Scaling Apache for LAMP



bud@thinkcube.com | twitter @geekaholic

The folk story is that Apache was named after "A-patchy-server", which was the result of NCSA httpd server being patched a lot. The project was started by Brian Behlendorf



Today, Apache is still the *most popular* web server out there running more than half of the websites on the net. It is actively developed by the <u>Apache</u> <u>Software Foundation</u> along with many other software projects.

Besides its primary function of being a website, Apache can also be configured as a *reverse proxy* for load balancing.

.notes: We assume your working on Ubuntu. Translate to your favorite distro accordingly.

The easiest method of installing Apache along with PHP and MySQL (aka LAMP) is to use the tasksel command.

tasksel

Alternatively install each package manually:

apt-get install apache2 libapache2-mod-php5 mysql-server

In order to test out Apache performance as we tune it, it is good to setup a real world full fledged CMS such as Drupal.

- Download the latest version of Drupal from <u>drupal.org</u>
- Follow the <u>Drupl setup guide</u>
- Install the <u>Devel module</u> into Drupal *modules directory*
- Login to Drupal as admin and using the devel plugin, populate Drupal with sample data for testing

(Configuration -> Development -> Generate Content)

Autobench is a handy script to stress test a webserver by sending an increasing number of requests. It works by calling the httperf tool iteratively with increasing parameters.

<u>Download autobench</u> and follow directions to compile.

In order to plot graphs, you need to install gnuplot via apt. As of this writing, the script used to plot the graph has a bug calling the current version of gnuplot and requires the following minor modification.

```
$ sudo vi which bench2graph
```

line ~78 should be

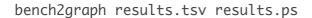
```
echo set style data linespoints >> gnuplot.cmd
```

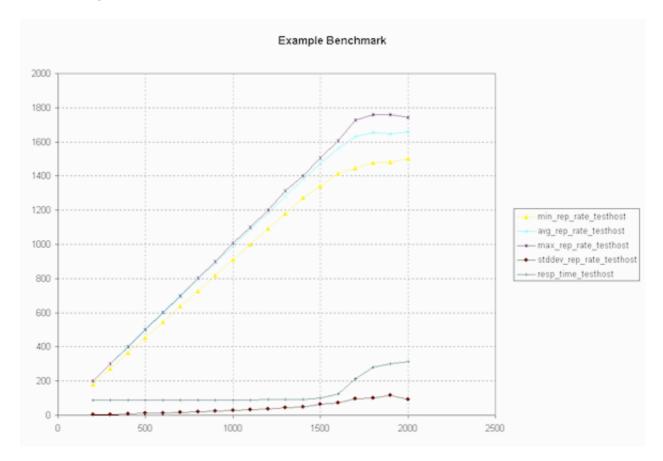
Lets benchmark our standard Apache setup to get an idea of default performance.

--num_conn 5000 --timeout 5 --file results.tsv

Basically the above will test a single host, localhost/drupal by sending it 20 connections per second, each having 10 requests up to 200 connections per second incrementing by 20. The total number of connections are capped at 5000 while any request that takes more than 5 seconds to respond is considered unsuccessful.

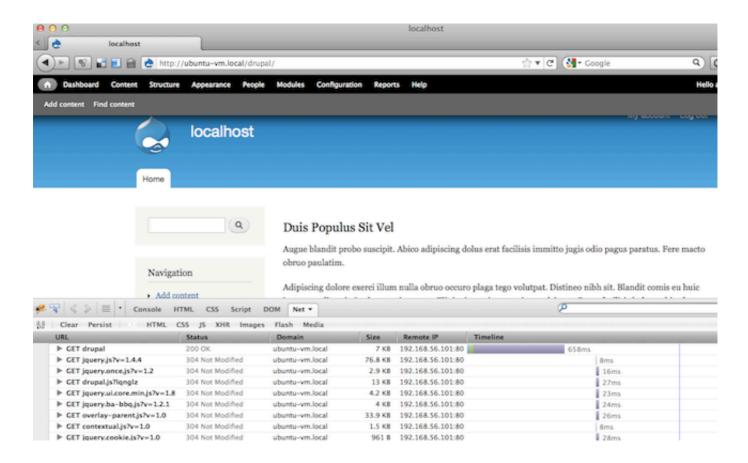
Using the result.tsv file and the included bench2graph utility, you can plot a graph into a postscript file.





You can decrease network overhead and make pages load faster, there by reducing the amount of time a client is connected by compressing pages using gzip. All modern browser support rendering compressed files.

In order to benchmark its effect, you can install a tool such as <u>Firebug</u> on the client side.



Enable the <u>mod_deflate</u> module. On Ubuntu:

a2enmod deflate && a2enmod headers

Then we'll configure deflate to compress everything except images.

sudo vi /etc/apache2/modules-enabled/deflate.conf

```
!apache
<Location />
    # Insert filter
    SetOutputFilter DEFLATE

# Don't compress images
    SetEnvIfNoCase Request_URI .(?:gif|jpe?g|png)$ no-gzip dont-vary
```

Make sure proxies don't deliver the wrong content Header append Vary User-Agent env=!dont-vary

</Location>

There are a few key parameters that can be tuned:

- KeepAlive By default its set to ON which is good. Clients will make all requests in one shot via http 1.1.
- KeepAliveTimeout Better to keep it low. Defaults to 15 sec. Make sure thats enough. Rule is 1.5 to 2 times your page load speed.
- TimeOut The default is 5 minutes which might be long to allow for one process. Adjust accordingly.
- StartServers, MinSpareServers, MaxSpareServers Generally even on a busy site you may not need to tweak. Apache can self regulate.
- MaxClients The maximum number of clients (threads) Apache will handle simultaneously.

```
ps -eafly |grep apache2|awk '{print $8}'|sort -n
```

Use free to figure out how much memory is available. Cache is also considered free memory but you might want to leave some and not assume all cache will be used.

free

By deviding free memory by the average memory used by an Apache thread, you can estimate the number of MaxClients.

e.g: Assuming Apache memory usage and free memory are as follows

```
$ ps -eafly |grep apache2|awk '{print $8}'|sort -n
816
3896
3896
```

3896 3896 20844

\$ free

•	total	used	free	shared	buffers	cached
Mem:	508904	447344	61560	0	141136	213468
<pre>-/+ buffers/cache:</pre>		92740	416164			
Swap:	407544	4364	403180			

Memory avail \sim = 60000 (free) + 100000 (cached) \sim = 160 MB and Memory per thread \sim = 4 MB Then a safe value for MaxClients = 40

We can improve PHP performance by

- 1. Caching pages (useful if dynamic content doesn't change often)
- 2. PHP Opcode optimizations (pre-compile php)

Fortunately we can get the benefit of both using <u>PHP APC</u>, which is a PHP accellerator!

apt-get install php-apc

You can verify installation by loading a php page having phpinfo(); and searching for apc. Or if you have php5-cli installed:

php -r "phpinfo();" | grep apc

<u>Memcached</u> is a distributed cache for storing key-value pairs in memory for faster access with reduced trips to the database. Some popular PHP apps can use memcache if available. memcached does not instantly accellerate PHP!

apt-get install memcached php5-memcache

service memcached start

- Keep DirectoryIndex file list as short as possible.
- Whenever possible disable .htaccess via AllowOverride none
- Use Options FollowSymLinks to simplify file access process in Apache
- Minimize the use of mod_rewrite or at least complex regexs
- If logs are unnecessary disable them or log to another server via syslog.
- For Deny/Allow rules use IPs rather then domains. (prevents superfluous DNS lookups).
- Do not enable HostnameLookups (DNS is slow).
- For dynamic sites see if you can separate dynamic vs static content into two servers