Apache unleashed

Apache 2.x und PHP, multithreaded dank FastCGI

23.Februar 2012 Goesta Smekal

smekal.at :: Ihr Open Source Systemintegrator
http://smekal.at :: office@smekal.at

smekal.at :: Goesta Smekal

- 15 Jahre als IT Professional
- unterschiedliche Umgebungen:
  - 24x7 Operating im medizinischen Bereich
  - Support, Training, Implementierung
  - IT Leiter bei einer NPO
  - Senior Consultant Systemmanagement
- 15 Jahre Open Source Erfahrung
- enger Kontakt zur Community
  (Vorstandsmitglied der Linux User Group Austria)

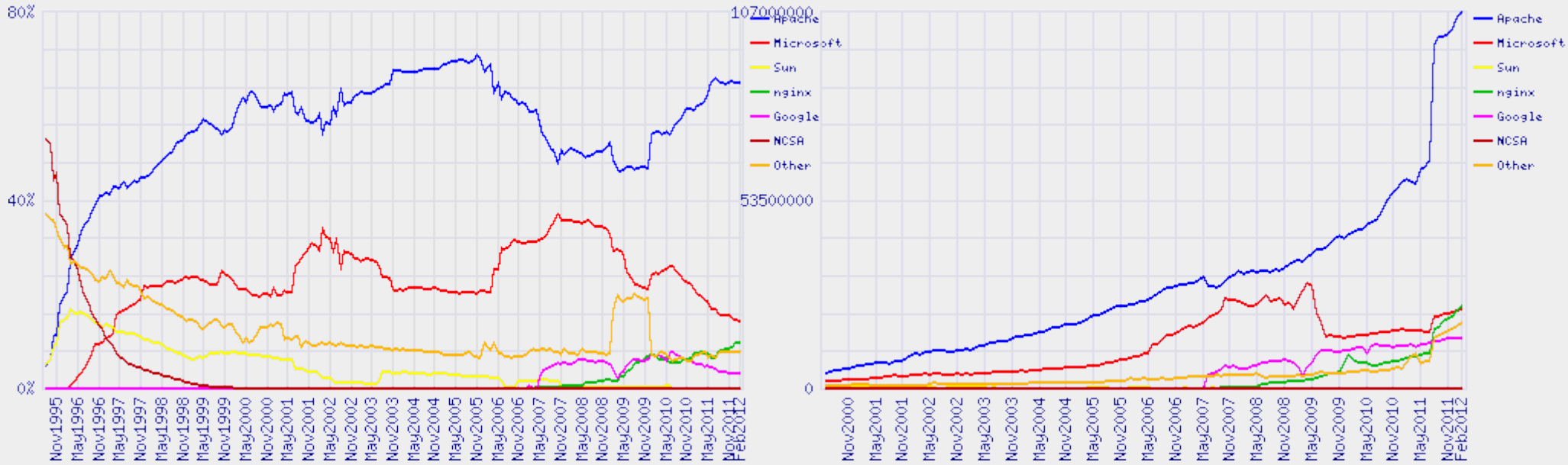
Erfahrungen kann man nicht lernen, man muss sie machen

OpenSource :: Philosophie

Free Software Definition:

- Die Freiheit, **das Programm für jeden Zweck zu benutzen** (Freiheit 0).
- Die Freiheit, zu verstehen, wie das Programm funktioniert und wie man es **für seine Ansprüche anpassen** kann (Freiheit 1). Der Zugang zum Quellcode ist dafür Voraussetzung.
- Die Freiheit, **Kopien weiter zu verbreiten**, so dass man seinem Nächsten weiterhelfen kann (Freiheit 2).
- Die Freiheit, das Programm zu verbessern und die **Verbesserungen der Öffentlichkeit zur Verfügung zu stellen**, damit die ganze Gemeinschaft davon profitieren kann (Freiheit 3). Der Zugang zum Quellcode ist dafür Voraussetzung.

Apache :: Relevanz



Quelle: http://www.netcraft.com, Stand: 7.2.2012

Apache 2.0

„The original reason for creating Apache 2.0 was scalability, and the first solution was a hybrid web server; one that has both processes and threads. This solution provides the reliability that comes with not having everything in one process, combined with the scalability that threads provide.“

Quelle: Apacheweek issue 137, 24. Sep. 1999
http://www.apacheweek.com/issues/99-09-24

Apache 2.0 released: 6.4.2002

Apache 2.0 :: Vorteile

Die wichtigsten Neuerungen in Apache 2.0 sind unter anderem:

- Threading
  - verbesserte Skalierbarkeit
- vereinfachte Konfiguration
  - `Listen` ersetzt `BindAddress` und `Port`
  - virtuelle Server werden durch `ServerName` ausgewählt
  - Wildwuchs an Direktiven reduziert
- überarbeitete Module, zB:
  - `mod_auth_ldap`
  - `mod_proxy`
  - `mod_headers`
- IPv6 Support

Apache :: MPM

MPM prefork

- neue Prozesse für neue Requests (Fork)
- hält einen Pool an Prozessen vorrätig (`MinSpareServers`)
- saubere Trennung der Requests

MPM worker

- neue Threads für neue Requests
- Mischung aus Forks und Threads (`ThreadsPerChild`)
- geringerer Speicherverbrauch

Forking vs. Threading

Forking

- neuer Prozess ist Kopie des Parents
- eigene PID, Filehandles, ...
- kostet Zeit und RAM
- Parent/Child sind komplett getrennt
- IPC schwierig/teuer
- „verlorene“ Kinder werden Zombies

Threading

- neuer Thread entsteht im alten Prozess
- wenig Overhead durch gemeinsames Datensegment, FD, ...
- schnell (vor allem bei multicore Hardware)
- einfache „IPC“
- Vorsicht bei Zugriff auf gemeinsame Daten!

Apache :: Legacy

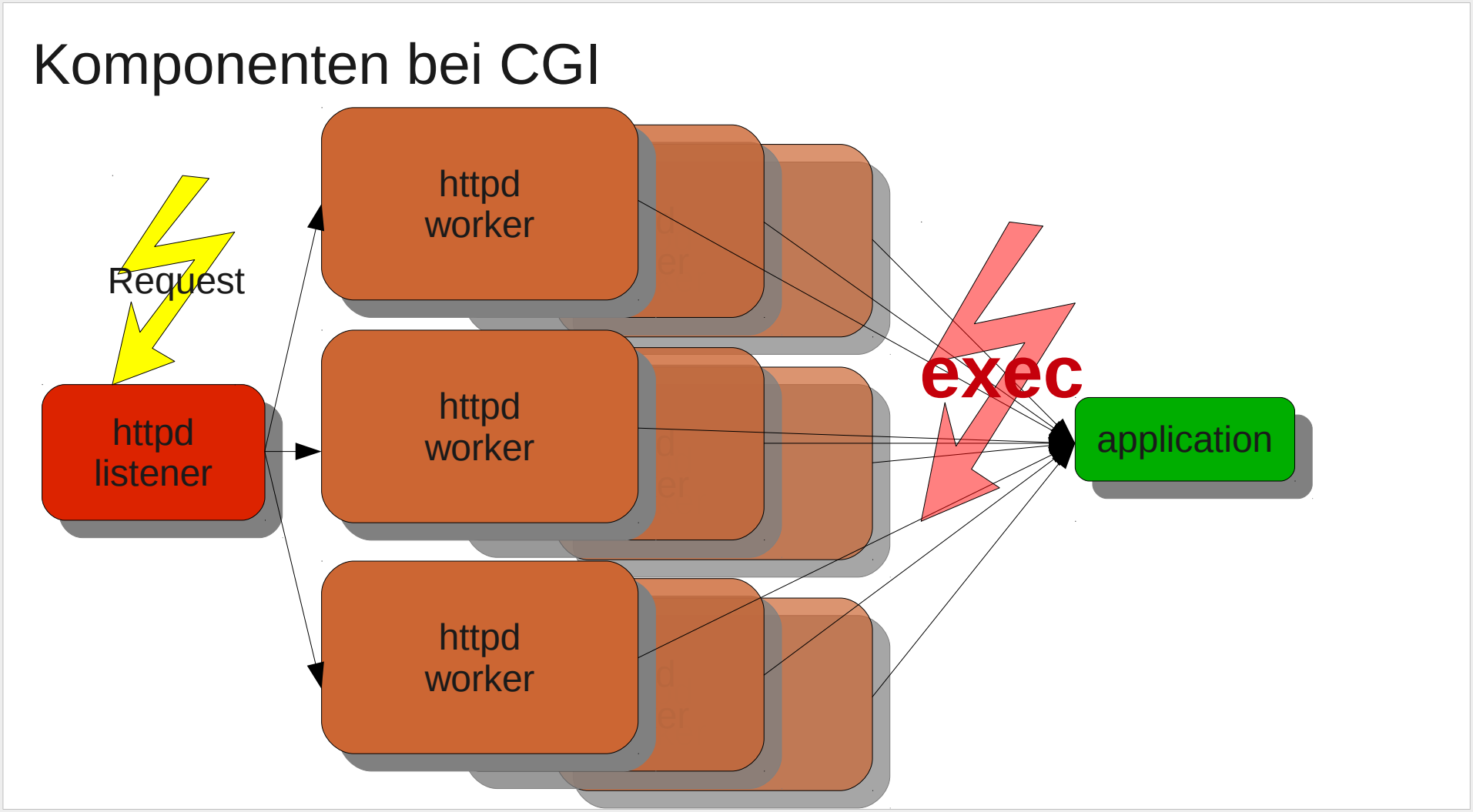
Warum laufen die meisten Apache Server mit dem Prefork Modul?

- Sicherheit
  - „Threading in Apache ist zu neu“ ;-)
  - „Apache 1.3 ist stabil und gründlich getestet“ :-P
- PHP
  - PHP (oder zugrunde liegende Bibliotheken) ist nicht „threadsafe“
  - PHP liegt der überwiegenden Zahl der Webanwendungen zugrunde.
    Suche bei freecode.com nach „CMS“: PHP: 209, Java: 27, C: 7

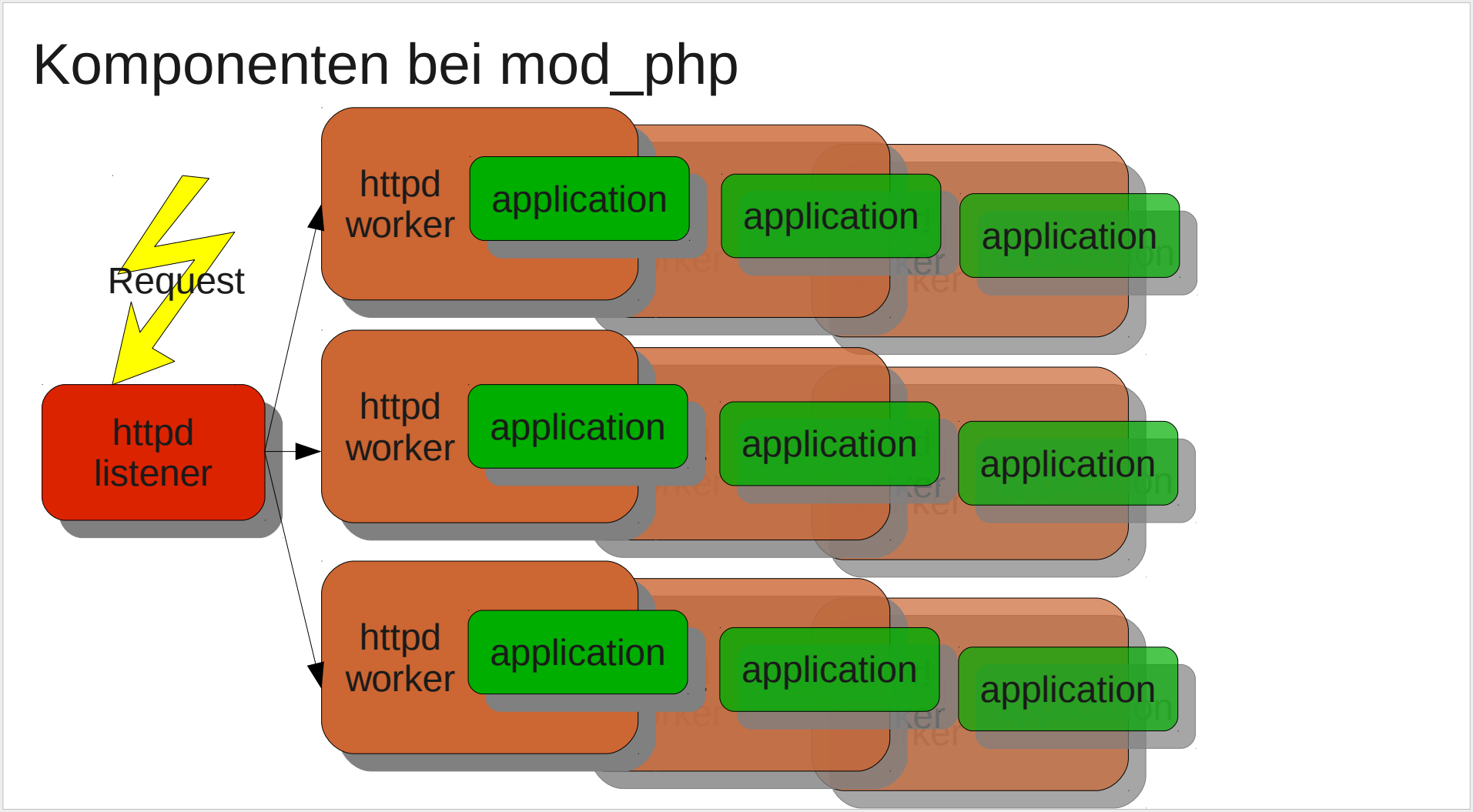
Apache erweitern

- CGI
  - startet externes Programm bei jedem Aufruf
  - einfach, aber langsam
  - kann mittels `chroot` und `suEXEC` eingeschränkt werden
- Apache-Module
  - sind in den Server integriert
  - schnell
  - verbrauchen Speicher in jeder Serverinstanz
  - laufen unter den Rechten des Apache Users
- FastCGI (FCGI)
  - verbindet Vorteile beider Ansätze

Apace-CGI :: Schema



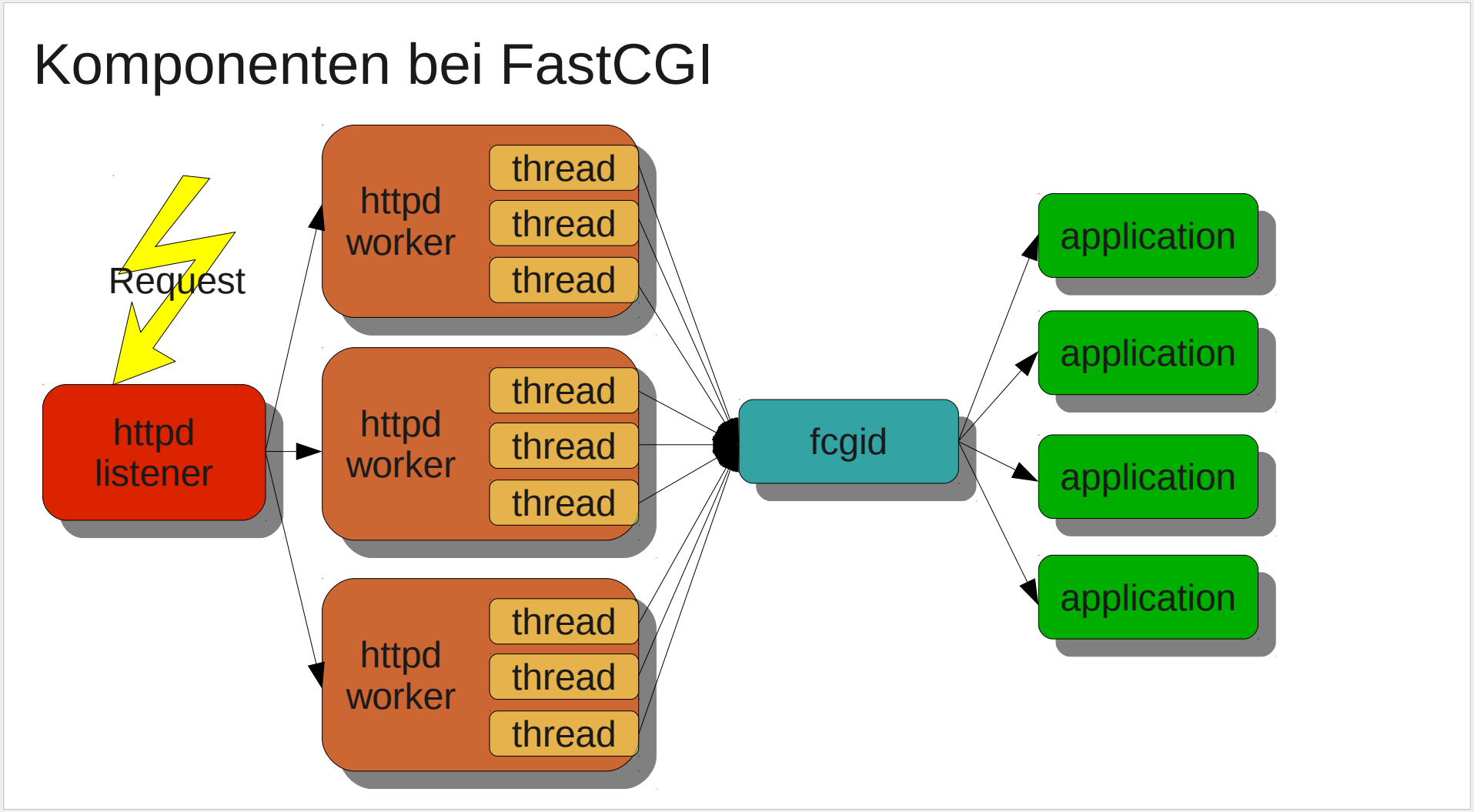
Apace-Modul :: Schema



FastCGI :: Grundlagen

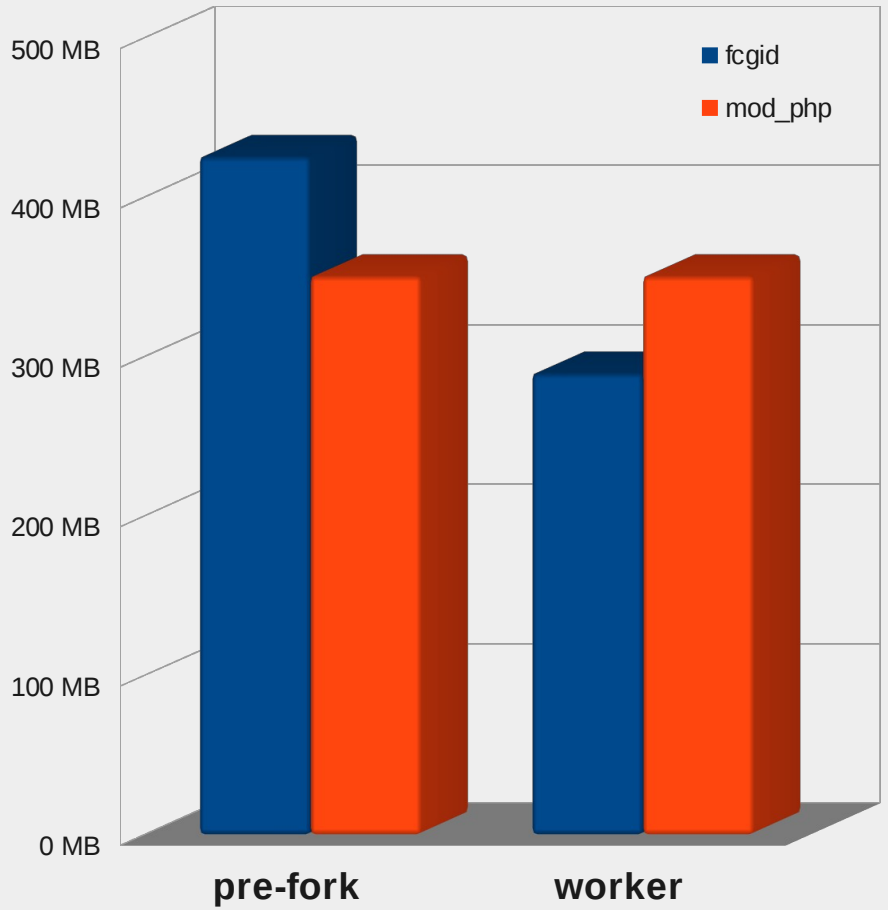
- Apache-Modul `mod_fcgid` (nicht `mod_fastcgi`!)
- `fcgid` läuft als Daemon und reicht Anfragen an „Scripts“ weiter
  - schneller als CGI
  - sicherer als integriertes Modul
- `fcgid` erhält Anfragen vom Apache via Socket – auch IP! → kann auch ein (oder mehrere) anderer Host sein
- Zahl der `fcgid` Prozesse konfigurierbar, je nach erwarteter Last

FastCGI :: Schema

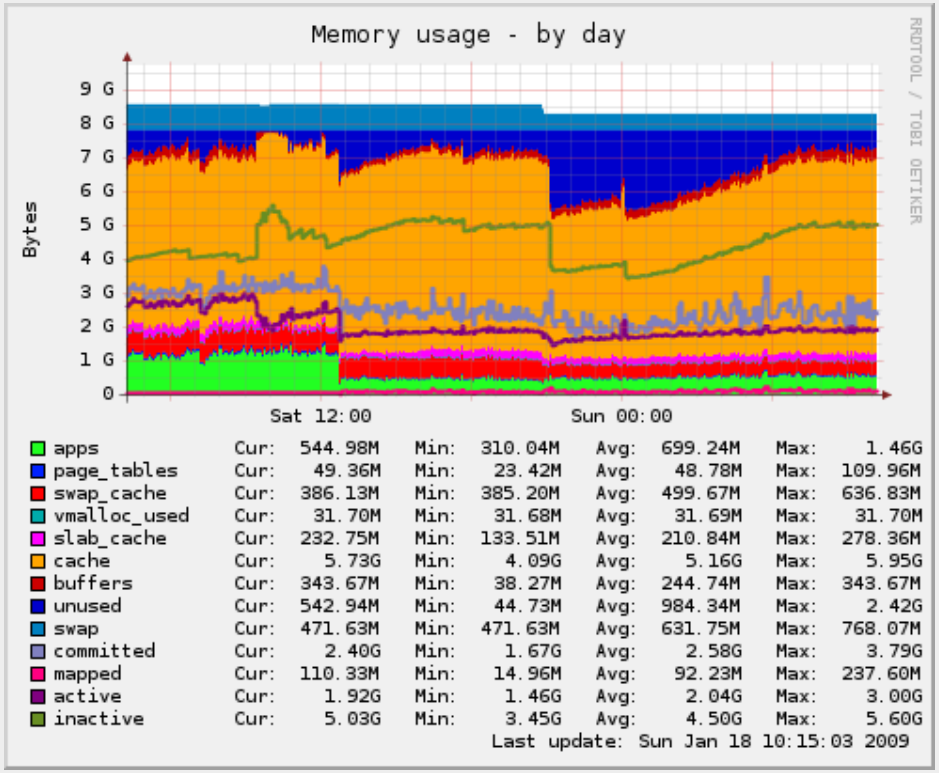


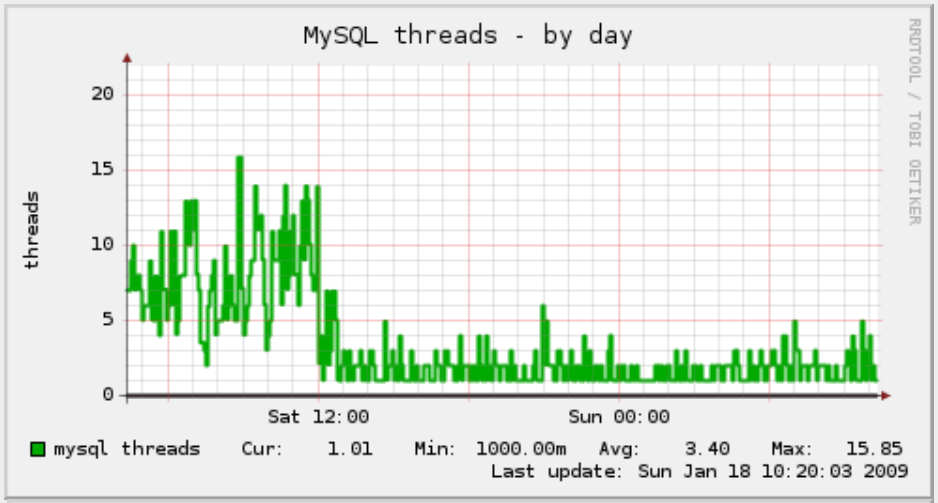
FastCGI :: Resourcenverbrauch

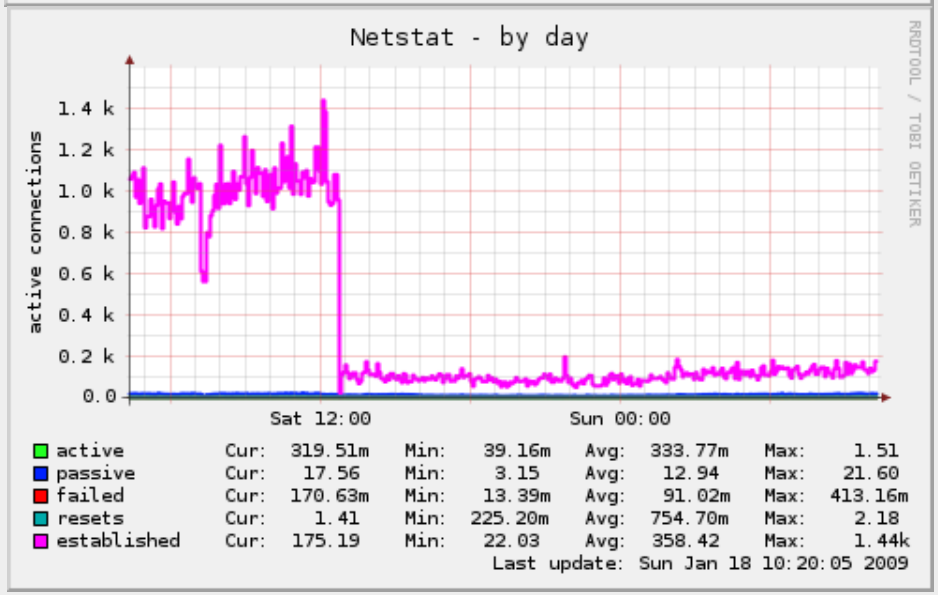
- Apache pre-fork:
  - `fcgi` 15 processes * (40 MB - 21 MB shared) + 21 MB shared + (20 Apache procs * 6 MB) = **426 MB**
  - `mod_php` 26 processes * (26 MB - 13 MB shared) + 13 MB shared = **351 MB**
- Apache threaded:
  - `fcgi` 15 processes * (40 MB - 21 MB shared) + 21 MB shared + (10 Apache procs * 25 MB) = **290 MB**
  - `mod_php` 26 processes * (26 MB - 13 MB shared) + 13 MB shared = **351 MB**



PHP :: FastCGI :: Performance







Apache :: suEXEC

suEXEC erlaubt das Ausführen von CGI/SSI unter einem anderem Benutzer/Gruppe als der Webserver selbst.

Vorteile:

- bessere Trennung der Prozesse verschiedener Applikationen oder VHosts
- bessere Trennung der Applikationen im Dateisystem
- Dateisystemquota, ulimit pro VHost

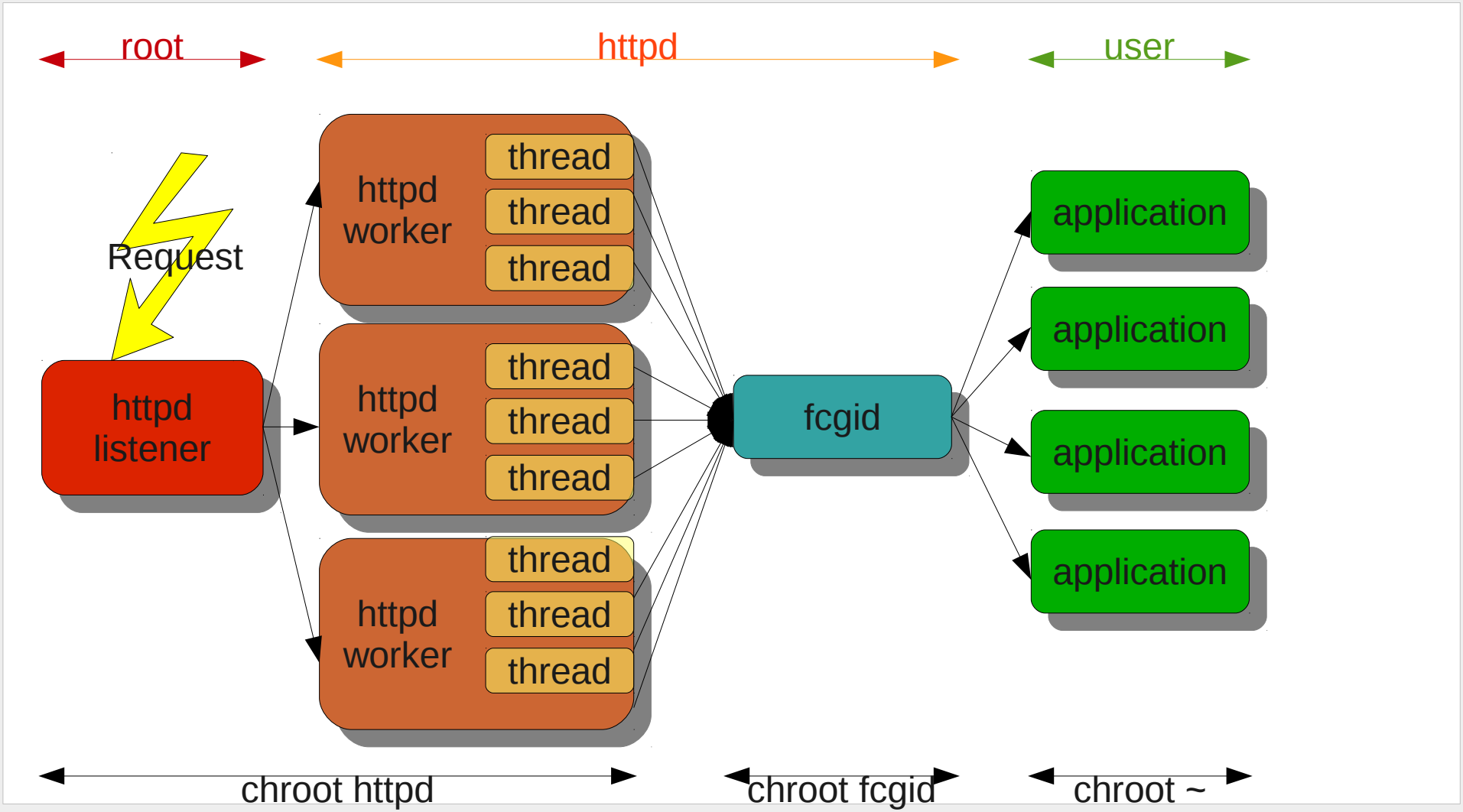
Nachteile:

- verwaltet durch einen setuid „Wrapper“
- bei falscher Konfiguration hohes Sicherheitsrisiko!

Chroot

- chroot Jail kann Sicherheit weiter erhöhen
  - Apache in chroot → Schutz des Hosts vor Apache Schwachstellen
  - fcgid in chroot → Schutz des Apache vor Fehler in fcgid
  - CGIs in chroot → Schutz vor Fehlern in CGI, aka „user contributed code“ =:-O

FastCGI :: suEXEC :: chroot



Apache :: FastCGI :: PHP

- PHP kann als Modul oder CGI kompiliert werden
- bei Distributionen muss CGI-Variante meist bewusst installiert werden:

```
apt-get -s install php5 apache2
The following extra packages will be installed:  apache2-mpm-prefork
libapache2-mod-php5 …

apt-get -s install php5-cgi apache2
The following extra packages will be installed:  apache2-mpm-worker php5-cgi
…
```

FastCGI :: PHP :: Konfiguration

- mod_fcgid:

```
<IfModule mod_fcgid.c>
  AddHandler    fcgid-script .fcgi
  FcgidConnectTimeout 20
</IfModule>
```

- VirtualHost:

```
<VirtualHost whatever.tld:80>
  ServerName   www.whatever.tld:80
  ServerAdmin  webmaster@whatever.tld
  DocumentRoot /var/www/whatever
  <Directory /var/www/whatever>
    AddHandler  fcgid-script .php
    FCGIWrapper /usr/lib/cgi-bin/php5 .php
    Options     ExecCGI …
...
  </Directory>
</VirtualHost>
```

Anwendungsbeispiele

- Server unter hoher Last (RAM Einsparung, Nutzung der Multicore CPUs)
- Anwendungen mit hohem Sicherheitsbedarf (suEXEC, chroot)
- Kundensysteme auf Webhost (Trennung der Applikationen, Quota, keine Rechteprobleme bei FTP Zugriff, ...)

Zusammenfassung

- geringer Mehraufwand in der Konfiguration
- Ressourcen schonend bei hohen Lasten
- verbesserte Trennung der Applikationen/Vhosts
- durch PHP-CGI mehrere PHP Versionen parallel möglich durch `FCGIWrapper`
- Userquota pro Applikation/Vhost
- chroot pro FCGIWrapper

Literatur

- Apache Doku
  http://httpd.apache.org/
- Howto (für Debian)
  http://www.howtoforge.com/how-to-set-up-apache2-with-mod_fcgid-and-php5-on-debian-etch
- Forking vs. Threading
  http://www.geekride.com/fork-forking-vs-threading-thread-linux-kernel/
- Apache fcgid Performance
  http://2bits.com/articles/apache-fcgid-acceptable-performance-and-better-resource-utilization.html