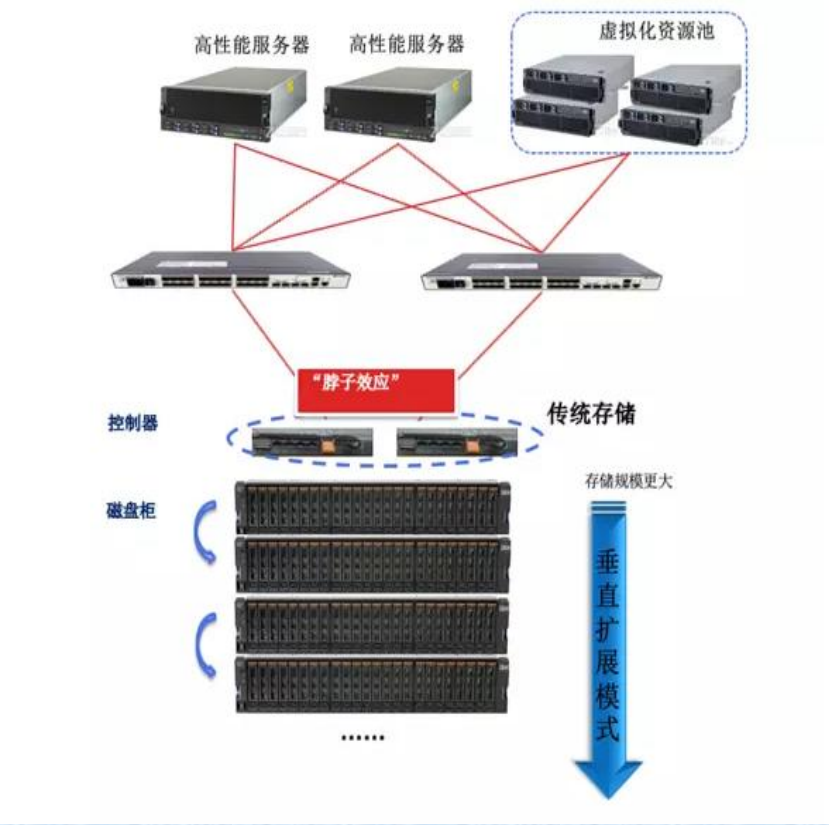
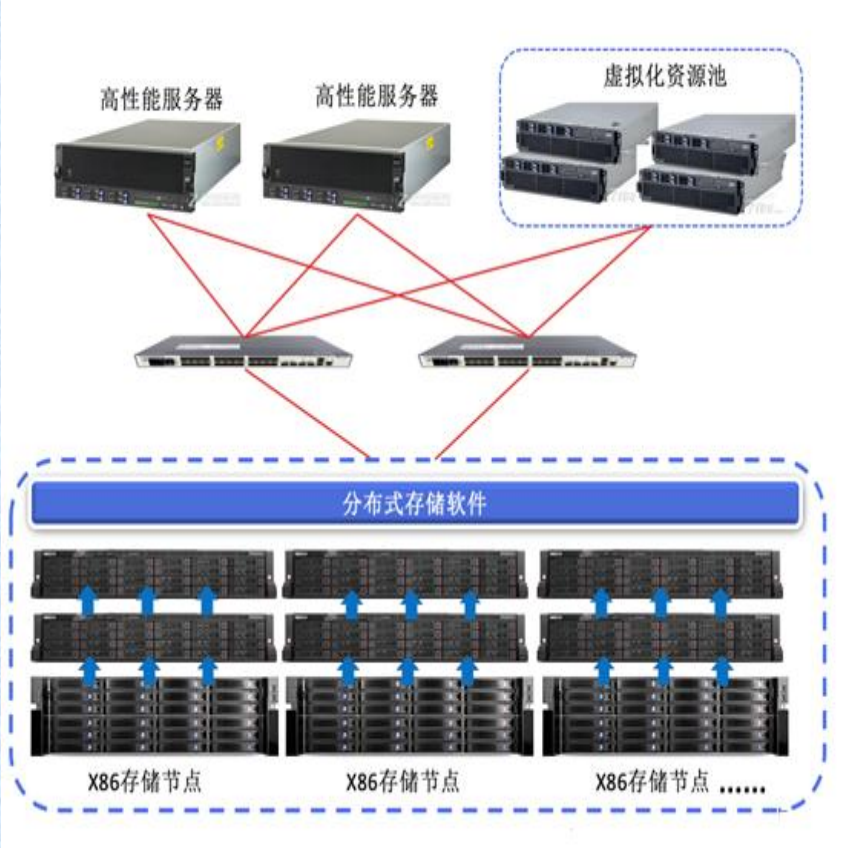
Lecture 13 分布式存储架构

- Hbase存储模型与架构
- 二次索引表机制
- 分布式协同管理

10.3 分布式存储架构

集中式存储 vs. 分布式存储





分布式存储的三种模式

- 块存储
- 文件存储
- 对象存储

块存储：将裸磁盘空间整个映射给主机使用，通过划分逻辑盘、做Raid或者LVM（逻辑卷）等方式分出多个逻辑硬盘，再将逻辑盘映射给主机使用（O/S文件系统，华为的FusionStorage ）

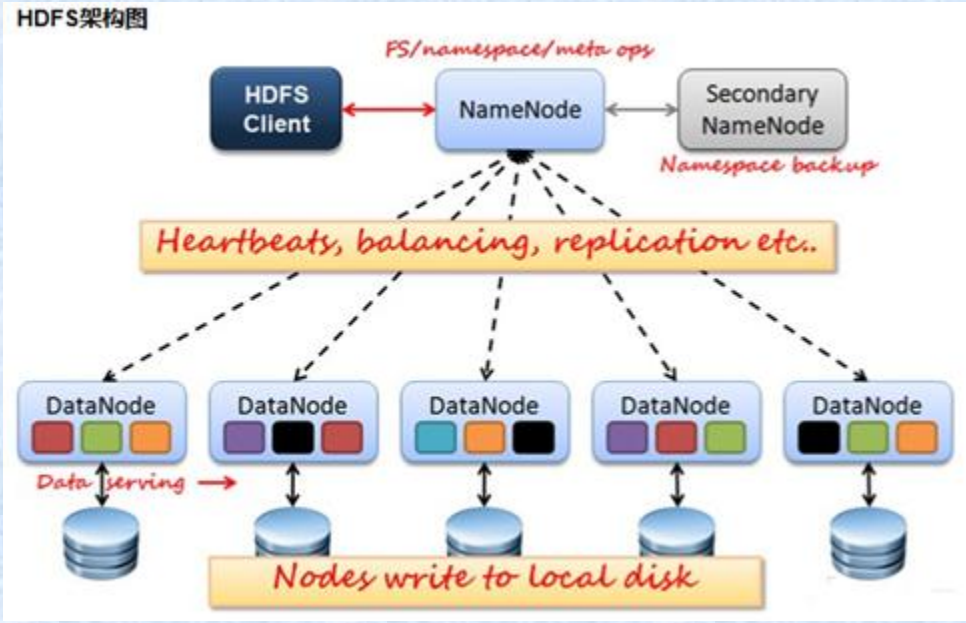
文件存储：拿一台服务器装上合适的操作系统与软件，即可架设FTP与NFS服务了，这种提供文件存储的服务器就是文件存储的一种方式（ FTP、NFS服务器等）

对象存储：即基于对象的存储，将文件被拆分成多个称为“对象”的离散单元并分散存储在多个服务器上。对象存储需要一个简单的HTTP应用编程接口(API)，以供大多数客户端（各种语言）使用（ Ceph，HDFS，google基于GFS的存储）

三种分布式存储架构

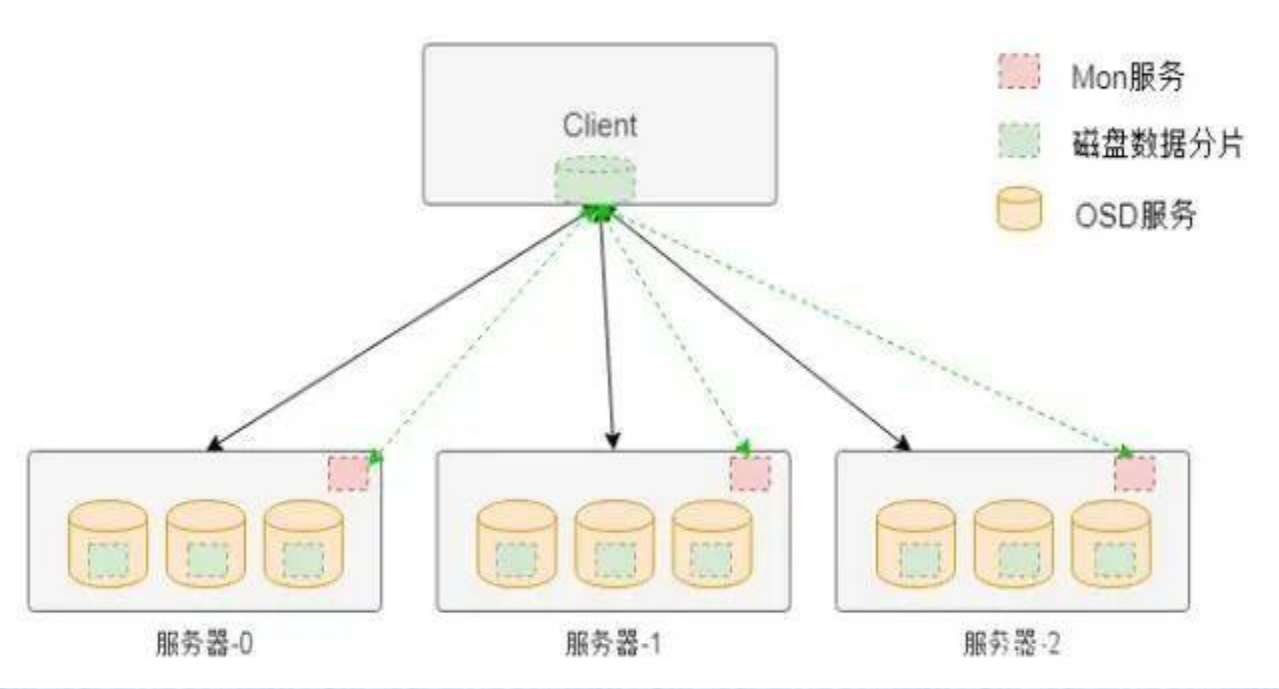
- 主从架构（Master/Slave）
- 去中心架构（peer-to-peer）
- 一致性哈希架构

主从架构：HDFS/Hbase



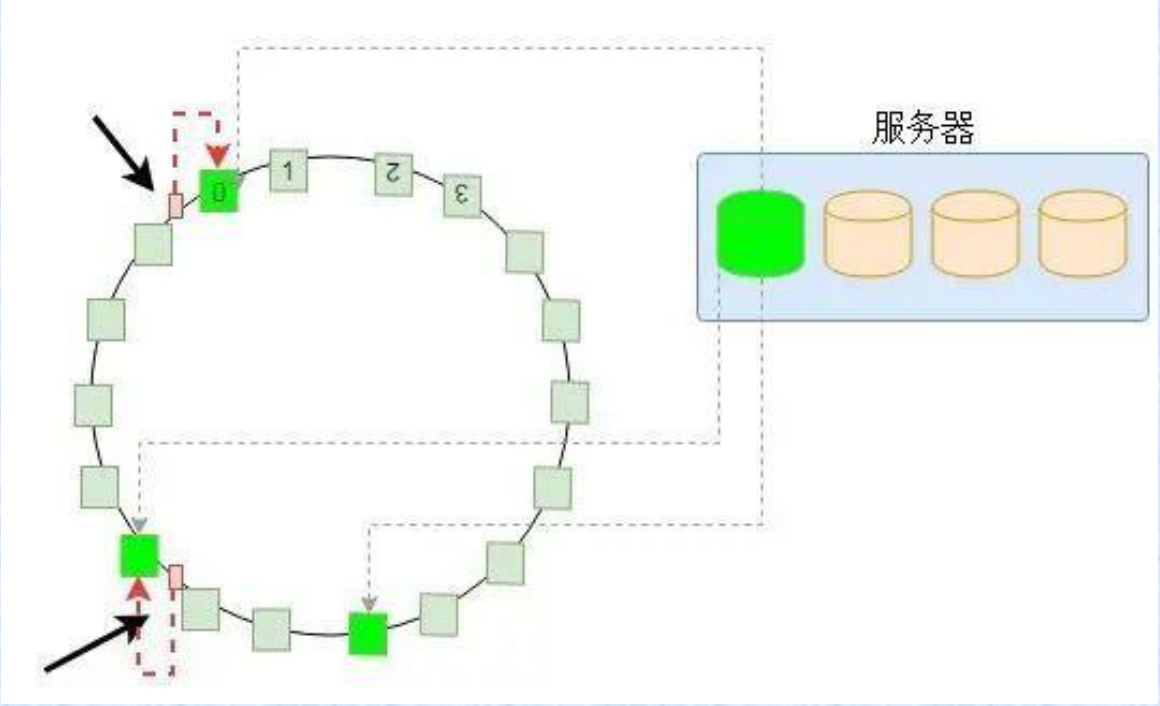
去中心架构：Ceph存储系统

该架构没有中心节点。客户端是通过一个设备映射关系计算出来写入数据的位置，直接与存储节点通信，从而避免中心节点的性能瓶颈。



去中心架构（一致性哈希）：Swift系统

该架构没有中心节点。客户端通过一致性哈希方式获得数据存储位置。一致性哈希的方式就是将设备位置做成一个哈希环，然后根据数据参数计算出的哈希值映射到哈希环的某个位置，从而实现数据的定位。

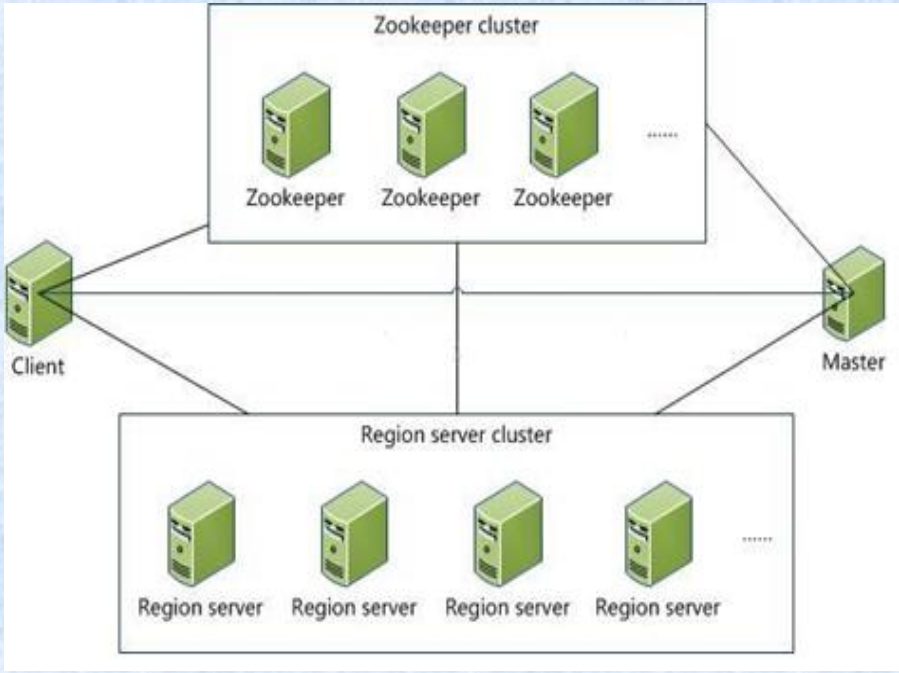


Hbase集群部署

- 物理部署：Hadoop集群
- 软件部署：Hadoop/HDFS/Hbase

四大组件

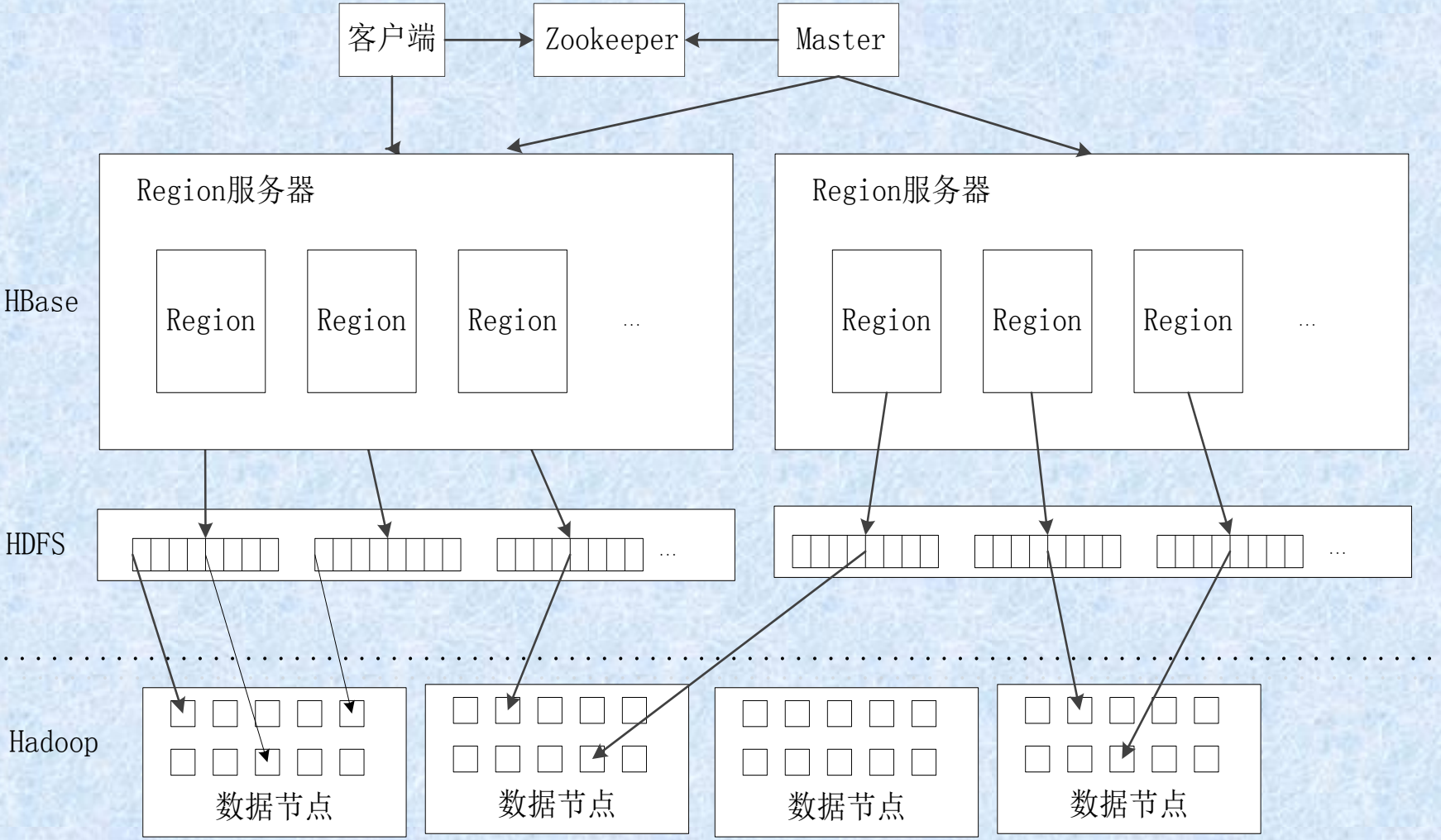
- Master
- Region Server
- Zookeeper
- Client



Hbase系统架构

- Hadoop基础平台提供了计算结构
- HDFS提供了底层数据物理存储结构
- Hbase提供了上层数据逻辑存储结构
  - Master节点管理着整个HBase集群
  - Region Server管理多个regions并提供数据访问服务
  - 客户端提供了数据库访问接口
  - Zoopkeeper负责分布式协调服务（Hadoop平台提供）

10.3 分布式存储架构


客户端
Zookeeper
Master
Region服务器
Region服务器
HBase
Region
Region
Region
…
Region
Region
Region
…
HDFS
…
…
Hadoop
数据节点
数据节点
数据节点
数据节点

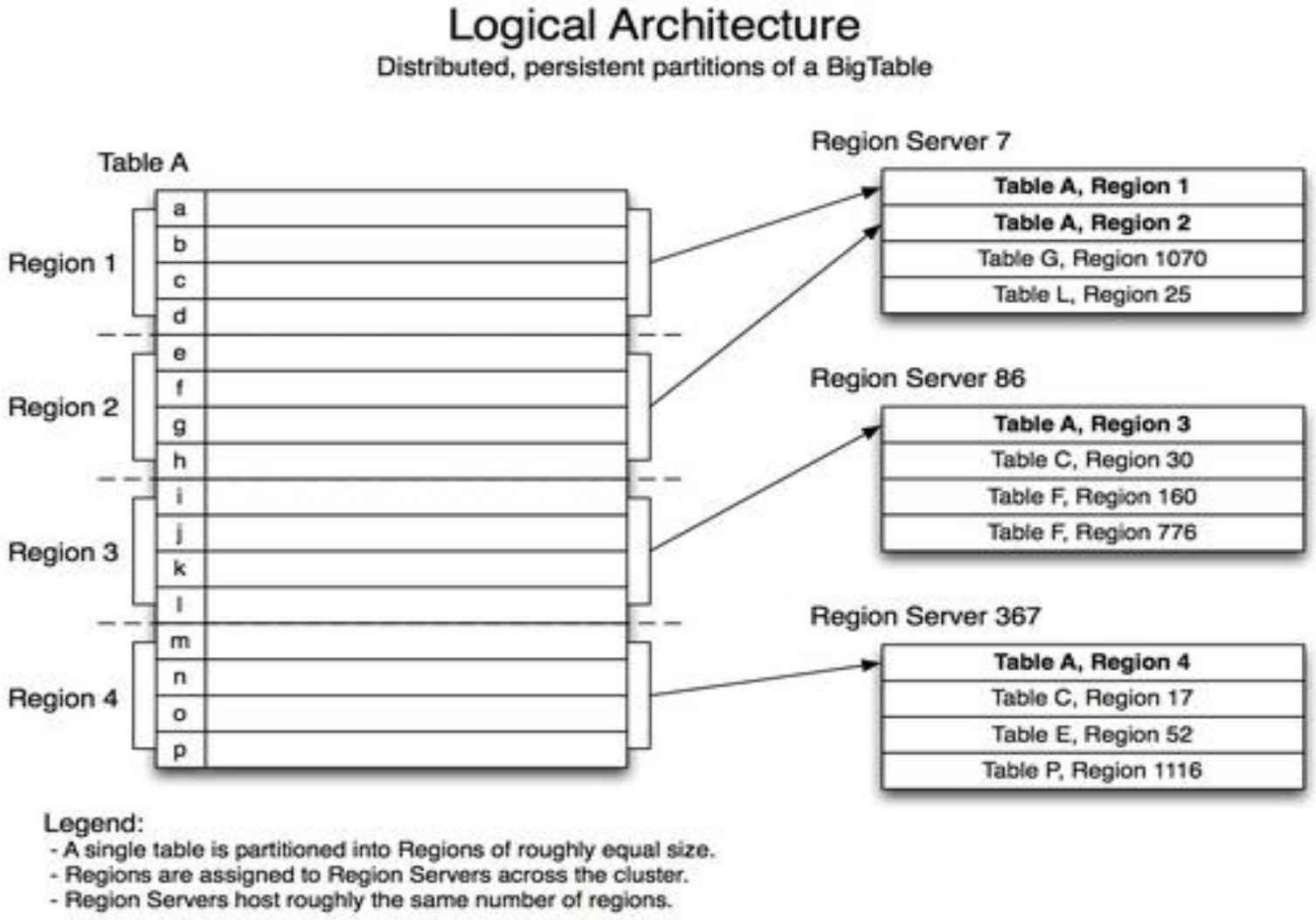
Hbase相关基本概念

- Region
- Store
- HFile

Region

- 是将数据表按照RowKey划分形成的子表
- 是数据表在集群中存储的最小单位
- 可以被分配到某一个Region Server进行存储管理
- 各个Region Server存放的Region数目大致相同，以达到负载均衡的目的
- Region内部包含一个HLog日志和多个Store，数据实际上是存储在Store单元中

10.3 分布式存储架构


Logical Architecture
Distributed, persistent partitions of a BigTable
Table A
Region 1
Region 2
Region 3
Region 4
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
Region Server 7
Table A, Region 1
Table A, Region 2
Table G, Region 1070
Table L, Region 25
Region Server 86
Table A, Region 3
Table C, Region 30
Table F, Region 160
Table F, Region 776
Region Server 367
Table A, Region 4
Table C, Region 17
Table E, Region 52
Table P, Region 1116
Legend:
- A single table is partitioned into Regions of roughly equal size.
- Regions are assigned to Region Servers across the cluster.
- Region Servers host roughly the same number of regions.

Region

- 是将数据表按照RowKey划分形成的子表
- 是数据表在集群中存储的最小单位
- 可以被分配到某一个Region Server进行存储管理
- 各个Region Server存放的Region数目大致相同，以达到负载均衡的目的
- Region内部包含一个HLog日志（WAL类型）和多个Store，数据实际上是存储在Store单元中

Region = HLog + m*Store

Store = MemStore + n*StoreFile

Store

- Region内部按照列簇分为不同的Store
- 每个Store由一个memStore和多个StoreFile组成
- memStore是内存中的一个缓存区
- StoreFile是写到硬盘上的数据文件
- 数据首先会放入MemStore中，当MemStore满了以后会清空形成一个新StoreFile
- 检索数据时，先在memStore找，然后找StoreFile

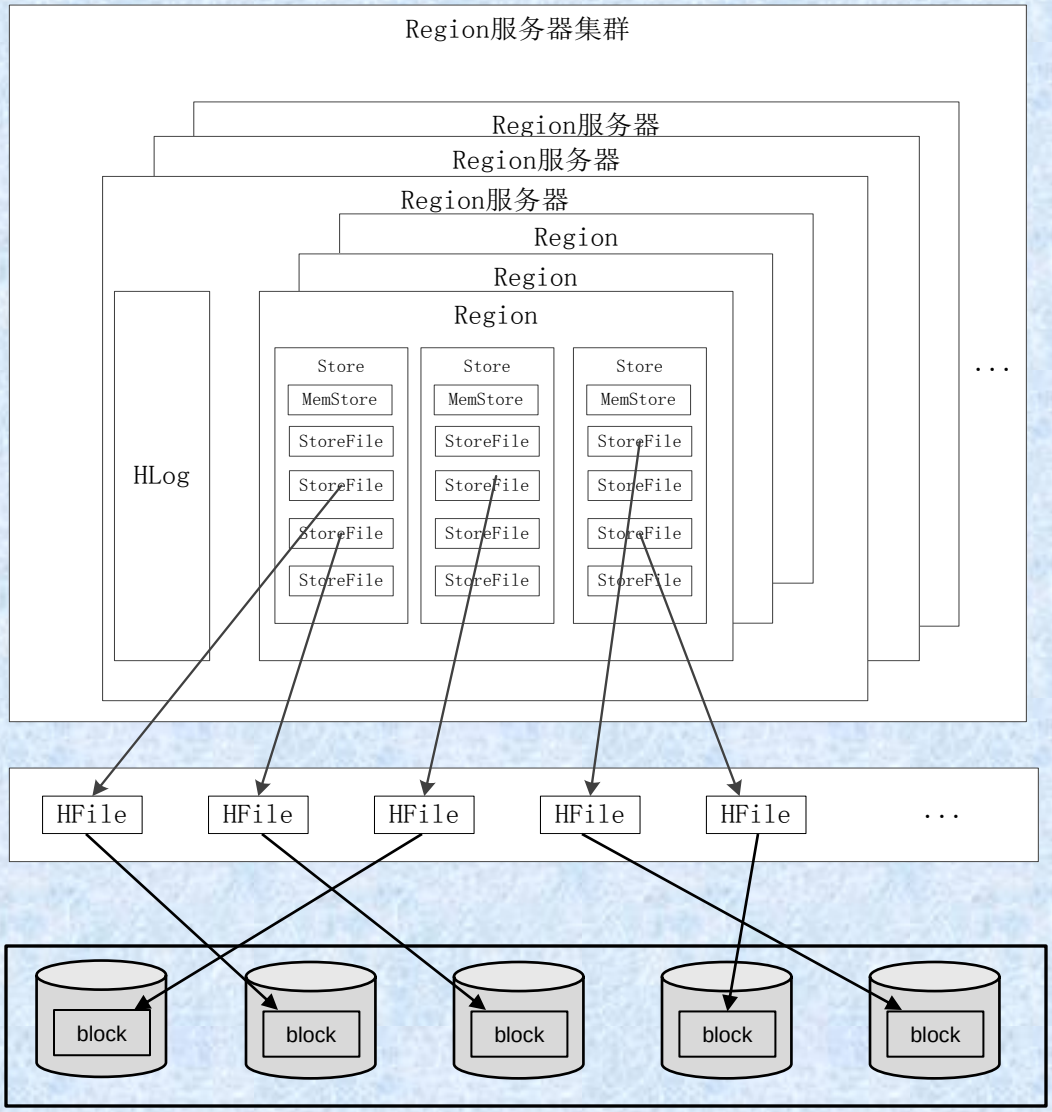
Store操作

- compact操作：
  - 当StoreFile文件数量增长到一定阈值时触发
  - 将多个StoreFile合并成一个StoreFile
  - 在合并过程中会进行StoreFile版本合并和数据删除。
- split操作：
  - 当单个StoreFile大小超过一定阈值后触发
  - 把当前的Region分裂成2个子Regions
  - 子Region会被Master分配到相应的Region Server上
  - 是HBase提供的负载均衡机制

HFile

- StoreFile包含的一个HFile文件
- 是Hadoop的二进制格式文件
- StoreFile是HFile的轻量级包装，数据最终是以HFile的形式存储在Hadoop平台上
- 采用一个简单的byte数组存储数据的每个KeyValue对
- 这个byte数组里面包含了很多项，有固定的格式，每项有具体的含义。

10.3 分布式存储架构



Hbase 数据表(逻辑存储结构)

HDFS文件 (物理存储结构)

Block数据块 (存储单元)

Hbase数据模型与存储模式

- HBase表特性：
  - 面向列的、稀疏的、分布式的、持久化存储的多维排序映射表
- Hbase表索引：
  - 行关键字、列簇名、列关键字及时间戳
- Hbase表值形式：
  - 一个未经解析的byte数组

Hbase数据模型与存储模式

- Hbase数据模型
  - 以表的形式存储数据
  - 表由行和列族组成
  - 一个表可包含若干个列族
  - 一个列族内可用列限定符来标志不同的列
  - 存于表中单元的数据尚需打上时间戳

Hbase数据模型与存储模式

- Hbase数据模型基本元素
  - 表
  - 行键
  - 列族
  - 单元格
  - 时间戳

Hbase数据模型与存储模式

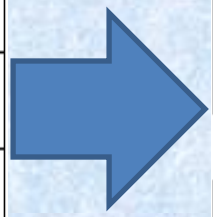
- Hbase存储逻辑视图：
  - 一个三元组（行键，列族:列限制符，时间戳）可以唯一地确定存储在单元（Cell）中的数据
  - Key是一个三元组（行键，列族:列限制符，时间戳）
  - Value就是这个三元组定位的数据值

Row Key	Time Stamp	ColumnFamily contents	ColumnFamily anchor	ColumnFamily people
"com.cnn.www"	t9		anchor:annsi.com ="CNN"	
"com.cnn.www"	t8		anchor:my.look.c a="CNN.com"	
"com.cnn.www"	t6	contents:html= "<html>..."		
"com.cnn.www"	t5	Contents:html= "<html>..."		
"com.cnn.www"	t3	contents:html= "<html>..."		

Hbase数据模型与存储模式

- Hbase存储物理视图：
  - 一个列族对应生成一个Region

Row Key	Time Stamp	ColumnFamily contents	ColumnFamily anchor	ColumnFamily people
"com.cnn.www"	t9		anchor:annsi.com ="CNN"	
"com.cnn.www"	t8		anchor:my.look.c a="CNN.com"	
"com.cnn.www"	t6	contents:html= "<html>..."		
"com.cnn.www"	t5	Contents:html= "<html>..."		
"com.cnn.www"	t3	contents:html= "<html>..."		



行键	时间戳	列族 contents
"com.cnn.www"	t6	contents:html="<html>..."
	t5	contents:html="<html>..."
	t3	contents:html="<html>..."

行键	时间戳	列族 anchor
"com.cnn.www"	t9	anchor:cnnsi.com="CNN"
	t8	contents:my.lool.ca="CNN.com"

Hbase数据模型与存储模式

- Hbase物理存储
  - 表划分出的列族对应着物理存储区的Region
  - 列族所包含的列对应着的存储区Region所包含的Store
  - 当增大到一个阀值的时候，Region就会等分成两个新的Region

Hbase寻址机制

- 三层机构：
  - Zookeeper文件
  - -ROOT-表
  - .META.表
- 客户端从Zookeeper获得Region的存储位置信息后，直接在Region Server上读写数据
- 流程：Zookeeper→-ROOT-表→.META.表→找到存放用户数据的Region Server位置

Hbase扫描读取数据

- 所有的存储文件被划分成若干个存储块
- 存储块在get或scan操作时会加载到内存中
- HBase顺序地读取一个数据块到内存缓存中
- 再读取相邻数据时从内存中读取而不需要读磁盘

Hbase写数据

- Client向Region Server提交写数据请求；
- Region Server找到目标Region；Region检查数据是否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；
- 将数据更新写入HLog（WAL），只有HLog写入完成之后，commit()才返回给客户端；
- 将数据更新写入MemStore；
- 判断MemStore的是否需要flush为StoreFile，若是，则flush生成一个新StoreFile;
- StoreFile数目增长到一定阈值，触发compact合并操作，多个StoreFile合并成一个StoreFile，同时进行版本合并和数据删除；
- 若单个StoreFile大小超过一定阈值，触发split操作，把当前Region拆分成2个子Region，原来的Region会下线，新分出的2个子Region会被Master重新分配到相应的Region Server上

Hbase更新表

- 首先写入HLog和MemStore
- MemStore中的数据是排序的
- 当MemStore累计到一定阈值时：
  - 创建一个新的MemStore，
  - 将老的MemStore添加到flush队列，由单独的线程刷写到磁盘上，成为一个新StoreFile
  - 系统在HLog中记录一个检查点，表示这个时刻前的变更已持久化

Hbase预防数据丢失

- 每个Region服务器都有一个自己的HLog 文件
- 每次启动都检查HLog文件，确认最近一次执行缓存刷新操作之后是否发生新的写入操作
- 发现更新时：
  - 写入MemStore
  - 刷写到StoreFile
  - 删除旧的Hlog文件，开始为用户提供服务

StoreFile合并与分裂

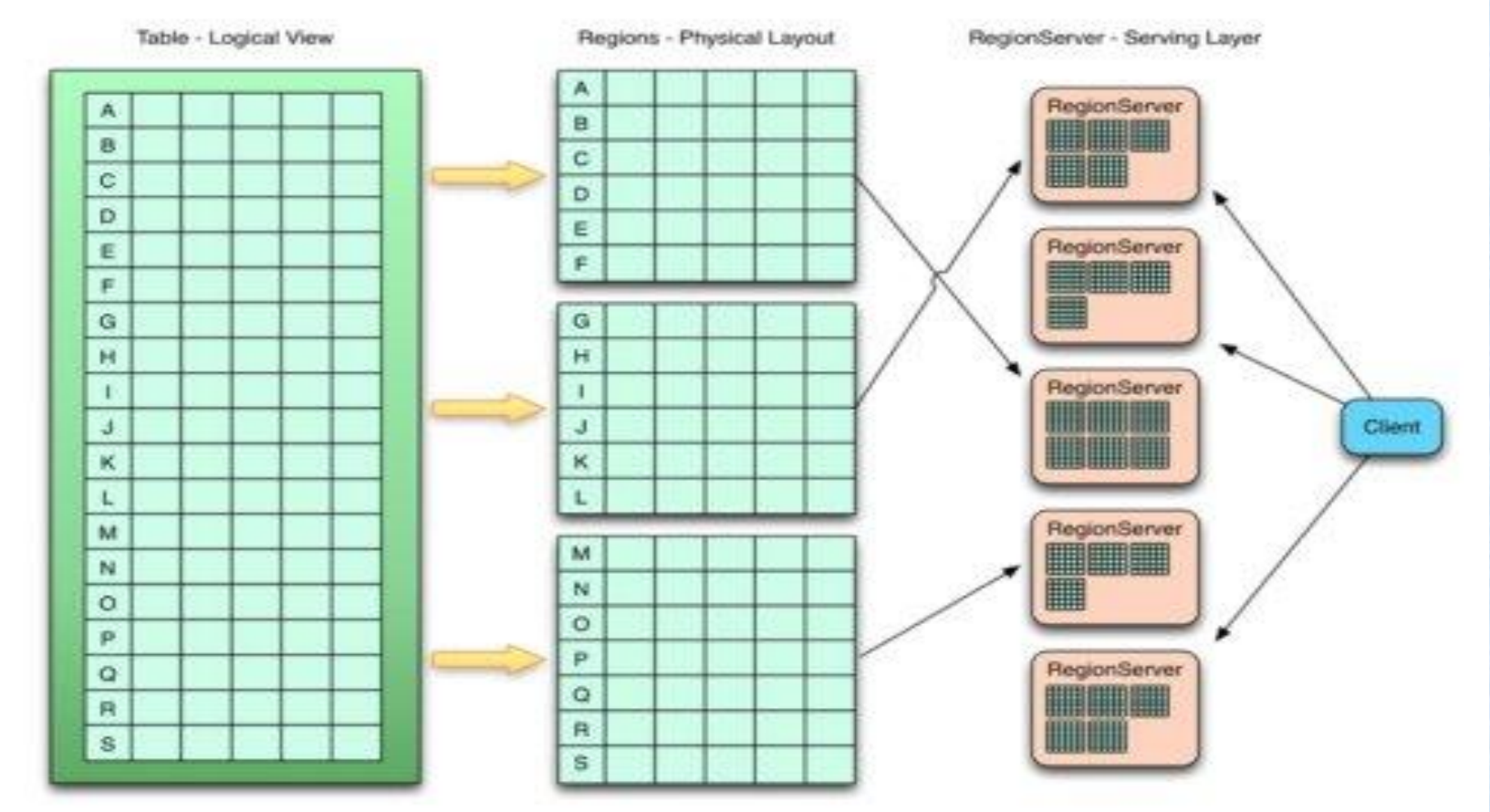
- 合并：
  - 时机：当一个Store中的StoreFile达到一定的阈值时
  - 操作：将同一个key的修改合并到一起，形成一个大的StoreFile
- 分裂：
  - 时机：当StoreFile的大小达到一定阈值后
  - 操作：等分为两个StoreFile。

HBase索引与检索

- 表存储模型
- HBase的三种查询方式
  - 基于单个RowKey的查询
  - 通过一个RowKey的区间来访问
  - 全表扫描
- 二次索引表技术

10.3 分布式存储架构

数据表分块存储



HBase的三种查询方式

1）基于单个RowKey的查询：只利于已知行键（RowKey）抽取一条数据项（data record）的查询

2）通过一个RowKey的区间来访问：一次性读取一个子表（数据块）

3）全表扫描：搜索不知RowKey的数据项、或者读取数据的某类属性（统计分析中常常用到）

结论：基于行键（RowKey）的搜索方式不利于读取数据的某类属性值（列），延迟高、速度慢、效率低，浪费计算资源

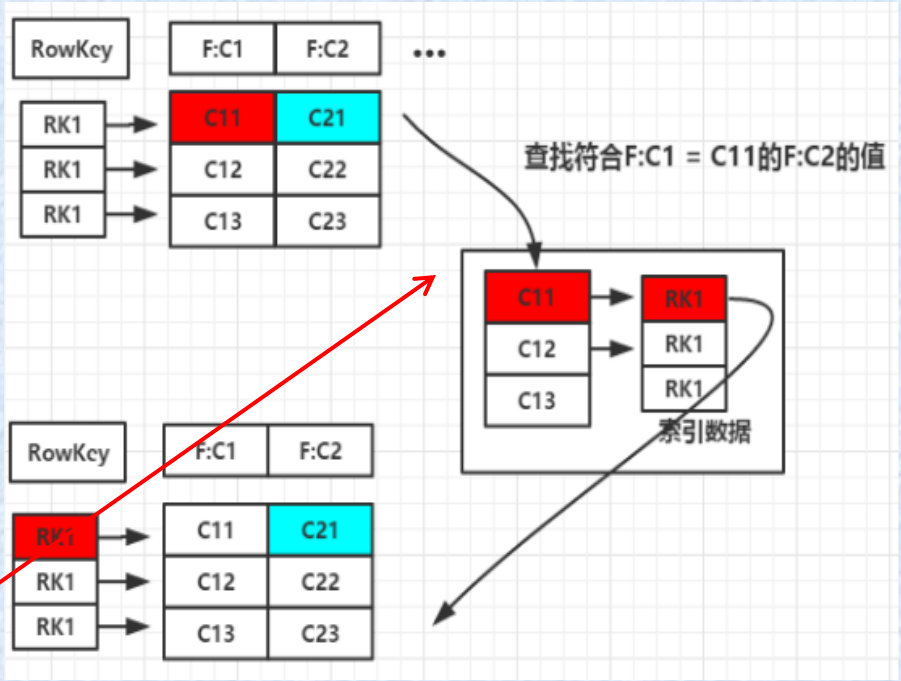
Hbase索引与检索

二次索引表机制

问题：如何根据C11的值，
在表中找到C21的值？
（不知道RK1）

原始方法：

1）全表扫描找到C11
2）根据C11的位置找到RK1
3）再根据RK1的值找到C21



索引表解决办法：

1）建立一个索引表F:C1->RK
2）现根据C11从索引表查到RK1
3）再使用RK1回到主表找到C21

二次索引表机制（续）

- 关键原理：建立主表列到RowKey的逆向映射关系
  - 成本：索引表占用额外空间，多一级搜索
  - 收益：避免了全表搜索，大大提高搜索效率
- 实现技术
  - 表索引
    主表的索引列值为索引表的RowKey，
    主表的RowKey做为索引表的Qualifier或Value
  - 列索引
    增加一个单独列族存储索引值
    主表的用户数据列值做为索引列族的Qualifier
    用户数据Qualifier做为索引列族的列值

实现方案：三大组件

- Zookeeper提供分布式协同服务
- Oozie 提供作业调度和工作流执行
- YARN 提供集群资源管理服务

分布式协同管理组件Zookeeper

- 提供服务
  - 统一命名服务
  - 应用配置管理
  - 分布式锁服务
  - 分布式消息队列
- 架构：主从架构

分布式协同管理组件Zookeeper

- Zookeeper服务由一组Server节点组成
  - 每个节点上运行一个Zookeeper程序
- 每个server维护内容：
  - 自身的内存状态镜像、持久化存储的事务日志和快照
- ZooKeeper集群的数量一般为奇数
- 有过半Server可用，整个系统即保持可用性。

分布式协同管理组件Zookeeper

- 节点角色
  - Leader
  - Follower
  - Observer

分布式协同管理组件Zookeeper

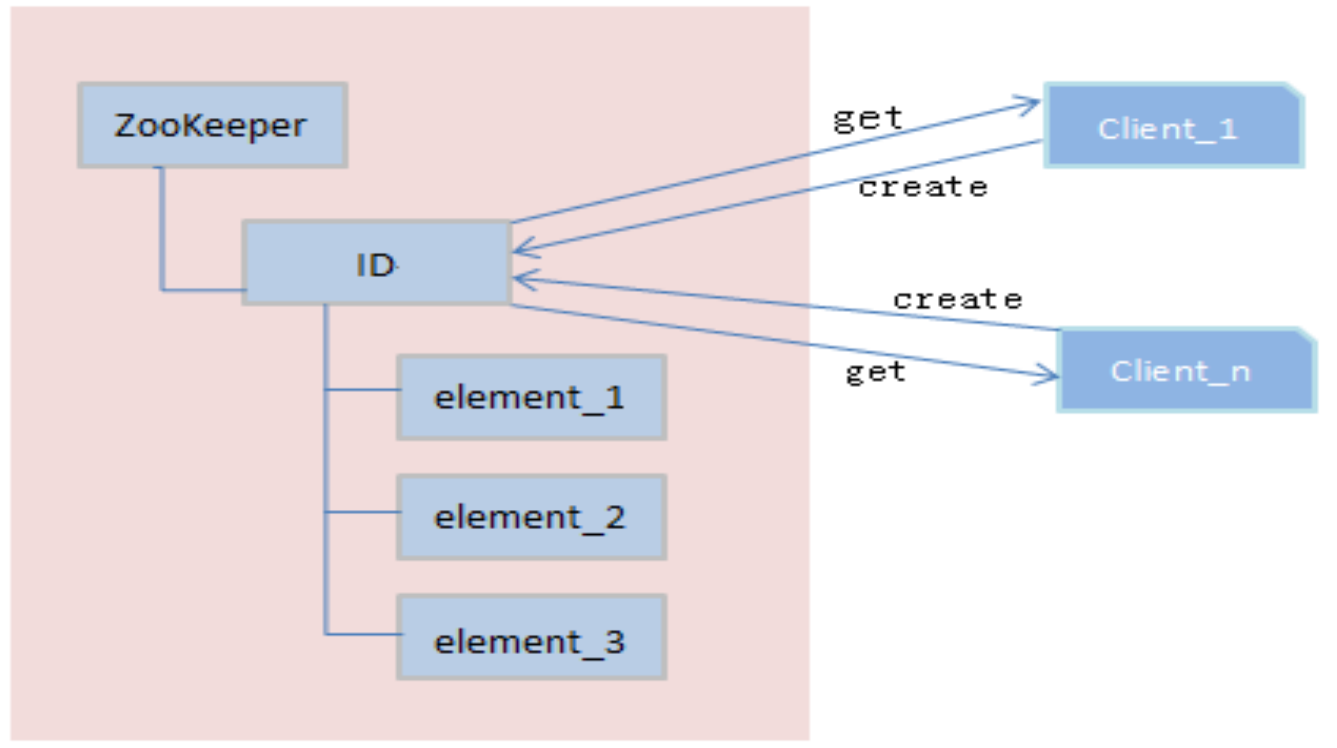
- 失效处理机制
  - Zookeeper作出快速响应
  - 消息层基于Fast Paxos算法重新推举一个Leader，继续作为协调服务中心处理客户端的写数据请求，并将ZooKeeper协同数据的变更同步（广播方式）到其他的Follower节点

分布式协同管理组件Zookeeper

- 业务流程
  - 客户端Client连接到Follower发出写数据请求
  - 请求发送到Leader节点
  - Leader完成元数据更新
  - Leader上的数据同步更新到其他Follower节点

分布式协同管理组件Zookeeper

- 统一命名服务
  - 把各种服务名称、地址、及目录信息存放在分层结构中供需要时读取
  - 提供一个分布式序列号生成器
  - 流程



分布式协同管理组件Zookeeper

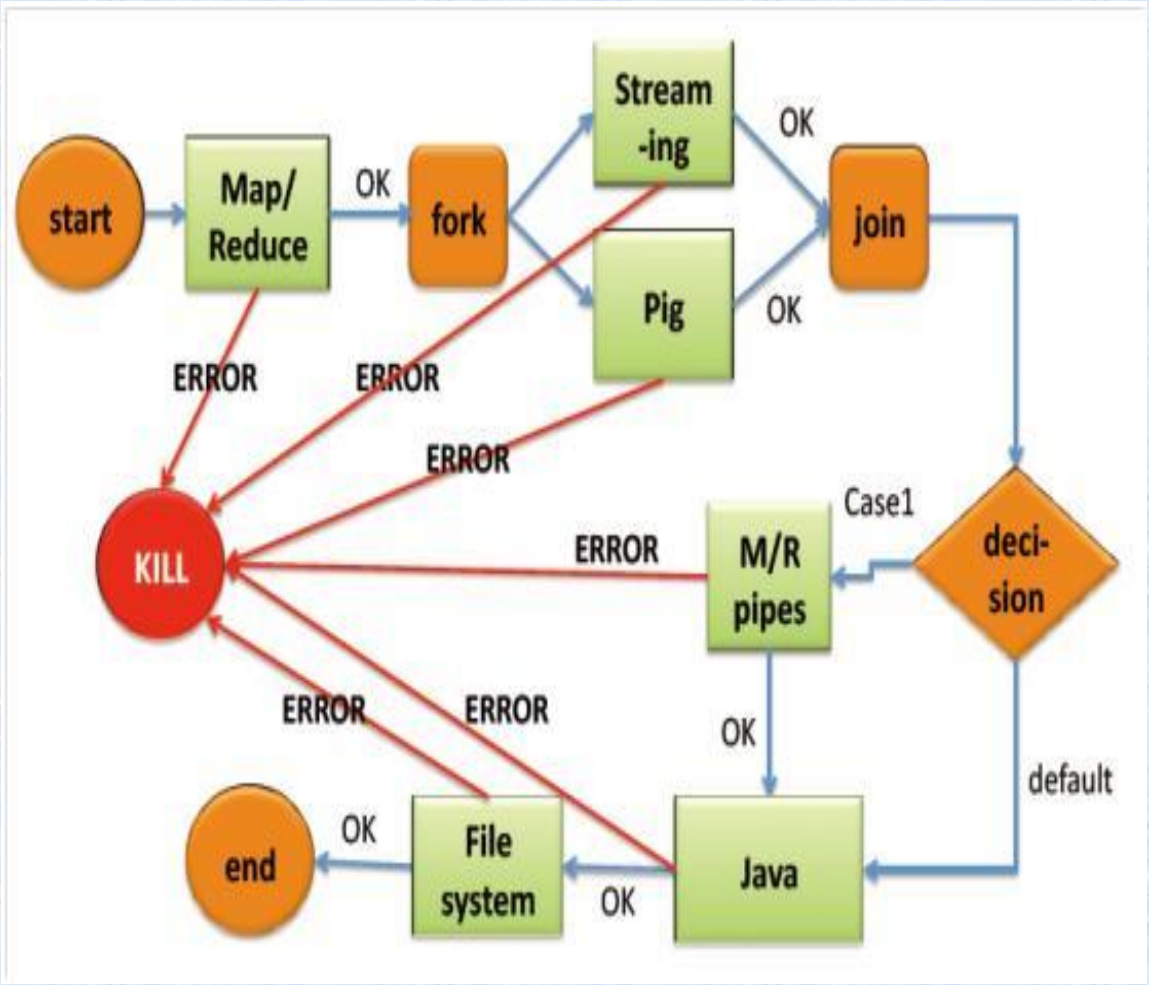
- 配置管理服务
  - 发布（publish）和订阅（watch）模式
- 分布锁的实现
  - 独占锁和控制时序锁
- 分布式消息队列
  - 同步队列和FIFO队列

作业调度与工作流引擎Oozie

- 核心功能：
  - 工作流：定义作业任务的拓扑和执行逻辑
  - 协调器：负责工作流的关联和触发分布式消息队列
- 工作流包括：
  - 控制流节点：定义工作流的开始和结束，控制执行路径
  - 动作节点：支持不同任务类型

作业调度与工作流引擎Oozie

- 工作流流节点：
- 启动控制节点
- 末端控制节点
- 停止控制节点
- 决策控制节点
- 分支-联接控制节点



集群资源管理框架YARN

- 优势：
  - 允许多个应用程序运行在一个集群上，并将资源按需分配给它们，这大大提高了集群资源利用率
  - YARN允许各类短作业和长服务混合部署在一个集群中，并提供了容错、资源隔离及负载均衡等方面的支持，这大大简化了作业和服务的部署和管理成本，强化了对应用程序的支持

集群资源管理框架YARN

- 体系架构——Master/Slave架构：
  - Master为YARN的Resource Manager
  - Slave为NodeManager
  - Application Master
  - Container
  - YARN Client

集群资源管理框架YARN

- 部署方式：
  - Resource Manager：部署并运行在NameNode上
  - Node Manager：部署在每个DataNode上，作为Resource Manager的节点代理；
  - 每个DataNode都包含一个或多个多个Container用于资源调度
  - 每一个提交给Hadoop集群的Application都有一个Application Master与之对应，运行在某个DataNode上

Hadoop/HDFS/Hbase存储系统总结

- 大数据存储架构：分布式文件系统HDFS，HBase分布式数据库，行存储vs.列存储
- HDFS底层存储结构：Namenode/Datanode，分片（partition），数据块（block），冗余备份（replica），机架感知备份存放
- HBase分布式存储结构：
  - 逻辑存储结构：key-vlaue键值对、三元组（行键、列族：列限制符、时间戳）、Hbase数据表
  - 物理存储结构：Region，Store，HFile
- HBase索引与检索：二次索引表设计，技术解决方案