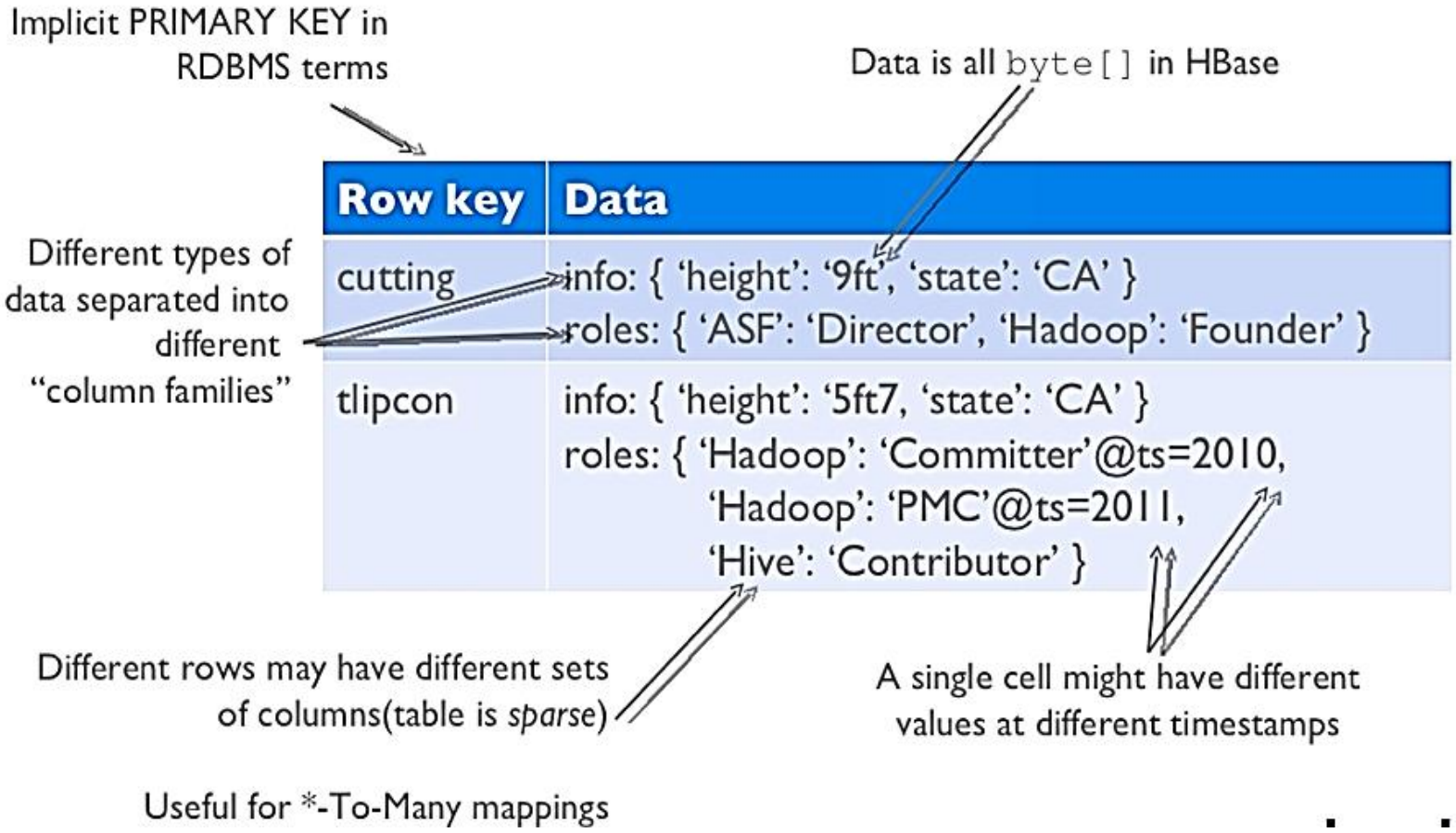
目录

Hbase数据模型

- **HBase是基于Google BigTable模型开发的，典型的key/value系统；**



Hbase逻辑视图


Implicit PRIMARY KEY in RDBMS terms
Data is all byte[] in HBase
Row key
Data
Different types of data separated into different "column families"
cutting
info: { 'height': '9ft', 'state': 'CA' }
roles: { 'ASF': 'Director', 'Hadoop': 'Founder' }
tlipcon
info: { 'height': '5ft7, 'state': 'CA' }
roles: { 'Hadoop': 'Committer'@ts=2010,
'Hadoop': 'PMC'@ts=2011,
'Hive': 'Contributor' }
Different rows may have different sets of columns(table is sparse)
A single cell might have different values at different timestamps
Useful for *-To-Many mappings

Rowkey与Column Family

每条记录被划分到若干个 ***Column Family***中

每一行与一个 ***Rowkey***

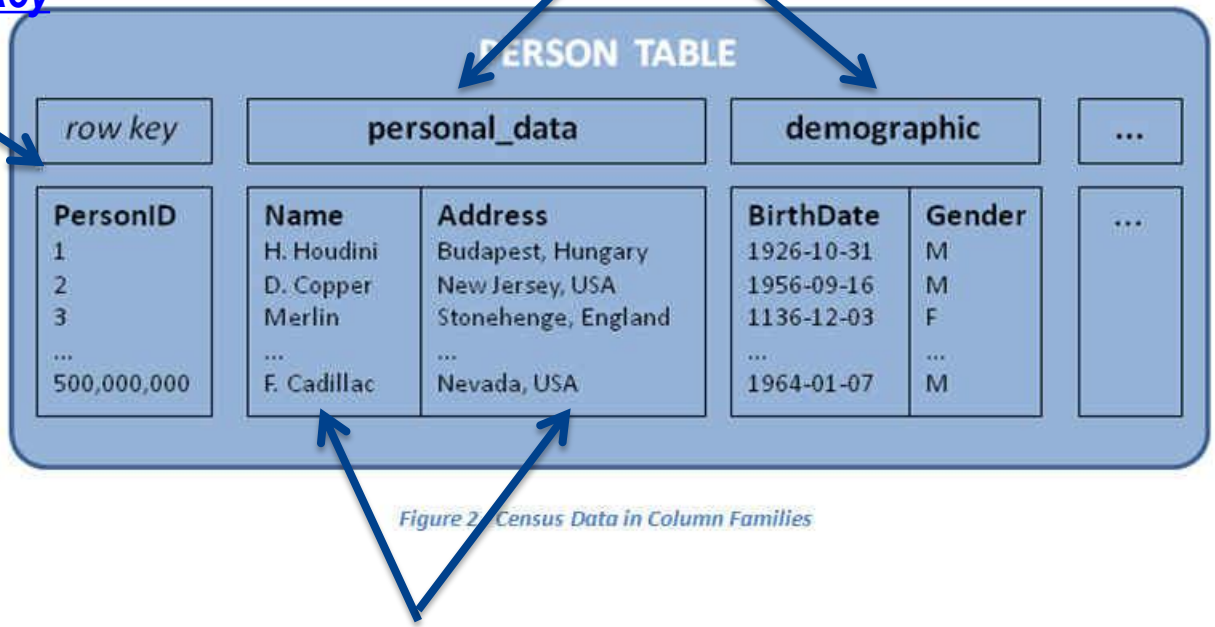


Figure 2 Census Data in Column Families

每个column family 由一个或者多个 ***Column*** 组成

Hbase基本概念

- **Row Key**
  - ✓Byte array
  - ✓表中每条记录的“主键”
  - ✓方便快速查找
- **Column Family**
  - ✓拥有一个名称(string)
  - ✓包含一个或者多个相关列
- **Column**
  - ✓属于某一个column family
  - ✓包含在某一列中
    - ***familyName:columnName***

名称为“Contents”的column family

名称为“anchor”的column family

名称为“apache.com的列”

Row key	Time Stamp	Column “contents:”	Column “anchor:”	
“com.apache.www”	t12	“<html>…”		
	t11	“<html>…”		
	t10		“anchor:apache.com”	“APACHE”
“com.cnn.www”	t15		“anchor:cnnsi.com”	“CNN”
	t13		“anchor:my.look.ca”	“CNN.com”
	t6	“<html>…”		
	t5	“<html>…”		
	t3	“<html>…”		

Hbase基本概念

每一行有一个版本号

- **Version Number**
  - ✓每个rowkey唯一
  - ✓默认值→ 系统时间戳
  - ✓类型为Long
- **Value (Cell)**
  - ✓Byte array

Row key	Time Stamp	Column “contents:”	Column “anchor:”	
“com.apache.www”	t12	“<html>…”		value
	t11	“<html>…”		
	t10		“anchor:apache.com”	“APACHE”
“com.cnn.www”	t15		“anchor:cnnsi.com”	“CNN”
	t13		“anchor:my.look.ca”	“CNN.com”
	t6	“<html>…”		
	t5	“<html>…”		
	t3	“<html>…”		

Hbase数据模型

- HBase schema可以有多个 ***Table***
- 每个表可由多个***Column Family***组成
- **HBase 可以有 *Dynamic Column***
  - ✓列名称是编码在cell中的
  - ✓不同的cell可以拥有不同的列

"Roles" column family has different columns in different cells

Row key	Data
cutting	info: { 'height': '9ft', 'state': 'CA' } roles: { 'ASF': 'Director', 'Hadoop': 'Founder' }
tlipcon	info: { 'height': '5ft7, 'state': 'CA' } roles: { 'Hadoop': 'Committer'@ts=2010, 'Hadoop': 'PMC'@ts=2011, 'Hive': 'Contributor' }

Hbase数据模型

- ***version number*** 可由用户提供
  - ✓无需以递增的顺序插入
  - ✓每一行的rowkey必须是唯一的
- **Table 可能非常稀疏**
  - ✓很多 cell 可以是空的
- ***Row Key*****是主键**

Row Key	Time Stamp	ColumnFamily contents	ColumnFamily anchor
"com.cnn.www"	t9		anchor:cnnsi.com = "CNN"
"com.cnn.www"	t8		anchor:my.look.ca = "CNN.com"
"com.cnn.www"	t6	contents:html = "<html>..."	
"com.cnn.www"	t5	contents:html = "<html>..."	
"com.cnn.www"	t3	contents:html = "<html>..."	

Hbase支持的操作

- 所有操作均是基于rowkey的；
- 支持CRUD（Create、Read、Update和Delete）和Scan；
- 单行操作
  - ✓ Put
  - ✓ Get
  - ✓ Scan
- 多行操作
  - ✓ Scan
  - ✓ MultiPut
- 没有内置join操作，可使用MapReduce解决。

目录

Hbase物理模型

- 每个column family存储在HDFS上的一个单独文件中；
- Key 和 Version number在每个 column family中均由一份；
- 空值不会被保存。

HBase 为每个值维护了多级索引，即：
<key, column family, column name, timestamp>

Table 5.3. ColumnFamily contents

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

Table 5.2. ColumnFamily anchor

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

一个实例

Row key	Data
cutting	info: { 'height': '9ft', 'state': 'CA' } roles: { 'ASF': 'Director', 'Hadoop': 'Founder' }
tlipcon	info: { 'height': '5ft7, 'state': 'CA' } roles: { 'Hadoop': 'Committer'@ts=2010, 'Hadoop': 'PMC'@ts=2011, 'Hive': 'Contributor' }

info Column Family

Row key	Column key	Timestamp	Cell value
cutting	info:height	1273516197868	9ft
cutting	info:state	1043871824184	CA
tlipcon	info:height	1273878447049	5ft7
tlipcon	info:state	1273616297446	CA

roles Column Family

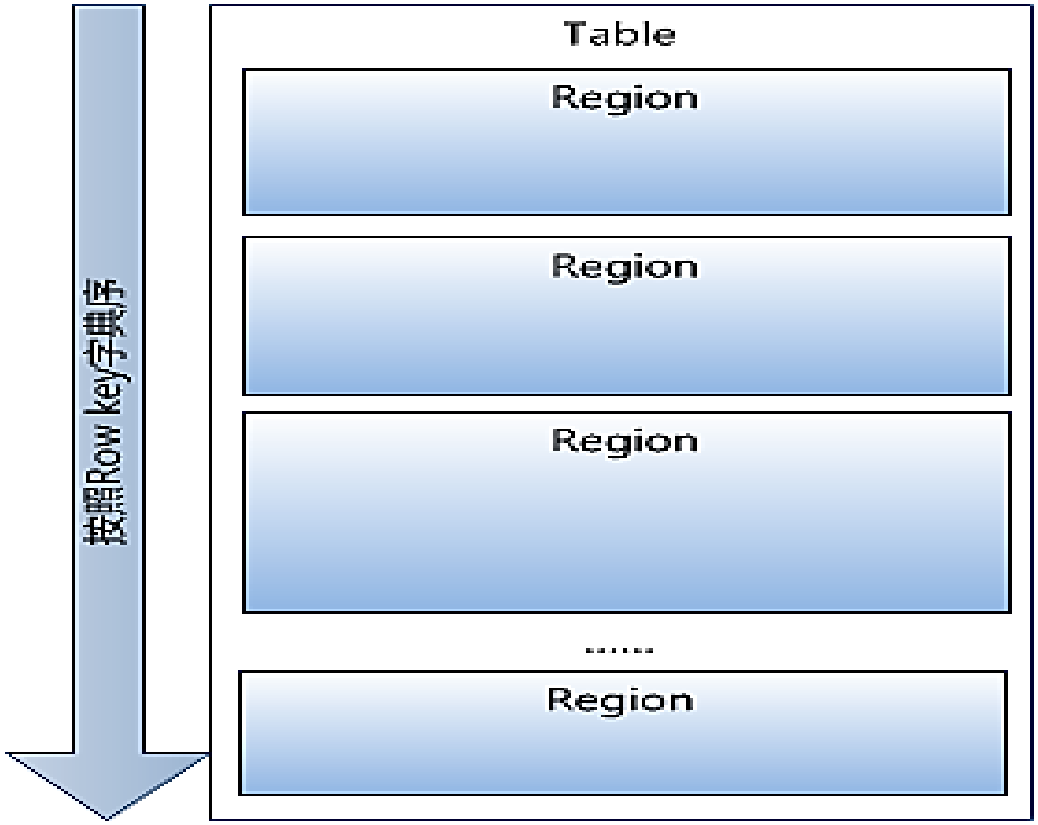
Row key	Column key	Timestamp	Cell value
cutting	roles:ASF	1273871823022	Director
cutting	roles:Hadoop	1183746289103	Founder
tlipcon	roles:Hadoop	1300062064923	PMC
tlipcon	roles:Hadoop	1293388212294	Committer
tlipcon	roles:Hive	1273616297446	Contributor

Sorted on disk by Row key, Col key, descending timestamp

Milliseconds since unix epoch

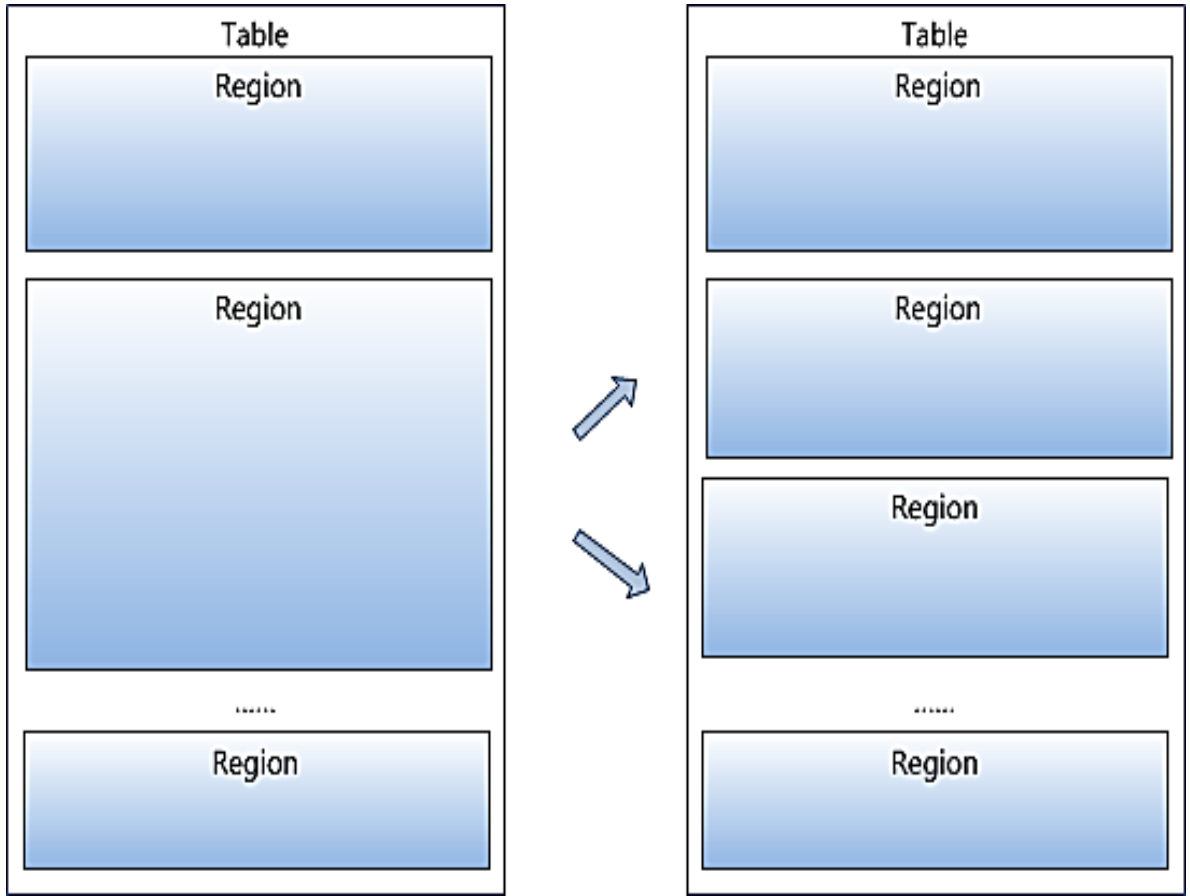
物理存储

1、Table中的所有行都按照row key的字典序排列；
2、Table 在行的方向上分割为多个Region；



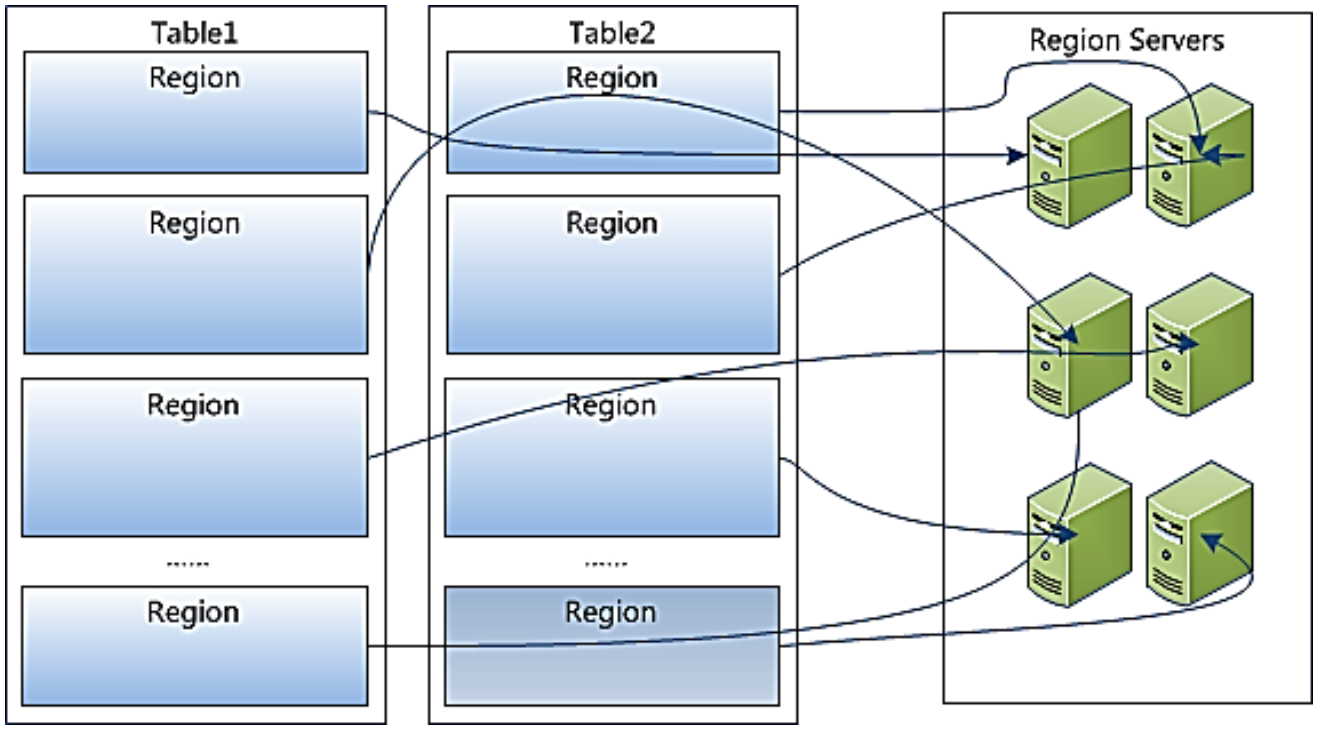
物理存储

3、Region按大小分割的，每个表开始只有一个region，随着数据增多，region不断增大，当增大到一个阀值的时候，region就会等分会两个新的region，之后会有越来越多的region；



物理存储

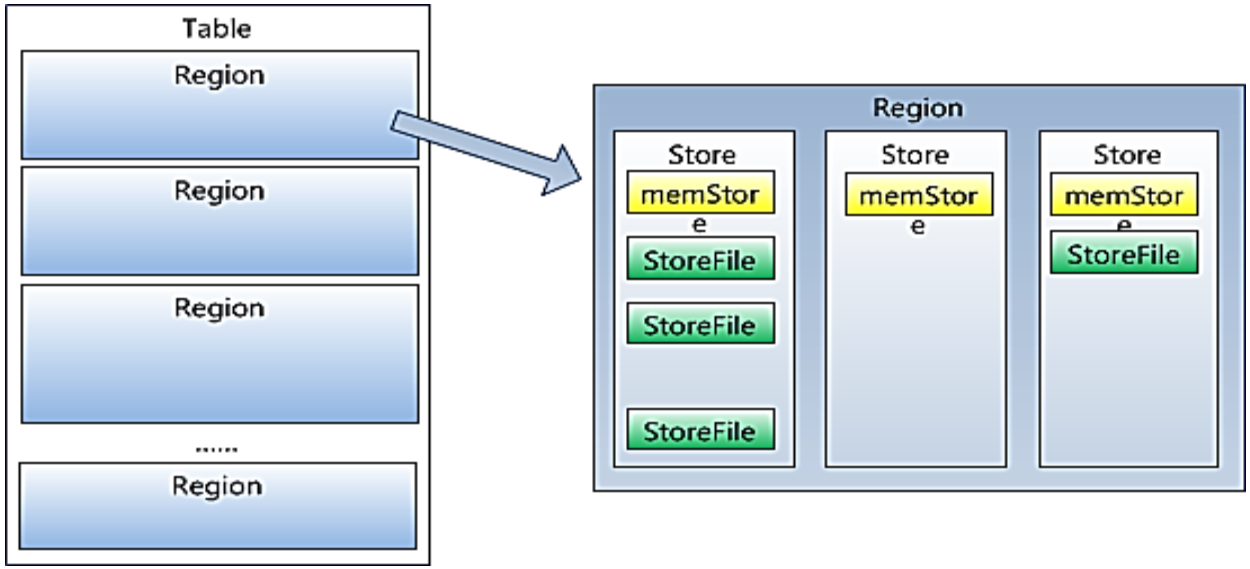
4、Region是HBase中分布式存储和负载均衡的最小单元。不同Region分布到不同RegionServer上；



物理存储

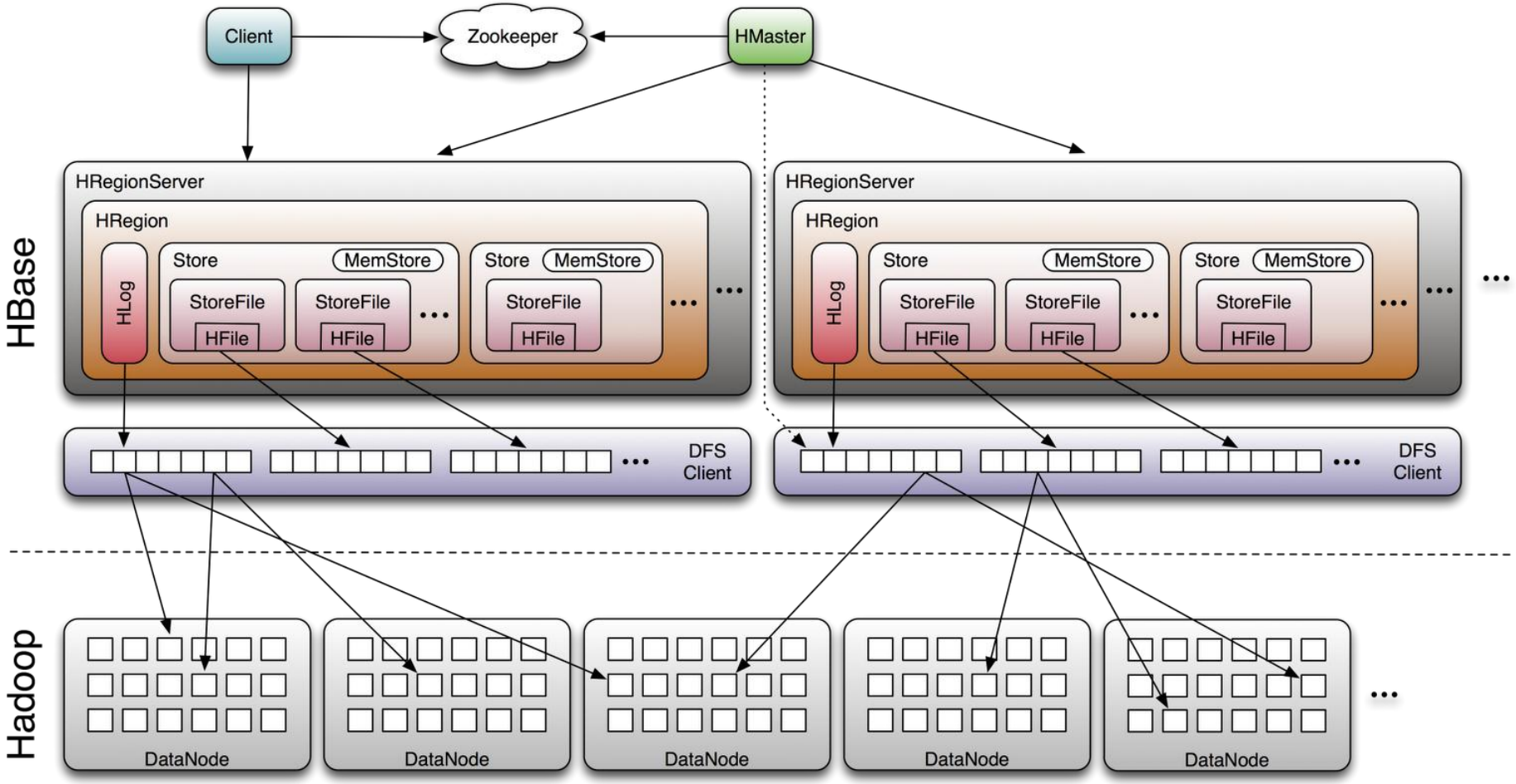
5、Region虽然是**分布式存储**的最小单元，但并不是**存储**的最小单元。

- Region由一个或者多个Store组成，每个store保存一个columns family；
- 每个Strore又由一个memStore和0至多个StoreFile组成；
- memStore存储在内存中，StoreFile存储在HDFS上。



目录

HBase架构

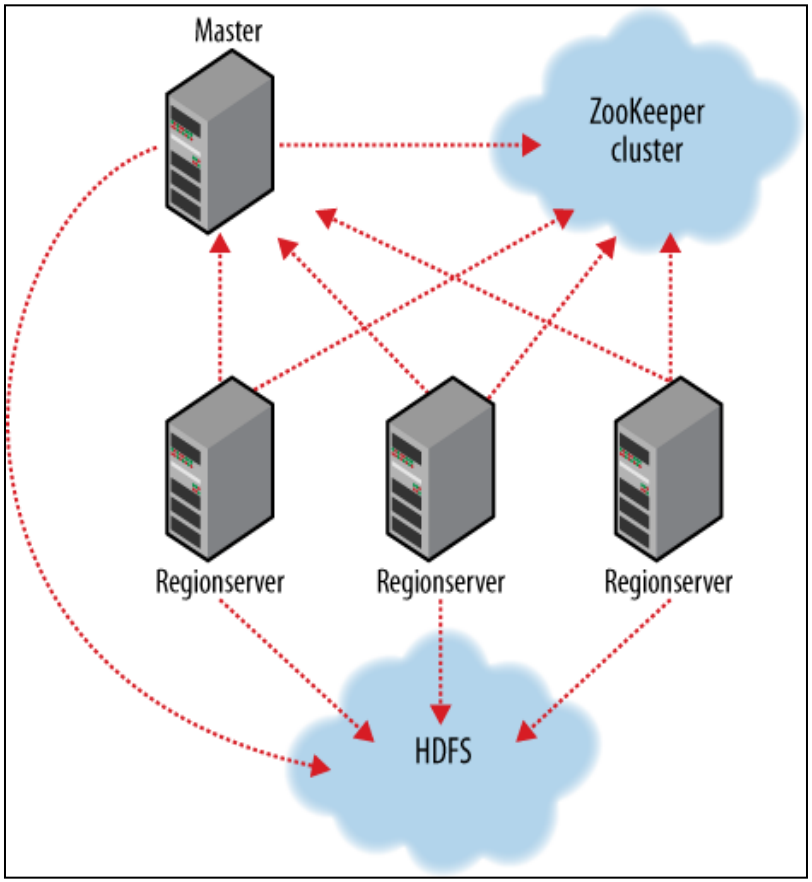

Client
Zookeeper
HMaster
HBase
HRegionServer
HRegion
HLog
Store
MemStore
StoreFile
HFile
DFS Client
Hadoop
DataNode

Hbase基本组件

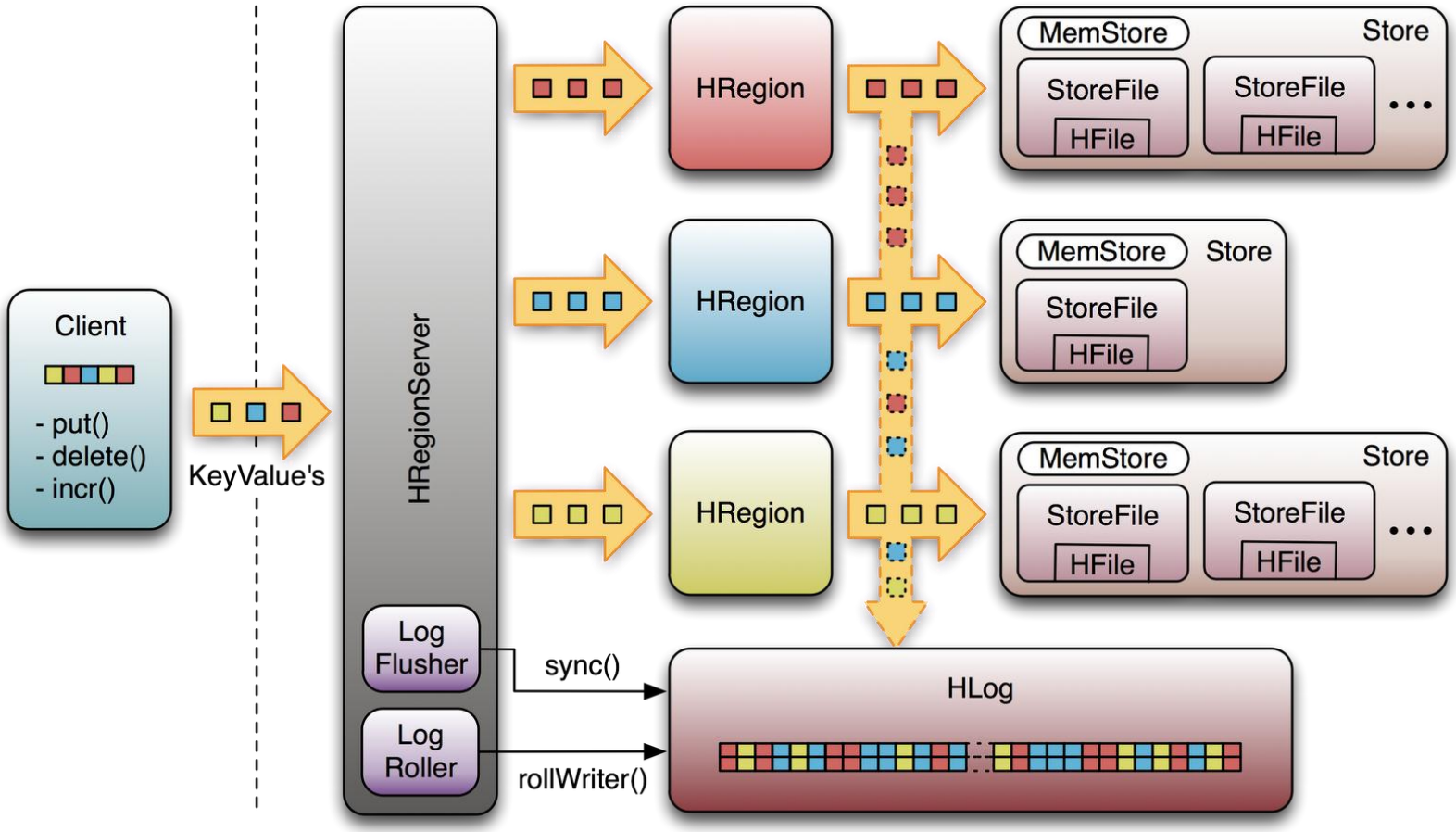
- **Client**
  - ✓包含访问HBase的接口，并维护cache来加快对HBase的访问
- **Zookeeper**
  - ✓保证任何时候，集群中只有一个master
  - ✓存贮所有Region的寻址入口
  - ✓实时监控Region server的上线和下线信息。并实时通知给Master
  - ✓存储HBase的schema和table元数据
- **Master**
  - ✓为Region server分配region
  - ✓负责Region server的负载均衡
  - ✓发现失效的Region server并重新分配其上的region
  - ✓管理用户对table的增删改查操作
- **Region Server**
  - ✓Region server维护region，处理对这些region的IO请求
  - ✓Region server负责切分在运行过程中变得过大的region

Zookeeper作用

- HBase 依赖ZooKeeper
- 默认情况下，HBase 管理ZooKeeper 实例
  - ✓比如， 启动或者停止ZooKeeper
- Master与RegionServers启动时会向ZooKeeper注册
- Zookeeper的引入使得Master不再是单点故障



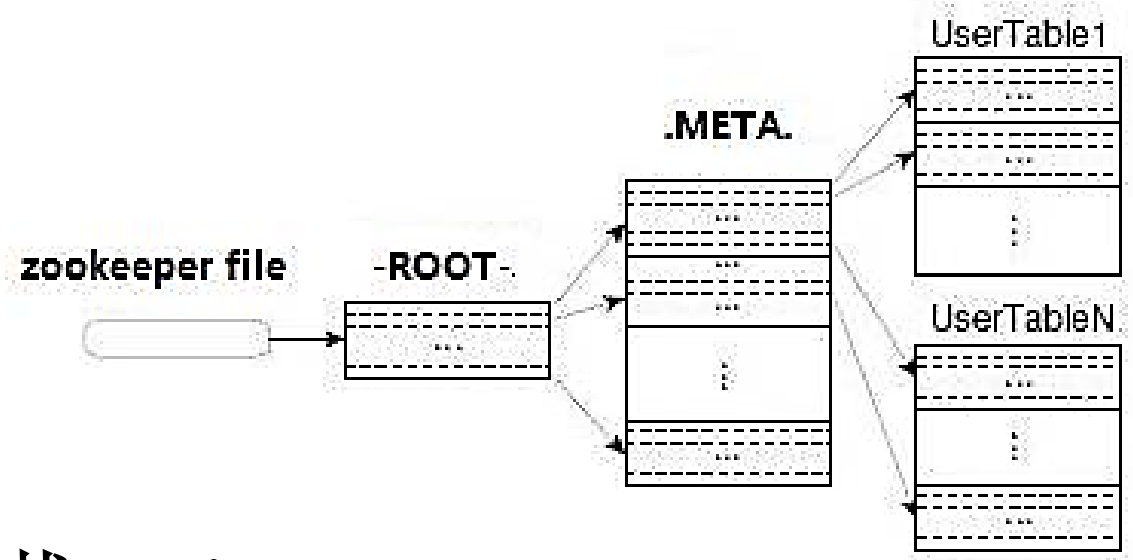
Write-Ahead-Log（WAL）



HBase容错性

- **Master容错：Zookeeper重新选择一个新的Master**
  - ✓无Master过程中，数据读取仍照常进行；
  - ✓无master过程中，region切分、负载均衡等无法进行；
- **RegionServer容错：定时向Zookeeper汇报心跳，如果一旦时间内未出现心跳**
  - ✓Master将该RegionServer上的Region重新分配到其他RegionServer上；
  - ✓失效服务器上“预写”日志由主服务器进行分割并派送给新的RegionServer
- **Zookeeper容错：Zookeeper是一个可靠地服务**
  - ✓一般配置3或5个Zookeeper实例。

Region定位



- **寻找RegionServer**
  - ✓ZooKeeper
  - ✓-ROOT-(单Region)
  - ✓.META.
  - ✓用户表

-ROOT-表与.META.表

- **-ROOT-**
  - ✓表包含.META.表所在的region列表，该表只会有一个Region；
  - ✓Zookeeper中记录了-ROOT-表的location。

- **.META.**
  - ✓表包含所有的用户空间region列表，以及RegionServer的服务器地址。

HDFS与Hbase比较

	Plain HDFS/MR	HBase
Write pattern	Append-only	Random write, bulk incremental
Read pattern	Full table scan, partition table scan	Random read, small range scan, or table scan
Hive (SQL) performance	Very good	4-5x slower
Structured storage	Do-it-yourself / TSV / SequenceFile / Avro / ?	Sparse column-family data model
Max data size	30+ PB	~1PB

关系数据库与Hbase比较

	RDBMS	HBase
Data layout	Row-oriented	Column-family-oriented
Transactions	Multi-row ACID	Single row only
Query language	SQL	get/put/scan/etc *
Security	Authentication/Authorization	Work in progress
Indexes	On arbitrary columns	Row-key only
Max data size	TBs	~1PB
Read/write throughput limits	1000s queries/second	Millions of queries/second

目录

何时使用HBase

- 需对数据进行随机读操作或者随机写操作；
- 大数据上高并发操作，比如每秒对PB级数据进行上千次操作；
- 读写访问均是非常简单的操作。

- storing large amounts of data (100s of TBs)
- need high write throughput
- need efficient random access (key lookups) within large data sets
- need to scale gracefully with data
- for structured and semi-structured data
- don't need full RDMS capabilities (cross row/cross table transactions, joins, etc.)

什么公司在使用HBase

facebook

twitter

mozilla

Adobe

Meetup

TREND MICRO™

NING

Su.pr by StumbleUpon

YAHOO!

Hbase在淘宝的应用

Hbase在淘宝的应用

- 交易历史记录查询系统
  - 百亿行数据表，千亿级二级索引表
  - 每天千万行更新
  - 查询场景简单，检索条件较少
  - 关系型数据库所带来的问题
  - 基于userId + time + id rowkey设计
  - 成本考虑

Hbase在facebook应用—消息系统

- **Facebook创建了Cassandra，最后却弃用Cassandra，使用了HBase；**
- **消息系统（聊天系统、邮件系统等）需求：**
  - ✓一个较小的临时数据集，是经常变化的。
  - ✓一个不断增加的数据集，是很少被访问的。
- **Hbase同时解决了以上两种需求**
- **参照：**

http://www.facebook.com/note.php?note_id=454991608919#

总结

1. HBase概述
2. HBase数据模型
3. HBase物理模型
4. HBase基本架构