

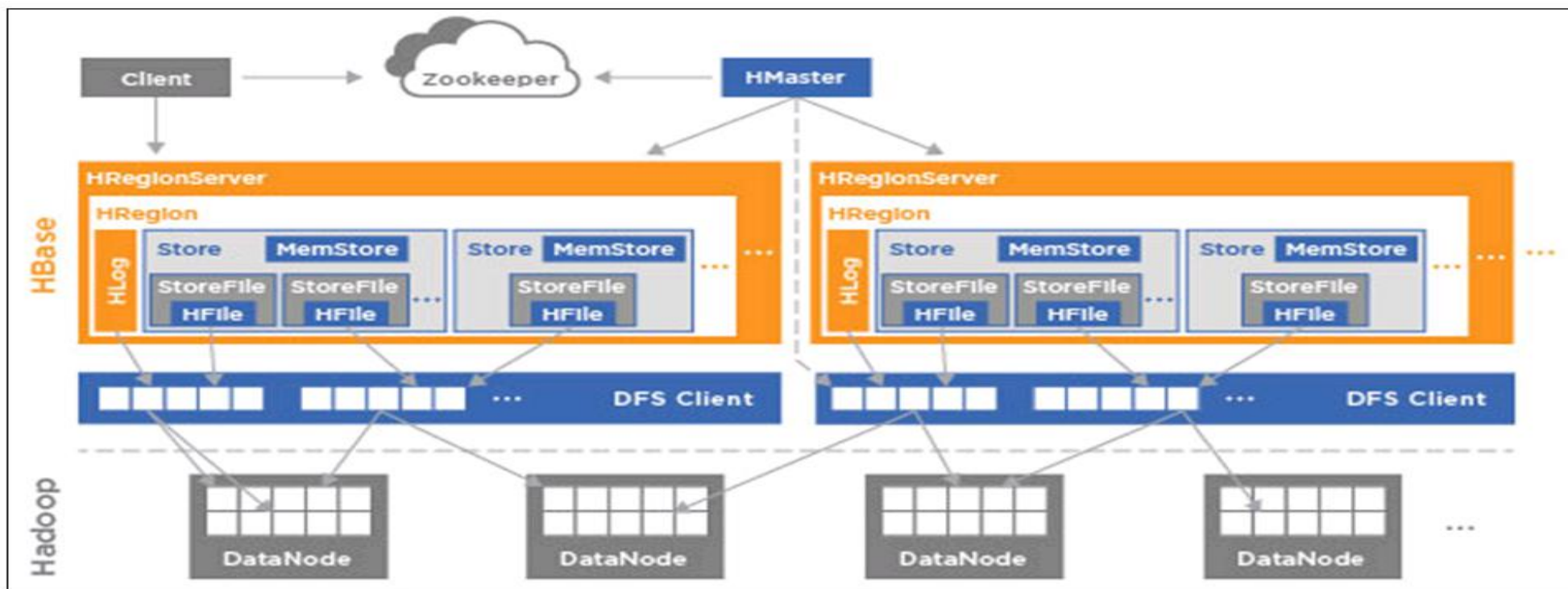
深入HBase

• 谭唐华

本课内容

- ◆ HBase数据检索流程讲解
- ◆ 深入HBase数据存储讲解.
- ◆ HBase Java API使用讲解
- ◆ HBase 架构深入剖析讲解
- ◆ HBase集成MapReduce
- ◆ HBase 的数据迁移及importTsv功能

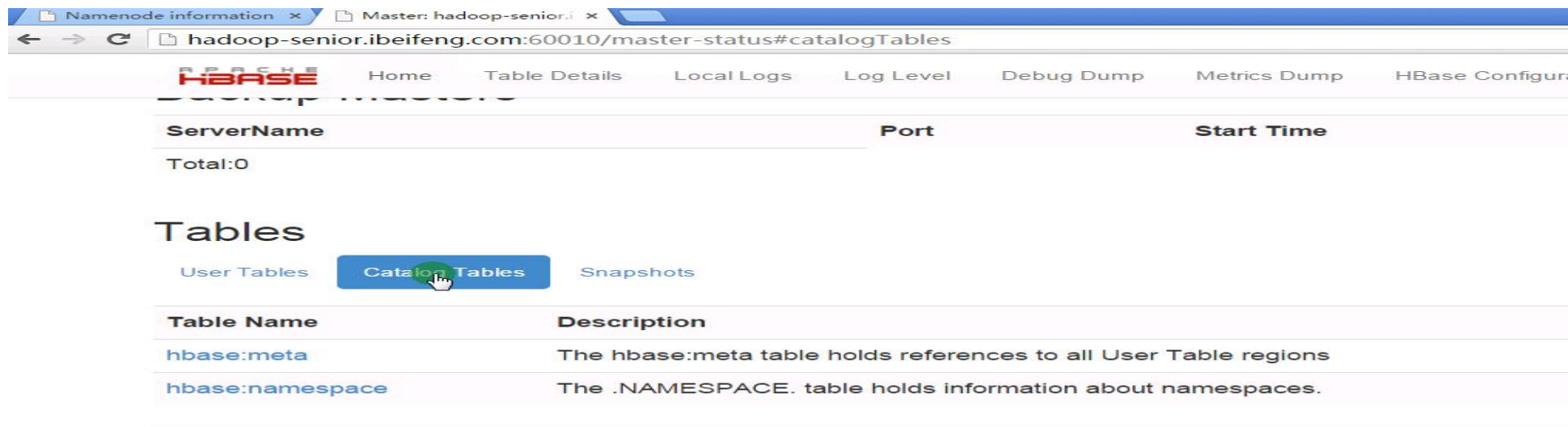
HBase数据检索流程讲解



- ◆ 上图中，我们可以看出不管客户端读或写一个表的数据，首先链接Zookeeper，因为需要到ZooKeeper中找读的数据，表是通过Region来管理，每个Region由RegionServer管理。每个region都有startkey及endkey

HBase数据检索流程讲解

HBase的表格分为User Tables（用户表）和Catalog Tables（系统自带表）。



The screenshot shows the HBase Master Status web interface. The browser address bar indicates the URL is `hadoop-senior.ibEIFeng.com:60010/master-status#catalogTables`. The page has a navigation bar with links: Home, Table Details, Local Logs, Log Level, Debug Dump, Metrics Dump, and HBase Configuration. Below the navigation bar, there is a table with columns: ServerName, Port, and Start Time. The table is currently empty, with 'Total:0' displayed. Below this table, there is a section titled 'Tables' with three tabs: 'User Tables', 'Catalog Tables' (which is selected and highlighted with a green circle), and 'Snapshots'. Under the 'Catalog Tables' tab, there is a table with two columns: 'Table Name' and 'Description'. The table lists two catalog tables: 'hbase:meta' and 'hbase:namespace'.

ServerName	Port	Start Time
Total:0		

Table Name	Description
hbase:meta	The hbase:meta table holds references to all User Table regions
hbase:namespace	The .NAMESPACE. table holds information about namespaces.

- User Tables（用户表）包含user信息、regions信息（startkey和endkey）。例：user表的region-01存在regionserver-03中。该信息是保存在meta-table中。
- 在HBase新版本中，有类似于RDBMS中DataBase的命名空间的概念。如上图。hbase的所有表都在data目录内，data下包含default目录和hbase目录，如下图。这里目录的概念就是命名空间的概念。

HBase数据检索流程讲解

- ◆ 用户自定义的表默认情况下命名空间为default，而系统自带的元数据表的命名空间为hbase。接下来介绍几个命令：

- 查看命名空间 list_namespace



Permission	Owner	Group	Size	Replication	Block Size	Name
drwxr-xr-x	beifeng	supergroup	0 B	0	0 B	default
drwxr-xr-x	beifeng	supergroup	0 B	0	0 B	hbase

```
hbase(main):007:0> list_namespace
NAMESPACE
default
hbase
2 row(s) in 0.0480 seconds
```

HBase数据检索流程讲解

- 查看某个命名空间下的表: list_namespace_tables '命名空间'

```
hbase(main):008:0> list_namespace_tables 'hbase'
TABLE
meta
namespace
2 row(s) in 0.0370 seconds
```

- 查询hbase命名空间中的meta表数据:scan '目录名:表名'

```
hbase(main):009:0> scan 'hbase:meta'
ROW COLUMN+CELL
hbase:namespace,,1443300 column=info:regioninfo, timestamp=1443300435856, value={ENCODED => 1fb
435609.1fb24f13b8ef4654a 24f13b8ef4654a15ff027d247f878, NAME => 'hbase:namespace,,1443300435609
15ff027d247f878.', STARTKEY => '', ENDKEY => ''}
hbase:namespace,,1443300 column=info:seqnumDuringOpen, timestamp=1444598610329, value=\x00\x00\
435609.1fb24f13b8ef4654a x00\x00\x00\x00\x00\x05
15ff027d247f878.
hbase:namespace,,1443300 column=info:server, timestamp=1444598610329, value=hadoop-senior.ibeif
435609.1fb24f13b8ef4654a eng.com:60020
15ff027d247f878.
hbase:namespace,,1443300 column=info:serverstartcode, timestamp=1444598610329, value=1444598602
435609.1fb24f13b8ef4654a 575
15ff027d247f878.
user,,1443301190074.cc61 column=info:regioninfo, timestamp=1443301190385, value={ENCODED => cc6
db08370714019a6014243651 1db08370714019a6014243651ecac, NAME => 'user,,1443301190074.cc61db0837
ecac. 0714019a6014243651ecac.', STARTKEY => '', ENDKEY => ''}
user,,1443301190074.cc61 column=info:seqnumDuringOpen, timestamp=1444598610328, value=\x00\x00\
db08370714019a6014243651 x00\x00\x00\x00\x00\x0E
ecac.
user,,1443301190074.cc61 column=info:server, timestamp=1444598610328, value=hadoop-senior.ibeif
db08370714019a6014243651 eng.com:60020
ecac.
user,,1443301190074.cc61 column=info:serverstartcode, timestamp=1444598610328, value=1444598602
db08370714019a6014243651 575
ecac.
```

HBase数据检索流程讲解

- ◆ meta表只有一个Region，它的Region也需要RegionServer管理，即为meta-region-server的功能。
- ◆ 用户首先找到meta-region-server，然后找到meta表，scan命令即可看到表格中column被什么server管理。

综上所述：

- 用户表由很多region组成，region信息存储在hbase：meta中。
- 用户表的每个region都有key。Client需要先读zookeeper，其实通过meta-region-server找到的是meta表的region，找到后扫描meta表的数据，然后再找到数据再操作。

本课内容

- ◆ HBase数据检索流程讲解
- ◆ 深入HBase数据存储讲解.
- ◆ HBase Java API使用讲解
- ◆ HBase 架构深入剖析讲解
- ◆ HBase集成MapReduce
- ◆ HBase 的数据迁移及importTsv功能

HBase结构图详解

■ HBase能高速实现数据存储和访问源于Hbase数据存储

1. 连接ZooKeeper，从ZooKeeper 中找要读的数据。我们需要知道表中的Rowkey在region的位置。
2. 客户端查找HRegionServer，HRegionServer 管理众多Region
3. HMaster也需要连接ZooKeeper，链接的作用是：HMaster需要知道哪些HRegionServer是活动的及HRegionServer所在位置，然后管理HRegionServer。
4. HBase内部是把数据写到HDFS上的，DFS有客户端
5. Region中包含HLOG、Store。若一张表有几个列族，就有几个Store。Store中有多个MemStore及StoreFile。StoreFile是对HFile的封装。StoreFile真正存储在HDFS上。
6. 所以写数据时先往HLog上写一份，再往MemStore上写一份。当MemStore达到一定大小则往StoreFile上写。若MemStore数据有丢失，则从HLog上恢复。
7. 而读数据时先到MemStore上读，再到StoreFile上读，之后合并。

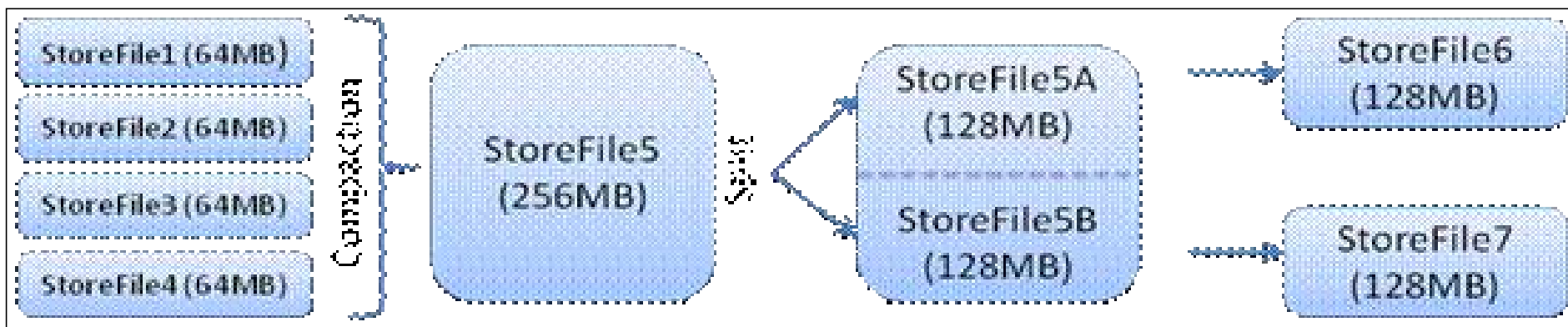
HBase数据存储详解

1. HBase中的所有数据文件都存储在Hadoop HDFS文件系统上，主要包括上述提出的两种文件类型：
 - 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，进行数据的存储。
 - HLog File，HBase中WAL（Write Ahead Log）的存储格式，物理上是Hadoop的Sequence File。
2. HRegionServer内部管理了一系列HRegion对象，每个HRegion对应了table中的一个region，HRegion中由多个HStore组成。每个HStore对应了Table中的一个column family的存储，可以看出每个columnfamily其实就是一个集中的存储单元，因此最好将具备共同IO特性的column放在一个column family中，这样最高效。

HBase数据存储详解

3. HStore存储是HBase存储的核心，由两部分组成，一部分是MemStore，一部分是StoreFile。
4. MemStore是 Sorted Memory Buffer，用户写入的数据首先会放入MemStore，当MemStore满了以后会Flush成一个StoreFile（底层实现是HFile）。
5. HLog 文件结构：WAL意为Write ahead log，类似Mysql中的binlog，用来做灾难恢复。Hlog记录数据的所有变更，一旦数据修改，就可以从log中进行恢复。
6. 每个HRegionServer维护一个HLog,而不是每个HRegion一个。这样不同region（来自不同table）的日志会混在一起，这样做的目的是不断追加单个文件相对于同时写多个文件而言，可以减少磁盘寻址次数，因此可以提高对table的写性能。带来的麻烦是，如果一台HRegionServer下线，为了恢复其上的region，需要将HRegionServer上的log进行拆分，然后分发到其它HRegionServer上进行恢复。

用户写入数据流程



- ◆Client客户端写入数据后 -> 数据存入MemStore，一直到MemStore满之后 Flush成一个StoreFile，直至增长到一定阈值 -> 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 -> 当StoreFiles Compact后，逐步形成越来越大的StoreFile -> 单个StoreFile大小超过一定阈值后，触发Split操作，把当前Region分成2个Region，Region会下线，新分出的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

本课内容

- ◆ HBase数据检索流程讲解
- ◆ 深入HBase数据存储讲解.
- ◆ **HBase Java API使用讲解**
- ◆ HBase 架构深入剖析讲解
- ◆ HBase集成MapReduce
- ◆ HBase 的数据迁移及importTsv功能

HBase Java API使用讲解

- 接下来我们讲解如何使用Java进行HBase编程。学习了这节后，就能用Java语言操作HBase的数据了。内容将包括：

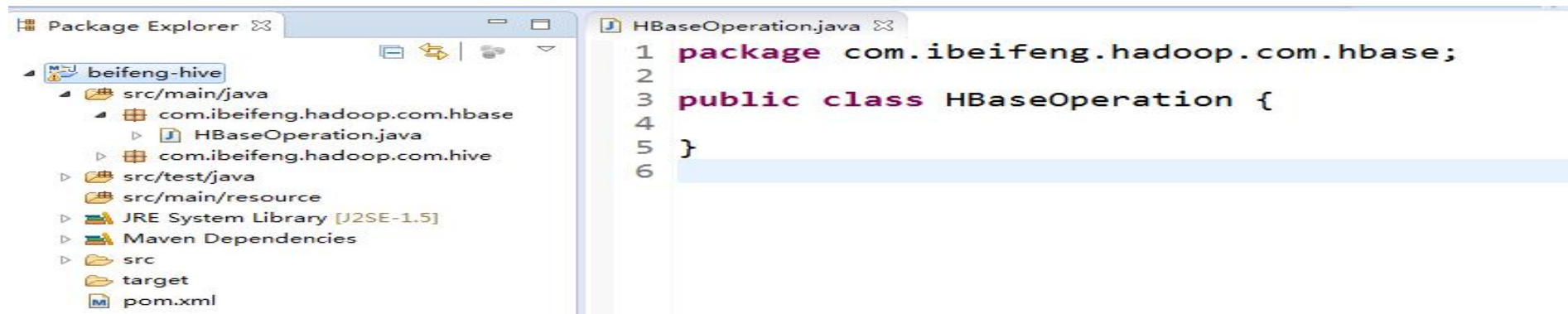
- 1.如何为HBase Java编程准备开发环境，接着介绍相关IDE及依赖的JAR包等。

- 2.介绍API，让我们熟练使用API在Eclipse环境中对HBase进行编码。

配置环境

■ 步骤如下：

- **第一步：**从网站下载eclipse，并下载jdk，安装好eclipse和jdk。
- **第二步：**打开项目beifeng-hive。单击右键创建源代码文件夹main，main中存放java文件夹和resource文件夹。在java文件夹内单击右键，继续创建文件包com.beifeng.senior.hadoop.hbase；在该文件夹内创建一个名为HBaseOperation的class。结构如图：



配置环境

■ 第三步：准备加载相关xml文件

① 在项目的pom.xml文件中添加以下配置信息

```
<dependency>
  <groupId>org.apache.hbase</groupId>
  <artifactId>hbase-server</artifactId>
  <version>${hbase.version}</version>
</dependency>

<dependency>
  <groupId>org.apache.hbase</groupId>
  <artifactId>hbase-client</artifactId>
  <version>${hbase.version}</version>
</dependency>
```

② 在项目的pom.xml文件中的properties中添加有关hbase的版本有关配置信息，加好后如下图所示

```
<properties>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEnc
  <hadoop.version>2.5.0</hadoop.version>
  <hive.version>0.13.1</hive.version>
  <hbase.version>0.98.6-hadoop2</hbase.version>
</properties>
```


java中操作HBase

- ◆ Configuration类
- ◆ HTable
- ◆ 使用Get获取数据
- ◆ 插入数据
- ◆ 删除数据
- ◆ 修改数据
- ◆ 利用scan查询数据

Configuration类

- ◆ 获取配置文件实例对象，使用Configuration类。Hadoop使用了一套独有的配置文件管理系统，并提供自己的API，即使用org.apache.hadoop.conf.Configuration处理配置信息。

- 在使用Java API时，Client端需要知道HBase的配置环境，如存储地址，ZooKeeper等信息。这些信息通过Configuration对象来封装，可通过如下代码构建该对象：

```
Configuration config=HBaseConfiguration.create();
```

- 在调用HBaseConfiguration.create()方法时，**HBase首先会在classpath下查找hbase-site.xml文件，将里面的信息解析出来封装到Configuration对象中，如果hbase-site.xml文件不存在，则使用默认的hbase-core.xml文件。**
- ◆ 除了将hbase-site.xml放到classpath下，开发人员还可通过config.set(name, value)方法来手工构建Configuration对象：

```
Configuration.set(String name, String value);
```

HTable

- ◆ 在HBase中，HTable封装表格对象。
- ◆ 使用前先导入import org.apache.hadoop.hbase.client.HTable;
- ◆ 接下来对表格的增删改查操作主要通过HTable来完成，构造方法如下：

```
HTable table=new HTable(config,tableName);
```

- ◆ 在构建多个HTable对象时，HBase推荐所有的HTable使用同一个Configuration。
- ◆ 这样，HTable之间便可共享HConnection对象、ZooKeeper信息以及Region地址的缓存信息。

使用Get获取数据

■ Get操作

1. 首先导入import org.apache.hadoop.hbase.client.Get;
2. 其次获取行键为某值的表格信息。语法格式如下：

```
Get get=new Get(rowkey的字节数组);
```

如图：

```
Get get=new Get(Bytes.toBytes("10002"));
```

使用Get获取数据

■ Result的使用

- ▶ 首先导入import org.apache.hadoop.hbase.client.Result;
- ▶ 使用HTable的get方法获取结果，语法格式如下：

```
Result res=HTable对象.get(Get对象);
```

如图：

```
Result res=table.get(get);
```

3. 遍历结果集。有很多方法，具体可以查看API，这里介绍使用rawCells（）方法，获取所有查询到的单元格信息。

- ① 首先导入import org.apache.hadoop.hbase.Cell;
- ② 其次利用Cell[] cells=res.rawCells();获取单元格信息。单元格信息包括行键rowkey、列族columnfamily、列名column及版本信息、值等

使用Get获取数据

4. CellUtil类：用于操作单元格的工具类。常用静态方法如下：

- ① CellUtil.cloneFamily(某列对象)：获取列族信息
- ② CellUtil.cloneQualifier(某列对象)：获取列信息
- ③ CellUtil.cloneValue(某列对象)：获取值

使用Get获取数据

5. 之后利用for循环遍历。代码片段如下图所示：

```
Configuration configuration = HBaseConfiguration.create();
try {
    HTable table=new HTable(configuration, "user");
    Get get=new Get(Bytes.toBytes("10002"));
    Result res=table.get(get);
    Cell[] cells=res.rawCells();
    for(Cell cell:cells)
    {
        System.out.print(Bytes.toString(CellUtil.cloneFamily(cell))+":");
        System.out.print(Bytes.toString(CellUtil.cloneQualifier(cell))+"->");
        System.out.println(Bytes.toString(CellUtil.cloneValue(cell)));
    }
    table.close();
} catch (IOException e) {
    //e.printStackTrace();
}
}
```

结果如图所示：

```
info:age->30
info:name->Wangwu
info:qq->231294737
```

6. 关闭表格

➤ 语法格式：table.close();

使用Get获取数据

■ 根据列族和列名查询信息

- 语法格式：`get.addColumn(“列族”，“列名”);`例题,查询info列族下的age信息，代码如图：

```
get.addColumn(Bytes.toBytes("info"), Bytes.toBytes("age"));
```

- 在获取result前加入上面这句话，执行程序后得到结果如下：

```
info:age->30
```


使用Get获取数据

■ 根据列族和列名查询信息

- 当需要查询多个列族或多个列名下的表格信息时，get.addColumn()方法可以出现多次。

```
Configuration configuration = HBaseConfiguration.create();
try {
    HTable table=new HTable(configuration, "user");
    Get get=new Get(Bytes.toBytes("10002"));
    get.addColumn(Bytes.toBytes("info"), Bytes.toBytes("age"));
    get.addColumn(Bytes.toBytes("info"), Bytes.toBytes("name"));
    Result res=table.get(get);
    Cell[] cells=res.rawCells();
    for(Cell cell:cells)
    {
        System.out.print(Bytes.toString(CellUtil.cloneFamily(cell))+":");
        System.out.print(Bytes.toString(CellUtil.cloneQualifier(cell))+ "->");
        System.out.println(Bytes.toString(CellUtil.cloneValue(cell)));
    }
    table.close();
} catch (IOException e) {
    //e.printStackTrace();
}
```

结果如图所示：

```
info:age->30
info:name->Wangwu
```

插入数据

◆ 在HBase中，实体的新增是通过Put操作来实现。操作步骤如下：

- ① 导入Put类import org.apache.hadoop.hbase.client.Put;
- ② 新建Put对象。语法格式：Put put=new Put(行键对应的字节数组)

```
Put put=new Put(Bytes.toBytes("10004"));
```

- ③ 添加列信息。语法格式：put.add(“列族”，“列名”，“列值”)

```
put.add(Bytes.toBytes("info"),  
        Bytes.toBytes("name"),  
        Bytes.toBytes("zhao"));
```

注意：一段代码中可以添加多个列的信息。

- ④ 将列信息添加到表格。语法格式：
 - a. 表格对象.put(put对象)：添加一个put对象入表格
 - b. 表格对象.put(含有多个put对象的list集合)：添加一个由多个put 对象组合成的list集合入表格

插入数据

⑤ 关闭表格。

⑥ 例题如下：

```
Configuration configuration = HBaseConfiguration.create();
try {
    HTable table=new HTable(configuration, "user");
    Put put=new Put(Bytes.toBytes("10005"));

    put.add(Bytes.toBytes("info"),
            Bytes.toBytes("name"),
            Bytes.toBytes("zhao"));
    put.add(Bytes.toBytes("info"),
            Bytes.toBytes("age"),
            Bytes.toBytes("25"));
    put.add(Bytes.toBytes("info"),
            Bytes.toBytes("address"),
            Bytes.toBytes("shanghai"));
    table.put(put);
    table.close();
} catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

删除数据

■ 以下删除操作是按rowkey行键来做删除，步骤如下：

① 导入import org.apache.hadoop.hbase.client.Delete;

② 新建Delete对象。语法格式：Delete delete=new Delete(行键对应的字节数组)

```
Delete delete=new Delete(Bytes.toBytes("10005"));
```

③ 列对象中指明要删除的列信息。常用方法如下：

➤ delete.deleteColumn(“列族”, “列名”)删除某一行

➤ delete.deleteFamily(“列族”)删除整个列族

删除数据

④ 从表格中删除信息。语法格式：table.delete(delete对象);

⑤ 关闭表格。语法格式：table.close();

⑥ 例题：

```
Configuration configuration = HBaseConfiguration.create();
try {
    HTable table=new HTable(configuration, "user");
    Delete delete=new Delete(Bytes.toBytes("10005"));
    delete.deleteColumn(Bytes.toBytes("info"),
        Bytes.toBytes("name"));
    //delete.deleteFamily(Bytes.toBytes("info"));
    table.delete(delete);
    table.close();
} catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

```
Configuration configuration = HBaseConfiguration.create();
try {
    HTable table=new HTable(configuration, "user");
    Delete delete=new Delete(Bytes.toBytes("10005"));
    //delete.deleteColumn(Bytes.toBytes("info"),
    //    Bytes.toBytes("name"));
    delete.deleteFamily(Bytes.toBytes("info"));
    table.delete(delete);
    table.close();
} catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```


修改数据

■ 在HBase中，实体的修改也是通过Put操作来实现。操作步骤如下：

① 导入Put类import org.apache.hadoop.hbase.client.Put;

② 新建Put对象。语法格式：Put put=new Put(行键对应的字节数组)，例：

```
Put put=new Put(Bytes.toBytes("10004"));
```

③ 将需要修改的列信息加入put对象 语法格式：put.add(“列族”，“列名”，“列值”)，例：

```
put.add(Bytes.toBytes("info"),  
        Bytes.toBytes("age"),  
        Bytes.toBytes("25"));  
put.add(Bytes.toBytes("info"),  
        Bytes.toBytes("name"),  
        Bytes.toBytes("zhao1"));
```

注意：一段代码中可以添加多个列的信息。

修改数据

④ 将列信息修改到表格,语法格式：

- 表格对象.put(put对象)：修改一个put对象入表格
- 表格对象.put(含有多个put对象的list集合)：修改一个由多个put 对象组合成的list集合入表格

④ 关闭表格。

⑤ 例题如图：

```
Configuration configuration = HBaseConfiguration.create();
try {
    HTable table=new HTable(configuration, "user");
    Put put=new Put(Bytes.toBytes("10004"));

    put.add(Bytes.toBytes("info"),
            Bytes.toBytes("age"),
            Bytes.toBytes("25"));
    put.add(Bytes.toBytes("info"),
            Bytes.toBytes("name"),
            Bytes.toBytes("zhao1"));
    table.put(put);
    table.close();
} catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

利用scan查询数据

- 在HBase中，利用scan来实现获取多条数据。

1. 实现全表扫描，步骤如下：

① 导入IOUtils类：`import org.apache.hadoop.io.IOUtils;`

导入Scan类：`import org.apache.hadoop.hbase.client.Scan;`

导入ResultScanner：

`import org.apache.hadoop.hbase.client.ResultScanner;`

② 新建Scan对象。语法格式：`Scan scan=new Scan();`

`ResultScanner result=table.getScanner(scan);`

③ 扫描表格得到结果集ResultScanner迭代器接口对象。语法格式：`ResultScanner result=表格对象.getScanner(Scan对象);`例题如下：

利用scan查询数据

- ④ 遍历结果，ResultScanner是由查询的结果Result对象组成。在之前的图17-11中我们已经演示了Result对象的访问方法rawCells()获取到多个单元格的方法。以下例题中：
 - 首先用getRow()获取行键。
 - 遍历Result对象中结果。
- ④ 关闭资源、表格。在之前的例题中，我们用table.close()方法关闭。本例题中我们系统提供的IOUtils类的closeStream()方法来关闭表格及其它资源。语法格式：IOUtils.closeStream(对象);

利用scan查询数据

⑥ 例题如下

```
Configuration configuration = HBaseConfiguration.create();
HTable table=null;
ResultScanner resultScanner=null;
try {
    table=new HTable(configuration, "user");
    Scan scan=new Scan();
    resultScanner=table.getScanner(scan);
    for(Result result : resultScanner)
    {
        System.out.println(Bytes.toString(result.getRow()));
        Cell[] cells=result.rawCells();
        for(Cell cell:cells)
        {
            System.out.print(Bytes.toString(CellUtil.cloneFamily(cell))+":");
            System.out.print(Bytes.toString(CellUtil.cloneQualifier(cell))+"->");
            System.out.println(Bytes.toString(CellUtil.cloneValue(cell)));
        }
        System.out.println("-----");
    }
} catch (Exception ex)
{
}
finally
{
    IOUtils.closeStream(resultScanner);
    IOUtils.closeStream(table);
}
```

结果如图：

```
10001
info:address->shanghai
info:age->25
info:name->ZHangsan
info:sex->male
-----
10002
info:age->30
info:name->Wangwu
info:qq->231294737
-----
10004
info:address->shanghai
info:age->25
info:name->zha02
-----
```

利用scan查询数据

2. 利用Scan有条件的查询

- ① 利用行键Rowkey设置开始值和结束值查询。使用到以下方法setStartRow()与set StopRow()方法。

- a. 格式：Scan对象.setStartRow(行键开始值);

Scan对象.setStopRow(行键结束值);

- b. 在以上例题的scan对象新建后加入设置开始值和结束值的方法，代码片段如下：

```
Scan scan=new Scan();  
scan.setStartRow(Bytes.toBytes("10002"));  
scan.setStopRow(Bytes.toBytes("10005"));  
resultScanner=table.getScanner(scan);
```

利用scan查询数据

c. 运行程序后，结果如下图所示：

```
2016-03-05 19:41:25,633 DEBUG
10002
2016-03-05 19:41:25,640 INFO
info:age->30
info:name->Wangwu
info:qq->231294737
-----
10004
info:address->shanghai
info:age->25
info:name->zha02
-----
```

【注】该方法查询到的最后结果所对应的Rowkey大于等于setStartRow()括号中的行键记录，小于setStopRow()括号中的行键记录

d. 代码简化

```
// Scan scan=new Scan();
// scan.setStartRow(Bytes.toBytes("10002"));
// scan.setStopRow(Bytes.toBytes("10003"));
Scan scan=new Scan(Bytes.toBytes("10002"),Bytes.toBytes("10005"));
```

利用scan查询数据

- ② 利用列族筛选。方法如下：利用Scan对象的addFamily（）方法通过列族筛选。例：

```
Scan scan=new Scan(Bytes.toBytes("10002"),Bytes.toBytes("10005"));
scan.addFamily(Bytes.toBytes("info"));
resultScanner=table.getScanner(scan);
```

- ③ 利用列筛选。方法如下：利用Scan对象的addFamily（"列族"，"列名"）方法通过列族筛选。例：

```
Scan scan=new Scan(Bytes.toBytes("10002"),Bytes.toBytes("10005"));
scan.addColumn(Bytes.toBytes("info"), Bytes.toBytes("name"));
resultScanner=table.getScanner(scan);
```

结果如图：

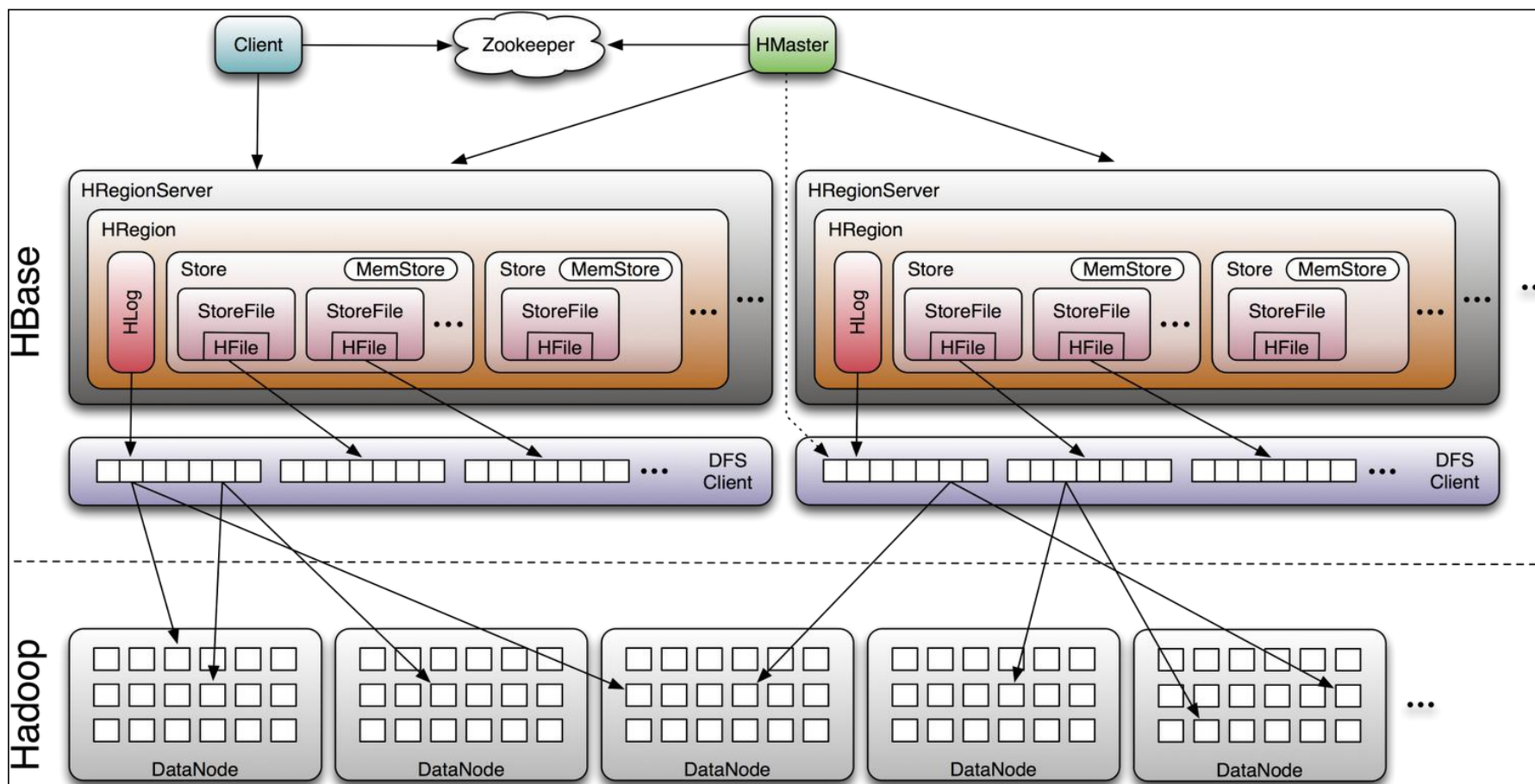
```
10002
info:name->Wangwu
-----
10004
info:name->zhaos2
-----
```

本课内容

- ◆ HBase数据检索流程讲解
- ◆ 深入HBase数据存储讲解.
- ◆ HBase Java API使用讲解
- ◆ HBase 架构深入剖析讲解
- ◆ HBase集成MapReduce
- ◆ HBase 的数据迁移及importTsv功能

HBase 架构深入剖析讲解

◆HBase Architecture



HBase架构中的客户端Client

■ 客户端有以下几点作用

1. 整个Hbase集群的访问入口
2. 使用HBase RPC机制与HMaster和HRegionServer进行通信
3. 使用HMaster进行通信进行管理类操作
4. 与HRegionServer进行数据读写类操作
5. 包含访问HBase的接口，并维护cache来加快对HBase的访问

协调服务组件ZooKeeper

■ ZooKeeper的作用如下：

1. 保证任何时候，集群中只有一个HMaster
2. 存贮所有HRegion的寻址入口
3. 实时监控HRegion Server的上线和下线信息，并实时通知给HMaster
4. 存储HBase的schema和table元数据
5. ZooKeeper Quorum存储-ROOT-表地址、HMaster地址

主节点HMaster

HMaster的主要功能如下：

1. HMaster没有单节点问题,HBase中可以启动多个HMaster，通过ZooKeeper的Master Election机制保证总有一个Master在运行，主要负责Table和Region的管理工作
 2. 管理用户对表的增删改查操作
 3. 管理HRegionServer的负载均衡，调整Region分布
 - 在命令行里面有个tools，tools这个分组命令其实全部都是Master做的事情。
1. Region Split后，负责新Region的分布;
 2. 在HRegionServer停机后，负责失效HResgionServer上Region迁移工作。

Region节点HRegionServer

■ HRegionServer的功能如下：

1. 维护HRegion,处理HRegion的IO请求，向HDFS文件系统中读写数据；
2. 负责切分运行过程钟变得过大的HRegion。
3. Clie访问HBase上数据的过程并不需要Master参与(寻址访问zookper和HRegion Server，数据读写访问HRegionServer),HMaster仅仅维护着table和Region的元数据信息，负载很低。

HBase与ZooKeeper关系

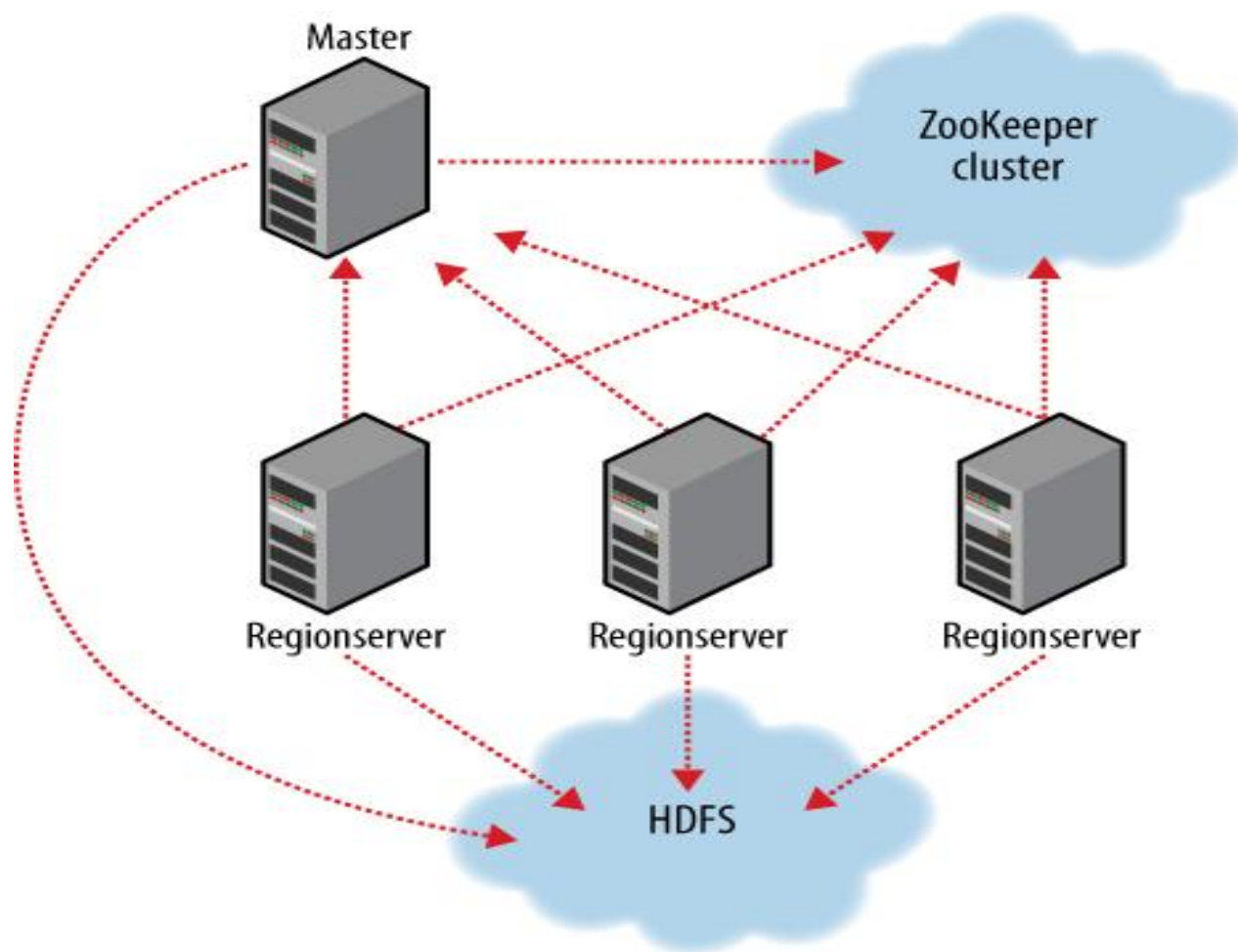


图17-46 ZooKeeper与HBase各组件关系

HBase与ZooKeeper关系

1. HBase 依赖ZooKeeper

- 首先HMaster和RegionServer都需要和ZooKeeper交互，因为RegionServer上线了还需要交互，之后ZooKeeper知道了告诉HMaster，而下线或断开了ZooKeeper知道了也告诉HMaster；同时HMaster还管理RegionServer，HMaster还会在HDFS上写Region数据。

1. 默认情况下，HBase 管理ZooKeeper 实例，比如， 启动或者停止ZooKeeper；
2. HMaster与HRegionServers 启动时会向ZooKeeper注册；
3. Zookeeper的引入使得HMaster不再是单点故障。

本课内容

- ◆ HBase数据检索流程讲解
- ◆ 深入HBase数据存储讲解.
- ◆ HBase Java API使用讲解
- ◆ HBase 架构深入剖析讲解
- ◆ HBase集成MapReduce
- ◆ HBase 的数据迁移及importTsv功能

HBase集成MapReduce配置

■ HBase与MapReduce集成时是需要jar包的，加载步骤如下：

1. 可以通过bin/hbase mapreduce命令查看。如图所示，为集成需要的jar包。

```
[beifeng@hadoop-senior hbase-0.98.6-hadoop2]$ bin/hbase mapredcp
2015-10-12 09:02:34,993 WARN [main] util.NativeCodeLoader: Unable to load native-hadoop library
for your platform... using builtin-java classes where applicable
/opt/modules/hbase-0.98.6-hadoop2/lib/hbase-common-0.98.6-hadoop2.jar:/opt/modules/hbase-0.98.6-
hadoop2/lib/protobuf-java-2.5.0.jar:/opt/modules/hbase-0.98.6-hadoop2/lib/hbase-client-0.98.6-ha
doo2.jar:/opt/modules/hbase-0.98.6-hadoop2/lib/hbase-hadoop-compat-0.98.6-hadoop2.jar:/opt/modu
les/hbase-0.98.6-hadoop2/lib/hbase-server-0.98.6-hadoop2.jar:/opt/modules/hbase-0.98.6-hadoop2/l
ib/hbase-protocol-0.98.6-hadoop2.jar:/opt/modules/hbase-0.98.6-hadoop2/lib/high-scale-lib-1.1.1.
jar:/opt/modules/hbase-0.98.6-hadoop2/lib/zookeeper-3.4.5.jar:/opt/modules/hbase-0.98.6-hadoop2/
lib/guava-12.0.1.jar:/opt/modules/hbase-0.98.6-hadoop2/lib/htrace-core-2.04.jar:/opt/modules/hba
se-0.98.6-hadoop2/lib/netty-3.6.6.Final.jar _
```

2. 设置HBase、Hadoop环境变量

```
export HBASE_HOME=/opt/modules/hbase-0.98.6-hadoop2
```

```
export HADOOP_HOME=/opt/modules/hadoop-2.5.0-cdh5.3.6
```

3. 设置Hadoop_classpath环境变量

```
HADOOP_CLASSPATH=`${HBASE_HOME}/bin/hbase mapredcp`
```

HBase集成MapReduce

- HBase默认集成的一些MapReduce程序，都在hbase-server-0.98.6-hadoop2.jar这个包里面。

1. rowcounter：统计hbase中有多少条数据

步骤如下：

- 启动resourcemanager
- 启动nodemanager
- 启动historyserver
- 运行rowcounter

```
HADOOP_CLASSPATH=`${HBASE_HOME}/bin/hbase mapredcp` $HADOOP_HOME/bin/yarn jar $HBASE_HOME/lib/hbase-server-0.98.6-hadoop2.jar  
rowcounter user
```


本课内容

- ◆ HBase数据检索流程讲解
- ◆ 深入HBase数据存储讲解.
- ◆ HBase Java API使用讲解
- ◆ HBase 架构深入剖析讲解
- ◆ HBase集成MapReduce
- ◆ HBase 的数据迁移及importTsv功能

HBase 的数据迁移及importTsv功能

◆ 数据迁移场景举例

1. Hbase集群装好了，若需要往表中装数据;
2. 数据的备份，如测试集群往生产集群上迁移。

◆ 几种HBase数据导入方式

- A. 使用Hbase Put API：例如MapReduce也是使用Put API
- B. 使用Hbase批量加载工具
- C. 自定义的MapReduce job

- 这三种方式都涉及到Put内置,大多数数据迁移场景都涉及到数据的导入(import)，从存在的RDBMS导入到Hbase中去，大多数简单直接的方法是直接获取数据,使用单线程，这种效果非常慢，其实可以写多线程完成。

importTsv功能

- ImportTsv是HBase官方提供的基于Mapreduce的批量数据导入工具。同时ImportTsv是Hbase提供的一个命令行工具，可以将存储在HDFS上的自定义分隔符（默认\t）的数据文件，通过一条命令方便的导入到HBase表中，对于大数据量导入非常实用
- 以下将介绍如何使用importTsv

■ **第一步：**建立student.tsv文件。如图：

```
[beifeng@hadoop-senior hadoop-2.5.0]$ cd /opt/datas/  
[beifeng@hadoop-senior datas]$ touch student.tsv  
[beifeng@hadoop-senior datas]$ vi student.tsv
```

■ **第二步：**编辑student.tsv文件，添加以下内容，并保存。如图：

10001	zhangsan	35	male	beijing	0109876543
10002	lisi	32	male	shanghai	0109876563
10003	zhaoliu	35	female	hangzhou	01098346543
10004	qianqi	35	male	shenzhen	01098732543

HBase 的数据迁移及importTsv功能

■ importTsv功能

■ 第三步：创建一个目录。如图：

```
[beifeng@hadoop-senior hadoop-2.5.0]$ bin/hdfs dfs -mkdir -p /user/beifeng/hbase/importtsv
```

■ 第四步：上传文件。如图：

```
[beifeng@hadoop-senior datas]$ /opt/modules/hadoop-2.5.0/bin/hdfs dfs -put student.tsv /user/beifeng/hbase/importtsv
```

■ 第五步：建表。在hbase中创建一个名为student的表，列族为info。如图：

```
hbase(main):040:0> create 'student', 'info'
```

■ 第六步：开始运行MapReduce。命令如下：

```
HADOOP_CLASSPATH=`${HBASE_HOME}/bin/hbase mapredcp`${HBASE_HOME}/conf bin/yarn jar  
/opt/modules/hbase-0.98.6-hadoop2/lib/hbase-server-0.98.6-hadoop2.jar importtsv -  
Dimporttsv.columns=HBASE_ROW_KEY,info:name,info:sex student  
hdfs://192.168.242.128:8020/user/beifeng/hbase/importtsv
```

HBase 的数据迁移及importTsv功能

■ importTsv功能

■ **第七步**：查看student表格结果。如图：

```
hbase(main):041:0> scan 'student'
ROW COLUMN+CELL
10001 column=info:address, timestamp=1444616091857, value=beijing
10001 column=info:age, timestamp=1444616091857, value=35
10001 column=info:name, timestamp=1444616091857, value=zhangsan
10001 column=info:phone, timestamp=1444616091857, value=0109876543
10001 column=info:sex, timestamp=1444616091857, value=male
10002 column=info:address, timestamp=1444616091857, value=shanghia
10002 column=info:age, timestamp=1444616091857, value=32
10002 column=info:name, timestamp=1444616091857, value=lisi
10002 column=info:phone, timestamp=1444616091857, value=0109876563
10002 column=info:sex, timestamp=1444616091857, value=male
10003 column=info:address, timestamp=1444616091857, value=hangzhou
10003 column=info:age, timestamp=1444616091857, value=35
10003 column=info:name, timestamp=1444616091857, value=zhao Liu
10003 column=info:phone, timestamp=1444616091857, value=01098346543
10003 column=info:sex, timestamp=1444616091857, value=female
10004 column=info:address, timestamp=1444616091857, value=shenzhen
10004 column=info:age, timestamp=1444616091857, value=35
10004 column=info:name, timestamp=1444616091857, value=qianqi
10004 column=info:phone, timestamp=1444616091857, value=01098732543
10004 column=info:sex, timestamp=1444616091857, value=male
4 row(s) in 0.0840 seconds
```

通过以上步骤可以将一个tsv文件的数据导入到hbase的表格中