Der Webserver

Marcel Holtmann
Universität Bielefeld - Technische Fakultät
AG Rechnernetze und verteilte Systeme
marcel@rvs.uni-bielefeld.de

16. Juni 1999

Das Internet

- Bild des ARPANET aus dem Jahre 1969

Entstehung des World Wide Web

- Die Wiege des WWW liegt am Kernforschungsinstitut CERN in Genf (`www.cern.ch`)
- 1989 begann Tim Berners-Lee mit der Entwicklung des WWW
- 1990 wurde das CERN an das Internet angeschlossen
- Kurz danach wurde der erste WWW-Vorschlag veröffentlicht
- Das WWW war für die Physiker gedacht, damit sie auf die Daten mit einem zeilenorientierten Browser zugreifen konnten

Der erste graphische Browser

- 1993 wurde von Marc Andreesen der erste Web-Browser mit graphischer Oberfläche veröffentlicht
- Mit dem Browser *Mosaic* wurde das WWW schlagartig populär und es gab eine Vielzahl von Web-Servern
- 1994 verließ Marc Andreesen NSCA und gründete mit James Clark (Gründer von SGI) die Firma Netscape Communications
- Mit dem Netscape Navigator wurde ein neuer Standard gesetzt
- Nach dieser Zeit verbreitete sich das World Wide Web rasant und ist zu dem bekanntesten Dienst im Internet geworden

Das W3-Konsortium

- Die Koordination der Protokolle und Spezifikationen des WWW ging bis vor einiger Zeit vom CERN aus
- Das starke Wachstum des WWW überstieg jedoch die Möglichkeiten des CERN für weitere Entwicklungen und es wurde das W3C ins Leben gerufen
- Das W3C ist eine Kooperation des CERN und des MIT. Weiterhin gehören viele Firmen zum W3C (z.B. Netscape, Microsoft etc.)
- Das W3C ist für **alle** Standards im Internet zuständig
- Der derzeitige Direktor ist Tim Berners-Lee

Der Webserver

- Neben dem Browser Mosaic gab es den CERN Web-Server, der von Tim Berners-Lee entwickelt wurde
- Der CERN Server war lange der Standard Web-Server im Internet
- 1995 war der NCSA Web-Server (von Rob McCool entwickelt) der meistgenutzte
- Nachdem Rob McCool das NSCA verlassen hatte kam die Weiterentwicklung ins stocken
- Viele Leute begannen den Web-Server in eigener Regie weiterzuentwickeln und es gab eine große Anzahl von Patches

Der Apache

- Aus den gepatchen Servern (*a patchy server*) ist dann der Apache Web-Server geworden
- Im April 1995 wurde dann die erste Beta-Version (basierend auf NCSA Web-Server, Version 1.3) des Apache veröffentlicht
- Im Dezember 1995 wurde dann die erste Vollversion 1.0 freigegeben
- Nicht ganz ein Jahr später hatte der Apache den NSCA-Server von dem ersten Platz der meistgenutzen Web-Server verdrängt

Was braucht man für eine Web-Site

- Ein Rechnersystem mit ausreichender CPU-Leistung, viel Speicher und Festplatte mit schnellen Durchsatz und einen guten Controller
- Ein Netzwerkinterface und eine Netzanbindung mit mind. 2 MBit/s
- Die Hardware sollte aufeinanderabgestimmt sein um eine optimale Verfügbarkeit zu garantieren (z.B. Workstation oder Server-Systeme)
- Einen Web-Server (z.B. den Apache)

Die Apache-Distribution

- Die Apache-Software ist frei im Sourcecode verfügbar, jedoch nicht Public Domain. Die Lizenzbestimmungen liegen in der Datei `LICENSE`

- Die neuste Version (aktuell ist die Version 1.3.6) bekommt man unter `www.apache.org/dist`

- Zum kompilieren sollte man den GNU C-Compiler benutzen

- Nach dem entpacken des Tar-Files erhält man folgende Verzeichnisse:

```
conf    icons     cgi-bin   htdocs
src     support   logs
```

Die `Configuration`-Datei

- In dem Verzeichnis `src` befindet sich die Datei `Configuration`, die zum konfigurieren des Apaches für das entsprechende Betriebssystem dient

- In der Datei können Paramter für zusätzliche Konfigurationen eingetragen werden (z.B. Einbindung von SQL-Libs)

```
EXTRA_CFLAGS=
EXTRA_LFLAGS=
EXTRA_LIBS=
EXTRA_INCLUDES=
#CC=
OPTIM=-O2
```

Die Flags der `Configuration`

HTTPD_ROOT

Hiermit wird das Installationsverzeichnis für den Apache festgelegt (z.B. `EXTRA_CFLAGS= -DHTTPD_ROOT=\"/opt/apache\"`)

SERVER_CONFIG_FILE

Ausgehend von dem ROOT-Verzeichnis wird die Hauptkonfigurationsdatei angegeben (z.B. `-DSERVER_CONFIG_FILE=\"conf/httpd.conf\"`)

SERVER_SUBVERSION

Hiermit kann eine Patch-Version für den Apache angegeben werden, die dann bei jeder Anfrage zurückgeliefert wird (z.B. `-DSERVER_SUBVERSION=\"MegaPatch/1.0\"`)

Die Rules der `Configuration`

- Mit den *Rule*-Angaben kann man dem Server weitere Funktionalitäten übermitteln

- Die Syntax ist folgendermaßen: `Rule` *Name*=*Wert*

- Die Werte dabei können *yes*, *no* oder *default* sein

- Standardmässig werden die Werte an Hand der OS-Konfiguration eingestellt

- Mögliche Rules sind hierbei *IRIXNIS*, *SOCKS4*, *STATUS* und *WANTSREGEX*

Die Module der `Configuration`

- Der Apache hat einen modularen Aufbau
- Alle Funktionen außer der Implementierung des HTTP sind in Module ausgelagert (z.B. CGI, SSI, Zugriffskontrollen etc.)
- Die meisten Module sind schon als Standard in der Konfiguration eingetragen
- Syntax: `Module cgi_module mod_cgi.o`
- Die Kommunikation der Module untereinander und mit dem Server-Kern erfolgt über das Apache-API, welches gut dokumentiert und frei verfügbar ist

Das Compilieren

- Zum Einstellen der Konfiguration führt man das Script `Configure` aus

```
> ./Configure
Using config file: Configuration
Using Makefile template: Makefile.tmpl
 + configured for Linux platform
 + setting C compiler to gcc
 + setting C compiler optimization-level to -O2
```

- Wenn keine Konfiguration für das OS existiert, meldet das Script

```
Sorry, but we cannot grok "..."
```

Das Compilieren

- Nach dem Aufruf von `make` werden die Module übersetzt, dann wird der Server `httpd` erzeugt und zum Schluß die Programme aus dem Verzeichnis `support` kompiliert

- Danach sollte man Kontrollieren, ob die dynamischen Libraries gefunden werden

```
> ldd src/httpd
    libgdbm.so.1 => /usr/lib/libgdbm.so.1 (0x40008000)
    libm.so.6 => /lib/libm.so.6 (0x4000f000)
    libdl.so.2 => /lib/libdl.so.2 (0x40029000)
    libcrypt.so.1 => /lib/libcrypt.so.1 (0x4002d000)
    libnsl.so.1 => /lib/libnsl.so.1 (0x4005b000)
    libresolv.so.2 => /lib/libresolv.so.2 (0x40062000)
    libdb.so.2 => /lib/libdb.so.2 (0x40074000)
    libc.so.6 => /lib/libc.so.6 (0x40084000)
    /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x2aaaa000)
```

Die Erstinstallation

```
# cd apache_1.2.6
# ./Configure
# make
# mkdir /opt/apache
# cp -r icons conf logs /opt/apache
# rm /opt/apache/icons/README
# mkdir /opt/apache/bin /opt/apache/sbin
# cp src/httpd /opt/apache/sbin
# chmod 644 /opt/apache/icons/*
# chmod 200 /opt/apache/sbin/*
# chmod 755 /opt/apache/*
# cp my_httpd.conf /opt/apache/conf/
# /opt/apache/sbin/httpd
```

Die `httpd.conf` Datei

```
# ServerType is either inetd, or standalone.
ServerType standalone

# Port: The port the standalone listens to.
Port 80

# If you wish httpd to run as a different user or group,
# you must run httpd as root initially and it will switch.
User nobody
Group nogroup

# BindAddress: is used to tell the server which IP address to listen to.
BindAddress 192.168.5.100

# ServerAdmin: Your e-mail address.
ServerAdmin webmaster@domain.de

# ServerName allows you to set a host name which is sent back to clients
ServerName www.domain.de
```

Die `httpd.conf` Datei

```
# ServerRoot: The directory the server's config are kept in.
ServerRoot "/opt/apache"

# ErrorLog: The location of the error log file.
ErrorLog logs/error_log

# The location of the access logfile (Common Logfile Format).
CustomLog logs/access_log common

# PidFile: The file the server should log its pid to
PidFile logs/httpd.pid

# Number of servers to start.
StartServers 10
MinSpareServers 5
MaxSpareServers 20

# Number of client connections.
MaxClients 15
MaxRequestsPerChild 3
```

Die srm.conf Datei

```
# DocumentRoot: The directory out of which you will serve your docs
DocumentRoot "/opt/www/htdocs"

# UserDir: The name of the directory which is appended onto a user's home
UserDir public_html

# DirectoryIndex: Name of the file to use as a pre-written HTML directory
DirectoryIndex index.html

# IndexIgnore is a set of filenames which directory indexing should ignore
IndexIgnore .??* *~ *# HEADER* README* RCS

# AccessFileName: The name of the access control information for a dir.
AccessFileName .htaccess

# Aliases: Add here as many aliases as you need (with no limit).
Alias /icons/ "/opt/apache/icons/"

# ScriptAlias: This controls which directories contain server scripts.
ScriptAlias /cgi-bin/ "/opt/www/cgi-bin/"
```

Die access.conf Datei

```
# First, we configure the "default" to be a very restrictive set of perms.
<Directory />
Options FollowSymLinks
AllowOverride None
</Directory>

# This should be changed to whatever you set DocumentRoot to.
<Directory "/opt/apache/htdocs">
Options Indexes FollowSymLinks
AllowOverride None
order allow,deny
allow from all
</Directory>

# cgi-bin dir should be changed to whatever your ScriptAliased
# CGI directory exists, if you have that configured.
<Directory "/opt/apache/cgi-bin">
AllowOverride None
Options None
</Directory>
```

Einführung in HTTP

- Der *Uniform Resource Identifier* (URI) ist für die Adressierung der Ressourcen im World Wide Web zuständig

- Eine spezielle Form des URI ist der *Uniform Resource Locater* (URL)

- Ein URI besteht aus zwei Teilen, dem *Schema* und dem *Pfad*. Der Pfad wiederum besteht aus Hostnamen und einem Dateipfad

```
Syntax: Schema://Hostname:Port/Server-Pfad
```

Einführung in HTTP

- Die in einem URI vorkommenden Zeichen müssen 7-Bit kodiert sein
- Sonderzeichen (z.B. Tilde, Umlaute etc.) müssen durch eine Escape-Sequenz dargestellt werden, die durch ein % eingeleitet und von einem Hex-Code gefolgt wird (`%7E` entspricht dabei der $\sim$)
- Ein Fragezeichen (`?`) stellt die Grenze zum Query-String da (z.B. für CGI-Scripte)
- Das Doppelkreuz (#) dient als Begrenzung zum *Fragment Identifier*

Einführung in HTTP

- Der Query-String wird vom Server ausgewertet, während ein FI nur vom Browser bearbeitet wird

- Ein paar Beispiele:

```
http://www.domain.de/test/index.html
http://www.domain.de/cgi-bin/script?test=0815
http://www.domain.de/publish/paper.html#abstract
https://www.domain.de/%7Ejoe
ftp:ftp.domain.de/program-1.0.tar.gz
news:de.comm.infosystems.www.servers
```

Die Funktionsweise des HTTP

- Das *HyperText Transfer Protocol* ist ein Protokoll der Anwendungsebene (also Schicht 7 nach OSI)
- Es benutzt TCP als Transportprotokoll
- Entwickelt wurde es von Tim Berners-Lee im Rahmen des WWW-Projektes
- Wenn ein URI angefragt werden soll, so wird erst eine Verbindung zum Server erstellt (normalerweise auf 80) und dann wird mit einem HTTP-Befehl die Seite angefordert

Die Funktionsweise des HTTP

```
> telnet www.rvs.uni-bielefeld.de 80
Trying 129.70.123.140...
Connected to localhost.
Escape character is '^]'.
GET /
<HTML>
<HEAD>
...
</BODY>
</HTML>
Connection closed by foreign host.
```

Die Funktionsweise des HTTP

- Normalerweise werden vom Browser noch Infos über den Client mit übermittelt
- Diese Übermittlung geschieht durch die Angabe der Protokollversion und dann die Paramter gefolgt von einer Leerzeile
- Der Server liefert dann seine Header, gefolgt von einer Leerzeile, zurück
- Danach folgt dann die eigentliche Datei, die man angefordert hat

Die Funktionsweise des HTTP

```
GET /test/index.html HTTP/1.1
Host: meinrechner.domain.de
User-Agent: TestClient/0.1

HTTP/1.1 200 OK
Date: Wed, 16 Jun 1999 14:52:10 GMT
Server: Apache/1.3.3 (Unix)
Last-Modified: Tue, 19 Jan 1999 21:42:01 GMT
Content-Type: text/html

<HTML>
<HEAD>
...
</BODY>
</HTML>
```

Die Methoden von HTTP

GET Hiermit wird das in einem URI angegebene Objekt von einem Web-Server angefordet

HEAD Diese Methode ist ähnlich zu GET, fordert aber nur den Header an (z.B. für das Datum der letzten Änderung)

PUT PUT ist das Gegenstück zu GET und dient zum Speichern von Daten auf dem Server (muß vom Server erlaubt sein)

POST Durch POST können dem Server Daten übermittelt werden, die dann weiterverarbeitet werden (z.B. mit CGI-Skripten)

DELETE Löscht ein Objekt von dem Server

Aufbau einer Client-Anfrage

1. Das *Method-Token*, der *Request-URI* und die Protokollversion *HTTP-Version*

2. Der *General-Header*: Connection, Cache-Control etc.

3. Der *Request-Header*: Host, User-Agent, Accept, Referer etc.

4. Der *Entity-Header*: Content-Type, Content-Length etc.

5. Der *Entity-Body*: Die bei einer POST-Anfrage übergebenen Daten

Aufbau der Server-Antwort

1. Die *HTTP-Version*, der *Statuscode* und eine *Reason-Phrase* (Kurzbeschreibung des Statuscodes)

2. Der *General-Header*: Connection, Date, Cache-Control etc.

3. Der *Response-Header*: Location, Server etc.

4. Der *Entity-Header*: Content-Type, Content-Length etc.

5. Der *Entity-Body*: Daten der Server-Antwort, falls erforderlich

Die Statuscodes

100-199	Informative Meldungen, die Aktionen des Client anregen sollen
200-299	Client-Anfrage ist gültig
300-399	Die Anfrage wurde nicht ausgeführt
400-499	Die Anfrage ist fehlerhaft oder unvollständig
500-599	Server-Fehler

200	OK	Die Anfrage war erfolgreich - Daten geliefert!
400	BAD REQUEST	Die Anfrage war fehlerhaft
404	NOT FOUND	Die Datei konnte nicht gefunden werden
500	INTERNAL ERROR	Z.B. beim Absturz eines CGI-Skriptes

Das *Common Gateway Interface*

- Das CGI dient zum Erzeugen von dynamischen Web-Seiten
- CGI ist unabhängig von Programmiersprachen, denn es stellt lediglich eine Schnittstelle bereit
- Die Erzeugung von Web-Seiten geschieht auf dem Server
- JAVA zum Beispiel wird auf der Client-Seite ausgeführt, während CGI-Skripte oder auch CGI-Prgramme auf dem Server ausgeführt werden und nur das Ergebnis übermittelt wird

Ein kleines CGI-Skript

- Die Datei `/cgi-bin/showdate` ist ein CGI-Skript
- Beim Aufrufen von `http://www.domain.de/cgi-bin/showdate` wird die aktuelle Uhrzeit und das Datum auf dem Bildschirm angezeigt
- Die Schnittstelle zum CGI soll mit Hilfe eines Shell-Skriptes implementiert werden
- Zur Datumsberechnung wird das Unix-Programm `date` benutzt

Ein kleines CGI-Skript

```
#!/bin/sh
echo "Content-Type: text/html"
echo ""
echo "<HTML>"
echo "<HEAD>"
echo "  <TITLE>Aktuelles Datum und Uhrzeit</TITLE>"
echo "</HEAD>"
echo "<BODY>"
echo "<H1>"
date
echo "</H1>"
echo "</BODY>"
echo "</HTML>"
```