Concurrency & Proxy Lab

Recitation 13: November 28, 2017

Part A

Visualizing Concurrency: Progress Graphs

Assembly Code for Counter Loop

C code for counter loop in thread i

```
Asm code for thread i
```

movq testq jle	(%rdi), %rcx %rcx,%rcx .L2	<i>H</i> _{<i>i</i>} : Head
movl	\$0, %eax	
.L3: movq addq movq	cnt(%rip),%rdx \$1, %rdx %rdx, cnt(%rip)	<pre> L_i: Load cnt U_i: Update cnt S_i: Store cnt </pre>
addq cmpq jne .L2:	\$1, %rax %rcx, %rax .L3	T_i : Tail

11/20/2017 (©J.P. Shen)

Visualizing Concurrency: Progress Graphs

- Recall this ordering from yesterday's lecture slide 10
 - Incorrect ordering: two threads increment the counter, but the result is 1 instead of 2

i (thread)	instr _i	%rdx ₁	%rdx ₂	cnt	
1	H ₁	-	-	0	
1	L ₁	0	-	0	
1	U ₁	1	-	0	
2	H ₂	-	-	0	
2	L ₂	-	0	0	
1	S ₁	1	-	1	
1	T ₁	1	-	1	
2	U ₂	-	1	1	
2	S ₂	-	1	1	
2	T ₂	-	1	1	Oops!

We can analyze the behavior using a progress graph

Progress Graphs



A *progress graph* depicts the discrete *execution state space* of concurrent threads.

Each axis corresponds to the sequential order of instructions in a thread.

Each point corresponds to a possible *execution state* ($Inst_1$, $Inst_2$).

E.g., (L_1, S_2) denotes state where thread 1 has completed L_1 and thread 2 has completed S_2 .

11/20/2017 (©J.P. Shen)

Trajectories in Progress Graphs



A *trajectory* is a sequence of legal state transitions that describes one possible concurrent execution of the threads.

Example:

H1, L1, U1, H2, L2, S1, T1, U2, S2, T2

Critical Sections and Unsafe Regions



L, U, and S form a *critical section* with respect to the shared variable cnt

Instructions in critical sections (wrt some shared variable) should not be interleaved

Sets of states where such interleaving occurs form *unsafe regions*

Trajectory safeness and correctness



Def: A trajectory is *safe* iff it does not enter any unsafe region

Claim: A trajectory is correct (wrt cnt) iff it is safe

Deadlock Visualized in Progress Graph



Locking introduces the potential for *deadlock:* waiting for a condition that will never be true

Any trajectory that enters the *deadlock region* will eventually reach the *deadlock state*, waiting for either s_0 or s_1 to become nonzero

Other trajectories luck out and skirt the deadlock region

Unfortunate fact: deadlock is often nondeterministic (race)

Avoided Deadlock in Progress Graph



No way for trajectory to get stuck

Processes acquire locks in same order

Order in which locks released immaterial

10

Part B

Getting content on the web: Telnet/cURL

- How the web really works
- Networking Basics
- Echo Client & Server Demo
- Proxy
 - Due Tuesday, December 12th
 - Grace days allowed
- String Manipulation in C

The Web in a Textbook

Client request page, server provides, transaction done.



- A sequential server can handle this. We just need to serve one page at a time.
- This works great for simple text pages with embedded styles.

Telnet/Curl

Telnet

- Interactive remote shell like ssh without security
- Must build HTTP request manually
 - This can be useful if you want to test response to malformed headers

```
[rjaganna@makoshark ~]% telnet www.cmu.edu 80
Trying 128.2.42.52...
Connected to WWW-CMU-PROD-VIP.ANDREW.cmu.edu (128.2.42.52).
Escape character is '^]'.
GET http://www.cmu.edu/ HTTP/1.0
HTTP/1.1 301 Moved Permanently
Date: Sat, 11 Apr 2015 06:54:39 GMT
Server: Apache/1.3.42 (Unix) mod gzip/1.3.26.1a mod pubcookie/3.3.4a mod ssl/2.8.31 OpenSSL/0.9.8e-
                                                                                                           fips-rhel5
Location: http://www.cmu.edu/index.shtml
Connection: close
Content-Type: text/html; charset=iso-8859-1
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<HTML><HEAD>
<TITLE>301 Moved Permanently</TITLE>
</HEAD><BODY>
<H1>Moved Permanently</H1>
The document has moved <A HREF="http://www.cmu.edu/index.shtml">here</A>.<P>
<HR>
<ADDRESS>Apache/1.3.42 Server at <A HREF="mailto:webmaster@andrew.cmu.edu">www.cmu.edu</A> Port
                                                                                                            80</ADDRESS>
</BODY></HTML>
Connection closed by foreign host.
```

Telnet/cURL

cURL

- "URL transfer library" with a command line program
- Builds valid HTTP requests for you!

```
[prodney@makoshark ~]% curl http://www.cmu.edu/
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<HTML><HEAD>
<TITLE>301 Moved Permanently</TITLE>
</HEAD><BODY>
<H1>Moved Permanently</H1>
The document has moved <A HREF="http://www.cmu.edu/index.shtml">here</A>.<P>
<HR>
<ADDRESS>Apache/1.3.42 Server at <A HREF="mailto:webmaster@andrew.cmu.edu">www.cmu.edu</A> Port
80</ADDRESS>
</BODY></HTML>
```

• Can also be used to generate HTTP proxy requests:

```
[prodney@makoshark ~]% curl --proxy lemonshark.ics.cs.cmu.edu:3092 http://www.cmu.edu/
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<HTML><HEAD>
<TITLE>301 Moved Permanently</TITLE>
</HEAD><8DUY>
<H1>Moved Permanently</H1>
The document has moved <A HREF="http://www.cmu.edu/index.shtml">here</A>.<P>
<HR>
<ADDRESS>Apache/1.3.42 Server at <A HREF="mailto:webmaster@andrew.cmu.edu">www.cmu.edu</A> Port 80</ADDRESS>
</BODY></HTML>
```

How the Web Really Works

- In reality, a single HTML page today may depend on 10s or 100s of support files (images, stylesheets, scripts, etc.)
 - Builds a good argument for concurrent servers
 - Just to load a single modern webpage, the client would have to wait for 10s of back-to-back request
 - I/O is likely slower than processing, so back
- Caching is simpler if done in pieces rather than whole page
 - If only part of the page changes, no need to fetch old parts again
 - Each object (image, stylesheet, script) already has a unique URL that can be used as a key

Sequential Proxy

			Carnegi	e Mellon University	I CMU				
Carnegie Vertex vertex									
🚧 💱 🔇 📚 🗮 💌 Console HTML (,,, Clear Persist All HTML CSS JS	CSS Script DOM No XHR Images Flash	et - Cookies Media			ç	P			
URL	Status	Domain	Size	Remote IP	Timeline				
GET index.shtml	200 OK	cmu.edu	14 KB	128.2.220.20:3092	llms				
► GET cmu.css	200 OK	cmu.edu	47.5 KB	128.2.220.20:3092	13ms				
GET cmu-new.css	200 OK	cmu.edu	18.5 KB	128.2.220.20:3092	21ms				
► GET cmu-new-print.css	200 OK	cmu.edu	1.8 KB	128.2.220.20:3092	40ms				
GET dojo.js	200 OK	cmu.edu	126.2 KB	128.2.220.20:3092	40ms				
GET scripts.js	200 OK	cmu.edu	6.8 KB	128.2.220.20:3092	55ms				
GET jquery.js	200 OK	cmu.edu	70.5 KB	128.2.220.20:3092	67ms				
GET homepage.js	200 OK	cmu.edu	1.1 KB	128.2.220.20:3092	68ms				
GET app_ad.js	200 OK	cmu.edu	2.6 KB	128.2.220.20:3092	75ms				
► GET ga.js	200 OK	google-analytics.com	15.3 KB	128.2.220.20:3092		40ms			
GET CarnegieMellonUniversity_wordma	200 OK	cmu.edu	48.7 KB	128.2.220.20:3092		55ms			
▶ GET btn_go.gif	200 OK	cmu.edu	166 B	128.2.220.20:3092		45ms			
GET mellon_institute_748x460.jpg	200 OK	cmu.edu	93 KB	128.2.220.20:3092		75ms			
GET autism_research_month_240x126.j	200 OK	cmu.edu	29.8 KB	128.2.220.20:3092		81ms			
GET poets_on_tour_240x126.jpg	200 OK	cmu.edu	19.8 KB	128.2.220.20:3092		97ms			
GET pres_lared_conon_240x126.jpg	200 OK	cmu.edu	13 KB	128.2.220.20:3092		61ms			
E GET lang li alf	200 OK	cmu.edu	4.4 KB	128.2.220.20:3092		119ms			
GET logo-il.gif	200 OK	cmu.edu	5.3 KB	128.2.220.20:3092		87ms			
GET logo-cmtoday.gir	200 OK	cmu.edu	49 KB	128.2.220.20:3092		116ms			
CET mollon institute 748x460 ins	200 OK	cmu.edu	5.9 KB	128.2.220.20.3092		92ms			
► GET transGrav85 pmg	200.0K	cmu edu	93 KB	128.2.220.20.3092		1160			
GET horizontal rule gif	200.0K	cmu edu	3 2 8	128 2 220 20.3092		1190	36 mr		
► GET utm oif?utmwy=5.4.1&u cmd%21	200 OK	google-analytics.com	35.0	128 2 220 20:3092		1	30ms		
24 requests	200.010	google-analytics.com	670 4 KB	120.2.220.20.3092			528ms (onload: 591ms)		
z4 requests			070.4 KB				Szoms (officad: S91ms)		

Sequential Proxy

Note the sloped shape of when requests finish

- Although many requests are made at once, the proxy does not accept a new job until it finishes the current one
- Requests are made in batches. This results from how HTML is structured as files that reference other files.
- Compared to the concurrent example (next), this page takes a long time to load with just static content

Concurrent Proxy

000			Carnegi	e Mellon University	CMU		R
(www.cmu.edu/index.shtml						☆ マ C Soogle	۹ 🍙
Carnegie Mellon University	RECTORY SEARCH NEWS	CALENDAR LIBRARIES	CAREERS	SUPPORT CMU GLOBAL SOCI Nation Status What do ce World War rubber hav	If I I I III IIIIIIIIIIIIIIIIIIIII		Carnegie Mellon Universi
Academics	CSS Script DOM Net	Cookies		These and	many other innovations	Ø	
thr Clear Persist All HTML CSS JS	XHR Images Flash M	ledia					
URL	Status	Domain	Size	Remote IP	Timeline		
GET index.shtml	200 OK	cmu.edu	14 KB	128.2.220.20:3092	11ms		
CET cmu.css	200 OK	cmu.edu	47.5 KB	128.2.220.20:3092	11ms		
GET cmu-new.css	200 OK	cmu.edu	10.5 KB	128.2.220.20.3092	18ms		
GET doio is	200 OK	cmu.edu	126.2 KB	128.2.220.20.3092	17ms		
GET scrints is	200 OK	cmu edu	6.8 KB	128 2 220 20:3092	20ms		
GET iquervis	200 OK	cmu edu	70.5 KB	128.2.220.20:3092	31ms		
GET homepage is	200 OK	cmu.edu	1.1 KB	128.2.220.20:3092	23ms		
► GET app ad is	200 OK	cmu.edu	2.6 KB	128.2.220.20:3092	28ms		
► GET ga.is	200 OK	google-analytics.com	15.3 KB	128.2.220.20:3092	Lonis	42	ms
GET CarnegieMellonUniversity wordma	200 OK	cmu.edu	48.7 KB	128.2.220.20:3092		25ms	
▶ GET btn go.gif	200 OK	cmu.edu	166 B	128.2.220.20:3092		8ms	
GET mellon_institute_748x460.jpg	200 OK	cmu.edu	93 KB	128.2.220.20:3092		21ms	
GET autism_research_month_240x126.j	200 OK	cmu.edu	29.8 KB	128.2.220.20:3092		27ms	
GET poets_on_tour_240x126.jpg	200 OK	cmu.edu	19.8 KB	128.2.220.20:3092			228ms
GET pres_jared_cohon_240x126.jpg	200 OK	cmu.edu	13 KB	128.2.220.20:3092			230ms
► GET global.gif	200 OK	cmu.edu	4.4 KB	128.2.220.20:3092		25ms	
► GET logo-ii.gif	200 OK	cmu.edu	5.3 KB	128.2.220.20:3092		27ms	
GET logo-cmtoday.gif	200 OK	cmu.edu	49 KB	128.2.220.20:3092		32m:	5
GET social-sprites.png	200 OK	cmu.edu	5.9 KB	128.2.220.20:3092		11m	5
GET mellon_institute_748x460.jpg	200 OK	cmu.edu	93 KB	128.2.220.20:3092		20	ims
GET transGray85.png	200 OK	cmu.edu	935 B	128.2.220.20:3092		13m	s
GET horizontal_rule.gif	200 OK	cmu.edu	3 KB	128.2.220.20:3092		18n	ns
▶ GETutm.gif?utmwv=5.4.1&ucmd%3I	200 OK	google-analytics.com	35 B	128.2.220.20:3092			31ms
24 requests			670.4 KB				524ms (onload: 545ms)

Concurrent Proxy

- Now, we see much less purple (waiting), and less time spent overall.
- Notice how multiple green (receiving) blocks overlap in time
 - Our proxy has multiple connections open to the browser to handle several tasks at once

Part B

- Getting content on the web: Telnet/cURL Demo
 - How the web really works
- Networking Basics
- Echo Client & Server Demo
- Proxy
 - Due Tuesday, December 12th
 - Grace days allowed
- String Manipulation in C

Sockets

- What is a socket?
 - To an application, a socket is a file descriptor that lets the application read/write from/to the network
 - (all Unix I/O devices, including networks, are modeled as files)
- Clients and servers communicate with each other by reading from and writing to socket descriptors



The main difference between regular file I/O and socket I/O is how the application "opens" the socket descriptors

Overview of the Sockets Interface



Host and Service Conversion: getaddrinfo

- getaddrinfo is the modern way to convert string representations of host, ports, and service names to socket address structures.
 - Replaces obsolete gethostbyname unsafe because it returns a pointer to a static variable
 - Advantages:
 - Reentrant (can be safely used by threaded programs).
 - Allows us to write portable protocol-independent code(IPv4 and IPv6)
 - Given host and service, getaddrinfo returns result that points to a linked list of addrinfo structs, each pointing to socket address struct, which contains arguments for sockets APIs.

getnameinfo is the inverse of getaddrinfo, converting a socket address to the corresponding host and service.

int socket(int domain, int type, int protocol);

- Create a file descriptor for network communication
- used by both clients and servers
- int sock_fd = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
- One socket can be used for two-way communication

int bind(int socket, const struct sockaddr *address, socklen_t address_len);

- Associate a socket with an IP address and port number
- used by servers
- struct sockaddr_in sockaddr family, address, port

- int listen(int socket, int backlog);
 - socket: socket to listen on
 - used by servers
 - backlog: maximum number of waiting connections
 - err = listen(sock_fd, MAX_WAITING_CONNECTIONS);
 - int accept(int socket, struct sockaddr *address, socklen_t *address_len);
 - used by servers
 - socket: socket to listen on
 - address: pointer to sockaddr struct to hold client information after accept returns
 - return: file descriptor

int connect(int socket, struct sockaddr *address, socklen_t address_len);

- attempt to connect to the specified IP address and port described in address
- used by clients

int close(int fd);

- used by both clients and servers
- (also used for file I/O)
- fd: socket fd to close

ssize_t read(int fd, void *buf, size_t nbyte);

- used by both clients and servers
- (also used for file I/O)
- fd: (socket) fd to read from
- buf: buffer to read into
- nbytes: buf length

ssize_t write(int fd, void *buf, size_t nbyte);

- used by both clients and servers
- (also used for file I/O)
- fd: (socket) fd to write to
- buf: buffer to write
- nbytes: buf length

Part B

- Getting content on the web: Telnet/cURL
 - How the web really works
- Networking Basics
- Echo Client & Server Demo
- Proxy
 - Due Tuesday, December 12th
 - Grace days allowed
- String Manipulation in C

Byte Ordering Reminder

- So, how are the bytes within a multi-byte word ordered in memory?
- Conventions
 - Big Endian: Sun, PPC Mac, Internet
 - Least significant byte has highest address
 - Little Endian: x86, ARM processors running Android, iOS, and Windows
 - Least significant byte has lowest address

Byte Ordering Reminder

- So, how are the bytes within a multi-byte word ordered in memory?
- Conventions

Big Endian: sun, PPC Mac, Internet

- Least significant byte has highest address
- Make sure to use correct endianness

Proxy - How

- Proxies are a bit special they are a server and a client at the same time.
- They take a request from one computer (acting as the server), and make it on their behalf (as the client).
- Ultimately, the control flow of your program will look like a server, but will have to act as a client to complete the request
 - Start small
 - Grab yourself a copy of the echo server (pg. 946) and client (pg. 947) in the book
 - Also review the tiny.c basic web server code to see how to deal with HTTP headers
 - Note that tiny.c ignores these; you may not

Proxy - How

What you end up with will resemble:



Proxy - Functionality

Should work on vast majority of sites

- Twitch, CNN, NY Times, etc.
- Some features of sites which require the POST operation (sending data to the website), will not work
 - Logging into websites, sending Facebook message
- HTTPS is not expected to work
 - Google, YouTube (and some other popular websites) now try to push users to HTTPs by default; watch out for that
- Cache previous requests
 - Use LRU eviction policy
 - Must allow for concurrent reads while maintaining consistency
 - Details in write up

Proxy - Functionality

- Why a multi-threaded cache?
 - Sequential cache would bottleneck parallel proxy
 - Multiple threads can read cached content safely
 - Search cache for the right data and return it
 - Two threads can read from the same cache block
 - But what about writing content?
 - Overwrite block while another thread reading?
 - Two threads writing to same cache block?

Summary

Step 1: Sequential Proxy

Works great for simple text pages with embedded styles

Step 2: Concurrent Proxy

multi-threading

Step 3 : Cache Web Objects

- Cache individual objects, not the whole page
- Use an LRU eviction policy
- Your caching system must allow for *concurrent reads* while maintaining consistency. Concurrency? Shared Resource?

Proxy – Testing & Grading

Autograder

- ./driver.sh will run the same tests as autolab:
 - Ability to pull basic web pages from a server
 - Handle a (concurrent) request while another request is still pending
 - Fetch a web page again from your cache after the server has been stopped
- This should help answer the question "is this what my proxy is supposed to do?"
- Please don't use this grader to definitively test your proxy; there are many things not tested here

Proxy – Testing & Grading

Test your proxy liberally

- The web is full of special cases that want to break your proxy (think small images, large images, videos, etc.)
- Generate a port for yourself with ./port-for-user.pl [andrewid]
- Generate more ports for web servers and such with ./free-port.sh

Create a handin file with make handin

 Will create a tar file for you with the contents of your proxylab-handin folder

Part B

- Getting content on the web: Telnet/cURL
 - How the web really works
- Networking Basics
- Echo Client & Server Demo
- Proxy
 - Due Tuesday, December 12th
 - Grace days allowed
- String Manipulation in C

String manipulation in C

sscanf: Read input in specific format

int sscanf(const char *str, const char *format, ...);

Example:

buf = "213 is awesome"

// Read integer and string separated by white space from buffer 'buf'
// into passed variables

ret = sscanf(buf, "%d %s %s", &course, str1, str2);

This results in:

course = 213, str1 = is, str2 = awesome, ret = 3

String manipulation (cont)

sprintf: Write input into buffer in specific format

int sprintf(char *str, const char *format, ...);
Example:
buf[100];
str = "212 is gwasama"

str = "213 is awesome"

// Build the string in double quotes ("") using the passed arguments

// and write to buffer 'buf'

sprintf(buf, "String (%s) is of length %d", str, strlen(str));

This results in:

buf = String (213 is awesome) is of length 14

String manipulation (cont)

Other useful string manipulation functions:

- strcmp, strncmp, strncasecmp
- strstr
- strlen
- strcpy, strncpy

String Manipulation (cont)

- Beware: String operations will NOT work properly with binary data
 - E.g. images, videos, etc
 - Think about the null terminator string operations check for
 - Remember this when caching data objects
- Solution: use memcpy instead
 - void *memcpy(void *dest, const void *src, size_t n);

Aside: Setting up Firefox to use a proxy

Configure Proxies to Access the Internet							
Cc	O No proxy						
Co Auto-detect proxy settings for this network							
Ca	 Use system pro Manual proxy of 	oxy settings configuration:					
	HTTP Proxy:	catshark.ics.cs.cmu.edu	Port:	3092 🗘			
		Use this proxy server for	all protoc	ols	5		
	SSL Proxy:	catshark.ics.cs.cmu.edu	Port:	3092 🔹			
	FTP Proxy:	catshark.ics.cs.cmu.edu	Port:	3092 🔹			
01	SOCKS Host:	catshark.ics.cs.cmu.edu	Port:	3092 (‡			
Y		◯ SOCKS v4 ④ SOCKS v5			D		
•	No Proxy for:)		
Т	localhost, 127	7.0.0.1					
	Example: .mo:	zilla.org, .net.nz, 192.168.1.	0/24				
	O Automatic prov	xy configuration URL:					
				Reload	J		

- You may use any browser, but we'll be grading with Firefox
- Preferences > Advanced > Network > Settings... (under Connection)
- Check "Use this proxy for all protocols" or your proxy will appear to work for HTTPS traffic.

Questions?