CS 3430: Python & Perl
Assignment 8: Analyzing NASA Web Server Logs

Vladimir Kulyukin
Department of Computer Science
Utah State University

1. Learning Objectives

1. Regular Expressions
2. File Processing & Filtering
3. Pipes & Pipelines

2. Web Log Analysis & Pipes

This assignment will give you a flavor of what it is like to be a sys or cloud admin. One of the most useful data sources are web server logs. Different web servers can be configured differently to log various pieces of information. But a typical web server log is just a text file where each line is a access entry record. Some web server logs are publicly available. I got the data for this assignment from http://ita.ee.lbl.gov/html/contrib/NASA-HTTP.html. The data are available in **nasa_data.zip**. Before I tell you what is in that zip archive, let me briefly analyze the structure of a typical web log entry. Below are three entries from **access_01jul1995.txt** in **nasa_data.zip**

```
199.72.81.55 - - [01/Jul/1995:00:00:01 -0400] "GET /history/apollo/ HTTP/1.0" 200 6245
unicomp6.unicomp.net - - [01/Jul/1995:00:00:06 -0400] "GET /shuttle/countdown/ HTTP/1.0" 200 3985
199.120.110.21 - - [01/Jul/1995:00:00:09 -0400] "GET /shuttle/missions/sts-73/mission-sts-73.html HTTP/1.0" 200 4085
```

The first segment in each entry, e.g., 199.72.81.55 or unicomp6.unicomp.net, is the IP address. The next segment after the IP, - -, is the user name and password. Many sys admins do not make it publicly available and simply put dashes or stars in their place. Following the user login info is the timestamp, e.g., [01/Jul/1995:00:00:01 -0400]. Then come the request, e.g., "GET /history/apollo/ HTTP/1.0", the result status code, e.g., 200, and the number of bytes transferred, e.g., 3985. Many log entries also contain the user agent segment that specifies the browser type. But this segment is missing in the NASA web logs.

In this assignment, we will be dealing with the IPs, result status codes, and the number of bytes transferred. Recall from Lecture 18 on pipes that the classical Unix/Linux programming paradigm is to develop and maintain a large number of relatively small programs (scripts) that can be used to solve various problems by feeding their outputs to each other and then on to other programs. A mechanism of feeding the output of one program as input to another program is called a pipe. A sequence of programs that uses pipes to produce the right output is called a pipeline. A program in a pipeline can be referred to as a pipe segment.

2. Contents of NASA_DATA.ZIP

The archive has two small files (both have the prefix **small_**) which you should use to develop the scripts described in the next section. When your scripts are working on the small files, you can test them on the other ten access files that contain web log entries of a NASA Apache server for the first 10 days of July, 1995. Here are the contents of small_access_log_Jul95_01.txt:

```
199.72.81.55 - - [01/Jul/1995:00:00:01 -0400] "GET /history/apollo/ HTTP/1.0" 200 6245
unicomp6.unicomp.net - - [01/Jul/1995:00:00:06 -0400] "GET /shuttle/countdown/ HTTP/1.0" 200 3985
199.120.110.21 - - [01/Jul/1995:00:00:09 -0400] "GET /shuttle/missions/sts-73/mission-sts-73.html HTTP/1.0" 200 4085
burger.letters.com - - [01/Jul/1995:00:00:11 -0400] "GET /shuttle/countdown/liftoff.html HTTP/1.0" 304 0
199.120.110.21 - - [01/Jul/1995:00:00:11 -0400] "GET /shuttle/missions/sts-73/sts-73-patch-small.gif HTTP/1.0" 200 4179
burger.letters.com - - [01/Jul/1995:00:00:12 -0400] "GET /images/NASA-logosmall.gif HTTP/1.0" 304 0
burger.letters.com - - [01/Jul/1995:00:00:12 -0400] "GET /shuttle/countdown/video/livevideo.gif HTTP/1.0" 200 0
163.205.53.14 - - [28/Jul/1995:13:32:22 -0400] "GET /shuttle/technology/images/srb_mod_compare_3-small.gif HTTP/1.0" 200 55666
163.205.53.14 - - [28/Jul/1995:13:32:22 -0400] "GET /shuttle/technology/images/srb_mod_compare_6-small.gif HTTP/1.0" 200 28219
163.205.53.14 - - [28/Jul/1995:13:32:23 -0400] "GET /images/KSC-logosmall.gif HTTP/1.0" 200 1204
tiger2.ocs.lsu.edu - - [28/Jul/1995:13:32:23 -0400] "GET /shuttle/missions/missions.html HTTP/1.0" 200 8677
199.0.2.27 - - [28/Jul/1995:13:32:23 -0400] "GET /images/ksclogo-medium.gif HTTP/1.0" 200 5866
tornado.umd.edu - - [28/Jul/1995:13:32:25 -0400] "GET /shuttle/missions/sts-74/sts-74-patch-small.gif HTTP/1.0" 200 5494
```

3. Pipe Segments

In this section, I will describe the scripts that you will write for this assignment. Each script must be written both in Python and Perl and must take its input from STDIN and write its output to STDOUT, as we did in class. Each script should be at most 15 lines long, including the use, import, and #! pragmas. If you find yourself writing more code for a script, stop and think a little about the problem. Your scripts should have no functions and subroutines. Just statements and regexes.

I would encourage you to spend 15-30 minutes a day on just one script instead of putting it off until Saturday. Write and debug each pipe segment thoroughly, build and debug a small pipeline out of two or three segments and then move on to longer pipelines.

Several people asked me which language they should start with on each assignment, Python and Perl. The answer is, it depends. If you feel more comfortable with Python, as I suspect, most of you do, start with Python and then port your solution to Perl. I would make one exception to this rule on regexes. Since Python borrowed 99% of the Perl regex matching engine, I would develop and test all my regexes in Perl first. Once they work in Perl, all I need to do is copy and paste them into Python. One regex is sufficient for all of the scripts described below.

3.1 Segment 1: FLTR_WLOG_TRBTS.PY & FLTR_WLOG_TRBTS.PL

This name stands for “filter web log’s transferred bytes.” This script takes a web log file from STDIN and outputs into STDOUT the number of transferred bytes for each log entry. Here are two sample runs. Note that since most of you develop on Windows and some of you have had little or no experience with command lines (unfortunately, modern IDEs seem to do their best hiding them away from developers), my examples below use Windows command line syntax. If you are a Linux developer (Macs have Linux-like terminals), replace back slashes with forward slashes. On Windows, **type** is the command that prints the contents of a file into the command line window. On Linux, replace it with **more**. The **more** command is available on Windows as well. Sometimes, if your script does not like **type**, try **more**. In the command line example below the output of **type** is piped into **fltr_wlog_trbts.py** which filters each web log entry for the number bytes transferred and prints each number on a separate line into the STDOUT.

```
>type ..\..\nasa_data\small_access_log_Jul95_01.txt | python fltr_wlog_trbts.py
6245
3985
4085
0
4179
0
0
55666
28219
1204
8677
5866
5494
```

```
>type ..\..\nasa_data\small_access_log_Jul95_01.txt | perl fltr_wlog_trbts.pl
6245
3985
4085
0
4179
0
0
55666
28219
1204
8677
5866
5494
```

3.2 Segment 2: FLTR_WLOG_IP_TRBTS_TOOPS.PY & FLTR_WLOG_IP_TRBTS_TOOPS.PL

This name stands for "filter web log's ip and transferred bytes toops." This script takes a web log file from STDIN, converts each web log entry line into a 2-toop which consists of an IP and the corresponding number of bytes transferred and outputs that toop into STDOUT on a separate line.

```
>type ..\..\nasa_data\small_access_log_Jul95_01.txt | perl fltr_wlog_ip_trbts_toops.pl
199.72.81.55 6245
unicomp6.unicomp.net 3985
199.120.110.21 4085
burger.letters.com 0
199.120.110.21 4179
burger.letters.com 0
burger.letters.com 0
163.205.53.14 55666
163.205.53.14 28219
163.205.53.14 1204
tiger2.ocs.lsu.edu 8677
```

```
199.0.2.27 5866
tornado.umd.edu 5494

>type ..\..\nasa_data\small_access_log_Jul95_01.txt | python fltr_wlog_ip_trbts_toops.py
199.72.81.55 6245
unicomp6.unicomp.net 3985
199.120.110.21 4085
burger.letters.com 0
199.120.110.21 4179
burger.letters.com 0
burger.letters.com 0
163.205.53.14 55666
163.205.53.14 28219
163.205.53.14 1204
tiger2.ocs.lsu.edu 8677
199.0.2.27 5866
tornado.umd.edu 5494
```

3.3 Segment 3: FLTR_WLOG_STATUS_CODES.PY & FLTR_WLOG_STATUS_CODES.PL

The name stands for "filter web log's status codes." This script takes a web log file from STDIN and a command line argument that specifies the status code, e.g., 200. The script outputs all web log entries with that status code into STDOUT.

```
>type ..\..\nasa_data\small_access_log_Jul95_01.txt | python fltr_wlog_status_codes.py 200
199.72.81.55 - - [01/Jul/1995:00:00:01 -0400] "GET /history/apollo/ HTTP/1.0" 200 6245
unicomp6.unicomp.net - - [01/Jul/1995:00:00:06 -0400] "GET /shuttle/countdown/ HTTP/1.0" 200 3985
199.120.110.21 - - [01/Jul/1995:00:00:09 -0400] "GET /shuttle/missions/sts-73/mission-sts-73.html HTTP/1.0" 200 4085
199.120.110.21 - - [01/Jul/1995:00:00:11 -0400] "GET /shuttle/missions/sts-73/sts-73-patch-small.gif HTTP/1.0" 200 4179
burger.letters.com - - [01/Jul/1995:00:00:12 -0400] "GET /shuttle/countdown/video/livevideo.gif HTTP/1.0" 200 0
163.205.53.14 - - [28/Jul/1995:13:32:22 -0400] "GET /shuttle/technology/images/srb_mod_compare_3-small.gif HTTP/1.0" 200 55666
163.205.53.14 - - [28/Jul/1995:13:32:22 -0400] "GET /shuttle/technology/images/srb_mod_compare_6-small.gif HTTP/1.0" 200 28219
163.205.53.14 - - [28/Jul/1995:13:32:23 -0400] "GET /images/KSC-logosmall.gif HTTP/1.0" 200 1204
tiger2.ocs.lsu.edu - - [28/Jul/1995:13:32:23 -0400] "GET /shuttle/missions/missions.html HTTP/1.0" 200 8677
199.0.2.27 - - [28/Jul/1995:13:32:23 -0400] "GET /images/ksclogo-medium.gif HTTP/1.0" 200 5866
tornado.umd.edu - - [28/Jul/1995:13:32:25 -0400] "GET /shuttle/missions/sts-74/sts-74-patch-small.gif HTTP/1.0" 200 5494

>type ..\..\nasa_data\small_access_log_Jul95_01.txt | perl fltr_wlog_status_codes.pl 200
199.72.81.55 - - [01/Jul/1995:00:00:01 -0400] "GET /history/apollo/ HTTP/1.0" 200 6245
unicomp6.unicomp.net - - [01/Jul/1995:00:00:06 -0400] "GET /shuttle/countdown/ HTTP/1.0" 200 3985
199.120.110.21 - - [01/Jul/1995:00:00:09 -0400] "GET /shuttle/missions/sts-73/mission-sts-73.html HTTP/1.0" 200 4085
199.120.110.21 - - [01/Jul/1995:00:00:11 -0400] "GET /shuttle/missions/sts-73/sts-73-patch-small.gif HTTP/1.0" 200 4179
burger.letters.com - - [01/Jul/1995:00:00:12 -0400] "GET /shuttle/countdown/video/livevideo.gif HTTP/1.0" 200 0
163.205.53.14 - - [28/Jul/1995:13:32:22 -0400] "GET /shuttle/technology/images/srb_mod_compare_3-small.gif HTTP/1.0" 200 55666
163.205.53.14 - - [28/Jul/1995:13:32:22 -0400] "GET /shuttle/technology/images/srb_mod_compare_6-small.gif HTTP/1.0" 200 28219
163.205.53.14 - - [28/Jul/1995:13:32:23 -0400] "GET /images/KSC-logosmall.gif HTTP/1.0" 200 1204
tiger2.ocs.lsu.edu - - [28/Jul/1995:13:32:23 -0400] "GET /shuttle/missions/missions.html HTTP/1.0" 200 8677
199.0.2.27 - - [28/Jul/1995:13:32:23 -0400] "GET /images/ksclogo-medium.gif HTTP/1.0" 200 5866
tornado.umd.edu - - [28/Jul/1995:13:32:25 -0400] "GET /shuttle/missions/sts-74/sts-74-patch-small.gif HTTP/1.0" 200 5494
```

3.4 Segment 4: COUNT_LINES & TOP_N

These two scripts simply count no-white-space lines and output the top n lines from the STDIN, respectively. Here is a possible Py solution followed by its PL equivalent for count_lines.

```
#!/usr/bin/python
import sys
import re
sys.stdout.write(str(len([line for line in sys.stdin.readlines() if re.match(r'\S+', line)]))+'\n');
sys.stdout.flush();
```

```
#!/usr/bin/perl
use strict;
use warnings;
my $line_count = 0;
while ( <> ) { $line_count++ if ( $_ =~ /\S+/ ); }
print $line_count, "\n";
```

Note the use of **sys.stdout.write** in the Py solution. I recommend that you use it instead of print. It is more robust and, since **sys.stdout** is a file stream, it can be flushed. By the way, as I indicated above, your other scripts should not be significantly longer that these two scripts if you use wisely your regexes, list comprehensions, etc.

3.5 Segment 4: DSC_SORT_IP_TRBTS_TOOPS.PY & DSC_SORT_IP_TRBTS_TOOPS.PL

The name stands for “descending sort of the IP and transferred bytes toops.” As the name suggests, these scripts are designed to work in conjunction with **FLTR_WLOG_IP_TRBTS_TOOPS** described above. They take the 2-toops of ips and transferred bytes and sort them from largest to smallest by the number of transferred bytes.

```
>type ..\..\nasa_data\small_access_log_Jul95_01.txt | python fltr_wlog_ip_trbts_toops.py | python dsc_sort_ip_trbts_toops.py
163.205.53.14 55666
163.205.53.14 28219
tiger2.ocs.lsu.edu 8677
199.72.81.55 6245
199.0.2.27 5866
tornado.umd.edu 5494
199.120.110.21 4179
199.120.110.21 4085
unicomp6.unicomp.net 3985
163.205.53.14 1204
burger.letters.com 0
burger.letters.com 0
burger.letters.com 0
```

```
>type ..\..\nasa_data\small_access_log_Jul95_01.txt | perl fltr_wlog_ip_trbts_toops.pl | perl dsc_sort_ip_trbts_toops.pl
163.205.53.14 55666
163.205.53.14 28219
tiger2.ocs.lsu.edu 8677
```

```
199.72.81.55 6245
199.0.2.27 5866
tornado.umd.edu 5494
199.120.110.21 4179
199.120.110.21 4085
unicomp6.unicomp.net 3985
163.205.53.14 1204
burger.letters.com 0
burger.letters.com 0
burger.letters.com 0
```

```
>type ..\..\nasa_data\small_access_log_Jul95_01.txt | python fltr_wlog_ip_trbts_toops.py | python dsc_sort_ip_trbts_toops.py
163.205.53.14 55666
163.205.53.14 28219
tiger2.ocs.lsu.edu 8677
199.72.81.55 6245
199.0.2.27 5866
tornado.umd.edu 5494
199.120.110.21 4179
199.120.110.21 4085
unicomp6.unicomp.net 3985
163.205.53.14 1204
burger.letters.com 0
burger.letters.com 0
burger.letters.com 0
```

4. Pipelines

Now we use our pipe segments to construct pipelines. Let us construct a pipeline to answer the question: how many bytes have been transferred on a given date. Here is a possible solution that uses the **sum_stdin** script that we developed in Lecture 18. Recall that it computes the sum of integers from STDIN.

```
>type ..\..\nasa_data\access_01jul1995.txt | python fltr_wlog_trbts.py | python sum_stdin.py
1617628039
```

```
>type ..\..\nasa_data\access_01jul1995.txt | perl fltr_wlog_trbts.pl | perl sum_stdin.pl
1617628039
```

Evidently, there were a lot of bytes transferred on July 1, 1995!

Now construct the pipelines to answer the following questions:

1) What are the top 5 IPs that transferred the largest numbers of bytes on a given date?
2) How many requests on a given date had the status code of 200, i.e., were fielded by the server correctly?

5. What to Submit

Submit your Py and PL scripts outlined in sections 3.1 – 3.5. Do not change the script names. They will be tested in the pipelines described in Section 4 by a grading script.

Happy Hacking!