

HBase

1、HBase简介

高可靠、高性能、面向列、可伸缩的分布式存储系统，分布式数据库。可以用来存储非结构化和半结构化的松散数据

为什么要去设计HBase数据库？

内部切分、水平扩展都是后台全自动实现，具有非常好的水平可扩展性

数据结构多变，HBase能满足不断增长的数据存储需求

2、HBase数据模型

HBase是一个稀疏的多维度的排序映射表



列族

支持动态扩展

保留旧的版本（由于HDFS只支持写入操作，不允许修改）

列限定符

列限定符



name



major



email

单元格（具体存储数据的地方）

	Info		
	name	major	email
201505001	Luo Min	Math	luo@qq.com
201505002	Liu Jun	Math	liu@qq.com
201505003	Xie You	Math	xie@qq.com you@163.com

行键

列限定符

列族

单元格

ts1

ts2

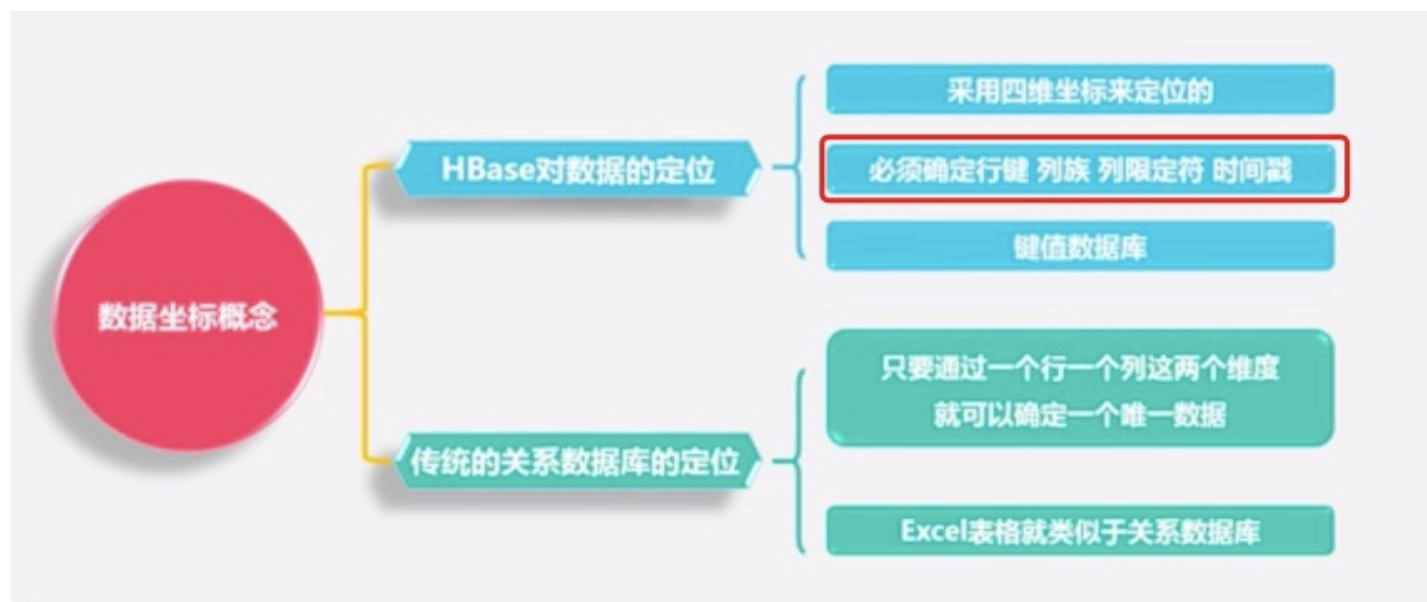
该单元格有2个时间戳ts1和ts2
每个时间戳对应一个数据版本
ts1=1174184619081 ts2=1174184620720

存储的数据类型都是未经解释的字符串

时间戳：

在一个单元格里面，旧的版本会被保存，新的版本会通过时间戳来进行区分

数据坐标概念



面向列的存储

便于数据分析

3、HBase实现原理

HBase的功能组件

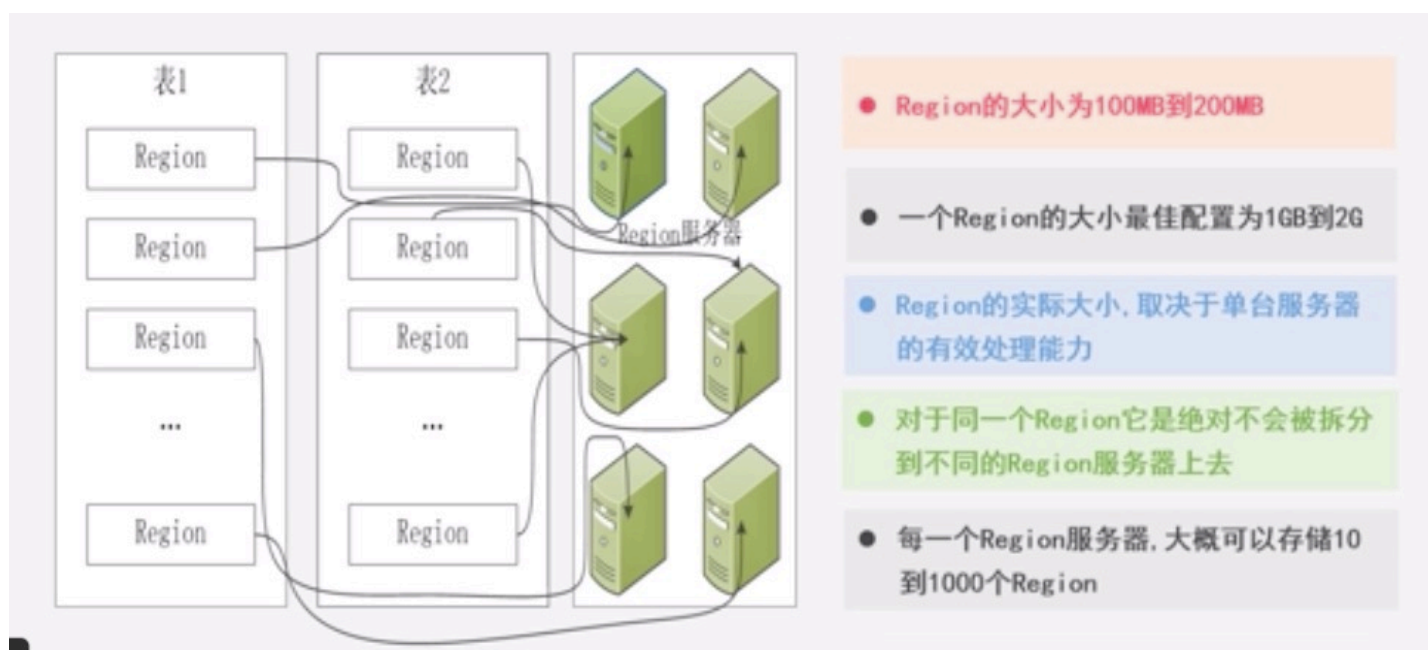


Master服务器



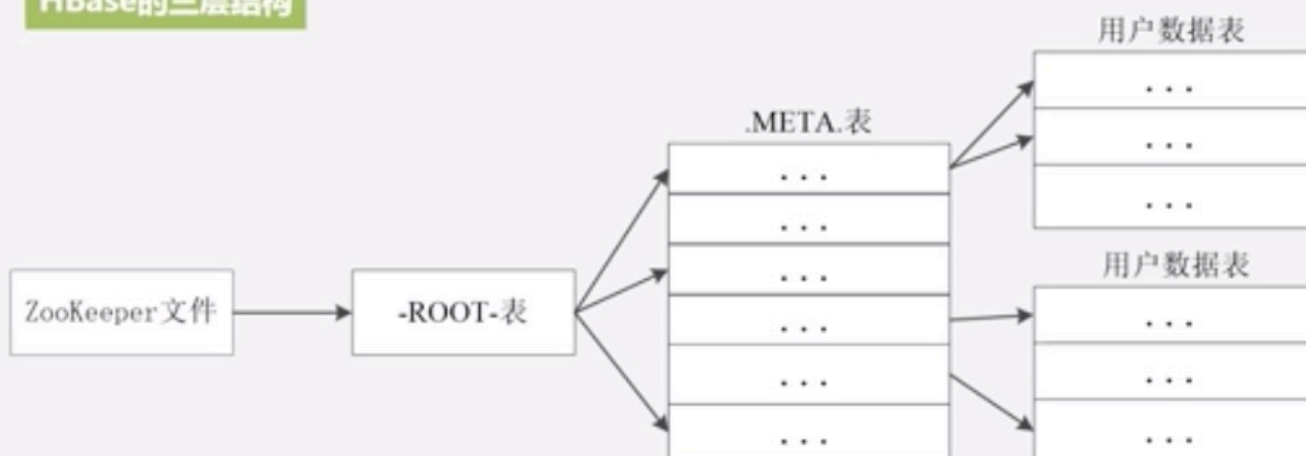
Region服务器

具体负责数据的存储



HBase三层结构

HBase的三层结构



-ROOT-表：存取的是元数据表的元数据信息被存放到哪里

元数据的.META.表：负责存储具体数据存取到哪些服务器上

.META.表的全部Region都会被保存在内存中

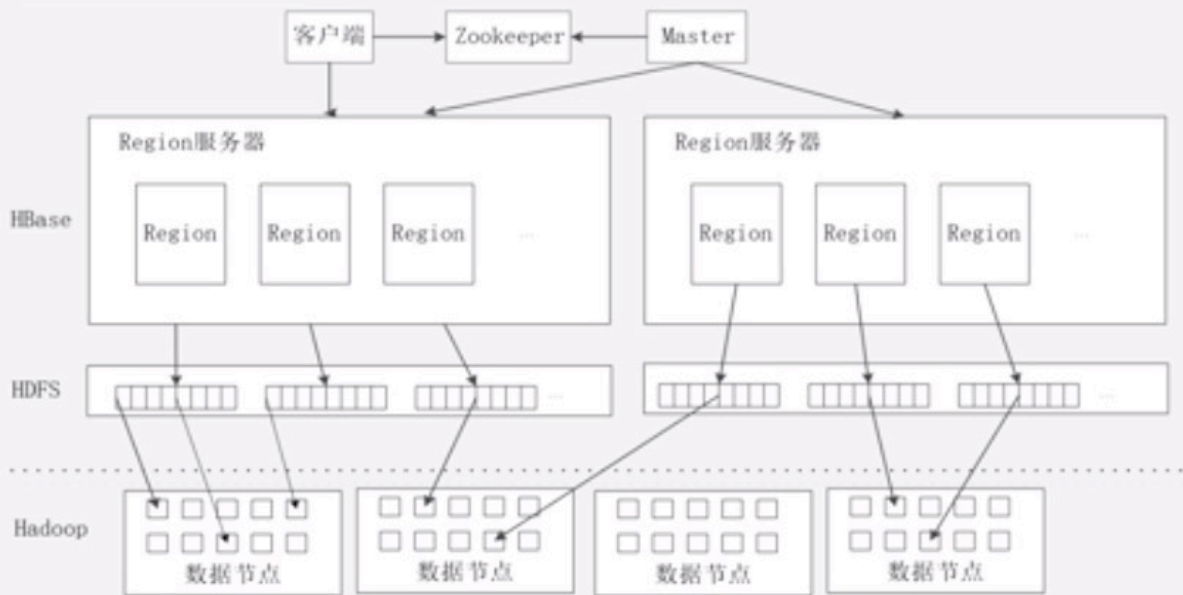
HBase的三层结构中各层次名称和作用

层次	名称	作用
第一层	Zookeeper文件	记录了-RROOT-表的位置信息
第二层	-ROOT-表	记录了.META.表的Region位置信息 -ROOT-表只能有一个Region。通过-RROOT-表， 就可以访问.META.表中的数据
第三层	.META.表	记录了用户数据表的Region位置信息。 .META.表可以有多个Region，保存了HBase中所有用户数 据表的Region位置信息

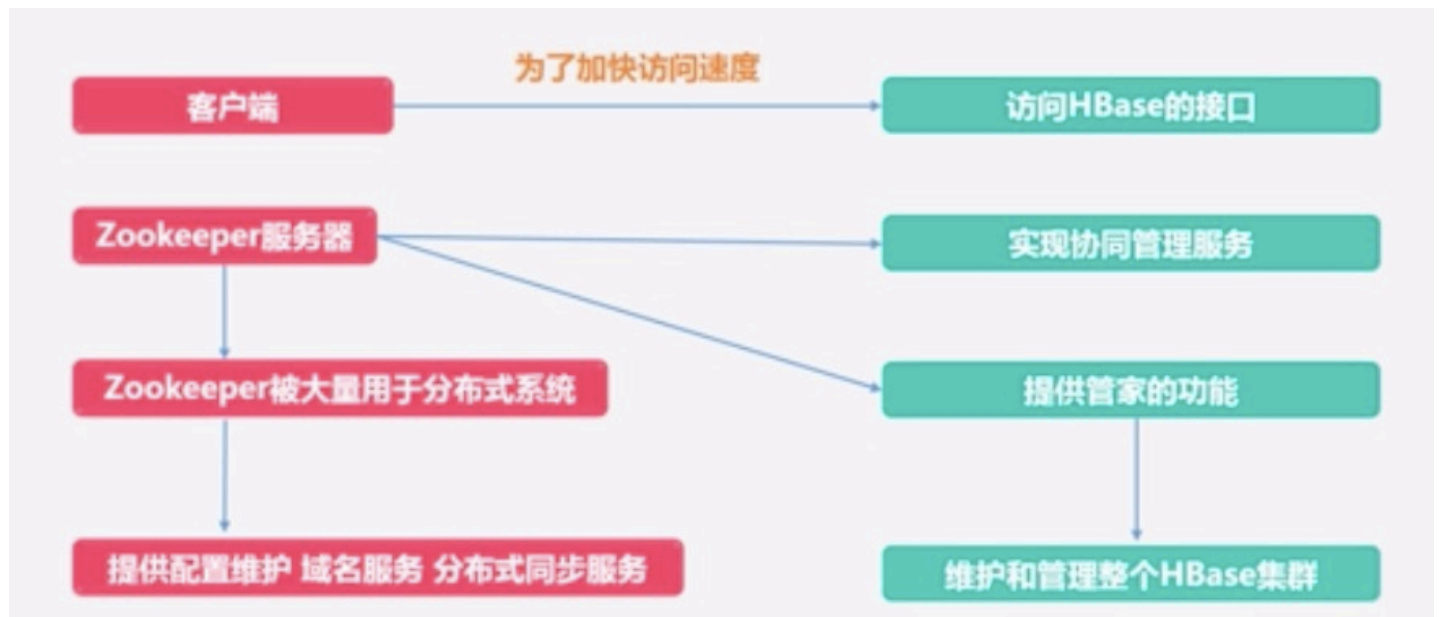
4、HBase运行机制

HBase系统架构

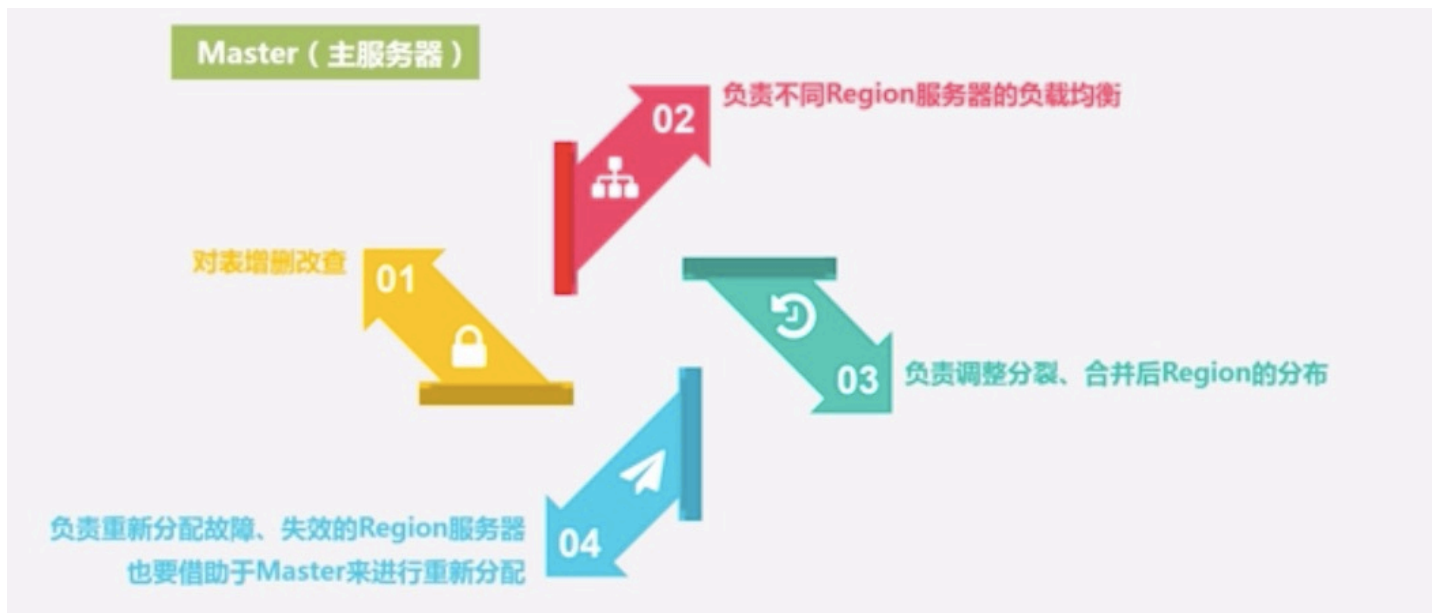
HBase的系统架构



客户端和Zookeeper



Master（主服务器）



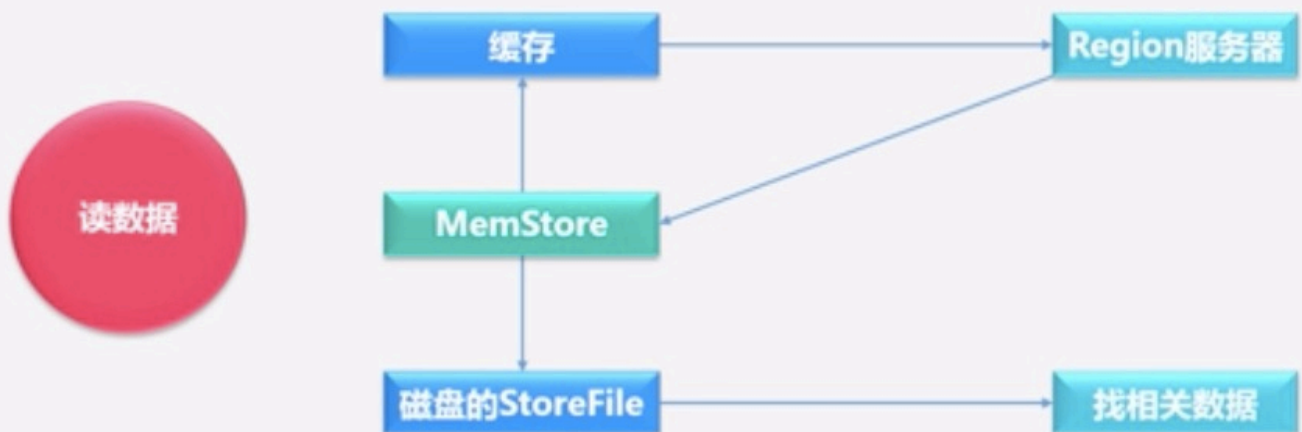
Region服务器

Region服务器向HDFS文件系统中读写数据

- 用户读写数据过程



用户读写数据过程



• 缓存的刷新

缓存的刷新

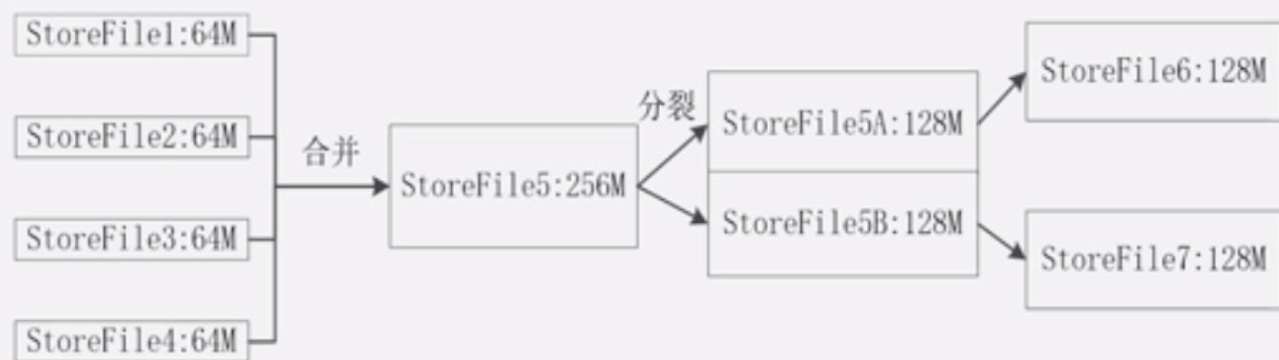


• StoreFile的合并和分裂==>即HBase的分裂操作

StoreFile的合并

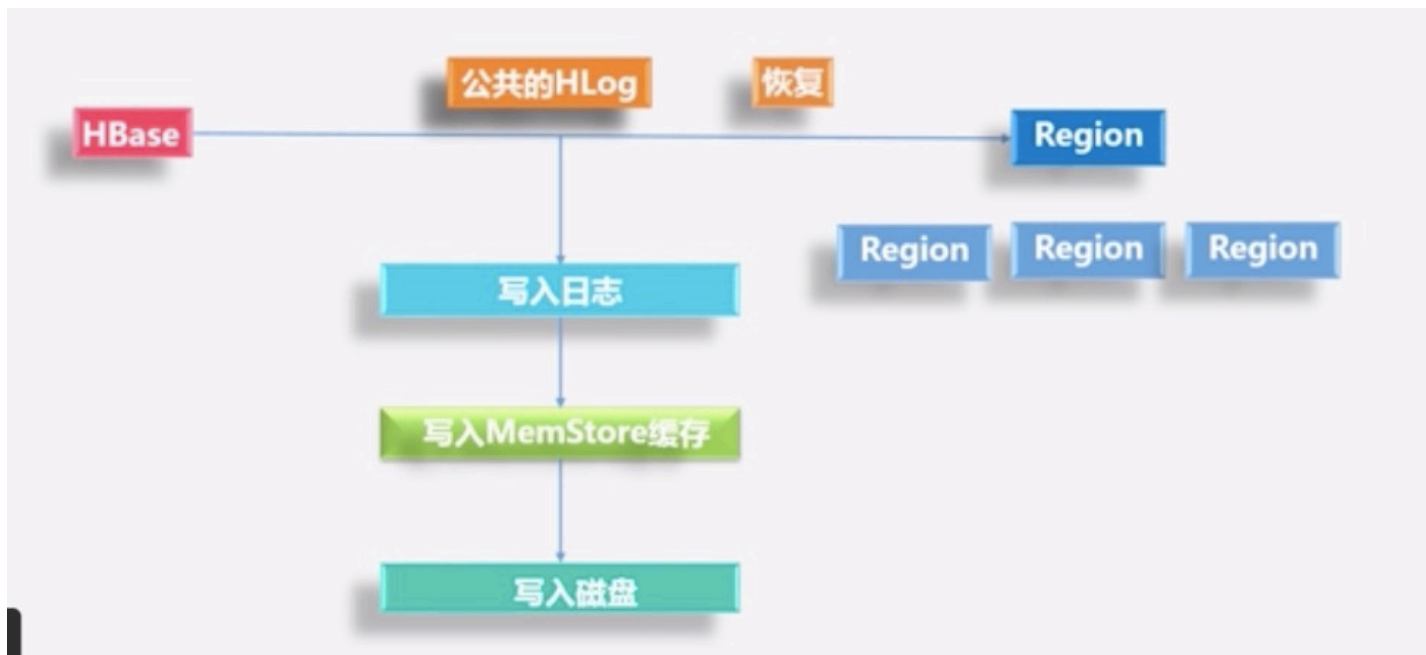


StoreFile的合并和分裂过程



- HLog的工作原理

通过日志的方法来恢复数据

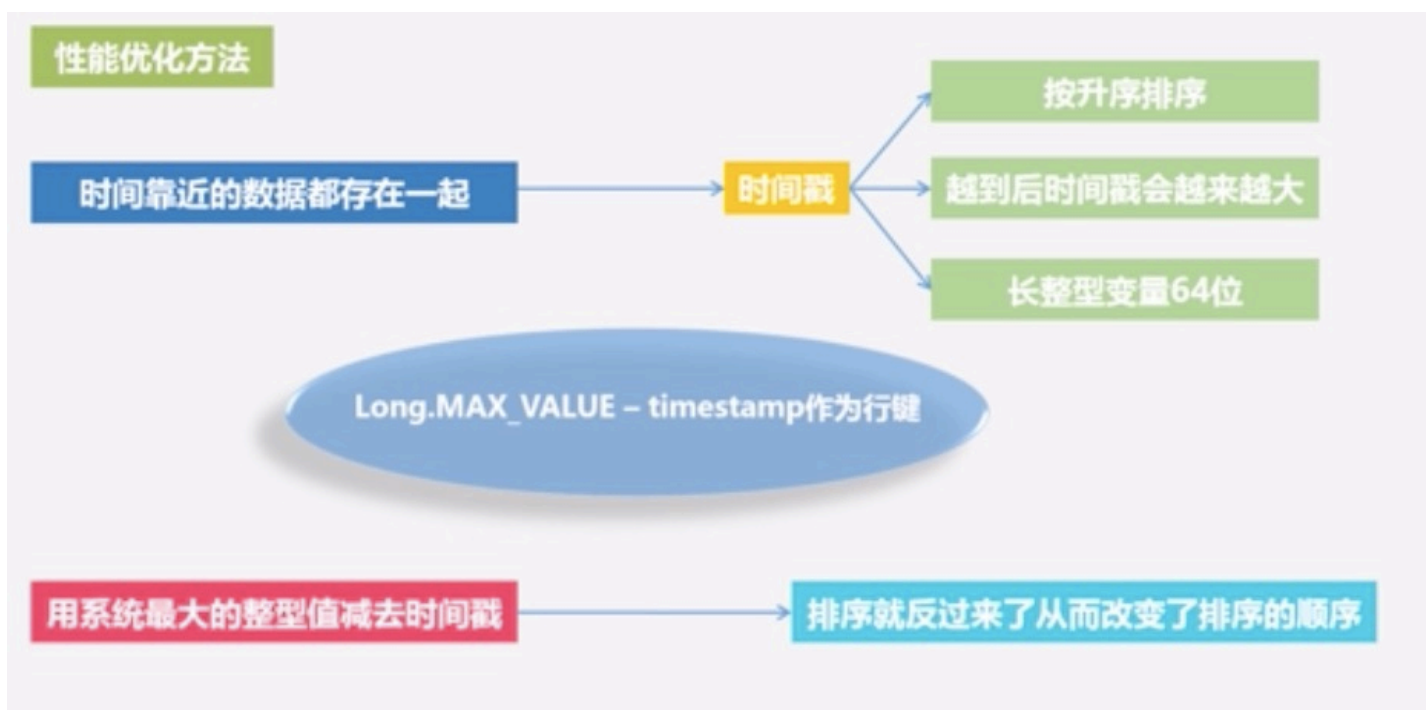


Zookeeper负责监视整个集群，告诉哪个服务器出现问题，并通知给相关的Master。则Master就会迁移故障服务器上的数据

5、HBASE应用方案

性能优化方法

- 时间戳



- 提升读写性能，将必要表方案Region服务器的缓存中

提升读写性能

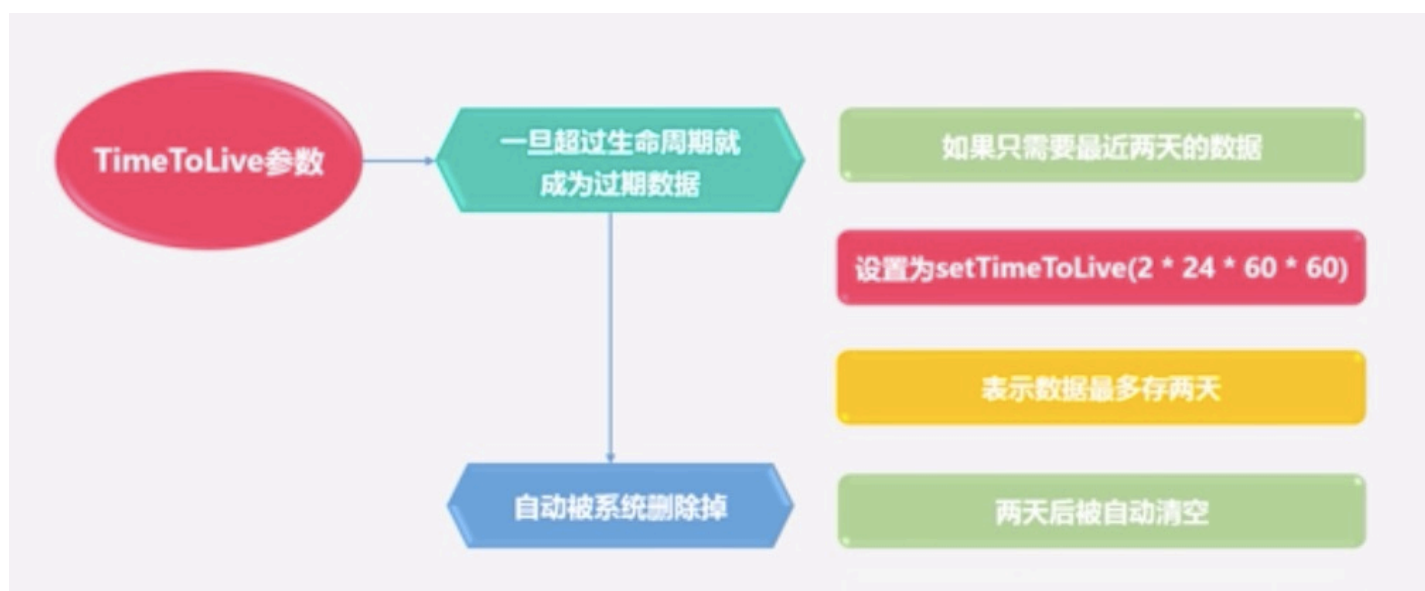


设置HColumnDescriptor.setInMemory选项为true



把相关的表放到Region服务器的缓存中,根据需要来决定是否放入缓存

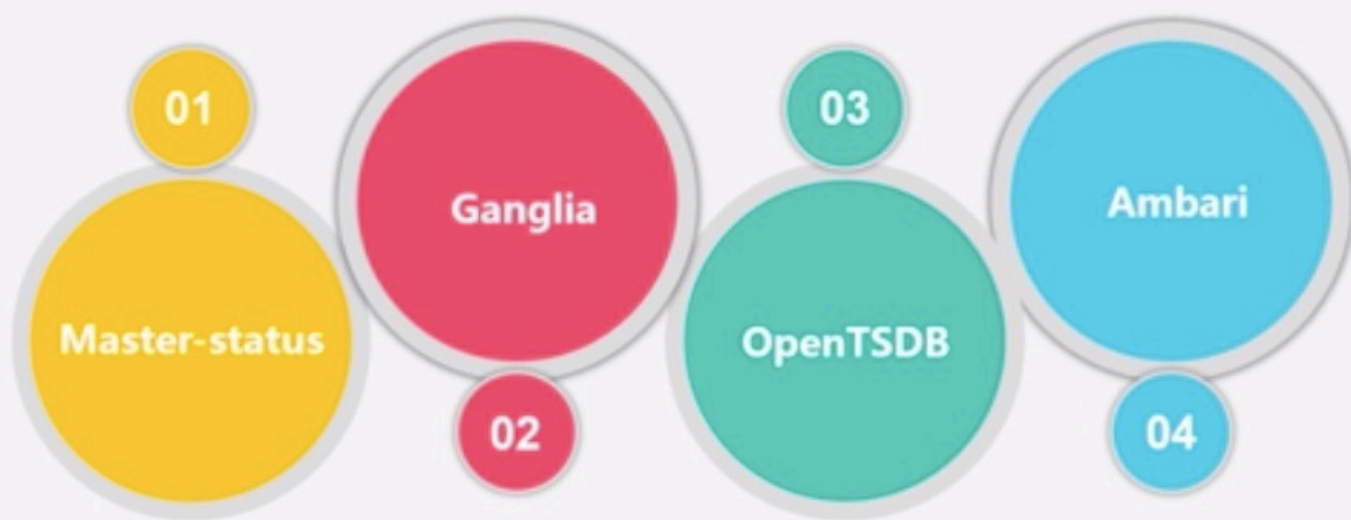
- 自动清除过期数据



HBase怎么检测性能?

性能检测相关工具

HBase怎么检测性能？

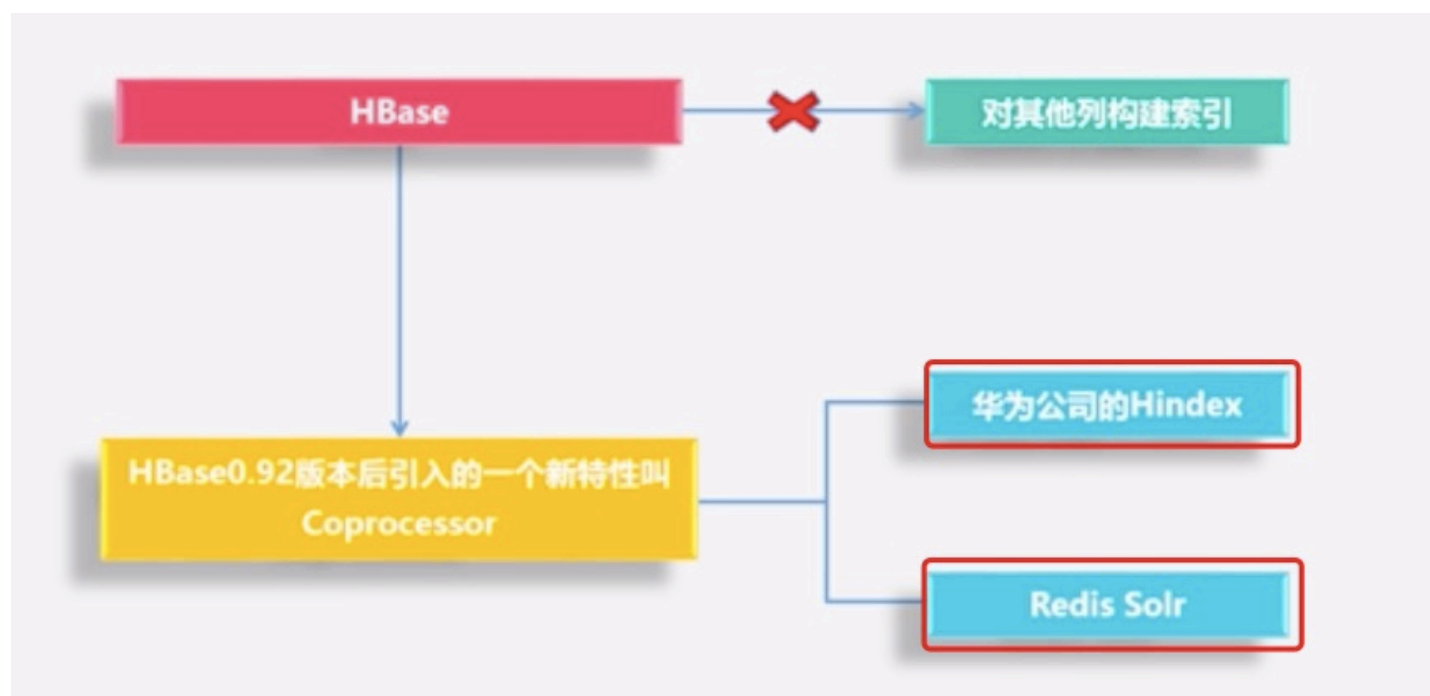


在HBase进行SQL查询工具



构建HBase二级索引

对不同的列进行索引



6、HBase安装与配置

注意

- 一个是JAVA_HOME这个环境变量，若没有配置，它会报错,因此要安装JDK
- 还有Hadoop要配置好,要实现无密码SSH登录,否则HBase实验会遇到障碍
- 启动HBase时候需要注意，要先去启动Hadoop,启动完之后，再去把HBase启动起来关闭的时候也是这样，先把HBase关掉,然后再去关Hadoop
- 修改配置文件的时候，里面有个选项叫hbase.managers.zk

这是配置Zookeeper的

7、Shell命令

Shell命令

create : 创建表

list命令 : 列出HBase中所有的表信息

创建表

create '表名', '列名', '列名', '列名'

如何添加数据--put

列名称可以添加时自定义

如何添加数据



一次只能为一个表的一行数据里的一个列添加数据



```
hbase(main):005:0> put 'tempTable', 'r1', 'f1:c1', 'hello, dlab'
0 row(s) in 0.0240 seconds

hbase(main):006:0> scan 'tempTable'
ROW                      COLUMN+CELL
r1                        column=f1:c1, timestamp=1430036599391, value=hello, dlab
1 row(s) in 0.0160 seconds
```

如何查看数据--get

如何查看数据



get命令,通过表名、行、列、时间戳

```
hbase(main):012:0> get 'tempTable', 'r1', {COLUMN=>'f1:c1'}
COLUMN                      CELL
f1:c1                        timestamp=1430036599391, value=hello, dlab
1 row(s) in 0.0090 seconds

hbase(main):013:0> get 'tempTable', 'r1', {COLUMN=>'f1:c3'}
COLUMN                      CELL
0 row(s) in 0.0030 seconds
```

如何删除一个表--disable drop

如何删除一个表



首先必须要先让这个表失效

```
hbase(main):016:0> disable 'tempTable'  
0 row(s) in 1.3720 seconds  
  
hbase(main):017:0> drop 'tempTable'  
0 row(s) in 1.1350 seconds  
  
hbase(main):018:0> list  
TABLE  
testTable  
wordcount  
2 row(s) in 0.0370 seconds
```

8、HBase常用Java API及应用实例

HBase使用Java API准备过程

使用Java语言方法

- 首先在工程中导入jar包
- HBase的lib目录文件夹当中所有jar包都导到工程中不导入之前课程中介绍的Hadoop的jar包

实例

整体过程

```

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.hbase.*;
import org.apache.hadoop.hbase.client.*;
import org.apache.hadoop.hbase.util.Bytes;
import java.io.IOException;
public class Chapter4{
    public static Configuration configuration;
    public static Connection connection;
    public static Admin admin;

    public static void main(String[] args)throws IOException{
        createTable( "student",new String[]{ "score" });
        insertData( "student", "zhangsan", "score", "English", "69" );
        insertData( "student", "zhangsan", "score", "Math", "86" );
        insertData( "student", "zhangsan", "score", "Computer", "77" );
        getData( "student", "zhangsan", "score", "English" );
    }

    .....
    public static void init(){.....//建立连接
    public static void close(){.....//关闭连接
    public static void createTable(){.....//创建表
    public static void insertData() {.....//插入数据
    public static void getData{.....//浏览数据
    }
    }

```

Configuration

对配置信息管理的的一个类

Connection

对连接进行管理的的一个类

Admin

对数据库进行管理的的一个类
用于管理对表的创建删除等

建立连接、关闭连接、创建表

- 建立连接

```

//建立连接
public static void init(){
    configuration = HBaseConfiguration.create();
    configuration.set("hbase.rootdir","hdfs://localhost:9000/hbase");
    try{
        connection = ConnectionFactory.createConnection(configuration);
        admin = connection.getAdmin();
    }catch (IOException e){
        e.printStackTrace();
    }
}

```

备注： hbase-site.xml

```

<configuration>
<property>
<name>hbase.rootdir</name>
<value>hdfs://localhost:9000/hbase</value>
</property>
</configuration>

```

(单机版) file:///DIRECTORY/hbase

伪分布式HBase是用HDFS去存储数据

放置在分布式文件系统HDFS下面

- 关闭连接

```
//关闭连接
public static void close(){
    try{
        if(admin != null){
            admin.close();
        }
        if(null != connection){
            connection.close();
        }
    }catch (IOException e){
        e.printStackTrace();
    }
}
```

- 创建表

```
/*创建表*/
/**
 * @param tableName 表名
 * @param colFamily列族数组
 * @throws Exception
 */
public static void createTable(String myTableName,String[] colFamily) throws IOException {
    init();
    TableName tableName = TableName.valueOf(myTableName);
    if(admin.tableExists(tableName)){
        System.out.println("table exists!");
    }else {
        HTableDescriptor hTableDescriptor = new HTableDescriptor(tableName);
        for(String str: colFamily){
            HColumnDescriptor hColumnDescriptor = new HColumnDescriptor(str);
            hTableDescriptor.addFamily(hColumnDescriptor);
        }
        admin.createTable(hTableDescriptor);
    }
    close();
}
```

myTableName

表示要创建表的名称

colFamily

表示要创建的表包含了哪些列族

HTableDescriptor

专门用来对表进行信息理



在运行程序时，需要指定参数myTableName为“student”，colFamily为“{ “score” }”
程序的运行效果与如下HBase Shell命令等效：create ‘student’， ‘score’

- 添加数据

```

/*添加数据*/
/**
 * @param tableName 表名
 * @param rowkey 行键
 * @param colFamily 列族
 * @param col 列限定符
 * @param val 数据
 * @throws Exception
 */
public static void insertData(String tableName, String rowkey, String colFamily, String col, String val) throws IOException {
    init();
    Table table = connection.getTable(TableName.valueOf(tableName));
    Put put = new Put(Bytes.toBytes(rowkey));
    put.addColumn(Bytes.toBytes(colFamily), Bytes.toBytes(col), Bytes.toBytes(val));
    table.put(put);
    table.close();
    close();
}

```

表名 行键 列族 列名

Val : 表示那个单元格的数据

指定参数

添加数据

```

insertData("student","zhangsan","score","English","69");
insertData("student","zhangsan","score","Math","86");
insertData("student","zhangsan","score","Computer","77");

```

- 浏览数据

浏览数据

```
/*获取某单元格数据*/  
/**  
 * @param tableName 表名  
 * @param rowkey 行键  
 * @param colFamily 列族  
 * @param col 列限定符  
 * @throws IOException */  
public static void getData(String tableName,String rowkey,String colFamily,String col)throws IOException{  
    init();  
    Table table = connection.getTable(TableName.valueOf(tableName));  
    Get get = new Get(Bytes.toBytes(rowkey));  
    get.addColumn(Bytes.toBytes(colFamily),Bytes.toBytes(col));  
    Result result = table.get(get);  
    System.out.println(new String(result.getValue(colFamily.getBytes(),col==null?null:col.getBytes())));  
    table.close();  
    close();  
}
```

Table

用来获取相关表信息进行管理

get类

用来查看具体的相关单元格数据

Result类

管理输出的结果

指定参数

浏览数据



比如，想看一下张三的English成绩

```
getData("student", "zhangsan", "score", "English");
```



用Shell去读取相关数据

```
get 'student','zhangsan',{COLUMN=>'score:English'}"
```