Hacking Apache HTTP Server at Yahoo!

Michael J. Radwin

http://public.yahoo.com/~radwin/

ApacheCon 2005
Wednesday, 14 December 2005

Since 1996, Yahoo has been running Apache HTTP Server on thousands of servers and serving billions of requests a day. This session reveals the secrets of how Yahoo gets maximum performance out of minimal hardware by tweaking configuration directives and hacking the source code. Radwin will cover topics such as reducing bandwidth costs, extensible logfile format and rotation schemes, dumping core gracefully, and how to avoid the dreaded MaxClients, Max/MinSpareServers, StartServers configuration nightmare.

The Internet's most trafficked site

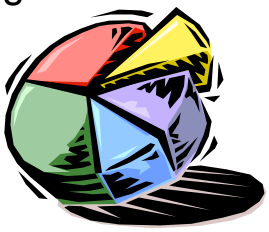
YAHOO!

25 countries, 13 languages

YAHOO!

Yahoo! by the Numbers

- 411M unique visitors per month
- 191M active registered users
- 11.4M fee-paying customers
- 3.4B average daily pageviews



October 2005

YAHOO!

Numbers from Q3 2005 Yahoo! Earnings

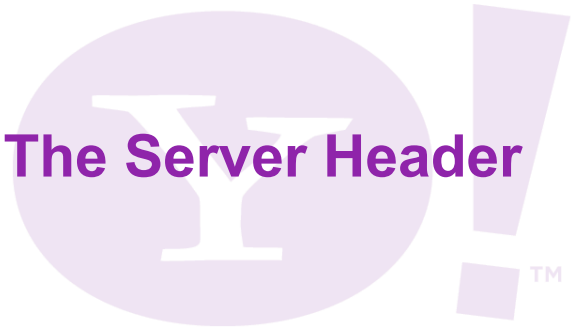
October 18, 2005

This talk is about yapache

- Yahoo's modified version of Apache
- Pronounced *why·apache*
- Based on Apache/1.3
  - Port to Apache/2.2 in 2006

YAHOO!

The Server Header



The HTTP “Server” header

```
HTTP/1.1 200 OK
Date: Thu, 08 Dec 2005 17:49:59 GMT
Server: Apache/1.3.33 (Unix) DAV/1.0.3 PHP/4.3.10
  mod_ssl/2.8.22 OpenSSL/0.9.7e
Last-Modified: Mon, 14 Nov 2005 21:07:07 GMT
ETag: "12c7ace-1475-4378fc7b"
Content-Length: 5237
Connection: close
Content-Type: text/html

<html> ...
```


YAHOO!

Suppressing the Server header

```
HTTP/1.1 200 OK
Date: Thu, 08 Dec 2005 17:52:37 GMT
Cache-Control: private
Connection: close
Content-Type: text/html; charset=ISO-8859-1
Set-Cookie: B=fvsru911pgsn5&b=2; expires=Thu, 15
  Apr 2010 20:00:00 GMT; path=/; domain=.yahoo.com

<html> ...
```


YAHOO!

Why does Y! suppress “Server”?

- 3 reasons

YAHOO!

Reason 1

- Security through obscurity

YAHOO!

Reason 2

- Bandwidth conservation

11

YAHOO!

That's 80 bytes of content that no user sees, and few User-Agent care about.

Apparently Windows Media Player actually does care about it, but most browsers (MSIE, Firefox, Sarafi, Opera, etc) do not.

Reason 3 (the real reason)

- “Netscape Guide by Yahoo”

YAHOO!

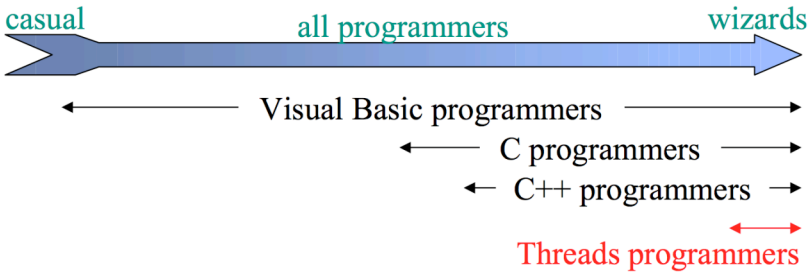
Apache 1.3

Yes, we're still using Apache 1.3

- It has most of the features we need
  - We added gzip support in June 1998
- It performs really well
- It's very stable
- We understand the codebase
- We don't need no stinkin' threads anyways

YAHOO!

What's Wrong With Threads?



- Too hard for most programmers to use
- Even for experts, development is painful

Source: John Ousterhout, *Why Threads Are a Bad Idea (for most purposes)*, September 28, 1995, slide 5

YAHOO!

The prefork MPM R00LZ!!!1!1!

- We prefer processes over threads
- Better fault isolation
  - When one child crashes, only a single user gets disconnected
- Better programming model for C/C++
  - Private data by default
  - Shared data requires extra work (mmap + synchronization)

16

YAHOO!

When we do migrate to Apache 2 (likely 2006) we will only use the prefork MPM.

17

Common Log Format

- a.k.a. Combined Log Format

```
69.64.229.166 - - [08/Dec/2005:14:00:06 -0800]
  "GET /nba/rss.xml HTTP/1.1" 200 9295 "-"
  "Mozilla/5.0 (Macintosh; U; PPC Mac OS X Mach-O;
  en-US; rv:1.7.10) Gecko/20050716 Firefox/1.0.6"
66.60.182.2 - - [08/Dec/2005:14:00:06 -0800] "GET
  /ncaaf/news?slug=ap-congress-
  bcs&prov=ap&type=lgns HTTP/1.0" 200 44148
  "http://sports.yahoo.com/ncaaf" "Mozilla/4.0
  (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET
  CLR 1.0.3705; .NET CLR 1.1.4322)"
```


YAHOO!

Problems with Common Log Format

- No standard place to put extra info
  - Cookies
  - Advertisement IDs
  - Request duration
- Time spent on formatting
  - Escaping unsafe chars (\")
  - Format timestamps to human-readable
    - Eventually get converted back to time_t

YAHOO!

Problems with CLF (cont'd)

- Wasted bytes
  - 200 status code field is common
    - Could be skipped
  - HTTP protocol version in %r
    - Do we really care if it's 1.0 vs. 1.1?

YAHOO!

yapache Access Log

1. IP address
2. Request end time (time_t + ms)
3. Request duration (µs)
4. Bytes sent
5. URI + HTTP Host
6. HTTP method (+ Content-Length if POST/PUT)
7. Response status (only if not 200 OK)
8. Cookies
9. User-Agent
10. Referer
11. Advertisement IDs
12. User-defined values from `notes`, `subprocess_env`, `headers_{in,out}`

YAHOO!

Access Log Format

- One request per line
- First 32 bytes numeric values in hex, followed by URI, followed ^E-delimited named fields
- First byte following ^E describes field

```
46b9b466438b6fd30000a91c00001d5a/nfl/news^E
  gMozilla/4.0 (compatible; MSIE 6.0; Windo
  ws NT 5.1)^EmGET^Ewsports.yahoo.com^Erhtt
  p://sports.yahoo.com/nfl^EcB=ar0qr8t1ohcn
  i&b=3&s=hp; Y=...
```


YAHOO!

Signal-free Log Rotation

- Look ma, no signals!
  - No pipes, either
- Rotate logfiles by renaming them
  - `stat()` logfile every 60 seconds
  - If inode changed, close and reopen
  - During 60-second interval, child procs may write to either logfile
- Log directory must be writable by `User`

YAHOO!

Bandwidth Reduction

Smaller 30x response bodies

```
GET /astrology HTTP/1.1
Host: astrology.yahoo.com
User-Agent: Mozilla/5.0 (compatible; example)

HTTP/1.1 301 Moved Permanently
Date: Sun, 27 Nov 2005 21:10:22 GMT
Location: http://astrology.yahoo.com/astrology/
Connection: close
Content-Type: text/html

The document has moved <A
   HREF="http://astrology.yahoo.com/astrology/">here</A>.
   <P>
```


YAHOO!

In fact, we could probably get away with skipping the response body completely since the Location header is the only part that actually matters. Only really broken (HTTP/0.9) User-Agents are going to display the HTML content anyways.

Apache/1.3 on-the-fly gzip

- Similar in spirit to mod_deflate
- Prerequisites
  - HTTP/1.1
  - Accept-Encoding: gzip
  - IE 6+ or Mozilla 5+
- Disabled when CPU < 10% idle

YAHOO!

Default gzip level 6.

Default memory level 8.

Not for the faint of heart

```
BUFF *outbuf = fb->cmp_outbuf;
fb->z.next_in = fb->outbase + fb->cmp_start_here;
fb->z.avail_in = fb->outcnt - fb->cmp_start_here;
fb->z.next_out = outbuf->outbase + outbuf->outcnt;
uInt len = fb->z.avail_out =
   outbuf->bufsiz - outbuf->outcnt;
int err = deflate(&(fb->z), Z_SYNC_FLUSH);
fb->crc = crc32(fb->crc, fb->outbase+fb->cmp_start_here,
                fb->outcnt - fb->cmp_start_here -
                fb->z.avail_in);
len = len - fb->z.avail_out;
outbuf->outcnt += len;
fb->cmp_start_here = 0;
```

27

YAHOO!

Implementing the equivalent of mod_deflate without Apache2's Filtered I/O framework meant touching a bunch of code in the core of httpd. This extract is just part of the patch. It gets worse. We had to modify the following:

- buff.c
  - ap_bwrite(), bflush_core(), ap_bclose()
  - Introuduced new constants B_GZIP, B_GZIP_CHUNK
- http_protocol.c
  - ap_send_http_header(), ap_finalize_request_protocol()

How Many Servers?

How Many Servers?

- StartServers
- MaxSpareServers
- MinSpareServers
- MaxClients

YAHOO!

There Can Be Only One

- MaxClients

YAHOO!

Constant Pool Size is Good

- Predictable performance under spiky load
  - Start all MaxClients servers at once
  - Put host into load-balancer rotation
  - Never kill off idle servers
  - Any servers killed by MaxRequestsPerChild still get replaced
- For 99% of sites, MaxClients is sufficient
  - Therefore, we disable Min/Max/StartServers

YAHOO!

If you know you can comfortably deal with 80 processes, then why let it drop to 5?

Constant Pool Implementation

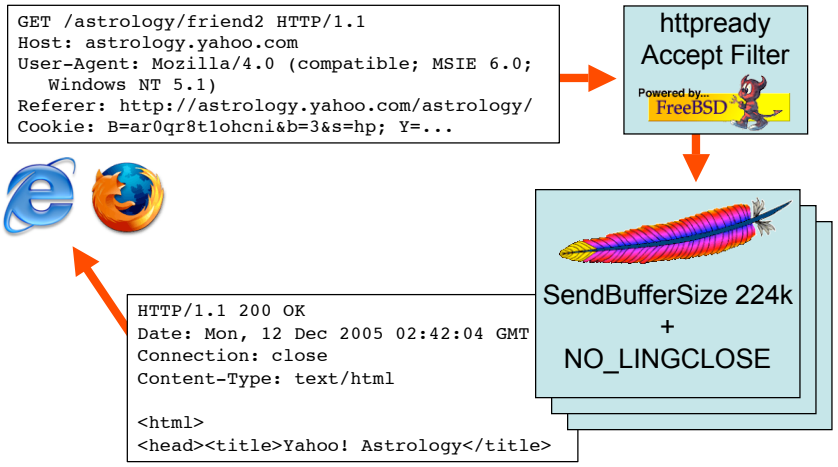
- `HARD_SERVER_LIMIT = 2048;`
- `ap_daemons_limit = ap_daemons_max_free = ap_daemons_min_free = ap_daemons_to_start =` *`MaxClients`*`;`
- MaxClients usually < 100

YAHOO!

Waiting for the Client Sucks

33

Let the kernel do the buffering



YAHOO!

Accept Filtering on FreeBSD

- SO_ACCEPTFILTER with “httpready”
  - Apache won’t wake up from `accept()` until a full HTTP GET request has been buffered by kernel
  - Entire request present in first `read()`
- Apache child processes able to do useful work immediately
  - More efficient use of server pool

YAHOO!

Accept Filtering on FreeBSD:
http://www.freebsd.org/cgi/man.cgi?query=accf_http&sektion=9

SO_ACCEPTFILTER is not available on Linux. There is a socket option called TCP_DEFER_ACCEPT, which is roughly equivalent to the “dataready” accept filter on FreeBSD. It’s not quite as good as “httpready”, since with TCP_DEFER_ACCEPT, accept() will return as soon as the socket becomes readable (i.e. after at least one byte of the request is received).

http://builder.com.com/5100-6372-1050771.html

SendBufferSize

- SendBufferSize 229376
  - To go higher, adjust kernel tunable `kern.ipc.maxsockbuf` (FreeBSD) or `net.core.wmem_{default,max}` (Linux)
  - Set to max response size (HTML + headers)
- Tradeoff
  - Avoids blocking on `write()` to socket
  - More kernel memory consumed

YAHOO!

229376 is 224k. That's 256k - 32k. It's the largest default value you can use without increasing the kernel tunables.

Luckily, that's bigger than your typical HTML page.

NO_LINGCLOSE

- Don't wait for the client to read the response
  - Write full response into the socket buffer
  - Close the socket
- Apache child returns to pool
  - Kernel worries about completing data transfer to client
- No idea if client read whole response
  - If client bails out halfway through or goes away, Apache logs won't show it

YAHOO!

Hostname hacks

YahooHostHtmlComment

- Comment at end of HTML pages

```
<!-- p22.sports.scd.yahoo.com
compressed/chunked Sun Nov 27
15:59:14 PST 2005 -->
```

- For debugging page or cache problems
  - Users save HTML, send to Customer Care
  - Engineers examine error log on server

YAHOO!

ap_finalize_request_protocol() patch

```
if (!r->next && !r->header_only && !r->proxyreq &&
    yahoo_footer_check_content_type(r) &&
    !ap_table_get(r->headers_out, "Content-Length") &&
    !ap_table_get(r->headers_out, "Content-Range"))
{
  ap_hard_timeout("send pre-finalize body", r);
  ap_rvputs(r, "<!-- ", yahoo_gethostname(), " ",
            yahoo_footer_compression_type(r), " ",
            ap_gm_timestr_822(r->pool, r->request_time),
            " -->\n", NULL);
  ap_kill_timeout(r);
}
```

40

YAHOO!

```
static const char * yahoo_footer_compression_type(request_rec
*r)
{
  int flags = r->connection->client->flags;

  if (flags & B_GZIP_CHUNK)
    return "compressed/chunked";
  else if (flags & B_GZIP)
    return "compressed";
  else if (flags & B_CHUNK)
    return "uncompressed/chunked";
  else
    return "uncompressed";
}
```

```
static int yahoo_footer_check_content_type(request_rec *r)
{
  const char *ctype = ap_table_get(r->headers_out, "Content-
Type");

  if (ctype != NULL &&
      (strncasecmp(ctype, "text/html", 9) == 0 ||
       strncasecmp(ctype, "text/xml", 8) == 0 ||
       strncasecmp(ctype, "application/xml", 15) == 0))
```

http://foo.yahoo.com/bin/hostname

```
static int yahoo_hostname_handler(request_rec *r) {
    char host[MAXHOSTNAMELEN] = "unknown";
    if (r->method_number != M_GET)
        return HTTP_NOT_IMPLEMENTED;
    r->content_type = "text/plain";
    ap_send_http_header(r);
    if (r->header_only)
        return OK;
    (void) gethostname(host, sizeof(host) - 1);
    ap_rvputs(r, host, "\n", NULL);
    return OK;
}
```


YAHOO!

42

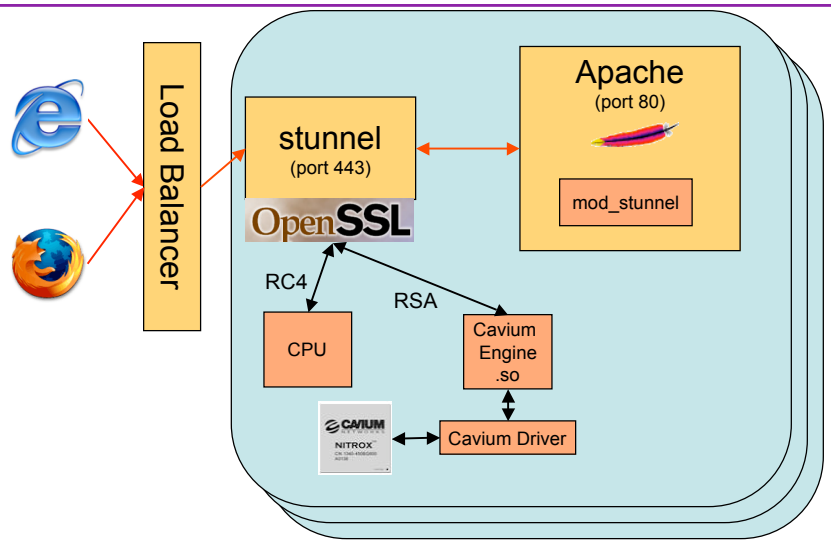
SSL Acceleration

- Cavium Nitrox CN1120
- 14k RSA ops/s
- OpenSSL 0.9.7 engine API

- With card, can handle about as much SSL traffic as a port 80 server w/o card

YAHOO!

SSL Architecture



YAHOO!

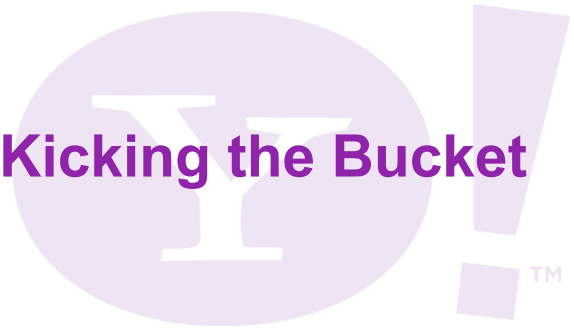
mod_stunnel: Apache+stunnel glue

- Overrides `getpeername()`
  - Returns IP address of actual client
- Emulates mod_ssl environment

```
int mod_stunnel_post_read_request (request_rec *r) {
  if (ntohs(r->connection->local_addr.sin_port) == 443) {
    ap_ctx_set(r->ctx, "ap::http::method", "https");
    ap_ctx_set(r->ctx, "ap::default::port", "443");
    ap_table_set(r->subprocess_env, "HTTPS", "on");
  }
  return DECLINED;
}
```


YAHOO!

Kicking the Bucket



Avoid mod_whatkilledus.c

- Trashed stacks frequently cause SEGV or BUS
- Fatal signal handlers can get into an infinite coredump loop
- Our set_signals() never uses sig_coredump()
  - Let child core quickly and in-context

YAHOO!

Corefiles w/o CoreDumpDirectory

- FreeBSD

```
sysctl -w kern.coredump=1 \
  kern.sugid_coredump=1 \
  kern.corefile="/var/crash/%N.core.%U"
```

- Linux

```
sysctl -q -w kernel.core_pattern=\
  "/var/crash/%e.core.%u" \
  kernel.suid_dumpable=1 \
  kernel.core_uses_pid=0
```

48

YAHOO!

Since we disable fatal signal handling, we render the CoreDumpDirectory directive useless. This slide describes how to get corefiles without Apache explicitly chdir()ing into the directory. We run these as part of our /usr/local/etc/rc.d

If you want one corefile per pid:

FreeBSD: sysctl -w kern.corefile="$ROOT/var/crash/%N.core.%U.%P

Linux: sysctl -q -w kernel.core_uses_pid=0

Don’t multi-signal in reclaim_child_processes()

- Parent process sends SIGHUP
  - Waits 0.3s, sends another SIGHUP
  - Waits 1.4s, sends SIGTERM
  - Waits 6.0s, sends SIGKILL
- yapache skips second HUP and TERM

YAHOO!

Misc

The Include directive

- Our httpd.conf ends with

```
Include conf/include/*.conf
```

- Wildcard safer than entire directory
  - Avoid Emacs `abc.conf~` backup files
- Yahoo sites install their own `$SR/conf/include/foobar.conf`
  - Override settings such as `ServerAdmin` or `MaxClients`

YAHOO!

setproctitle() in child_main()

```
while ((r = ap_read_request(current_conn))
       != NULL) {
#if defined(YAHOO) && defined(__FreeBSD__)
    setproctitle("%s %s",
                 r->remote_ip,
                 r->unparsed_uri);
#endif
    /* ... */
}
```


YAHOO!

ysar - inspired by System V `sar(1)`

Time	Yapache req/s	rt msec	cpu %util	mem %util	sysc /pkt	bge0 outkbps
11/28-08:30	105.6	29.0	47.7	66.7	4.5	11048.4
11/28-09:00	117.3	32.7	53.1	70.6	4.6	11412.9
11/28-09:30	122.6	30.2	52.6	71.8	4.5	11905.8
11/28-10:00	120.4	32.3	52.2	74.8	4.7	11360.0
11/28-10:30	115.7	29.0	50.2	73.9	4.5	11739.2
11/28-11:00	114.8	31.8	52.3	76.0	4.7	11371.4
Min	55.1	17.2	26.9	64.4	4.3	5938.9
Mean	86.3	26.8	40.6	70.0	4.9	8947.6
Max	122.6	34.7	53.7	76.0	5.5	11905.8

YAHOO!

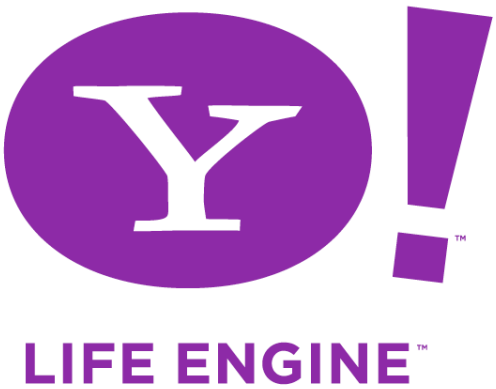
Summary

Take-aways

- Every byte counts
- Every CPU cycle counts
- Use the right tool for the job
  - Apache: dynamic content generation
  - OS: buffering content in & out
  - Dedicated chips: crypto
- When it's time to die
  - Fail fast and in context
  - Use multi-process for fault isolation

YAHOO!

Slides: http://public.yahoo.com/~radwin/



LIFE ENGINE™