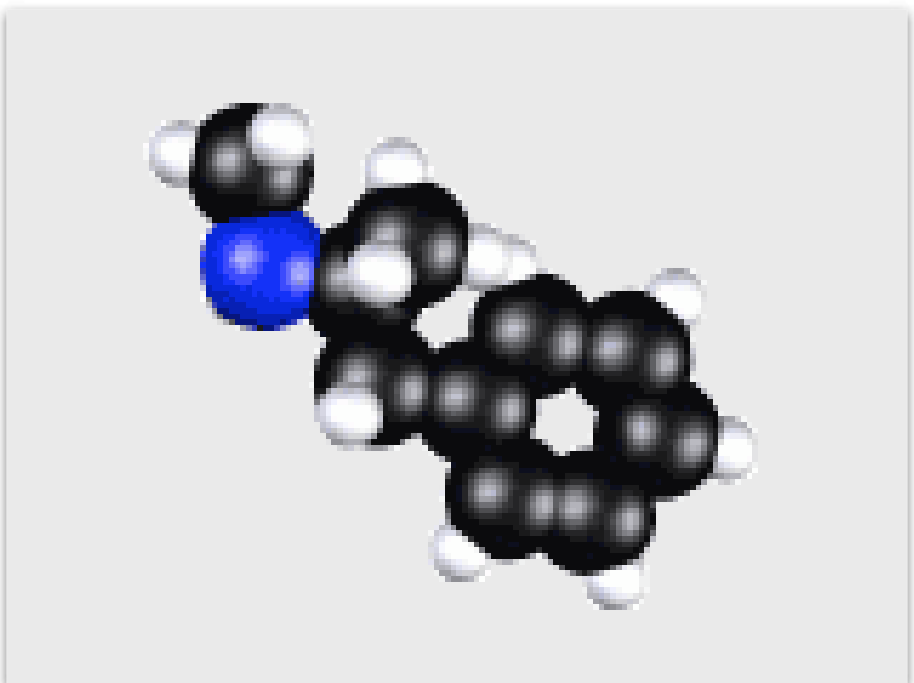
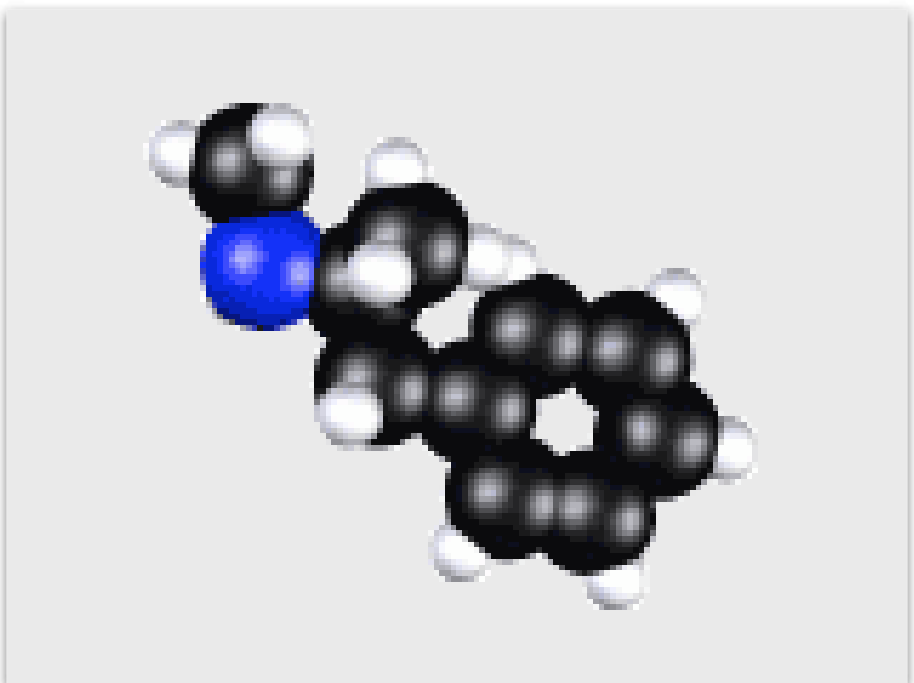
mod_perl For Speed Freaks!

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)
              min  mean[+/-sd] median   max
Connect:        1    2   1.6      3       5
Processing:     7   12   2.5     13      18
Waiting:        3    7   2.6      7      12
Total:         11   15   2.7     15      22
```

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_staus

- 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

Apache Server Status for www.apache.org

Server Version: Apache/2.0.53 (Unix)
Server Built: Feb 7 2005 06:11:24

Current Time: Monday, 24-Oct-2005 19:36:37 PDT
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

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

Srv	PID	Acc	M	CPU	SS	Req	Conn	Child	Slot	Client	VHost	Request
0-19	36922	0/1328/159920	_	5.84	2	23	0.0	86.00	4689.57	207.68.146.53	tomcat.apache.org	GET /tomcat-5.5-doc/config/http.html HTTP/1.0
1-19	36928	0/1564/162956	_	6.85	1	1	0.0	49.89	4364.08	210.72.237.66	jakarta.apache.org	GET /struts/dtds/struts-config_1_0.dtd HTTP/1.1
2-19	38006	0/1681/164770	W	5.30	151	0	0.0	69.34	4671.74	200.89.159.231	xml.apache.org	GET /dist/xerces-c/stable/xerces-c-src2_4_0.zip HTTP/1.1
3-19	38347	0/1430/159965	_	5.88	4	1	0.0	35.40	4656.92	192.51.44.47	ws.apache.org	GET /axis/ja/java/user-guide.html HTTP/1.0
4-19	36929	0/1351/162839	_	4.61	5	1478	0.0	46.89	4546.63	141.158.43.144	jakarta.apache.org	GET /commons/io/api-release/index-all.html HTTP/1.1
5-19	40336	0/1140/164617	_	3.91	2	0	0.0	59.89	4609.04	200.83.187.137	db.apache.org	GET /derby/ HTTP/1.1
6-19	36955	0/1403/159575	_	5.43	2	1	0.0	67.13	4261.59	210.17.229.149	jakarta.apache.org	GET /struts/dtds/tiles-config.dtd HTTP/1.1
7-19	41068	2/1273/160847	K	4.44	1	1	10.2	43.52	4832.32	218.80.212.235	jakarta.apache.org	GET /images/jakarta-logo.gif HTTP/1.1
8-19	36956	0/1263/159593	W	4.02	189	0	0.0	41.71	4573.99	220.231.111.3	www.apache.org	GET /dist/tomcat/tomcat-5/v5.5.12/bin/apache-tomcat-5.5.12-embe
9-19	36957	0/1635/164205	_	5.45	6	0	0.0	44.28	4733.30	24.15.13.165	www.apache.org	GET /favicon.ico HTTP/1.1
10-19	36958	0/1659/160260	_	5.91	0	0	0.0	47.78	4855.84	67.191.175.125	struts.apache.org	GET /images/logos/maven-button-1.png HTTP/1.1
11-19	38350	0/1278/153356	_	6.94	3	0	0.0	32.99	4422.39	61.152.127.10	tomcat.apache.org	GET /index.html HTTP/1.1

mod_status explained

```
Current Time: Monday, 24-Oct-2005 19:36:37 PDT
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

```
__W____KW___W___K__K__K_W_RKK__K_K___WK__KKK___KKK__KKKWK__C_WK_
KKR_K__W_KC__K___WWW_RW____KK_KC_KW_WRK__R__CW___KKC___K_K__K___
_K___K_K_W_KK_W____CW__W___RW__KR____KCCKK____KC_W__CWKWWKWK__K_
K_WWCC_W_K_K_WW_W_K__W__W___K_____K__WWWK__KK____W_K_______K_W__
K___KK___C___W_W_K_W_KK__WK_____K_W_KK__WW_K_W___W_W_WKK__K__K_
_W____K_K___KKW_KW_W_K_.....................................
```

```
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

```
$> wget -O- http://www.apache.org/server-status?auto
```

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

Apache2::Status

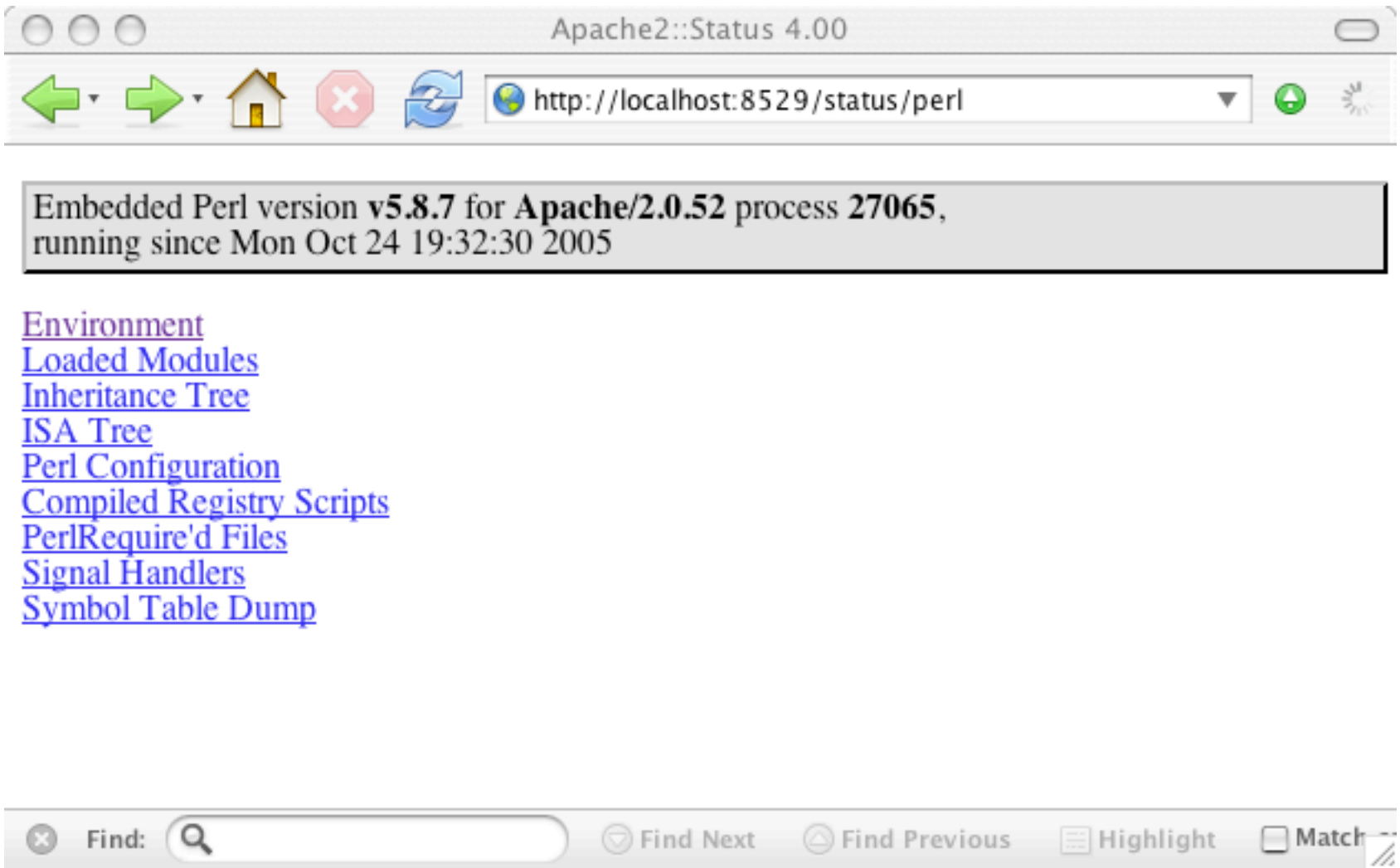
- Perl status reporting
- Great to poke at a running mod_perl server
- Comes with mod_perl, so you already have it

Enabling Apache2::Status

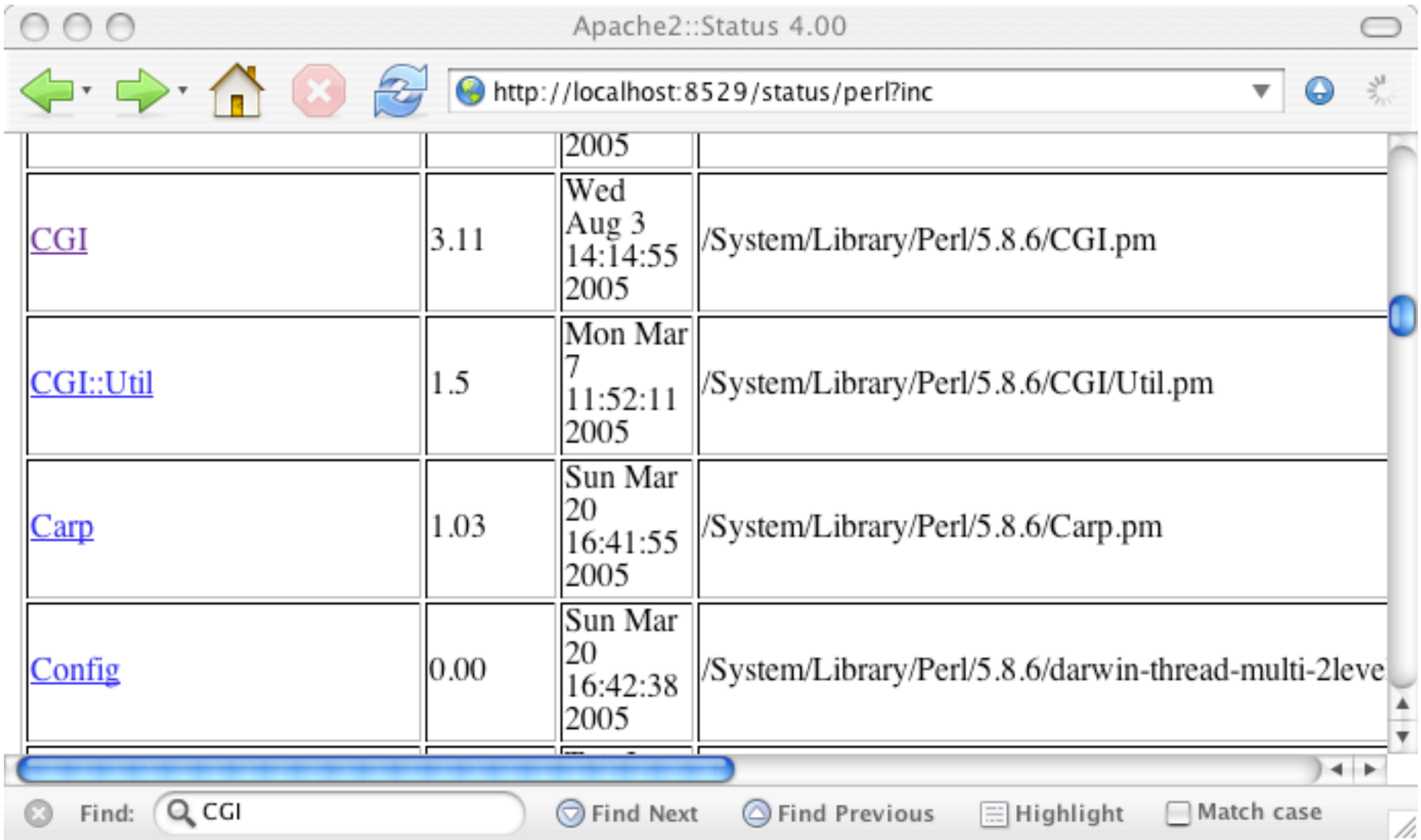
```
<Location /perl-status>
  SetHandler perl-script
  PerlHandler Apache2::Status
  PerlSet Var StatusOptionsAll On
</Location>
```

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

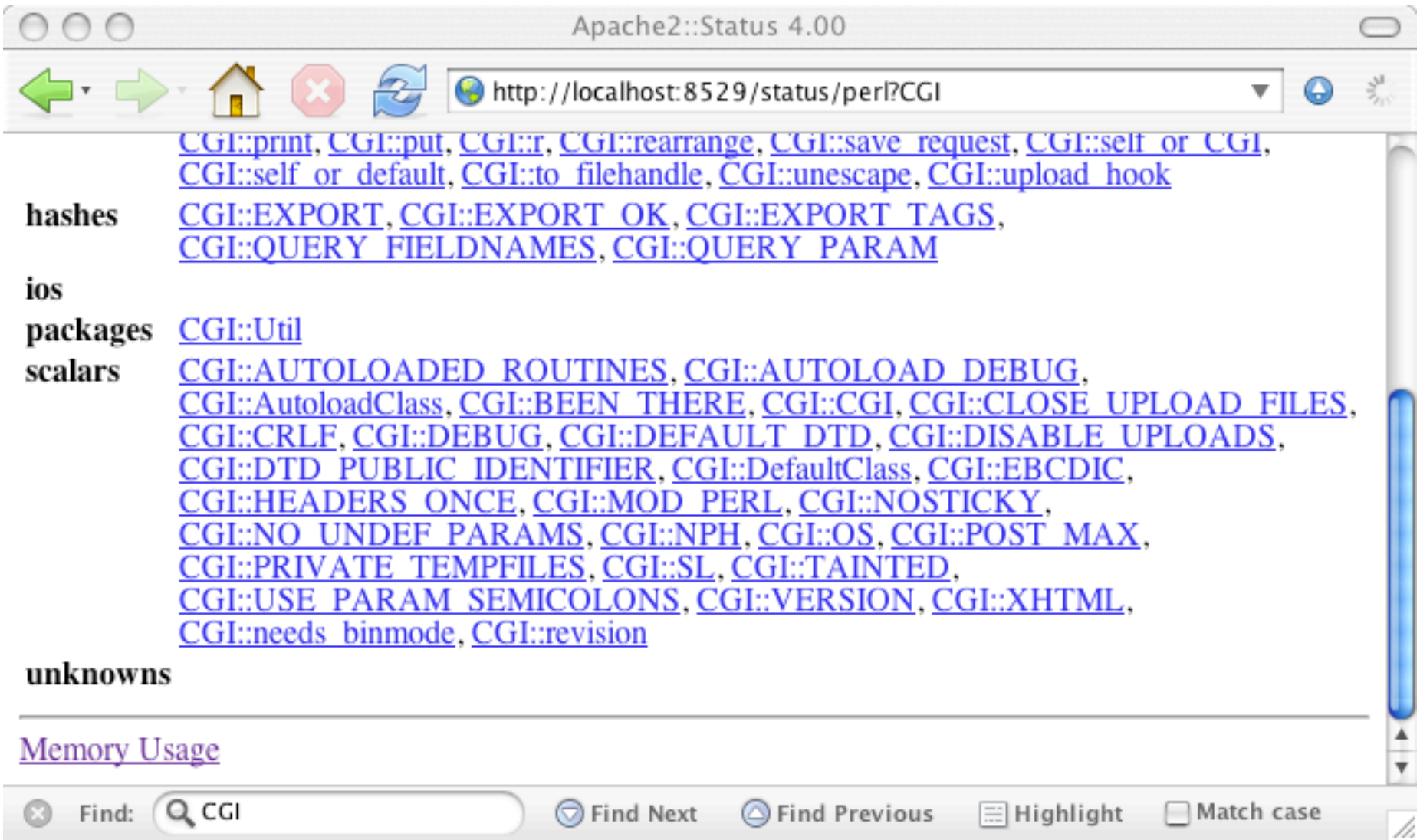
Apache2::Status



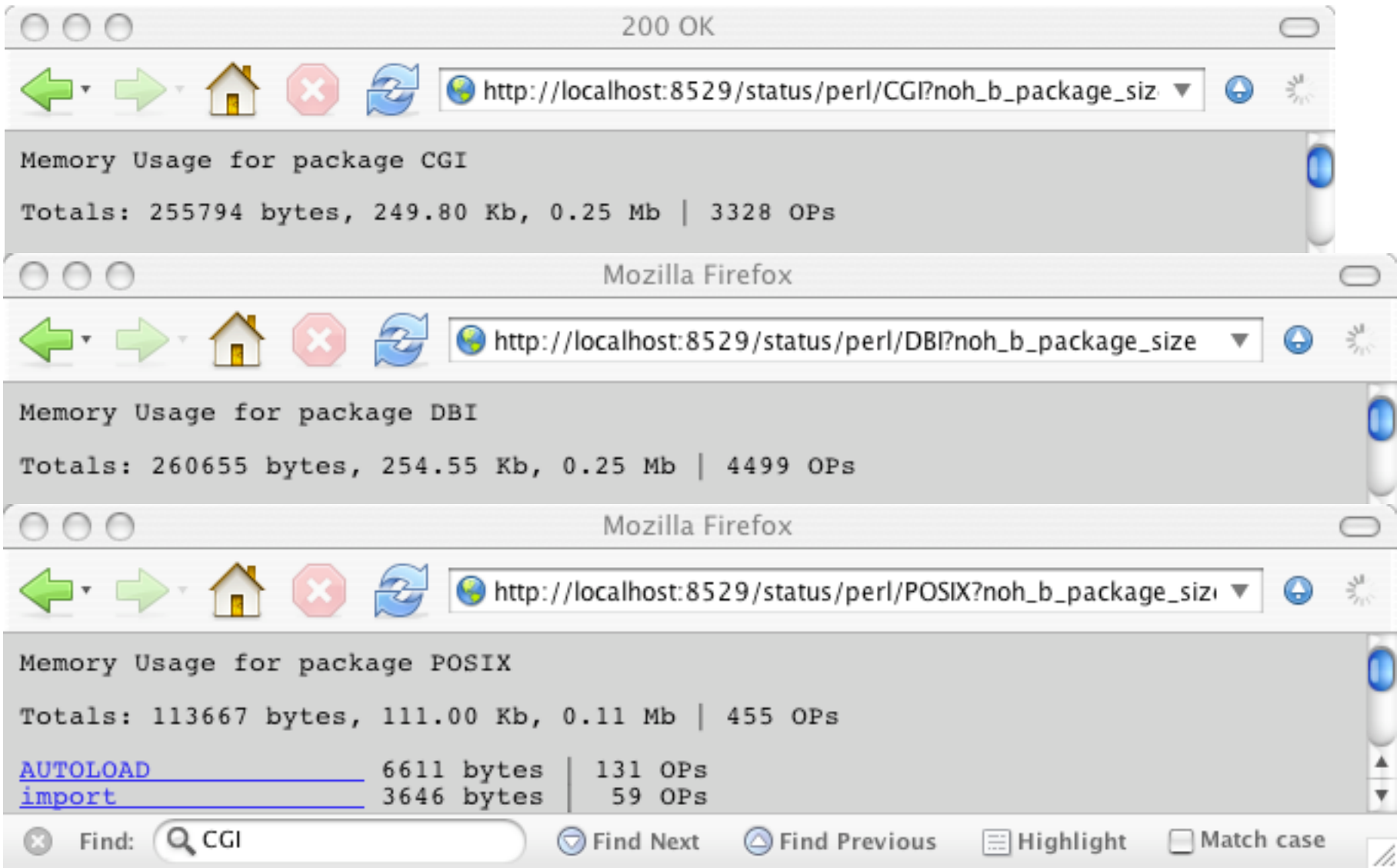
Apache2::Status



Apache2::Status



Apache2::Status



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 usage

```
$> 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 $> a
Copyright (c) 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeus
Copyright (c) 1998-2002 The Apache Software Foundation, http://www.

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/

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:       23   24   1.9      24      28
Processing:    27   32   9.1      29      67
Waiting:       27   32   9.0      29      66
Total:         51   57   9.0      56      91

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

top

```
Processes:  81 total, 2 running, 79 sleeping... 237 threads            22:06:18
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

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

GTop

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

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

```
size  => 1900544
vsize => 3108864
share => 1392640
rss   => 1900544
```

BSD::Resource

```
use BSD::Resource;
($usertime,
 $systemtime,
 $maxrss,
 $ixrss,
 $idrss,
 $isrss,
 $minflt,
 $majflt,
 $nswap,
 $inblock,
 $oublock,
 $msgsnd,
 $msgrcv,
 $nsignals,
 $nvcsw,
 $nivcsw) = 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

Registry Preloading

```
#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
```

Apache2::Const

```
use Apache2::Const qw(OK DECLINED);
```

```
use Apache2::Const -compile => qw(OK DECLINED);
```

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

Slurping

```
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;
}
```

Apache::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

Apache::DProf

```
  PerlModule Apache::DProf
```

```
$> apachectl restart
$> cd /var/httpd/logs/prof/12345
$> dprofpp
Total Elapsed Time = 6.238159 Seconds
  User+System Time = 1.188159 Seconds
Exclusive Times
%Time ExclSec CumulS #Calls sec/call Csec/c  Name
 10.8   0.129  0.463     19   0.0068 0.0244  Mail::SpamAssassin::BEGIN
 9.17   0.109  0.207     71   0.0015 0.0029  Mail::SpamAssassin::PerMsgStatus::
                                             BEGIN
 6.73   0.080  0.138      4   0.0200 0.0345  Mail::Transport::IMAP4::BEGIN
 6.65   0.079  1.121     24   0.0033 0.0467  main::BEGIN
 3.96   0.047  0.079      3   0.0158 0.0265  Mail::IMAPClient::_read_line
 3.37   0.040  0.049      8   0.0050 0.0062  Mail::SpamAssassin::Conf::BEGIN
 3.37   0.040  0.059     17   0.0023 0.0035  Net::DNS::Resolver::Base::BEGIN
 3.37   0.040  0.186     16   0.0025 0.0117  Mail::Box::IMAP4::BEGIN
 3.28   0.039  0.293     32   0.0012 0.0092  base::import
 2.86   0.034  0.034   1425   0.0000 0.0000  MIME::Type::simplified
 2.52   0.030  0.089      3   0.0100 0.0297  Net::DNS::Resolver::UNIX::BEGIN
```

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 preapred statements into DB handles

```
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>

$> GET http://localhost:8529/news/story/print/pdf/12345

     stat(“/var/www/html/root/news/story/print/pdf/12345”);
     stat(“/var/www/html/root/news/story/print/pdf”);
     stat(“/var/www/html/root/news/story/print”);
     stat(“/var/www/html/root/news/story”);
     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>
```

$r->finfo

- 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 <<’EOF’;
<HTML><BODY>
<H1>Hello</H1>
</BODY></HTML>
EOF
```

$|

- $|++;
- Buffering causes slower page loads

$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*
  - http://perl.apache.org/maillist/modperl.html
  - <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/