JOINT INSTITUTE
交大密西根学院

VE572
Methods and tools for big data
Lab 1 Report

Lin Zhi 5143709113
Liu Yihao 515370910207
Zhou Yanjun 515370910219
Zhou Zhuangzhuang 5143709184

28 May 2018

1 Server Access

The IP address of the server is 45.76.213.89. User can use the two private SSH key to login the server as administrative account.

2 Documentation

2.1 TCPServer

Functions & Classes in TCPServer.java
public static void main(String[] args) Listen the socket request from clients through port 7650 and create a new Handler to solve each command from clients.
public static class Handler A class extending Threads. It has functions as follows:

1. **Handler(Socket socket, int clientNumber)** Constructor.
2. **public void run()** Read commands from socket, call functions to deal with the command and close socket after the command finished.
3. **private boolean readLine(BufferedReader input, PrintWriter output)** Parse the commands and check whether a command has finished.
4. **private boolean runCommand(String[] args, PrintWriter output)** Check whether the commands are in the right order and format and deal with the correct commands by calling other functions.
5. **private String saveFile(int size, String type)** Get the input stream from the socket and save it as the file with a name consisted of date and hash string and the corresponding type.
6. **private void log(String message)** Print information for the server end.

2.2 DataExtractor

Functions & Classes in DataExtractor.java
public class DataExtractor

1. **public DataExtractor(String Filename)** Constructor.
2. **private class MeaQuantity** A class storing informatin for a measure quantity.

(a) **private void parseShort(ByteBuffer buffer)** Read short type data from ByteBuffer, store them in a private Number variable data and sum them up. We also provide parseLong for Integer, parseFloat for Float and parseDouble for Double.

(b) **private Number[] calculate()** Sort and find min and max for data.

(c) **private void calculateShort()** Call calculate() and find the median and average of the data. We also provide calculateLong for Integer, calculateFloat for Float and calculateDouble for Double.

3. **public void parse()** Extract and save the data from the XML file, including the Name, DataType, Quantity and Unit. The Quantity and Unit are first recorded as Id from MeaQuantity Node and then checked the Quantity Node or Unit Node to find the Name of them.

4. **public String query(String name, String op)** Check the Option from the command, search the min, max, median, sum or average of the measure quantity and format the output string.

5. **public void readBinary(String fileName)** Read the binary file and parse the data with the corresponding data type.

6. **public void writeXlsx(String fileName)** Write the quantities' name, unit and data in binary file into an excel file with poi package.

3 Progress

I. Own a remote server on a public cloud platform

We have set up a remote server at IP address 45.76.213.89 and upload the programs on it. Users can login the server as root account with their private keys.

II. Implement a socket protocol in Java

In TCPServer.java, we implement the socket protocol with the ServerSocket and Socket package. The server can recieve commands from the client with socket protocol. The commands can be dealt with are

1. BEGIN;
2. SIZE Type Size; Type is one of XML or BIN
3. QUERY Op Name; Op is one of MAX, MIN, AVG, SUM, and MEDIAN.
4. END;

We expect the commands are in the following order: BEGIN, SIZE XML, SIZE BIN, a certain number of QUERY, and END. If the command is not in such an order, the sever will report "Error: command unexpected". If an unexpected Op is receieved, the server will report ""Error: Op not available". Message "OK"

will be sent to client for each valid command and query result will be showed for QUERY command.

The corresponding BIN or XML file sent by client will be saved on the server. The file name is comprised by the date and a hash string to make sure its uniqueness.

III. Write a Java program to extract data from an XML file

In DataExtractor.java, we use external package dom4j to extract information from XML file. The recorded information includes Name, Unit, Quantity and the location like the startOffset, blocksize and valueOffset of the corresponding data in the BIN file.

IV. Dump the data from a binary file With the information extracted from the XML file, we locate the data for each measure quantity in the binary file, read it in little endian and save it. At the same time, we calculate the MAX, MIN, AVG, SUM, and MEDIAN for each measure quantity so that it can be used for the QUERY commands.

We also use external package poi to save the data from the binary file along with the name and unit of the measure quantities into an XLSX file.

V. Complete all the tasks into a remote server application

All the source files including a sample client are uploaded into the server. Users can compile and run the programs according to the following commands:

Install

mvn install

Run the server

mvn exec:java -Dexec.mainClass="TCPServer"

Run the client

mvn exec:java -Dexec.mainClass="TCPClient" -Dexec.args="<host>"

These commands can also be found in the README.md on the server.

4 Main Discoveries

Here are the main discoveries during the lab:

1. When setting up a socket connectiong, we learn that the server needs new threads to deal with the new commands. To do that, we define a class extending Thread to handle each command from client.

2. When extracting data from the XML file, we find a useful external library dom4j.

3. When reading data from the BIN file, we discover a convienient data type named ByteBuffer which can be set to read data as little endian and has functions to get variables in different data types.

5 Attempts And Failures

We attempt to read the binary file directly but notice that the result gives a number like "-6.5798...E307". The number seems so strange that we recheck the process and find that in Java the default I/O stream is big endian while it might be little endian for the binary file. Then we read the binary file in little endian and comfirm that it is right with the given decoding set of files.

We once confused about why for the same data type ieeefloat8, we have blocksize with both 8 and 18 in the validate.xml file. After more considerations, we realize that it might be a struct contained several different data type. And we successfully find another ieeefloat8 type and a dt_short type with blocksize of 18.

6 Self-evaluation

Tasks	Time (hh:mm)	Difficulty level				
		1	2	3	4	5
1. Own a remote server on a public cloud platform	02:30	●	○	○	○	○
2. Implement a socket protocol using Java	02:30	●	○	○	○	○
3. Deploy your socket program on your remote server	02:30	●	○	○	○	○
4. Write a Java program to extract data from an XML file	02:30	●	○	○	○	○
5. Dump the data from a binary file	02:30	●	○	○	○	○
6. Complete all the tasks into a remote server application	02:30	●	○	○	○	○