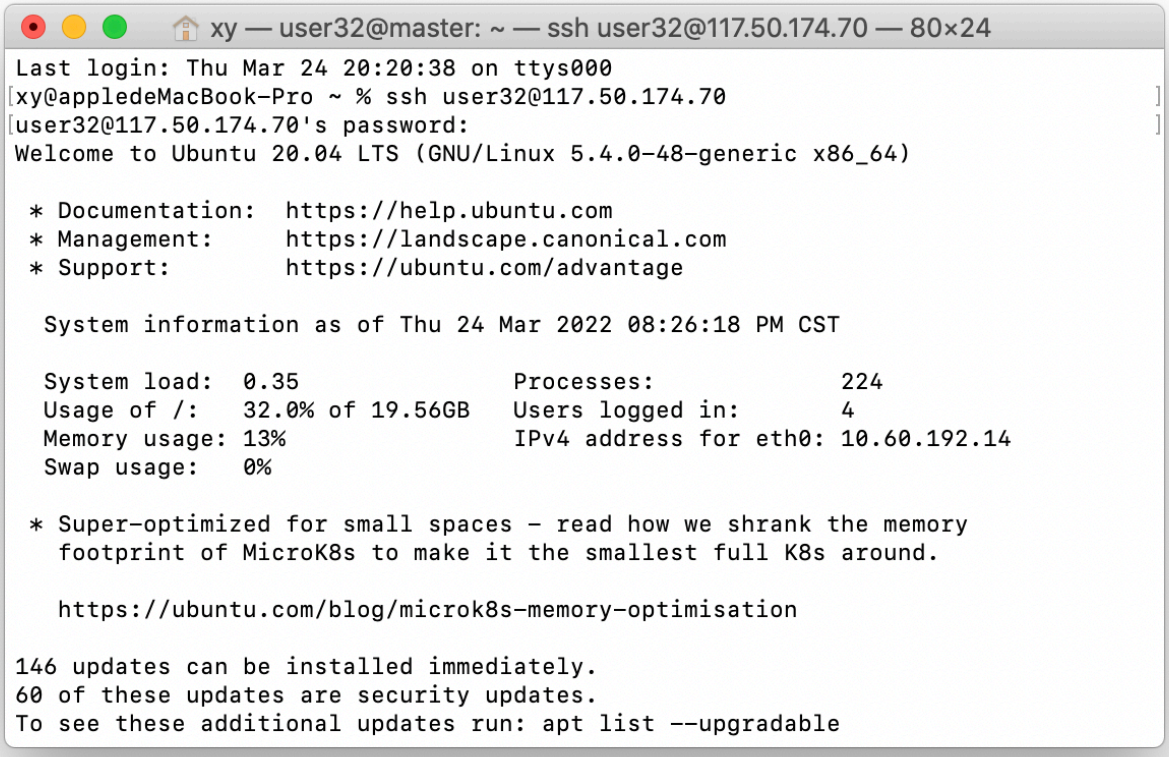
云计算第二次作业

2051495 徐奕 大数据

2022.3.26

零、登陆集群



修改密码

> passwd

一、熟悉常用的HDFS操作

1.向HDFS中上传任意文本文件，如果指定的文件在HDFS中已经存在，由用户指定是追加到原有文件末尾还是覆盖原有的文件

```
$ cd hw2
$ hdfs dfs -test -e text.txt
$ echo $?
```

```
user32@master:~/hw2$ hdfs dfs -test -e text.txt

user32@master:~/hw2$ echo $?
0

user32@master:~/hw2$ hdfs dfs -appendToFile /home/user32/hw2/text.txt /user/user
32/text.txt
```

验证:

```
user32@master:~$ hadoop jar HDFSApi.jar
读取文件: /user/user32/text.txt

love
and
peace
~
读取完成
```

```java
import java.io.FileInputStream;
import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FSDataOutputStream;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
public class CopyFromLocalFile {
    /**
     * 判断路径是否存在
     */
    public static boolean test(Configuration conf, String path) {
        try (FileSystem fs = FileSystem.get(conf)) {
            return fs.exists(new Path(path));
        } catch (IOException e) {
            e.printStackTrace();
            return false;
        }
    }
    /**
     * 复制文件到指定路径 若路径已存在，则进行覆盖
     */

    public static void copyFromLocalFile(Configuration conf, String localFilePath,
String remoteFilePath) {
        Path localPath = new Path(localFilePath);
        Path remotePath = new Path(remoteFilePath);
        try (FileSystem fs = FileSystem.get(conf)) {
            /* fs.copyFromLocalFile 第一个参数表示是否删除源文件，第二个参数表示是否覆盖 */
            fs.copyFromLocalFile(false, true, localPath, remotePath);
        }
        catch (IOException e) {
            e.printStackTrace();
        }
    }
    /**
```

```java
     * 追加文件内容
     */
    public static void appendToFile(Configuration conf, String localFilePath, String
remoteFilePath) {
        Path remotePath = new Path(remoteFilePath);
        try (FileSystem fs = FileSystem.get(conf);
             FileInputStream in = new FileInputStream(localFilePath);) {
            FSDataOutputStream out = fs.append(remotePath);
            byte[] data = new byte[1024];
            int read = -1;
            while ((read = in.read(data)) > 0) {
                out.write(data, 0, read);
            }
            out.close();
        }
        catch (IOException e) {
            e.printStackTrace();
        }
    }
    /**
     * 主函数
     */
    public static void main(String[] args) {
        Configuration conf = new Configuration();
        //conf.set("fs.defaultFS", "hdfs://localhost:9000");
        String localFilePath = "/home/user32/hw2/text.txt"; // 本地路径
        String remoteFilePath = "/user/user32/text.txt"; // HDFS路径
        String choice = "append"; // 若文件存在则追加到文件末尾
        //String choice = "overwrite"; // 若文件存在则覆盖
        try {
            /* 判断文件是否存在 */
            boolean fileExists = false;
            if (CopyFromLocalFile.test(conf, remoteFilePath)) {
                fileExists = true;
                System.out.println(remoteFilePath + " 已存在.");
            }
            else {
                System.out.println(remoteFilePath + " 不存在.");
            }
            /* 进行处理 */
            if (!fileExists) { // 文件不存在，则上传
                CopyFromLocalFile.copyFromLocalFile(conf, localFilePath,
remoteFilePath);
                System.out.println(localFilePath + " 已上传至 " + remoteFilePath);
            }
            else if (choice.equals("overwrite")) { // 选择覆盖
                CopyFromLocalFile.copyFromLocalFile(conf, localFilePath,
remoteFilePath);
                System.out.println(localFilePath + " 已覆盖 " + remoteFilePath);
```

```
            }
            else if (choice.equals("append")) { // 选择追加
                CopyFromLocalFile.appendToFile(conf, localFilePath, remoteFilePath);
                System.out.println(localFilePath + " 已追加至 " + remoteFilePath);
            }
        }
        catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

```
user32@master:~$ vim CopyFromLocalFile.java
user32@master:~$ javac -cp $(hadoop classpath) -d . CopyFromLocalFile.java
user32@master:~$ jar cfm CopyFromLocalFile.jar Manifest.txt CopyFromLocalFile.cl
ass
user32@master:~$ vim CopyFromLocalFile.java
user32@master:~$ hadoop jar CopyFromLocalFile.jar
/user/user32/text.txt 已存在.
/home/user32/hw2/text.txt 已追加至 /user/user32/text.txt
```

验证:

```
user32@master:~$ hadoop jar HDFSApi.jar
读取文件: /user/user32/text.txt

love
and
peace
~
读取完成
```

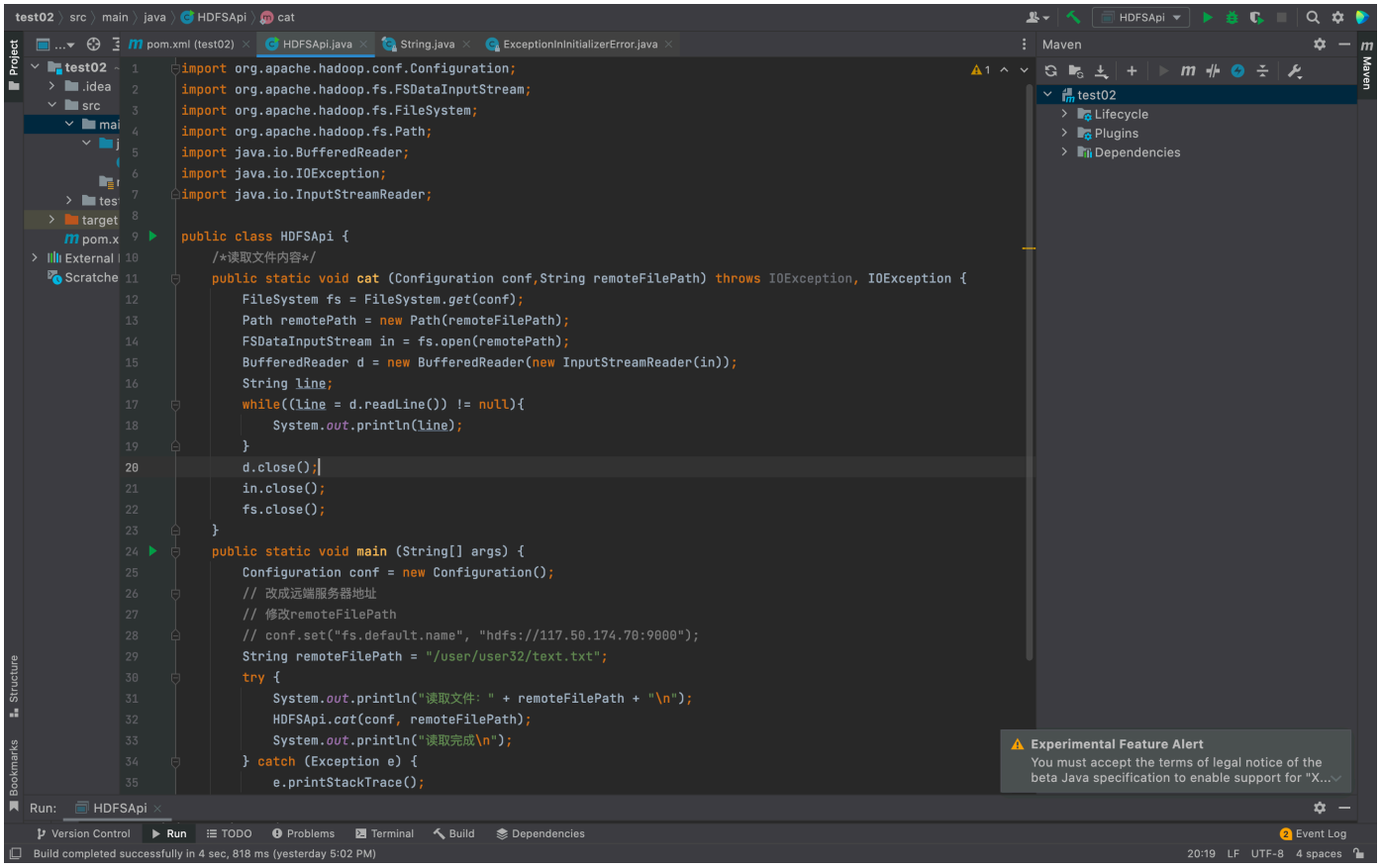
2.将HDFS中指定文件的内容输出到终端

Shell命令

```
$hdfs dfs -cat text.txt
```

```
user32@master:~$ hdfs dfs -cat text.txt
love
and
peace
~
```

Java代码实现




```java
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FSDataInputStream;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class HDFSApi {
    /*读取文件内容*/
    public static void cat (Configuration conf,String remoteFilePath) throws
IOException, IOException {
        FileSystem fs = FileSystem.get(conf);
        Path remotePath = new Path(remoteFilePath);
        FSDataInputStream in = fs.open(remotePath);
        BufferedReader d = new BufferedReader(new InputStreamReader(in));
        String line;
        while((line = d.readLine()) != null){
            System.out.println(line);
        }
        d.close();
        in.close();
        fs.close();
    }
    public static void main (String[] args) {
        Configuration conf = new Configuration();
        // 改成远端服务器地址
```

```
        // 修改remoteFilePath
        // conf.set("fs.default.name", "hdfs://117.50.174.70:9000");
        String remoteFilePath = "/user/user32/text.txt";
        try {
            System.out.println("读取文件: " + remoteFilePath + "\n");
            HDFSApi.cat(conf, remoteFilePath);
            System.out.println("读取完成\n");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

在本地/home/user32/hw2/text.txt目录下创建并编写text.txt 在hdfs文件目录下创建text.txt并且把home目录下的追加到/user/user32/text.txt

```
=[user32@master:~$ hdfs dfs -appendToFile /home/user32/hw2/text.txt text.txt
```

编写java

```
[user32@master:~$ vim HDFSApi.java
```

执行 javac 命令，编译时javac会在当前目录下创建子目录， 将所有.class放进该目录。

```
[user32@master:~$ jar cfm HDFSApi.jar Manifest.txt HDFSApi.class
```

编辑 Manifest.txt 文件，指出 main 函数所在的类文件

```
xy — user32@master: ~ — ssh user32@117.50.174.70 — 80×24
Main-Class: HDFSApi

~
```

执行 MapReduce 程序

```
[user32@master:~$ hadoop jar HDFSApi.jar
读取文件: /user/user32/text.txt

love
and
peace
~
读取完成
```

####

3.删除HDFS中指定的文件

Shell命令

```
$hdfs dfs -rm text.txt
```

```
[user32@master:~$ hdfs dfs -rm text.txt
Deleted text.txt
```

Java代码实现

```java
import org.apache.hadoop.configuration;
import orrg.apache.hadoop.fs.*;
import java.io.*;

public class HDFSApi{
  /*删除文件*/
  public static boolean rm(configuration conf,String remoteFilePath)throws
    IOException{
    FileSystem fs=FileSystem.get(conf);
    Path remotePath=new Path(remoteFilePath);
    boolean result =fs.delete(remoteParh,false);
    fs.close();
    retrun result;
  }
  /*主函数*/
  public static void main(String[] args)
  {
    Configuration conf=new Configuration();
    conf.set("fs.default.name","hdfs://localhost:9000");
    String remoteFilePath ="/user/hadoop/text.txt"; //HDFS文件
      try{
        if(HDFSApi.rm(conf,remoteFilePath))
        {
          System.out.printin("文件删除: "+remoteFilePath);
        }
        else
        {
          system.out.println("操作失败（文件不存在或删除失败）");
        }
        catch (Exception e)
        {
          e.printStackTrace();
        }
      }
  }
}
```

```
user32@master:~$ vim Remove.java
user32@master:~$ javac -cp $(hadoop classpath) -d . Remove.java
user32@master:~$ vim Manifest.txt
user32@master:~$ jar cfm Remove.jar Manifest.txt Remove.class
user32@master:~$ hadoop jar Remove.jar
删除文件: /user/user32/text.txt
```

验证:

```
user32@master:~$ hadoop fs -ls /user/user32
user32@master:~$
```

```java
import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;

public class Remove {
    /**
     * 删除文件
     */
    public static boolean rm(Configuration conf, String remoteFilePath) {
        Path remotePath = new Path(remoteFilePath);
        try (FileSystem fs = FileSystem.get(conf)) {
            return fs.delete(remotePath, false);
        } catch (IOException e) {
            e.printStackTrace();
            return false;
        }
    }
    public static void main(String[] args) {
        Configuration conf = new Configuration();
       // conf.set("fs.defaultFS", "hdfs://master:9000");
        String remoteFilePath = "/user/user32/text.txt"; // HDFS路径
        try {
            Remove.rm(conf, remoteFilePath); // 删除
            System.out.println("删除文件: " + remoteFilePath);
        }
        catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

二、编程实现一个类MyFSDataInputStream实现按行读取HDFS中指定文件的方法readLine()

注：该类继承org.apache.hadoop.fs.FSDataInputStream

要求：如果读到文件末尾，则返回空；否则返回文件一行的文本。

```java
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FSDataInputStream;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import java.io.*;
public class MyFSDataInputStream extends FSDataInputStream {
  public MyFSDataInputStream(InputStream in) {
```

```
    super(in);
  }
  /**
    *实现按行读取
    *每次读入一个字符，遇到"n"结束，返回一行内容
    */
  public static String readline(BufferedReader br) throws IOException {
    char[] data = new char[1024];
    int read = -1;
    int off = 0; //循环执行时，br每次会从上- -次读取结束的位置继续读取，因此该函数里，off每次都从0
开始
    while ( (read = br.read(data, off, 1)) != -1 ) {
      if (String.valueOf(data[off]).equals("\n") ) {
      off += 1;
      break;
      }
      off += 1;
    }
    if (off> 0) {
      return String.valueOf( data);
    } else {
    return null;
    }
  }
  /**
    *读取文件内容
    */
    public static void cat(Configuration conf, String remoteFilePath) throws
IOException {
      FileSystem fs = FileSystem.get(conf);
        Path remotePath = new Path(remoteFilePath);
      FSDataInputStream in = fs.open(remotePath);
      BufferedReader br = new BufferedReader(new InputStreamReader(in));
      String line = null;
      while ( (line = MyFSDataInputStream.readline(br)) != null ) {
        System.out.println(line);
      }
      br.close();
      in.close();
      fs.close();
    }
  /**
   *主函数
   */
  public static void main(String[] args) {
    Configuration conf = new Configuration();
    conf.set(" fs.default.name", "hdfs://localhost:9000");
    String remoteFilePath = "/user/hadoop/file/text.txt"; // HDFS路径
    try {
```

```
      MyFSDataInputStream.cat(conf,remoteFilePath);
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

```
user32@master:~$ hdfs dfs -appendToFile /home/user32/hw2/text2.txt /user/user32/
text2.txt
user32@master:~$ hdfs dfs -cat text2.txt
I would be complex
I would be cool
They'd say I played the field before I found someone to commit to
And that would be ok
For me to do
Every conquest I had made would make me more of a boss to you
I'd be a fearless leader
I'd be an alpha type
When everyone believes ya
What's that like I'm so sick of running as fast as I can
Wondering if I'd get there quicker
```

```
user32@master:~$ hadoop jar MyFSDataInputStream.jar
I would be complex

I would be cool

They'd say I played the field before I found someone to commit to

And that would be ok

For me to do

Every conquest I had made would make me more of a boss to you

I'd be a fearless leader

I'd be an alpha type

When everyone believes ya

What's that like I'm so sick of running as fast as I can

Wondering if I'd get there quicker
```

三.查看Java帮助手册或其它资料输出HDFS中指定文件的文本到终端中

注：用"java.net.URL"和"org.apache.hadoop.fs.FsURLStreamHandlerFactory"编程完成

```
import org.apache.hadoop.fs.*;
import org.apache.hadoop.io.IOUtils;
import java.io.*;
import java.net.URL;
```

```java
public class HDFSApi {
    static {
        URL.setURLStreamHandlerFactory(new FsUrlStreamHandlerFactory());
    }

    /**
     * 主函数
     */
    public static void main(String[] args) throws Exception {

        String remoteFilePath = "hdfs://localhost:9000/user/hadoop/file/text.txt"; //
HDFS 文件
        InputStream in = null;
        try {
            /* 通过 URL 对象打开数据流，从中读取数据 */
            in = new URL(remoteFilePath).openStream();
            IOUtils.copyBytes(in, System.out, 4096, false);
        } finally {
            IOUtils.closeStream(in);
        }
    }
}
```

```
user32@master:~$ vim HDFSApiCmd.java
user32@master:~$ vim Manifest.txt

user32@master:~$ javac -cp $(hadoop classpath) -d . HDFSApiCmd.java
user32@master:~$ jar cfm HDFSApiCmd.jar Manifest.txt HDFSApiCmd.class
user32@master:~$ hadoop jar HDFSApiCmd.jar
I would be complex
I would be cool
They'd say I played the field before I found someone to commit to
And that would be ok
For me to do
Every conquest I had made would make me more of a boss to you
I'd be a fearless leader
I'd be an alpha type
When everyone believes ya
What's that like I'm so sick of running as fast as I can
Wondering if I'd get there quicker
```

四、思考题

1、 HDFS 文件系统怎样识别流水线断裂?

每个datanode会定期向namenode发送“心跳“信息，向它报告自己的状态。当数据节点发生故障，流水线断裂时，namenode就无法受到来自一些datanode的”心跳“信息，此时HDFS系统中namenode就能识别到流水线断裂，节点上的所有数据都会被标记为”不可读“，namenode也不会再给它们发送任何请求。

2、 崩掉的 DataNode，重新启动后，系统（datanode 进程）怎样处理其上完整数据块?

将数据块删除。

3、 2 副本 block，第三个副本何时补上?

namenode会定期检查数据块的副本情况，一旦发现2副本block数量小于冗余因子，就会启动数据冗余复制，为它生成新的副本。

4、 未经 ACK 的 packet 会重新写入流水线。流水线断裂后，可能出现同一个 packet 多次写入流水线的情况。在这种情况下，HDFS 文件系统会出错吗?

packet包被data queue管理，data streamer接受并处理data queue，向namenode申请blocks，获取用来存储的datanode列表，把它们排成一个pipeline管道，packet沿着管道呈流水线方式写入。在流水线上最后一个datanode成功存储之后会返回一个ACK信号，如果在此之前有一个datanode出现了故障，这个pipelline就会被关闭，出现故障的datanode就会从pipeline中移除，文件继续写入剩余datanode，因此即使多次写入流水线，但是写入的是不同的datanode，不会出错。