

Ernesto Baschny

# Schnell, schneller, am schnellsten...

## TYPO3 Performance für jedermann

Nicht nur der Powersurfer erkennt sofort, wenn eine Website mehrere Sekunden zum Laden braucht. Die Laune geht sofort runter, das interaktive Erlebnis wird zur Qual, die Produktivität sinkt. Betreibern von TYPO3-Websites stehen jedoch einfache Mittel zur Verfügung, um mehr Performance aus einer TYPO3-Installation herauszuholen und diesen Fall zu vermeiden.

Ein TYPO3-Auftritt besteht immer aus mehreren Schichten: Das Design wird als XHTML/CSS umgesetzt, dieses wiederum unter TYPO3 mit Hilfe von TypoScript und TYPO3-Erweiterungen realisiert. Die TYPO3-Umgebung läuft auf einem Webserver, wo PHP mit MySQL und dem darunter liegenden Betriebssystem (Dateisystem) kommuniziert. Zu guter Letzt kommuniziert das Betriebssystem mit der Hardware und mit den Netzwerkschnittstellen, die letztendlich im Internet über Leitungen erreichbar sind. Diesen Weg gilt es, tausendfach pro Sekunde abzuwickeln, um jeden Besucher glücklich zu machen.

### TYPO3-Caching – das A und O

Auf TYPO3-Ebene passiert die meiste „Magie“, die der Administrator direkt beeinflussen kann. Auch wenn man die Servereinstellungen nicht direkt unter Kontrolle hat (z. B. bei Shared-Hosting-Paketen), ist eine intelligente Umsetzung bereits die halbe Miete für eine performante Webseite. Die wichtigsten Tipps:

- Schlankes XHTML ausgeben
- CSS statisch auslagern
- TYPO3-Caching nutzen
- USER\_INT nur gezielt einsetzen
- Erweiterungen mit „no\_cache()“ vermeiden
- Nicht verwendete Erweiterungen deinstallieren

Inhalt und Design sollten prinzipiell getrennt werden. Der Inhalt sollte als vom Design unabhängiges XHTML ausgegeben werden und das Styling möglichst komplett in statische CSS-Dateien ausgelagert werden. Der Browser muss diese Anweisungen nur einmal laden und kann diese dann auf die schlanke XHTML-Struktur auf jeder besuchten Seite des Webauftritts anwenden. Das spart Bandbreite (und damit auch Kosten für den Betreiber) sowie Server-Last und führt auch bei Benutzern mit langsameren Datenleitungen zu einem flüssigen Aufbau der Seiten.

Auf der TYPO3-Schicht schlägt das Herz der Website: Bei jeder Anfrage entscheidet sich hier, was der Server tun muss, um die Anfrage beantworten zu können. Um ein genaues Bild des Rendering-Prozesses der Seite zu bekommen und Engpässe zu finden, ist das Admin-Panel ein sehr hilfreiches Tool. Aktiviert wird es im TypoScript-Haupttemplate mit der Zeile „config.admPanel = 1“. Für Nicht-Admin-User muss der Administrator das Admin-Panel im UserTS gezielt aktivieren:

```
UserTS (bei Gruppe oder User)
admPanel {
    enable.tsdebug = 1
    enable.info = 1
}
```

Listing 1

Sofern man im Backend eingeloggt ist, ist das Admin-Panel nun sichtbar. Alle Preview- und Edit-Optionen sollten deaktiviert wer-

den, im „TypoScript“-Debugging und „Info“-Reiter können die interessanten Auswertungen bewertet werden.

./Start Template	0
./Get Page from cache	2
./Get Page from cache/Cache Row	0
./Page from cache/Cache Row/Cache Query	1
./Page generation	0
./Print Content	0
./Stat	0
Info	
id:	1
type:	0
no_cache:	0,1
fe_user, frame:	1
fe_user, uid:	
Total parse time:	39 ms

Eine Seite komplett aus dem Cache, im Admin-Panel: „no\_cache=0“ und keine „Non-Cached objects“.

Im Vergleich zur gecachten Ausgabe muss TYPO3 viel mehr bei jedem Aufruf tun, wenn „no\_cache“ gesetzt ist. Auslieferungszeiten von über drei Sekunden sind keine Seltenheit.

./Get Page from cache	2
./Get Page from cache/Cache Row	1
./Page from cache/Cache Row/Cache Query	0
./Parse template	5
./Setting the config-array	11
./Page generation/initializing	26
./Non-cached objects	0
./Non-cached objects/Include libraries	4
./Non-cached objects/Split content	0
./Ship	7
./User, uid:	
./User, uid:	
Total parse time:	102 ms

Die Seite ist zwar gecacht, jedoch muss ein USER\_INT-Plugin geladen werden. Die Serverlast geht bereits deutlich nach oben.

ist selbst ein USER\_INT zu vermeiden.

./Get Page from cache	0
./Parse template	68
./Parse template/substitute Constants (190)	1
./Setting the config-array	0
./Page generation	28
./Page generation/initializing	19
./Page generation/rendering	0
./Page generation/Rendering	43
./substituteMarkerArray	1
./Page generation/Index page	1
type:	0
no_cache:	1
fe_user, frame:	1
fe_user, uid:	
Total parse time:	277 ms

Im Vergleich zur gecachten Ausgabe muss TYPO3 viel mehr bei jedem Aufruf tun, wenn „no\_cache“ gesetzt ist. Auslieferungszeiten von über drei Sekunden sind keine Seltenheit.

Katastrophal verhält sich das Rendering, wenn das TYPO3-Caching komplett umgangen wird. Immer noch sind viele Erweiterungen im Umlauf (und sogar neu programmierte), die durch „\$this->no\_cache()“ den Cache für eine Seite komplett deaktivieren. Der Erweiterungsentwickler hat es so einfacher – der Seitenbetreiber jedoch deutlich schwerer. Das Caching einer Seite kann auch via TypoScript (config.no\_cache=1) oder durch Übergabe eines GET-Parameters (&no\_cache=1) deaktiviert werden. Es gilt somit, diese Fälle zu erkennen, sie zu analysieren und die Erweiterung passend umzuschreiben oder umschreiben zu lassen. Viele Erweiterungen nutzen die USER\_INT-Methode, ohne dass explizit ein Plugin auf einer Seite eingebunden ist (lediglich durch Installieren der Erweiterung). Somit ist oft nur durch den TypoScript-Object-Browser oder das Admin-Panel erkennbar, dass ein solches Plugin aufgerufen wird. Erweiterungen, die nicht gebraucht werden, sollten auf jeden Fall deinstalliert werden, um unnötige Ladezeiten zu vermeiden.

## Apache und MySQL

In der Regel werden TYPO3-Websites auf einer LAMP-Umgebung (Linux/Apache/MySQL/PHP) betrieben. Falls Root-Zugriff auf die Umgebung des Webservers vorhanden ist und der Provider keine TYPO3-Expertise oder Tuning-Dienstleistungen anbieten kann, gilt es, zumindest rudimentär das Zusammenspiel von MySQL, PHP und Apache auf die vorhandene Hardware abzustimmen.

Grundsätzlich sollte der Server ausschließlich für den Webserver-Betrieb genutzt werden. E-Mail-Verwaltung, Antivirus, Antispam und sonstige Server-Dienste gehören nicht auf den TYPO3-Server. Diese Dienste erhöhen lediglich die Serverlast, erschweren das Tuning speziell für TYPO3 und bieten weitere Angriffsflächen, nicht nur für DoS-Attacken.

Der verfügbare RAM des Servers spielt beim Tuning eine bedeutende Rolle. MySQL und Apache müssen sich den vorhandenen RAM teilen. Im Hinterkopf sollte stets behalten werden, dass für den Regelbetrieb diese RAM-Grenze nicht überschritten wird: Denn falls das passiert, muss der Server „swappen“, also Daten aus dem RAM auf die Festplatte auslagern und später von dort wieder zurückholen, was sich oft schnell hochschauzelt und den Server komplett in die Knie zwingt.

Apache wird in der Regel mit „prefork“ konfiguriert, was mit PHP als Modul am einfachsten von der Hand geht. Jeder Apache-Prozess belegt eigenen Speicher. Zu Bedenken ist, dass jeder Prozess nur einen Client gleichzeitig bedienen kann. Bei der Einstellung „MaxClients“ sollte man dies für die Peak-Zeit berücksichtigen, wobei natürlich auch die durchschnittliche Auslieferungzeit und der verfügbare RAM eine Rolle spielen.

### Apache prefork Tunings

```
<IfModule prefork.c>
StartServers 10
MinSpareServers 12
MaxSpareServers 20
MaxClients 20
MaxRequestsPerChild 20
</IfModule>
# Um weniger clients im "keep alive" Zustand zu behalten:
KeepAliveTimeout 5
```

Listing 2

Bei Apache sollte „MaxClients“ passend konfiguriert werden, eine Gratwanderung zwischen „Ressourcen schonen“ und „alle Anfragen sofort beantworten können“. Pro Apache-Prozess inklusive PHP-Modul und Speicher für eine reguläre TYPO3-Auslieferung müssen mindestens 15 MB RAM gezählt werden. Somit sind bei „MaxClients 20“ in einem „Peak“, bei dem alle Clients eine TYPO3-Seite aufrufen, bereits 300 MB belegt. Bei 1 GB RAM kein Problem, bei einer kleinen virtuellen Maschine mit 256 MB RAM je nach Auslastung bereits zu viel.

Nun gilt es, sich MySQL zuzuwenden: Das MySQL-Tuning sollte einmalig vor dem Live-Gang grob erarbeitet werden. Eine Woche nach der Inbetriebnahme sollte es jedoch nochmals überarbeitet werden: Nur dann hat MySQL notwendige Statistiken aus dem Betrieb sammeln können, die geschickterweise für das Tuning verwertet werden können. Im Idealfall sollte man dieses Tuning von Zeit zu Zeit wiederholen, um eventuell auch potenziell notwendige Hardware-Upgrades rechtzeitig einleiten zu können.

Beim MySQL-Tuning gilt es ebenfalls, die RAM-Grenze zu berücksichtigen, wobei der bereits durch Apache belegte Teil und der Grund-RAM des Betriebssystems abgezogen werden müssen. Die Einstellung „max\_connections“ ist ausschlaggebend für die Gesamtmenge an RAM, die MySQL maximal belegen wird. Diese sollte im Einklang mit „MaxClients“ von Apache stehen.

### MySQL-Tuning, mv.cnf

```
[mysqld]
long_query_time = 2
# Etwas über MaxClients von Apache
max_connections = 30
key_buffer_size = 16M
table_cache = 1024
query_cache_size = 32M
sort_buffer_size = 4M
```

Listing 3

MySQL-Einstellungen sollten je nach TYPO3-Umgebung angepasst und insbesondere der Query-Cache höher gesetzt werden. Der Einsatz eines Auswertungsskripts [1] erleichtert das Tuning der vielen MySQL-Parameter. Damit „max\_connections“ nie überschritten werden, sollte PHP keine persistenten MySQL-Verbindungen aufbauen, zu erreichen durch die Zeile „mysql.allow\_persistent = Off“ in der „php.ini“ im Bereich „[MySQL]“.

## Skalierbarkeit

Falls der Server viele statische Dateien ausliefern muss, diverse virtuelle Server beherbergt oder Flexibilität wichtig ist, ist es eine gute Alternative, Apache „prefork“ durch „workers“ zu ersetzen. So wird auf „mod\_php“ verzichtet und stattdessen PHP als FastCGI betrieben: Schlankere Apache-Prozesse und per-VirtualHost-PHP-Umgebungen sind möglich [2]. Wenn tatsächlich die Grenzen der Skalierbarkeit eines Servers erreicht werden, ist stärkere Hardware die einfachste Lösung. Falls die Skalierbarkeit von vornherein berücksichtigt werden muss (um z. B. kurzzeitige Zugriffsspeaks nach dem Versand eines Massen-Newsletters abzufedern), sollte über den Einsatz von einem Proxy vor dem Webserver beziehungsweise einem Webserver-Cluster vor der Datenbank oder über andere Methoden wie Datei-basiertes Caching nachgedacht werden.

## Vertiefung

Das Tuning fängt mit den dargestellten kleinen Schritten erst richtig an. Wer eine kleine Seite betreibt, ist mit der Wahl des richtigen Providers (schnelle Internet-Anbindung, hochwertige Hardware, passendes Betriebssystem, guter TYPO3-Service) gut beraten und kann mit einfachen Mitteln seine Umgebung selbst tunen.

Wer professionelle Seiten betreiben möchte (hohe Zugriffszahl, Verfügbarkeitsgarantien etc.) sollte sich jedoch professionelle Hilfe holen und einen Provider favorisieren, der auf TYPO3-Hosting spezialisiert ist und Erfahrung in dem Bereich nachweisen kann. Ein TYPO3-Hosting-Paket oder ein Managed-TYPO3-Server sind in der Regel gut für den Standardeinsatz vorkonfiguriert. Der Provider ist in diesen Fällen gewöhnlicherweise auch TYPO3-Dienstleister und kann entsprechenden Service für das Tuning anbieten oder ein komplettes Performance-Konzept erarbeiten.

## Links und Literatur

☞ [Softlink 2312](#)

- [1] Tuning-Primer-Skript: <http://www.day32.com/MySQL/>
- [2] Using PHP with mod\_fcgid: <http://typo3.org/development/articles/using-php-with-mod-fcgid/>

### DER AUTOR



Ernesto Baschny ist Diplom-Informatiker, TYPO3-Core-Entwickler und Geschäftsführer bei cron IT. Langjährige Erfahrung im Betrieb von Webservern - im Speziellen auch für TYPO3 - erlangte er durch den Betrieb des Datacenters von cron IT in Frankfurt.