大数据技术基础实验三 实验报告

毛子恒 2019211397 北京邮电大学 计算机学院

日期: 2022 年 4 月 1 日

1 概述

1.1 实验目的

掌握 HBase、ZooKeeper 的安装与使用,使用 MapReduce 批量将 HBase 表上的数据导入到 HDFS 中,学习本实验能快速掌握 HBase 数据库在分布式计算中的应用,理解 Java API 读取 HBase 数据等相关内容。

1.2 实验步骤

- 1. 下载安装并配置 Zookeeper;
- 2. 下载并安装 HBase;
- 3. HBase 实践。

2 实验结果及分析

HBase Shell 实践 进入 HBase Shell,输入命令,创建名为 2019211397-mzh 的表,向其中插入数据,行键分别为 2019211397-mzh-rk001~3,列族为 cf1,列限定符为 keyword,之后查看表的内容,结果如图 1。

程序编写 编写 Java 代码, 其中 MemberMapper 类的代码如图 2所示。

该类中的 map 方法遍历表的每一行,再遍历该行的每一个单元格,获取每个单元格的值、 列族、列限定符、时间戳,将这四者连接起来成为值,以行键作为键,将这样的键值对输出到上 下文中。

程序打包和运行 程序打包后复制到主机上,运行结果如图 3。

可见输出了键值对,其中键为行键,值为列族、列限定符、单元格、时间戳的值连接起来。

```
[root@mzh-2019211397-0001 -]# hbase shell
SLF41: Class path contains multiple SLF41 bindings.
SLF41: Class path contains multiple SLF41 bindings.
SLF41: Found binding in [jar:file:/usr/local/hbase-2.0.2/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF41: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF41: Actual binding is of type [org.slf4].impl.Log4jloggerFactory]
2022-03-31 19:33:01,037 WARN [main] util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applica ble
Blase Shell
Use "help" to get list of supported commands.
Use "exit" to quit this interactive shell.
Version 2.0.2, "Icfab033e779df840d5612a85277f42a6a4e8172, Tue Aug 28 20:50:40 PDT 2018
Took 0.0083 seconds
hbase(main):001:0> create '2019211397-mzh', 'cf1'
Created table 2019211397-mzh
Took 3.1564 seconds
hbase(main):002:0> put '2019211397-mzh', '2019211397-mzh-rk001', 'cf1:keyword', 'applicate'
Took 0.0064 seconds
hbase(main):003:0> put '2019211397-mzh', '2019211397-mzh-rk002', 'cf1:keyword', 'iHOAWEI P50 Pro'
Took 0.0064 seconds
hbase(main):003:0> put '201921397-mzh', '2019211397-mzh-rk003', 'cf1:keyword', 'HUAWEI P50 Pro'
Took 0.0067 seconds
hbase(main):005:0> scan '201921397-mzh'
COLUMN+CELL
2019211397-mzh-rk001
2019211397-mzh-rk003
```

图 1: HBase Shell 操作

3 实验总结

本次实验中我使用 HBase Shell 和 Java API 对 HBase 进行了简单的操作,使我对 HBase 和 Zookeeper 的原理理解更加深刻。

```
package org.mzh2019211397.hbase.inputSource;
3
         import org.apache.hadoop.hbase.Cell;
         import org.apache.hadoop.hbase.client.Result;
         import org.apache.hadoop.hbase.io.ImmutableBytesWritable;
         import org.apache.hadoop.hbase.mapreduce.TableMapper;
7
         import org.apache.hadoop.hbase.util.Bytes;
 8
         import org.apache.hadoop.io.Writable;
         import org.apache.hadoop.io.Text;
10
         import java.io.IOException;
         public class MemberMapper extends TableMapper<Writable, Writable> {
             private Text k = new Text();
             private Text v = new Text();
             public static final String FIELD_COMMON_separator = "\u0001";
16
18
             @Override
19 이
             {\tt protected\ void\ setup(Context\ context)\ throws\ {\tt I0Exception,\ InterruptedException\ \{}}
20
21
22
             @Override
23 이 @
             protected void map(ImmutableBytesWritable row, Result columns, Context context)
24
                     throws IOException, InterruptedException {
25
                  String <u>value</u> = null;
                 String rowkey = new String(row.get());
27
28
                 byte[] columnFamily = null;
                  byte[] columnQualifier = null;
                 long ts = 0L;
30
31
                  try {
                     for (Cell cell : columns.listCells()) {
                         value = Bytes.toStringBinary(cell.getValueArray());
                         columnFamily = cell.getFamilyArray();
35
36
                          columnQualifier = cell.getQualifierArray();
37
                         ts = cell.getTimestamp();
                          k.set(rowkey);
                          v.set(Bytes.toString(columnFamily) + FIELD_COMMON_separator + Bytes.toString(columnQualifier)
39
40
                                  + FIELD_COMMON_separator + value + FIELD_COMMON_separator + ts);
41
                          context.write(k, v);
                     }
43
                 } catch (Exception e) {
44
                      e.printStackTrace();
45
                     System.err.println("Error:" + e.getMessage() + ",Row:" + Bytes.toString(row.get()) + ",Value" + value);
46
47
48
         }
```

图 2: MemberMapper 类

```
[root@ab-2019211397-0001 -]# hadoop fs -cat /tmp/2019211397-mzh/npart-m-00000
2019211397-mzh-16001 * 2019211397-mzh-rk0010rflkeymord open pilorape | plantorm... using builtin-java classes where applicable
2019211397-mzh-rk001 * 2019211397-mzh-rk0010rflkeymord open pilorape | 2019211397-mzh-rk0010rflkeymord open pilorape | plantorm... using builtin-java classes where applicable
2019211397-mzh-rk001 * 2019211397-mzh-rk0010rflkeymord open pilorape | plantorm... using builtin-java classes where applicable
2019211397-mzh-rk001 * 2019211397-mzh-rk0010rflkeymord open pilorape | plantorm... using builtin-java classes where applicable
2019211397-mzh-rk001 * 2019211397-mzh-rk0010rflkeymord open pilorape | plantorm... using builtin-java classes where applicable
2019211397-mzh-rk001 * 2019211397-mzh-rk001142019211397-mzh-rk001\2019211397-mzh-rk001\2019211397-mzh-rk001\2019211397-mzh-rk001\2019211397-mzh-rk001\2019211397-mzh-rk001\2019211397-mzh-rk001\2019211397-mzh-rk001\2019211397-mzh-rk001\2019211397-mzh-rk001\2019211397-mzh-rk001\2019211397-mzh-rk001\2019211397-mzh-rk001\2019211397-mzh-rk001\2019211397-mzh-rk001\2019211397-mzh-rk001\2019211397-mzh-rk001\2019211397-mzh-rk001\2019211397-mzh-rk001\2019211397-mzh-rk001\2019211397-mzh-rk001\2019211397-mzh-rk001\2019211397-mzh-rk001\2019211397-mzh-rk001\2019211397-mzh-rk001\2019211397-mzh-rk001\2019211397-mzh-rk001\2019211397-mzh-rk001\2019211397-mzh-rk001\2019211397-mzh-rk001\2019211397-mzh-rk001\2019211397-mzh-rk001\2019211397-mzh-rk001\2019211397-mzh-rk001\2019211397-mzh-rk001\2019211397-mzh-rk001\2019211397-mzh-rk001\2019211397-mzh-rk001\2019211397-mzh-rk001\2019211397-mzh-rk001\2019211397-mzh-rk001\2019211397-mzh-rk001\2019211397-mzh-rk001\2019211397-mzh-rk001\2019211397-mzh-rk001\2019211397-mzh-rk001\2019211397-mzh-rk001\2019211397-mzh-rk001\2019211397-mzh-rk001\2019211397-mzh-rk001\2019211397-mzh-rk001\2019211397-mzh-rk001\2019211397-mzh-rk001\2019211397-mzh-rk001\2019211397-mzh-rk001\2019211397-mzh-rk001\2019211397-mzh-rk001\2019211397-mzh-rk001\2019211397-mzh-rk001\2
```

图 3: 运行结果