TECHNISCHE UNIVERSITÄT DRESDEN

Center for Information Services and High Performance Computing (ZIH)

Linux Cluster in Theorie und Praxis

Verteilte Dateisysteme & Services

02. November 2009

INF 1046
Nöthnitzer Straße 46
01187 Dresden
0351 - 463 38783

Andy Georgi

ZIH
Center for Information Services & High Performance Computing

Foliensatz

Verfügbarkeit der Folien

Vorlesungswebseite:
`http://tu-dresden.de/die_tu_dresden/zentrale_einrichtungen/zih/lehre/ws0910/lctp`

Definition I

Verteilte Dateisysteme

Ein verteiltes Dateisystem erlaubt den Zugriff auf Dateien eines oder mehrerer entfernter, über ein Rechnernetz verbundener Server.

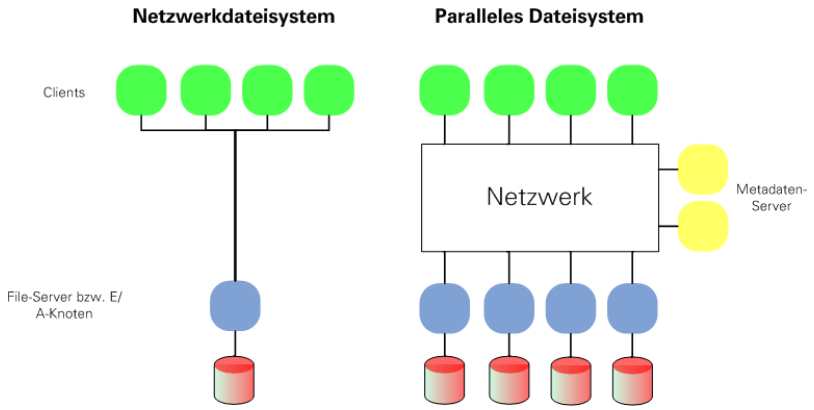
Definition II

Ausprägungen

In **Netzwerkdateisystemen** stellt ein Server Daten zur gemeinsamen Nutzung für verschiedene Benutzer über ein Netzwerk bereit.
Beispiele: AFS, NFS, SMB

Parallele Dateisysteme dagegen halten die Daten auf mehreren Servern und bilden diese auf ein ganzheitliches virtuelles Dateisystem ab. Sie verfügen weiterhin über separate Metadaten-Server. Vor dem Zugriff eines Clients müssen die zugehörigen Berechtigungen vom Metadaten-Server erfragt werden.
Beispiele: Lustre, PVFS, Ceph, GFS

Definition III



Auswahlkriterien

- **Zugriff:** Identische Zugriffe auf lokale und entfernte Ressourcen
- **Ortstransparenz:** Der physikalische Speicherort ist für Nutzer transparent
- **Konsistenz:** Verhalten bei simultanen Zugriffen durch verschiedene Benutzer
- **Verhalten im Fehlerfall:** Wiederherstellung nach Ausfall; Sichtbarkeit für Benutzer
- **Replikation:** Sind Daten redundant vorhanden
- **Skalierbarkeit:** Verhalten der Zugriffsgeschwindigkeit bei steigender Anzahl von Servern und Clients
- **Heterogenität:** Unterstützung unterschiedlicher Hard- und Software
- **Administration:** Dezentral oder zentral
- **Sicherheit:** Authentifizierung beim Server; Verschlüsselung der Pakete

Benutzersicht

- Explizite Knotennennung:
  - Der Hostname wird dem lokal gültigen Pfad vorangestellt (z.B. `headnode:/foo/bar`)
  - Einfache Implementierung
  - Fehlende Zugriffs- und Ortstransparenz
- Einbindung:
  - Entfernte Dateisysteme werden durch Einbindung in den lokalen Verzeichnisbaum importiert
  - Zugriffs- und Ortstransparenz sind gegeben
  - Die Sicht der einzelnen Knoten auf das Dateisystem ist abhängig vom ,,mount-point"
- Globaler Namensraum:
  - Garantiert eine identische Sicht aller Knoten auf das Dateisystem
  - Lokal gültige Pfadnamen sind auch auf anderen Knoten gültig
  - Aufwändige Implementierung und höherer Overhead

Deskriptoren

- Abbildung symbolischer Namen auf systeminterne Bezeichner (*unique file identifier (UFID)*)
- Dient der eindeutigen Identifikation der Datei und deren Gruppe
- Eindeutigkeit kann mit Hilfe einer zentralen Autorität oder durch anfügen einer großen Zufallszahl erreicht werden
- Möglichst hohe Fälschungssicherheit

Caching I

- Zur Minimierung des Datenverkehrs ist eine lokale Zwischenspeicherung sinnvoll
- Schreibstrategien:
  - Write-through:
    - Cache wird lediglich zum Lesen benutzt
    - Schreiboperationen werden direkt an den Server weitergegeben
  - Write-on-close:
    - Vollständige Übertragung bei erstem Zugriff
    - Lokale Zwischenspeicherung aller Änderungen
    - Zurückschreiben beim Schließen der Datei
  - Asynchronous:
    - Änderungen werden blockweise Zurückgeschrieben
- Konsistenz:
  - False sharing:
    - Keine Konsistenz
    - Bei simultaner Bearbeitung eines Blockes durch mehrere Benutzer können Schreibvorgänge verloren gehen

Caching II

- One copy semantics (UNIX):
  - Jede Änderung einer Datei ist sofort für alle Prozesse sichtbar
  - Problem: ,,Sofort" ist in einem verteilten Dateisystem relativ
- Session semantics:
  - Veränderungen finden zunächst lokal im Adressraum des Prozesses statt
  - Beim Beenden der Sitzung werden die Änderungen global sichtbar
- Unveränderliche Dateien:
  - Änderungen nur durch Erstellung einer modifizierten neuen Datei
  - Erlaubt effiziente gemeinsame Nutzung und Replikation der Dateien, ist allerdings nicht effizient für große Dateien
- Synchronisation:
  - Das Dateisystem bietet Schreib- und Lesesperren an
  - Der Schreibvorgang muss explizit über ,,Commit"-Operationen eingeleitet werden

Stateless vs. Stateful

Stateless Server

- Fehlertolerant
- Auf- und Abbau von Nutzungsbeziehungen entfällt
- Kein Overhead durch Statusinformationen
- Keine Beschränkung hinsichtlich der Anzahl geöffneter Dateien

Stateful Server

- Kürzere Requests
- Nachfolgezustände vorhersehbar
- Effizienter
- Verwendung von Lese-/Schreibsperren möglich

Replikation vs. Verteilung

Wird eine Datei lediglich von einem Server angeboten, so stellt dieser ein Bottleneck dar!

Lösung 1: Replikation

- Redundante Datenhaltung
- Pro: Bessere Lastverteilung möglich, wenn Dateien vollständig gelesen werden
- Contra: Erhaltung der Konsistenz bei Schreibvorgängen ist sehr aufwändig

Lösung 2: Verteilung

- Daten werden auf mehrere Server verteilt
- Pro: Effizient wenn unterschiedliche Knoten auf verschieden Bereiche zugreifen
- Contra: Keine Skalierung bei Zugriffen auf gleiche Speicherbereiche

Historie

- 1984 von Sun entwickelt
  - NFSv1 für experimentelle Zwecke (firmenintern)
  - NFSv2 als Release vom weiter entwickelten NFSv1 für externen Gebrauch (UDP/SunRPC als Protokoll)
  - NFSv3 mit Support für TCP Transport Layer
    - Handeln von Files > 2GB möglich
    - Asynchrones Schreiben
    - `READDIRPLUS` für File Handles bei Suchen in Verzeichnis
  - NFSv4 baut auf Erfahrungen von anderen Distributed File Systems auf
    - Security (erstmals Nutzerrechte)
- Bis NFSv3 SunRPC (früher "ONC RPC" Open Network Computing), historisch auch als Konkurrenz zu Apollo Computer
- Gegenstand vieler Kontroversen wie zum Beispiel den Big- vs. Little-Endian Streit
- Später wurde NFS vor allem wegen Sicherheitsbedenken kritisiert
- NFSv4 als Lösung der Sicherheitsprobleme und mit pNFS sogar als paralleles Dateisystem?!?

Architektur - Serverseitig

- Unter Linux: Unterscheidung zw. NFS-Kernel-Server und NFS-Userspace-Server
- Server stellt Daemon (meist `nfsd`) bereit
- `portmap` und `mountd` stellt Zuordnung für Anfragen sicher
- In `/etc/exports` können die zu exportierenden Verzeichnise freigegeben werden
  - Beispiel: `/sample 192.168.0.0/24(rw,no_root_squash,sync)`
  - Rechte sind immer nur für Rechnernamen/IPs zu vergeben (NFSv3)
- Der Server versendet bei `lookup`/`open`-Request Inode-Nummer und Geräte-ID von seinem Massenspeicher
- Für `write`/`read`-Request muss der Client sich diese Informationen merken und sendet sie zusätzlich zu einem Offset an den Server
- Bei NFSv3 ist der Server stateless, d.h. jede Anfrage muss der Client so stellen das der Server unabhängig von vorrangegangener Operation weiß was er zu tun hat

Architektur - Clientseitig

- Client benutzt die üblichen UNIX system calls (`read/write/open/close/mkdir/readdir/...`)
- Der NFS Client bildet diese auf RPC-Calls ab, die dann aber wirkliche file handels des Server zurückgeben

Einhängen entfernter Dateisysteme

Szenario

Auf `headnode` läuft `nfsd` und zugehörigen Dienste. In der `/etc/exports` des `headnode` steht die Zeile:

```
/cluster        192.168.0.0/24(rw,no_root_squash,sync)
```

`node01` habe die IP Adresse 192.168.0.2 und möchte das Verzeichniss einbinden.

- Über `/etc/fstab`
  - `headnode:/cluster  /cluster  nfs  rw,intr  0  0`
  - `rw` - read/write Zugriff
  - `intr` - unterbrechen von Dateioperationen bei Timeout
- Über `mount` Befehl
  - `mount -t nfs -o rw,intr headnode:/cluster /cluster`

DHCP

- **D**ynamic **H**ost **C**onfiguration **P**rotocol
- Adressiert das Problem das viele Rechner im Cluster mit valider Netzwerkkonfiguration versorgt werden müssen
- Server antwortet auf den Broadcast eines Clients mit beliebiger Netzwerkkonfigurationsinformation
  - Server wartet auf UDP Port 67 auf Anfragen
  - Dann kann statisch, automatisch oder dynamisch(-wechselnd) die Konfiguration herausgegeben werden
  - Im statisch/automatischen Fall immer abhängig von MAC Adresse des anfragenden Netzwerkdevices
- Meist IP-Adresszuweisung, Gateway, Netmask, Nameserver und Hostname
- Es können aber auch BOOTP Server übermittelt werden von denen bspw. gebootet werden soll
- NTP, WINS und weitere DNS Informationen sind auch übermittelbar

DHCP Funktionsweise

- Client sendet `DHCPDISCOVER` mit seiner MAC als Broadcast in sein Netz (Achtung!, Router bilden Broadcast Domänen)
- Die Server antworten mit `DHCPOFFER`-Broadcast (Client hat zu dem Zeitpunk noch keine IP) ins Netz und machen dabei IP-Adressen Vorschläge
- Client sendet `DHCPREQUEST` als Broadcast an den Server mit "bestem" Angebot
- Server antwortet dann mit `DHCPACK` und bestätigt die Daten nocheinmal oder sendet `DHCPNAK` für Abbruch
- Es ist dem Client überlassen ob er vor dem setzen der Daten überprüft ob z.B. IP schon im Netz existiert

- Jeder Client bekommt eine gewisse lease time, ist der lease (also die Zuordnung die vom DCHP Server kam) abgelaufen so fragt der Client mit `DHCPREQUEST` den Server nocheinmal per Unicast um eine Verlängerung

DHCP Sicherheit

- DHCP ist explizit nicht dafür entwickelt worden sicher zu sein
  - Broadcasts können mitgehört werden
  - Fakte DHCP-Server kann falsche Leases verteilen
  - Bösartige DHCP Client kann alle Adressen übernehmen die ein Server anbietet und dabei das Netz "leerräumen"
- Sicherheitsbedenken sind zu beachten, trotzdem meist für geschlossenes Cluster zu vernachlässigen

Namensauflösung im Cluster

- Auf Nodes in einem Cluster soll möglichst komfortabel per Namen zugegriffen werden können (nicht per IP-Adresse)
- Eine Möglichkeit bietet die `/etc/hosts` Datei, die dann synchron über alle Nodes und Headnodes gehalten werden muss
  - Für (sehr) kleine Cluster durchaus ausreichend
  - Fehlerquelle wegen evtl. Asynchronitäten
  - Aufwändig zu warten

- Die bessere Möglichkeit ist einen DNS-Server im Cluster zu betreiben, der sich um die Namensauflösung kümmert.
  - Dieser interne DNS-Server kann und sollte losgelöst vom Internet DNS funktionieren
  - Also autorativer Nameserver mit eigener Zone
  - Bei Cluster wird faktisch nur *forward-lookup* benötigt (Name zu IP)
  - Interessant ist die DNS Konfiguration über LDAP

Begriffe des Domain Name Systems I

Einige Begriffe:

- **Zone**: Ausschnitt aus einem DNS-Baum
- **Autorativer Nameserver**: Nameserver der garantiert valide Informationen für eine Zone hat, weil er für "seine" Zone zuständig ist
- **SOA Resource Record**: Start Of Authority, definiert für einen DNS Server seine Zone
- **A Resource Record**: Ordnet einen Namen eine IPv4 Adresse zu
- **AAAA Resource Record**: Ordnet einen Namen eine IPv6 Adresse zu
- **CNAME Resource Record**: Definiert einen Alias auf einen vorhandenen Namen im DNS
- **MX Resource Record**: Bezieht sich nur auf SMTP (eMail) und sagt aus, welcher Mailserver für einen Namen zuständig ist
- **PTR Resource Record**: Ordnen Hostnamen einer IP zu (für Reverse Lookup wichtig)

Begriffe des Domain Name Systems II

- **NS Resource Record**: Name Server Records können zwei Funktionen haben
  1. Er definiert welches autorative Nameserver für diese Zone zuständig ist
  2. Er deligiert Zonen in einem Zonenbaum (Verkettung)
- **Reverse-Lookup**: Client fragt Server nach einem Namen zu einer gegebenen IP-Adresse

BIND als Nameserver I

Einige Informationen zum **B**erkeley **I**nternet **N**ame **D**omain/Daemon

- Weit verbreitet und gut Dokumentiert
- BIND ist sehr mächtig und groß ⇒ daher sei der Student zum experimentieren mit "more lightweight" DNS-Servern aufgerufen
- `named` als Hintergrunddienst
- Mindestens zwei Konfigurationsdateien `named.conf` und für jede Zone eine `db.*` Datei
  - Für die `named.conf` ist zu beachten:
  - `version none;` Flag
  - `allow-query { <localnet> ;}`
  - Einrichten einer Zone `cluster` o.ä. mit `type master;`
  - Für `db.*` Files ist zu beachten:
  - Zonefiles in die `named.conf` mit eintragen
  - In jeder Zonefile ist die lokale IP des Headnodes diejenige für DNS Server

BIND als Nameserver II

  - Bei PTR Einträgen oder Bennenung der Reverse Zone-File ist zu beachten das die IP Adressen "andersherum" geschrieben werden. Also z.B.: `2.0.168.192.in-addr.arpa.  IN PTR  node01.cluster.`
- Bei Problemen hilft das BIND9 Administrator Reference Manual (ARM) weiter.

2 Dienste
- DHCP
- DNS und BIND
- NTP

NTP I

Es ist darauf zu achten das alle Uhren im Cluster gleich laufen! Schon geringe Unterschiede können dazu führen das bspw. das Batchsystem seinen Dienst verweigert oder andere Fehler auftreten die teilweise sogar transient sind.

- NTP liegt aktuell in Version 4 vor und benutzt UDP Port 123
- `ntpd` als Hintergrundprozess der sowohl Zeit an Clienten weitergeben kann als auch die lokal Uhr "stellt"
- Hierarchiches System von NTP Servern und Zeitgebern über das Internet
- Genauigkeit hängt stark von Aufwand ab
  - Über Internet kann eine Genauigkeit von etwa 10ms gehalten werden
  - In lokalem Ethernet bis zu 150μs
  - Für Spezialanwendung können über hochpräzisen lokalen Taktgeber (Quarz, lokale Atomuhr, etc.) auch wenige μs im lokalen Netz erreicht werden.
- Unter Debian benötigte Pakete: ntp, ntpdate

NTP II

- `/etc/ntp.conf` als Konfigurationsdatei
- Als Timeserver kann für Headnode verwendet werden: `time.zih.tu-dresden.de`, für Compute Nodes entsprechend der Headnode

3 Ligthweigth Directory Access Protocol (LDAP)
- Historie
- Überblick
- LDAP-Datenmodell
- Installation & Konfiguration
- Nutzung von LDAP

Historie

- Ende der 70er erarbeitet die ITU (International Telecommunication Union) einen neuen E-Mail-Standard (X.400)
- Dafür ist ein globales Verzeichnis von Namen notwendig - ähnlich DNS
- ITU entwickelt dafür X.519-Standard: DAP (Directory Access Protocol)
- Nachteil DAP: benötigt gesamten OSI Protokoll-Stack
- Lösung: Entwicklung von **Light** DAP (benötigt nur TCP/IP Protokoll-Stack)

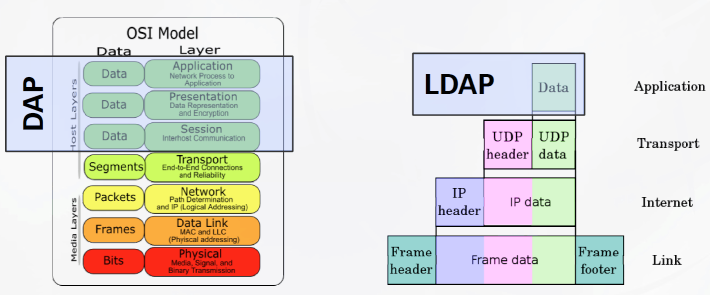


Abbildung: Gegenüberstellung von DAP und LDAP

Überblick

- LDAP definiert
  - wie die Daten repräsentiert werden,
  - auf welches Verzeichnis/welche Daten zugegriffen wird und
  - wie Daten importiert/exportiert werden können (LDAP Data Interchange Format, LDIF)
- LDAP definiert **nicht** wie die Daten gespeichert werden (meistens Verwendung eines RDBMS als Backend)
- LDAP charakterisiert sich als ,,Write-once-read-many-times"-Dienst:
  - Keine Transaktionsunterstützung
  - ,,Performance hit" liegt im Aktualisieren des Indexes
  - Lese-/Schreib-Verhältnis liegt i.d.R. bei 1000:1
- Asynchroner Replikationsprozess (führt zu Inkonsistenzen zwischen Master und Slave)
- Standardisiert (im Vergleich zu den SQL-Dialekten)
- Weiterleitungen (sog. ,,referals"), was eine Lastverteilung in großen Verzeichnissystemen erlaubt

LDAP-Datenmodell

- Hierarchie von Objekten
- Der sich daraus ergebende Baum wird als *Data Information Tree (DIT)* bezeichnet
- Die Spitze/Wurzel des Baums wird als *root*, *Basis*, *Suffix* oder *Partition* bezeichnet
- Data Information Tree:
  - Jedes Objekt hat genau einen Eltern- und einen oder mehrere Kindknoten
  - Ein Eintrag ist eine Komposition von einer oder mehreren Objektklassen
  - Objektklassen enthalten ein oder mehrere Attribute
  - Attribute: Tupel aus Namen (names) (teilweise mit Abkürzungen oder Aliasen) und Daten/Werten (values)

Attribute Top10

Abkürzung	Name	Beschreibung
dc	domainComponent	beliebiger Teil einer Domäne z.B. „tu-dresden“ von „tu-dresden.de“
o	organizationName	Name einer Organisation
ou	organisationalUnitName	Name einer Organisations-/Struktureinheit
cn	commonName	der vollständige Name, z.B. von einer Person
sn	surname	Familienname einer Person
gn	givenName	Vorname einer Person
uid	userid	beliebige Form eines Benutzernamens
userPassword	-	Benutzerkennwort für Zugriffskontrolle
mail	rfc822Mailbox	E-Mail-Adresse
telephoneNumber	-	Telefonnummer
...	...	...

Die Komposition von Attributen ergibt eine Objektklasse

Objektklassen Top10

Name	MUST	MAY
account	userid	description, seeAlso, localityName, organizationName, organizationalUnitName, host
country	c	searchGuide, description
dcObject	dc	-
groupOfNames	member, cn	businessCategory, seeAlso, owner, ou, o, description
groupOfUniqueNames	uniqueMember, cn	businessCategory, seeAlso, owner, ou, o, description
inetOrgPerson		audio, businessCategory, carLicense, departmentNumber, displayName, employeeNumber, employeeType, givenName, homePhone, homePostalAddress, initials, jpegPhoto, labeledURI, mail, manager, mobile, o, pager, photo, roomNumber, secretary, uid, userCertificate, x500uniqueIdentifier, preferredLanguage, userSMIMECertificate, userPKCS12
organizationalPerson		title, x121Address, registeredAddress, destinationIndicator, preferredDeliveryMethod, telexNumber, teletexTerminalIdentifier, **telephoneNumber**, internationaliSDNNumber, facsimileTelephoneNumber, street, postOfficeBox, postalCode, postalAddress, physicalDeliveryOfficeName, ou, st, l
organization	o	userPassword, searchGuide, seeAlso, businessCategory, x121Address, registeredAddress, destinationIndicator, preferredDeliveryMethod, telexNumber, teletexTerminalIdentifier, telephoneNumber, internationaliSDNNumber, facsimileTelephoneNumber, street, postOfficeBox, postalCode, postalAddress, physicalDeliveryOfficeName, st, l, description
organizationalUnit	ou	userPassword, searchGuide, seeAlso, businessCategory, x121Address, registeredAddress, destinationIndicator, preferredDeliveryMethod, telexNumber, teletexTerminalIdentifier, telephoneNumber, internationaliSDNNumber, facsimileTelephoneNumber, street, postOfficeBox, postalCode, postalAddress, physicalDeliveryOfficeName, st, l, description
person	sn, cn	**userPassword**, **telephoneNumber**, seeAlso, description
posixAccount	cn, uid, uidNumber, gidNumber, homeDirectory	userPassword, loginShell, gecos, description

Die Komposition von Objektklassen ergibt ein Schema

Objektklassen

- Sind Teil eines oder mehrerer Schemata
- Können hierarchisch organisiert werden, wodurch sie alle Attribute der Elternobjektklasse erben
- Arten:
  - Strukturell (Structural): Erzeugen einen Eintrag
  - Helfend (Auxiliary): Hinzufügen weiterer Attribute
  - Abstrakt (Abstract): Terminiert Objekthierarchien
- Die Art einer Objektklasse wird vererbt (außer bei abstrakten Klassen)

3 Ligthweigth Directory Access Protocol (LDAP)
- Historie
- Überblick
- LDAP-Datenmodell
- Installation & Konfiguration
- Nutzung von LDAP

LDAP-Server

- Server-Daemon: *slapd*

Hinweise
- Konfiguration erfolgt mit Hilfe von *dpkg-reconfigure*
- DNS-Domainname: dc=inf,dc=tu-dresden,dc=de
- Datenbank-Backend: Berkley Database
- Konfigurationsdatei *slapd.conf* wird unter */etc/ldap/* abgelegt

- *migrationtools*: Import der Nutzerdaten

Hinweise
- Anpassung von *migrate_common.ph*
- Ausführung von *migrate_all_online*
- Erscheint die Fehlermeldung *ldap_add: Already exists* kann der Import-Vorgang wie folgt fortgesetzt werden:
  ldapadd -x -c -D "cn=admin,dc=inf,dc=tu-dresden,dc=de" -f [temp file] -W

LDAP-Client I

- *libnss-ldap* erlaubt die Authentifizierung von Benutzern über LDAP

Hinweise
- Konfiguration erfolgt ebenfalls mit Hilfe von *dpkg-reconfigure*
- Überprüfung der */etc/ldap/ldap.conf* ob Server korrekt eingetragen ist
- Eintragen von LDAP in die */etc/nsswitch.conf*, welche für *Name Service Switch* steht, aber nicht nur auf den Namensdienst beschränkt ist
- Mit Hilfe der *ldap-utils* kann die Verbindung zum Server überprüft werden

- Pluggable Authentication Module (PAM) stellt eine API für Authentisierungsdienste bereit
- Ermöglicht es Authentisierungsmodule einzelnen Diensten zuzuordnen, ohne erneute Übersetzung der Anwendungen die diese Dienste realisieren

LDAP-Client II

- Damit PAM mit LDAP kommunizieren kann wird *libpam-ldap* benötigt
- Nach der Installation müssen die einzelnen Module unter */etc/pam.d/* angepasst werden

Hinweise
- Argument 1 (*type*):
  - *account*: Spezifiziert ob sich ein Benutzer einloggen darf
  - *auth*: Überprüft ob ein Benutzer auch derjenige ist für den er sich ausgibt
  - *password*: Ermöglicht es dem Benutzer sein Passwort zu ändern
  - *session*: Setzt die Umgebung nach der Authentifizierung

LDAP-Client III

Hinweise

- Argument 2 (*control*):
  - *requisite*: Ein Authentifizierungsfehler in diesem Modul führt zu einem direkten Abbruch der kompletten Authentifizierung
  - *required*: Ein Authentifizierungsfehler in diesem Modul führt zu einem Abbruch der Authentifizierung - Module mit dem gleichen Typ können allerdings noch abgearbeitet werden
  - *sufficient*: Wenn die Authentifizierung mit diesem Modul erfolgreich ist, ist es die Authentifizierung auch
  - *optional*: Dieses Modul ist nur signifikant, wenn es das einzige Modul für diesen Dienst ist
- Argument 3: Spezifiziert welches Modul verwendet werden soll und optional wo dieses zu finden ist

Einträge suchen, bearbeiten oder entfernen

Einträge suchen

- Anonyme Suche in LDAP: `ldapsearch -x uid=[UID]`
- Ausführliche Informationen:
  `ldapsearch -x uid=[UID] -D "[admin-account]" -W`

Einträge entfernen

```
ldapdelete -x -D "[admin-account]" -W "[account-to-delete]"
```

Einträge bearbeiten

```
ldapmodify -x -D "[admin-account]" -W
```

- Anschließend den Eintrag angeben der bearbeitet werden soll
- `changetype: modify`
- Eingabe des Attributs (z.B. "homeDirectory")
- Zum Abschluss dem Attribut einen neuen Wert geben

Neuen Nutzer hinzufügen

- Erstellung eines Templates:

  ```
  ldapsearch -LLL -x uid=[existing username] >>
  /tmp/template
  ```

- Anpassung der darin enthaltenen Informationen
- Hinzufügen des Accounts:

  ```
  ldapadd -x -D "[admin-account]" -W -f /tmp/template
  ```

- Eingabe eines Passworts für den neuen Nutzer:

  ```
  ldappasswd -x -D "[admin-account]" -W -S "[new-account]"
  ```

Literatur I

- Debian.
  Debian Reference (version 1), 2008.
  http://www.debian.org/doc/manuals/debian-reference/ch-gateway.de.html
- Debian.
  debianWiki, 2009.
  http://wiki.debian.org/de/DHCP_Server
- Internet System Consortium.
  BIND 9 Administrator Reference Manual, 2007.
  https://www.isc.org/software/bind/documentation/arm94
- Christopher Smith.
  Linux NFS-HOWTO, 2007.
  http://nfs.sourceforge.net/nfs-howto/

Literatur II

- University of Delaware.
  The Network Time Protocol (NTP) Distribution, 2009.
  http://www.eecis.udel.edu/~mills/ntp/html/index.html
- The OpenLDAP Project.
  OpenLDAP Software 2.4 Administrator's Guide, 2009.
  http://www.openldap.org/doc/admin24/