Cluster und Rechnerpools per DSH administrieren

Muschelsammlung

Wer viele ähnlich aufgebaute Rechner administriert, muss oft auf allen Maschinen die gleichen Kommandos absetzen. Statt sich überall einzeln einzuloggen, setzt der erfahrene Admin eine Distributed Shell ein: Sie kümmert sich um die SSH-Verbindungen, führt die Befehle aus und sammelt deren Ausgaben. Michael Renner

Oft hat der Administrator eines Clusters oder eines verteilten Systems gleich lautende Befehle auf allen Computern auszuführen. Statt von Rechner zu Rechner zu laufen und die Kommandos einzutippen, greift er zu einem Tool wie DSH [1] (Distributed Shell, auch Dancers Shell genannt). Es startet einen einmal eingegebenen Befehl auf allen Rechnern. DSH glänzt durch hohe Geschwindigkeit, kann die Maschinen in Gruppen verwalten und die Kommandos nacheinander oder parallel absetzen.

Die DSH erledigt ihre Arbeit nicht allein. Sie benötigt für die Verbindung zum entfernten Rechner eine Remote Shell wie »rsh« oder besser »ssh«. Nach erfolgreichem Login übergibt DSH die vom Admin gewünschten Befehle. Sie sammelt auch die Textausgaben der einzelnen Befehlsläufe und zeigt sie an der Administrationsmaschine an.

Viele Konkurrenten

Neben der Distributed Shell gibt es für diese Aufgabe auch einfache Bash-Skripte, Perl-basierte Lösungen, sogar ein Python-Skript ist im Umlauf [2]. Umfangreiche kommerzielle Softwarepakete zur Verwaltung von Clustern wie IBMs PSSP (Parallel System Support Program) oder SC-Venus von Science + Computing [3] enthalten ebenfalls Werkzeuge, die Befehle auf vielen Rechnern gleichzeitig ausführen.

Während die kommerziellen Lösungen außer auf der Administrationsmaschine meist auch auf allen administrierten Rechnern installiert sein müssen, begnügen sich die freien Anwendungen mit einem SSH-Daemon auf den Zielrechnern. Nur auf dem Computer des Administrators müssen DSH & Co. vorhanden sein. Der angenehme Nebeneffekt: Von einem Linux-Rechner aus lassen sich auch andere Betriebssysteme verwalten, sie brauchen nur einen Kommando-Interpreter und einen SSH-Daemon.

DSH konfigurieren

Eine Übersicht aller DSH-Kommandozeilenschalter ist in Tabelle 1 zu sehen. Es ist zwar möglich, die Zielrechner nach dem Parameter »--machine« aufzulisten, aber zu empfehlen ist es nicht. Wer viele Maschinen verwaltet, sollte die Rechner zu Klassen zusammenfassen. Sauber voneinander getrennt lassen sich zum Beispiel alle Linux-Clients in der Datei »linux« aufführen und alle Irix-Workstations in die Datei »irix« packen.

Normalerweise befinden sich diese Gruppendateien im Verzeichnis »/etc/dsh/group/«. Um dort zu schreiben, muss der Administrator als Root eingeloggt sein – bei der täglichen Arbeit ist das aber recht unsicher. Besser loggt er sich als unprivilegierter Benutzer ein und legt die Gruppendateien in »~/.dsh/group/« ab.

Um dennoch auf den Maschinen mit Root-Rechten zu agieren, stellt der Admin jedem Befehl ein »sudo« voran – vorausgesetzt, dass Sudo auf den administrierten Maschinen installiert ist. Eleganter ist es, den SSH-Login als Benutzer Root zu erzwingen. Dazu muss vor jedem Rechnernamen in der Gruppendatei »root@« stehen.

Interessant und praktisch ist die Option »--all«, mit der DSH alle Rechner auf einmal anspricht. Welche Maschinen sie beachten soll, liest die Software aus der speziellen Gruppendatei »all«. Dieses File muss der Admin selbst im Verzeichnis »~/.dsh/group/« anlegen und mit dem Inhalt der einzelnen Gruppendateien füllen. Soll ein Befehl nur auf einer Rechnergruppe laufen, ist die Option

»--group *Gruppenname*« anzugeben. Damit die Login-Shell Systembefehle auch in »/sbin« und »/usr/local/sbin« ohne explizite Pfadangabe findet, muss die »PATH«-Variable auf jedem Client entsprechend gesetzt sein. Das geschieht – je nach Shell – in »~/.bashrc« oder »~/.cshrc«.

Die DSH ist jetzt einsatzbereit. Allerdings muss der Admin das Login-Passwort für jeden Zielrechner von Hand eingegeben. Hier kommt Hilfe von der Secure Shell: Sie kann die Authentifizierung vom Passwort-basierten System auf kryptographische Schlüssel umstellen.

SSH-Schlüssel erzeugen und verteilen

Um einen Login ohne Passwort zu erlauben, benötigt die SSH ein benutzerspezifisches Schlüsselpaar [5]. Der öffentliche Teil muss auf den Zielrechnern liegen, der private auf der Arbeitsmaschine des Admin. Dort genügt sein nicht privilegiertes Alter Ego, beispielsweise der Benutzer Admin. Unter dieser Kennung erzeugt er per »ssh-keygen« ein Schlüsselpaar, siehe Abbildung 1.

SSH-Keygen fragt vor dem Speichern nach einer Passphrase – nur wer seinem Arbeitsrechner vertraut, darf darauf verzichten und die Frage mit der [Enter]-Taste beantworten. Daraufhin entsteht ein Schlüsselpaar in der Datei »~/.ssh/id_dsa« und in »~/.ssh/id_dsa.pub«. Den öffentlichen Teil des Key (hier »~/.ssh/id_dsa.pub«) muss der Admin auf jedem Zielrechner an die Datei »~/.ssh/authorized_keys« anhängen. Das lässt sich auch per SSH erledigen:

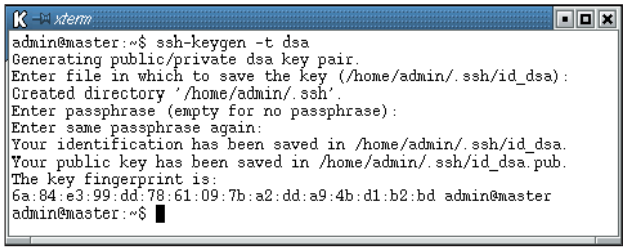

Abbildung 1: Der Benutzer mit Namen »admin« erzeugt über das SSH-Kommando »ssh-keygen« ein DSA-Schlüsselpaar. Mit dem geheimen Key authentifiziert er sich später an den Zielrechnern im Cluster.

```
$ cat ~/.ssh/id_dsa.pub|ssh root@vollmilch ➥
  'cat>>.ssh/authorized_keys'
```

Für einen ersten Test soll der Administrator die Uptime aller in der Datei »all« eingetragenen Rechner sammeln. Dazu reicht ein einziger Aufruf von DSH. Mit der Option »--show-machine-names« fügt DSH den Namen jedes Zielrechners in die Ausgabe ein. Nur so lässt sich die Ausgabe der Uptime dem jeweiligen Rechner zuordnen. Wie die einzelnen Komponenten ineinander greifen, verdeutlicht Abbildung 2.

Tabelle 1: DSH-Optionen

Option	Parameter	Bedeutung
-v --verbose	-	Die Meldungen sollen ausführlicher sein
-q --quiet	-	DSH soll weniger Meldungen ausgeben
-M --show-machine-names	-	Die einzelnen Hostnamen in der Textausgabe mit anzeigen
-i --duplicate-input	beliebiger Ascii-Text	Verteilt die Texteingaben des DSH-Benutzers an alle parallelen Sitzungen
-b --bufsize	Byte	Puffergröße für verteilte Texteingaben
-m --machine	Hostnamen	Zielrechner einzeln angeben
-n --num-topology	1, 2 oder 4	Verteilungsmodus auf weitere Clients
-a --all	-	Auf allen Rechnern ausführen
-g --group	Rechnergruppe	Auf den Rechnern der genannten Rechnergruppe ausführen
-f --file	Dateiname	Auf allen in der Datei gelisteten Rechnern ausführen
-r --remoteshell	»rsh« oder »ssh«	Zu verwendende Remote-Shell
-o --remoteshellopt	beliebiger Ascii-Text	Optionen für die Remote-Shell
-h --help	-	Hilfe zur Bedienung ausgeben
-w --wait-shell	-	Sequenzielle Abarbeitung auf den Rechnern (Default-Fall)
-c --concurrent-shell	-	Parallel auf allen Rechnern arbeiten
-F --forklimit	Anzahl	Maximale Rechneranzahl bei paralleler Verarbeitung
-V --version	-	Gibt die Versionsnummer aus

Soll ein Kommando nur auf einzelnen Rechnern laufen, die nicht zu einer Gruppe zusammengefasst sind, kommt die Option »--machine« zum Einsatz. Sie verlangt eine durch Kommas getrennte Liste der Computer als Argument. Das Kommando »dsh --machine root@webserv,gate --show-machine-names uptime« führt zu der folgenden Ausgabe:

```
root@webserv:2:57pm up 9 days,7:29,0 users
gate:14:57:53 up 7 days,19:24,3 users
```

Rechnergruppen im NIS verwalten

Sofern im Netzwerk der Network Information Service (NIS, ehemals YP: Yellow Pages) seinen Dienst versieht, lassen sich auch über ihn Rechnergruppen spezifizieren. Ein zentraler NIS-Server stellt die Informationen zu Login-Namen, Passwörtern und Homeverzeichnissen, Gruppenzugehörigkeiten und Rechnernamen zur Verfügung. Der Dienst ergänzt die Einträge in beliebigen Konfigurationsdateien der Clients, etwa »/etc/passwd«, »/etc/groups« oder »/etc/hosts«. Ob und zu welcher Konfigurationsdatei der NIS-Server zu befragen ist, steht in »/etc/nsswitch.conf«.

Per »ypcat« lassen sich alle in der NIS-Map »netgroup« eingetragenen Rechner und Keys inklusive ihrer NIS-Domäne (hier »uni«) anzeigen:

```
$ ypcat -k netgroup
linux (vollmilch.local,,uni) ➥
  (zartbitter.local,,uni)
irix (jever.local,,uni) ➥
  (becks.local,,uni) (rothaus.local,,uni)
```

Die DSH kann in der NIS-Map »netgroup« alle Rechner abfragen, die einem bestimmten Key zugeordnet sind. Dabei erkennt die DSH an dem »@«-Zeichen,

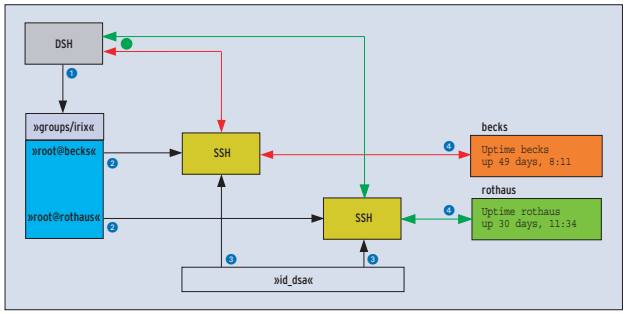

Abbildung 2: DSH liest beim Kommando »dsh --group irix uptime« aus der Gruppendatei »groups/irix« (❶), auf welchen Rechnern sie das Uptime-Kommando absetzen soll. Sie öffnet je eine SSH-Verbindung (❷) und authentifiziert sich mit einem DSA-Schlüssel (❸); die Ausgaben der Zielrechner (❹) sammelt sie (❺).

dass es in der NIS-Map »netgroup« nach allen Rechnern mit dem angegebenen Key suchen soll, nicht etwa in »~/.dsh/groups/irix«:

```
$ dsh --group @irix --show-machine-names ➥
  uname -a
rothaus: IRIX64 max 6.5 01101245 IP27
becks: IRIX64 olive 6.5 07151432 IP27
jever: IRIX cashew 6.2 03131015 IP22
```

Wie NIS zu konfigurieren ist und welche Sicherheitsaspekte es zu beachten gilt, ist auf [6] nachzulesen.

Lastverteilung möglich

Mit dem Schalter »--num-topology« verspricht DSH eine sternförmige Verteilung der Last auf andere Rechner aus der Liste. Dazu nutzt sie einige Rechner als Relaisstation für weitere Aufrufe. Da sich kaum festlegen lässt, welche Rechner dies sind, muss DSH auf allen in Frage kommenden Rechnern installiert sein, bevor diese Option Verwendung finden kann. Die SSH-Keys auf diesen Rechnern müssen außerdem darauf ausgelegt sein, eine Authentifizierung auf weiteren Clients zu erlauben, am einfachsten gelingt das mit Public-Key-Authentifizierung und leerer Passphrase. Fraglich ist dabei allerdings, ob der Nutzen den Aufwand rechtfertigt.

Unter normalen Umständen arbeitet DSH die Befehle sequenziell auf allen Rechnern ab. Um Zeit zu sparen und Verzögerungen durch hängende Prozesse oder nicht erreichbare Rechner zu vermeiden, sorgt der Schalter »--concurrent-shell« für eine parallelisierte Abarbeitung:

```
$ dsh --group clients --concurrent-shell ➥
  'last | grep renner'
```

Übersteigt die Anzahl der gleichzeitig angesprochenen Rechner (und damit auch die Anzahl der parallelen SSH-Prozesse) die Leistungsfähigkeit des Administrationsrechners, lässt sich die maximale Anzahl der gleichzeitig gestarteten Remote Shells mit der Option »--forklimit *Grenzwert*« festlegen. Im Zusammenhang mit der parallelen Abarbeitung steht auch die Option »--duplicate-input« zur Verfügung. Damit leitet die DSH alle Eingaben von der Standardeingabe (Tastatur, Pipe) unverändert an alle Clients weiter:

```
$ dsh --group linux --concurrent-shell ➥
  --duplicate-input 'cat > /tmp/foo'
```

Das Kommando überträgt den eingegebenen Text auf alle Rechner und schreibt ihn in die Datei »/tmp/foo«. Erst die Tastenkombination [Ctrl] + [C] beendet den Vorgang. Achtung: Der Programmaufruf enthält einen Stolperstein: Jetzt muss der Befehl für die Clients in einfachen Anführungszeichen stehen, zuvor waren sie nur optional.

Mit dieser Funktion lässt sich das eingangs aufgetauchte Problem der Schlüsselverteilung an alle Clients lösen. Doch sollte der Admin sorgfältig mit seinem SSH-Schlüssel umgehen und Schlüssel, die zu Root-Rechten führen, nur auf sicheren Rechnern ablegen.

Beim ersten Kontakt übermittelt der SSH-Daemon auf dem Zielrechner seinen öffentlichen Host-Key zum Admin-Rechner. Wer Sicherheit ernst nimmt, prüft an dieser Stelle den Fingerprint oder überträgt die Host-Keys über ein sicheres Medium. Wer einfach nur geduldig »yes« eintippt, setzt sich der Gefahr eines Man-in-the-Middle-Angriffs aus (siehe auch Artikel über ARP-Spoofing in diesem Heft).

Richtig angewandt reduziert die DSH die Administration eines Rechner-Pools auf die Frage, welches Kommando man auf welcher Gruppe von Computern auszuführen gedenkt. Doch ist eine falsche Konfiguration auch in Rekordzeit auf alle Rechner verteilt. Ein Grund mehr, jeden Befehl gründlich zu überdenken, bevor man ihn absetzt. *(fjl/jre)* ■

Installation

Wer Debian einsetzt, installiert DSH bequem per »apt-get install dsh«. Für andere Distributionen gibt es derzeit keine Binärpakete. Das Konvertierungstool »alien« [4] wandelt das Debian-Paket jedoch in das RPM-Format um.

Alternativ besorgt man sich den Sourcecode von »dsh« und der zugehörigen Bibliothek »libdshconfig« von der DSH-Website [1] und kompiliert das Programm selbst. Die Bibliothek ist zuerst zu installieren. Sie übernimmt später die Aufgabe, die Konfigurationsdateien der DSH zu parsen.

```
$ tar xfz libdshconfig-0.20.10.tar.gz
$ cd libdshconfig-0.20.10
$ ./configure && make
$ su
# make install
```

Anschließend lässt sich die DSH nach dem gleichen Schema installieren.

```
$ tar xfz dsh-0.25.1.tar.gz
$ cd dsh-0.25.1
$ ./configure && make
$ su
# make install
```

Nach der Installation liegt die DSH im Verzeichnis »/usr/local/bin/« bereit.

Infos

[1] DSH: [http://www.netfort.gr.jp/~dancer/software/dsh.html]
[2] Tentakel: [http://tentakel.biskalar.de/]
[3] SC-Venus: [http://www.science-computing.de/products/scvenus.html]
[4] Alien: [http://www.kitenet.net/programs/alien/]
[5] Karl-Heinz Haag und Achim Leitner, Artikelserie zu SSH: Linux-Magazin 05/02, S. 56; 07/02, S. 70 und 09/02, S. 72
[6] Thorsten Kukuk, „NIS für Linux", Linux-Magazin 09/98: [http://www.linux-magazin.de/Artikel/ausgabe/1998/09/NIS/nis.html]

Der Autor

Michael Renner arbeitet als Netzwerk- und Unix-Administrator an einem Max-Planck-Institut in Tübingen. Neben der Verwaltung der Linux-, Irix- und FreeBSD-Rechner gehört die aufopferungsvolle Betreuung der Anwender zu seinen täglichen Aufgaben.