Fachhochschule
Bonn-Rhein-Sieg

Apache Webserver

Verteilte und Parallele Systeme 2

P.Fiksman, M.Horoz

Fachhochschule Bonn-Rhein-Sieg
November 2004

Inhaltsverzeichnis

1. Einleitung

1.1. Vorgeschichte

Die Webserver an sich wurden etwa Mitte 1994 ins Leben gerufen, kurz nachdem der erste grafische Webbrowser entwickelt wurde, mit dem Namen Mosaic. Kurz nachdem CERN-Server (Anfang 1995), entwickelte Rob McCool den NCSA-Webserver, der zu den am meisten genutzten Standardservern galt. Nachdem McCool das NCSA verlassen hatte, entwickelten immer mehr Leute Patches in eigener Regie und verbesserten den Webserver nach und nach. Der dadurch entstandene gepatchte NCSA-Server (*a patchy server*), wurde zum Apache umgetauft. Im April 1995 wurd die erste Beta-Version des Apaches ins Leben gerufen. Die im Dezember 1995 herausgebrachte Apache-Server mit der Version 1.0 verdrängte den NCSA-Server, der zu seiner Zeit an erster Stelle war. Der Grund für seine Beliebtheit hatte der Webserver seinem modularen Konzept zu verdanken. Im April 2002 veröffentlichte die Apache Software Foundation die erste stabile Version (2.0) des Apache-Webservers. Zudem war der Apache der erste Server, der vollständig die HTTP/1.1-Spezifikation unterstützte.

1.2. Apache Software Foundation

Die Apache Software Foundation wurde im Mai 1999 von der Apache Group, deren Mitgliederzahl sich auf ca. 70 aktive Mitglieder aus verschiedensten Ländern beläuft, gegründet. Gründe hierfür war eine Instanz zu haben, die auch zukünftig dafür sorgt, dass die Verbreitung des Apaches erhöht wird. Außerdem sollten für die Open-Source-Projekte organisatorische, rechtliche und finanzielle Unterstützung zur Verfügung gestellt werden. Die Foundation hat ihren Sitz in den USA.

1.3. Apache heute

Der Apache Webserver gehört als Open-Source Webserver heutzutage mit 65% aller Webserver, zu den am meisten eingesetzten Webservern neben Microsoft(IIS) und Sun. Durch seinen modularen Aufbau lässt dieser sich leicht durch zusätzliche Funktionen erweitern. Der Lieferumfang umfasst mehr als 100 Module. Zu diesen zählen Authentifizierungsmodule, Skript- bzw. Interpretermodule für verschiedene Programmiersprachen, Module für Datenbankanbindung(PHP), sowie Module für sichere Verbindungen per SSL. Nicht alle Module sind kostenfrei, da Firmen ihre Produkte als natives Apache-Modul anbieten, wie z.B. die Firma Allaire mit der Java-Servlet-Engine.
Weiterhin ist der Server für verschiedene Betriebssysteme erhältlich, wie z.B. Linux, Windows, Solaris und BSD. Die Lizenzbestimmungen erlauben außerdem den Apache in kommerzielle Produkte zu integrieren, weswegen auch viele größere Firmen wie IBM den Apache als Basis für ihr WebSphere-Produkt nehmen.

2. Einrichtung/Installation

Alle folgenden Punkte zum Apache beziehen sich zum größten Teil auf die Version 2.0.

2.1. Installation der Apache Version 2 unter Windows

Bei der Installation unter Windows, wird man nachdem die Exe-Datei gestartet wurde, durch den Installations-Wizard Schritt für Schritt durch die Installation geleitet. In den ersten Schritten klickt man sich durch Lizensvertrag und der Readme-Datei zum Apache durch. Nachdem man die Readme-Datei durchgelesen hat, gelangt man auf die Seite in der die Netzwerk-Domäne, der Rechnername auf dem der Server läuft und die E-Mail-Adresse des Administrators angeben werden muss. Man kann außerdem festlegen, ob für alle Benutzer unter dem Port 80 der Server eingerichtet und automatisch gestartet werden soll oder der Server nur für den aktuellen Benutzer manuell gestartet werden kann. Im nächsten Fenster kann man zwischen der typischen Installation und der benutzerdefinierten Installation wählen. Bei der typischen Installation werden die Binärdateien und Konfigurationsdateien (Apache Runtime) mitinstalliert. Bei der anderen Variante kann man zusätzlich wählen ob die Header-Dateien und die Bibliotheken mit installiert werden sollen. Mit *next* klickt man sich in das darauffolgende Fenster, in dem das Verzeichnis gewählt wird, indem der Apache-Webserver installiert wird.

2.2. Installation der Apache Version 2 unter Linux + Grundkonfiguration

2.2.1. Installation

(nicht anwendbar für RedHat- und Slack-Distributionen, hierfür steht ein spezielles rpm-Format zur Verfügung)

Den Quellcode des Apache-Webservers entpackt man, wenn dieser als gzip-Datei vorliegt, mit dem Befehl *gunzip httpd-2.0.52.tar.gz*.
Die im selben Verzeichnis entstandene TAR-Datei entpackt man wiederum und zwar mit *tar -xf httpd-2.0.52.tar*. Es werden Quelldateien und die Dokumentation, in den durch das Entpacken entstandenen Ordner httpd-2.0.52, extrahiert. Als nächstes müssen dann die Übersetzungsroutinen konfiguriert werden. Dies geschieht mit dem Aufruf des Skriptes *configure, w*elches als Parameter das Zielverzeichnis für die Installation erwartet. Z.B. *~/httpd.*
Der Aufruf sähe wie folgt aus: *./configure --prefix=/home/pmuster2s/httpd.*
Während der Konfiguration wird z.B. geprüft ob ein Compiler vorhanden ist und welchen Funktionen dieser unterstüzt. Im Fehlerfall wird der Konfigurationsverlauf abgebrochen, so dass evtl. Pakete nachinstalliert werden müssen. Nachdem die Übersetzung fehlerfrei vollendet wurde, kann der eigentliche Übersetzungsvorgang, durch die vorher erzeugten Make-Dateien, mit dem Befehl *make* durchgeführt werden.

Nachdem der Quelltext übersetzt wurde, kann man nun die entsprechenden Programme installieren, der Befehl hierzu lautet: *make install.*
Durch diesen Befehl werden nur die Standardmodule mitinstalliert. Die Installation weiterer Module ist durch den Befehl *configure* möglich, wobei eine Auflistung der Module durch Eingabe von *./configure --hel*p angezeigt wird.

Jetzt muss nur noch der Server gestartet werden!

2.2.2. Grundkonfiguration (ab SUSE 8.0)

Nachdem das System erfolgreich auf dem Rechner installiert wurde muss noch geprüft werden, ob der Server standardmäßig auf den Runleveln 3 und 5 gestartet wird. Runlevel 3 steht für System mit Netzwerk und Runlevel 5 für X (grafische Oberfläche) mit Netzwerk. Um dies zu überprüfen startet man den Runlevel-Editor von YaST2. Dort können unter der Spalte, in dem sich die Systemdienste befinden, die Runlevel-Einstellungen für den Apache angeschaut werden. Außerdem kan man dort unter der Spalte Aktiv einsehen, ob der Server schon läuft.
Läuft der Server nach Markierung der gewünschten Runlevels und Neustart des Servers dennoch unter den eingestellten Runleveln nicht, wechseln Sie in das Verzeichnis */etc/init.d/rc5.d* zum Start des Servers unter Runlevel 5 bzw. */etc/init.d/rc3.d* zum Start des Servers unter dem Runlevel 3. Dort prüfen Sie, ob die Links mit den Namen *S21apache* und *K03apache* gesetzt worden sind. Ansonsten erzeugt man diese mit den Befehlen

ln -s ../apache S21apache

und *ln -s ../apache K03apache.*

Nun wird der Server nach Neustart des Rechners automatisch gestartet.

Jetzt muss der inetd-Dämon noch gestartet werden, damit der Server auch die Netzwerkschnittstellen und die entsprechenden Protokolle nutzen kann. Die dazugehörigen Einstellmöglichkeiten findet man in YaST2 unter *Netzwerk/Basis/Start* bzw. *Stop* von Systemdiensten. Dort genügt im Regelfall schon die Wahl der Standardkonfiguration. Nach Beendigung von YaST2 wird der Dämon gestartet.

Die durchgeführte Konfiguration war erfolgreich, wenn nach dem Neustart die Zeilen *Starting service httpd* und *Starting INET services* zu sehen sind mit einem abschließenden *done*.

2.3. Apache-Module

Der Apache Webserver ist der einzige Webserver, der das Konzept der modularen Konfiguration verwendet. Nach diesem Konzept, kann der User die Funktionalitäten des Webservers einfach nach seinen Bedürfnissen anpassen, indem er Module hinzufügt bzw. entfernt, da standardmäßig bei der Installation bestimmte Module automatisch schon miteingebunden werden. Zusätzlich zu den Modulen, die der Server bereitstellt, können noch eigene geschrieben werden. Die Kommunikation der Module mit dem Apache-Kern erfolgt über das Application Programming Interface(API).

Die Module werden unterteilt in Standardmodule und Extramodule, von denen im Folgenden nur die wichtigsten kurz beschreiben werden.

2.3.1 Standardmodule

Übersicht der wichtigsten Module, die per Default bei der Installation eingebunden werden.

mod_access
Über dieses Modul kann eine Art Zugriffskontrolle realisiert werden, auf Basis von Hostnamen des Clients oder IP-Adressen, so dass nur eine bestimmte Gruppe auf den Server Zugriff hat. Z.B. Freigabe von lokaler Domain

mod_cgi
Das Modul wird eingebunden, um die Ausführung von CGI-Skripten zu ermöglichen (später mehr)

mod_env
Durch dieses Modul können die Umgebungsvariablen an CGI-Skripte oder SSI-Dokumente übergeben werden. Außerdem können eigene Variablen in der Serverkonfiguration gesetzt werden.

mod_log_config
Durch Einbinden dieses Moduls können die Anfragen an den Server protokolliert werden. Dabei können auch Log-Files an die Bedürfnisse des Users angepasst werden.

mod_mime
Über das mime-Modul kann der Client anhand der Dateiendung den Inhalt einer Datei bestimmen. Andernfalls wüsste der Client nichts mit der empfangenen Datei anzufangen.

mod_setenvif
hier können abhängig von Attributen (Http-Header oder z.B. IP-Adresse) der Anfrage die Umgebungsvariablen gesetzt werden. Ein Beispiel hierfür wäre z.B. das Ausschalten der KeepAlive Funktionalität des Webservers für den Netscape Navigator 2.x.

mod_auth
Das Authentifizierungs-Modul stellt Funktionen zur Verfügung, die eine Benutzerauthentifizierung unter Verwendung von Textdateien bereitstellt.
Bei einer größeren Anzahl von User sollten aus Performance-Gründen die Module *mod_auth_db* bzw. *mod_auth_dbm* stattdessen eingebunden werden. Die Authentifikation der Benutzer finden dann über eine Datenbankschnittstelle statt.

2.3.2. Extramodule

Übersicht der wichtigsten Module die standardmäßig nicht mitinstalliert werden und bei Bedarf eingebunden werden können.

mod_dav und *mod_dav_fs*
Das Modul dient dazu, dass man Ressourcen des Webservers, der sich auf einem entfernten Rechner befindet, über Http verwalten kann.

mod_deflate (weniger wichtig, aber interessant)
Erlaubt sofortige Komprimierung der Inhalte vor dem Senden an den Client, damit die Bandbreite eingespart werden kann.

mod_expires
Über das Modul kann festgelegt werden, wie lange ein Dokument im Cache als aktuell betrachtet werden kann. D.h. nachgeladen werden muss

mod_file_cache
In diesem Modul können oft genutzte Dateien ausgewählt werden, die beim Start des Apaches in den Speicher geladen werden, damit der Zugriff auf diese Dateien beschleunigt wird.

mod_proxy
Mit Hilfe dieses Moduls, werden dem Webserver die Funktionalitäten eines Proxys zur Verfügung gestellt. Die Caching-Funktionalität ist seit Version 2.0 in andere Caching-Module ausgelagert worden.

mod_ssl
Das Modul stellt Funktionen zur Verfügung, um eine Verbindung zu verschlüsseln und einen Benutzer sicher zu authentifizieren.

mod_vhost_alias
Das Modul stellt die Funktionalitäten zur Verwendung der konfigurierten virtuellen Hosts zur Verfügung.

3. Konfiguration und Verwendung

In diesem Kapitel möchten wir die wichtigsten Einstellungen und Optionen von Apache untersuchen und ihren Einsatzzweck erläutern. Konkrete Syntax der httpd.conf soll hier aus Platzgründen nicht untersucht werden, da primäres Ziel dieser Ausarbeitung ein Überblick der Möglichkeiten des Apache-Webservers ist.

3.1. Konfigurationsdatei httpd.conf

Die Entwickler des Apache haben vereinbart, dass alle Einstellungen des Webservers unter dem Pfad und in der Datei

.../apache/ conf/httpd.conf

vorgenommen werden; bei früheren Versionen (1.3.x) gab es im selben Verzeichnis auch andere Dateien. (Mit der Include-Anweisung kann man auch selbständig eine hierarchische Struktur der Konfigurationsdateien organisieren.)

Bei der Notwendigkeit, den Webserver mit unterschiedlichen Konfigurationen zu starten, kann man auch andere Konfigurationsdateien beim Starten des Apache vorgeben. Dies erlaubt mehrere Betriebsmodi bzw. Tests. Eingabe:

./apachectl start –f <conf_file>

Weiterhin kann jedes Verzeichnis eine Datei „.htaccess“ enthalten. In dieser Datei dürfen bestimmte Einstellungen des Apache speziell für dieses und darunter liegende Verzeichnisse vorgenommen werden. So können einzelne Benutzer, die die globalen Einstellungen in der httpd.conf nicht verändern dürfen, in ihrem eigenen Verzeichnis abweichende Optionen vorgeben. Dies ist nur in dem Rahmen möglich, den die httpd.conf definiert.

3.2. DSO – Dynamic Shared Objects

Syntax: LoadModule foo_mod.so

Dank dieser Technik ist es möglich, dem Webserver zur Laufzeit neue Funktionsmodule zur Verfügung zu stellen. Der dynamisch geladene Code wird in den Adressbereich des gestarteten Programms eingeblendet.
Um dies zu erreichen, muss Apache mit der Konfigurationsanweisung „--enable-so“ installiert werden.
Bei Bedarf kann man während der Kompilierung alle Apache-Module (bis auf mod_so, das für die DSO-Objekte zuständig ist) als DSO erzeugen lassen, damit noch mehr Flexibilität beim Serverbetrieb entsteht: nur mit einer Installation können sehr verschiedene Serverinstanzen gestartet werden, deren Aufbau beliebig anpassbar ist.

Bei späterem Hinzufügen der Module von Drittanbietern wie PHP oder Perl muss man sie mit dem Apaches mitgeliefertem Tool APXS erzeugen. Z.B. in der Installationsanleitung von PHP findet man entsprechende Optionen für .configure, wie man auf apxs Bezug nimmt.

Der Nachteil der DSO-Technik besteht hauptsächlich in der etwas langsamerer Programmausführung (20% längere Startzeiten und 5% längere Ausführungszeiten).

3.3. Grundlegende Direktiven

3.3.1. Server Type

Syntax: standalone|inetd

Es gibt zwei Möglichkeiten, Apache zu betreiben. Als Standalone wird Apache ganz normal als eingenständiges Programm ausgeführt, bei der „inetd“-Option wird er bei jeder Anfrage neu gestartet durch den Internetdaemon inetd. Der zweite Ansatz kann aus Performanzgründen ausgeschlossen werden.

3.3.2. User/Group

Syntax: User UserName / Group GroupName

Bestimmt den Benutzer, im Kontext dessen der Server ausgeführt wird. Somit verfügt Apache über die gleichen Zugriffsrechte im Betriebssystem wie der eingestellte Benutzer.
Aus Sicherheitsgründen sollte der Webserver daher nie mit uneingeschränkten Zugriffsrechten ausgeführt werden, deren Missbrauch katastrophale Auswirkungen haben könnte. Beim Systemstart wird aber Apache vom root gestartet. Durch diese Direktiven wechselt Apache in ein anderes Benutzerkontext. Stadardeinstellung des Benutzernamens für Apache ist „nobody". (Benutzergruppe heisst auch „nobody")

3.3.3. Port und Listen

Syntax: Port PotNumber / Listen [IP:|Host:]Port

Die Default-Einstellung ist der für http vorgesehene Port 80; auf diesem Port und allen vorhandenen Netzwerkschnittstellen wird Apache ohne weiteren Einstellungen auf Client-Anfragen warten.

Mit der Direktive Listen kann man festlegen, auf welchen Ports zusätzlich gelauscht werden muss, bei Bedarf in Kombination mit einer Netzwerkschnittstelle oder Host-ID

3.3.4. ServerRoot

Syntax: ServerRoot Path

Gibt an, wo Apache installiert wurde: dadurch wird der Ort der Logfiles und Konfigurationsdateien sowie (per default) der Webdokumenten bestimmt.

3.3.5. DocumentRoot

Syntax: DocumentRoot Path

In der Anfangskonfiguration zeigt der Pfad zu den Webdokumenten auf den Ordner „htdocs", der sich im ServerRoot-Ordner befindet. Man kann natürlich die Dokumente woanders abspeichern, dies kann die Verwaltung erleichtern. Insbesondere interessant wird diese Einstellung dann, wenn man mit VirtualHosts mehrere Dokumentpfade verwenden muss.
So wird z.B. Apache unter /usr/local/apache installiert; die Documentverzeichnisse der virtuellen Server können unter /var/www/virtualhost_name/ abgespeichert werden.

3.3.6. ServerName

Syntax: ServerName FQDN

Diese Einstellung wird nicht unbedingt direkt verwendet, muss sie vorgenommen werden.
An dieser Stelle muss man unbedingt entweder eine im Nameserver eingetragene TLD-Adresse notieren oder mindestens die IP-Adresse eintragen:

sonst kann Apache nicht gestartet werden. FQDN steht für „fully qualified domain name“, damit ist www.domain.tld gemeint.

3.4. Zugriffskontrolle und das Dateisystem

Mit den Direktiven wie <Directory>, <Files>, <Options> ist es möglich, für ausgewählte Bereiche eine große Vielfalt an unterschiedlichen Rahmenbedingungen für den Serverbetrieb zu schaffen.
Die zugehörige Vorgehensweise ist:

1) Bereiche einteilen, von direkter Angabe bis zu komplexen regulären Ausdrücken besteht große Auswahlmöglichkeit, um welche Pfade es geht.
2) Entscheiden, welche Funktionalitäten/Rechte mit diesen Bereichen assoziiert werden
3) Entscheiden, ob Benutzer eigene Konfigurationsdateien anlegen dürfen und welche Direktiven sie dort verwenden dürfen (evtl. abweichende Anweisungen für die Benutzerverzeichnisse)

Durch überlegte Kombination der Direktiven können sehr feine Abstufungen im Funktionsumfang bzw. Sicherheitsmodell, die mit einzelnen Benutzern sowie Benutzergruppen verbunden sind.

- AllowOverride: erlaubt das Überschreiben der Servereinstellungen durch .htaccess-Dateien; AllowOverride None würde dies unterlassen
- Options: konkrete Optionen wie ExecCGI/Includes (CGI- oder SSI-Ausführung), FollowSymLinks (Auflösung symbolischer Links)

Bei dieser Fülle an diversen Einstellungsmöglichkeiten sollte man sich bei jeder Einstellung im Klaren sein, welche Sicherheitslücken man damit schaffen kann. Einstellungen des Servers sollen nach einem durchdachten, vorher angefertigtem Sicherheitskonzept vorgenommen werden – insbesondere, wenn der Webserver von mehreren Benutzern verwendet wird.

3.5. Handler

Syntax: AddHandler Name Extention

Mit Handler wird der Mechanismus bezeichnet, der die Ausführung von bestimmten Dokumenttypen assoziativ steuert. Durch die Handler werden Anfragen an vordefinierte Module weitergeleitet.
Handler erlauben es, beim Aufruf bestimmter Dateien bzw. Dateitypen zugehörige Aktionen durchzuführen. Der wichtigste Handler ist der Handler „cgi-script“, denn er erlaubt Erzeugung des Outputs außerhalb von Apache.
Andere Handler sind für interne Verarbeitung (wie server_parsed für SSI) bzw. für spezielle Anfragen gedacht: mit dem Handler server-info erhält man Informationen über die Server-Einstellungen.

3.6. Logfiles

Die zwei wichtigsten Typen von Protokolldaten sind AccessLog und Error-Log. Mit AccessLog können ausführliche Informationen über Zugriffe auf den Server sowie seine Auslastung gespeichert werden. ErrorLog speichert Informationen über die Fehler beim Serverbetrieb; bei Problemen findet sich dort oft die Erklärung, denn nach außen, also den Webclients, dürfen keine wichtigen Informationen mitgeteilt werden.

Mit der Einstellung LogLevel für die Fehlermeldungen kann man einstellen, welche Klassen von Ereignissen protokolliert werden sollen. In dieser Auflistung enthält jede Stufe die vorhergehende.

Emerg,alert, crit: nur Notfall/beunruhigende Situation
Warn, notice: potentiell gefährlich bzw. neutral
Info, debug: Mitteilung bzw. für Entwickler

Die empfohlene Einstellung ist „crit".

Die Einstellung LogFormat steuert das Aussehen und die Ausführlichkeit der einzelnen Einträge.
Zu bedenken ist, dass bei Servern mit hohen Zugriffszahlen schnell enorme Datenmengen an Protokolldaten entstehen können. Sollte jedoch jemand einen Phishing-Angriff vorbereiten oder einfach nach Schwachstellen in einer Webanwendung suchen, können auch äußerlich unbedeutende Zugriffe als Vorwarnung dienen.

Zur Lösung dieses Dilemmas empfiehlt es sich, auf dem Webserver einen Logfile-Analyzer zu installieren. Ein solches Programm sollte nicht nur Statistiken mit Grafiken über Anzahl der Zugriffe erzeugen, sondern auch eventuelle Einbruchsversuche frühzeitig erkennen.

3.7. URL Rewriting

Die Funktionalität des „URL Rewriting" erlaubt ggf. Veränderung der Anfrage und Weiterleitung zu anderen Zielen anhand beliebiger Regeln.

Zur Zeit stellt Apache zwei Module zur Verfügung, *mod_alias* und *mod_rewrite*. Mod_alias hat sehr begrenzte Funktionalität und soll durch *mod_rewrite* ersetzt werden. Der einzige Nachteil des Letzteren ist eine sehr hohe Komplexität in der Konfiguration (vgl. das Epigraph auf der *mod_rewrite* Homepage: *„ Despite the tons of examples and docs, mod_rewrite is voodoo. Damned cool voodoo, but still voodoo."*) [Mod_rewrite]

Damit können folgende beispielhafte Aufgaben erledigt werden:

- Weiterleitung des Clients anhand der Browser-Version (z.B. zur anderer Webseitenversion)
- Aufteilung der Last in einem Webserver-Cluster, wo kein Bezug einer konkreten URL zum physikalischen Ort existiert
- Strukturelle Anpassungen der URL's

3.8. Konfiguration von CGI und suEXEC

CGI-Skripte kann man auf 3 Arten konfigurieren:

1. alle CGI-Skripte befinden sich in einem Verzeichnis z.B. */usr/local/etc/httpd/cgi-bin* und alle Dateien in diesem Verzeichnis werden als CGI-Skripte behandelt. Schreibrechte hat nur der Administrator

2. CGI-Skripte dürfen in allen Verzeichnissen enthalten sein, und werden durch ihre Dateiendung .cgi erkannt. Dazu setzt man den cgi-script-Handler, der dafür sorgt, dass Dateien mit Endung .cgi als CGi-Skript interpretiert und ausgeführt werden.
Damit ein SGI-Skript hier nicht unter der User- und Group-ID des Apaches gestartet wird, verwendet man den suEXEC-Wrapper.
Der ist dafür zuständig, dass ein CGI-Skript eines Nutzers unter dessen User-ID ausgeführt wird, damit die Skripte mit den Rechten und Privilegien des jeweiligen Eigentümers laufen, so dass die Programme keinen vollen Zugriff auf das Home-Directory haben.
Der Einsatz vom suEXEC-Wrapper ist auch bei virtuellen Servern möglich.

3. Einzelne Dateien können als CGI-Skript zugelassen werden

Die Fehler in CGI-Skripten sind nur schwer nachzuvollziehen, da die Fehlermeldung nur vom Apache zurückgegeben wird. Das CGI-Modul jedoch bietet die Möglichkeit eines Logfiles, in dem die Fehlermeldungen reingeschrieben werden, so dass der User sich unter dieser Logfile, die Fehlerausgabe anschauen kann.

3.9. Umgebungsvariablen

Der Apache Server stellt eine bestimmte Anzahl von Umgebungsvariablen zur Verfügung(z.B. SERVER_NAME), die den CGI-Skripten und den SSI-Dokumenten zur Verfügung stehen. Zusätzlich können noch weitere Variablen definiert werden.
Entsprechende Module erläutere ich im folgenden:

mod-env-Modul
Über die dort angebotenen Konfigurationsanweisungen, wie SetEnv können beliebige Variable erstellt werden,
über PassEnv, Variablen an CGI und SSI weitergereicht werden und über UnsetEnv Variablen gelöscht werden.

mod-setenvIf-Modul
hier können aufgrund von verschiedenen Attributen der Anfrage wie Remote-Addr, Remote_Host und Attribute des Http-Headers, sowie Attribute von Variablen die durch eine andere SetEnvIf-Anweisung gesetzt wurde, die Umgebungsvariablen gesetzt werden. Ausserdem können interne Variablen gesetzt werden, die das Verhalten des Apaches kontrollieren, so dass z.B. fehlerhaftes Verhalten eines schlecht geschriebenen Webclients, mit den vom Server übermittelten Seiten, besser zurechtkommt.
Seit Version 2.0 kann als Attribut auch ein regulärer Ausdruck verwendet

werden, wobei beim SetEnvIfNoCase-Modul beim Vergleich des regulären Ausdrucks mit dem Wert des Attributs, nicht zwischen Groß- und Kleinschreibung unterschieden wird.

mod_unique_id
Dieses Modul erzeugt für jede Client-Anfrage einen eindeutigen 19-stelligen aus Zahlen und Buchstaben bestehenden Identifier, der innerhalb der Variable UNIQUE_ID gespeichert wird. Die Generierung des Identifiers erfolgt mithilfe der IP-Adresse der Client-Anfrage, der Prozess-ID und weiteren Komponenten. Ausserdem wird dieses Modul zur Erzeugung von Sessions durch PHP verwendet.

mod_rewrite
-Es besteht die Möglichket auch über RewriteRule-Anweisungen Umgebungsvariablen zu setzen.

3.10. User-Tracking (Cookies)

Dieses Modul wird verwendet um das Verhalten der Benutzer zu überwachen und zwar im Sinne von, wie oft hat ein bestimmter User eine Seite besucht, wie groß ist der Zeitabschnitt zwischen zwei Anfragen, welche Seiten wurden bevorzugt, usw. Dazu verwendet man Cookies(Cookie-Header), die bei jeder neuen Anfrage vom Webserver mitgeschickt werden, falls nicht schon ein Cookie auf der Client-Seite gesetzt wurde. Die Aktivitäten der User werden dann in der vom Modul erzeugten Protokolldatei festgehalten.

Es gibt drei verschiedene Syntaxe, die für Cookies verwendet werden können und zwar Netscape, Cookie(RFC2109-Standard) und Cookie2(RFC2965-Standard). Über die *Cookie-Style*-Anweisung kann eingestellt werden, welche der genannten Syntaxe das User-Tracking Modul verwenden soll. Empfohlen wird hier der RFC2109 Standard, da dieser von fast allen Webbrowsern unterstützt wird.

Wenn *Cookie-Expires* nicht gesetzt wurde, werden die Cookies nur transient im Speicher gehalten. Damit für eine bestimmte Zeit diese persistent gespeichert werden, kann man über Cookie-Expires angeben, wie lange die Cookies innerhalb der Browser-Datei als gültig anzusehen sind.

3.11. Proxy

Proxies bzw. Caching-Proxies dienen als Vermittler zwischen den anfragenden Clients und dem Webserver. Wenn ein Client eine Anfrage schickt und die angefragten Seiten vorher noch nicht in den Proxy-Cache geladen wurden, werden vom Webserver die Daten an den Proxy übermittelt, zwischengespeichert und dann an den Client weitergeleitet. Möchte nun ein weiterer Client die gleichen Seiten sich betrachten, kann seine Anfrage nun schneller bearbeitet werden, da die Kopie der Daten vom Cache geladen werden kann und diese nicht erst neu beim Webserver angefordert werden muss. Das Modul *mod_proxy* stellt diesen Proxy zur Verfügung, welcher die Protokolle Http, Ftp und Connect (für SSL) unterstützt. Der Cache ist seit der 2. Version nicht mehr im *mod_proxy-Modul* integriert, sondern wurde in die Module

mod_cache, *mod_mem_cache* und *mod_disk_cache* ausgelagert.

Über das *ProxyTimeout* kann eingestellt werden, wie lange in den Proxy-Cache geladene Seiten zwischengespeichert werden. Das voreingestellte Zeitlimit beträgt 300 Sekunden.

Soll der Apache in ein Proxy- oder Cache-Netzwerk integriert werden oder ein Proxy wird vor- bzw. nachgeschaltet, bedient man sich der Proxy-Remote-Anweisung, die auf Basis des Protokolls festgelegt werden kann oder auf Basis einer URL. Bei der zuerst genannten Variante kann anhand eines Protokolls festgelegt werden, welcher Cache genutzt wird. Bei der zweiten, kann falls eine bestimmte Seite geöffnet werden soll, ein bestimmter Proxy definiert werden. An diesen Proxy werden dann alle Anfragen weitergeleitet. Zusätzlich kann man noch über die NoProxy-Anweisung festlegen, ob eine Anfrage, abhängig von der angegebenen Adresse(auch Domain- oder Subnetz-Angabe möglich), über den Proxy laufen soll oder der Webserver direkt angesprochen werden soll.

Über *ProxyPassReverse* kann der Server als Reverse-Proxy eingesetzt werden, so dass angefragte Daten von mehreren Webservern stammen können die hinter diesem Proxy geschaltet werden können, ohne das der User etwas davon mitbekommt.

Durch die *ProxyBlock*-Anweisung können mehrere Sites oder die gesamte Domain für Zugriffe gesperrt werden, so dass Nutzer nicht darauf zugreifen können.

Der Proxy reicht im Normalfall die Fehlermeldungen die vom Apache ausgehen an die Clients weiter. Mit der ProxyErrorOverride-Anweisung ist es jedoch möglich, eine eigene Fehlermeldung zu schreiben. Im Fehlerfall würde dann statt der Apache-Fehlermeldung, die selbstdefinierte an die Clients verschickt werden.

Die Nutzung des Proxys kann auch auf bestimmte Personen eingegrenzt werden z.B. auf Nutzer der eigenen Domain, indem man die Proxy und ProxyMatch Anweisungen verwendet.

Weiterhin ist es möglich, über die ProxyAuthentifizierung nur Leuten den Zugriff auf den Webserver über das Proxy zu gewähren, die entsprechendes Passwort besitzen, um sich somit zu authentifizieren oder Leuten die einer bestimmten Domaine angehören.

3.12. Virtual Hosts

Syntax: <VirtualHost Host[:Port]...>...</VirtualHost>

Mit der Direktive „VirtualHost“ kann man mit einer Apache-Installation mehrere virtuelle Webserver betreiben. Das bedeutet, dass nach außen tatsächlich mehrere Webserver simuliert werden. Mehrere Zuordnungen der Servernamen zu den WWW-Präsenzen (DocumentRoot-Pfade) sind auf einer Maschine mit einer Apache-Hauptinstanz möglich. Dadurch können sehr

viele Ressourcen gespart werden. So werden bei den Webhosting-Unternehmen mehrere Kundenaccounts auf je einem Server untergebracht.

Die VirtualHost-Direktiven dürfen nicht in den .htaccess-Dateien auftauchen, d.h. virtuelle Webserver können sinnvollerweise nur durch den Serveradministrator verwaltet werden.
Apache bietet mehrere Möglichkeiten zur Virtualisierung des Webbetriebs an: Virtuelle Hosts können auf einer Port-, Hostname- oder IP-Angabe basieren.
Angaben mehrerer IP-Adressen ist nur dann sinnvoll, wenn der Server, auf dem Apache ausgeführt wird, über mehrere Netzwerkschnittstellen verfügt. Man kann davon ausgehen, dass aus Kostengründen nur selten Maschinen mit mehreren physikalischen Netzwerkschnittstellen installiert werden.
Portbasierte Webserver sind auch eher selten und trotzdem sehr bequem wegen einer einfachen Konfiguration im Vergleich zu den anderen Möglichkeiten: man muss nur die Konfigurationsdatei des Apache ändern und keine Anpassungen im System vornehmen. Für die Entwicklung und mehrere Webpräsenzen unter der gleichen Host-ID (Hostname oder IP-Adresse), z.B. Dokumentation zum Projekt ist diese Möglichkeit ziemlich interessant. Der Nachteil hier ist die Abweichung von der Standard-Portnummer 80.
Die dritte Möglichkeit ist etwas aufwendiger zu konfigurieren, dafür ist die Transparenz der virtuellen Webserver auf günstigem Wege erreicht. Virtuelle Hosts werden verschiedenen Hostnamen zugeordnet, dafür muss der Rechner über DNS zusätzlich konfiguriert werden, so dass Anfragen an verschiedene Hostnamen auf die gleiche IP umgeleitet werden.
Durch namensbasierte virtuelle Webserver wurde das Webhosting von Millionen von WWW-Präsenzen möglich, sogar jeder Kunde eines Webhosters kann selbst virtuelle Server einrichten, falls er mehrere Domains bestellt. Der Vorgang wird durch Automatismen erleichtert; intuitive Web-Konfigurationsschnittstellen erlauben auch unqualifizierten Personen, das Webhosting zu verwalten.

Die minimale Konfiguration eines Virtuellen Hosts ist die Angabe der ID, durch die er erreichbar sein soll (Port, IP oder Hostname) sowie eine Angabe der DocumentRoot-Variable. Alle anderen Einstellungen werden aus der Datei httpd.conf geerbt; dennoch kann man fast alle Einstellungen des Hauptservers im Block „VirtualHosts“ überschreiben.

Bei sehr vielen (>100) gleich konfigurierten (bis auf das Dokumentverzeichnis) Virtual Hosts kann das Modul *mod_vhost_alias* eingesetzt werden. Seine Aufgabe besteht darin, Anfragen an virtuelle Webserver an die vorher angelegten, gleichnamigen Verzeichnisse abzubilden, z.B. bei einer Anfrage an *http://www.domain.tld/page.html* wird nach */usr/local/apache/htdocs/www.domain.tld/page.html* gesucht.

4.0 Common Gateway Interface(CGI)

Damit Webseiten dynamisch und interaktiv gemacht werden können, bedient man sich dem Common Gateway Interface. Anwendungsbeispiel hierfür ist z.B. der Warenkorb. Durch das Interface ist es möglich, durch clienseitigen Aufruf von CGI-Skripten auf Server-Seite, aufgrund von übergebenen Parametern, Dateien zu verarbeiten und deren Ausgabe als Html-Kode über den Webserver an den Webbrowser weiterzuleiten. Außerdem ist auch auf dem Server die Durchführung von Datei-Operationen möglich.

CGI-Skripte können in jeder beliebigen Programmiersprache vorliegen. Voraussetzung hierfür ist das auf der Serverseite ein entsprechender Laufzeit-Interpreter vorhanden sein muss. Die Wahl der Programmiersprache hängt davon ab, ob die Ausführung des Programms effizient sein soll z.B. wie in C, oder ob die Programmiersprache komfortabel sein soll, wei bei den interpretierenden Sprachen durch umfassende Unterstützung durch Text- und Zeichenverarbeitung, z.B. Perl, Pyton etc. .

Der Aufruf eines CGI-Skriptes kann über die die beiden Http-Methoden GET oder POST aufgerufen werden. Bei der GET-Methode werden die Parameter über die URL an den Server weitergereicht. Die übertragenen Parameter werden in die Umgebungsvariable Query-String abgespeichert und können vom CGI-Skript abgefragt werden. Bei der POST-Methode werden die Daten über den Entity-Body übergeben, und können dann über die Standardeingabe des Skriptes übergeben werden.

Der durch das CGI-Skript generierte Html-Kode sollte zumindest das Html-Gerüst und den Content-Type-Header beinhalten, so dass der Webbrowser weiß welche Art von Daten er übermittelt bekommt. Ansonsten wird der im Apache eingestellte Default MIME-Type hinzugefügt. Die weiteren Http-Header Connection und Date werden auch automatisch vom Apache hinzugefügt und können durch NPH-Skripte umgangen werden.

Im Umgang mit CGI-Skripten sollte man vorsichtig sein, da diese leicht zur Sicherheitslücke werden können. Deswegen ist auf folgende Dinge zu achten:

-Der Aufruf einer Shell aus dem CGI-Programm aus sollte vermieden werden.
-Der Eingabepuffer sollte immer festlegbar sein, so dass keine Buffer Overflows entstehen können. Deswegen Vermeidung von Funktionen wie gets(), da Größe nicht festlegbar ist.
-Übergebene Daten sollten nach nicht zu erwartende Zeichen hin überprüft werden, z.B. durch regulären Ausdruck
-Bei Dateizugriffen sollte das Metazeichen „ .. “ nicht verwendet werden, da man sonst auf übergeordnete Verzeichnisse zugreifen könnte.
-usw.

5. HTTPD-Daemon

5.1. Konzeptuelle Architektur des Apache

Apache wurde vor seiner Entwicklung nicht formal spezifiziert, dennoch ist es möglich, anhand des Quellcodes und Kommentaren sich ein Bild über den internen Aufbau des Apache zu machen.
Apache ist eine Weiterentwicklung des NCSA-Webservers von R.McCool. Seit dem ersten Release weist Apache eine modulare Architektur auf. Dadurch können mehrere Entwickler auch ohne tiefe Kenntnisse über interne Struktur des Apache schnell und unabhängig von den Anderen neue Funktionalität hinzufügen.

Apache ist ein plattformunabhängiger, leistungsfähiger Webserver, der durch Abstraktion der betriebssystemspezifischen Funktionen überall einsetzbar ist.

Auf der obersten Architekturebene besteht Apache aus zwei Schichten: der eigentliche Webserver und die OS-Schicht. Die OS-Schicht kapselt alle Funktionen, die für die Kommunikation mit einem Betriebssystem notwendig sind und macht den Apache betriebssystemunabhängig. Diese Schicht ruft niemals Funktionen der anderen Schicht auf.

Die Webserverschicht enthält die mittlere Ebene der Architektur und verwendet implizite Aufrufe. Das bedeutet, dass die Modulentwickler ihre Module für den Aufruf durch den Apache-Kern in speziellen Konfigurationsdateien registrieren müssen.
Die Abarbeitung einer Client-Anfrage besteht aus mehreren Schritten, etwa

- Anfrage lesen
- URL umformen
- Zugriffsrechte prüfen
- Dokument bzw. Fehlermeldung zurückgeben
- Protokollieren

Jeder dieser Schritte definiert ein Ereignis, bei dem geprüft wird, ob Module existieren, die Interesse an der Teilnahme haben. Hat ein Modul Interesse (manifestiert durch die Konfigurationsdateien), so wird es implizit aufgerufen und führt seine Funktion aus. Abhängig vom Ergebnis der Bearbeitung meldet das Modul gegenüber dem Apache-Kern ein „OK“ oder einen HTTP-Fehlercode.
Je nach Abarbeitungsschritt kann die gesamte Ausführung im Falle eines „OK“ weitergereicht oder unterbrochen werden. Für alle Schritte gilt jedoch, dass schon der erster Fehler den Abbruch und Fehlermeldung gegenüber dem Client erzwingt.
Es wurde eine Art Nachrichten-Pool definiert, damit die Module nicht nur mit dem Kern, sondern auch miteinander kommunizieren können. Ein Modulentwickler muss sich um die Ressourcenverwaltung nicht kümmern – sie wird vom System erledigt. Nach der Abarbeitung einer Anfrage werden alle belegten Ressourcen wieder freigegeben.

Die unterste Ebene der Apache-Architektur ist der Apache-Kern, in dem die wichtigsten Funktionen ausgeführt werden. Auf dieser Ebene werden Konzepte der Pipes und Filter eingesetzt.
Beim Starten des Apache analysiert der Konfigurationsfilter des Kerns die Konfigurationsdatei, wendet Direktiven an und erzeugt die sog. „Dispatch Table“. Die “Dispatch Table“ enthält Informationen, in welcher Reihenfolge welche Module aufgerufen werden sollen, wenn eine Client-Anfrage bearbeitet wird.
Sobald eine Client-Anfrage ankommt, wird mit dem Protokoll-Filter geprüft, ob es sich um eine gültige HTTP-Anfrage handelt. Erst dann werden durch das Dispatcher-Filter im Server geladene Module in einer Pipeline aufgerufen (auf der höheren Abstraktionsschicht entspricht dies den impliziten Aufrufen). Anschließend werden die Log-Dateien geschrieben und eine Antwort an den Client versendet.

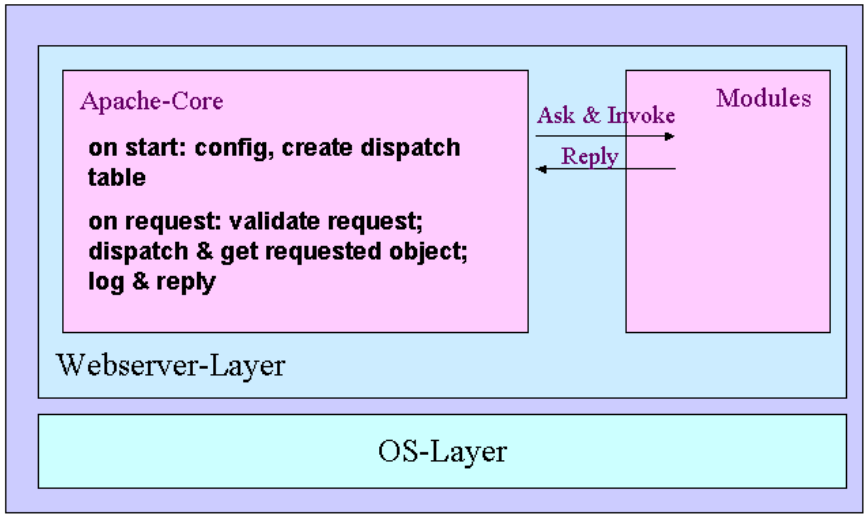

Abb. 1 : Apache-Architektur

5.3. Apache aus der Sicht des Betriebssystems

Beim Starten des Apache entsteht ein Hauptprozess, der weitere Kindprozesse erzeugt. Der Hauptprozess selbst bearbeitet keine Anfragen, seine Aufgabe ist nur die Kindprozesse zu koordinieren.
Es muss immer eine bestimmte Anzahl an nicht beschäftigten Kindprozessen geben, damit neue Anfragen sofort und gleichzeitig bearbeitet werden können. Wird ein Kindprozess belegt, erzeugt der Hauptprozess einen Neuen. Sollte eine Situation auftreten, dass alle freien Kindprozesse gleichzeitig belegt werden, werden vom Hauptprozess jedes Mal immer größere Mengen von Kindprozessen erzeugt: zuerst 2, dann 4, 8 usw.
Aufgrund dieser Vorgehensweise gehört Apache zur Kategorie der sogenannten „Pre-Forking“ Webserver.

Alle oben beschriebene Eigenschaften sind über die Datei httpd.conf konfigurierbar, und zwar mit den folgenden Direktiven:

StartServers: Anzahl der Kindprozesse beim Start. Default-Einstellung ist 5.
<Min|Max>SpareServers: maximale bzw. minimale zugelassene Anzahl der leer laufenden Kindprozessen. Bei zu vielen leeren Prozessen werden einige von ihnen beendet.

MaxRequestsPerChild: gibt vor, wie viele Anfragen ein Kindprozess in seinem Lebenszyklus abarbeiten darf. Wird die angegebene Anzahl erreicht, ersetzt diesen Prozess ein Neuer. Dadurch können potentielle Speicherlecks der Servermodule umgangen werden. Bei sehr vielen Anfragen kann diese Option den Apache etwas verlangsamen.

5.4. Neuerungen in Apache 2.0

Erweiterungen des Kerns:

- Unterstützung der PThreads: erlaubt Ausführung als Multiprozess, führt zu besserer Skalierbarkeit

- Neues Bild-System: Anpassung des Konfigurationssystems an andere Packages
- Multi-Protokoll Unterstützung
- neues API: Verbesserung der Bugs gegenüber 1.3, weitgehende Automatisierung der Dispatch Table (Priorität der Module, Reihenfolge)
- Verwendung standardisierder Perl-Bibliothek für RegExp

Erweiterungen der Module:

- neue Module: mod_ssl, mod_dav, mod_auth_ldap, mod_deflate
- umgeschrieben/erweitert: mod_proxy, mod_headers und andere. S. dazu Abschnitt 2.3.

6. Quellen

[Apache 2004] http://httpd.apache.org/

[Eilebrecht 2000] Eilebrecht, L.: “Apache Web-Server”, 3. Auflage, 2000

[Eilebrecht 2003] Eilebrecht, L.: “Apache Web-Server”, 5. Auflage, 2003

[Mod_rewrite] http://httpd.apache.org/docs/mod/mod_rewrite.html

[Hassan 1999] Hassan, A.: "Apache: Conceptual Architecture", http://www.undergrad.math.uwaterloo.ca/~aeehassa/cs746/apache1.htm, 1999

[Parry 1999] Parry, T.: “Conceptual Architecture of Apache” http://plg.uwaterloo.ca/%7Etoparry/746/a1/a1.htm, 1999

[Sieler-Homke 2003] Sieler-Homke, J: Apache, 2003