CS 421 – Computer Networks
Programming Assignment 2
ParallelCloudDownloader

Due: 05.05.2017

In this programming assignment, you are asked to implement a program in Java. This program is supposed to download an index file to obtain a list of possible locations to find and download the different parts of the file. Additionally, you are asked to write a report which you will evaluate the performance of the program for different number of parallel connections. For this programming assignment, you are not allowed to use any third party HTTP client libraries, or the HTTP specific core or non-core APIs supplied with the JDK including but not limited to *java.net.HttpURLConnection*. **Your project grade will be substantially penalized if you use one of these libraries.** The goal of this assignment is to make you familiar with the internals of the HTTP protocol and using any (third party, core or non-core JDK supplied) API providing any level of abstraction specific to the HTTP protocol is prohibited. In short, you have to implement your program using barely the Socket API of the JDK. If you have any doubt about what to use or not to use, please contact us.

Your program must be **a console application** (no graphical user interface(GUI) is required) and should be named as ParallelCloudDownloader (i.e., the name of the class that includes the main method should be CloudDownloader). Your program should run with the

java ParallelCloudDownloader <index_file> <username>:<password> <thread_count>

command where <index_file>, <username>:<password> and <thread_count> are the command-line arguments. The details of the command-line arguments are as follows:

- <index_file>: [Required] The URL of the index that includes the list of partial file locations and their authentication information.
- <username>:<password>: [Required] The authentication information of the index server. You will use "**Basic**" authentication method of the HTTP. For more information: https://en.wikipedia.org/wiki/Basic_access_authentication#Client_side. You can use *java.util.Base64* library for Base64 encoding when necessary.

- <thread_count>: [Required] The number of threads created per partial file.

When a user enters the command above, your program will send an HTTP GET request to the server to download the index file with URL <index_file>. If the index file is not found, i.e. the response is a message other than 200 OK, your program should print an error message to the command-line and exits. If the index file is found, then the response is a 200 OK message. In this case your program should continue and follow the URLs, available bytes and authentication information provided in the index file to download the complete file. However, during the download operation, your program must establish <thread_count> parallel connections concurrently with each server that includes a partial file, so your program must establish (<thread_count> * number of partial files) parallel connections in total. **You must not wait for the download of a partial file to start the download of another one.**

If your program successfully obtains the complete file, it saves the content under the directory in which your program runs. The name of the saved file is provided in the initial index file. A message showing the total download time of the complete file should be printed to the command-line.

Basically, your program;

1. Sends a HTTP GET to an initial server with authentication credentials.
2. Reads the index file and establishes required number of connections to each URL at the same time by sending HTTP GETs with correct range fields and authentication credentials.
3. Save the downloaded files into a single file.

All servers have different parts of the file, but none of them are overlapping. **This is different from the first programming assignment.** For example, if complete file is 1100 bytes and it is distributed to three servers, then first server contains bytes 1-367, second server contains bytes 338-734 and third server contains bytes 735-1100.

Therefore, your program should:

1. Calculate the range field of each part downloaded correctly.
2. Combine the files in correct order.

Assumptions and Hints

- Please refer to W3Cs RFC 2616 for details of the HTTP messages.
- You will assume the authentication provided for the all servers are correct. Note that there should be a colon ':' character between the endpoints.
- You will assume each line of the index file includes one file URL followed by its authentication and followed by its byte-range all in separate lines.
- You will assume that each consecutive server stores further bytes of the file. For example, if one server has 300-500 bytes, the next server will have byte 501. Therefore, you must connect all the servers.
- Your program will not save the index file to the local folder.
- Your program should print a message to the command-line to inform the user about the download time of the file.
- The number of bytes downloaded through each connection should differ by at most one byte. The number of bytes downloaded through each connection is n/k if n is divisible by k, where n and k respectively denote the number of bytes in the file and the number of connections. Otherwise, ($\lfloor n/k \rfloor$+1) bytes should be downloaded through the first ($n - \lfloor n/k \rfloor$ k) connections and $\lfloor n/k \rfloor$ bytes should be downloaded through the remaining connections.
- The downloaded file should be saved under the directory containing the source file ParallelCloudDownloader.java and the name of the file should be the same as the name of the downloaded file.
- You may use the following URL, authentication and resulting file to test your program:
    - 139.179.50.45/descriptor1.txt, karasan:ea225, http://hasan.balci.bilkent.edu.tr/smallText.txt
    - 139.179.50.45/descriptor2.txt, karasan:ea225, http://hasan.balci.bilkent.edu.tr/mediumText.txt
    - 139.179.50.45/descriptor3.txt, karasan:ea225, http://hasan.balci.bilkent.edu.tr/bigText.txt
- You will assume that all servers are accepting connections through port 80.
- Please make sure the file your program downloads is exactly same with the test file.
- Please contact your assistant if you have any doubt about the assignment.

Example

Let 111.111.11.11/descriptor.txt be the URL of the index file to be downloaded whose content is given as

TestFile1	Name of the file
1000	Size of the file in bytes
222.222.22.22/partial1.txt	URL of the first server
yarkin:cetin	Authentication of the first server
1-500	The range of the available bytes in this server
233.233.33.33/partial2.txt	URL of the second server
deniz:1234	Authentication of the second server
501-1000	The range of the available bytes in this server

The username of the server 1 is ‘cs’ and password is ‘421’

Example run 1 Let your program start with the

java ParallelCloudDownloader 111.111.11.11/descriptor.txt cs:421 5

command.

```
Command-line:
URL of the index file: 111.111.11.11/descriptor.txt
File size is 1000 Bytes
Index file is downloaded
There are 2 servers in the index
TextFile1 is downloaded in xyz miliseconds with 10 threads
```

Assignment Report

You need to submit a short report **in pdf format named as “**AliVelioglu20111222.pdf” where you evaluate your results in three samples given. (Note that the program will be evaluated on other servers and files.)

Your report must include at least the following:

- *number of threads vs download speed* graph that includes data from three sample files. In the x axis, there should be total number of threads that is used to download the file. In the y axis, there should be download speed (bytes/ms). For uniformity, just indicate the thread values **3** (1 per subfile), **15** (5 per subfile), **30** (10 per file), **60** (20 per file), **120** (40 per file) and **300** (100 per subfile).
- Evaluation on the effect of changing number of threads when downloading same file.

- Evaluation on the effect of changing the file size when downloading with same number of threads.
- Explanations of the effects observed.

Submission rules

You need to apply all the following rules in your submission. **You will lose points if you do not obey the submission rules below or your program does not run as described in the assignment above.**

- The assignment should be submitted as an e-mail attachment sent to hasan.balci[at]bilkent.edu.tr. Any other methods (Disk/CD/DVD) of submission will not be accepted.
- The subject of the e-mail should start with [CS421_PA2], and include your name and student ID. For example, the subject line must be

  [CS421_PA2]AliVelioglu20111222

  if your name and ID are Ali Velioglu and 20111222. If you are submitting an assignment done by two students, the subject line should include the names and IDs of both group members. The subject of the e-mail should be

  [CS421_PA2]AliVelioglu20111222AyseFatmaoglu20255666

  if group members are Ali Velioglu and Ayse Fatmaoglu with IDs 20111222 and 20255666, respectively.
- All the files must be submitted in a zip file whose name is the same as the subject line **except the [CS421_PA2] part**. The file must be a .zip file, not a .rar file or any other compressed file.
- All of the files must be in **the root of the zip file**; directory structures are not allowed. Please note that this also disallows organizing your code into Java packages. The archive should not contain:

  – Any class files or other executables,

  – Any third-party library archives (i.e. jar files),

  – Any text files **except the report in pdf format**,

  – Project files used by IDEs (e.g., JCreator, JBuilder, SunOne, Eclipse, Idea or NetBeans etc.). You may, and are encouraged to, use these programs while developing, but the end result must be a clean, IDE-independent program.
- The standard rules for plagiarism and academic honesty apply, if in doubt refer to ‘Student Disciplinary Rules and Regulation’, items 7.j, 8.l and 8.m.