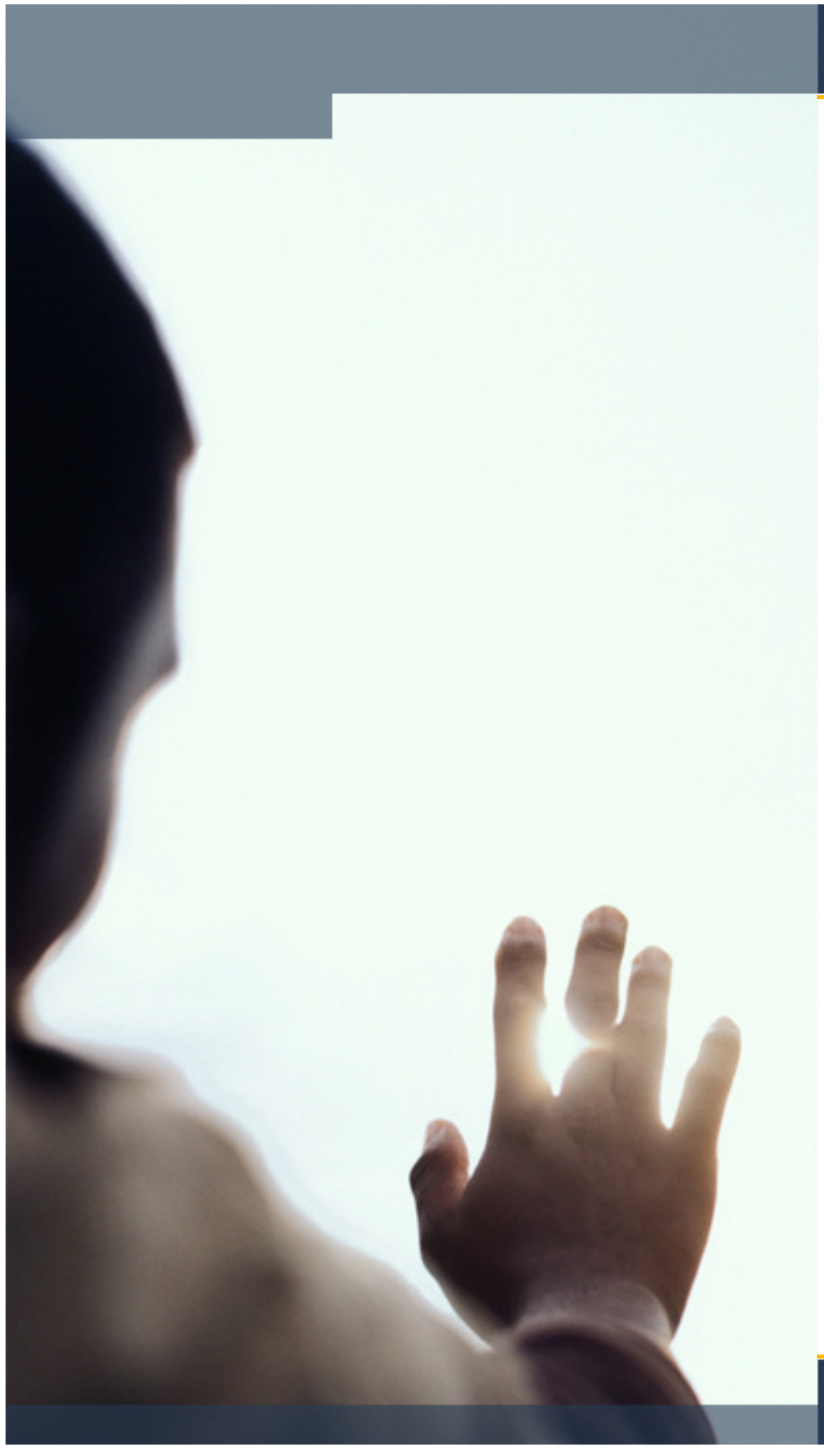


Surviving Slashdot

Wenn ein einfacher Webserver
einfach nicht mehr reicht

Volker Tanger
Senior Consultant System Security
HiSolutions AG, Berlin

Hisolutions

Agenda - Surviving Slashdot

Server down...

- Auslöser
- Technische Gründe
- Fehlerfall-Szenarien

Hochlast vorbereiten

- Architektur / Administration
- Server-/Daemon-Optimierung
- Cacheing
- Load-Balancing
- HTML-Optimierung
- Anwendungsoptimierung

Was tun beim Kind im Brunnen?

- Statusaufnahme
- Notbetrieb / 1. Hilfe

Über die HiSolutions AG

Gründung	1994
Mitarbeiter	55
Firmensitz	Berlin
Beratungsfeld	Beratungs- und Lösungshaus für (IT-) Governance, Risk & Compliance
Leistungen	▪ Information Security ▪ Risk Consulting ▪ Business Continuity Management ▪ IT-Service Management ▪ Project Portfolio Management
Kunden	▪ 50% der DAX-30-Unternehmen ▪ 75% der deutschen TOP20-Banken ▪ Behörden (u.a. BMI, BSI)
Auszeichnungen	▪ Innovationspreis Berlin/Brandenburg ▪ Zweifacher Preisträger „Fast50 Germany“ von Deloitte ▪ Zweifacher Preisträger „ Fast500 Europe“ von Deloitte

Hisolutions

Agenda - Surviving Slashdot

Server down...

- Ausfall-Szenarien
- Auslöser
- Technische Gründe

Hochlast vorbereiten

- Architektur / Administration
- Server-/Daemon-Optimierung
- Cacheing
- Load-Balancing
- HTML-Optimierung
- Anwendungsoptimierung

Was tun beim Kind im Brunnen?

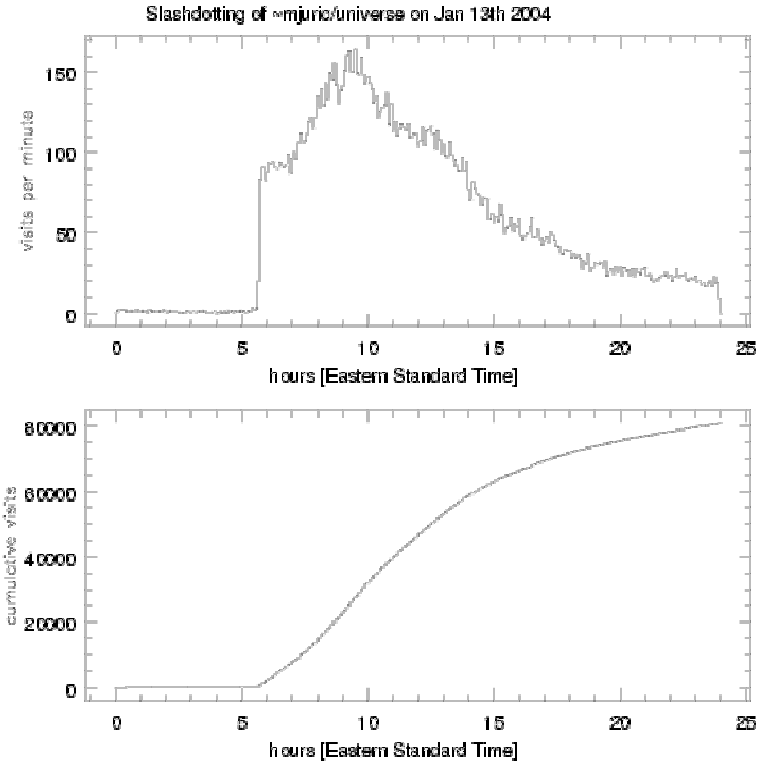
- Statusaufnahme
- Notbetrieb / 1. Hilfe

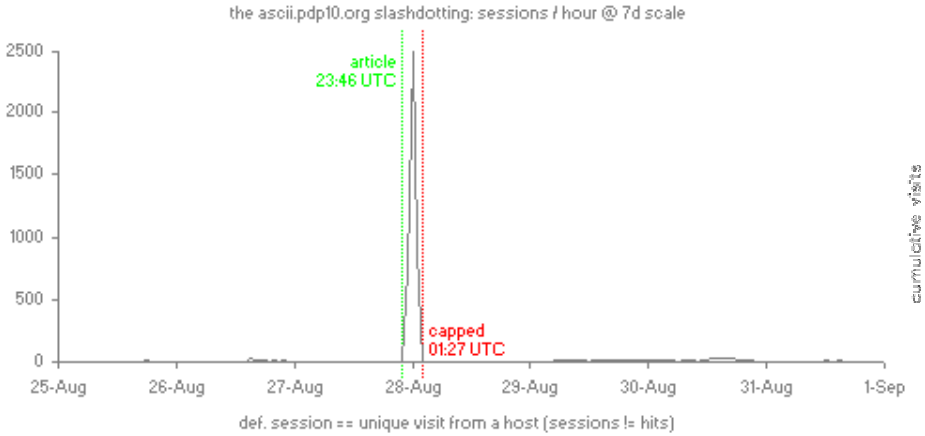
Ausfall-Szenarien

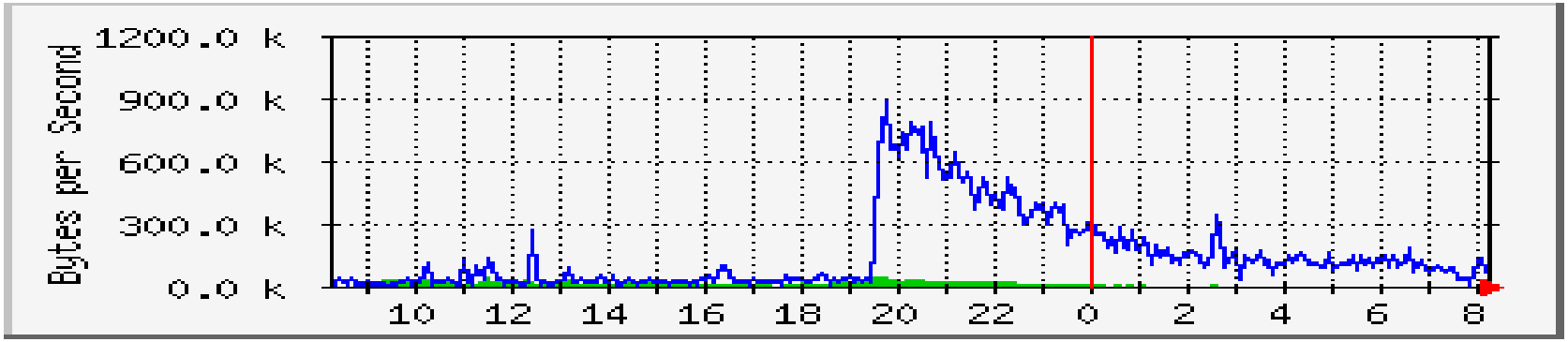
Server down...

Slashdot-Effekt / slashdotted

„geheist“ (von heise.de)







Ausfall-Szenarien

Server down...

Totalausfall

- Hoster schaltet Webhosting-Account ab
- alles auf einem Server, bei Überlast geht nichts mehr
- zentrale kritische Teile angreifbar (eBay DNS-Subnetz)

➔ **keine Information der Nutzer mehr möglich!**

Teilausfall

- Hoster drosselt Webhosting-Account
- Co-gehostete verbrauchen viele Ressourcen (shared Hosting, virtual Server)
- "nur" Shop / Download / CMS offline

kritisch:

- Mail / Infrastruktur gehen noch
- WWW (Hauptseite) geht noch, kann Nachricht zeigen

Hisolutions

Agenda - Surviving Slashdot

Server down...

- Ausfall-Szenarien
- Auslöser
- Technische Gründe

Hochlast vorbereiten

- Architektur / Administration
- Server-/Daemon-Optimierung
- Cacheing
- Load-Balancing
- HTML-Optimierung
- Anwendungsoptimierung

Was tun beim Kind im Brunnen?

- Statusaufnahme
- Notbetrieb / 1. Hilfe

Auslöser – wann/wieso es passiert

Server down...

unerwartet: DDoS, Slashdot-Effect

- (D)DoS-Angriff
- News/Blog: Slashdot (slashdotted), Heise (geheist),
- TV-Total, Spiegel-online, HiProfile-Blog, ...
- Event: Release (Film, CD, ...), Wahl (Partei), ...
- Crawler (rekursive URLs, Beispiel Fillshop)
- fehlendes Monitoring (gut genutzt ➔ am Anschlag ➔ Überlast)
- auch: fehlende Kommunikation (eigene Werbekampagne)

absehbar = planbar

- drohender Termin (Werbung, Release, Wahl, ...)
- beobachtete (langsame) Entwicklung
- einfach "nur" erfolgreiche/große Site (Facebook, Twitter, Slashdot, LiveJournal)

Hisolutions

Agenda - Surviving Slashdot

Server down...

- Ausfall-Szenarien
- Auslöser
- Technische Gründe

Hochlast vorbereiten

- Architektur / Administration
- Server-/Daemon-Optimierung
- Cacheing
- Load-Balancing
- HTML-Optimierung
- Anwendungsoptimierung

Was tun beim Kind im Brunnen?

- Statusaufnahme
- Notbetrieb / 1. Hilfe

Technische Gründe

Server down...

Langsame Anwendung / CPU-Überlast

(dadurch auch: längere Belegung von Ressourcen)

Server Ressource Exhaustion (Details später):

- Konfig-Limits (Default-Einstellungen Apache, MySQL)
- File-Handles (Zahl der Sockets)
- RAM

Netzwerk-Limits

- Anbindung: LAN Mbit/s, RZ-Anbindung (eBay-DNS)
- Aktionen Hosting-Firma:
  - automatische DoS-Abwehr
  - AGB-Limits (Kontingent, danach Drossel oder Abschaltung)

Hisolutions

Agenda - Surviving Slashdot

Server down...

- Ausfall-Szenarien
- Auslöser
- Technische Gründe

Hochlast vorbereiten

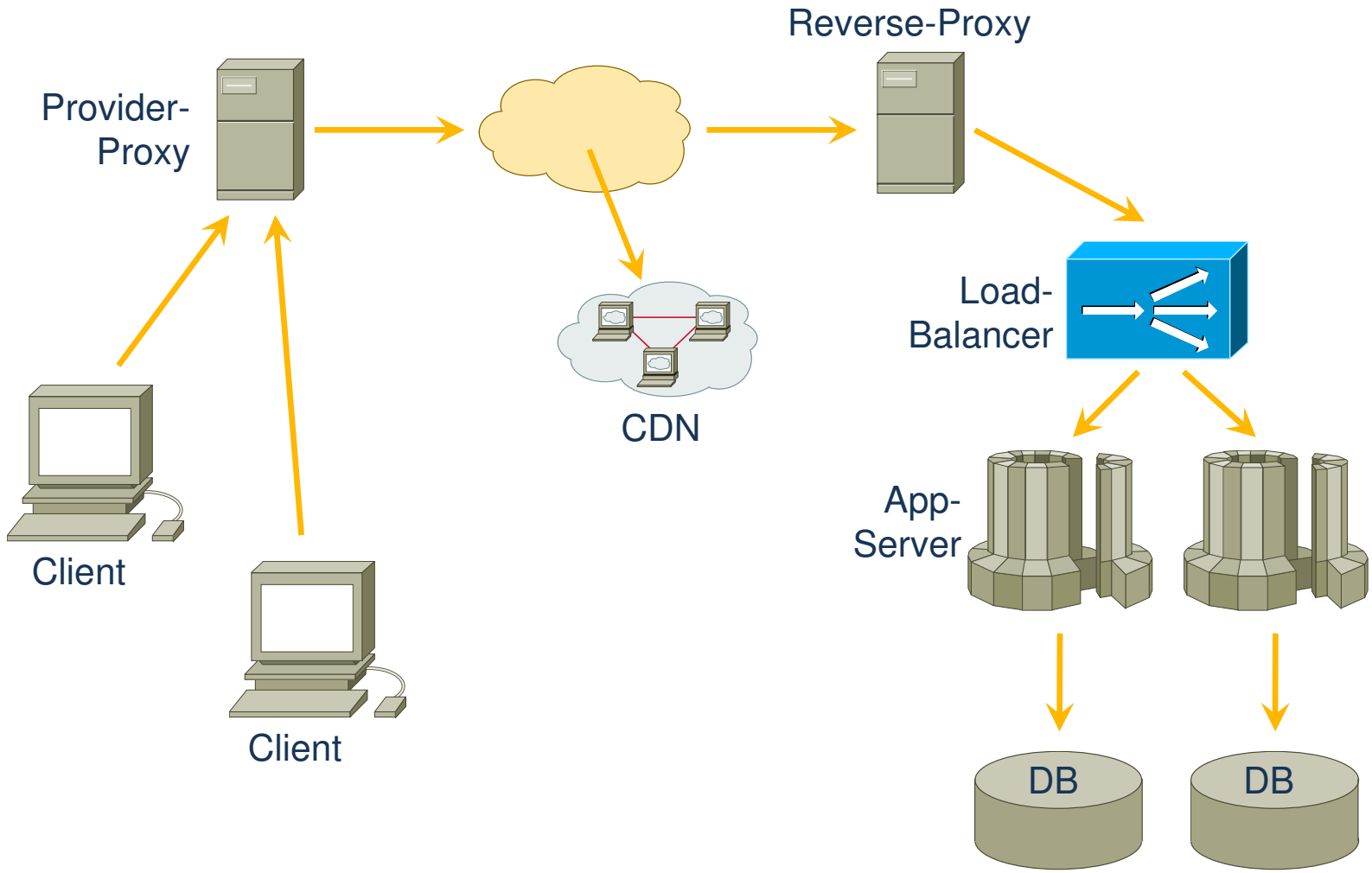
- Architektur / Administration
- Server-/Daemon-Optimierung
- Cacheing
- Load-Balancing
- HTML-Optimierung
- Anwendungsoptimierung

Was tun beim Kind im Brunnen?

- Statusaufnahme
- Notbetrieb / 1. Hilfe

Übersicht möglicher Beteiligter

Hochlast vorbereiten



Architektur

Hochlast vorbereiten

Verteilung der Bestandteile auf mehrere Maschinen

- statischer Server (z.B. Images)
- Application Server
- DB Server
- bei HTTPS: SSL Offloader
- CDN = Content Delivery Network z.B. akamai, Coral, MirrorDot, Google, ...
- über Reverse Proxy (➔ später...)

Hauptseite auf separaten Server / in separates Netz

- möglichst statisch
- vorbereitete, statische Notfallkopie (➔ später)
- lokal gecached (➔ später)
- Shop/Forum/Wiki/Medien etc. auf anderen "Opfer"-Server, auf anderes DNS (Um-/Abschalten, z.B. shop.wyae.de / streaming.wyae.de), in anderes Netz oder "kostenlose" Dienste (Notfall-Video auf YouTube)

Architektur / Administration

Hochlast vorbereiten

Dienst- und(!) Performance-Monitoring

Hochlast *vorher testen*

... und nachsehen, was der Engpass ist

Server mit reichlich Hardware-Reserve

aber: schlechte Ausnutzung / zu hohe Kosten

DNS mit kurzer TTL

dadurch schnellere Umschaltung im Falle eines Falles
(bei passenden FQDNs auch: Dienstumlenkung)

aber auch: schnellerer Ausfall bei DNS-Problemen

Hisolutions

Agenda - Surviving Slashdot

Server down...

- Ausfall-Szenarien
- Auslöser
- Technische Gründe

Hochlast vorbereiten

- Architektur / Administration
- Server-/Daemon-Optimierung
- Cacheing
- Load-Balancing
- HTML-Optimierung
- Anwendungsoptimierung

Was tun beim Kind im Brunnen?

- Statusaufnahme
- Notbetrieb / 1. Hilfe

Server-/Daemon-Optimierung

Hochlast vorbereiten

Failure-Handling:

entsprechende vorbereitete ***statische(!)*** 404 / 5xx Fehlerseiten für alle Server
insbesondere 5xx-er nicht dynamisch, um auch bei Hochlast zu funktionieren

Drosselung / throttle – generell / je IP

+ mehr "Platz" für mehr Clients
- längere Belegung von Sessions / Sockets

Kompression (gzip, compress, deflate, ...)

+ weniger Traffic
- braucht mehr CPU

http/1.0 vs. 1.1 - keep-alive

- 1.1 weniger aber längere Sessions als 1.0
- Browser machen per Default mehr 2-4x mehr parallele Sessions bei 1.0

Server-/Daemon-Optimierung

Hochlast vorbereiten

RAM

besser für Anwendungen oder Buffers nutzen? Austesten!

Server-Wahl

(apache, lighttpd, nginx, thttpd, fnord/gatling, ...)

- forking (unter Linux ähnlich „billig“ wie threading – unter Windows unterirdisch)
- threading
- select/poll/queue = fast gleichbleibende Daemon-Größe

➔ `http://www.kegel.com/c10k.html`

FastCGI / FCGI bzw. mod-PHP statt CGI

Interpreter bleibt im Speicher statt jedes Mal neu laden/init

optimierte Appliaction-Server

Programm und Daten bleiben immer im Speicher (LISP-Server wie z.B. AllegroServe)

Server-/Daemon-Optimierung

Hochlast vorbereiten

Linux – Server-Ressourcen

File-Handler/Sockets

```
/proc/sys/fs/file-max  204811                    (rauf)
/proc/sys/net/ipv4/inet_peer_maxttl  600         (runter!)
/proc/sys/vm/max_map_count 32768                 auf > 32000
```

Linux 2.4 zusätzlich

```
/proc/sys/kernel/threads-max
```

Datenbank-Ressourcen

remember: Filehandler/Sockets - min. 3 je Request (http-Request, PHP-File, DB-Connect)

max_connections :

maximale Zahl gleichzeitiger Connections (*„DB-Error: Too many connections“*)

default 150 = zu wenig

je nach RAM-Größe auf mehrere tausend hochsetzen (Proxy kann mehr)

Server-/Daemon-Optimierung

Hochlast vorbereiten

Apache Server-Settings

```
MaxRequestsPerChild 1000 (fork)   0        (thread)
MaxClients 150                             (besser nicht über 256)
MinSpareServers 5 + MaxSpareServers 20
MinSpareThreads 25 + MaxSpareThreads 75
ThreadsPerChild 25
```

default forking:

20 servers x 1000 = 20tausend Requests

default threading:

20 servers x 20 Threads per Child x 75 = 30tausend Requests

Typisches Apache-Problem: zu wenig Laufzeit, zu oft Restarts

Hisolutions

Agenda - Surviving Slashdot

Server down...

- Ausfall-Szenarien
- Auslöser
- Technische Gründe

Hochlast vorbereiten

- Architektur / Administration
- Server-/Daemon-Optimierung
- Cacheing
- Load-Balancing
- HTML-Optimierung
- Anwendungsoptimierung

Was tun beim Kind im Brunnen?

- Statusaufnahme
- Notbetrieb / 1. Hilfe

Cacheing – Header

Hochlast vorbereiten

HTTP-Header

...damit Cacheing (auch auf dem Client) überhaupt funktioniert, müssen entsprechende Header (`Last-Modified` oder `Expires`) gesetzt sein. Bei falschen/fehlenden Headern wird nicht gecached – weder auf Client, noch auf zwischengeschalteten Proxies!

```
Last-Modified: Sun, 06 Dec 2009 23:00:01 GMT
Expires: Sun, 19 Nov 1978 05:00:00 GMT
```

Client-Request: `If-Modified-Since: Thu, 01 Nov 2009 21:17:22 GMT`

Server-Antwort `304 Not changed`

Weitere Header, die Cache-Bearbeitung steuern und mit denen u.a. zwischen Client- und Proxy-Cacheing unterschieden werden kann:

```
Cache-Control: public, max-age=0
Pragma: no-cache
```


Cacheing – Header

Hochlast vorbereiten

HTTP-Header auf Server setzen / modifizieren

Lighttpd

```
server.modules  += ( "mod_expire" )
$HTTP["url"] =~ "\.(jpg|png|gif|css|js|mp3|ogg|swf|zip)$" {
        expire.url = ( "" => "access 1 days" )
}
```

Apache

```
<FilesMatch "\.(ico|pdf|flv|jpg|jpeg|png|gif|swf|mp3|mp4)$">
        Header set Cache-Control "public"
        Header set Expires "Thu, 15 Apr 2010 20:00:00 GMT"
        Header unset Last-Modified
</FilesMatch>
```

Cacheing – Reverse Proxy

Hochlast vorbereiten

Reverse Proxy

...antwortet „wie ein HTTP-Server“, liefert schon bekannte Seiten aus lokalem SPeicher, fragt nur unbekannte auf HTTP-Server „hintendran“ ab.

Reverse-Proxy mit stale-if-error

- möglichst viele Abfragen schon vor dem Hauptserver abfangen
- Bei Wegbrechen des Hauptservers möglichst die bisherigen Webseiten ausliefern: **stale-if-error** (siehe http://www.mnot.net/blog/2007/12/12/stale und http://tools.ietf.org/html/draft-nottingham-http-stale-if-error-01)

Software

- **Squid** ab 2.7 - kann auch stale-while-revalidate
- **Varnish** stale-if-error kann *evtl* mit VCL nachprogrammiert werden, non-default
- **nginx** (+ncache): stale-if-error vielleicht/unbekannt, anscheinend empfindlicher auf Expiry-Settings
- **apache** + mod-proxy + mod-cache : stale-if-error vielleicht/unbekannt

Cacheing – Reverse Proxy: Squid3

Hochlast vorbereiten

```
# squid3 Konfiguration
# --------------------------------------------------------------------------
# hier die IP des Caches statt 444.333.222.11 eintragen!
http_port 444.333.222.111:80 accel defaultsite=WWW.PARENT1.DE vhost
# ggfs. Spool-Location und Size eintragen, hier: 5 GByte = 5000 MByte
cache_dir ufs /var/spool/squid3 5000 16 256

# ------------------------------
# die echten Webserver und was darauf laeuft
cache_peer 999.888.777.666 parent 80 0 no-query originserver name=PARENT1NAME
cache_peer_domain PARENT1NAME WWW.PARENT1.DE

# ------------------------------
# ACL - wir erlauben nichts anderes als zu den RevProxy-Seiten
acl alloweddoms .PARENT1.DE
acl alloweddoms .PARENT2.COM
http_access deny !alloweddoms

# default ACLs
acl Safe_methods method GET HEAD PUT POST CONNECT
http_access deny !Safe_methods
acl Safe_ports port 80 443
http_access deny !Safe_ports

# ------------------------------
# Logging - "none" = off
access_log /var/log/squid3.log
```

Cacheing – Reverse Proxy: Apache als Proxy

Hochlast vorbereiten

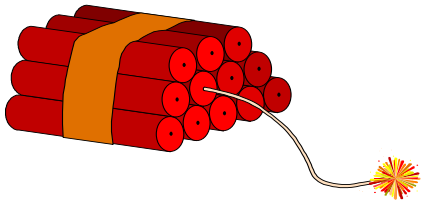
```
# apache mod-proxy + mod-cache
# ------------------------------------------------------------------------
LoadModule  proxy_module         modules/mod_proxy.so
LoadModule  proxy_http_module    modules/mod_proxy_http.so

ProxyRequests Off
<Proxy *>
  Order deny,allow
  Allow from all
</Proxy>
ProxyPass /foo http://foo.example.com/bar
ProxyPassReverse /foo http://foo.example.com/bar

# ------------------------------
LoadModule  cache_module       modules/mod_cache.so
<IfModule mod_cache.c>
  #LoadModule disk_cache_module modules/mod_disk_cache.so
  <IfModule mod_disk_cache.c>
    CacheRoot /var/spool/cacheroot
    CacheSize 256
    CacheEnable disk /
    CacheDirLevels 5
    CacheDirLength 3
  </IfModule>
  #LoadModule mem_cache_module modules/mod_mem_cache.so
  <IfModule mod_mem_cache.c>
    CacheEnable mem /
    MCacheSize 4096
    MCacheMaxObjectCount 100
    MCacheMinObjectSize 1
    MCacheMaxObjectSize 2048
  </IfModule>
</IfModule>
```

Cacheing – Reverse Proxy: MySQL Buffer

Hochlast vorbereiten

MySQL max_connections :

- **default: 150 connections beim mysqld**
- **hochsetzen!**



expliziter MySQL-Proxy

- kann Verteilung/Unterscheidung lesende und schreibende Requests für r/w-Master und Multi-r/o-Slave
- **max. 5000 connections beim Proxy**

Cache nur zum Puffern taugt nur bei (stark) verteilten Servern,
sonst braucht builtin Buffer weniger Ressources

Cacheing – Reverse Proxy: MySQL Buffer

Hochlast vorbereiten

```
>>>>>>   max_connections = 2000
>>>      key_buffer = 256M
         max_allowed_packet = 1M
>        table_cache = 256
         sort_buffer_size = 1M
         read_buffer_size = 1M
         read_rnd_buffer_size = 4M
         myisam_sort_buffer_size = 64M
         thread_cache_size = 8
>>>>>    query_cache_size = 16M
         # Try number of CPU's*2 for thread_concurrency
         thread_concurrency = 8
```

Hisolutions

Agenda - Surviving Slashdot

Server down...

- Ausfall-Szenarien
- Auslöser
- Technische Gründe

Hochlast vorbereiten

- Architektur / Administration
- Server-/Daemon-Optimierung
- Cacheing
- Load-Balancing
- HTML-Optimierung
- Anwendungsoptimierung

Was tun beim Kind im Brunnen?

- Statusaufnahme
- Notbetrieb / 1. Hilfe

Load-Balancing

Hochlast vorbereiten

Hardware – LVS-Modelle

- ICMP-Redirector
- NAT / Firewall
- TCP Redirector / Reverse-Proxy
- HTTP/302 Moved Temporarily (allerdings: mehrfache Anfragen)

Clientseitig

- CARP = clientseitig Auswahl nach IP-Hash (ursprünglich: PAC-File)
- DNS-RR: Browser behält DNS-Auflösung üblicherweise, DNS-Caches

Session-keeping z.B. bei komplexeren Anwendungen (oft: MVC-basierte Modelle) kritisch, da Daten sonst auf „falschem“ Server liegen.

Load-Balancing – DNS-RR

Hochlast vorbereiten

DNS Round-Robin (DNS-RR)

```
~$ dig  +short www.l.google.com
209.85.129.99
209.85.129.147
209.85.129.104
209.85.129.103
```

- billig
- global verteilt
- kaum aktiv steuerbar, reagiert mit Zeitverzögerung
- kein Session-Keeping

Wenn auch HTTP**S** auf denselben Adressen: Weiterleiten auf SSL-Server

```
iptables -A PREROUTING -t nat -p tcp --dport 443 -d 1.2.3.444 -i eth0 -j DNAT --to 999.8.7.6:443
iptables -A FORWARD -t filter -p tcp --dport 443 -d 999.8.7.6 -i eth0 -j ACCEPT
iptables -A POSTROUTING -t nat -p tcp --dport 443 -d 999.8.7.6 -o eth0 -j MASQUERADE
```


Load-Balancing – Lastverteilung nach Typ

Hochlast vorbereiten

Aufteilung nach Server per DNS

www.wyae.de / mail.wyae.de / forum.wyae.de / wiki.wyae.de / shop.wyae.de

Aufteilung per Reverse-Proxy

```
# NGINX – http://www.debian-administration.org/article/Speeding_up_dynamic_websites_via_an_nginx_proxy
location ~* \.(js|css|rdf|xml|ico|txt|gif|jpg|png|jpeg)$ {
  root   /home/www/www.debian-administration.org/htdocs/;
  access_log  /var/log/nginx/d-a.direct.log ;
}
# Proxy all remaining content to (F)CGI-Apache
location / {
  proxy_pass          http://127.0.0.1:8080/;
  proxy_redirect      off;
[...]
```

Dadurch auch: einfaches Umschwenken auf Notfall-Seiten möglich

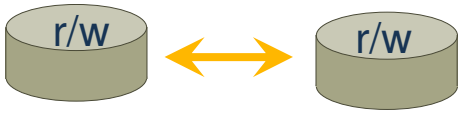
Load-Balancing – Datenbanken

Hochlast vorbereiten

Loadbalancing schwieriger: DB-Server müssen Sync'd werden

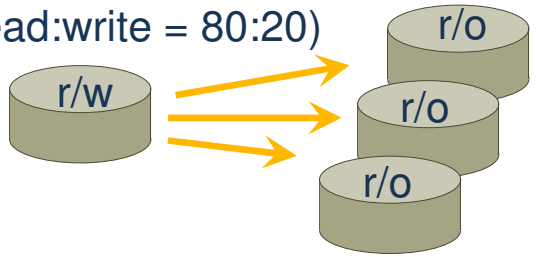
symmetrisch Aktiv / Aktiv

- Master/Master
- Ring
- immer: hoher Aufwand



asymmetrisch 1:n Aktiv / Aktiv

- read/write, der auf mehrere readonly DBs verteilt (read:write = 80:20)
- auch async ok: MySQL-Replikation, Dump



spezialisierte DBs

- spaltenorientierte DBs (Facebook, ...)
- Hypethema „NoSQL“

Hisolutions

Agenda - Surviving Slashdot

Server down...

- Ausfall-Szenarien
- Auslöser
- Technische Gründe

Hochlast vorbereiten

- Architektur / Administration
- Server-/Daemon-Optimierung
- Cacheing
- Load-Balancing
- HTML-Optimierung
- Anwendungsoptimierung

Was tun beim Kind im Brunnen?

- Statusaufnahme
- Notbetrieb / 1. Hilfe

HTML- / Layout-Optimierung – weniger Dateien

Hochlast vorbereiten

weniger Files sind bei gleicher Gesamtgröße weniger ressourcenfressend

- weniger Filehandles/Sockets
- weniger HTML-Overhead
- CSS/JS-Files zusammenfassen (nicht übertreiben – zu viel ungenutztes?)

wiederverwendete Teile (CSS, JS) cachebar ausgelagert

zusammengefasste Bilder / Icon-Maps mit CSS

```
http://img0.gmodules.com/ig/images/v2/sprite0_classic.gif
div{
    background: red url('sprite0_classic.gif') repeat-x left bottom;
    background-slice: 0 0 15px 15px; /*x y width height */
}
```

iGoogle™

HTML- / Layout-Optimierung – Frontend-Performance

Hochlast vorbereiten

weniger Files sind bei gleicher Gesamtgröße weniger ressourcenfressend

- Filehandles/Sockets
- weniger HTML-Overhead
- wiederverwendete Teile (CSS, JS) cachebar ausgelagert

Web2.0 / AJAX kann schnell ein Server-Killer werden...

+ Arbeitslast auf Client auslagerbar
- viele kleine Requests

„14 Rules“ v.a. für Frontend-Perfomance

- Stylesheets an Anfang, JavaScriot ans Ende (Frontend-Tempo)
- JS und CSS in (je eine) Datei auslagern
- interpretierten Code (v.a. JS) optimieren (Größe/minify, Duplikate)
- cachebare Snippets (CSS, JS, AJAX-Requests wenn verwendet)
- passende HTML-Header (siehe: Cacheing)

Hisolutions

Agenda - Surviving Slashdot

Server down...

- Ausfall-Szenarien
- Auslöser
- Technische Gründe

Hochlast vorbereiten

- Architektur / Administration
- Server-/Daemon-Optimierung
- Cacheing
- Load-Balancing
- HTML-Optimierung
- Anwendungsoptimierung

Was tun beim Kind im Brunnen?

- Statusaufnahme
- Notbetrieb / 1. Hilfe

Andwendungsoptimierung

Hochlast vorbereiten

Optimierung Plattform / Programmauswahl

- Auswahl einer geeigneten Plattform (BASH ist langsam...)
- Vermeidung externer Programmaufrufe (CGIs) für häufige Seiten
- Viele CMS haben Cacheing auf verschiedenen Ebenen eingebaut (Drupal, Typo3). Sind die eigenen Caches konfiguriert? Groß genug?
- Oft haben CMS Development- und Produktions-Modus - ist umgeschaltet? (Drupal: Develop = viele CSS-/JS-Einzeldateien vs. Produktion = 1-2 kombinierte)
- vorcompilierte / zusammengestellte / gecachte Teile
- statisches 'Rausrendern möglich? (nur wenigs CMS - warum eigentlich?) statische Seiten auf 1.3GHz Celeron: 100 Mbit/s von Disk, 500 Mbit/s aus Buffer

Optimierung Programmierung

- Filehandler (Dateien) in CGIs: möglichst wenige, möglichst schnell schließen
- DB: ein DB-Connect per Seite, oder einer je Sub-Abfrage?
- Datenbankabfragen minimieren - Graus-Beispiel: Anzahl Posts für `insert` statt Autoincrement oder nur `select count(*)` über: `select *;` dann `foreach () { $count+=1; }`

Andwendungsoptimierung – memcached

Hochlast vorbereiten

memcached.org

- ein großer Cache-Pool, der dynamisch unter Anwendungen aufgeteilt wird, also optimale Nutzung über mehrere Anwendungen hinweg
- Key-Value Store
- nutzbar z.B. für vorgerenderte Seiten (oder Seitenteile)
- nur im RAM: schnell, ohne File-Access (aber Netzwerk)
- Verteilung über mehrere Server möglich, über CARP-ähnliches Protokoll daher: Hinzunehmen/Rausfallen von Nodes löscht den ganzen Cache

Performance

1. Zugriff auf interne Ressourcen (Array) ist am schnellsten
2. Memcache, wenn lokales RAM nicht groß genug
3. Buffers / FileCache
4. Query-Cache

`http://code.google.com/p/memcached/wiki/WhyNotMemcached`

`http://www.mysqlperformanceblog.com/2006/08/09/cache-performance-comparison/`

Andwendungsoptimierung – memcached

Hochlast vorbereiten

Memcache „impfen“

```
//  http://code.google.com/p/memcached/wiki/TutorialCachingStory

$huge_data_for_frong_page = $memcache->get("huge_data_for_frong_page");
if($huge_data_for_frong_page === false){
    $huge_data_for_frong_page = array();
    $sql = "SELECT * FROM hugetable WHERE timestamp > lastweek
        ORDER BY timestamp ASC LIMIT 50000";
    $res = mysql_query($sql, $mysql_connection);
    while($rec = mysql_fetch_assoc($res)){
        $huge_data_for_frong_page[] = $rec;
    }
    // cache for 5 minutes
    $memcache->set("huge_data_for_frong_page", $huge_data_for_frong_page, 600);
}
```

Hisolutions

Agenda - Surviving Slashdot

Server down...

- Ausfall-Szenarien
- Auslöser
- Technische Gründe

Hochlast vorbereiten

- Architektur / Administration
- Server-/Daemon-Optimierung
- Cacheing
- Load-Balancing
- HTML-Optimierung
- Anwendungsoptimierung

Was tun beim Kind im Brunnen?

- Statusaufnahme
- Notbetrieb / 1. Hilfe

Incident Management

Was tun beim Kind im Brunnen?

1. Statusaufnahme
2. BCM-Maßnahmen anwenden
3. Rückführung in Normalbetrieb

Incident Management

Was tun beim Kind im Brunnen?

1. **Statusaufnahme**
2. **BCM-Maßnahmen anwenden**
3. **Rückführung in Normalbetrieb**

Ist hoffentlich vorhanden...

... oder?

... ODER?!

Upps ...

Hisolutions

Agenda - Surviving Slashdot

Server down...

- Ausfall-Szenarien
- Auslöser
- Technische Gründe

Hochlast vorbereiten

- Architektur / Administration
- Server-/Daemon-Optimierung
- Cacheing
- Load-Balancing
- HTML-Optimierung
- Anwendungsoptimierung

Was tun beim Kind im Brunnen?

- Statusaufnahme
- Notbetrieb / 1. Hilfe

Notfall – Triage : Was ist wichtig? Was ist los?

Was tun beim Kind im Brunnen?

Quick&Dirty-Maximen:

- Es muss eine minimale Basisinformation der Kunden / Interessenten möglich bleiben.
- Ein Ausfall der Kommunikationskanäle (E-Mail, Notfall-Webseite) muss vermieden werden.

Schnelle / grobe Analyse:

- was verursacht die Last? Welche Files/Anwendung ?
  - Webserver-Logs
  - Firewall-Logs / Statistiken
  - „top“ / „netstat“
- Slashdotted (viel Interesse) oder (D)DoS-Angriff ?
- einzelne IPs (Crawler) oder viele (Clients) ?

Erste Entscheidungen / Kommunikation

Was tun beim Kind im Brunnen?

Provider informieren, um Fremdabschaltung zu vermeiden:

- was los ist (DoS - oder aber hohes Interesse wegen ...)
- dass man Gegenmaßnahmen einleitet
- bitte (nicht) wegen DoS/Trafficüberschreitung abschalten (Achtung: Kosten!)

Webseiten priorisieren, auch unwichtigere temporäre Opfer

1. Frontseite = *Kommunikationskanal*
2. Shop (Bestelldienst)
3. Forum, in dem sich immer mehr Kunden über langsamen Shop (und langsame werdendes Forum) beschweren
4. Mediaserver
5. Page-Tracking Server

Wenn Infrastruktur gefährdet (besonders einzelner Gesamtserver):

Last erzeugenden Dienst abschalten - dann ggfs. auf Notbetrieb umbauen.

Hisolutions

Agenda - Surviving Slashdot

Server down...

- Ausfall-Szenarien
- Auslöser
- Technische Gründe

Hochlast vorbereiten

- Architektur / Administration
- Server-/Daemon-Optimierung
- Cacheing
- Load-Balancing
- HTML-Optimierung
- Anwendungsoptimierung

Was tun beim Kind im Brunnen?

- Statusaufnahme
- Notbetrieb / 1. Hilfe

1. Hilfe - Notbetrieb

Was tun beim Kind im Brunnen?

Eine (in Zahlen: 1) statische Ersatz-Start-Seite zur Kundeninformation:

„Sorry, Überlast, bitte später (Zeitrahmen) wiederkommen...“

entsprechende 404 / 5xx Fehlerseiten

„Sorry, Überlast/Serverfehler bitte später wiederkommen...“

Einzelne Lastspitzen-Files oder -Anwendungen nach Prio abklemmen:

- Forum, Wiki, Shop, ... bzw. Media-Files
- Herunterfahren von entsprechenden Anwendungen / Anwendungsservern

Umschwenken von Last"verzeichnissen":

```
mv /var/www/forum /var/ww/forum.disabled
```

... und dann ziehen die 404 / 5xx Seiten mit Überlast-Meldungen

1. Hilfe - Notbetrieb

Was tun beim Kind im Brunnen?

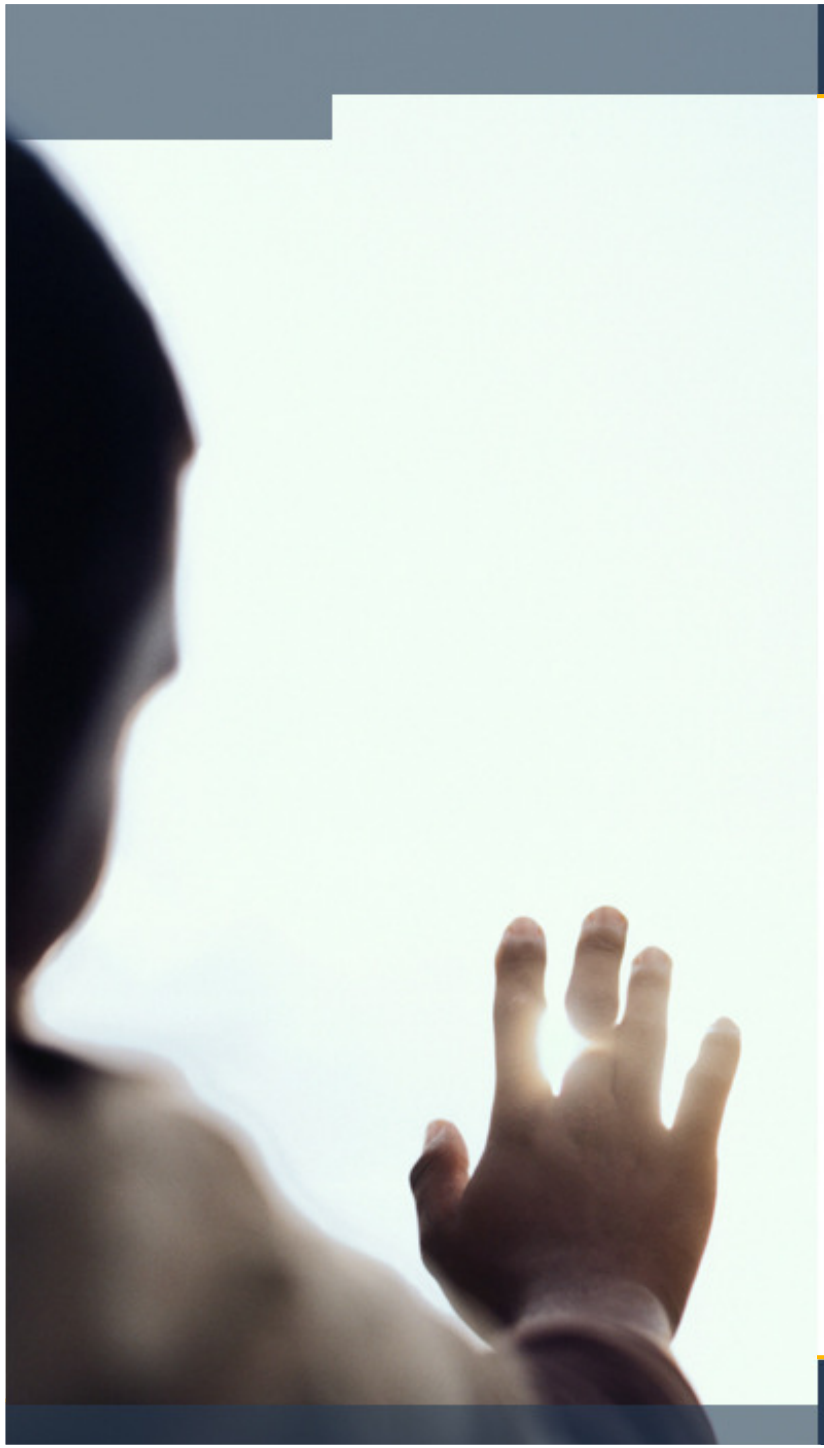
statische Notfall-Kopie

- Server nur auf 127.0.0.1 hochfahren
- statische Kopie erstellen
  ```
  httrack -%P -O /var/www/staticcopy/ --update localhost
  ```
- Server auf statische Kopie umstellen und wieder hochfahren

Laufendes Monitoring!

Server-Limits anpassen

- Bandbreitenlimit
- Connections global / per IP
- max. Threads/Children
- einzelne IPs (Crawler) limitieren/aussperren



Hisolutions

Surviving Slashdot

Fragen?

Server down...

- Ausfall-Szenarien
- Auslöser
- Technische Gründe

Hochlast vorbereiten

- Architektur / Administration
- Server-/Daemon-Optimierung
- Cacheing
- Load-Balancing
- HTML-Optimierung
- Anwendungsoptimierung

Was tun beim Kind im Brunnen?

- Statusaufnahme
- Notbetrieb / 1. Hilfe

http://wyae.de/volker.tanger/papers/

Kontakt

Anschrift

HiSolutions AG
Bouchéstraße 12
D-12435 Berlin

Fon: +49 30 533289-0
Fax: +49 30 533289-99
www.hisolutions.com

Information Security

Timo Kob
Mitglied des Vorstands
Leiter Information Security
kob@hisolutions.com

Volker Tanger
Senior Consultant
tanger@hisolutions.com