HBase基本原理与程序设计

摘要

- HBase基本工作原理
- HBase基本操作
- HBase编程方法示例

摘要

- HBase基本工作原理
- HBase基本操作
- HBase编程方法示例

CAP定理

- 在计算机科学中，CAP定理（CAP theorem），又被称作布鲁尔定理（Brewer‘s theorem），它指出对于一个分布式计算系统来说，不可能同时满足以下三点：
  - **一致性(Consistency)**：所有节点在同一时间具有相同的数据
  - **可用性(Availability)**：保证每个请求不管成功或者失败都有响应
  - **分隔容忍(Partition tolerance)**：系统中任意信息的丢失或失败不会影响系统的继续运作
- CAP理论的核心是：一个分布式系统不可能同时很好的满足一致性，可用性和分区容错性这三个需求，最多只能同时较好的满足两个。

关系数据库

- 关系数据库的理论局限性
  - RDBMS选择ACID（CAP定理中的C），然后是A
    - 网络分片(Network Partitions)在分布式系统中不可避免
    - 系统扩展时性能和可靠性下降
  - Scale up, not out
    - 并行数据库的扩展性
      - 经验定律：当集群结点数每增加4-16台，每个节点的效率下降一半
    - 无法扩展超过40节点
  - “One size does not fit all.”
    - 在所有数据库的主要应用领域，新的架构轻易地有10x倍性能提升（数据仓库、流处理、科学计算、非结构化数据处理、OLTP在线事务处理）

关系数据库

- 科德(Edgar F. Codd)十二定律
  - 使数据库管理系统关系化需满足的十三条准则（0-12），又称黄金十二定律。
- 标准的RDBMS是模式固定、面向行的数据库且具有ACID性质和复杂的SQL查询处理引擎。
- RDBMS强调事务的“强一致性”、参照完整性、数据抽象与物理存储相对独立，以及基于SQL语言的复杂查询支持。
- RDBMS可以非常容易建立二级索引，执行复杂的内连接和外连接，执行计数、求和、排序、分组等操作，或对表、行和列中的数据进行分页存放。

关系数据库

- RDBMS的实现局限性
  - RDBMS实现和操作上的局限性--不适合新的应用
    - 大表 -- 在一张表中存储500GB的数据?
    - 灵活动态可变的表结构 -- 为大表修改表结构(Alter Table)?
    - 无停机时间的在线大表分区和动态扩容

NoSQL数据库

- NoSQL = Not Only SQL：不仅仅是SQL
- 非关系型数据库
- 可用于超大规模数据的存储，这些类型的数据存储不需要固定的模式，无需多余操作就可以横向扩展。

RDBMS vs. NoSQL

- RDBMS
  - 高度组织化结构化数据
  - 结构化查询语言SQL
  - 数据和关系存储在单独的表中
  - 数据操纵语言，数据定义语言
  - 严格的一致性
  - 基础事务
- NoSQL
  - 代表着不仅仅是SQL
  - 没有声明性查询语言
  - 没有预定义的模式
  - 键-值对存储，列存储，文档存储，图形数据库
  - 最终一致性，而非ACID属性
  - 非结构化和不可预知的数据
  - CAP定理
  - 高性能、高可用和可伸缩性

NoSQL

- 优点
  - 高可扩展
  - 分布式计算
  - 低成本
  - 架构灵活
  - 半结构化数据
  - 没有复杂的关系
- 缺点
  - 没有标准化
  - 有限的查询功能
  - 最终一致是不直观的程序

BASE

- BASE： Basically Available, Soft-state, Eventually Consistent。 由 Eric Brewer 定义。
- BASE是NoSQL数据库通常对可用性及一致性的弱要求原则：
  - Basically Available --基本可用
  - Soft-state --软状态/柔性事务。 “Soft state” 可以理解为“无连接”的，而 “Hard state” 是“面向连接”的。
  - Eventual Consistency --最终一致性。最终一致性， 也是 ACID 的最终目的。

ACID vs. BASE

- ACID
  - 原子性(Atomicity)
  - 一致性(Consistency)
  - 隔离性(Isolation)
  - 持久性(Durable)
- BASE
  - 基本可用(Basic Available)
  - 软状态/柔性事务(Soft state)
  - 最终一致性(Eventual consistency)

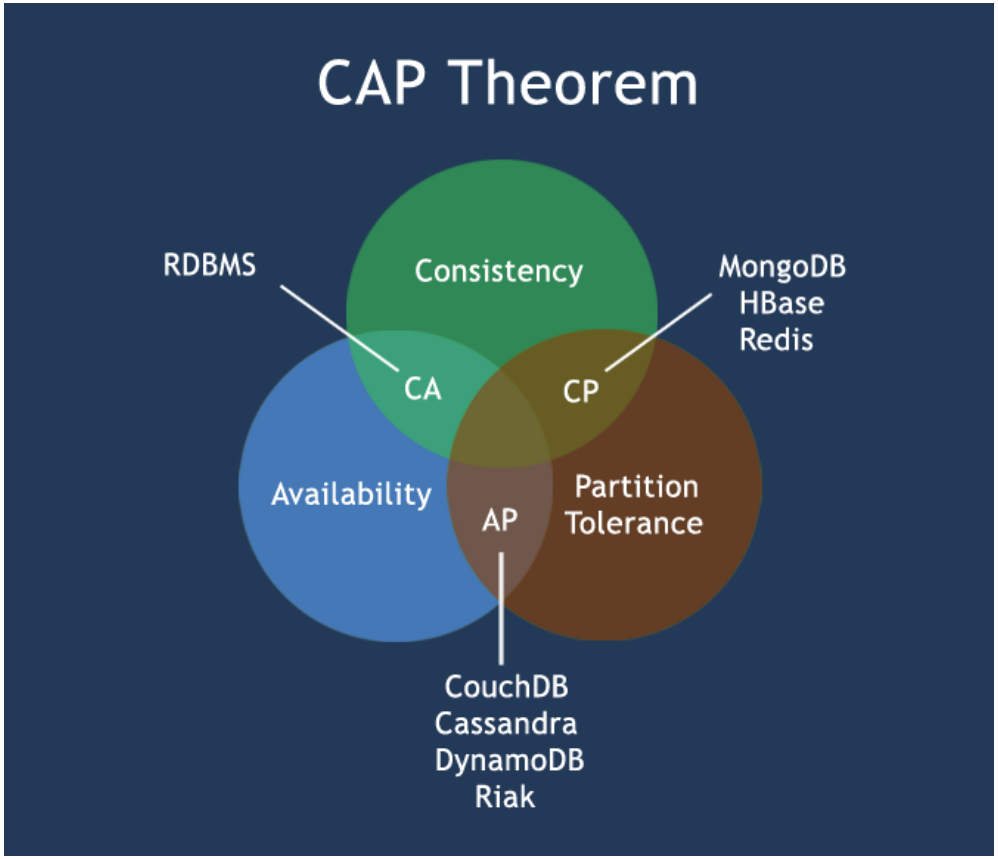
NoSQL

- 常见的NoSQL数据库：

类型	部分代表	特点
列存储	HBase Cassandra Hypertable	顾名思义，是按列存储数据的。最大的特点是方便存储结构化和半结构化数据，方便做数据压缩，对针对某一列或者某几列的查询有非常大的I/O优势。
文档存储	MongoDB CouchDB	文档存储一般用类似json的格式存储，存储的内容是文档型的。这样也就有机会对某些字段建立索引，实现关系数据库的某些功能。
key-value存储	Tokyo Cabinet / Tyrant Berkeley DB MemcacheDB Redis	可以通过key快速查询到其value。一般来说，存储不管value的格式，照单全收。（Redis包含了其他功能）
图存储	Neo4J FlockDB	图形关系的最佳存储。使用传统关系数据库来解决的话性能低下，而且设计使用不方便。
对象存储	db4o Versant	通过类似面向对象语言的语法操作数据库，通过对象的方式存取数据。
xml数据库	Berkeley DB XML BaseX	高效的存储XML数据，并支持XML的内部查询语法，比如XQuery, Xpath。

CAP定理

- 根据 CAP 原理将 NoSQL 数据库分成了满足 CA 原则、满足 CP 原则和满足 AP 原则三大类



HBase的设计目标

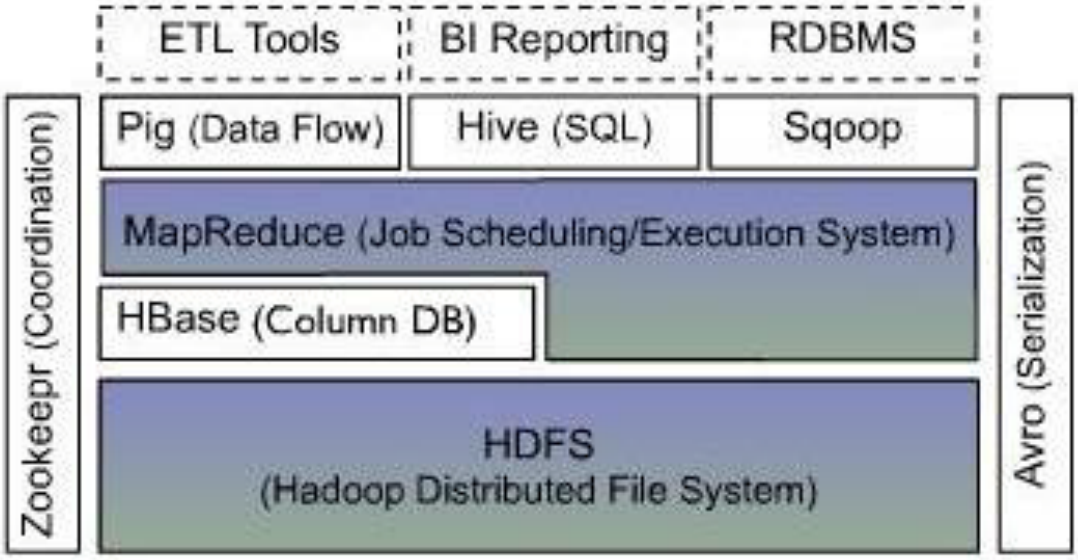
- 针对HDFS缺少结构化半结构化数据存储访问能力的缺陷，提供一个分布式数据管理系统，解决大规模的结构化和半结构化数据存储访问问题
- Google BigTable的一个开源实现
- 提供基于列存储模式的大数据表管理能力
- 可存储管理数十亿以上的数据记录，每个记录可包含百万以上的数据列
- HBase试图提供随机和实时的数据读写访问能力
- 具有高可扩展性、高可用性、容错处理能力、负载平衡能力、实时数据查询能力

HBase的功能特点

- 列式存储
- 表数据是稀疏的多维映射表
- 读写的严格一致性（区别于Cassandra的最终一致性）
- 提供很高的数据读写速度，为写数据进行了特别优化
- 良好的线形可扩展性
- 提供海量数据存储能力
- 数据会自动分片
- 具有自动的失效检测和恢复能力，保证数据不丢失
- 提供了方便的与HDFS和MapReduce集成的能力
- 提供Java API作为编程接口

HBase在Hadoop中的生态环境

- 构建于分布式文件系统HDFS之上
- 为上层应用提供结构化半结构化海量数据存储访问能力



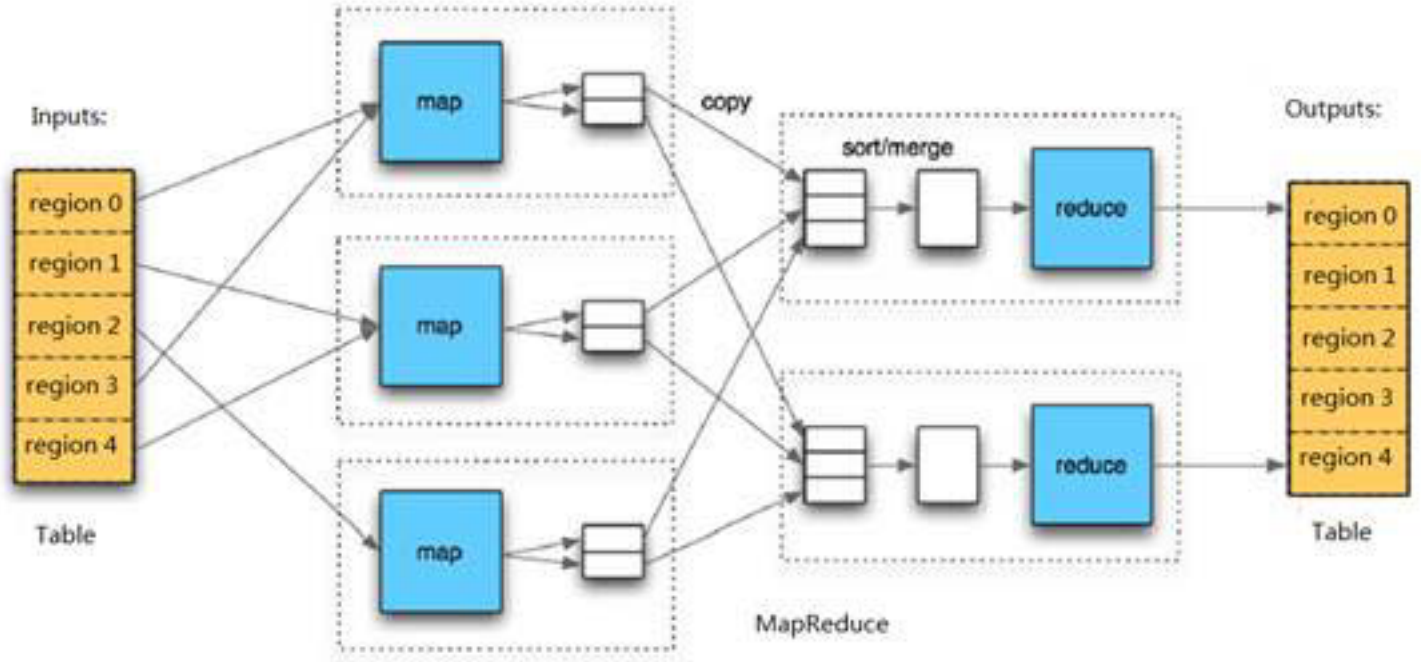
HBase的运行依赖于Hadoop HDFS文件系统提供数据的持久化（支持append功能），依赖Zookeeper提供集群的同步和协调。

HBase在Hadoop中的生态环境

- 分布式协调服务器Zookeeper
  - 保证任何时候，集群中只有一个HBase Master
  - 实时监控Region Server的状态，将Region Server的上线和下线信息实时通知给HBase Master
  - 存储HBase目录表的寻址入口
  - 存储HBase的schema，包括有哪些表，每个表有哪些列族等各种元信息

HBase在Hadoop中的生态环境

- 可与MapReduce协同工作，为MapReduce提供数据输入输出，以完成数据的并行化处理



HBase数据模型

- 逻辑数据模型
  - 数据存储逻辑模型与BigTable类似，但实现上有一些不同之处
  - 分布式多维表，表中的数据通过：一个行关键字(row key)，一个列关键字(column key)，一个时间戳(time stamp)进行索引和查询定位的

行关键字	时间戳	列"contents:"	列"anchor:"		列"mime:"
"com.cnn.www"	t9		"anchor:cnnsi.com"	"CNN"	
	t8		"anchor:my.look.ca"	"CNN.com"	
	t6	"<html>..."			"text/html"
	t5	"<html>..."			
	t3	"<html>..."			

HBase数据模型

- HBase物理存储格式

行关键字	时间戳	列"contents:"	
"com.cnn.www"	t6	"<html>..."	
	t5	"<html>..."	
	t3	"<html>..."	
行关键字	时间戳	列 "anchor:"	
"com.cnn.www"	t9	"anchor:cnnsi.com"	"CNN"
	t8	"anchor:my.look.ca"	"CNN.com"
行关键字	时间戳	列 "mime:"	
"com.cnn.www"	t6	"text/html"	

- 按照列存储的稀疏行/列矩阵。物理存储格式上按逻辑模型中的行进行分割，并按照列族存储。
- 值为空的列不予存储，节省存储空间。

行主键

- 行主键row key是用来检索记录的主键。这样的话，访问HBasetable中的行，有三种方式：1 通过单个row key访问；2 通过row key的范围range来访问；3 全表扫描。
- 行键（Row key）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。
- 存储时，数据按照Row key的字典序（byte order）排序存储。这样的话，设计key时，要充分排序存储这个特性，将经常一起读取的行存储放到一起。（空间局部性）
- 行的一次读写是原子操作（不论一次读写多少列）。这样的设计兼顾了用户可以理解在一行中的读写行为以及设计上的可扩展性。

列族

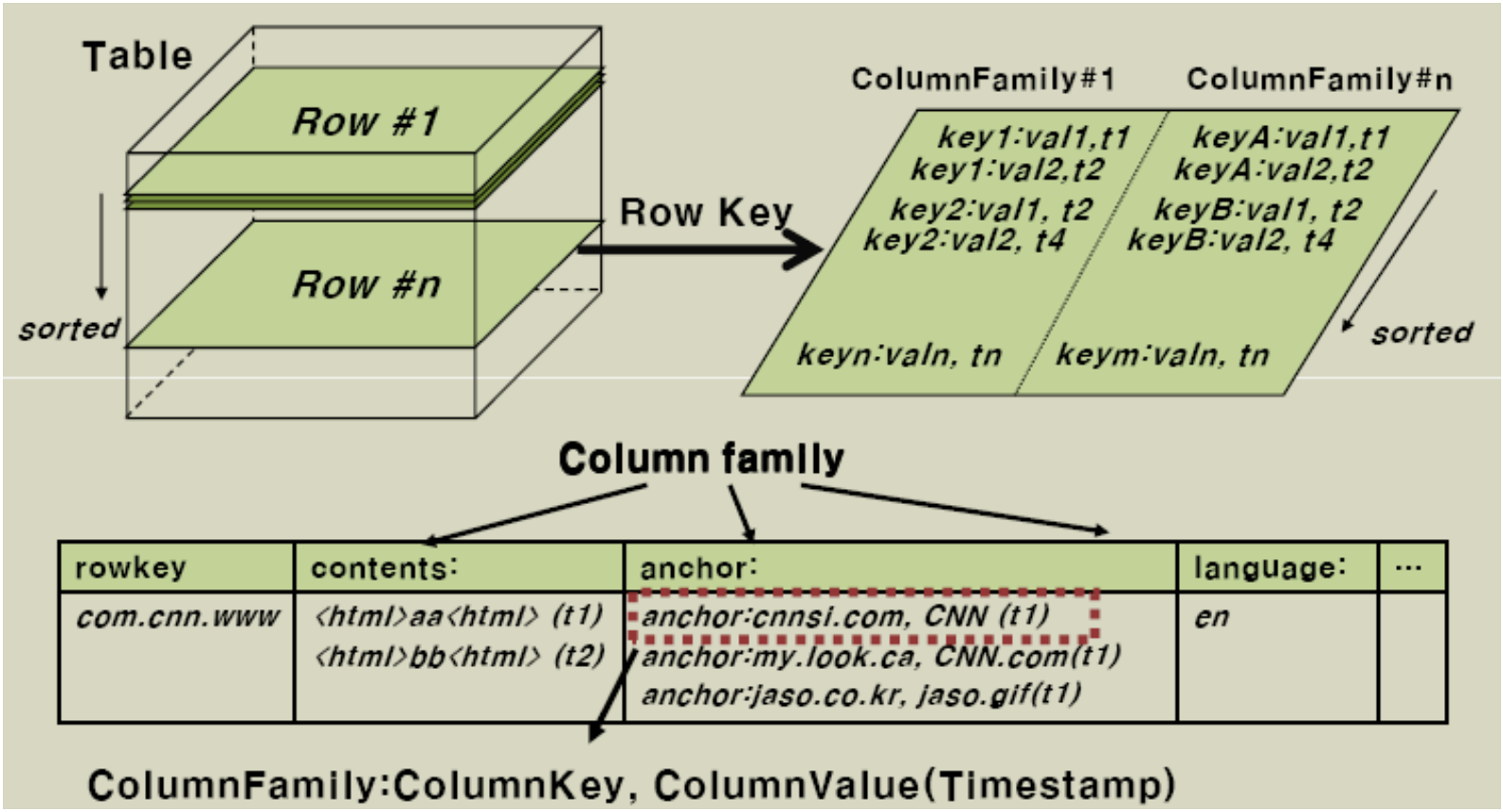
- 列族的设计与传统数据库中的列不一致
- 与BigTable中的模式一样，HBase表中的每个列，都归属于某个列族。列族是表的schema的一部分，必须在使用表之前定义。列名都以列族作为前缀。例如courses:history，courses:math都属于courses这个列族。
- 访问控制、磁盘和内存的使用统计都是在列族层面进行的。实际应用中，列族上的控制权限能帮助管理不同类型的应用：允许一些应用可以添加新的基本数据、一些应用可以读取基本数据并创建继承的列族、一些应用则只允许浏览数据（甚至可能因为隐私的原因不能浏览所有数据）

时间戳timestamp

- HBase中通过row和column确定的一个存贮单元称为单元格cell。每个cell都保存着同一份数据的多个版本。版本通过时间戳来索引。时间戳的类型是64位整型。时间戳可以由HBase（在数据写入时自动）赋值，此时时间戳是精确到毫秒的当前系统时间。时间戳也可以由客户显式赋值。如果应用程序要避免数据版本冲突，就必须自己生成具有唯一性的时间戳。每个cell中，不同版本的数据按照时间倒序排序，即最新的数据排在最前面。
- 为了避免数据存在过多版本造成的管理（包括存贮和索引）负担，HBase提供了两种数据版本回收方式。一是保存数据的最后n个版本，二是保存最近一段时间内的版本（比如最近七天）。用户可以针对每个列族进行设置。

HBase数据模型

- HBase物理存储格式



HBase的查询模式

- HBase通过行关键字、列（列族：列名）和时间戳确定一个存储单元，即：
- {row key, column family, column name, timestamp} → value
- HBase可以支持的查询方式：
  - 通过单个row key访问
  - 通过row key的范围来访问
  - 全表扫描
- 合理设计row key

HBase表设计

- Row key的设计要点
  - 例：收集一个集群（4000个节点）中的所有log并在HBase表LOG_DATA中存储。需要收集的字段有：（机器名，时间，事件，事件正文）
  - 插入操作：尽可能高效地插入
  - 查询操作：
    - 需求一：对于某台机器，查询一个大的时间段（例如1个月）内的所有满足条件的记录（单机查询）
    - 需求二：查询某个时间段内对所有机器满足条件的记录（全局查询）

HBase表设计

- row key设计
  - 方案一：[机器名][时间][事件]
  - 方案二：[时间][机器名][事件]
  - 方案一对插入友好，对单机查询友好
  - 方案二对插入不友好，对全局查询友好
- row key设计的弥补措施
  - salted（加盐）：对单调数据，通过计算盐值，放在单调数据前让其不单调。例如：
    - 盐值 = 时间 % 桶个数
    - row key：[盐值][时间][机器名][事件]
    - 副作用：查询变慢，降低了系统的吞吐量。桶个数不易太大

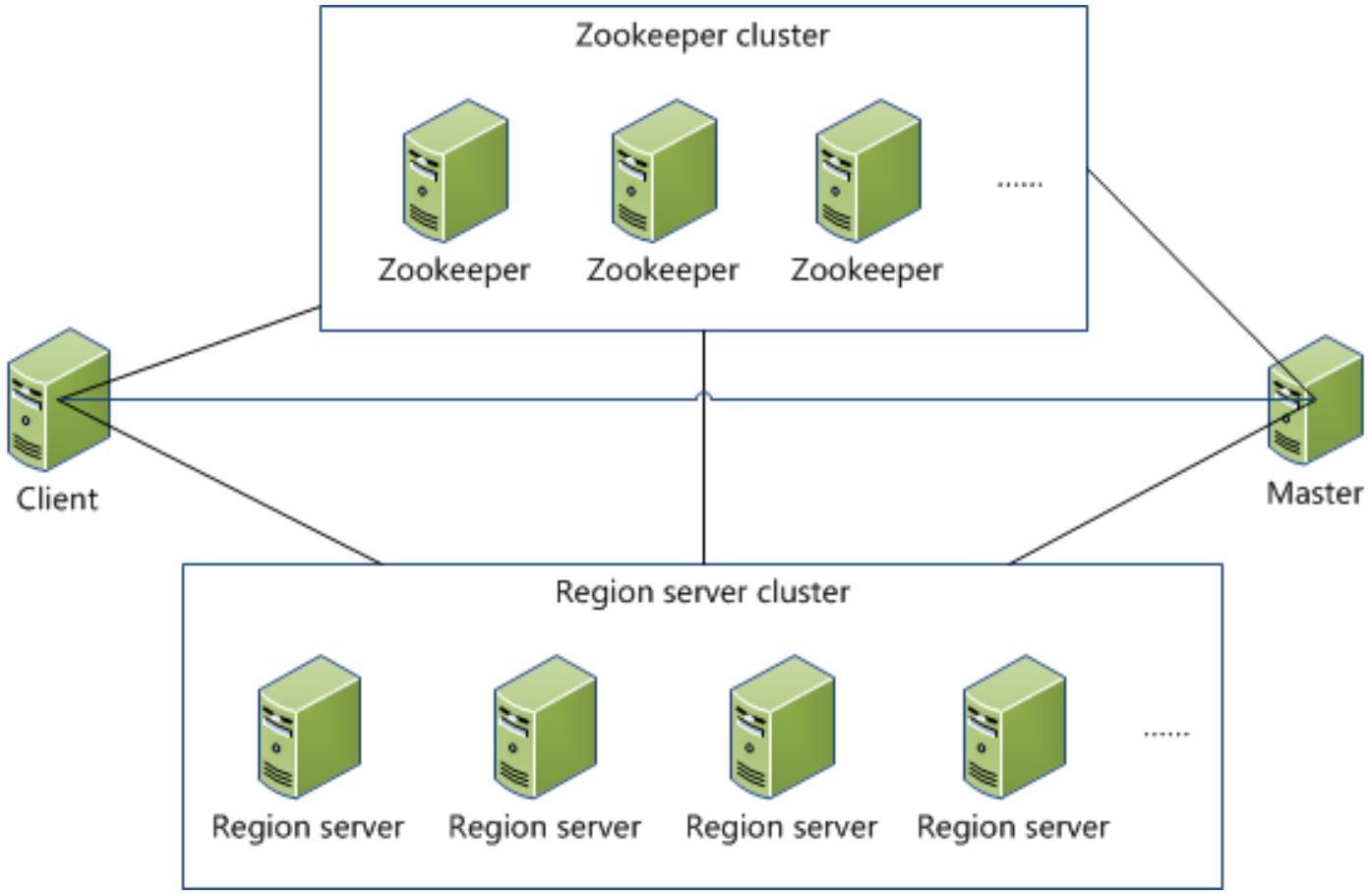
HBase表设计

- 表的规范化设计
  - 传统数据库：通常需要满足第3范式
  - NoSQL数据库：反规范化。应该将相关的数据都存放在一起，不担心冗余。
    - 常见设计和推荐意见：
      - 一行非常多时间戳 or 每个时间戳一行？一般推荐后者
      - 一行无数列 or 无数行？通常选择后者

HBase表设计

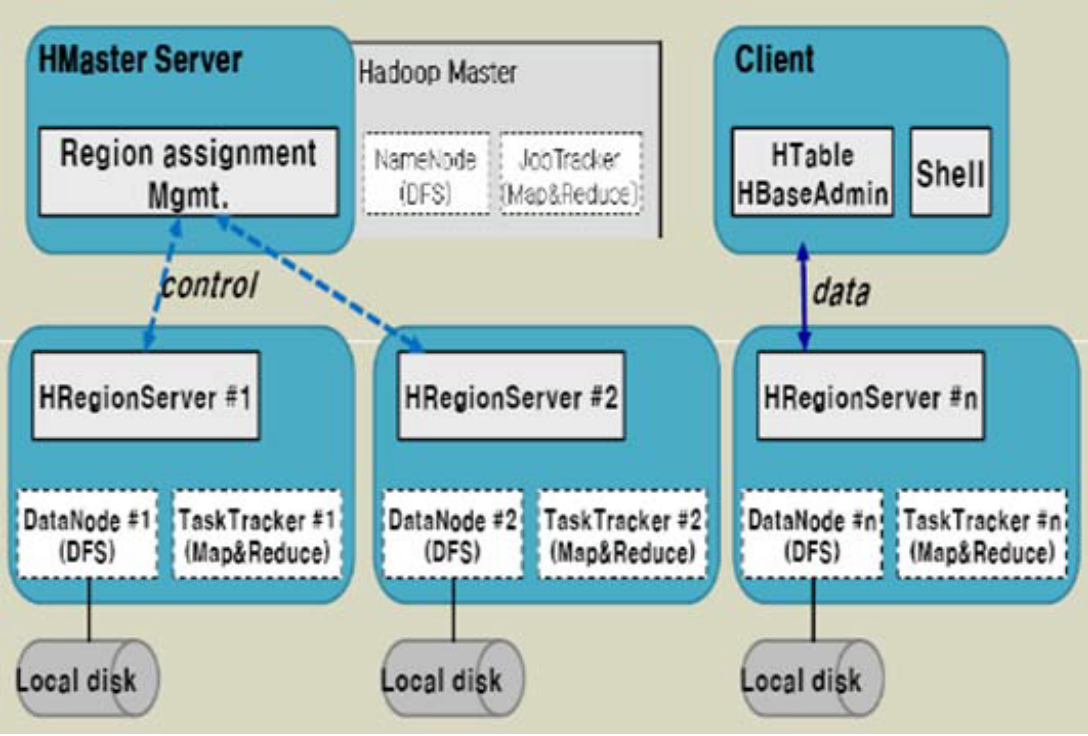
- 表设计的选择
  - 一张大表 or 按时间进行分表？--根据应用需求选择
  - 大表设计：
    - 优点：查询都在单张表完成
    - 缺点：数据过期时需要依赖major compaction进行压缩，会造成大量的数据读写；活跃Region在RegionServer间的分布可能不均匀。
  - 按时间分表设计：
    - 优点：数据过期可以通过简单删除整张表完成；活跃Region可以得到分布均匀。
    - 缺点：需要跨表查询时，性能会比较差。

HBase的运行组成


Zookeeper cluster
Zookeeper
Zookeeper
Zookeeper
.....
Client
Master
Region server cluster
Region server
Region server
Region server
Region server
.....

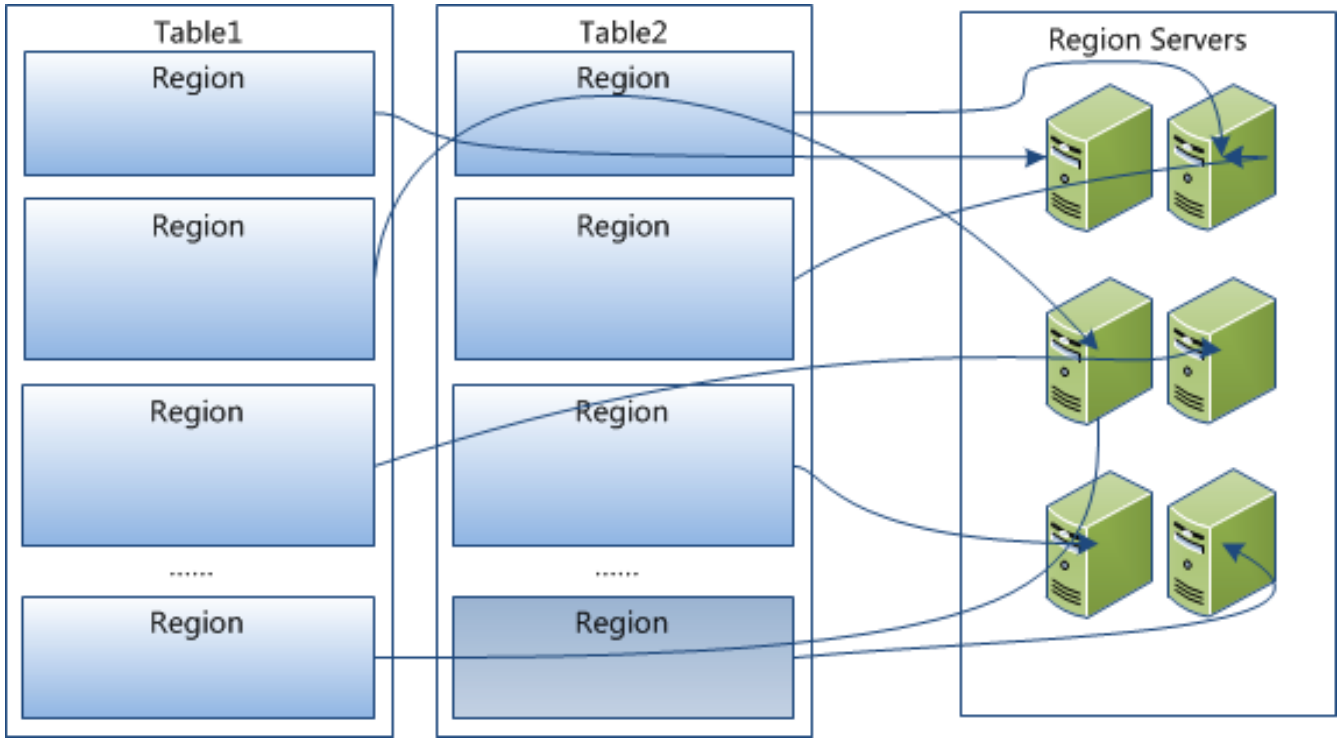
HBase的基本构架

- 由一个MasterServer和由一组子表数据区服务器RegionServer构成，分别存储逻辑大表中的部分数据。大表中的底层数据存于HDFS中。



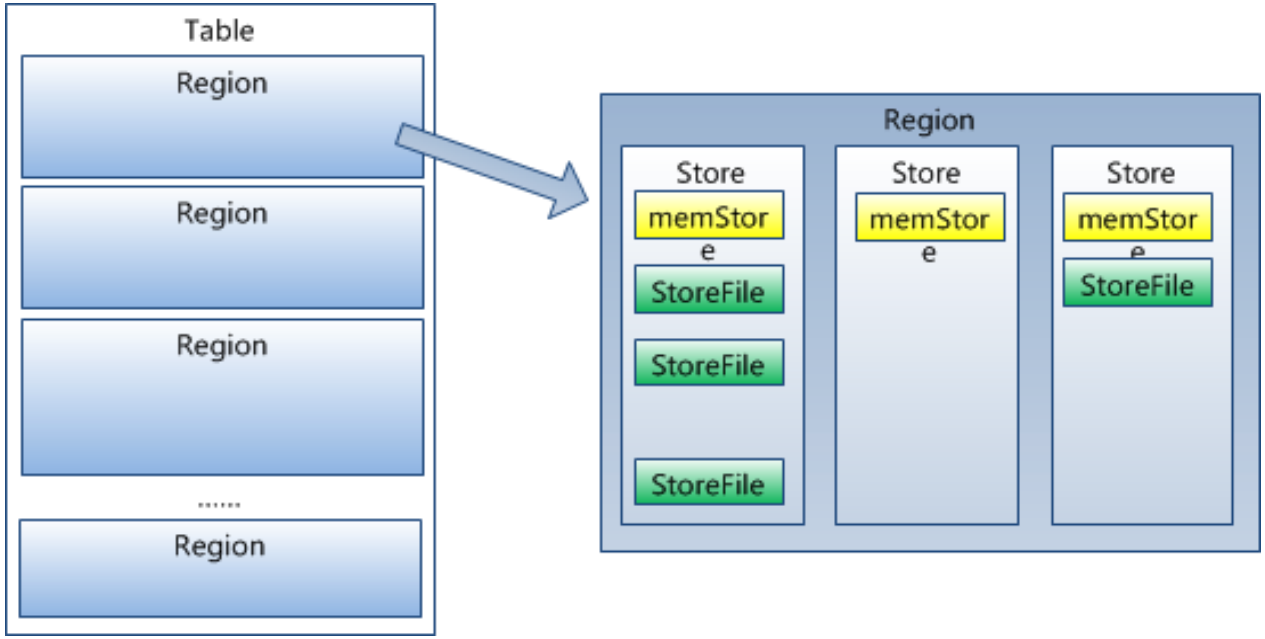
HBase数据存储管理方法

- HBase子表数据存储与子表服务器
  - 与BigTable类似，大表被分为很多个子表（Region），每个子表存储在一个子表服务器RegionServer上



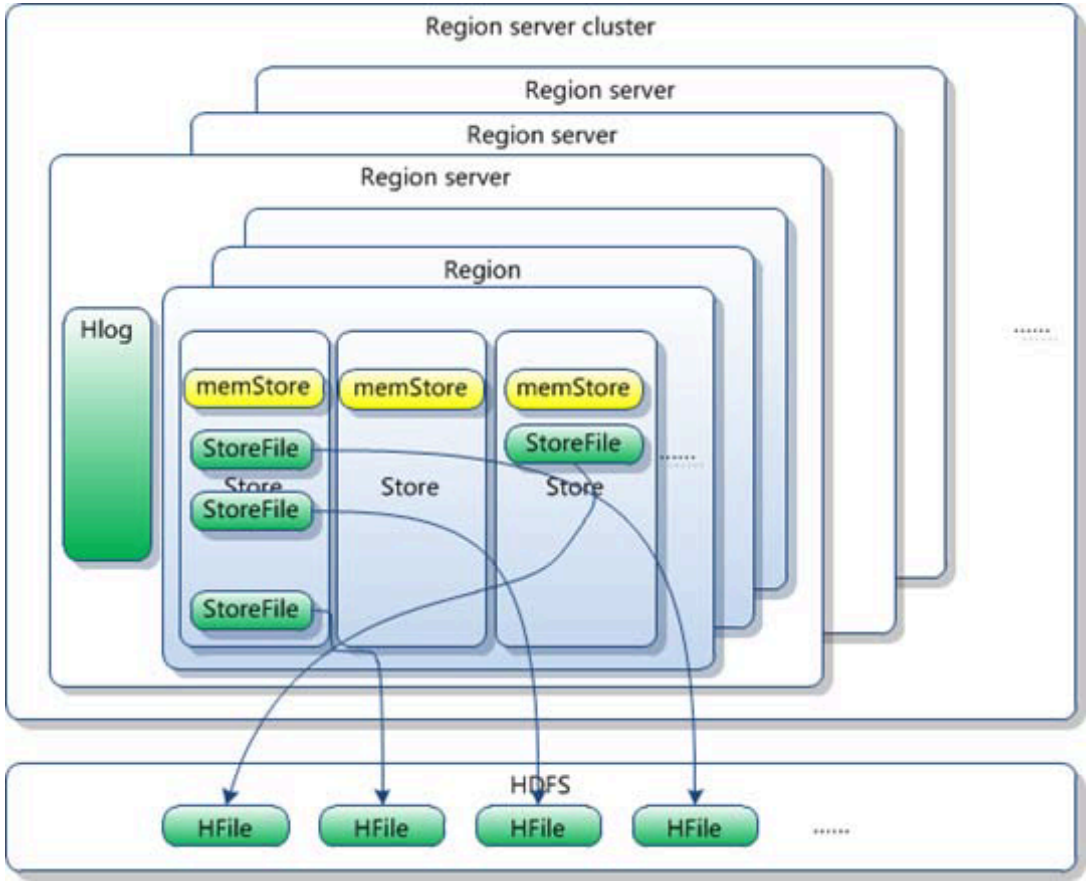
HBase数据存储管理方法

- HBase子表数据存储与子表服务器
  - 每个子表中的数据区Region由很多个数据存储块Store构成，而每个Store数据块又由存放在内存中的memStore和存放在文件中的StoreFile构成



HBase数据存储管理方法

- HBase子表数据存储与子表服务器



HBase数据存储管理方法

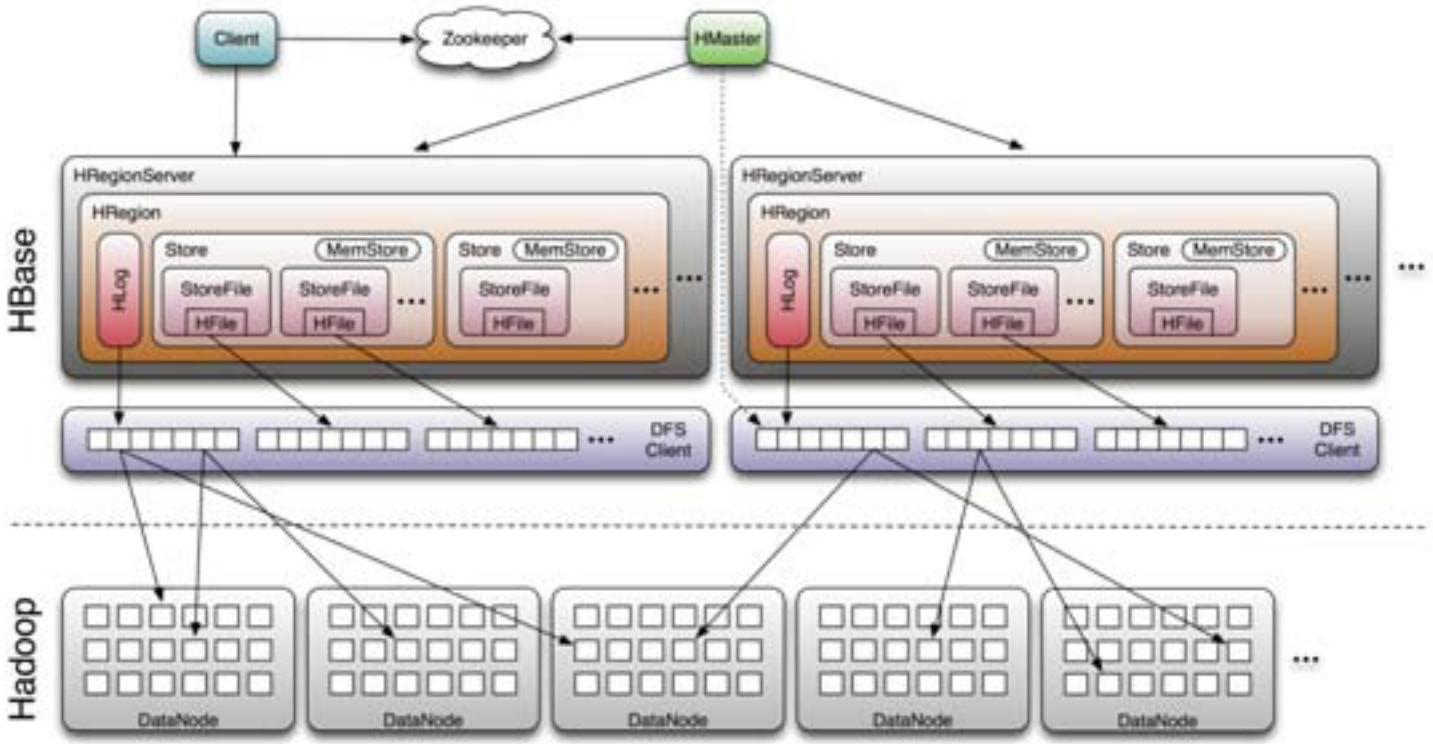
- HBase数据的访问
  - 当客户端需要进行数据更新时，先查到子表服务器，然后向子表提交数据更新请求。提交的数据并不直接存储到磁盘上的数据文件中，而是添加到一个基于内存的子表数据对象memStore中，当memStore中的数据达到一定大小时，系统将自动将数据写入到文件数据块StoreFile中。
  - 每个文件数据块StoreFile最后都写入到底层基于HDFS的文件中。

HBase数据存储管理方法

- HBase数据的访问
  - 需要查询数据时，子表先查memStore。如果没有，则再查磁盘上的StoreFile。每个StoreFile都有类似B树的结构，允许进行快速的数据查询。StoreFile将定时压缩，多个压缩为一个。
  - 两个小的子表可以进行合并；子表大到超过某个指定值时，子表服务器就需要调用HRegion.closeAndSplit()，把它分割为两个新的子表。

HBase数据存储管理方法

- HBase子表服务器与主服务器



HBase数据存储管理方法

- HBase主服务器HServer
  - 与BigTable类似，HBase使用主服务器HServer来管理所有子表服务器。主服务器维护所有子表服务器在任何时刻的状态。当一个新的子表服务器注册时，主服务器让新的子表服务器装载子表。若主服务器与子表服务器连接超时，那么子表服务器将自动停止，并重新启动；而主服务器则假定该子表服务器已死机，将其上的数据转移至其它子表服务器，将其上的子表标注为空闲，并在重新启动后另行分配使用。

HBase数据存储管理方法

- HBase数据记录的查询定位
  - 描述所有子表和子表中数据块的元数据都存放在专门的元数据表中，并存储在特殊的子表中。子表元数据会不断增长，因此会使用多个子表来保存。而所有元数据子表的元数据都保存在根子表中。主服务器会扫描根子表，从而得到所有的元数据子表位置，再进一步扫描这些元数据子表即可获得所寻找子表的位置。

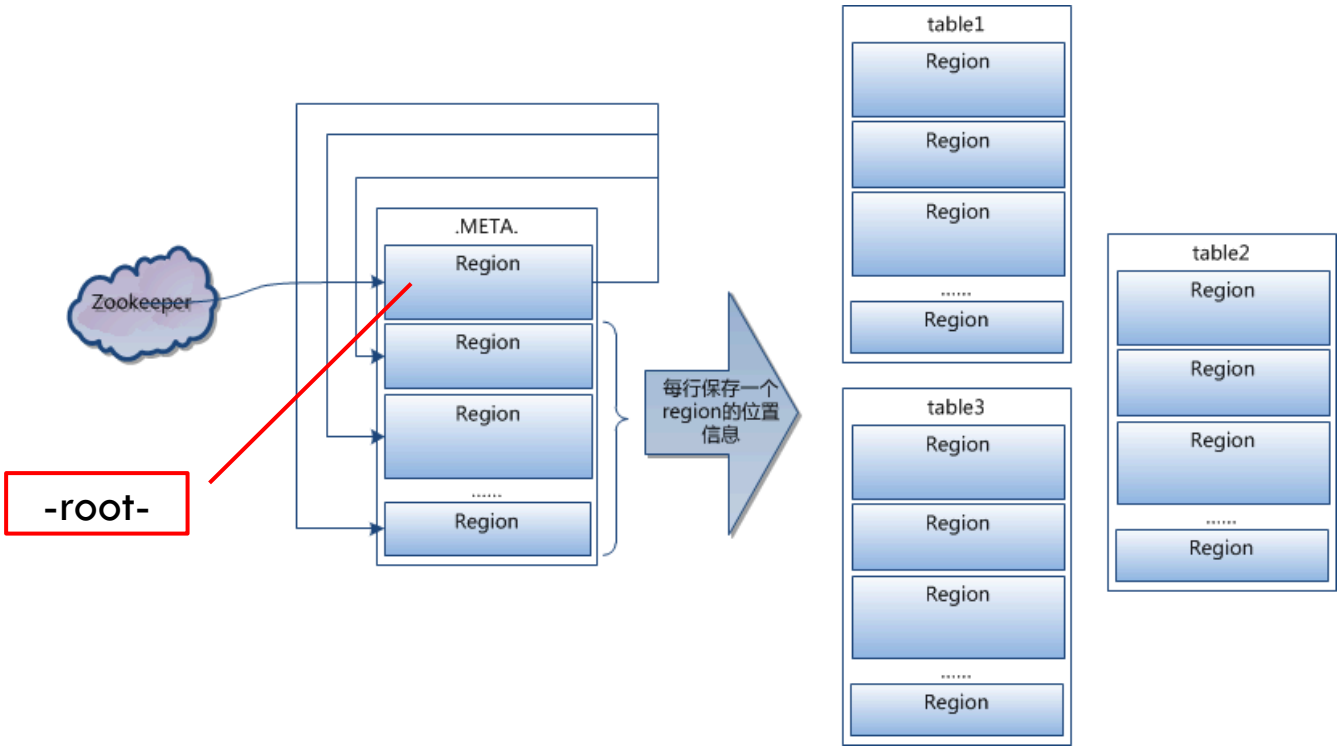
HBase数据存储管理方法

- HBase使用三层类似B+树的结构来保存Region位置
  - 通过Zookeeper里的文件得到-ROOT-表的位置，-ROOT-表永远不会被分割为多个Region
  - 通过-ROOT-表查找.META.表中相应Region的位置，为了加快访问，.META.表的全部Region的数据都会全部保存在内存中
  - 通过.META.表找到所要的用户表相应Region的位置

.META.表中，每行的row key为：<用户表名，region的起始row key，创建时间>
-ROOT-表中，每行的row key为：<.META., <用户表名，region的起始row key，创建时间>，创建时间>

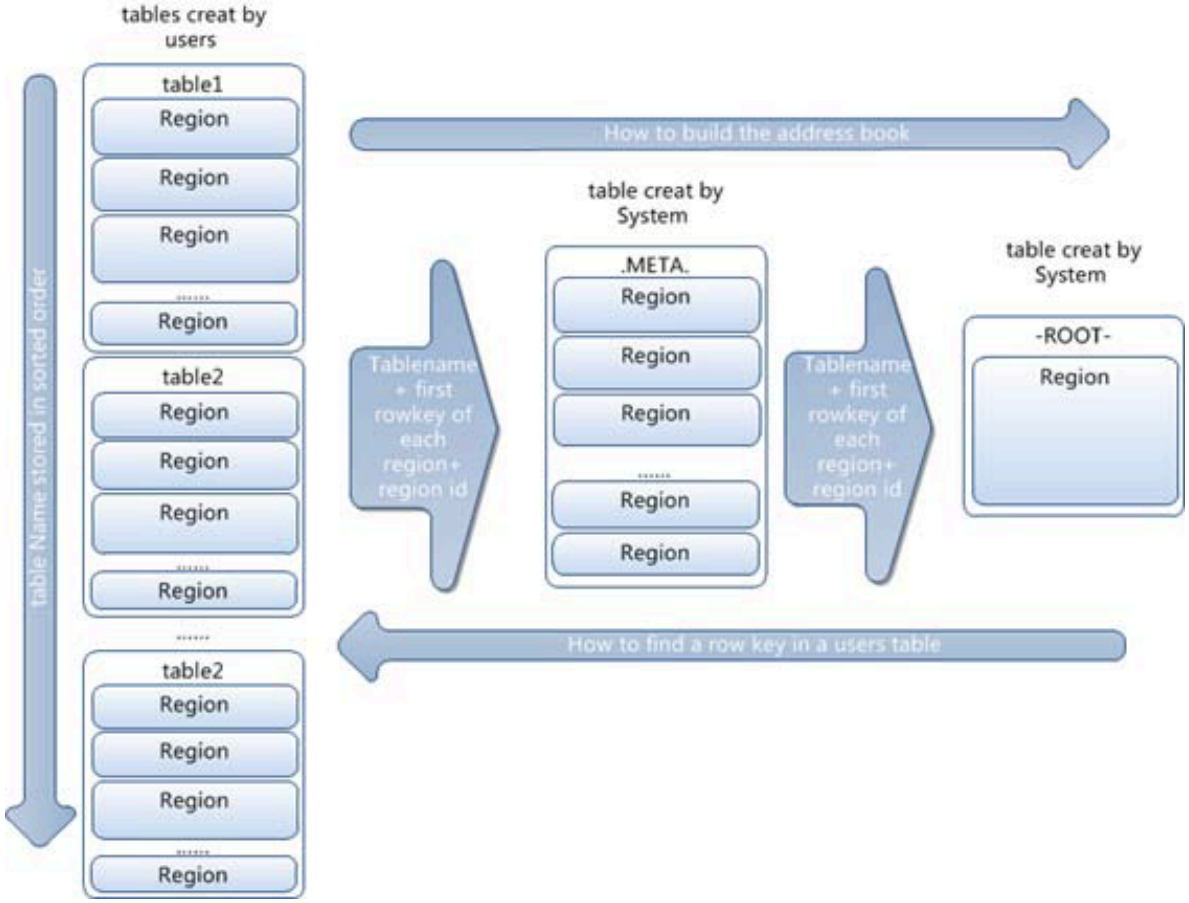
HBase数据存储管理方法

- HBase数据记录的查询定位
  - 元数据子表采用三级索引结构
    - 根子表→用户表的元数据表→用户表



HBase数据存储管理方法

- HBase数据记录的查询定位



HBase存储格式

- HBase中的所有数据文件都存储在Hadoop HDFS文件系统上，主要包括两种文件类型：
  - 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StoreFile就是对HFile做了轻量级包装，即StoreFile底层就是HFile
  - HLogFile，HBase中WAL（Write Ahead Log）的存储格式，物理上是Hadoop的Sequence File

摘要

- HBase基本工作原理
- HBase基本操作
- HBase编程方法示例

HBase

- 官网：
  - http://hbase.apache.org
- 最新版本：
  - 2.3.3　Nov 2, 2020
  - 1.6.0　Mar 6, 2020
- 参考指南：
  - http://hbase.apache.org/book.html

HBase配置和安装

- hbase-env.sh
  - export JAVA_HOME=$(/usr/libexec/JAVA_HOME)
- hbase-site.xml
  - Standalone模式
  - 伪分布式本地模式
  - 全分布式模式
- Web Interface：
  - http://localhost:16010

HBase基本操作与编程方法

- HBase Shell操作
  - Hbase shell常用的操作命令有create, describe, disable, enable, drop, list, scan, put, get, delete, deleteall, count, status等，通过help可以看到详细的用法。

HBase Shell操作

- 表的管理
  - 查看有哪些表
    - list
  - 创建表
    - create <table>, {NAME => <family>, VERSIONS => <VERSIONS>}
  - 删除表
    - disable <table>
    - drop <table>
  - 查看表结构
    - describe <table>
  - 修改表结构
    - alter 't1', {NAME => 'f1'}, {NAME => 'f2', METHOD => 'delete'}

HBase Shell操作

- 表数据的增删查改
  - 添加数据
    - put <table>,<rowkey>,<family:column>,<value>,<timestamp>
  - 查询数据
    - 查询某行记录
      - get <table>,<rowkey>,[<family:column>,....]
    - 扫描表
      - scan <table>, {COLUMNS => [<family:column>,....], LIMIT => num}
    - 查询表中的数据行数
      - count <table>, {INTERVAL => intervalNum, CACHE => cacheNum}

HBase Shell操作

- 表数据的增删查改
  - 删除数据
    - 删除行中的某个列值
      - delete <table>, <rowkey>, <family:column> , <timestamp>
      - 必须指定列名
    - 删除行
      - deleteall <table>, <rowkey>, <family:column> , <timestamp>
      - 可以不指定列名，删除整行数据查询表中的数据行数
    - 删除表中的所有数据
      - truncate <table>

HBase Shell操作

- 我们建立如下的表以及数据table:students

ID	Description		Courses			Home
	Name	Height	Chinese	Math	Physics	Province
001	Li Lei	176	80	90	95	Zhejiang
002	Han Meimei	183	88	77	66	Beijing
003	Xiao Ming	162	90	90	90	Shanghai

创建表格与列举表格


```
hadoop@master: ~
File  Edit  View  Terminal  Help
hadoop@master:~$ ls
cleanhadooplogs   hadoop-1.0.3  runhive          TempStatsStore
cleanhbaselogs    hadoopcode    starthadoop      tutorial
csapp             hbase-0.94.1  starthbase       updatehadoopconfig
data              hive-0.9.0    startzookeeper   updatehbaseconfig
derby.log         metastore_db  stophadoop       Videos
Desktop           Music         stophbase        zkdata
Documents         Pictures      stopzookeeper    zookeeper-3.4.3
Downloads         Public        temp             zookeeper.out
examples.desktop  ratings.dat   Templates
hadoop@master:~$ hbase shell
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 0.94.1, r1365210, Tue Jul 24 18:40:10 UTC 2012

hbase(main):001:0> list
TABLE
scores
1 row(s) in 3.8490 seconds

hbase(main):002:0> create 'students','ID','Description','Courses','Home'
0 row(s) in 7.8970 seconds

hbase(main):003:0>
```

插入数据


```
hadoop@master: ~
File  Edit  View  Terminal  Help
Documents          Pictures      stopzookeeper   zookeeper-3.4.3
Downloads          Public        temp            zookeeper.out
examples.desktop   ratings.dat   Templates
hadoop@master:~$ hbase shell
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 0.94.1, r1365210, Tue Jul 24 18:40:10 UTC 2012

hbase(main):001:0> list
TABLE
scores
1 row(s) in 3.8490 seconds

hbase(main):002:0> create 'students','ID','Description','Courses','Home'
0 row(s) in 7.8970 seconds

hbase(main):003:0> put 'students','001','Description:Name','Li Lei'
0 row(s) in 0.4350 seconds

hbase(main):004:0> put 'students','001','Description:Height','176'
0 row(s) in 0.0280 seconds

hbase(main):005:0> put 'students','001', 'Courses:Chinese' ,'80'
```

显示描述表信息


```
hadoop@master: ~
File  Edit  View  Terminal  Help
 SSION => 'NONE', MIN_VERSIONS => '0', TTL => '21474
 83647', KEEP_DELETED_CELLS => 'false', BLOCKSIZE =>
  '65536', IN_MEMORY => 'false', ENCODE_ON_DISK => '
 true', BLOCKCACHE => 'true'}]}
1 row(s) in 0.1260 seconds

hbase(main):042:0> list
TABLE
scores
students
2 row(s) in 0.0810 seconds

hbase(main):043:0> describe 'students'
DESCRIPTION                                                  ENABLED
 {NAME => 'students', FAMILIES => [{NAME => 'Courses', DATA_BLOCK_ENCODIN true
 G => 'NONE', BLOOMFILTER => 'NONE', REPLICATION_SCOPE => '0', VERSIONS =
 > '3', COMPRESSION => 'NONE', MIN_VERSIONS => '0', TTL => '2147483647',
 KEEP_DELETED_CELLS => 'false', BLOCKSIZE => '65536', IN_MEMORY => 'false
 ', ENCODE_ON_DISK => 'true', BLOCKCACHE => 'true'}, {NAME => 'Descriptio
 n', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'NONE', REPLICATION_SC
 OPE => '0', VERSIONS => '3', COMPRESSION => 'NONE', MIN_VERSIONS => '0',
  TTL => '2147483647', KEEP_DELETED_CELLS => 'false', BLOCKSIZE => '65536
 ', IN_MEMORY => 'false', ENCODE_ON_DISK => 'true', BLOCKCACHE => 'true'}
 , {NAME => 'Home', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'NONE',
  REPLICATION_SCOPE => '0', VERSIONS => '3', COMPRESSION => 'NONE', MIN_V
 ERSIONS => '0', TTL => '2147483647', KEEP_DELETED_CELLS => 'false', BLOC
 KSIZE => '65536', IN_MEMORY => 'false', ENCODE_ON_DISK => 'true', BLOCKC
 ACHE => 'true'}, {NAME => 'ID', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILT
 ER => 'NONE', REPLICATION_SCOPE => '0', VERSIONS => '3', COMPRESSION =>
 'NONE', MIN_VERSIONS => '0', TTL => '2147483647', KEEP_DELETED_CELLS =>
 'false', BLOCKSIZE => '65536', IN_MEMORY => 'false', ENCODE_ON_DISK => '
 true', BLOCKCACHE => 'true'}]}
1 row(s) in 0.0780 seconds

hbase(main):044:0>
```

输入数据与扫描数据


```
hadoop@master: ~
File  Edit  View  Terminal  Help
hbase(main):051:0> put 'students','002','Home:Province','Bei Jing'
0 row(s) in 0.0420 seconds

hbase(main):052:0> put 'students','003','Home:Province','Shang Hai'
0 row(s) in 0.0270 seconds

hbase(main):053:0> put 'students','002','Description:Name','Han Meimei'
0 row(s) in 0.0470 seconds

hbase(main):054:0> put 'students','002','Description:Height','183'
0 row(s) in 0.0360 seconds

hbase(main):055:0> put 'students','003','Description:Height','162'
0 row(s) in 0.0240 seconds

hbase(main):056:0> put 'students','003','Description:Name','Xiao Ming'
0 row(s) in 0.0190 seconds

hbase(main):057:0> scan 'students'
ROW                           COLUMN+CELL
 001                          column=Courses:Chinese, timestamp=1351776664719, value=80
 001                          column=Courses:Math, timestamp=1351776678749, value=90
 001                          column=Courses:Physics, timestamp=1351776693609, value=95
 001                          column=Description:Height, timestamp=1351776295307, value=176
 001                          column=Description:Name, timestamp=1351776277573, value=Li Lei
 001                          column=Home:Province, timestamp=1351776717090, value=Zhe Jiang
 002                          column=Description:Height, timestamp=1351776794960, value=183
 002                          column=Description:Name, timestamp=1351776780963, value=Han Meimei
 002                          column=Home:Province, timestamp=1351776742893, value=Bei Jing
 003                          column=Description:Height, timestamp=1351776806923, value=162
 003                          column=Description:Name, timestamp=1351776822022, value=Xiao Ming
 003                          column=Home:Province, timestamp=1351776757867, value=Shang Hai
3 row(s) in 0.2140 seconds

hbase(main):058:0>
```

限制列进行扫描


```
hbase(main):098:0* scan 'students',{COLUMNS=>'Courses'}
ROW                           COLUMN+CELL
 001                          column=Courses:Chinese, timestamp=1351776664719, value=80
 001                          column=Courses:Math, timestamp=1351776678749, value=90
 001                          column=Courses:Physics, timestamp=1351776693609, value=95
 002                          column=Courses:Chinese, timestamp=1351776979636, value=88
 002                          column=Courses:Math, timestamp=1351776989455, value=77
 002                          column=Courses:Physics, timestamp=1351776999603, value=66
 003                          column=Courses:Chinese, timestamp=1351776946110, value=90
 003                          column=Courses:Math, timestamp=1351776952592, value=90
 003                          column=Courses:Physics, timestamp=1351776963651, value=90
3 row(s) in 0.1560 seconds

hbase(main):099:0>
```

HBase中的disable和enable

- disable和enable都是HBase中比较常见的操作，很多对table的修改都需要表在disable的状态下才能进行
- disable 'students'将表students的状态更改为disable的时候，HBase会在zookeeper中的table结点下做记录
- 在zookeeper记录下该表的同时，还会将表的region全部下线，region为offline状态
- enable的过程和disable相反，会把表的所有region上线，并删除zookeeper下的标志。如果在enable前，META中有region的server信息，那么此时会在该server上将该region上线；如果没有server的信息，那么此时还要随机选择一台机器作为该region的server

摘要

- HBase基本工作原理
- HBase基本操作
- HBase编程方法示例

HBase的Java编程

- HBase Java编程接口概述HBaseConfiguration
  - HBaseConfiguration是每一个HBase client都会使用到的对象，它代表的是HBase配置信息。
  - 默认的构造方式会尝试从hbase-default.xml和hbase-site.xml中读取配置。如果classpath没有这两个文件，就需要你自己设置配置。
  - 配置Java代码示例：

```
Configuration HBASE_CONFIG = new Configuration();
HBASE_CONFIG.set("hbase.zookeeper.quorum", "zkServer");
HBASE_CONFIG.set("hbase.zookeeper.property.clientPort", "2181");
HBaseConfiguration cfg= new BaseConfiguration(HBASE_CONFIG);
```

HBase的Java编程：创建表

- 创建表是通过Admin对象来操作的。Admin负责表的META信息处理。Admin提供了createTable这个方法：
  - public void createTable(HTableDescriptor desc)
- HTableDescriptor代表的是表schema
- HColumnDescriptor代表的是column的schema

HBase的Java编程：创建表

- Java代码示例：

```
Connection conn= ConnectionFactory.createConnection(config);
Admin  admin= conn.getAdmin();
HTableDescriptor  t = new HTableDescriptor(tableName);
t.addFamily(new HColumnDescriptor("f1"));
t.addFamily(new HColumnDescriptor("f2"));
t.addFamily(new HColumnDescriptor("f3"));
t.addFamily(new HColumnDescriptor("f4"));
admin.createTable(t);
```

HBase的Java编程：插入数据

- Table通过put方法来插入数据，可以传递单个批Put对象或者List put对象来分别实现单条插入和批量插入。Put对象的常用方法：

```
public static void addData(String tableName, String rowKey, String family,  String qualifier, String value) throws Exception{
    try{
        Connection conn= ConnectionFactory.createConnection(config);
        Table table =conn.getTable(tableName);
        Put put= new Put(Bytes.toBytes(rowKey));
        put.add(Bytes.toBytes(family), Bytes.toBytes(qualifier),Bytes.toBytes(value));
        table.put(put);
        System.out.println("insert record success!");
    } catch(IOException e) { e.printStackTrace(); }
}
```

HBase的Java编程：删除表

- 删除表也通过Admin来操作，删除表之前首先要disable表。这是一个非常耗时的操作，所以不建议频繁删除表。disableTable和deleteTable分别用来disable和delete表
- Java代码示例：

```
if(admin.tableExists(tableName)){
        admin.disableTable(tableName);
        admin.deleteTable(tableName);
}
```

HBase的Java编程：查询数据

- 查询分为单条随机查询和批量查询
  - 单条查询是通过row key在table中查询某一行的数据。Table提供了get方法来完成单条查询。
  - 批量查询是通过制定一段row key的范围来查询。Table提供了个getScanner方法来完成批量查询。
- Java代码示例：

```
Scan s = new Scan();   s.setMaxVersions();
ResultScanner  ss= table.getScanner(s);
for(Result r:ss){
   System.out.println(new String(r.getRow()));
   for(KeyValuekv:r.raw()){  System.out.println(new String(kv.getColumn()));}
}
```

HBase的Java编程：删除数据

- Table通过delete方法来删除数据：delete(final Delete delete)
- Delete常用方法：
  - deleteFamily或deleteColumns：
  - 指定要删除的family或者column的数据。如果不调用任何这样的方法，将会删除整行

```
Table table = conn.getTable("mytest");
Delete d = new Delete("row1".getBytes());
table.delete(d)
```

HBase的Java编程：切分表

- 参数hbase.hregion.max.filesize指示在当前ReigonServer上单个Reigon的最大存储空间，单个Region超过该值时，这个Region会被自动split成更小的region。
  - Admin提供split方法来将table进行手工split。
    - public void split(TableName tableName)
  - Admin提供splitRegion方法来将region进行手工split。
    - public void splitRegion(byte[] regionName)
  - 由于split是一个异步操作，并不能确切地控制region的个数。

分布式关系数据库

- 1.场地自治性（Local Autonomy)
- 2.非集中式管理（NoReliance On Central Site)
- 3.高可靠性（Contiuous Operation)
- 4.位置独立性（Location Transparency and Location Independence)
- 5.数据分割独立性（Fragmentation Independence)
- 6.数据复制独立性（Replication Independence)
- 7.分布式查询处理（Distributed Query Processing)
- 8.分布式事务管理（Distributed Transaction Management)
- 9.硬件独立性（Hardware Independence)
- 10.操作系统独立性 (Operating System Independence)
- 11.网络独立性(Network Independence)
- 12.数据库管理系统独立性(DBMS Independence)

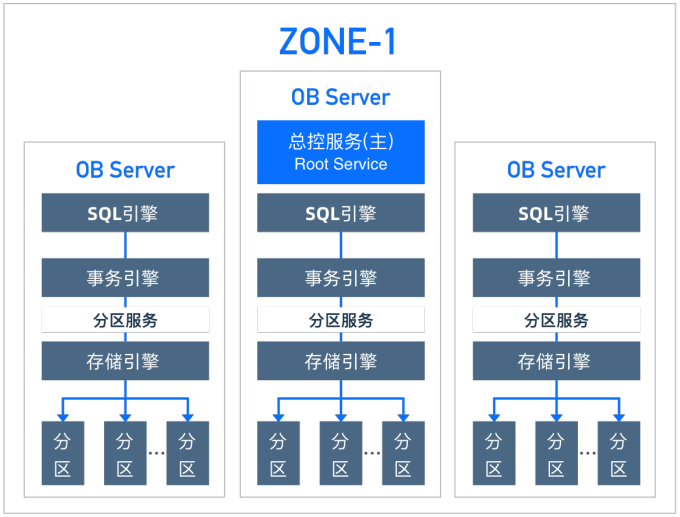
OceanBase

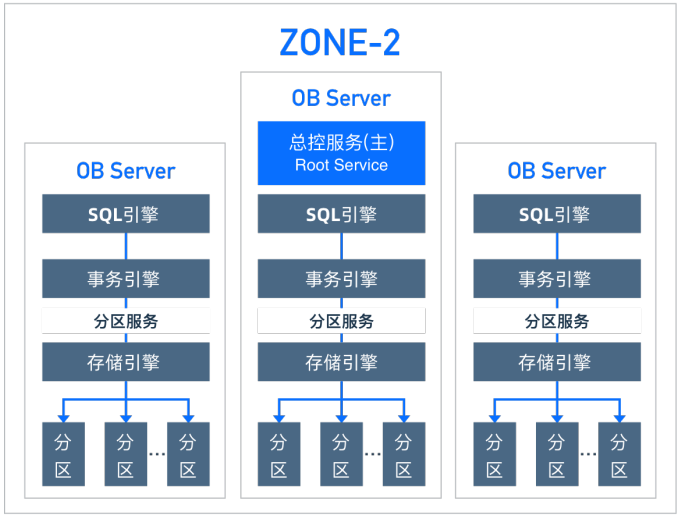
- OceanBase 数据库是蚂蚁集团不基于任何开源产品，完全自研的企业级分布式关系数据库，在普通硬件上实现金融级高可用，首创“三地五中心”城市级故障自动无损容灾新标准，具备卓越的水平扩展能力，全球首家通过 TPC-C 标准测试的分布式数据库，2020 年 5 月，OceanBase 以 7.07亿 tpmC 的在线事务处理性能，打破了去年自己创造的TPC-C 世界纪录。截止至目前，OceanBase 是第一个也是唯一一个上榜的中国数据库。产品立项于 2010 年，具有数据强一致、高可用、高性能、在线扩展、高度兼容 SQL 标准和主流关系数据库、低成本等特点，承担支付宝 100% 核心链路，在国内几十家银行、保险公司等金融客户的核心系统中稳定运行。

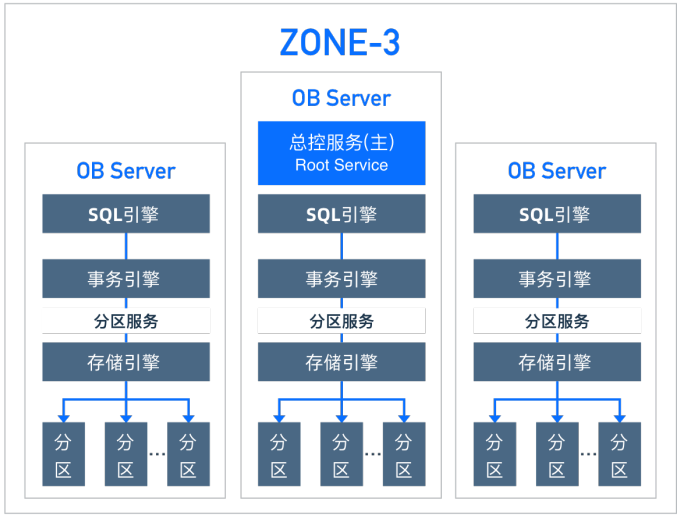
OceanBase

- 强一致
- 透明可扩展
- 极致高可用
- 多租户
- 高兼容性
- 完全自主知识产权
- 高性能
- 安全性

OceanBase


ZONE-1
OB Server
总控服务(主)
Root Service
SQL引擎
事务引擎
分区服务
存储引擎
分区
OB Server
SQL引擎
事务引擎
分区服务
存储引擎
分区
OB Server
SQL引擎
事务引擎
分区服务
存储引擎
分区


ZONE-2
OB Server
SQL引擎
事务引擎
分区服务
存储引擎
分区
OB Server
总控服务(主)
Root Service
SQL引擎
事务引擎
分区服务
存储引擎
分区
OB Server
SQL引擎
事务引擎
分区服务
存储引擎
分区


ZONE-3
OB Server
SQL引擎
事务引擎
分区服务
存储引擎
分区
OB Server
总控服务(主)
Root Service
SQL引擎
事务引擎
分区服务
存储引擎
分区
OB Server
SQL引擎
事务引擎
分区服务
存储引擎
分区