计算机科学与技术学院 大数据管理与分析 课程实验报告

实验题目：大数据系统基本实验		**学号：**201605130116
日期：2019.3.10	**班级：**2016 级泰山学堂	**姓名：**杜洪超
Email：1503345074@qq.com		

实验目的：

1.熟悉常用的 Linux 操作和 Hadoop 操作；

2.熟悉常用的 HDFS 操作；理解 HDFS 在 Hadoop 体系结构中的角色，熟练使用 HDFS 操作常用的 shell 命令，熟悉 HDFS 操作常用的 Java API；

3.熟悉常用的 HBase 操作；理解 HBase 在 Hadoop 体系结构中的角色，熟练使用 HBase 操作常用的 shell 命令，熟悉 HBase 操作常用的 Java API；

4.通过 NoSQL 数据库和常用数据库的比较，理解 4 种数据库(MySQL、HBase、Redis 和 MongDB)的概念及不同点，熟练使用 4 种数据库操作常用的 shell 命令，熟悉 4 种数据库操作常用的 Java API；

5. 通过实验掌握基本的 MapReduce 编程方法；掌握用 MapReduce 解决一些常见的数据处理问题,包括数据去重、数据排序和数据挖掘等。

实验软件和硬件环境：

软件环境：

系统：Ubuntu16.04 LTS 64 位

软件：openjdk-8-jre,openjdk-8-jdk,java1.8.0_191

Hadoop 2.9.2，HBase 1.2.11

MySQL 5.7.25，Redis 3.0.6，MongoDB 2.6.10

Eclipse,ssh

硬件环境：

CPU: Intel® Core™ i5-6260U CPU @ 1.80GHz × 4

磁盘：121.8 GB

内存：7.7 GiB

实验原理和方法：

1. 熟悉常用的 Linux 操作和 hadoop 操作；通过命令行实践 Linux 命令和 hadoop HDFS 操作
2. 熟悉常用的 HDFS 操作；练习 HDFS 命令行操作和并用 JAVA API 实现相同功能；
3. 熟悉常用的 HBase 操作；练习 HBase 命令行操作和并用 JAVA API 实现相同功能；
4. 比较 NoSQL 数据库和常用数据库；练习使用四种数据库的命令行操作并用相应的 JAVA API 实现相同功能；

5. 通过实验掌握基本的 MapReduce 编程方法；

实验步骤：（不要求罗列完整源代码）

1. 熟悉常用的 Linux 操作和 hadoop 操作；

   常用的 Linux 命令举例如下：

   cp -r src dst 递归复制文件夹

   head/tail -n -count 不显示最后 count 行和只显示最后 count 行

   touch -t time file 修改文件时间

   find path -name file_name 查找文件

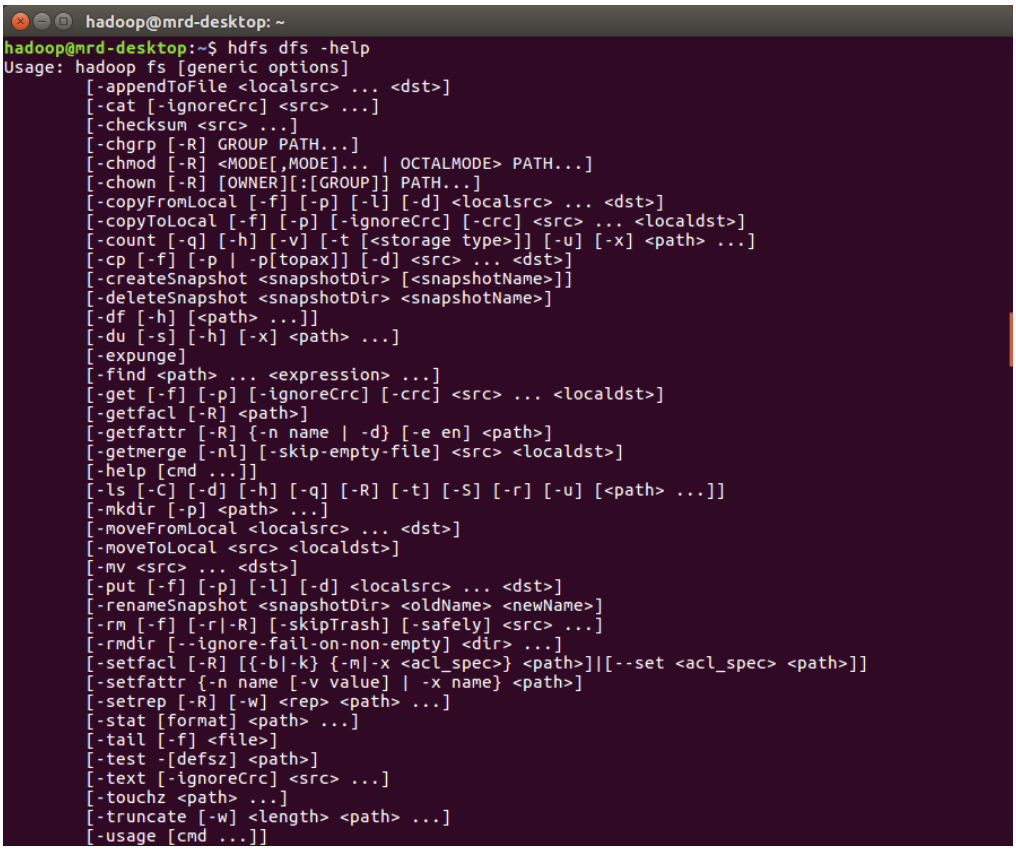
   tar -czf *.tar.gz file_list 压缩文件

   tar -xzf *.tar.gz -C path 解压缩文件到目录

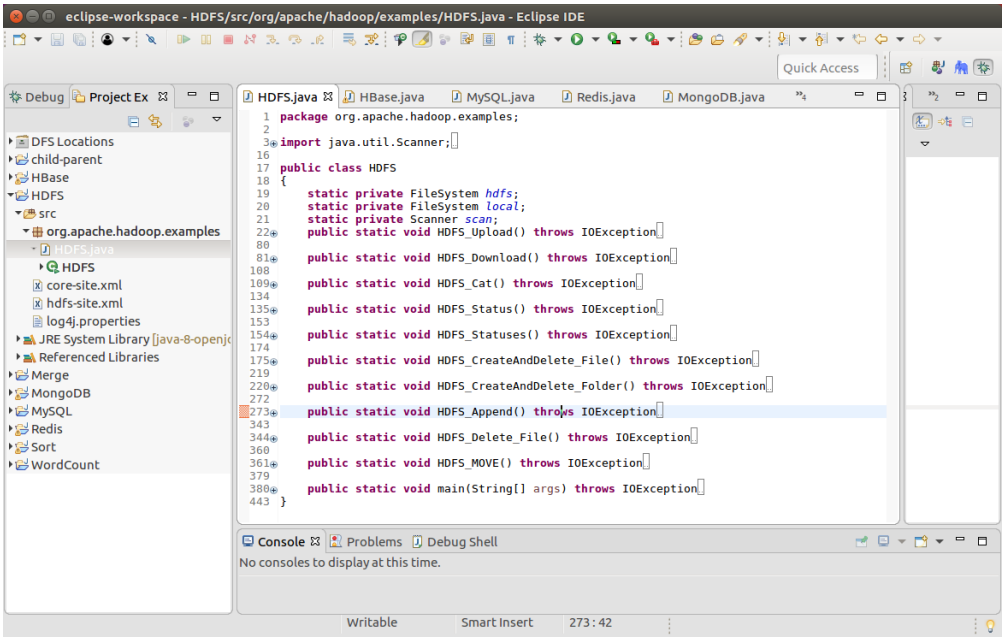
   grep string file 查找指定字符串

2. 熟悉常用的 HDFS 操作；

   常用的 HDFS 命令行如下：



   使用 HDFA java API 编写的函数如下，实现了相同的用 shell 命令完成的十项任务：

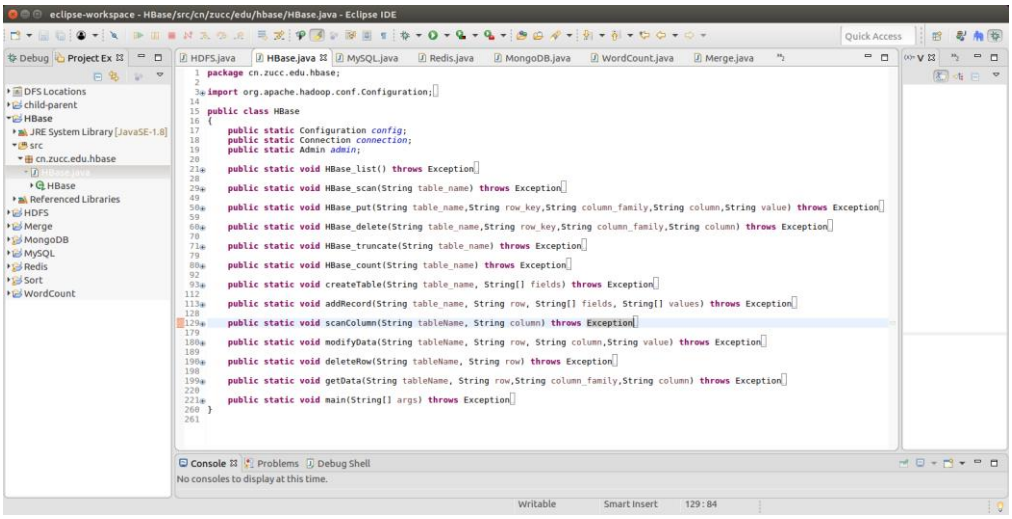


3. 熟悉常用的 HBase 操作；

HBase shell 常用命令如下：

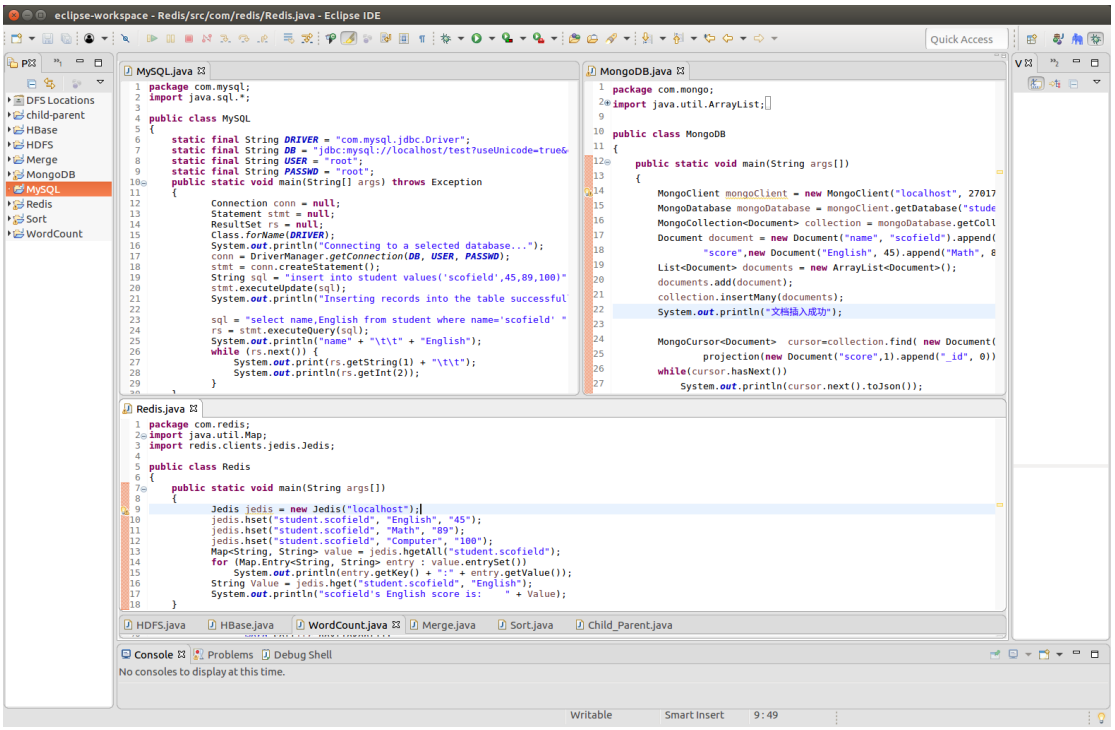
```
hadoop@mrd-desktop: ~
hbase(main):001:0> help
HBase Shell, version 1.2.11, rca53d58f5b7abde0c189c9f78baf4246bddffac3, Fri Feb 15 18:12:16 CST 2019
Type 'help "COMMAND"', (e.g. 'help "get"' -- the quotes are necessary) for help on a specific command.
Commands are grouped. Type 'help "COMMAND_GROUP"', (e.g. 'help "general"') for help on a command group.

COMMAND GROUPS:
  Group name: general
  Commands: status, table_help, version, whoami

  Group name: ddl
  Commands: alter, alter_async, alter_status, create, describe, disable, disable_all, drop, drop_all, en
able, enable_all, exists, get_table, is_disabled, is_enabled, list, locate_region, show_filters

  Group name: namespace
  Commands: alter_namespace, create_namespace, describe_namespace, drop_namespace, list_namespace, list_
namespace_tables

  Group name: dml
  Commands: append, count, delete, deleteall, get, get_counter, get_splits, incr, put, scan, truncate, t
runcate_preserve

  Group name: tools
  Commands: assign, balance_switch, balancer, balancer_enabled, catalogjanitor_enabled, catalogjanitor_r
un, catalogjanitor_switch, close_region, compact, compact_rs, flush, major_compact, merge_region, move,
normalize, normalizer_enabled, normalizer_switch, split, trace, unassign, wal_roll, zk_dump

  Group name: replication
  Commands: add_peer, append_peer_tableCFs, disable_peer, disable_table_replication, enable_peer, enable
_table_replication, list_peers, list_replicated_tables, remove_peer, remove_peer_tableCFs, set_peer_tabl
eCFs, show_peer_tableCFs

  Group name: snapshots
  Commands: clone_snapshot, delete_all_snapshot, delete_snapshot, list_snapshots, restore_snapshot, snap
shot

  Group name: configuration
  Commands: update_all_config, update_config

  Group name: quotas
  Commands: list_quotas, set_quota

  Group name: security
  Commands: grant, list_security_capabilities, revoke, user_permission

  Group name: procedures
  Commands: abort_procedure, list_procedures

  Group name: visibility labels
  Commands: add_labels, clear_auths, get_auths, list_labels, set_auths, set_visibility

SHELL USAGE:
Quote all names in HBase Shell such as table and column names.  Commas delimit
command parameters.  Type <RETURN> after entering a command to run it.
```

使用 HBase java API 编写的函数如下，实现了相同的用 shell 命令完成的几项任务：



4. 比较 NoSQL 数据库和常用数据库
   1. MySQL 常用命令：
      mysql -u root -p 使用密码登陆 MySQL root 用户
      create database d_name; 创建数据
      use d_name; 使用某个数据库
      create table test(field type); 创建表格
      insert into t_name values(field,value); 插入数据
      select * from t_name 查看数据
      update t_name set field=value 更新数据
   2. HBase 常用命令
      见实验第三部分
   3. Redis 常用命令：
      redis_cli 启动命令行客户端
      hset table.row field value 插入 key-value 数据
      hget table.row filed 获取指定数据
      hgetall table.row 获取某一范围所有 key-value 数据
   4. MongoDB 常用命令：
      mongo 启动命令行客户端
      use t_name 创建并使用某个数据库
      var stus=[{key:value}] 创建文档
      db.t_name.insert(stus) 插入数据
      db.t_name.find().pretty() 输出表格信息
      db.t_name.find({field},{field:0}) 只输出指定数据
      db.t_name.update({field},{$set:{}}) 更新数据

   除 HBase 外其它三种数据库使用 Java API 编写的类如下：



5. 通过实验掌握基本的 MapReduce 编程方法

1. 文件的合并和去重

对多个文件的行进行合并去重操作；

可以在 Map 函数中设置 key 为行，value 为空，Reduce 函数中对每个 key-values 只输出 key，函数如下：

```
public static class Map extends Mapper<Object, Text, Text, Text>
{
    public void map(Object key, Text value, Context context) throws IOExc
    {
        context.write(value, new Text(""));
    }
}

public static class Reduce extends Reducer<Text, Text, Text, Text>
{
    public void reduce(Text key, Iterable<Text> values, Context context)
    {
        context.write(key, new Text(""));
    }
}
```

2. 数据排序

对不同文件的数据进行排序，输出名次和数据；

利用 reduce 前的 sort 过程自动排序，Map 中把数据作为 key,value 可以设为任意值，就能在 Reduce 中获得排好序的数据，只需再处理名次的输出即可，函数如下：

```
public static class Map extends Mapper<Object, Text, IntWritable, IntWritable>
{
    private IntWritable data = new IntWritable();
    public void map(Object key, Text value, Context context) throws IOException, InterruptedExc
    {
        data.set(Integer.parseInt(value.toString()));
        context.write(data, new IntWritable(1));
    }
}

public static class Reduce extends Reducer<IntWritable, IntWritable, IntWritable, IntWritable>
{
    private IntWritable data = new IntWritable(1);
    public void reduce(IntWritable key, Iterable<IntWritable> values, Context context) throws I
    {
        for (IntWritable value:values)
        {
            context.write(data, key);
            data = new IntWritable(data.get()+1);
        }
    }
}
```

3. 亲属关系挖掘

通过原始数据中的父母和子女关系，挖掘爷爷奶奶和孙子孙女关系；

对给定的一个人，他的父母和和他的子女肯定满足爷孙类关系，同样的某个爷孙类关系肯定存在与某一个人的父母和子女关系中；因此对原始数据的每一行数据，Map 函数生成两组 key-value，对孩子：key 为孩子的姓名，value 为 parent+父母姓名；对父母，key 为父母姓名，value 为 value+孩子姓名；在 Reduce 函数中，对 key 为某个人的 values，包括父母关系和子女关系两部分，这两部分两两配对均为爷孙类关系；函数如下：

```
public static class Map extends Mapper<Object, Text, Text, Text>
{
    public void map(Object key, Text value, Context context) throws IOExcept
    {
        String[] data=value.toString().split(" ");
        if (!data[0].equals(new String("child")))
        {
            context.write(new Text(data[0]),new Text("parent "+data[1]));
            context.write(new Text(data[1]),new Text("child "+data[0]));
        }
    }
}

public static class Reduce extends Reducer<Text, Text, Text, Text>
{
    private boolean flag=true;
    public void reduce(Text key, Iterable<Text> values, Context context) thr
    {
        if (flag)
        {
            context.write(new Text("grandchild"), new Text("grandparent"));
            flag=false;
        }
        List<Text> child = new ArrayList<>();
        List<Text> parent = new ArrayList<>();
        for (Text value:values)
        {
            String[] relation=value.toString().split(" ");
            if (relation[0].equals("child"))
                child.add(new Text(relation[1]));
            else
                parent.add(new Text(relation[1]));
        }
        for (Text grandchild:child)
            for (Text grandparent:parent)
                context.write(grandchild,grandparent);
    }
}
```

结论分析与体会：

通过本次实验，熟悉了常用的 Linux 操作和 Hadoop 操作；了解了常用的数据库，如 MySQL,Redis，MongoDB，HBase 的基本操作和 Java API;并通过几个 Map_Reduce 程序的编写和实现，掌握了 MapReduce 编程的基本方法，加深了对大数据的理解

就实验过程中遇到和出现的问题，你是如何解决和处理的，自拟 1－3 道问答题：

1. HDFS 编程中实现在文件首部添加内容；

   为了实现添加多行，在本地新建了临时文件用于存储用户输入，如果在 HDFS 文件系统中新建临时文件会出现用户不匹配问题，因此把 HDFS 中的目标文件移动到本地文件系统中，在根据是在首部添加还是尾部添加，把两个文件的内容按顺序写到第三个文件中，最后把第三个文件移回 HDFS，删除两个临时文件；

2. HBase 无法正常启动；

   HBase 版本要和 HDFS 统一，官网中有版本对应关系，更换版本后解决