Modern LAMP

Roel Standaert

Introductie

- Lang geleden, meer dan 10 jaar, toen "Web 2.0" nog hip klonk, was de LAMP stack hot stuff
- Er was maar één antwoord als je een interactieve website wou maken, en dat was LAMP.

De klassieke LAMP stack bestond uit:

- Linux
- Apache
- MySQL
- PHP

Nu zijn we natuurlijk veel verder, dus dan rijst de vraag: is die LAMP stack nu eigenlijk nog wel relevant? Is die nog nuttig? Laten we even overlopen:

Linux

Is Linux nog relevant? Geen twijfel over.

Apache

Waarom Apache als nginx ook bestaat? Meestal een negatief antwoord, ik zal dat nog verder uitleggen waarom. Máár: ULYSSIS biedt shared hosting aan.

- `.htaccess` files
- Apache is oud, veel modules, o.a. `mod_shib` voor Shibboleth (Centrale KU Leuven login)

MySQL

- Overname van Sun door Oracle: toekomst onzeker
- Forks: we volgen er één: MariaDB
- Zelf meer fan van PostgreSQL:
  - volgt SQL standaard, implementeert meer van die standaard
  - meer features
  - ULYSSIS biedt dit ook aan als alternatief
- We gaan MariaDB opzetten, maar vooral focussen op Apache en PHP.

PHP

- Hekel aan PHP, vreselijke programmeertaal. PHP: a fractal of bad design.
- Realiteit: mensen willen Wordpress, Drupal, MediaWiki,…

- Lage instapdrempel, ULYSSIS brengt informatica dichter bij de studenten, wij moeten PHP aanbieden

Conclusie

LAMP nog relevant: voor ULYSSIS, voor het shared hosting-verhaal wel. Voor al de rest niet.

Omgeving

Xubuntu 14.04 VM, GUI zodat iedereen vlot kan meevolgen. Ik zou zelf SSH, vim en tmux gebruiken.

We kunnen spijtig genoeg geen 16.04 gebruiken, omdat 16.04 met PHP 7 komt, en we de htscanner module gebruiken en die niet werkt met PHP 7. Ik ga nog uitzoeken of ik dat misschien kan patchen, zodat we die module bij ULYSSIS kunnen blijven gebruiken.

Bij ULYSSIS draaien we momenteel ook nog 14.04.

We hebben wel een recentere versie van Apache en PHP nodig, dus PPA:

```
sudo add-apt-repository ppa:ondrej/php5
sudo apt-get update
```

Dit is al vooraf gedaan op de VM die jullie hebben.

Basic LAMP stack

Installeer Apache 2, PHP 5 en MySQL:

```
sudo apt-get install apache2 libapache2-mod-php5 mariadb-server php5-mysqlnd
ps aux | grep apache
ps aux | grep php # geeft niets, want module van Apache
ps aux | grep maria # geeft niets, want drop in replacement, met zelfde naam als mysql
ps aux | grep mysql
mysql --version
```

Surf naar http://localhost.

Maak bestand:

/var/www/html/info.php

```
<?php
phpinfo();
// Normaal worden PHP-tags niet gesloten
```

Surf naar http://localhost/info.php.

- Ctrl-F mysql
- Versie: deb.sury.org, want PPA, niet de gewone package
- Server API is Apache 2.0

Download een MySQL-testbestand:

```
cd /var/www/html
sudo wget 'ftp://ftp.ulyssis.org/Modern LAMP workshop/db.php'
cat db.php
```

Surf naar http://localhost/db.php

Ok. Nu hebben we een werkende LAMP stack. We zijn dus klaar en we kunnen naar huis…

Was het maar zo gemakkelijk. ULYSSIS heeft lang deze setup gebruikt, maar hier zijn een aantal problemen mee, die onderling een beetje verbonden zijn:

User

PHP draait als dezelfde user als Apache, nl. www-data

```
ps aux | grep apache
```

/var/www/html/user.php

```
<?php
echo `id`; // id toont huidige user, group, etc.
```

```
ps aux | grep apache2
```

Probleem voor veiligheid, probleem met veel shared hosting. Oplossing vaak halfslachtige safe modes, wij gaan het netter doen.

- Resources-verbruik moeilijk te volgen
- Owner is fout, we kunnen ze niet verwijderen,… Wij hadden een cronjob lopen, pain in the ass

Prefork

MPMs: Multi-Processing Modules

Apache heeft meerdere MPMs: prefork, worker, en event. Met `mod_php` kunnen we enkel `mpm_prefork` gebruiken, vanwege thread safety redenen.

Prefork

Prefork creëert een nieuw proces, en elk proces is verantwoordelijk voor één connectie. Veel connecties, keep-alive, schaalprobleem. Ook: veel geheugenverbruik, en PHP moet ingeladen worden voor elk Apache-proces, zelfs als je enkel een statische pagina wil laden.

Worker

Niet bruikbaar met `mod_php`. Lichte verbetering over prefork: in de plaats van processen, gebruikt worker threads, maar er is weer een mapping van connectie op thread.

Event

Niet bruikbaar met `mod_php`. Verbetering over worker, momenteel de best beschikbare MPM. In de plaats van connecties op threads te mappen, mapt deze requests op threads. Een connectie zit niet vast op één thread. Bevordert het hergebruik van threads. Minder threads spawnen = sneller, meer capaciteit. Wij gebruiken dus liefst `mpm_event`. Maar wat moeten we dan doen met PHP?

PHP-FPM

PHP-FPM lost ons probleem op: we kunnen voor elke user een ander PHP-proces draaien, als die user. En omdat PHP-FPM niet binnen Apache draait, kunnen we `mpm_event` gebruiken, en is het geheugenverbruik van Apache wat lichter.

PHP-FPM installeren:

```
sudo apt-get install php5-fpm
ps aux | grep php
```

```
sudo adduser jos
```

We maken een pool:

/etc/php5/fpm/pool.d/jos.conf

```
[jos]
user = jos # stelt de user in voor de processen in de pool
group = jos # stelt de group in voor de processen in de pool
listen = /var/run/php5-fpm-jos.sock # Stelt de socket in waarop deze pool luistert
naar requests
listen.owner = www-data # Stelt de owner van de socket in
listen.mode = 0600 # Stelt de socket in op lees+schrijfrechten voor www-data, geen
permissions voor group en other
request_terminate_timeout = 5m # Kill een proces als het langer dan 5 minuten met een
request bezig is
pm = ondemand # Begin met 0 processen, start meer processen on demand (alternatieven:
static en dynamic)
pm.max_children = 12 # Stelt het maximaal aantal processen voor deze pool in
pm.process_idle_timeout = 10s # Terminate een proces als het 10 seconden lang geen
request meer heeft gekregen
```

```
sudo service php5-fpm restart
ps aux | grep php
ls -la /run | grep sock
```

Intermezzo: infopagina

```
sudo a2enmod info
```

Ga naar http://localhost/server-info/.

Back to the main workshop

```
sudo a2enmod proxy_fcgi # er is ook mod_fastcgi en mod_fcgid, mod_fastcgi is oud,
mod_fcgid is niet geschikt
sudo service apache2 restart
```

Edit `/etc/apache2/sites-available/000-default.conf`. Voeg toe:

/etc/apache2/sites-available/000-default.conf

```
<FilesMatch \.php$>
  SetHandler "proxy:unix:/var/run/php5-fpm-jos.sock|fcgi://localhost/"
</FilesMatch>
```

```
sudo service apache2 restart
```

Ga naar `http://localhost/info.php`

- Kijk naar Server API

```
ps aux | grep php
```

Surf naar `http://localhost/user.php`.

```
sudo chown jos: /var/www/html/db.php
```

Verander naar `mpm_event`:

```
sudo a2dismod php5
sudo a2dismod mpm_prefork
sudo a2enmod mpm_event
sudo service apache2 restart
```

Ga naar http://localhost/server-info, nu staat er dat we de event MPM gebruiken.

Zijn we nu klaar? Nee!

htaccess files met php_value en php_flag

Ga naar documentatie over error logging: https://docs.ulyssis.org/Managing_PHP_errors. Wij willen dat users die dingen kunnen instellen.

Daarvoor bestaat er htscanner!

```
sudo apt-get install build-essential php5-dev apache2-dev
sudo pecl install htscanner
sudo pecl upgrade
```

Als je PECL-modules wil updaten:

```
sudo pecl upgrade
```

We moeten htscanner laden, dus maak file:

/etc/php5/mods-available/htscanner.ini

```
extension=htscanner.so
```

```
sudo php5enmod htscanner
sudo service php5-fpm restart
```

Kijk naar http://localhost/info.php, zoek op htscanner

Maak `/var/www/html/.htaccess`:

/var/www/html/.htaccess

```
php_flag log_errors on
php_value error_log /home/jos/error.log
```

Edit `/etc/apache2/sites-available/000-default.conf`:

Add:

```
<Directory /var/www/html>
    AllowOverride All
</Directory>
```

Internal server error! Toon errorlog:

```
sudo tail /var/log/apache2/error.log
```

```
pecl download htscanner
tar xvf htscanner-1.0.1.tgz
cd htscanner-1.0.1.tgz
sudo apxs -i -a -c mod_htscanner2.c # -i: install -a: activate -c compile
sudo apache2 restart
```

Laad http://localhost/info.php opnieuw

Ctrl-f htscanner

Toon http://localhost/server-info

Ctrl-f htscanner

```
sudo chown www-data: db.php
sudo cat /home/jos/error.log
```

Event-driven en nginx

`mpm_event` is niet écht event-driven.

Doe dit nooit op de ULYSSIS-servers (remember: we weten welke user ervoor verantwoordelijk is, en we kennen uw studentennummer)

/var/www/html/infinite_loop.php

```
<?php
for (;;) {}
```

Ga naar http://localhost/infinite_loop.php

```
for i in {0..1000}; do (wget http://localhost/infinite_loop.php & ); done
```

Ga naar http://localhost/

```
sudo tail /var/log/apache2/error.log
sudo service php5-fpm restart
```

Ga naar http://localhost/

Meestal wordt dit probleem niet veroorzaakt omdat PHP in een oneindige lus zit, maar omdat PHP zelf zit te wachten op IO, bv. als PHP zelf een request doet naar een server.

Dit probleem heeft nginx niet!