# Automatisierte Systemadministration

Chemnitzer Linux User Group

55. Stammtisch

Chemnitz, 4.5.2001

Matthias Clauß, URZ, TU Chemnitz

# Abstract

In den aktuellen Linux-Distributionen sind meist Software-Lösungen enthalten, die es einem Systemadministrator ermöglichen, eine überschaubare Anzahl von Rechnersystemen effektiv zu betreuen.

Bei einem massenhaften Einsatz von Linux-Rechnern mit unterschiedlicher HW-Ausstattung und unterschiedlichen Einsatz- Anforderungen stoßen die verfügbaren Admin-Software-Lösungen allerdings oft schnell an ihre Grenzen.

Das Universitätsrechenzentrum der Technischen Universität Chemnitz ist gegenwärtig für die Systembetreuung von über 1000 Linux-Rechnern verantwortlich.

Der Vortrag gibt einen Einblick in die Software-Tools und Technologien, die im URZ eingesetzt werden, um eine rationelle, skalierbare und konsistente Systemadministration zu gewährleisten.

# Entwicklung der Rechner-Anzahl

- Wieviele Linux-PCs betreut das URZ?
  - Frühjahr 1996: 1
  - Frühjahr 1997: ca. 15
  - Frühjahr 1998: ca. 100
  - Frühjahr 1999: ca. 200
  - Frühjahr 2000: ca. 340
  - Februar 2001: ca. 960
  - April 2001: ca. 1080
- heterogene Funktionalität der Rechner
  - Pools
  - Arbeitsplatzrechner
  - Server für verschiedenste Zwecke
  - Cluster-Rechner
  - Infoterminals

# Aufgaben der SysAdmin (1)

- Beschaffung
  - Planung
  - Mittelbeantragung (z.B. HBFG)
  - Spezifikation HW
  - Beschaffungsanforderung
  - Anbieterauswahl
  - Lieferungsüberprüfung
- Installation
  - Transporte, Aufstellung
  - Verkabelung, Netzintegration
  - HW-Testung
  - Einsatzplanung
  - Installationsvorbereitung
  - Installation, Nachinstallation (Konfiguration)
  - Inbetriebnahmetests

# Aufgaben der SysAdmin (2)

- Laufende Betreuung
  - Nutzerinformation u. -schulung
  - Überwachung des HW/SW-Zustandes
  - Fehlersuche und -korrektur
  - SW-Pflege inkl. BS-Upgrade
  - (Re-)Konfiguration
  - HW-Pflege, HW-Reparatur
- Rationalisierung der SysAdmin
  - Entwurf von Automatisierungsverfahren
  - Implementierung
  - Entwicklung von Tools und Script-Sets
  - Weiterbildung
- Organisation
  - Anleitung, Kontrolle
  - Koordination der Aktivitäten aller Beteiligten

# Kooperative SysAdmin

- horizontale und vertikale überlagernde Verantwortungsbereiche

- vertikale Verantwortungsbereiche
  - Network-Admin
  - Benutzerverwaltung
  - Authentifizierung und Security
  - Printer-Management
  - Web-Server-Management
  - Mailbox-Management
  - Hardware-Wartung
  - Benutzeroberflächen
  - ...

- horizontale Verantwortungsbereiche
  - Linux
  - Solaris
  - WNT
  - ...

# SW-Basis der SysAdmin

- verteiltes Filesystem: z.B. AFS, NFS
- Konfiguration: cfengine
- HW/SW-Monitoring: z.B. BigBrother (bb)
- Software-Pflege: z.B. rpm, autorpm
- Scriptsprachen: z.B. bash, awk, python, perl

- Anforderung an Konfigurations-Tools
  - Kommandozeilen-Schnittstelle (nicht nur GUI-Schnittstelle)
  - keine erzwungenen Dialoge
  - neutral hinsichtlich Policies und Technologie
  - Hinzufügen neuer Features, Optionen, Module
  - Unterstützung verschiedener Plattformen, möglichst unabhängig von Version und Release
  - Konfiguration muss datengesteuert sein, nicht im Tool verdrahtet
  - Text-Konfigurationsfiles (keine DB, registry)
  - Texteditierung der Files muss weiterhin möglich sein
  - zeilenorientierte UNIX-Tools (grep, cut, ...) müssen anwendbar sein
  - existierende Systemfiles dürfen nicht ersetzt werden

# Andrew File System (AFS)

- verteiltes Filessystem
- (weltweit) gemeinsamer/einheitlicher Namensraum
- Client-Server-Architektur
- Organisationsstruktur: AFS-Zellen
- KERBEROS-basierte Authentifizierungs
- Speichereinheit: AFS-Volumes
- Client-Rechner: AFS Cache Manager
  - Transparenter Serverzugriff
  - Zugriff auf File-Kopie im (lokalen) Cache
  - Zugriffsteuerung über Access Control Lists (ACL)

- Chemnitzer AFS-Zelle (/afs/.tu-chemnitz.de/)
- Übergang auf OpenAFS
- AFS ist Grundlage für Konfiguration von Linux-Rechnern

# Überlegungen zu Konfigurationsfiles

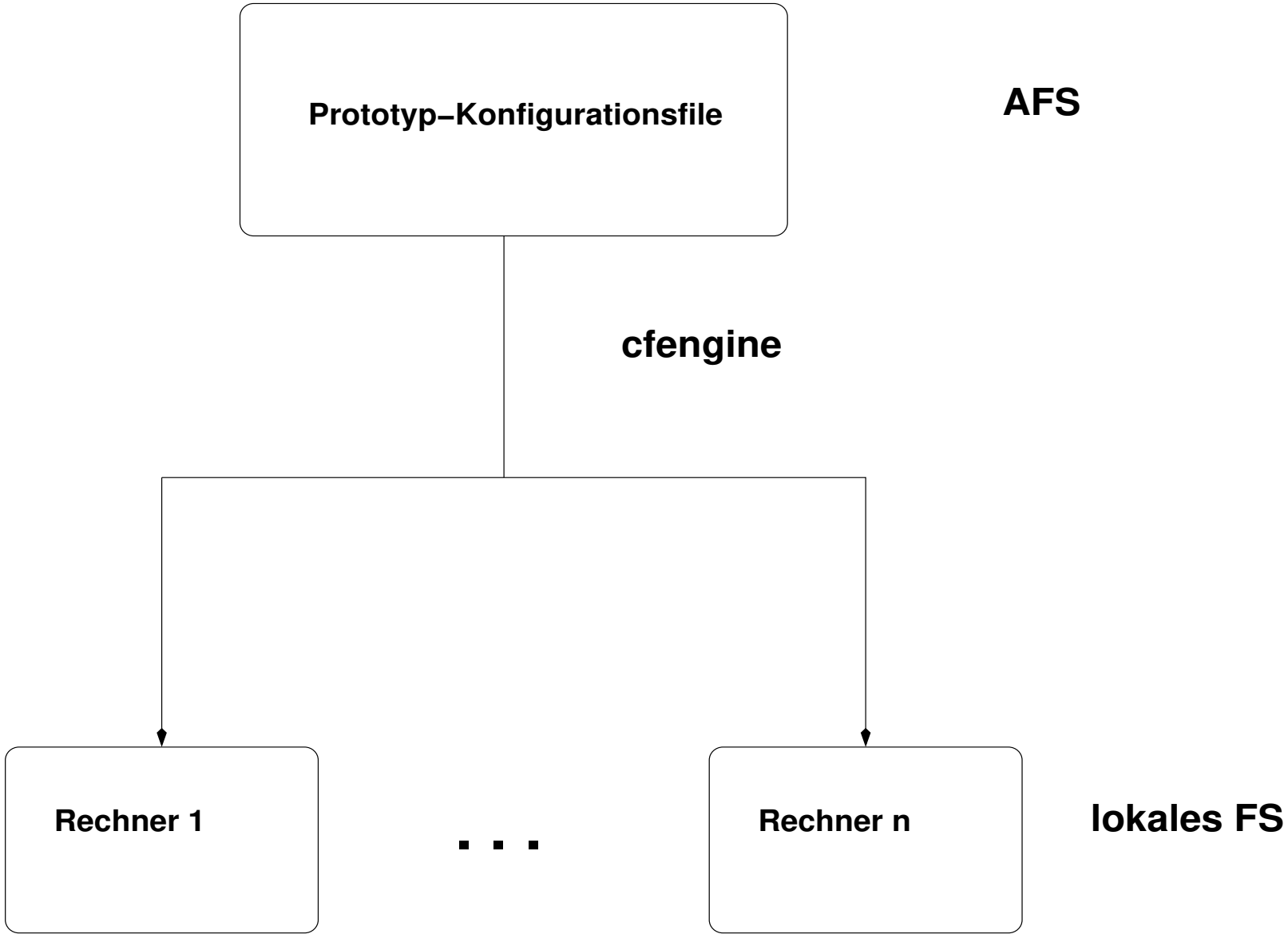
- Konfigurationsfiles sind Systemfiles
  - die individuelle Eigenschaften eines Rechners beschreiben
  - die einen installierten Rechner an lokale Bedingungen anpassen oder funktionell erweitern
  - deren Konsistenz für die korrekte Funktionsweise des Rechners notwendig ist

- möglichst nur Files administrieren, die
  - "lebensnotwendig" sind bzw.
  - nach der Installation modifiziert werden (müssen)

- Beispiele:
  - /etc/inetd.conf
  - /etc/hosts.allow
  - /etc/printcap
  - /usr/local/bb/etc/bb-dftab

- z.Z. insgesamt ca. 50000 Protytop-Konfigurationsfiles

# Konfigurationsansatz (0)



# Konfigurationsansatz (1)

- Prototypen (Master) der Konfigurationsfiles werden im AFS verwaltet
  - für alle Rechner besteht zu jedem Zeitpunkt Lesezugriff
  - Prototyp kann an beliebigen Rechnern modifiziert werden
- Änderungen der Konfiguration
  - in der Regel indirekt, d.h. durch Modifikation der entsprechenden Prototyp-Konfigurationsfiles
  - Änderungen können an einem beliebigen Rechner vorgenommen werden
  - Vergleich: Prototyp  mit entsprechendem Konfigurationsfile eines Rechners
  - Kopieren des Prototyps, falls unterschiedlich
- Rechner mit gleicher Funktionalität bzw. Eigenschaften zusammenfassen:
  - 1 Prototyp für (beliebig) viele Rechner
  - Korrektur des Prototyps --> spätere Änderung auf den Rechnern

# Konfigurationsansatz (2)

- Hierarchie von Rechner-Klassen entsprechend der Funktionalität (Funktional-Klasse, FU)
- FUs sind disjunkt: jeder Rechner gehört zu genau einer FU
- Beispiele:
  - alle Rechner des Poolraums 301 in der ReiStra bilden die Klasse FU_PC_Pool_RS_301
  - zur Klasse FU_SambaServer gehören alle Samba-Server

- Rechner-Klassen entsprechend spezifischer Eigenschaften (Attribut-Klasse, AT)
- ATs überschneiden sich, ein Rechner kann zu mehreren ATs gehören
- Beispiele:
  - alle Rechner, auf denen StarOffice 5.2 laufen soll: AT_StarOffice52
  - Rechner mit KDE: AT_KDE_1 oder AT_KDE_2

# Konfigurationsansatz (3)

- weitere Klassentypen
  - alle Rechner eines Subnetzes bilden eine Subnetz-Klasse
    - Beispiel: Rechner im Subnetz 134.109.72 gehören zur Klasse SN_72
  - jedem Rechner ist Distributions-Klasse zugeordnet
    - Beispiel: Rechner mit installiertem RedHat 6.2 gehören zur Klasse DIS_RH62
  - jeder Rechner gehört zu genau einer Hostklasse <hostname>
    - Beispiel: Rechner janus gehört zur Host-Klasse janus
  - generische Klasse any für Eigenschaften, die allen Rechnern gemeinsam sind

# Konfigurationsansatz (4)

- für jede Klasse wird eine Filesystemstruktur (Repository) aufgespannt
- Repository enthält alle Prototyp-Konfigurationsfiles, die diese Klasse von den Prototypen aller anderen Klassen unterscheiden
- für jeden Prototyp gilt es, die allgemeingültigste Klasse zu finden, in die er eingeordnet werden kann
- identische Prototypen sind möglichst zu vermeiden
- Beispiel Klasse any für alle Rechner:
  - /etc/shells
- Beispiel für die Funktional-Klasse FU_SambaServer:
  - /usr/local/bin/smbclients
- Beispiel für die Funktional-Klasse FU_LoginServer:
  - /etc/cron.d/wwwlog
- Beispiel für die Host-Klasse janus:
  - /etc/ssh_host_key (verschlüsselt!)

# Beispiel: Repository für FU_SambaServer

/afs/tu-chemnitz.de/.../NROOTS/FU_SambaServer/etc:
group.local
hosts.allow
hosts.equiv
login.access
passwd.local
printcap
smb.conf
smb.conf.Samba
smb.conf.UNKNOWN
smb.conf.WfWg
smb.conf.Win2K
smb.conf.Win95
smb.conf.WinNT
smb.conf.shares

/afs/tu-chemnitz.de/.../NROOTS/FU_SambaServer/etc/autorpm.d/autorpm.conf
/afs/tu-chemnitz.de/.../NROOTS/FU_SambaServer/etc/logrotate.d/samba
/afs/tu-chemnitz.de/.../NROOTS/FU_SambaServer/etc/pam.d/samba
/afs/tu-chemnitz.de/.../NROOTS/FU_SambaServer/etc/sysconfig/afs
/afs/tu-chemnitz.de/.../NROOTS/FU_SambaServer/usr/local/bin:
smbclients

# Überlegungen zu Log-Files

- Log-File-Verzeichnis für jeden Rechner als "Ablage" für Informationen
  - allgemeine Informationen, z.B. Funktion und Standort des Rechners
  - rotierte Log-Files, z.B.
    - /var/log/messages
    - /var/log/dmesg
  - Logfiles von cron-Jobs (Systemwartung, ...)
  - HW-Infos, System-Infos, Infos über SW-Bestand

- außerdem: Kopien von Files, deren Reproduzierbarkeit kritisch ist, z.B.
  - /etc/lilo.conf
  - /etc/fstab
  - /etc/modules.conf

- Grundsatz: Logs werden von SW-Tools erzeugt bzw. aktualisiert (nicht manuell!)
- Konsequenz: jede HW/SW-Änderung (RAM-Erweiterung, Plattentausch, SW-Upgrade etc.) wird automatisch protokolliert

# Configuration Engine (1)

- cfengine
  - Programmiersprache für die automatisierte Verwaltung und Konfiguration von vernetzten Rechnern
  - Autor: Mark Burgess (Oslo College, Norwegen)
  - unterliegt der GNU-Lizenz
  - Very High-Level Language
  - Klassen-basierte Entscheidungsstruktur

- Arbeitsweise von cfengine
  - cfengine-Programm beschreibt die Konfiguration der Rechner im Verantwortungsbereich
  - cfengine-Interpreter arbeitet das Programm auf jedem betreffenden Rechner ab und vergleicht die Konfiguration des Systems mit dem cfengine-Programm
  - dabei festgestellte Unterschiede werden im System korrigiert

# Configuration Engine (2)

- cfengine-Operatoren (Auswahl):
  - Files kopieren
  - symbolische Links prüfen und einrichten
  - Zugriffsrechte und Eigentümerangaben von Files prüfen und setzen
  - Files löschen
  - Shell-Scripte und Kommandos ausführen
  - Signale an Prozesse schicken
  - ACSII-Files editieren
  - Netz-Schnittstellen prüfen und konfigurieren
- Wann läuft cfengine auf einem Rechner?
  - per cron
    - zu bestimmten Zeitpunkten
    - einmal täglich (nachts)
  - beim Booten
  - gezielt bei Bedarf

# Configuration Engine (3): Beispiel:

```
classes:
 FU_CpuServer = ( janus pandora odoaker )
 SN_132  = ( FU_CpuServer )
 AT_Sensors = ( janus )
 AT_StarOffice52 = ( FU_CpuServer )
control:
 any::
  BASEDIR = ( /afs/tucz/admin/linux/REDHAT/CFENGINE/NROOTS )
  LOGDIR  = ( /afs/tucz/admin/linux/REDHAT/CFENGINE/LOGS )
  RPOW62  = ( /afs/tucz/ftp/pub/linux/redhat-ftp/redhat/powertools/6.2/i386 )
  actionsequence = ( copy files shellcommands )

copy:
  any::
    $(BASEDIR)/ROOT4ALLSYSTEMS   dest=/ o=root g=root rec=inf type=checksum
    $(BASEDIR)/FU_CpuServer      dest=/ o=root g=root rec=inf type=checksum
    $(BASEDIR)/$(host)           dest=/ rec=inf o=root g=root size=>0 type=checksum inform=off

files:
  any::
    /etc/passwd     m=444 o=root g=root act=fixall inform=off
```

# Überwachung des HW/SW-Zustandes (1)

- BigBrother (bb)
  - System und Netzwerkmonitor
  - Shell-Scripts, C
  - erweiterbar
- Web-Seiten
  - Zustandsanzeige
  - Benachrichtigung (Mail, Pager, SMS)
  - History-Logs
  - Statistikreports
  - Integration von Infos und Logs
- Farben kennzeichnen Zustand
  - grün --> OK
  - gelb --> ACHTUNG
  - rot --> (ernstes) PROBLEM
  - lila --> KEIN REPORT
  - farblos --> NICHT VERFÜGBAR
  - blau --> OFFLINE

# Überwachung des HW/SW-Zustandes (2)

- Welche Zustände werden überwacht?
  - Netzkonnektivität
  - CPU-Last
  - Filesystem-Füllstand
  - kritische System-Nachrichten
  - Prozesse
  - Netzdienste, z.B.
    - dns
    - ftp
    - http(s)
    - imap
    - ssh
    - pop
- realisierte Erweiterungen
  - HW-Sensoring
  - RAID-ARRAY-Überwachung
  - fping

# Softwarepflege mit rpm

- RedHat Package Manager (rpm)
  - Beispiel: gvim fehlte auf einigen RH 6.2-Systemen (cfengine-Ausschnitt)

```
...
shellcommands:
...
# wmc 26.04.2001
DIS_RH62::
   "/bin/rpm -Uhv ${RPMS62U}/vim-X11*.rpm 2>>${LOGDIR}/$(host)/rpm_error.log"
```

- Automatisches Paket-Upgrade
  - rpm -F .../*.rpm
  - autorpm

# Softwarepflege mit autorpm

- /etc/autorpm.d/autorpm.conf (Beispielauschnitt)

```
Eval_Var("RHVersion", "sed 's/\(Red Hat Linux \)\?release \([^ ]*\) (.*)/\2/' /etc/redhat-release");
Report_Queues_To ("PRINT");

Directory ("/afs/tu-chemnitz.de/ftp/pub/linux/redhat-ftp/redhat/updates/${RHVersion}/i386") {
  Report_To ("");
  # Don't upgrade kernel packages, netscape, kde
  Regex_Ignore ("^kernel-");
  Regex_Ignore ("^netscape-");
  Regex_Ignore ("^kde");

  action (updated) {
    Install (Auto);
    Report (Yes);
  }
}
```

# System-Installation

- Etappen
  - Basis-Installation
  - Nachinstallation
  - Konfiguration

- Vorgehensweise ist abhängig von Voraussetzungen und Randbedingungen
  - CD-ROM-basiert (Text-Modus)
    - bei Einzelinstallationen (Server, Desktop-Systeme)
    - Pools

  - RedHat-Kickstart
    - Cluster (Pools) mit >> 10 homogenen Rechnersystemen
    - in Vorbereitung: automatische Installation beliebiger Rechner

  - speziell vorbereitetes Verfahren beim Chemnitzer Linux Cluster
    - Boot-Diskette (etherboot) (oder Ethernet-Karte mit BOOT-ROM)
    - per DHCP Bootimage von TFTP-Server holen
    - Kickstart
    - Boot-Medien: Disk vor Diskette

# Kickstart (1)

- RedHat-Kickstart
  - programmierter Installations-Dialog
  - Konfigurationsfile (ks.cfg) auf Diskette, CD-ROM oder NFS

- Beispiel: Update per Kickstart

```
lang de
network --bootproto bootp
cdrom
keyboard de-latin1-nodeadkeys
upgrade
lilo --location mbr
%post
echo "Kickstart fertig ..."
```

# Kickstart (2)

- Beispiel: Installation über NFS mit Neupartitionierung

```
lang de
network --bootproto static --ip 134.109.200.38 --netmask 255.255.255.0 --gateway
134.109.200.254 --nameserver 134.109.200.4
nfs --server 134.109.132.32 --dir /afs/tu-chemnitz.de/global/i386_linux3/
install/redhat-5.2/i386
keyboard de-latin1-nodeadkeys
zerombr yes
clearpart --linux
part / --size 512
part swap --size 128
part /usr --size 1000 --grow --maxsize 4000
part /tmp --size 250
install
mouse ps/2
timezone MET
xconfig --server "SVGA" --monitor "IIYAMA MF-8515G"
%packages
@ Networked Workstation
@ C Development
```

# Referenzen

AFS: http://www.tu-chemnitz.de/urz/system/afs/

Mark Burgess: cfengine, a configuration engine; http://www.iu.hioslo.no/cfengine/

T. Müller, M. Clauß: Administration von Unix-Rechnern mit cfengine;
http://www.tu-chemnitz.de/urz/kurse/unterlagen/cfengine/

BigBrother System and Network Monitor; http://bb4.com/

Kirk Bauer: Autorpm; http://www.kaybee.org/~kirk/html/linux.html

Mike Becher: Remote Installation von RedHat 6.2;
http://www.tu-chemnitz.de/urz/anwendungen/cluster/publish/RemoteInstall/