Automatische Installation

Dipl.-Chem. Rainer Orth
Technische Fakultät
Universität Bielefeld
ro@TechFak.Uni-Bielefeld.DE

Übersicht

- Ziel und Alternativen
- Aufgaben, Varianten
- Netzwerk-Boot
- Partitionierung und Filesysteme
- Software-Installation
- Upgrades
- Lokale Konfiguration

Automatische Installation: warum?

- reproduzierbare, nicht interaktive, parallele Installation von Standalone-Maschinen
- ggfs. viele Maschinen (identisch oder verschieden)
- hier: vor allem Solaris 2 (JumpStart)
- ähnliche Funktionalität auf IRIX (Roboinst), Tru64 UNIX (RIS), Mac OS X (NetInstall), Debian Linux: FAI (Fully Automatic Installation, `http://www.informatik.uni-koeln.de/fai/`)
- Alternative: diskless clients, vereinfacht Installation und Patchen vieler Einzelmaschinen (shared /usr), aber Support teilweise problematisch

Aufgaben

- Booten (in der Regel über Netz)
- Platten partitionieren, Volume-Manager konfigurieren, Filesysteme anlegen
- Software-Selektion, -Installation
- lokale Anpassungen/Konfiguration

Varianten

- Neuinstallation oder Upgrade
- offline oder online (Live Upgrade)
- Installation von Packages oder Image-Cloning (flash archives)

Netzwerk-Boot im LAN

- vorher: Boot- und Install-Server
- Boot-Server: `inetboot` per tftp, diskless Root (read-only) via NFS
- Install-Server: Kopie der Installations-CDs via NFS
- Booten:
  - SPARC: OBP mit RARP, RPC Bootparams (klassisch, erfordert Boot-Server in jedem Subnetz)
  - . . . oder OBP mit DHCP (neuere PROMs)
  - x86: PXE (Pre-boot Execution Environment), DHCP und tftp in Netzwerkkarten-Firmware
  - diverse Installations-Parameter per Bootparams oder DHCP

Netzwerk-Boot im WAN: WANboot

- DHCP und NFS im WAN schlecht geeignet, daher `http`/`https` benutzen
- `wanboot` laden (ähnlich `inetboot` oder `ufsboot`, 1st-level bootstrap)
- miniroot-Image laden (statt NFS-Zugang auf diskless Root)
- flash-Archiv (`cpio`) laden und installieren
- alle Daten ggfs. signiert/verschlüsselt
- erfordert neue PROMs oder lokale Installations-CD

Konfiguration der Installation

- Konfiguration von Parametern, die bei manueller Installation interaktiv erfragt werden: `sysidcfg`: IP-Adressen, Hostname, Nameservice, Root-Password, Security Services (Kerberos), Locale, Terminal-Typ, Zeitzone, ...
- Konfiguration mit `rules`-File und `profile`
- `rules`: selektiert `profile` und pre-/post-Kommandos (begin/finish-Skripten) abhängig von z.B. Hostnamen, Maschinentyp, Platten-/Speicherausbau, bei Bedarf selbstdefinitierte Probes
- z.B. dynamische `profile`-Erzeugung in begin-Skript
- z.B. lokale Anpassungen im finish-Skript

Partitionierung

- Plattenauswahl
- Filesysteme, Swap
- Metadevices/Mirror anlegen

Software-Installation

- `profile`: Package-Auswahl etc.
  - `install_type` (`initial_install`, `upgrade`)
  - `system_type` (`standalone`, `...`)
  - `partitioning`, `filesys`
  - `cluster`, `package`, `patch`
- Interaktion mit Package-Management
- Auswahl über Metacluster, Cluster, einzelne Packages
- reduced network, core, end user, developer, entire, entire + OEM

Probleme mit Package-Installation

- kann recht langsam sein (dauerndes Update der Package-Datenbank)
- Packages, die davon ausgehen, in die laufende OS-Instanz installiert zu werden (ignorieren `PKG_INSTALL_ROOT`, `-R`)
- Packages, die neu installierte Kommandos ausführen (kann fehlschlagen, wenn z.B. shared Libraries fehlen)

Flash-Installation

- cpio-Archiv mit Komplettinstallation
- wesentlich schneller als package-weise Installation
- aber: potentiell viele Archive für viele Klassen von Maschinen
- full (Neuinstallation) oder differential (Upgrade/Ergänzung eines full archives: Änderungen zwischen Master-Archiv und aktuellem Stand)
- Vorgehen:
  - Master-System installieren
  - ggfs. anpassen
  - Archiv erzeugen
  - Archiv installieren
  - ggfs. Skripte zur Anpassung

Upgrade

- Vor- und Nachteile von Neuinstallation vs. Upgrade?
- Neuinstallation:
  - definierter Endzustand
  - Reproduzierbarkeit sichergestellt
- Upgrade:
  - sollte alle lokalen Anpassungen (allg.: Zustand) erhalten
  - bequemer bei häufigen „kleinen“ Upgrades (Solaris Express, u.U. Solaris Updates)

Live Upgrade

- legt Kopie der aktuellen OS-Installation an (Wahl der Zielpartitionen nach Belieben, auch Zusammenlegen von FSen etc.): `lucreate`
- Upgrade der Kopie, während die alte Version läuft, ggfs. auch Neuinstallation via flash-Archiv (`luupgrade`)
- Aktivieren der geupgradeten Version (`luactivate`)
- System-Filesysteme individuell pro BE (Boot Environment), User-Filesysteme (Volumes, Homes, . . .) werden geshared, üblicherweise auch Swap
- auch Mirror (SVM) als neues BE

Live Upgrade (ii)

- Vorteile:
  - minimale Downtime (ein Reboot)
  - schnelles Fallback auf alte Version bei Problemen
- ggfs. Modifikationen an neuem BE vor Aktivierung (mounten mit `lumount`)
  - Patch-/Package-Installation/-Entfernung
  - Konfigurationsänderungen
  - ...

Konfigurationsänderungen

- üblicherweise in finish-Skript
- potientiell komplett lokal (Kopieren von Files, Auspacken von `tar`-Files, ...)
- Problem: Integration mit Konfigurationsmanagement

JumpStart-Frameworks

- lokale Anpassungen einfacher mit vordef. Frameworks
- z.B. Caspar Dik's `auto-install`: `ftp://ftp.science.uva.nl/pub/solaris/auto-install/`
  - Standard-File-Operationen: copy, remove, link, symlink, ...
  - Ausführung beliebiger Skripten
  - Zuordnung der Systeme in verschiedene benutzerdefinierte Klassen
- Solaris Security Toolkit (JASS) (`http://www.sun.com/software/security/jass/`): Installation und Security-Hardening von Solaris-Systemen
- JumpStart Enterprise Toolkit (JET): `http://www.sun.com/bigadmin/content/jet/`