CSCI415—Systems Programming

Spring 2014

Programming Assignment 5

Multithreaded httpd Reference Stats

This project requires that you design and implement a multithreaded program that performs some rudimentary analysis of the `access_log` that records accesses made to a running Apache web server. Our server had been configured to record the source of all requests as dotted decimal IP addresses to avoid lots of name server access during normal operation. As such, a line from the `access_log` will appear as:

```
  128.239.220.66 - - [11/Apr/2011:12:06:00 -0400] "GET \
        /~kearns/415S14/index.html HTTP/1.1" 304 0 "-" \
               "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
```

For the purposes of this assignment, the first item on a log line is the only significant one. The above line was a result of a browser at IP address 128.239.220.66 accessing the top-level page for the course. Again, the IP address, not the fully qualified domain name (FQDN), is recorded as the source of the request.

The log files I will give you for this assignment will consist only of the IP address; the rest of the line, especially the target URL, are deleted to protect the privacy of users on our network.

Program Operation

Your program must be invoked as

`refstats -b` *numlines* `-N` *cachesize* `-d` *filedelay* `-D` *threaddelay* *file* `...`

The end effect of this program's execution is the output (on `stdout`) of a table of the number of log records from hosts, listed by FQDN, not IP address. The output should be in ascending lexicographic order. Only if a FQDN can not be obtained for an IP address (and this is not an uncommon case), the IP address should be used instead.

Required Thread Structure

You must have a **reader thread** whose job is to input lines of the designated `access_log` files and to store those lines in a buffer. The reader thread is the only thread to engage in file input. After reading a line of input, you must have the reader thread sleep for *filedelay* milliseconds as specified in the `-d` option on the command line.

The buffer must contain sufficient space to store *numlines* lines of log file contents (actually, dotted decimal IP addresses) as specified in the `-b` command line option.

You must also have two **lookup threads**. A lookup thread will (in a loop) get a line of text from the buffer and will map the IP address into a FQDN. After a lookup thread gets a line of text from the buffer, it must sleep for *threaddelay* milliseconds, as specified in the `-D` command line option, before obtaining the FQDN and updating the statistics.

Ultimately, after all input is complete, the program (**main thread**) will output a nicely formatted table.

A Performance Boost: a Cache

Accessing a domain name server is a nontrivial operation. We furthermore expect that a remote site may make several httpd accesses over a short time interval. Accordingly, your program must maintain a cache of (IP address, FQDN) pairs that it has seen in the recent past. Before querying the name server, a lookup thread should scan the cache to see if the IP address it is looking up is cached. This is a *cache hit*. On a cache hit, there is no need to query the name server. However, if the IP address is not in the cache (a *cache miss*), then the name server must be consulted. In addition to updating reference stats, the new (IP address, FQDN) should be entered in the cache. The goal of the cache is to minimize the number of times we must interact with the name server.

The cache must be able to hold *cachesize* pairs. If the cache is full, a cache miss requires that some pair in the cache be replaced. You must implement a "Least Recently Used" replacement strategy: the pair that is replaced is the one last used furthest in the past.

The lookup threads are jointly responsible for cache maintenance. The main thread must also output the cache hit ratio (the number of hits divided by the total number of lines in the log files) after outputting the reference stats list.

If an IP address cannot be resolved to an FQDN, you should have a cache entry for that IP address that prevents repeated resolution requests to the name server.

Your Source Code

The TA will examine your source code as part of the grading procedure. He will do this to make sure that the structure of your system is in keeping with the specifications. You would be well-advised to produce neat and readable source with a level of commentary to help the TA in his task.

Notes

The following points should be considered as you design your system:

- You must enforce that the input files contain valid dotted decimal quads. A bad input line should be silently ignored.
- Command line options may appear in any order. All options, and the associated values, are required for execution of the program. You must check command line arguments for validity, aborting with an appropriate message if there is a problem. The following rules define valid values for the options:
  - *numlines* is a positive non-zero integer that can be as large as 1000;
  - *cachesize* is a positive non-zero integer that can be as large as 10000;
  - *filedelay* is a positive integer (zero allowed); and
  - *threaddelay* is a positive integer (zero allowed).
- A file (path) name can be no longer than 4096 characters.
- All command line tokens after the required options are assumed to be file names.
- I suggest checking out the manual page for `nanosleep(2)` as the preferred way to arrange for required intervals of sleeping for your threads.
- Note that departmental systems may have odd FQDNs in the sense that host names without the `CS.WM.EDU` are the official names. This should not be considered an issue for your program. Accept the short name as the FQDN.
- You **must** use the deprecated `gethostbyaddr()` function. Note that this function is not threadsafe. You need to cope with this.

Due Date

See the description on the assignments web page.

Preparing Your Submission

You must put your source files in a single directory; call it `stats.d`. That directory should also contain your `Makefile`. Make sure that `stats.d` is protected 0700 to prevent others from looking at your source. Make the *parent* of `stats.d` your working directory and do the following:

- `tar zcvf stats.tgz stats.d`
- `gpg -r CSCI315 --sign -e stats.tgz`
- `mv stats.tgz.gpg ~ ; chmod 0644 ~/stats.tgz.gpg`

The TA will unpack your submission and change to the directory in which your source lives to do the `make`. She will then run your program from that directory. Note that the log files need not be in that directory.