mod_perl For Speed Freaks!

Philippe M. Chiasson
gozer@ectoplasm.org
This is NOT what we will be talking about
This is what we will be talking about
This is what we will be talking about

Requests per second: 1159.49 [#/sec] (mean)
Time per request: 17.249 [ms] (mean)
Time per request: 0.862 [ms] (mean, across all concurrent requests)
Transfer rate: 5681.49 [Kbytes/sec] received

Connection Times (ms)

<table>
<thead>
<tr>
<th></th>
<th>min</th>
<th>mean[ +/-sd]</th>
<th>median</th>
<th>max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connect:</td>
<td>1</td>
<td>2</td>
<td>1.6</td>
<td>3</td>
</tr>
<tr>
<td>Processing:</td>
<td>7</td>
<td>12</td>
<td>2.5</td>
<td>13</td>
</tr>
<tr>
<td>Waiting:</td>
<td>3</td>
<td>7</td>
<td>2.6</td>
<td>7</td>
</tr>
<tr>
<td>Total:</td>
<td>11</td>
<td>15</td>
<td>2.7</td>
<td>15</td>
</tr>
</tbody>
</table>
First, assumptions

- You have Apache/mod_perl running
- Your code is now much faster
- You think you could need a speedup
Information gathering

- Apples vs. Apples
- Reliable information
- metrics
mod_status

• Server status reporting
• Useful to get an overview of your server’s health
• Comes with Apache, so you already have it
Enabling mod_status

LoadModule status_module modules/mod_status.so
ExtendedStatus On

<Location /server-status>
  SetHandler server-status
</Location>
mod_status
mod_status explained

Restart Time: Wednesday, 05-Oct-2005 09:28:50 PDT
Parent Server Generation: 19
Server uptime: 19 days 10 hours 7 minutes 47 seconds
Total accesses: 48321153 - Total Traffic: 1391.0 GB
CPU Usage: u651.234 s827.719 cu1619.3 cs0 - .185% CPU load
28.8 requests/sec - 0.8 MB/second - 30.2 kB/request
154 requests currently being processed, 189 idle workers
mod_status explained

Scoreboard Key:
"_" Waiting for Connection, "S" Starting up, "R" Reading Request, "W" Sending Reply, "K" Keepalive (read), "D" DNS Lookup, "C" Closing connection, "L" Logging, "G" Gracefully finishing, "I" Idle cleanup of worker, "." Open slot with no current process
mod_status explained

```bash
```

- **Total Accesses**: 48340863
- **Total kBytes**: 1459565806
- **CPULoad**: .18793
- **Uptime**: 1678505
- **ReqPerSec**: 28.8
- **BytesPerSec**: 890432
- **BytesPerReq**: 30917.8
- **BusyWorkers**: 140
- **IdleWorkers**: 203
Apache2::Status

* Perl status reporting
* Great to poke at a running mod_perl server
* Comes with mod_perl, so you already have it
Enabling Apache2::Status

```
<Location /perl-status>
    SetHandler perl-script
    PerlHandler Apache2::Status
    PerlSetVar StatusOptionsAll On
</Location>
```

```
$> cpan \\
Data::Dumper \\
Devel::Peek \\
Apache::Peek \\
B::LexInfo \\
B::Deparse \\
B::Terse \\
B::TerseSize \\
Devel::Symdump \\
B::Fathom \\
B::Graph
```
Apache2::Status

Embedded Perl version v5.8.7 for Apache/2.0.52 process 27065, running since Mon Oct 24 19:32:30 2005

Environment
Loaded Modules
Inheritance Tree
ISA Tree
Perl Configuration
Compiled Registry Scripts
PerlRequire'd Files
Signal Handlers
Symbol Table Dump
# Apache2::Status

![Apache2::Status screenshot](image)

<table>
<thead>
<tr>
<th>Module</th>
<th>Version</th>
<th>Last Update</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>CGI</td>
<td>3.11</td>
<td>Wed Aug 3 14:14:55 2005</td>
<td>/System/Library/Perl/5.8.6/CGI.pm</td>
</tr>
<tr>
<td>CGI::Util</td>
<td>1.5</td>
<td>Mon Mar 7 11:52:11 2005</td>
<td>/System/Library/Perl/5.8.6/CGI/Util.pm</td>
</tr>
<tr>
<td>Carp</td>
<td>1.03</td>
<td>Sun Mar 20 16:41:55 2005</td>
<td>/System/Library/Perl/5.8.6/Carp.pm</td>
</tr>
<tr>
<td>Config</td>
<td>0.00</td>
<td>Sun Mar 20 16:42:38 2005</td>
<td>/System/Library/Perl/5.8.6/darwin-thread-multi-2level</td>
</tr>
</tbody>
</table>
Apache2::Status

Memory Usage
Apache2::Status

Memory Usage for package CGI
Totals: 255794 bytes, 249.80 Kb, 0.25 Mb | 3328 OPs

Memory Usage for package DBI
Totals: 260655 bytes, 254.55 Kb, 0.25 Mb | 4499 OPs

Memory Usage for package POSIX
Totals: 113667 bytes, 111.00 Kb, 0.11 Mb | 455 OPs

AUTOLOAD 6611 bytes | 131 OPs
import 3646 bytes | 59 OPs

But exactly how fast are you?

- Comparison points
- Benchmark early
- Benchmark often
ab

- Apache Bench
- A good HTTP benchmark tool
- Comes with Apache, so you already have it
- There are others
$> ab -h
Usage: ab [options] [http://]hostname[:port]/path
Options are:
   -n requests     Number of requests to perform
   -c concurrency  Number of multiple requests to make
   -t timelimit    Seconds to max. wait for responses
   -V              Print version number and exit
   -k              Use HTTP KeepAlive feature
   -e filename     Output CSV file with percentages served
   -h              Display usage information (this message)
ab output

$> ab -c10 -n50 http://www.google.com/

This is ApacheBench, Version 2.0.41-dev <$Revision: 1.121.2.12 $> apache-2.0
Copyright (c) 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeus

Benchmarking www.google.com (be patient).....done

Server Software: GWS/2.1
Server Hostname: www.google.com
Server Port: 80
Document Path: /
Document Length: 151 bytes
ab output

$> ab -c10 -n50 http://www.google.com/

Concurrency Level: 10
Time taken for tests: 0.313241 seconds
Complete requests: 50
Failed requests: 0
Write errors: 0
Non-2xx responses: 50
Total transferred: 24100 bytes
HTML transferred: 7550 bytes
Requests per second: 159.62 [#/sec] (mean)
Time per request: 62.648 [ms] (mean)
Time per request: 6.265 [ms] (mean, across all concurrent requests)
Transfer rate: 73.43 [Kbytes/sec] received
### ab output

$> ab -c10 -n50 http://www.google.com/

<table>
<thead>
<tr>
<th></th>
<th>min</th>
<th>mean[+/-sd]</th>
<th>median</th>
<th>max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connect:</td>
<td>23</td>
<td>24</td>
<td>1.9</td>
<td>24</td>
</tr>
<tr>
<td>Processing:</td>
<td>27</td>
<td>32</td>
<td>9.1</td>
<td>29</td>
</tr>
<tr>
<td>Waiting:</td>
<td>27</td>
<td>32</td>
<td>9.0</td>
<td>29</td>
</tr>
<tr>
<td>Total:</td>
<td>51</td>
<td>57</td>
<td>9.0</td>
<td>56</td>
</tr>
</tbody>
</table>

Percentage of the requests served within a certain time (ms)

- 50%: 56
- 66%: 57
- 75%: 57
- 80%: 58
- 90%: 74
- 95%: 79
- 98%: 91
- 99%: 91
- 100%: 91 (longest request)
Memory usage

- Total physical RAM is the limit
- SWAP doesn’t count
- Wired counts
- Shared, Resident, Total
Processes:  81 total, 2 running, 79 sleeping...  237 threads
Load Avg:  1.48, 1.31, 1.23      CPU usage:  90.9% user, 9.1% sys, 0.0% idle
SharedLibs: num = 30, resident = 7.40M code, 904K data, 2.71M LinkEdit
MemRegions: num = 14341, resident = 192M + 12.3M private, 109M shared
PhysMem: 77.3M wired, 284M active, 142M inactive, 504M used, 7.84M free
VM: 6.45G + 21.0M 1417790(0) pageins, 375968(0) pageouts

<table>
<thead>
<tr>
<th>PID</th>
<th>COMMAND</th>
<th>%CPU</th>
<th>TIME</th>
<th>#TH</th>
<th>#PRTS</th>
<th>#MREGS</th>
<th>RPRVT</th>
<th>RSHRD</th>
<th>RSIZE</th>
<th>VSIZE</th>
</tr>
</thead>
<tbody>
<tr>
<td>27068</td>
<td>Grab</td>
<td>0.0%</td>
<td>0:06.77</td>
<td>3</td>
<td>160</td>
<td>195</td>
<td>2.09M</td>
<td>6.77M</td>
<td>3.01M</td>
<td>153M</td>
</tr>
<tr>
<td>26955</td>
<td>TextWrangl</td>
<td>0.0%</td>
<td>1:14.01</td>
<td>4</td>
<td>110</td>
<td>260</td>
<td>4.77M</td>
<td>18.4M</td>
<td>15.6M</td>
<td>174M</td>
</tr>
<tr>
<td>26905</td>
<td>bash</td>
<td>0.0%</td>
<td>0:00.66</td>
<td>1</td>
<td>14</td>
<td>18</td>
<td>220K</td>
<td>668K</td>
<td>768K</td>
<td>27.1M</td>
</tr>
<tr>
<td>26904</td>
<td>login</td>
<td>0.0%</td>
<td>0:00.04</td>
<td>1</td>
<td>16</td>
<td>37</td>
<td>0K</td>
<td>320K</td>
<td>188K</td>
<td>26.9M</td>
</tr>
<tr>
<td>26890</td>
<td>firefox-bi</td>
<td>0.0%</td>
<td>9:31.50</td>
<td>12</td>
<td>213</td>
<td>514</td>
<td>38.3M</td>
<td>19.9M</td>
<td>48.3M</td>
<td>398M</td>
</tr>
<tr>
<td>26887</td>
<td>Keynote</td>
<td>0.0%</td>
<td>24:04.87</td>
<td>5</td>
<td>124</td>
<td>598</td>
<td>52.7M</td>
<td>24.4M</td>
<td>58.6M</td>
<td>284M</td>
</tr>
<tr>
<td>26882</td>
<td>thunderbir</td>
<td>0.0%</td>
<td>2:08.07</td>
<td>10</td>
<td>124</td>
<td>582</td>
<td>36.3M</td>
<td>24.4M</td>
<td>43.0M</td>
<td>230M</td>
</tr>
<tr>
<td>26792</td>
<td>bash</td>
<td>0.0%</td>
<td>0:00.24</td>
<td>1</td>
<td>14</td>
<td>17</td>
<td>196K</td>
<td>668K</td>
<td>548K</td>
<td>27.1M</td>
</tr>
<tr>
<td>26791</td>
<td>login</td>
<td>0.0%</td>
<td>0:00.05</td>
<td>1</td>
<td>16</td>
<td>37</td>
<td>0K</td>
<td>320K</td>
<td>16K</td>
<td>26.9M</td>
</tr>
<tr>
<td>26789</td>
<td>iTerm</td>
<td>0.0%</td>
<td>10:00.31</td>
<td>6</td>
<td>380</td>
<td>204</td>
<td>5.31M</td>
<td>10.3M</td>
<td>9.25M</td>
<td>159M</td>
</tr>
</tbody>
</table>
GTop

use GTop;
my $gtop = GTop->new;

my $proc_mem = $gtop->proc_mem($$);
for (qw(size vsize share rss)) {
    printf "  %s => %d
", _, $proc_mem->$_();
}

size => 1900544
vsize => 3108864
share => 1392640
rss   => 1900544
use BSD::Resource;
($usertime,
 $systemtime,
 $maxrss,
 $ixrss,
 $idrss,
 $isrss,
 $minflt,
 $majflt,
 $nswap,
 $inblock,
 $oublock,
 $msgsnd,
 $msgrcv,
 $nsignals,
 $nvcswh,  
$nivcswh) = getrusage();
Shared memory

• Identical memory shared by more than one process
• You want to get as much as possible
• Copy on write
Module Preloading

PerlModule CGI
PerlModule DBI
PerlModule DBD::mysql
PerlModule Apache2::RequestRec
PerlModule Apache2::ServerRec
Module initialization

• DBI->install_driver('mysql');
• CGI->compile(':all');
• Check the doc of the modules you use
#startup.pl
use ModPerl::RegistryLoader ();

my $rl = ModPerl::RegistryLoader->new(
    package => 'ModPerl::Registry',
    debug   => 1,
);

$rl->handler($url, $filename);
use Foo vs. use Foo ();

- use Foo: Importing default EXPORTS
- use Foo (): Importing nothing by default

use POSIX;

use POSIX added 696k

use POSIX ();

use POSIX () added 316k
use Apache2::Const qw(OK DECLINED);
return OK;

use Apache2::Const -compile => qw(OK DECLINED);
return Apache2::Const::OK;
Apache configuration
perl-script vs. modperl

- SetHandler perl-script
  - STDIN/STDOUT tied
  - %ENV, @INC saved/restored
  - %ENV changes propagated
- SetHandler modperl
  - nada
PerlOptions

• PerlOptions Autoload
• PerlOptions GlobalRequest
• PerlOption ParseHeader
• PerlOption SetupEnv
KeepAlive Off

- KeepAlives are an HTTP feature
- Generally a good idea
- mod_perl not so much so
Memory leaks

- OS memory usually not reclaimed
- Perl designed for run fast, then terminate
- mod_perl is a long running process
- Some Perl optimizations don’t apply
- all sort of thing can (and will) leak
sub handler {
    my $r = shift;
    open (my $fh, "'/tmp/motd.txt'" );
    my $motd = join ' ', <$fh> ;
    print $motd;
    return OK;
}

sub handler {
    my $r = shift;
    open (my $fh, "'/tmp/motd.txt'" );
    while (my $line = <$motd> ) {
        print $line;
    }
    return OK;
}
Reffing

sub handler {
    my $r = shift;
    open (my $fh, "~/tmp/motd.txt");
    my $motd = join '',<$fh>;
    print_motd($motd);
    return OK;
}

sub print_motd {
    my $motd = shift;
    print $motd;
}

sub handler {
    my $r = shift;
    open (my $fh, "~/tmp/motd.txt");
    my $motd = join '',<$fh>;
    print_motd($motd);
    return OK;
}

sub print_motd {
    my $motd = shift;
    print $$motd;
}
Apache2::SizeLimit

#startup.pl
use Apache2::SizeLimit;
$Apache2::SizeLimit::MAX_PROCESS_SIZE = 12000;
$Apache2::SizeLimit::MIN_SHARE_SIZE = 6000;
$Apache2::SizeLimit::MAX_UNSHARED_SIZE = 5000;
$Apache2::SizeLimit::CHECK_EVERY_N_REQUESTS = 4;

#httpd.conf
PerlCleanupHandler Apache2::SizeLimit
Profiling

- Finding bottlenecks in Perl code
- Don’t guess
- Profilers are out there
- use them
PerlModule Apache::DProf

$> apachectl restart
$> cd /var/httpd/logs/prof/12345
$> dprofpp
Total Elapsed Time = 6.238159 Seconds
  User+System Time = 1.188159 Seconds

<table>
<thead>
<tr>
<th>%Time</th>
<th>ExclSec</th>
<th>CumulS</th>
<th>#Calls</th>
<th>sec/call</th>
<th>Csec/c</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.8</td>
<td>0.129</td>
<td>0.463</td>
<td>19</td>
<td>0.0068</td>
<td>0.0244</td>
<td>Mail::SpamAssassin::BEGIN</td>
</tr>
<tr>
<td>9.17</td>
<td>0.109</td>
<td>0.207</td>
<td>71</td>
<td>0.0015</td>
<td>0.0029</td>
<td>Mail::SpamAssassin::PerMsgStatus::BEGIN</td>
</tr>
<tr>
<td>6.73</td>
<td>0.080</td>
<td>0.138</td>
<td>4</td>
<td>0.0200</td>
<td>0.0345</td>
<td>Mail::Transport::IMAP4::BEGIN</td>
</tr>
<tr>
<td>6.65</td>
<td>0.079</td>
<td>1.121</td>
<td>24</td>
<td>0.0033</td>
<td>0.0467</td>
<td>main::BEGIN</td>
</tr>
<tr>
<td>3.96</td>
<td>0.047</td>
<td>0.079</td>
<td>3</td>
<td>0.0158</td>
<td>0.0265</td>
<td>Mail::IMAPClient::_read_line</td>
</tr>
<tr>
<td>3.37</td>
<td>0.040</td>
<td>0.049</td>
<td>8</td>
<td>0.0050</td>
<td>0.0062</td>
<td>Mail::SpamAssassin::Conf::BEGIN</td>
</tr>
<tr>
<td>3.37</td>
<td>0.040</td>
<td>0.059</td>
<td>17</td>
<td>0.0023</td>
<td>0.0035</td>
<td>Net::DNS::Resolver::Base::BEGIN</td>
</tr>
<tr>
<td>3.37</td>
<td>0.040</td>
<td>0.186</td>
<td>16</td>
<td>0.0025</td>
<td>0.0117</td>
<td>Mail::Box::IMAP4::BEGIN</td>
</tr>
<tr>
<td>3.28</td>
<td>0.039</td>
<td>0.293</td>
<td>32</td>
<td>0.0012</td>
<td>0.0092</td>
<td>base::import</td>
</tr>
<tr>
<td>2.86</td>
<td>0.034</td>
<td>0.034</td>
<td>1425</td>
<td>0.0000</td>
<td>0.0000</td>
<td>MIME::Type::simplified</td>
</tr>
<tr>
<td>2.52</td>
<td>0.030</td>
<td>0.089</td>
<td>3</td>
<td>0.0100</td>
<td>0.0297</td>
<td>Net::DNS::Resolver::UNIX::BEGIN</td>
</tr>
</tbody>
</table>
Devel::Profiler::Apache

• Drop in replacement for Apache::DProf
• dprofpp compatible log files
• Pure Perl implementation
Odd Speedups
Compressed output

- Most modern browsers support gzip’ed
- Adds CPU cycles on the server
- Adds CPU cycles on the client
- Can reduce latency
- Saves bandwidth
mod_deflate

- A good compression module
- Comes with Apache, so you already have it
mod_deflate

LoadModule deflate_module modules/mod_defalte.so

<Location />
    SetOutputFilter DEFLATE
    DeflateFilterNote Input instream
    DeflateFilterNote Output outstream
    DeflateFilterNote Ratio ratio
</Location>

LogFormat '"%r" %{outstream}n/%{instream}n (%{ratio}n%%)' deflate
CustomLog logs/deflate_log deflate

"GET /PPMPackages/zips/ HTTP/1.1" 1080/4825 (22%)
"GET /ppm.css HTTP/1.1" 278/399 (69%)
"GET /BuildStatus/5.6plus/solaris/Test-ClassAPI-1.02.txt HTTP/1.0" 503/1157 (43%)
"GET /robots.txt HTTP/1.0" 105/328 (32%)
"GET /BuildStatus/5.8-hpux/hpux-ia64-5.8/Acme-Your-0.01.txt HTTP/1.0" 972/10103 (9%)
"GET /BuildStatus/5.8-windows/windows-5.8/DBD-PgPP-0.05.txt HTTP/1.1" 371/1507 (24%)
"GET /BuildStatus/5.6plus/linux/Data-Page-Pageset-1.02.txt HTTP/1.0" 640/1606 (39%)
Databases

- Such a common back-end
- Slow & expensive
- Bottleneck
Apache::DBI

- Wrapper to DBI
- Plug & Play, no code change
- Persistent DB connections
Apache::DBI

PerlModule Apache::DBI
PerlModule DBI

use DBI;
my $dbh = DBI->connect("dsn", $user, $pass);
connect_on_init

- Pre-connects to hot DBs
- Once for each httpd child

Apache::DBI->connect_on_init($DSN, $user, $pass);
prepare_cached

- DB handles are cached
- `prepare_cached()` caches prepared statements into DB handles

```perl
my $sth = $dbh->prepare_cached("SELECT id from User where name = ?");
```
stat()

- System call
- File information (size, owner, lastmod)
- Relatively expensive
AllowOverride

• Depends on default config
• .htaccess files are searched for:
  - /.htaccess
  - /var/.htaccess
  - /var/www/.htaccess
  - /var/www/htdocs/.htaccess
AllowOverride

- We are using mod_perl
- We are speed freaks
- When we need to override, we do it with Perl
AllowOverride None

<Directory />
  AllowOverride None
</Directory>
PerlTransHandler

DocumentRoot /var/www/html/root

<Location /news/story/print/pdf>
  SetHandler modperl
  PerlHandler Story::Print::Pdf
</Location>


  stat("/var/www/html/root/news");
  stat("/var/www/html/root");
PerlTransHandler

DocumentRoot /var/www/html/root

<Location /news/story/print/pdf>
  SetHandler modperl
  PerlHandler Story::Print::Pdf
  PerlTransHandler Apache2::Const::OK
</Location>

When httpd is serving a file, it had to stat() it

- It’s cached
- We can use it
$r->finfo

sub handler {
    my $r = shift;
    my $file = $r->filename;
    if (-M $file) {  # extra stat() / extra time()
        use APR::Finfo ();
        sub handler {
            my $r = shift;
            my $finfo = $r->finfo;
            if ($finfo->mtime > $r->request_time) {
            ...
print()

- STDOUT connected to the client
- Might not be buffered
- Generally expensive
print ... print ... print

print "<HTML>\n";
print "<BODY>\n";
print "<H1>Hello</H1>\n";
print "</BODY>\n";
print "</HTML>\n";

print "<HTML>\n";
print "<BODY>\n";
print "<H1>Hello</H1>\n";
print "</BODY></HTML> EOF
• $|++;
• Buffering causes slower page loads
```perl
$r->rflush();

$|++; print $header; print $info; print $news;
my $results = some_slow_search();
print $results;
print $footer;
print
  $header,
  $info,
  $news,
;r->rflush();
my $results = some_slow_search();
print
  $result,
  $footer,
;```
Thank you!
More info

• mod_perl User’s mailing-list
  - <modperl@perl.apache.org>

• mod_perl Developer's Cookbook
  - http://www.modperlcookbook.org/

• Practical mod_perl
  - http://www.modperlbook.org/

• mod_perl at the ASF
  - http://perl.apache.org/
Thank You!

Slides and bonus material:

http://gozer.ectoplasm.org/talk/