Konfigurationsmanagement

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

Übersicht

- Das Problem
- Der Lösungsraum
- Klassische Lösungen
- Filebasierte Konfiguration
- Frameworks

Was ist Konfigurationsmanagement?

- lokale Anpassungen bei/nach initialer Installation
- sehr vielfältig: Aktivierung/Deaktivierung/Konfiguration von Services, Netzwerk-Parameter, ...
- werden nicht von OS-Installation geleistet oder Konfiguration OS-abhängig
- manuelle Konfiguration aufwendig, fehlerträchtig
- daher: Automatisierung
- sowohl für initiale Installation als auch für spätere Anpassungen
- z.B. auch Rekonfiguration je nach Anforderungen (Last)
- seit mehr als 15 Jahren Thema von LISA-Vorträgen, Workshops

Mögliche Ansätze

- Spezifikation einer Konfiguration
  - Verhalten (SMTP-Server auf Host X) oder Implementierung (`sendmail.cf`, `aliases`, ...)
  - Host-Level oder Netzwerk-Level (Dienste erfordern i.d.R. kooperierende Konfigurationen auf verschiedenen Systemen: Mailserver + `MX`-Records + Firewall)
  - prozedural oder deklarativ (Realisierungsvorschrift vs. Absicht/Ziel)
  - komplett oder partiell (bzw. proskriptiv vs. inkrementell)

Klassische Lösungen

- Verteilung von Konfigurationsfiles etc. mit `rdist`, `rsync`, ...
- Einteilung von Hosts in Klassen, Attribute zur Auswahl
- Probleme:
  - ggfs. Service-Restart/-Reload nötig
  - Pflege von geshareten Konfigurationsfiles mit generischen und service-spezifischen Teilen (z.b. `hosts.allow`)
  - Konfigurationen nicht file-basiert (z.B. SMF)
  - Push vs. Pull: Erfassen/Überschreiben von lokalen Änderungen, Aktualisierung abgeschalteter Hosts

Filebasierte Konfiguration (i)

- Craig et al., UMich: `radmind`
- `http://www.radmind.org/`
- `rdist` meets `tripwire`
- Verteilung und Validierung installierter Files, zusammengesetzt aus mehreren Sets
- ermöglicht Verteilung von geänderten Files und Erfassen von Änderungen

Filebasierte Konfiguration (ii)

- Lefebvre, Snyder, CNN: `newfig`
- LISA 2004
- Konfig-Files werden durch Skripte erzeugt und ggfs. validiert
- kann so einfach Semantik und Syntax trennen (z.B. `inetd` vs. `xinetd`)
- derzeit nicht öffentlich verfügbar

Frameworks (i)

- Anderson et al., Edinburgh: `LCFG`
- `http://www.lcfg.org/`
- Konfigurationsparameter in Beschreibungsfiles auf zentralem Server für Aspekte der Systeme (z.B. Webserver, ...), sowohl generisch als auch maschinenspezifisch
- werden in XML-Profile für Einzelmaschinen kompiliert
- Maschinen übertragen sie per `http`
- Komponentenskripten werten sie aus und setzen sie in Konfigurationsänderungen um
- umfaßt auch Synchronisation von Software-Packages
- Plattformen: Linux (RedHat), teilweise Solaris, Mac OS X

Frameworks (ii)

- Roth, UIUC: `psgconf`
- `http://www.feep.net/psgconf/`
- modulares Framework für System-Konfiguration
- Perl-Objekte kapseln
  - Konfigurationsdaten (statt expliziter Verwendung bestimmter Formate oder Backends)
  - Policies (z.B. Anon-FTP als Aktiverung von `wu-ftpd`, `ftp`-User, `inetd.conf`, TCP-Wrappers)
  - Aktionen (Manipulation des Systems: Editieren von Konfig-Files, Daemon-Restart, . . .)

Literatur

- http://config.sage.org/
- http://homepages.informatics.ed.ac.uk/group/lssconf/
- http://www.infrastructures.org/
- http://ark.sourceforge.net/