分布式列数据库——HBase

贝毅君

beiyj@zju.edu.cn

如何存储数据?

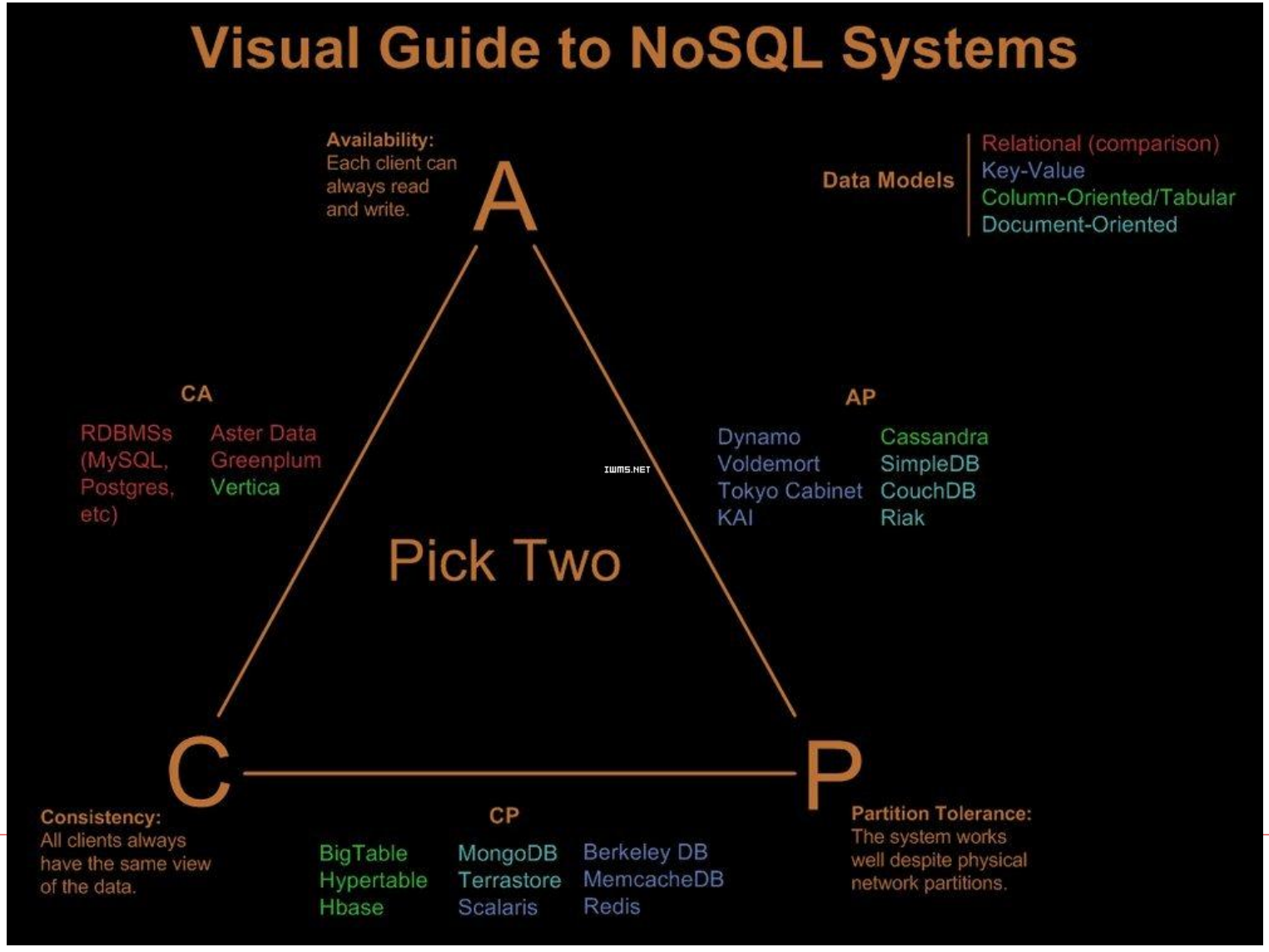
NoSQL

- NoSQL，指的是非关系型的数据库。随着web2.0网站的兴起，特别是超大规模和高并发的纯动态网站，传统关系型数据库显得力不从心，暴露出很多难以克服的问题。
- NoSQL一定程度上是基于一个很重要的原理——CAP原理提出来的。

CAP原理(1)

- CAP概念
  - 一致性(CONSISTENCY)
  - 可用性(AVAILABILITY)
  - 分区容忍性(PARTITION TOLERANCE)
- CAP原理指的是这三个要素最多只能同时实现两点，不可能三者兼顾。
- 传统关系型数据库具有ACID属性，对一致性要求很高，因此降低了A和P。为了提高系统性能和可扩展性，必须牺牲C。

CAP原理(2)


Visual Guide to NoSQL Systems
Availability:
Each client can always read and write.
A
Data Models
Relational (comparison)
Key-Value
Column-Oriented/Tabular
Document-Oriented
CA
RDBMSs (MySQL, Postgres, etc)
Aster Data
Greenplum
Vertica
IWMS.NET
AP
Dynamo
Voldemort
Tokyo Cabinet
KAI
Cassandra
SimpleDB
CouchDB
Riak
Pick Two
C
P
Consistency:
All clients always have the same view of the data.
CP
BigTable
Hypertable
Hbase
MongoDB
Terrastore
Scalaris
Berkeley DB
MemcacheDB
Redis
Partition Tolerance:
The system works well despite physical network partitions.

CAP原理(3)

- 从应用的需求不同，数据库的选择可从三方面考虑：
  - 考虑CA，这就是传统上的关系型数据库(RDBMS)。
  - 考虑CP，主要是一些Key-Value数据库，典型代表为Google的Big Table，将各列数据进行排序存储。数据值按范围分布在多台机器，数据更新操作有严格的一致性保证。
  - 考虑AP，主要是一些面向文档的适用于分布式系统的数据库，如Amazon的Dynamo，Dynamo将数据按key进行Hash存储。其数据分片模型有比较强的容灾性，因此它实现的是相对松散的弱一致性——最终一致性。

NoSQL

- NoSQL优势：
  - 易扩展
  - 大数据量，高性能
  - 灵活的数据模型
  - 高可用

HBase简介

- Big table 的开源版本
- 分布式的、列存储的、高可靠性、高性能的存储系统
- 不同于一般的关系数据库，它是一个适合于非结构化数据存储的数据库。
- 基于列的而不是基于行的模式。

HBase发展

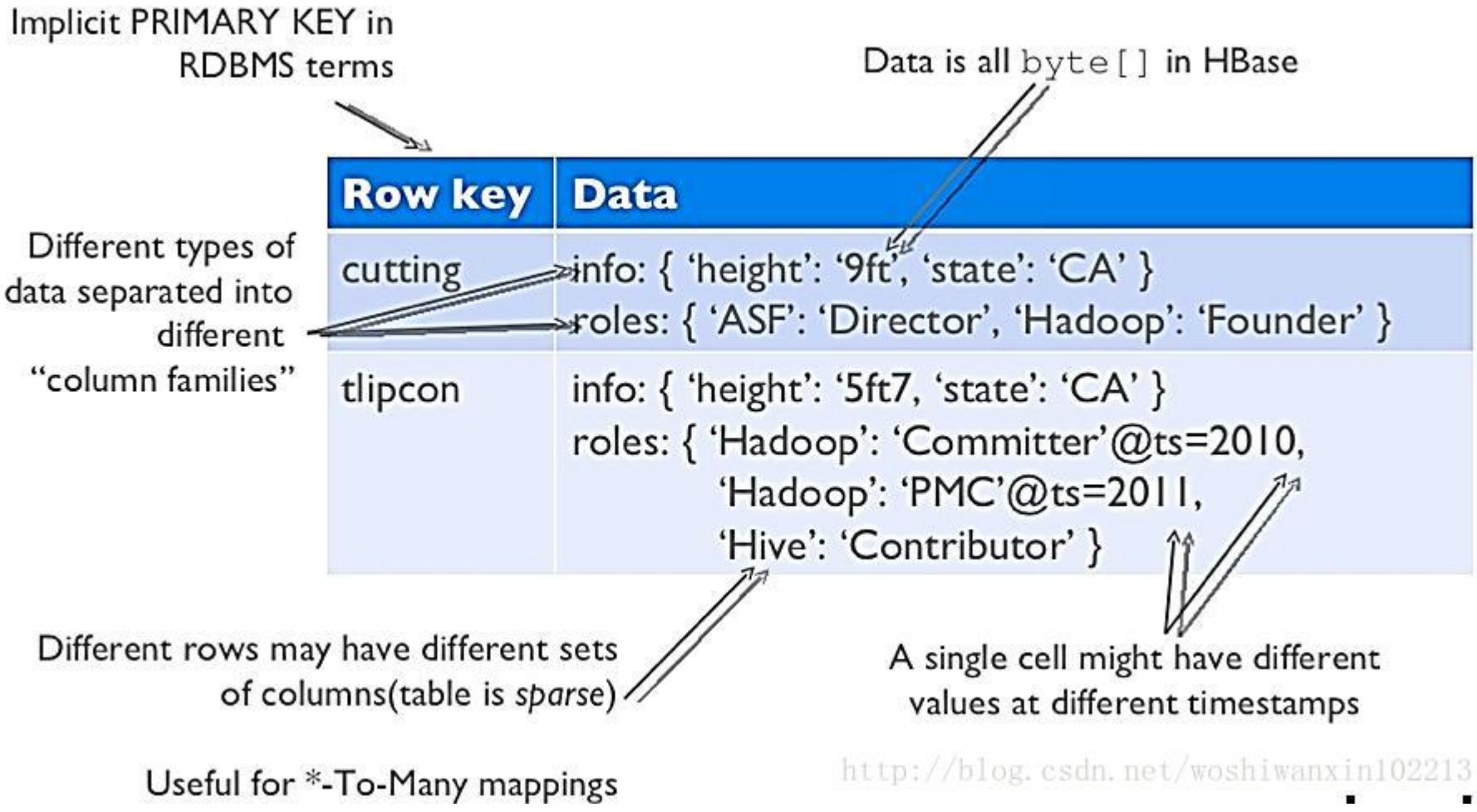
- 2009年9月发布的0.20.0版本（里程碑），online应用正式成为了HBase的目标
- 2011年1月0.90.0版本（facebook/ebay/yahoo内所使用于生产的HBase都是基于这一个版本）

HBase总体特点

- HBase – Hadoop Database，是一个分布式存储系统，可在廉价PC Server上搭建起具有高可靠性、高性能、面向列、可伸缩的大规模结构化存储集群，
- 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可以直接使用本地文件系统或者HDFS作为数据存储方式

HBase具体特点

- 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：空（null）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；



HBase Table Concept View

Table 4. Table webtable

Row Key	Time Stamp	ColumnFamily contents	ColumnFamily anchor	ColumnFamily people
"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>…"		
"com.example.www"	t5	contents:html = "<html>…"		people:author = "John Doe"

HBase Table Concept View


```
{
  "com.cnn.www": {
    contents: {
      t6: contents:html: "<html>..."
      t5: contents:html: "<html>..."
      t3: contents:html: "<html>..."
    }
    anchor: {
      t9: anchor:cnnsi.com = "CNN"
      t8: anchor:my.look.ca = "CNN.com"
    }
    people: {}
  }
  "com.example.www": {
    contents: {
      t5: contents:html: "<html>..."
    }
    anchor: {}
    people: {
      t5: people:author: "John Doe"
    }
  }
}
```

HBase Table Concept View


```
{
  "com.cnn.www": {
    contents: {
      t6: contents:html: "<html>..."
      t5: contents:html: "<html>..."
      t3: contents:html: "<html>..."
    }
    anchor: {
      t9: anchor:cnnsi.com = "CNN"
      t8: anchor:my.look.ca = "CNN.com"
    }
    people: {}
  }
  "com.example.www": {
    contents: {
      t5: contents:html: "<html>..."
    }
    anchor: {}
    people: {
      t5: people:author: "John Doe"
    }
  }
}
```

HBase Table Physical View

Table 5. 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"

Table 6. 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>…"

列存储的好处

- 降低IO
  - 面向列的存储系统可以单独查询某一列，从而大大降低IO
- 提高压缩效率
  - 同列数据具有很高的相似性，会增加压缩效率
- Hbase的很多特性都是由列存储决定的

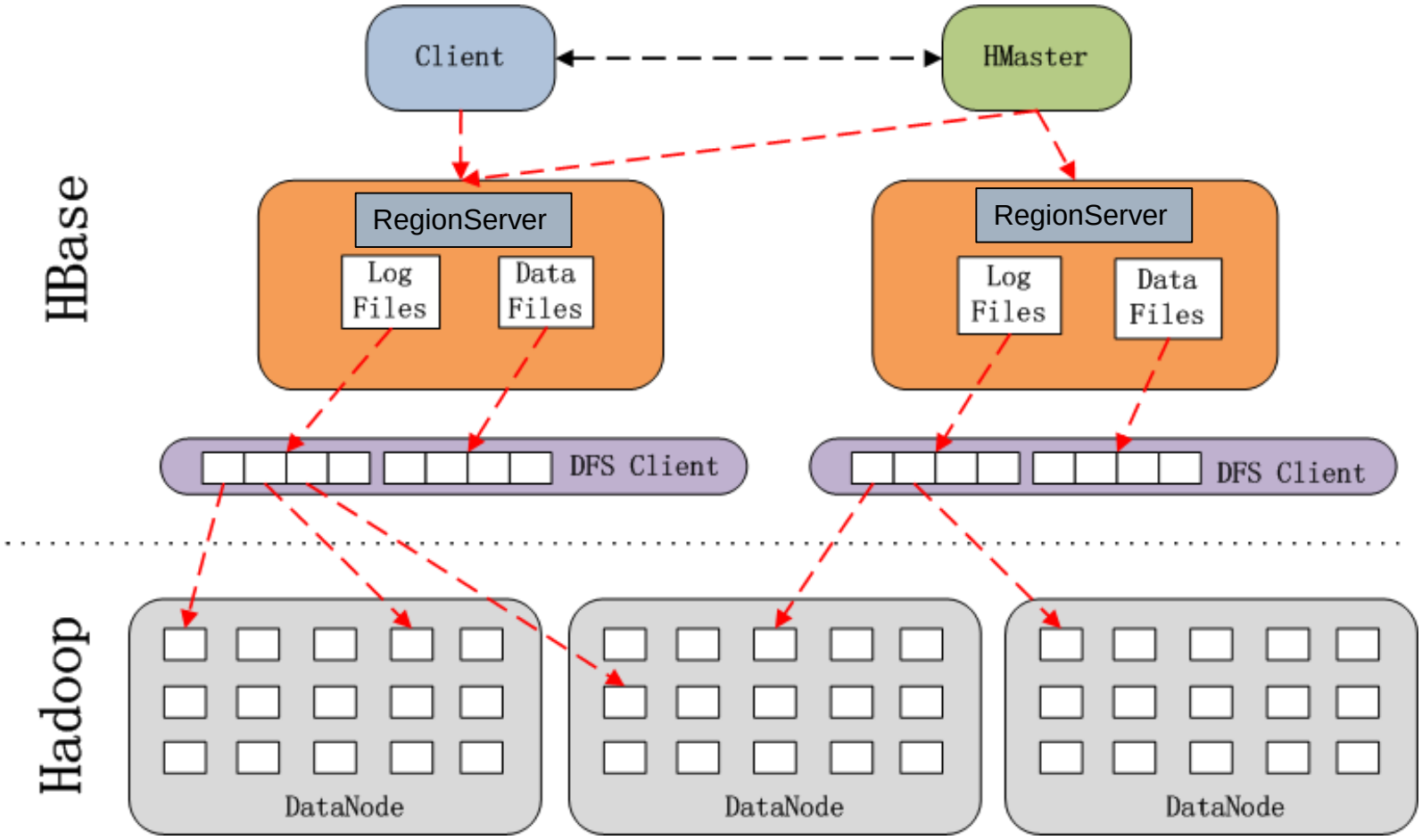
适合HBase的

- 半结构化或非结构化数据，对于数据结构字段不够确定或杂乱无章很难按一个概念去进行抽取的数据，适合用HBase。
- HBase支持动态增加，适应添加字段。
- 大数据量，高并发读写。
- 需要历史记录。
- 读 > 写

HBase使用情况

- facebook，ebay，taobao，百度等大量公司都在使用。

HBase系统架构


Client
HMaster
HBase
RegionServer
Log Files
Data Files
RegionServer
Log Files
Data Files
DFS Client
DFS Client
Hadoop
DataNode
DataNode
DataNode

HBase系统构架详解（1）

- Client
  - 使用HBase的RPC机制与HMaster和HRegionServer进行通信
  - 管理类操作，Client与HMaster进行RPC
  - 数据读写类操作，Client与HRegionServer进行RPC
- ZooKeeper
  - Zookeeper Quorum中存储了-ROOT-表的地址和HMaster的地址
  - HRegionServer将自己以短暂方式注册到ZooKeeper中，使得HMaster可以随时感知到各HRegionServer的健康状态
  - ZooKeeper可避免了HMaster的单点问题

HBase系统构架详解（2）

- 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Master运行，HMaster在功能上主要负责Table和Region的管理工作：
    - 管理用户对Table的增、删、改、查操作
    - 管理HRegionServer的负载均衡，调整Region分布
    - 在Region Split后，负责新Region的分配
    - 在HRegionServer停机后，负责失效HRegionServer 上的Regions迁移

HBase基本概念（1）

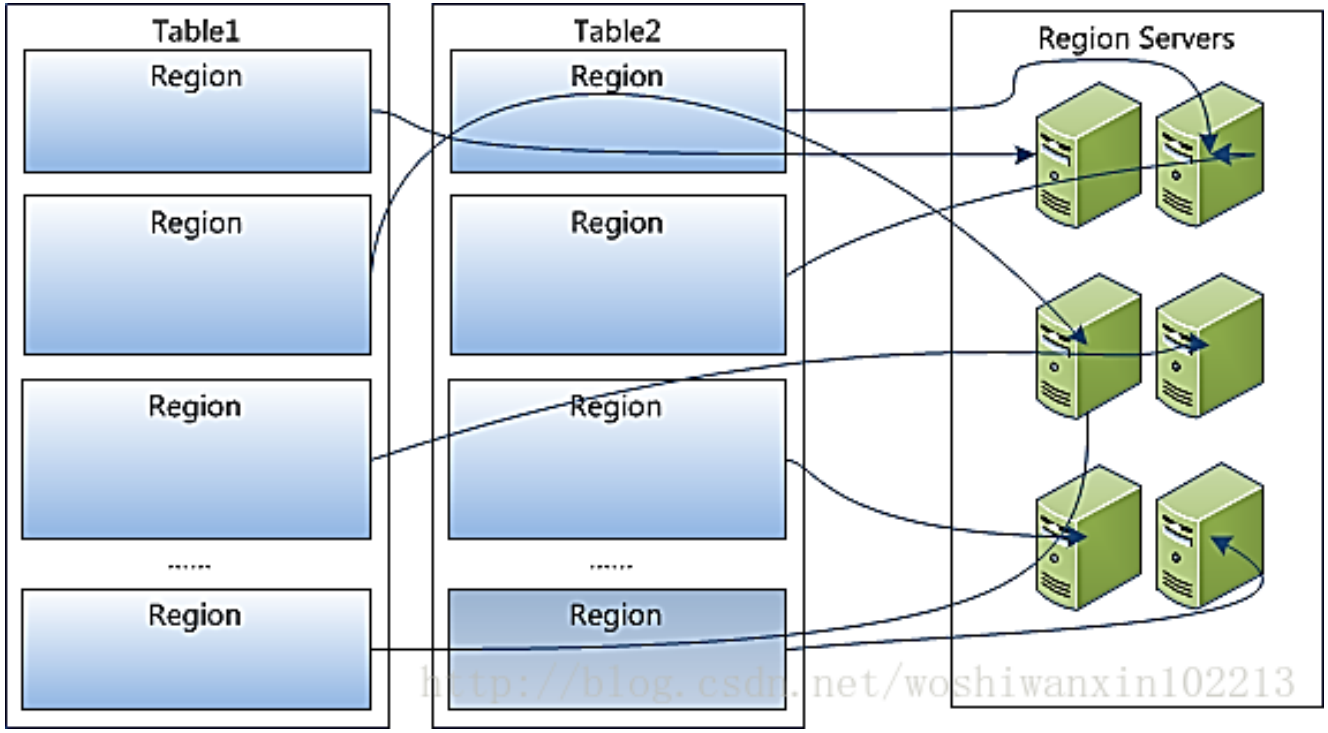
- Row Key（行键）
  - Table的主键，Table中的记录按照Row Key排序
  - 可以是任意字符串(最大长度是 64KB，实际应用中长度一般为 10-100bytes)，在hbase内部，row key保存为字节数组
- Timestamp（时间戳）：
  - HBase中通过row和column确定的为一个存储单元称为cell
  - 每次数据操作对应的时间戳，可看作是数据的版本号
  - 保存数据的最后n个版本或最近一段时间内的版本

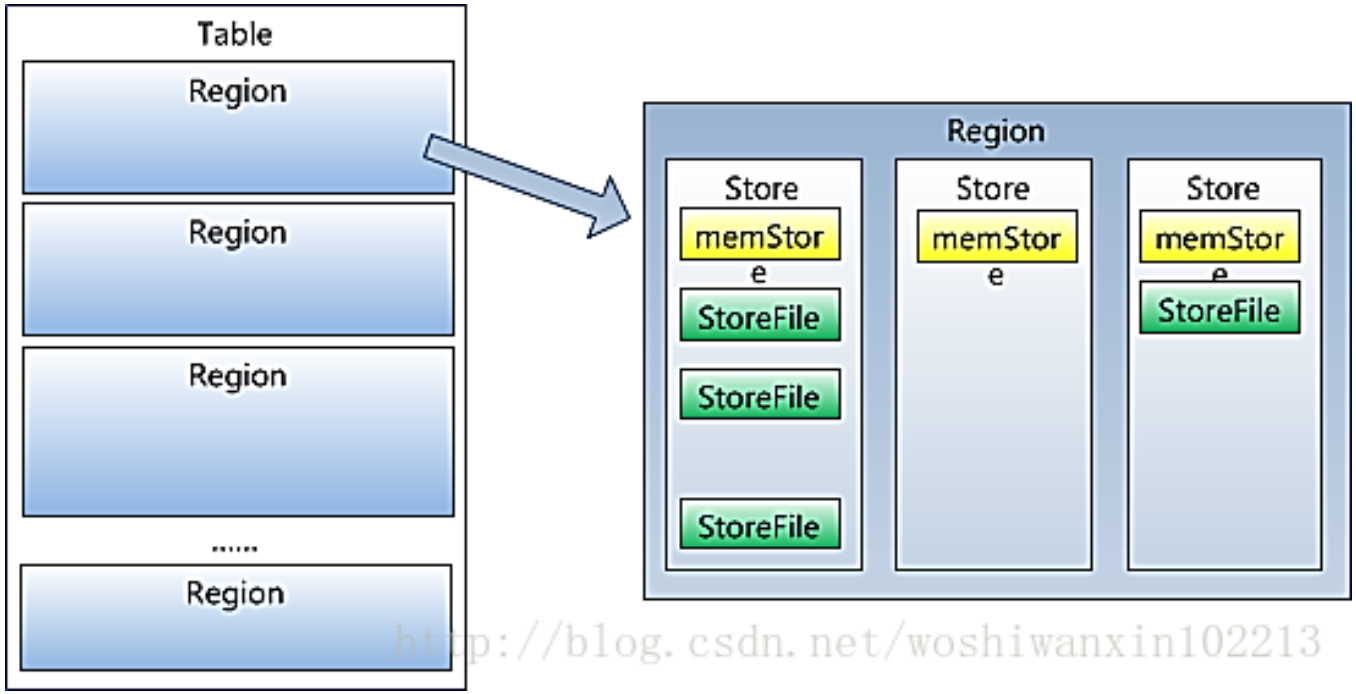
HBase基本概念（2）

- Column Family（列簇）
  - Table在水平方向由一个或者多个Column Family组成
  - 必须在使用表之前定义
  - 一个Column Family中可以由任意多个Column组成，即Column Family支持动态扩展，无需预先定义。
  - 所有Column均以二进制格式存储，用户需要自行进行类型转换。
- Cell
  - 由{row key, column(=<family> + <label>), version} 唯一确定的单元。cell中的数据是没有类型的，全部是字节码形式存储。

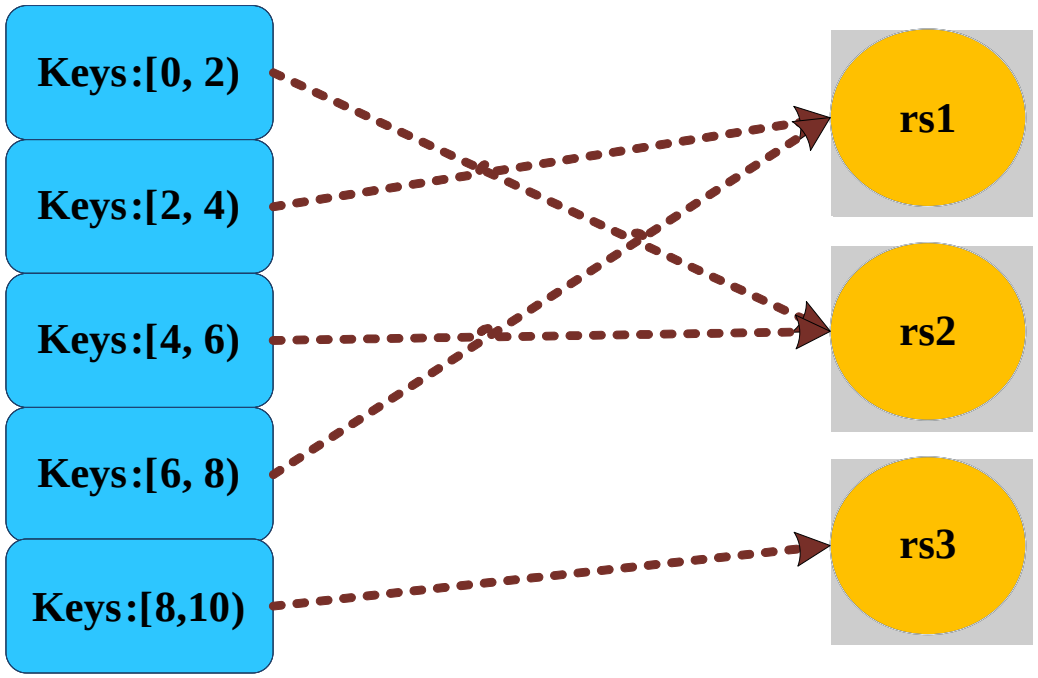
HBase基本概念（3）

- Table（表）和Region（区域）
  - 当Table随着记录数不断增加而变大后，会逐渐自动分裂成多份splits，成为regions
  - 一个region由[startkey,endkey)表示
  - 不同region会被Master分配给相应的RegionServer进行管理





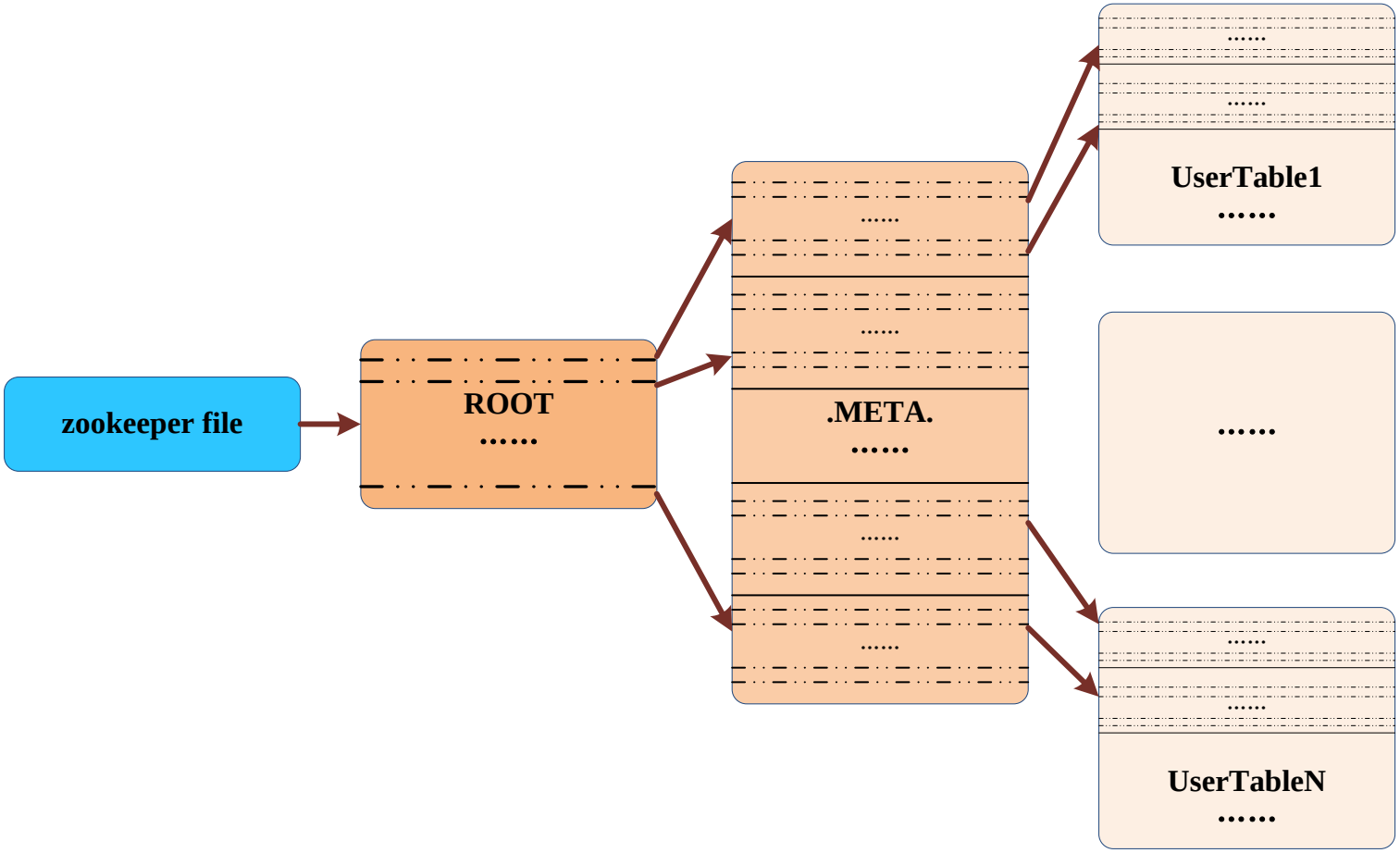
Table&Region示意图



HBase中特殊表

- HBase中有两张特殊的Table，-ROOT-和.META.
- ZooKeeper中记录了-ROOT-表的location
- -ROOT-：记录了.META.表的Region信息，-ROOT-只有一个region
- .META.：记录了用户表的Region信息，.META.可以有多个regoin
- 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
  - Client端会做cache缓存

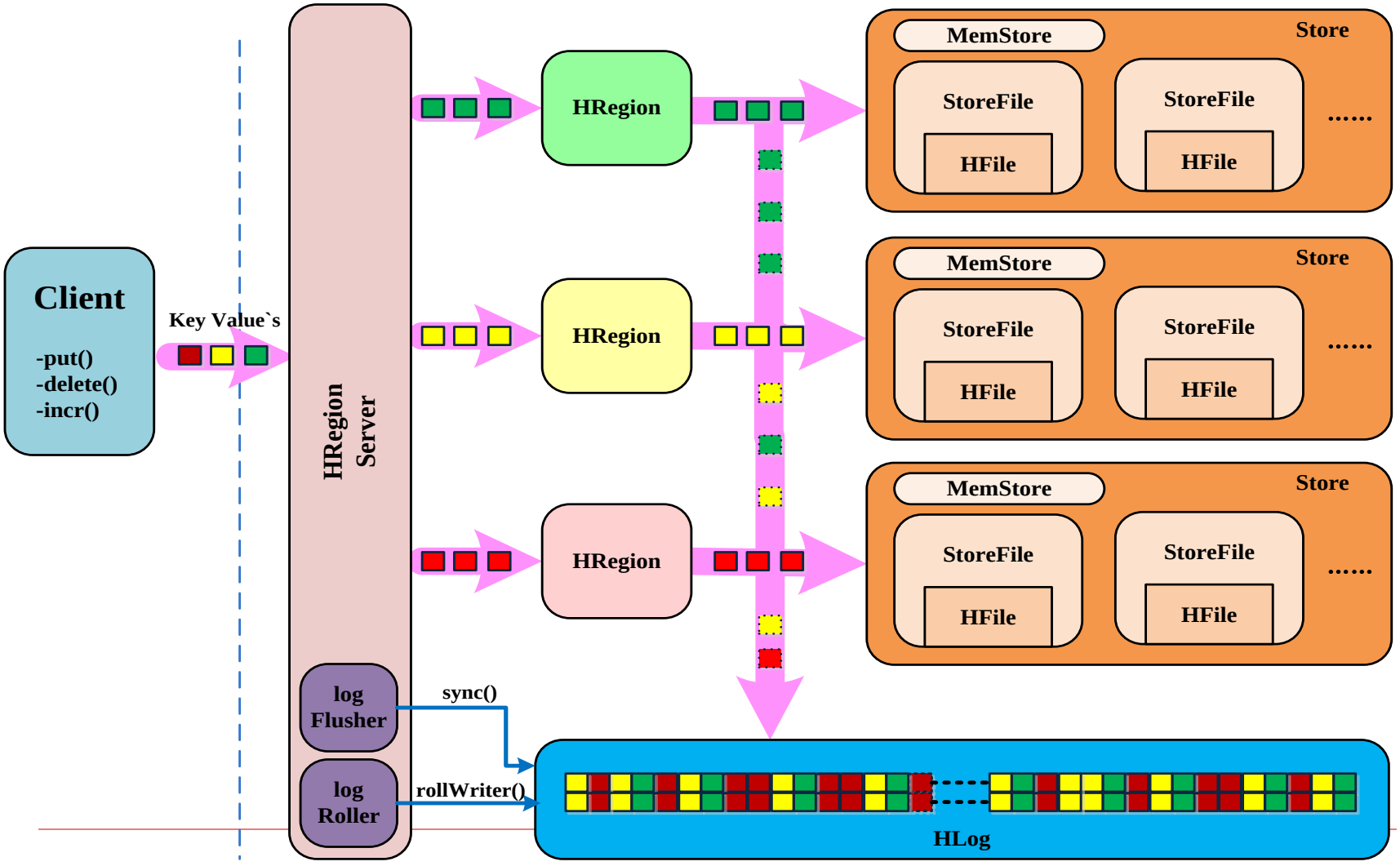
HBase中特殊表



HBase基本概念（4）

- RegionServer
  - Region读写操作的场所；
- Master
  - 管理Region的分配；
  - 基于zookeeper来保证高可用性HA；

HRegionServer架构



HRegionServer架构分析

- HRegionServer由多个HRegion组成
- 每个HRegion对应了Table中的一个Region
- HRegion由多个HStore组成
- 每个HStore对应了Table中的一个Column Family的存储
  - 每个Column Family其实就是一个集中的存储单元
  - 最好将具备共同IO特性的column放在一个Column Family中，这样最高效

HRegionServer架构——HStore

- HStore存储是HBase存储的核心
  - MemStore：是Sorted Memory Buffer，用户写入的数据首先会缓存至MemStore
  - StoreFile：当MemStore满了以后会Flush成一个StoreFile（底层实现是HFile）
- 当StoreFile文件数量增长到一定阈值，会触发Compact合并操作，将多个StoreFiles合并成一个StoreFile，合并过程中会进行版本合并和数据删除
- HBase只有增加数据，所有的更新和删除操作都是在后续的compact过程中进行的，这使得用户的写操作只要进入内存中就可以立即返回，保证了HBase I/O的高性能

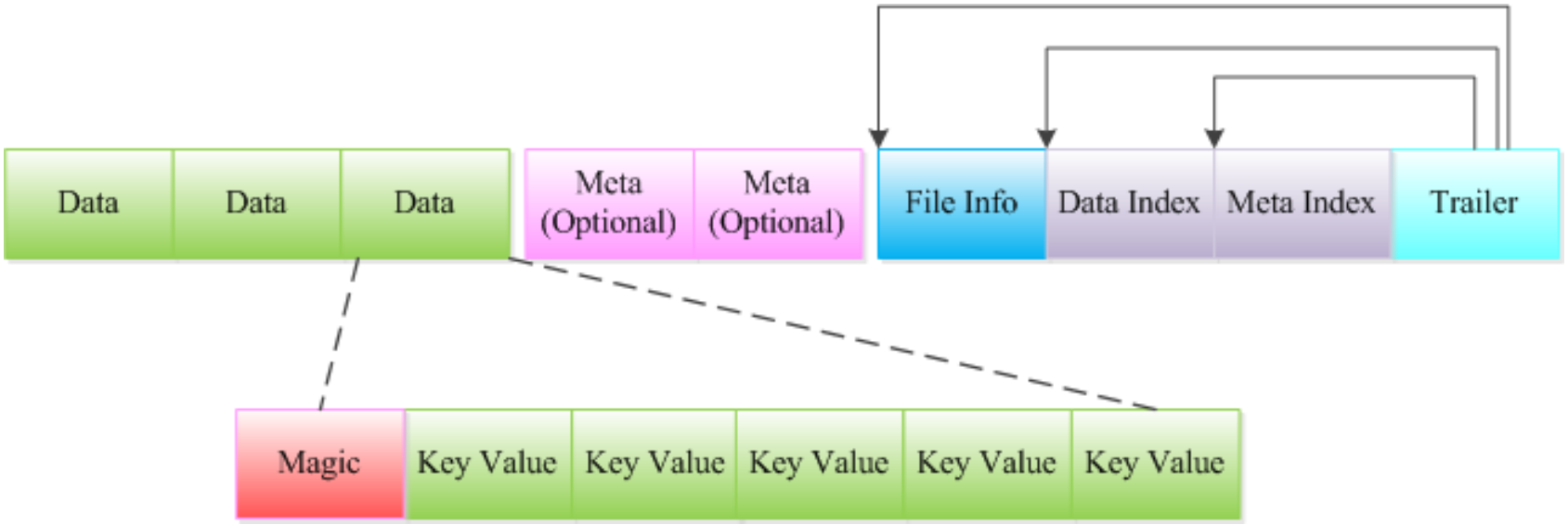
HRegionServer架构——HStore

- 当StoreFiles Compact后，会逐步形成越来越大的StoreFile，当单个StoreFile大小超过一定阈值后，会触发Split操作，同时把当前 Region Split成2个Region
- 父Region会下线，新Split出的2个孩子Region会被HMaster分配到相应的HRegionServer 上，使得原先1个Region的压力得以分流到2个Region上

HRegionServer架构——HLog

- HRegionServer中都有一个HLog对象
- HLog是一个实现Write Ahead Log的类，在每次用户操作写入MemStore的同时，也会写一份数据到HLog文件
- HLog文件定期会滚动出新的，并删除旧的文件（已持久化到StoreFile中的数据）。
- 当HRegionServer意外终止后，HMaster会通过Zookeeper感知到
  - HMaster处理遗留的 HLog文件，将其中不同Region的Log数据进行拆分，分别放到相应region的目录下，将失效的region重新分配
  - 领取到这些region的HRegionServer在Load Region的过程中，会发现有历史HLog需要处理，重放HLog数据到MemStore中，然后flush到StoreFiles，完成数据恢复

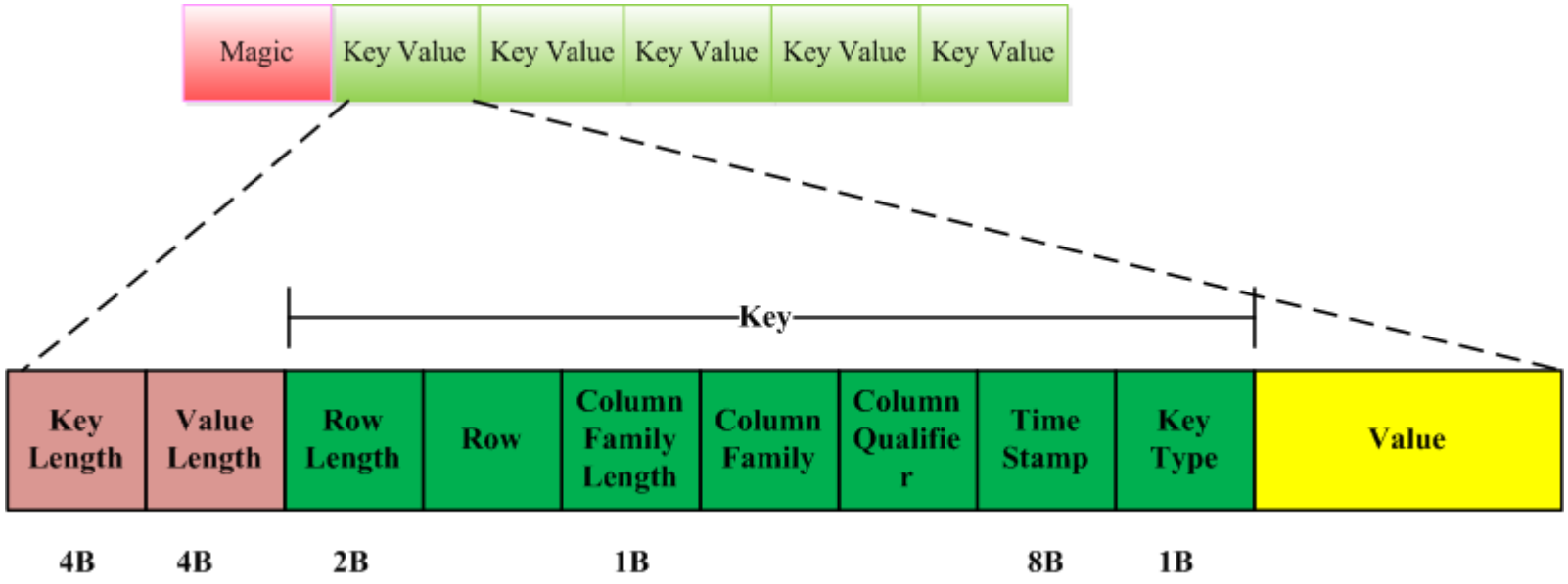
HBase存储格式——HFile



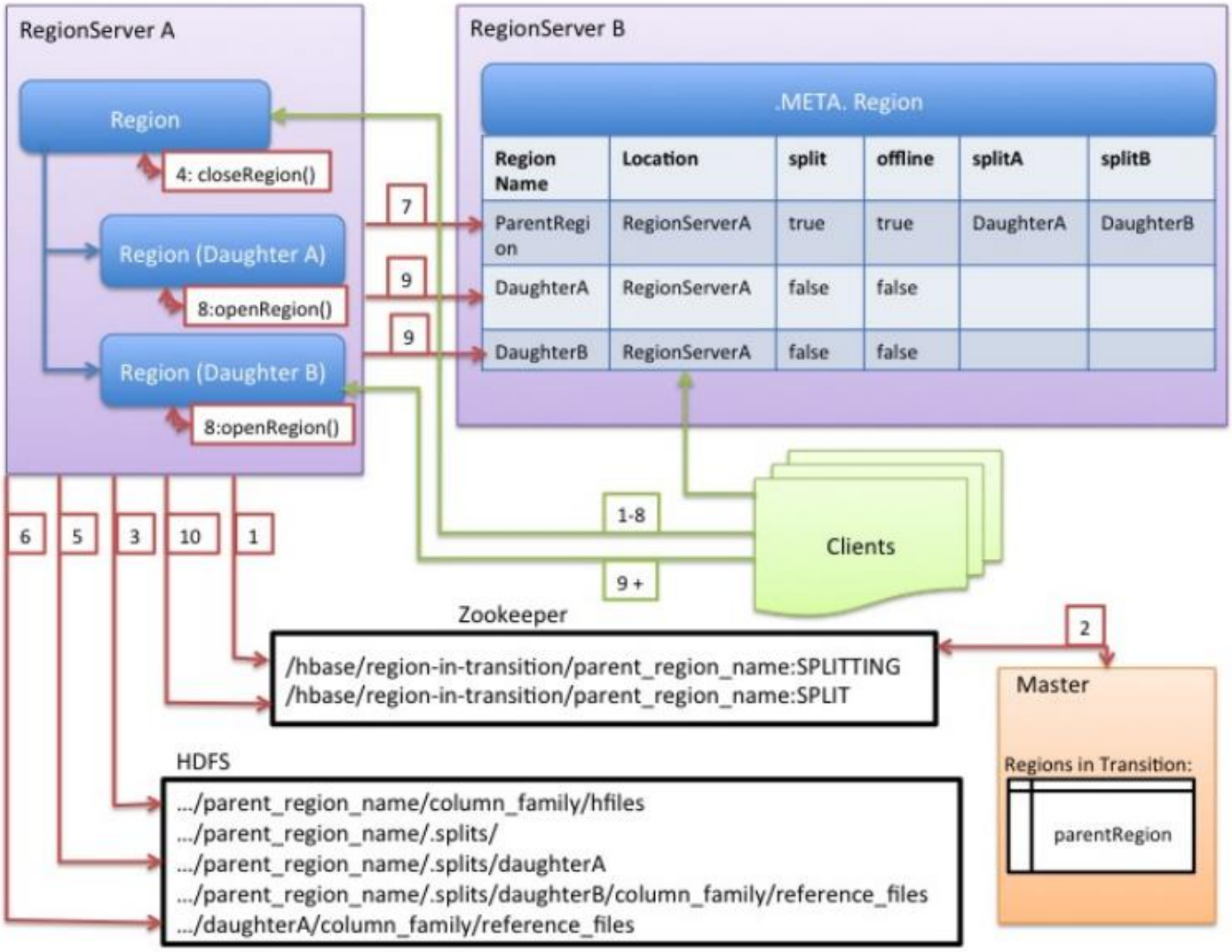
HBase存储格式

- 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StoreFile就是对HFile做了轻量级包装

HBase存储格式——Data



HBase Region Split过程



Hbase容错

- **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；
- **Zookeeper容错**：Zookeeper是一个可靠地服务，一般配置3或5个Zookeeper实例

Hbase容错

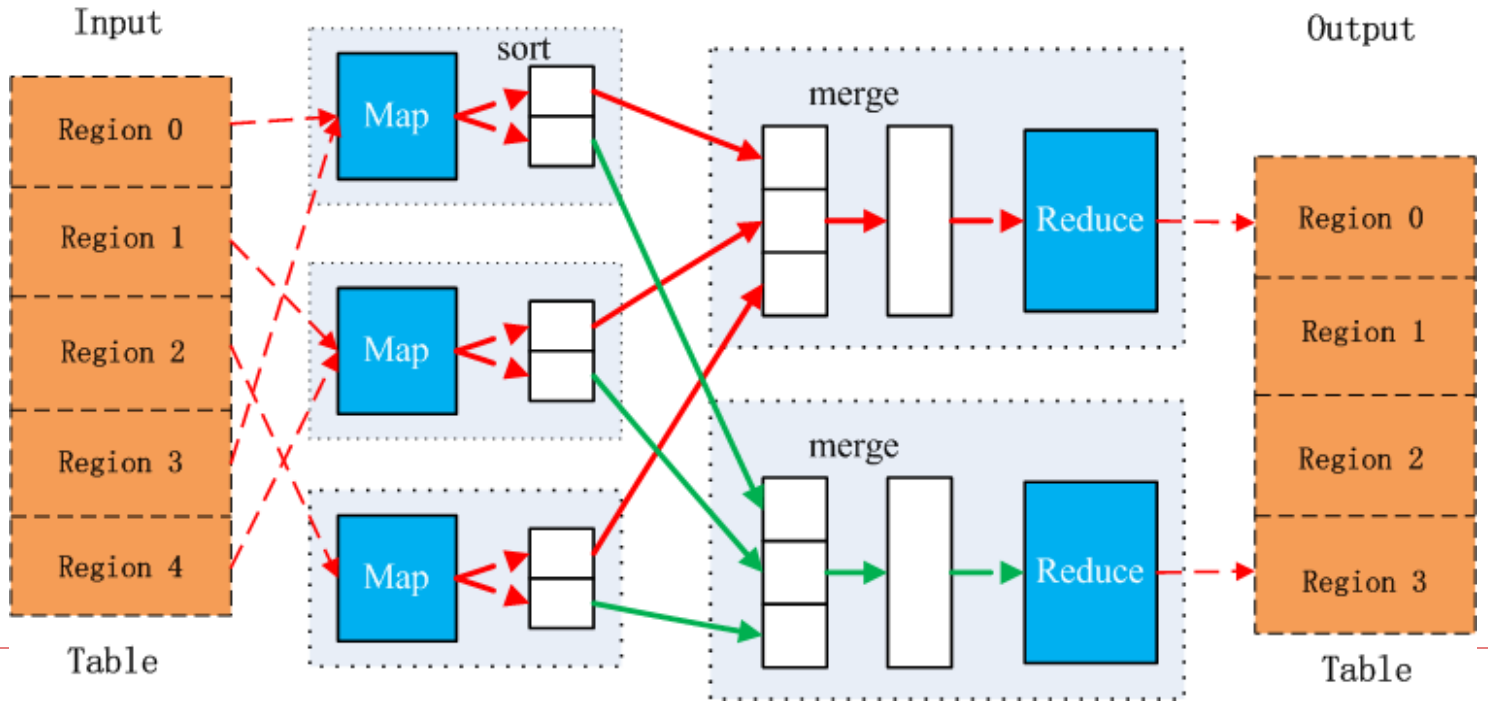
- **RegionServer容错**：定时向Zookeeper汇报心跳，如果一旦时间内未出现心跳，Master将该RegionServer上的Region重新分配到其他RegionServer上，失效服务器上WAL日志由主服务器进行分割并派送给新的RegionServer

HBase存储格式

- Data Block
  - Data Block是HBase I/O的基本单元
  - 每个Data块除了开头的Magic以外就是一个个KeyValue对拼接而成
  - Magic内容就是一些随机数字，目的是防止数据损坏

MapReduce on HBase

- HBase Table和Region的关系，类似于HDFS File和Block的关系，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



HBase特性(1)

- 强一致性
  - 同一行数据的读写只在同一台region server上进行；
- 水平伸缩
  - region的自动分裂以及master管理整个系统的平衡性；
  - 只用增加data node机器即可增加容量；
  - 只用增加region server机器即可增加读写吞吐量；

HBase特性(2)

- 行事务
  - 同一行的列写入是原子的；
- Column Oriented + 三维有序
  - SortedMap(RowKey, List(SortedMap(Column,List(Value,Timestamp))))
  - rowKey (ASC) + columnLabel(ASC) + Version (DESC) --> value

- rowkey，以字典顺序排序的。
- column key是第二维，数据按rowkey字典排序后，如果rowkey相同，则是根据column key来排序，也是按字典排序。
  - 在设计table的时候要学会利用这一点。比如收件箱，有时需要按主题排序，那就可以把主题这设置为column key，即设计为columnFamily+主题。
- timestamp 时间戳，是第三维，这是个按降序排序的，即最新的数据排在最前面。

HBase特性(3)

- 支持范围查询
  - Scan scan=new Scan(Bytes.toBytes("0"), Bytes.toBytes("20"));
- 一个表可以有上亿行，稀疏存储
- 高性能随机写
  - WAL (Write Ahead Log)

HBase特性(4)

- 和Hadoop无缝集成
  - Hadoop分析后的结果可直接写入HBase；
  - 存放在HBase的数据可直接通过Hadoop来进行分析。
- 支持多种压缩算法
  - GZIP, LZO, Snappy

HBase使用

- 访问hbase table中的行，有三种方法
  - 通过单个 row key 访问
  - 通过 row key 的 range
  - 全表扫描
- 不要创建过多的Column Family
- 根据应用场景合理发挥三维有序存储特性

HBase案例 - blog

- 一篇Blog entry由title，author，type，text组成
- 一位User由username，password组成
- 每篇Blog entry可有许多Comments
- 每一则Comment由title，author，text组成

HBase案例 - blog

BLOGENTRY	
PK	id_blogentry
FK1	title under_title id_user date text type

USER	
PK	id_user
FK1	name login password

COMMENT	
PK	id_comment
FK1	id_blogentry date text title author

ERD

HBase案例 - blog

Table	Row Key	Family	Attributs
blogtable	TTYYYYMMDDHHmmss	info:	Always contains the column keys author,title,under_title. Should be IN-MEMORY and 1 version
		text:	No column key. 3 version
		comment_title	Column keys are written like YYYYMMDDHHmmss. Should be IN-MEMORY and 1 version
		comment_author	Same keys. 1 version
		comment_text	Same keys. 1 version
usertable	login_name	info:	Always contains the column keys password and name. 1 version

HBase案例 - blog

- BLOGENTRY 与 COMMENT的”一对多”关系由comment_title, comment_author, comment_text 等column families 内的动态数量的column来表示。
- 每个Column的名称是由每则 comment的timestamp来表示，因此每个column family的 column 会按时间自动排序。

HBase in 淘宝(1)

- 淘宝在2011年之前所有的后端持久化存储基本上都是在mysql上进行的
- mysql由于开源，并且生态系统良好，本身拥有分库分表等多种解决方案，因此很长一段时间内都满足淘宝大量业务的需求。

HBase in 淘宝(2)

- 但是由于业务的多样化发展，有越来越多的业务系统的需求开始发生了变化。
  - 需要有一个海量分布式文件系统，能对TB级甚至PB级别的数据提供在线服务
  - 对系统水平扩展能力有比较强烈的需求，且不希望存在单点制约
  - 只需要简单的kv读取，没有复杂的join等需求。但对系统的并发能力以及吞吐量、响应延时有非常高的需求，并且希望系统能够保持强一致性

HBase in 淘宝(3)

- 通常系统的写入非常频繁，尤其是大量系统依赖于实时的日志分析
- 希望能够快速读取批量数据
- schema灵活多变，可能经常更新列属性或新增列
- 希望能够方便使用，有良好且语义清晰的java接口

- 以上需求综合在一起，淘宝认为HBase是一种比较适合的选择。
- 从2011年3月开始研究HBase如何用于在线服务。

HBase in 淘宝(4)

- 第一个上线的应用是数据魔方中的prom。
- 第二个上线的应用是TimeTunnel，一个高效的、可靠的、可扩展的实时数据传输平台，广泛应用于实时日志收集、数据实时监控、广告效果实时反馈、数据库实时同步等领域。

HBase in 淘宝(5)

- 随着业务的发展，目前定制的HBase集群已经应用到了线上超过二十个应用，数百台服务器上。包括淘宝首页的商品实时推荐、广泛用于卖家的实时量子统计等应用，并且还有继续增多以及向核心应用靠近的趋势。
- 未来计划除修复版本bug、优化hdfs层面外，做实时监控和调整regionserver的负载。

谢谢！