目次

PHP5-fpm(with nginx) の導入

MA-E3xx には標準で下記のスクリプト言語を導入してあります。

- Python-3.4
- nodejs-0.10.25
- LuaJIT-2.0.2
- Perl-5.18.2

オープンソースの Web Application では PHP で書かれたものが多くありますので PHP を導入したいということも多いと思います。
nginx と一緒に導入されることの多い PHP5-fpm をインストールする方法を紹介します。

セットアップ

パッケージのインストール

Ubuntu の apt-get により php5-fpm をインストールします。

```
user1@plum:~$ sudo apt-get update
Ign http://ppa.launchpad.net trusty InRelease
Ign http://ports.ubuntu.com trusty InRelease
Ign http://ports.ubuntu.com trusty-updates InRelease
Get:1 http://ppa.launchpad.net trusty Release.gpg [316 B]
Ign http://ports.ubuntu.com trusty-security InRelease
Get:2 http://ppa.launchpad.net trusty Release [14.0 kB]
Get:3 http://ports.ubuntu.com trusty Release.gpg [933 B]
Get:4 http://ppa.launchpad.net trusty/main armhf Packages [1328 B]
Get:5 http://ports.ubuntu.com trusty-updates Release.gpg [933 B]
Get:6 http://ports.ubuntu.com trusty-security Release.gpg [933 B]
Get:7 http://ports.ubuntu.com trusty Release [58.5 kB]
Get:8 http://ports.ubuntu.com trusty-updates Release [58.5 kB]
Ign http://ppa.launchpad.net trusty/main Translation-en
Get:9 http://ports.ubuntu.com trusty-security Release [58.5 kB]
Get:10 http://ports.ubuntu.com trusty/main armhf Packages [1295 kB]
Get:11 http://ports.ubuntu.com trusty/restricted armhf Packages [14 B]
Get:12 http://ports.ubuntu.com trusty/universe armhf Packages [5710 kB]
Get:13 http://ports.ubuntu.com trusty/main Translation-en [762 kB]
Get:14 http://ports.ubuntu.com trusty/restricted Translation-en [3457 B]
Get:15 http://ports.ubuntu.com trusty/universe Translation-en [4089 kB]
Get:16 http://ports.ubuntu.com trusty-updates/main armhf Packages [212 kB]
Get:17 http://ports.ubuntu.com trusty-updates/restricted armhf Packages [14
B]
Get:18 http://ports.ubuntu.com trusty-updates/universe armhf Packages [153
kB]
Get:19 http://ports.ubuntu.com trusty-updates/main Translation-en [99.5 kB]
```

```
Get:20 http://ports.ubuntu.com trusty-updates/restricted Translation-en [14
B]
Get:21 http://ports.ubuntu.com trusty-updates/universe Translation-en [75.1
kB]
Get:22 http://ports.ubuntu.com trusty-security/main armhf Packages [103 kB]
Get:23 http://ports.ubuntu.com trusty-security/restricted armhf Packages [14
B]
Get:24 http://ports.ubuntu.com trusty-security/universe armhf Packages [36.4
kB]
Get:25 http://ports.ubuntu.com trusty-security/main Translation-en [50.2 kB]
Get:26 http://ports.ubuntu.com trusty-security/restricted Translation-en [14
B]
Get:27 http://ports.ubuntu.com trusty-security/universe Translation-en [20.1
kB]
Fetched 12.8 MB in 45s (280 kB/s)
Reading package lists... Done
```

```
user1@plum:~$ sudo apt-cache search php5-fpm
php5-fpm - server-side, HTML-embedded scripting language (FPM-CGI binary)
```

```
user1@plum:~$ sudo apt-get install php5-fpm
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  libsystemd-daemon0 lsof php5-common php5-json psmisc
Suggested packages:
  php5-user-cache php-pear
The following NEW packages will be installed:
  libsystemd-daemon0 lsof php5-common php5-fpm php5-json psmisc
0 upgraded, 6 newly installed, 0 to remove and 2 not upgraded.
Need to get 2698 kB of archives.
After this operation, 8742 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://ports.ubuntu.com/ubuntu-ports/ trusty-updates/main libsystemd-
daemon0 armhf 204-5ubuntu20.3 [9040 B]
Get:2 http://ports.ubuntu.com/ubuntu-ports/ trusty/main psmisc armhf
22.20-1ubuntu2 [50.3 kB]
Get:3 http://ports.ubuntu.com/ubuntu-ports/ trusty/main lsof armhf
4.86+dfsg-1ubuntu2 [253 kB]
Get:4 http://ports.ubuntu.com/ubuntu-ports/ trusty/main php5-json armhf
1.3.2-2build1 [30.8 kB]
Get:5 http://ports.ubuntu.com/ubuntu-ports/ trusty-updates/main php5-common
armhf 5.5.9+dfsg-1ubuntu4.3 [431 kB]
Get:6 http://ports.ubuntu.com/ubuntu-ports/ trusty-updates/universe php5-fpm
armhf 5.5.9+dfsg-1ubuntu4.3 [1924 kB]
Fetched 2698 kB in 6s (444 kB/s)
Selecting previously unselected package libsystemd-daemon0:armhf.
(Reading database ... 18215 files and directories currently installed.)
Preparing to unpack .../libsystemd-daemon0_204-5ubuntu20.3_armhf.deb ...
```

```
Unpacking libsystemd-daemon0:armhf (204-5ubuntu20.3) ...
Selecting previously unselected package psmisc.
Preparing to unpack .../psmisc_22.20-1ubuntu2_armhf.deb ...
Unpacking psmisc (22.20-1ubuntu2) ...
Selecting previously unselected package lsof.
Preparing to unpack .../lsof_4.86+dfsg-1ubuntu2_armhf.deb ...
Unpacking lsof (4.86+dfsg-1ubuntu2) ...
Selecting previously unselected package php5-json.
Preparing to unpack .../php5-json_1.3.2-2build1_armhf.deb ...
Unpacking php5-json (1.3.2-2build1) ...
Selecting previously unselected package php5-common.
Preparing to unpack .../php5-common_5.5.9+dfsg-1ubuntu4.3_armhf.deb ...
Unpacking php5-common (5.5.9+dfsg-1ubuntu4.3) ...
Selecting previously unselected package php5-fpm.
Preparing to unpack .../php5-fpm_5.5.9+dfsg-1ubuntu4.3_armhf.deb ...
Unpacking php5-fpm (5.5.9+dfsg-1ubuntu4.3) ...
Processing triggers for ureadahead (0.100.0-16) ...
Setting up libsystemd-daemon0:armhf (204-5ubuntu20.3) ...
Setting up psmisc (22.20-1ubuntu2) ...
Setting up lsof (4.86+dfsg-1ubuntu2) ...
Setting up php5-common (5.5.9+dfsg-1ubuntu4.3) ...

Creating config file /etc/php5/mods-available/pdo.ini with new version
php5_invoke: Enable module pdo for fpm SAPI

Creating config file /etc/php5/mods-available/opcache.ini with new version
php5_invoke: Enable module opcache for fpm SAPI
Setting up php5-fpm (5.5.9+dfsg-1ubuntu4.3) ...

Creating config file /etc/php5/fpm/php.ini with new version
php5_invoke opcache: already enabled for fpm SAPI
php5_invoke pdo: already enabled for fpm SAPI
php5-fpm start/running, process 1888
Setting up php5-json (1.3.2-2build1) ...
php5_invoke: Enable module json for fpm SAPI
Processing triggers for libc-bin (2.19-0ubuntu6) ...
Processing triggers for ureadahead (0.100.0-16) ...
localepurge: Disk space freed in /usr/share/locale: 0 KiB
localepurge: Disk space freed in /usr/share/man: 0 KiB

Total disk space freed by localepurge: 0 KiB

user1@plum:~$
```

起動**しているか**確認**してみます**。

```
user1@plum:~$ ps ax|grep php
 1888 ?        Ss     0:00 php-fpm: master process (/etc/php5/fpm/php-
fpm.conf)
```

```
 1890 ?        S      0:00 php-fpm: pool www
 1891 ?        S      0:00 php-fpm: pool www
 1989 pts/1    S+     0:00 grep --color=auto php
user1@plum:~$
```

起動が確認できました。

nginx の設定変更

PHPスクリプトへのリクエストを php-fpm へ転送するよう nginx の設定を変更します。

```
user1@plum:~$ sudo nano -w /etc/nginx/sites-enabled/default
```

php5-fpm の設定部分のコメントを外します。

```
        # pass the PHP scripts to FastCGI server listening on 127.0.0.1:9000
        #
        location ~ \.php$ {
                fastcgi_split_path_info ^(.+\.php)(/.+)$;
        #       # NOTE: You should have "cgi.fix_pathinfo = 0;" in php.ini
        #
        #       # With php5-cgi alone:
        #       fastcgi_pass 127.0.0.1:9000;
                # With php5-fpm:
                fastcgi_pass unix:/var/run/php5-fpm.sock;
                fastcgi_index index.php;
                include fastcgi_params;
        }
```

indexディレクティブに “index.php” を追加しておきます。

```
server {
        listen 80 default_server;
        listen [::]:80 default_server ipv6only=on;

        root /usr/share/nginx/html;
        index index.html index.htm index.php;
        ...
```

CPUコア数は 1 つなので /etc/nginx/nginx.conf の設定も変更しておきます。
worker_processes を 2 にします。

```
user1@plum:~$ sudo nano -w /etc/nginx/nginx.conf
```

```
user www-data;
worker_processes 2;
pid /run/nginx.pid;

events {
        worker_connections 768;
        # multi_accept on;
        ...
```

nginx の再起動

これで必要最小限の設定は完了です。変更した設定を反映させるため、nginx を再起動させます。

```
user1@plum:~$ sudo service nginx restart
 * Restarting nginx nginx
[ OK ]
user1@plum:~$
```

動作確認

テストスクリプトの作成

デフォルトの設定では、nginx の root ディレクティブは “/usr/share/nginx/html” に設定されているため、簡単な PHP スクリプトを /usr/share/nginx/html/phptest.php として作成します。

```
user1@plum:~$ cd /usr/share/nginx/html/
user1@plum:/usr/share/nginx/html$ ls -l
total 2
-rw-r--r-- 1 root root 537 Mar  4 20:46 50x.html
-rw-r--r-- 1 root root 612 Mar  4 20:46 index.html
```

```
user1@plum:/usr/share/nginx/html$ sudo nano -w phptest.php
```

phptest.php

```
<?php echo "Hello, PHP!\n"; ?>
```

アクセスしてみる

curl コマンドで、上で作成したスクリプトの URL にアクセスしてみます。

```
% curl -v http://192.168.253.47/phptest.php
* About to connect() to 192.168.253.47 port 80 (#0)
*   Trying 192.168.253.47...
* Adding handle: conn: 0x754a40
* Adding handle: send: 0
* Adding handle: recv: 0
* Curl_addHandleToPipeline: length: 1
* - Conn 0 (0x754a40) send_pipe: 1, recv_pipe: 0
* Connected to 192.168.253.47 (192.168.253.47) port 80 (#0)
> GET /phptest.php HTTP/1.1
> User-Agent: curl/7.32.0
> Host: 192.168.253.47
> Accept: */*
>
< HTTP/1.1 200 OK
* Server nginx/1.4.6 (Ubuntu) is not blacklisted
< Server: nginx/1.4.6 (Ubuntu)
< Date: Mon, 14 Jul 2014 04:52:07 GMT
< Content-Type: text/html
< Transfer-Encoding: chunked
< Connection: keep-alive
< X-Powered-By: PHP/5.5.9-1ubuntu4.3
<
Hello, PHP!
* Connection #0 to host 192.168.253.47 left intact
```

動作が確認できました。

設定のチューニング

デフォルト設定のままですと、毎回 PHP スクリプトが実行されてしまい、PC サーバと比較すると処理能力が劣る MA-E3xx には
荷が重いことがあります。 チューニングをするまえに、Apache Bench で性能を測定してみます。

リクエスト数(-n)は 1000、並列数(-c)は 10,50,100 で行ってみます。

性能測定 (デフォルト設定)

並列数 10

```
Server Software:        nginx/1.4.6
Server Hostname:        192.168.253.48
Server Port:            80

Document Path:          /phptest.php
Document Length:        12 bytes

Concurrency Level:      10
Time taken for tests:   2.702 seconds
Complete requests:      1000
Failed requests:        0
Write errors:           0
Total transferred:      178000 bytes
HTML transferred:       12000 bytes
Requests per second:    370.13 [#/sec] (mean)
Time per request:       27.018 [ms] (mean)
Time per request:       2.702 [ms] (mean, across all concurrent requests)
Transfer rate:          64.34 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.1      0       1
Processing:     8   27   3.5     26      48
Waiting:        8   26   3.4     26      48
Total:          9   27   3.4     27      48

Percentage of the requests served within a certain time (ms)
  50%     27
  66%     27
  75%     28
  80%     28
  90%     28
  95%     29
  98%     41
  99%     46
 100%     48 (longest request)
```

並列数 50

```
Server Software:        nginx/1.4.6
Server Hostname:        192.168.253.48
Server Port:            80

Document Path:          /phptest.php
Document Length:        12 bytes
```

```
Concurrency Level:      50
Time taken for tests:   2.669 seconds
Complete requests:      1000
Failed requests:        0
Write errors:           0
Total transferred:      178000 bytes
HTML transferred:       12000 bytes
Requests per second:    374.64 [#/sec] (mean)
Time per request:       133.461 [ms] (mean)
Time per request:       2.669 [ms] (mean, across all concurrent requests)
Transfer rate:          65.12 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.4      0       3
Processing:    16  130  16.6    133     157
Waiting:       16  130  16.6    133     157
Total:         19  131  16.2    133     157

Percentage of the requests served within a certain time (ms)
  50%    133
  66%    134
  75%    135
  80%    135
  90%    137
  95%    151
  98%    154
  99%    155
 100%    157 (longest request)
```

並列数100

```
Server Software:        nginx/1.4.6
Server Hostname:        192.168.253.48
Server Port:            80

Document Path:          /phptest.php
Document Length:        12 bytes

Concurrency Level:      100
Time taken for tests:   2.670 seconds
Complete requests:      1000
Failed requests:        0
Write errors:           0
Total transferred:      178000 bytes
HTML transferred:       12000 bytes
Requests per second:    374.50 [#/sec] (mean)
```

```
Time per request:       267.020 [ms] (mean)
Time per request:       2.670 [ms] (mean, across all concurrent requests)
Transfer rate:          65.10 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    1   0.9      0       5
Processing:    22  255  47.5    267     491
Waiting:       22  255  47.5    266     491
Total:         24  255  47.1    267     496
WARNING: The median and mean for the initial connection time are not within
a normal deviation
        These results are probably not that reliable.

Percentage of the requests served within a certain time (ms)
  50%    267
  66%    268
  75%    268
  80%    269
  90%    275
  95%    280
  98%    286
  99%    303
 100%    496 (longest request)
```

Fast CGI Cache の導入

PHPの負荷を下げるため、nginx の FastCGIModule の Cache 機能を使用してみます。

nginx の設定

/etc/nginx/nginx.conf に、caching の設定を追加します。

```
user1@plum:~$ sudo nano -w /etc/nginx/nginx.conf
```

```
        #passenger_root /usr;
        #passenger_ruby /usr/bin/ruby;

        ##
        # Fastcgi_cache Settings
        ##
        fastcgi_cache_path /var/run/nginx-cache levels=1 keys_zone=PHP:4m
inactive=7d max_size=50m;
        fastcgi_cache_key "$scheme$request_method$host$request_uri";
```

```
        fastcgi_cache_use_stale error timeout invalid_header http_500;

        ##
        # Virtual Host Configs
        ##

        include /etc/nginx/conf.d/*.conf;
        include /etc/nginx/sites-enabled/*;
}
```

/etc/nginx/site-enabled/default も編集します。

```
user1@plum:~$ sudo nano -w /etc/nginx/sites-enabled/default
```

```
        # pass the PHP scripts to FastCGI server listening on 127.0.0.1:9000
        #
        location ~ \.php$ {
                fastcgi_split_path_info ^(.+\.php)(/.+)$;
        #       # NOTE: You should have "cgi.fix_pathinfo = 0;" in php.ini
        #
        #       # With php5-cgi alone:
        #       fastcgi_pass 127.0.0.1:9000;
                # With php5-fpm:
                fastcgi_pass unix:/var/run/php5-fpm.sock;
                fastcgi_index index.php;
                include fastcgi_params;

                # fastcgi cache
                fastcgi_cache PHP;
                fastcgi_cache_valid  200 1d;
                fastcgi_cache_valid  any 10d;
        }
```

※ 注意
この設定では、すべてのリクエストをキャシュしてしまいます。
WordPress などを利用する場合、URL によってキャッシュをバイパスするように設定を行う必要があります。

設定ができたら、変更を反映するため nginx を再起動します。

```
user1@plum:~$ sudo service nginx restart
 * Restarting nginx nginx
[ OK ]
user1@plum:~$
```

設定ができたかどうかは、キャッシュディレクトリが作られているかどうかで確認できます。

```
user1@plum:~$ ls -l /var/run/ | grep nginx
drwx------  2 www-data   root         40 Jul 14 15:26 nginx-cache <---- キャッシュディレクトリ
-rw-r--r--  1 root       root          5 Jul 14 15:26 nginx.pid
user1@plum:~$
```

性能測定 (FastCGI Cache)

デフォルト設定と同様のベンチマークを行ってみます。

並列数 10

```
Server Software:        nginx/1.4.6
Server Hostname:        192.168.253.47
Server Port:            80

Document Path:          /phptest.php
Document Length:        12 bytes

Concurrency Level:      10
Time taken for tests:   1.176 seconds
Complete requests:      1000
Failed requests:        0
Write errors:           0
Total transferred:      178000 bytes
HTML transferred:       12000 bytes
Requests per second:    850.24 [#/sec] (mean)
Time per request:       11.761 [ms] (mean)
Time per request:       1.176 [ms] (mean, across all concurrent requests)
Transfer rate:          147.80 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.1      0       3
Processing:     7   11   1.0     11      18
Waiting:        6   11   1.0     11      18
Total:          7   12   1.0     12      19

Percentage of the requests served within a certain time (ms)
  50%     12
  66%     12
  75%     12
  80%     12
  90%     12
  95%     13
```

```
  98%     14
  99%     16
 100%     19 (longest request)
```

並列数 50

```
Server Software:        nginx/1.4.6
Server Hostname:        192.168.253.47
Server Port:            80

Document Path:          /phptest.php
Document Length:        12 bytes

Concurrency Level:      50
Time taken for tests:   1.195 seconds
Complete requests:      1000
Failed requests:        0
Write errors:           0
Total transferred:      178000 bytes
HTML transferred:       12000 bytes
Requests per second:    837.03 [#/sec] (mean)
Time per request:       59.735 [ms] (mean)
Time per request:       1.195 [ms] (mean, across all concurrent requests)
Transfer rate:          145.50 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.4      0       3
Processing:     6   58   8.3     59      78
Waiting:        6   58   8.3     59      78
Total:          9   59   8.0     59      79

Percentage of the requests served within a certain time (ms)
  50%     59
  66%     60
  75%     60
  80%     60
  90%     62
  95%     70
  98%     78
  99%     78
 100%     79 (longest request)
```

並列数100

```
Server Software:        nginx/1.4.6
Server Hostname:        192.168.253.47
Server Port:            80

Document Path:          /phptest.php
Document Length:        12 bytes

Concurrency Level:      100
Time taken for tests:   1.164 seconds
Complete requests:      1000
Failed requests:        0
Write errors:           0
Total transferred:      178000 bytes
HTML transferred:       12000 bytes
Requests per second:    859.36 [#/sec] (mean)
Time per request:       116.365 [ms] (mean)
Time per request:       1.164 [ms] (mean, across all concurrent requests)
Transfer rate:          149.38 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    1   1.0      0       6
Processing:    11  112  45.6    118     326
Waiting:       11  112  45.6    118     326
Total:         13  113  46.2    119     332
WARNING: The median and mean for the initial connection time are not within
a normal deviation
        These results are probably not that reliable.

Percentage of the requests served within a certain time (ms)
  50%    119
  66%    120
  75%    121
  80%    121
  90%    123
  95%    129
  98%    304
  99%    321
 100%    332 (longest request)
```

考察

FastCGI Cacheを導入したことで、性能が改善することがわかりました。

ab 並列数	Requests per second	
	Default	FastCGI Cache
10	370.13	850.24
50	374.64	837.03
100	374.50	859.36