Webserver Performance Tuning

Jan Kneschke

jan.kneschke@incremental.de

incremental

Intro

Manche Webangebote werden erfolgreicher als ihre Schöpfer je eingeplant haben. Der Apache gibt sich die Kugel, das PHP wartet auf die Datenbank und von den 2 Gb RAM ist auch nicht mehr viel über.

Über den Referenten

- Dipl.-Ing. (FH) Jan Kneschke
- seit 2000 im PHP-Umfeld tätig (NetUSE AG)
- nach dem Abschluß des Studiums (Technische Informatik) Gründung der Firma incremental, die sich auf die Entwicklung von High-Performance Lösungen spezialisiert hat

Übersicht

- Grundszenario
- Problemstellen
  - Speicher
  - CPU
  - Netzwerk
- Meßverfahren und -werkzeuge
- Lösungsansätze

Grundszenario

Appliance Services

- Vorgaben
  - geringe Kosten
  - > 100 Kunden pro Rechner
  - hoher Datenbank Load
- Realisierung
  - eine CPU
  - 1 Gb RAM
  - Standard LAMP

Grundszenario

Hardware + Software

- Hardware
  - Intel P4 2 GHz
  - 1GByte RAM
  - RAID 10 - 40 Gbyte
  - 100BaseT
- Anbindung ans Internet
  - 34 Mbit/s

- Software
  - **L**inux 2.4.22
  - **A**pache 1.3.x (latest)
  - **M**ySQL 4.1.x (latest)
  - **P**HP 4.3.x (latest)

Das Problem

Slashdotting aka DDOS

- worst case design
- Wer Slashdot überlebt, überlebt auch einen DDOS
- viele parallele User zur gleichen Zeit
- lange Standzeiten der Verbindung

Das Problem

Die Folgen

- volle Ausnutzung der Bandbreite
- pro Connection besteht ein Apache Prozess
- jeder Apache Prozess benötigt im Schnitt 1 Mb RAM
- mod_php4 benötigt zusätzlich 3 Mb RAM pro Prozess
- 100 parallele User führen zu 400 Mb RAM Verbrauch, ohne daß etwas getan wurde
- viele parallel laufende Prozesse verbrennen unnötig CPU-Zeit (Overhead)

Speicher

“Mehr Speicher ist nur durch noch mehr Speicher zu ersetzen”

- man kann nie genug Speicher haben
- erst wenn man nicht mehr Speicher nachrüsten kann, lohnt die Suche nach Einsparpotenzialen im Speicherbereich
- Prozesse unter Linux ia32 können nur einen Adressraum von 2Gb verwalten
  - Threads laufen in einem Adressraum (MySQL)

Speicher

Einsparmöglichkeiten

- httpd.conf

  deaktivieren aller unnötigen Module wie z.B.
  mod_proxy
  mod_cgi
  mod_negotiation
  mod_include

- PHP

  nur die notwendigen Extensions

```
./configure --disable-all \
  --enable-...
```

Speicher

kein Apache

- Apaches Modell: Pro Connection ein Worker Prozess erzeugt das Speicherproblem
- single-process Webserver behandeln alle Verbindungen in einem Prozess
- auch 10.000 parallele Verbindungen lassen sich so mit 32 Mb RAM behandeln
- thttpd, lighttpd, zeus
- über das FastCGI Interface wird das PHP angebunden

CPU

Caching

- Caching der Ausgabe
- bei einem Cache-Hit wird direkt der Cache-Inhalt ausgegeben
- PEAR::Cache_Lite, PEAR::Cache

CPU

Caching - Beispiel

```
if (file_exists(‘‘cache.html’’) &&
    file_mtime(‘‘cache.html’’) >
    file_mtime(‘‘comments.txt’’)) {
  readfile(‘‘cache.html’’);
  exit(0);
} else {
  # generate output,
  # write output to cache.html,
  # send output to the browser
}
```

CPU

Caching - Vor- / Nachteile

- Nachteile
  - Caching erfordert ein Konzept
  - Abhängigkeiten-Modellierung kann sehr aufwendig sein
- Vorteile
  - Caching erzwingt ein Konzept
  - drastische Reduzierung der CPU-Last

Netzwerk

Caching - HTTP

- Timestamp
  Last-Modified, If-Modified-Since
- Entity Tags
  ETag, If-None-Match

Netzwerk

Caching - Beispiel

- initialer Request

```
HEAD / HTTP/1.0

HTTP/1.0 200 OK
Content-Length: 4348
Last-Modified: \
  Mon, 12 May 2003 12:54:17 GMT
[...]
```

Netzwerk

Caching - Beispiel

- folgender Request

```
HEAD / HTTP/1.0
If-Modified-Since: \
  Mon, 12 May 2003 12:54:17 GMT

HTTP/1.0 304 Not Modified
Content-Length: 0
[...]
```

Netzwerk

Output-Compression

- HTTP - Accept-Encoding, Content-Encoding
- alle Browser unterstützen Content-Encoding
- reduziert Traffic bis um den Faktor 10

Netzwerk

Output-Compression - Beispiel

```
HEAD / HTTP/1.0
Accept-Encoding: deflate

HTTP/1.0 200 OK
Content-Length: 1288
Content-Encoding: deflate
[...]
```

Netzwerk

Output-Compression - Aktivierung

- Apache - mod_gzip
- PHP - zlib.output_compression = “On”

Netzwerk

Output-Compression - Vor- / Nachteile

- Vorteile
  - drastische Reduzierung des Traffics
  - schnellere Beantwortung der Requests
  - höhere Gesamtperformance
- Nachteile
  - bei PHP ohne Caches merkbar höhere CPU-Last

Analyse

Meßwerkzeuge

- Load-Generatoren
  ab, httpperf, http_load
- Netzwerk
  mrtg, slurm
- Speicher
  vmstat, top, cat /proc/<pid>/status
- CPU
  vmstat, top

Analyse

Load-Generatoren

- belasten einen Server mit möglichst vielen Request
- bestimmen die Peak Leistung
- 4 Messungen sind notwendig
  lokal und über das äußere Netz
  mit geringem und mit hohem parallel Load

Analyse

Apache vs. lighttpd - static

```
$ ab -n 10000 -c <concurrent> \
  http://192.168.2.10:<port>/testfile100k
```

Requests/s

Server	50	500
Apache	697.25	544.07
lighttpd	3277.61	2047.50

- Peak-Durchsatz: 606 MByte/s bei -c 8 und -k

Analyse

ApacheBench (ab)

```
[...]
Concurrency Level:      8
Time taken for tests:   0.177 seconds
Complete requests:      1000
Failed requests:        0
Broken pipe errors:     0
Keep-Alive requests:    1000
Total transferred:      102676000 bytes
Requests per second:    5649.72 [#/sec] (mean)
Time per request:       1.42 [ms] (mean)
Time per request:       0.18 [ms] (mean, across all ... )
Transfer rate:          580090.40 [Kbytes/sec] received
```

Analyse

Apache + mod_php4 vs. lighttpd + FastCGI - PHP

```
$ ab -n 10000 -c <concurrent> \
  http://192.168.2.10:<port>/cgi.php
```

Server	50	500
Apache	969.93	779.42
lighttpd	1376.65	1319.96

- cgi.php: <?php print “12345”; ?>
- Untersuchung der Startup-Time, wichtig für Cache-Hits
- turckmm-cache 2.4.3 aktiv

Analyse

Speicherbedarf

```
$ cat /proc/<pid>/status
VmSize:     8024 kB  # auch in top
VmLck:         0 kB
VmRSS:      3384 kB  # auch in top
VmData:      812 kB
VmStk:        40 kB
VmExe:       220 kB
VmLib:      4792 kB
```

Analyse

Speicherbedarf

- swapoff -a
- rcapache stop
- top

```
182096 KBytes used with 100 apaches
138935 KBytes used after killing apache
 43160 KBytes difference

   430 kbyte for each apache process
```

Zusammenfassung

Design-Richtlinie

- Das Netz ist der Flaschenhals

Folgen der gezeigten Ansätze

- Reduzierung der Netzlast
- Optimierung des Durchsatzes
- Deutliche Erhöhung der Anfragen pro Sekunde
- Absicherung der Erreichbarkeit bei Slashdot-Announcements