

*Intelligente Virtualisierungs-
lösungen mit XEN 3*

**Deutsche
Originalausgabe**



XEN

Kochbuch



O'REILLY®

Hans-Joachim Picht

Vorwort	XI
1 Einführung in die Virtualisierung	1
1.1 Was ist Virtualisierung?	1
1.2 Die Geschichte der Virtualisierung	3
1.3 Einsatzgebiete und Vorteile	6
1.4 Virtualisierungsarten	8
1.5 Was ist Xen?	17
2 Xen installieren und das Hostsystem vorbereiten (Domain-0)	22
2.1 Xen-Installation auf Basis des Quellcodes	23
2.2 Erstellung einer initialen RAM-Disk	27
2.3 Anpassung des Boot-Managers	29
2.4 Xen unter Ubuntu installieren	32
2.5 Xen unter Debian installieren	35
2.6 Xen unter openSUSE installieren	37
2.7 Xen unter Fedora installieren	39
2.8 Installation der Binärpakete von xensource.com	41
2.9 Die TLS-Bibliothek deaktivieren	43
2.10 Starten, Stoppen und Neustarten des Xen-Daemons	46
2.11 Deinstallation von Xen unter Debian und Ubuntu	48
2.12 Deinstallation von Xen unter Fedora	49
2.13 Deinstallation von Xen unter openSUSE	50
2.14 Deinstallation von Xen als Binärpaket	51
2.15 Deinstallation eines aus den Quellen kompilierten Xen	52
2.16 Xen unter Debian und Ubuntu aktualisieren	53
2.17 Xen unter openSUSE aktualisieren	54

2.18	Xen unter Fedora-Linux aktualisieren.	55
2.19	Ein aus den Quellen oder als Binärpaket installiertes Xen aktualisieren. . .	56
2.20	Xend in den Boot-Vorgang einbinden.	57
2.21	Automatisches Starten der Gastsysteme	59
3	Konfiguration des Hostsystems (Domain-0)	60
3.1	Konfiguration des Xen-Daemons (xend).	60
3.2	Ein Bridging-Netzwerk einrichten.	66
3.3	Ein Routing-Netzwerk einrichten	73
3.4	Ein Netzwerk mit Network Address Translation konfigurieren	75
4	Vorbereitung des Systems auf die Gäste (Paravirtualisierung)	81
4.1	Speicherplatz auf Basis physikalischer Geräte einrichten	81
4.2	Speicherplatz auf Basis physikalischer Geräte mit LVM einrichten	83
4.3	Dateibasierte Geräte als Storage Backend einsetzen	89
4.4	Zugriff auf die root-Partition über NFS.	91
4.5	Erstellung des Dateisystems	93
4.6	Konfiguration des Gastsystems	94
4.7	Booten des Gastsystems	97
5	Distributionspezifische Gast-Installation (Paravirtualisierung)	101
5.1	Gastsysteme mit virt-install unter Fedora installieren.	101
5.2	Gastsysteme mit YaST auf SUSE installieren	105
5.3	Gastsysteme für Debian und Ubuntu mit Xen-Tools installieren	108
5.4	Paravirtualisiertes NetBSD	110
6	Unmodifizierte Gastsysteme	115
6.1	Einführung: Der Unterschied zwischen Para- und Hardware-Virtualisierung	115
6.2	Xen und die x86-Ringe paravirtualisiert	118
6.3	Hardware-Unterstützung prüfen.	119
6.4	Konfiguration für unmodifizierte Gastsysteme	122
6.5	Von einem CD/DVD-Laufwerk booten	123
6.6	Von einer (virtuellen) Festplatte booten	124
6.7	Booten einer Live-CD/DVD	125
6.8	Installation von Linux	128
6.9	Installation von Windows	131
6.10	Benutzung der virtuellen Windows-Systeme.	135

7	Hardwarezugriff	138
	7.1 Hotplugging virtueller Festplatten	138
	7.2 Hotplugging virtueller Netzwerkkarten	140
	7.3 Ein virtuelles Trusted Platform-Module (TPM) nutzen	142
	7.4 Der virtuelle Framebuffer	144
	7.5 PCI-Karten in einem Gastsystem zugänglich machen	146
	7.6 Eine Backend-Domain konfigurieren	147
	7.7 OpenGL-Hardware-3-D-Beschleunigung für virtuelle Maschinen: VMGL	149
8	Automatische Installation	153
	8.1 Einführung	153
	8.2 Installation minimaler Linux-Systeme in ein Verzeichnis	154
	8.3 Installation eines minimalen SUSE-Systems für eine virtuelle Maschine	158
	8.4 Installation einer virtuellen Maschine auf Basis einer RPM-basierten Linux-Distribution	161
	8.5 Einsatz vorgefertigter DomUs (paravirtualisiert)	164
	8.6 Austausch der Kernel-Module innerhalb der virtuellen Maschine	166
	8.7 Klonen virtueller Maschinen	168
	8.8 Vorbereitungen zum Klonen einer hardwarevirtualisierten Windows- Domäne mittels sysprep	171
9	Backup & Restore	173
	9.1 Den Zustand einer virtuellen Maschine speichern	173
	9.2 Eine virtuelle Maschine anhalten	175
	9.3 Backup via FTP	176
	9.4 Backup mit rsync	178
	9.5 Backup mit LVM	181
	9.6 Xen & RAID: Virtuelle Maschinen auf einem Software-RAID-1 betreiben	183
10	Migration virtueller Maschinen	189
	10.1 Einführung: Unterschiedliche Migrationskonzepte im Überblick	189
	10.2 Physical-to-Virtual-Migration (P2V) mit Hardwarevirtualisierung	190
	10.3 P2V-Migration ohne Hardwarevirtualisierung	195
	10.4 Ein physikalisches Fedora/Red Hat Enterprise Linux virtualisieren	199
	10.5 P2V von OpenSUSE/SUSE Linux Enterprise-Servern	202
	10.6 Von VMware zu Xen migrieren	204

11	Live-Migration	208
	11.1 Konfiguration des Xen-Relocation-Servers	208
	11.2 Live-Migration zwischen verschiedenen Hosts	211
12	Hochverfügbarkeit	218
	12.1 Verteilte Replikation von Dateisystemen via DRBD	220
	12.2 Xen-Cluster mit Ganeti verwalten	226
	12.3 Hochverfügbarkeitscluster mit Heartbeat einrichten	240
	12.4 Virtuelle Xen-Gäste im Hochverfügbarkeitscluster	244
13	Monitoring	249
	13.1 Einführung	249
	13.2 Monitoring mit Xens Bordmitteln	250
	13.3 Virtuelle Maschinen unter Fedora Linux überwachen	261
	13.4 Die CPU-Auslastung mit XenStats auswerten	263
	13.5 Monitoring virtueller Maschinen mithilfe von Nagios – vorbereitende Schritte	267
	13.6 Monitoring von Domain-0s und ihren virtuellen Maschinen mit Nagios	284
	13.7 Alternative Lösungen	291
14	Virtuelle Maschinen verwalten	295
	14.1 Textbasierte Verwaltung virtueller Maschinen mit der Xen-Shell	296
	14.2 Webbasierte Verwaltung mit Enomalism	299
	14.3 Grafische Verwaltung virtueller Maschinen mit XenMan	301
	14.4 Grafische Verwaltung virtueller Maschinen mit dem Virtual Machine Manager (virt-manager)	303
15	Eigene Anwendungen zum Xen-Management entwickeln	306
	15.1 Einführung: Die libvirt-Bibliothek	306
	15.2 Die libvirt unter Fedora installieren	308
	15.3 Die libvirt unter openSUSE installieren	309
	15.4 Die libvirt unter Debian installieren	310
	15.5 Die libvirt aus dem Quellcode installieren	311
	15.6 Das erste C-Programm mit libvirt	314
	15.7 Das erste Python-Programm mit libvirt	316
	15.8 Die Fähigkeiten des Hypervisors in C-Programmen anzeigen	317
	15.9 Die Fähigkeiten des Hypervisors in Python-Skripten anzeigen	318
	15.10 Eine Domäne verwalten	319
	15.11 Eine neue Domäne anlegen	321

15.12	Eine Domäne herunterfahren	322
15.13	Eine Domäne »einschlafen« lassen	323
15.14	Eine Domäne aufwecken.	324
15.15	Die Gastsysteme in einem C-Programm auflisten.	325
15.16	Die Gastsysteme in einem Python-Skript auflisten	327
15.17	Die Geräte einer Domäne mit einem C-Programm verwalten	328
15.18	Die Geräte einer Domäne mit einem Python-Skript verwalten.	329
16	Der XenStore	331
16.1	Einführung: Aufbau und Navigation der XenStore-Verzeichnisstruktur	332
16.2	Informationen auslesen.	333
16.3	Einträge anlegen und modifizieren	334
16.4	Einträge entfernen.	335
17	Ressourcenkontrolle	336
17.1	Einführung in die Speicherverwaltung	336
17.2	Speicherlimits für virtuelle Maschinen einrichten.	337
17.3	Einführung: CPUs verwalten.	340
17.4	Zuordnung zwischen virtuellen CPUs und Gastsystemen	340
17.5	Zuordnung zwischen logischen CPUs und Gastsystemen	343
17.6	Scheduling von CPU-Zyklen.	344
17.7	Traffic Shaping mit Xen	346
18	Sicherheit.	348
18.1	Einführung	348
18.2	Die Daten virtueller Maschinen verschlüsseln	350
18.3	Vorüberlegungen zum Firewall-Setup unter Xen	358
18.4	Firewall-Konfiguration in einem Bridge-Setup (Dom0)	359
18.5	Firewall-Konfiguration in einem NAT-Setup (Dom0)	366
18.6	Firewall-Konfiguration in einem Routing-Setup (DomU).	368
18.7	Firewall-Setup mit der DTC-Xen-Firewall unter Debian GNU/Linux oder Ubuntu	371
18.8	Mechanismen zur Zugriffskontrolle einrichten.	374
18.9	Xen und Intrusion-Detection-Systeme (IDS).	388
19	Troubleshooting.	391
19.1	Zeitabweichung in der virtuellen Maschine	391
19.2	Die Fehlermeldung »Backend Device not found«.	393
19.3	Die Meldung »Is a directory«	396

19.4	Die Meldung »Error: (22, 'Invalid argument')«	397
19.5	»4gb seg fixup« – eine Fehlermeldung unter Debian	399
19.6	Einen Bugreport einreichen	400
20	Erweiterte Netzwerkkonfiguration	404
20.1	Eine VLAN-Umgebung einrichten	404
20.2	Zugriff auf ein VLAN von der Domain-0 einrichten	406
20.3	Zugriff auf ein VLAN von einer Domain-U einrichten	408
20.4	Einer virtuellen Maschine den Zugriff auf ein VLAN ermöglichen	409
20.5	Bonding-Unterstützung aktivieren	410
20.6	Ein Bonding-Device einsetzen	412
A	Das Programm xm im Überblick	416
	Virtuelle Maschinen verwalten	417
	Befehle zur Steuerung der Domain-0	423
	Virtuelle Blockgeräte verwalten	424
	Virtuelle Netzwerkadapter verwalten	426
	Virtuelle TPM-Geräte	427
	Access Control Lists	428
B	Glossar	432
C	Weiterführende Informationen	438
	Literaturhinweise	438
	Mailinglisten	439
	Internet Relay Chat (IRC)	444
	Webseiten	444
	Veranstaltungen	446
Index		447

Monitoring

Monitoring war in den letzten Jahren sowieso immer wieder ein Diskussionsthema und bekommt noch einmal besondere Brisanz durch den aktuellen Virtualisierungs-Hype. Durch Virtualisierung multipliziert sich die Monitoring-Komplexität, da neben den physikalischen Maschinen auch die virtuellen Maschinen ein Monitoring benötigen. Wenn es zum Beispiel zu Lastproblemen kommt, sollte die dafür verantwortliche virtuelle Maschine zügig identifiziert werden, da eventuell auch andere Gastsysteme auf demselben Host davon betroffen sein können. Außerdem kommen durch die Virtualisierung natürlich auch einfach neue Komponenten hinzu, die es zu überwachen gilt.

Daher beschäftigt sich dieses Kapitel mit dem Warum, Was, Wann, Wo und Wie des Monitorings. Wir sprechen in diesem Kapitel übrigens absichtlich von Monitoring – nicht weil wir die Verwendung von Anglizismen in deutschen Texten fördern wollen, sondern weil das Wort »Überwachung« vielseitig negativ vorbelastet ist.

13.1 Einführung

Warum Monitoring?

Auf die Frage, warum sich das Monitoring von Systemen im Allgemeinen lohnt, gibt es gleich mehrere Antworten. In einem minimalistischen Setup geht es im Wesentlichen darum, sich darüber informieren zu lassen, wenn etwas nicht mehr funktioniert oder bereits ausgefallen ist, um entsprechend darauf reagieren zu können. Nichts ist weniger erfreulich, als erst vom Kunden oder einem verärgerten Endanwender zu erfahren, dass eine bestimmte Funktionalität nicht mehr zur Verfügung steht.

Daneben gibt es den Ansatz des proaktiven Monitorings, bei dem versucht wird, Probleme vorherzusehen und sie so vielleicht zu vermeiden. Ein klassisches Beispiel hierfür ist die Überwachung der Festplattenauslastung: Es ist sehr sinnvoll, Informationen über die Festplattenauslastung zu sammeln, um abzusehen, ob der zur Verfügung stehende Speicherplatz ausreichend ist, um dann rechtzeitig eine neue Festplatte beschaffen und den Einbau zeitlich planen zu können. Selbiges gilt natürlich auch für andere Parameter wie Hauptspeicher, Netzwerkauslastung usw.

Außerdem ist die Aufbereitung von Monitoring-Daten in grafischer Form bei der Systemanalyse hilfreich. Hierbei können unter anderem Lastspitzen erkannt, Trends über die Auslastung ermittelt oder Entscheider von der Notwendigkeit weiterer Investitionen überzeugt werden.

Typische Fragen, die ein Monitoring beantworten soll

- Warum ist die Auslastung des Proxy-Servers Freitagnachmittag besonders hoch?
- Wieso erreicht der Mailserver seine Lastspitze montags gegen neun?
- Weshalb ist der Server xy immer donnerstagabends für einen Zeitraum von dreißig Minuten nicht erreichbar?
- Hat ein Xen-Host Kapazitäten für weitere virtuelle Maschinen?
- Wie stark sind die Systeme im Moment gerade ausgelastet?
- Werden bestimmte Schwellwerte erreicht bzw. überschritten?
- Wie hoch ist die durchschnittliche Auslastung?
- Zeichnen sich Trends ab?

13.2 Monitoring mit Xens Bordmitteln

Problem

Mit welchen Xen-Bordmitteln ist es möglich, verschiedene Parameter einer Xen-Installation zu überwachen?

Lösung

Das für Xen zentrale `xm`-Kommando kann nicht nur verwendet werden, um virtuelle Maschinen zu starten, zu beenden, zu pausieren oder zu migrieren, sondern auch, um Informationen über virtuelle Maschinen und die Domain-0 abzufragen:

Anzeige von Informationen über die Domain-0 und laufende Gastsysteme:

```
root@knuth:~# xm list
Name                                ID Mem(MiB) VCPUs State  Time(s)
Domain-0                            0  1764      1 r----- 3961.0
xenvm1.xencookbook.com              3   256      1 -b----- 5.0
```

Anzeige der Uptime von Domain-0 und der aktiven virtuellen Maschinen:

```
root@knuth:~# xm uptime
Name                                ID Uptime
Domain-0                            0 8 days, 16:46:38
xenvm1                               3 6 days, 20:13:06
```

Auswertung der Auslastung in Gast und Host in Echtzeit:

```
root@knuth:~# xm top
```

Die Ausgabe des Befehls `xm top` sehen Sie in Abbildung 13-1.

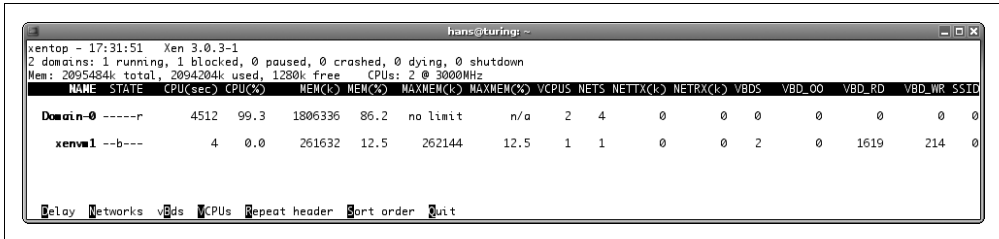


Abbildung 13-1: Ausgabe des Befehls »xm top«

Abfrage von Informationen über das Hostsystem:

```
root@knuth:~# xm info
host                : knuth.xencookbook.com
release             : 2.6.18-6-xen-686
version             : #1 SMP Wed Jan 23 07:51:35 UTC 2008
machine             : i686
nr_cpus              : 2
nr_nodes            : 1
sockets_per_node    : 1
cores_per_socket    : 2
threads_per_core    : 1
cpu_mhz             : 3000
hw_caps              : bfebfbff:20000000:00000000:00000180:0000e49d:00000000:00000001
total_memory        : 2046
free_memory          : 1
xen_major            : 3
xen_minor           : 0
xen_extra            : .3-1
xen_caps             : xen-3.0-x86_32p
xen_pagesize        : 4096
platform_params     : virt_start=0xf5800000
xen_changeset       : Tue Oct 17 22:09:52 2006 +0100
cc_compiler          : gcc version 3.4.6 (Debian 3.4.6-5)
cc_compile_by       : skx
cc_compile_domain   : debian.org
cc_compile_date     : Tue Oct 23 02:22:48 BST 2007
xend_config_format  : 2
```

Auflistung der virtuellen Prozessoren:

```
root@knuth:~# xm vcpu-list
Name                ID VCPUs  CPU State  Time(s) CPU Affinity
Domain-0            0  0      0  r--      2630.9 any cpu
Domain-0            0  1      -  --p      1470.1 any cpu
xenvm1.xencookbook.com 3  0      1  -b-       5.0 any cpu
```

Anzeige der Blockdevices:

```
root@knuth:~# xm block-list xenvm1
Vdev BE handle state evt-ch ring-ref BE-path
769  0  0  4  6  8  /local/domain/0/backend/vbd/6/769
770  0  0  4  7  9  /local/domain/0/backend/vbd/6/770
```

Anzeige der Netzwerkkonfiguration:

```
root@knuth:~# xm network-list xenvm1.xencookbook.com
Idx BE      MAC Addr.      handle state evt-ch tx-/rx-ring-ref BE-path
0 0 00:16:3e:70:01:04 0 4 8 522 /523 /local/domain/0/backend/vif/
3/0
```

Darstellung der xend-Protokolldatei:

```
root@knuth:~# xm log
[2008-02-08 02:00:11 xend 2549] INFO (__init__:1072) Xend Daemon started
[2008-02-08 02:00:11 xend 2549] INFO (__init__:1072) Xend changeset:
Tue Oct 17 22:09:52 2006 +0100 .
[...]
```

Anzeige des xend-Message-Buffers (analog zum dmesg-Kommando):

```
root@knuth:~# xm dmesg
Xen version 3.0.3-1 (Debian 3.0.3-0-4) (skx@debian.org) (gcc version 3.4.6
(Debian 3.4.6-5)) Tue Oct 23 02:22:48 BST 2007
Latest ChangeSet: Tue Oct 17 22:09:52 2006 +0100

(XEN) Command line: /xen-3.0.3-1-i386-pae.gz
[...]
(XEN) Dom0 has maximum 2 VCPUs
(XEN) Initrd len 0xa33a00, start at 0xc0397000
(XEN) Scrubbing Free RAM: .....done.
(XEN) Xen trace buffers: disabled
(XEN) Xen is relinquishing VGA console.
(XEN) *** Serial input -> DOM0 (type 'CTRL-a' three times to switch input to Xen).
```

Auflistung der Partitionen und des verfügbaren Speicherplatzes:

```
root@knuth:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda3        457G  23G  411G   6% /
tmpfs            948M   0  948M   0% /lib/init/rw
udev             10M   48K   10M   1% /dev
tmpfs            948M   0  948M   0% /dev/shm
/dev/sda2        92M   25M   62M  29% /boot
```

Diskussion

Anzeige von Informationen über die Domain-0 und laufende Gastsysteme

Die Ausgabe von `xm list` listet Informationen über den aktuellen Zustand der Domain-0 und der aktiven Gastsysteme in einer sechsspaltigen Ausgabe auf. Die Bedeutung der einzelnen Einträge erläutert Tabelle 13-1.

Tabelle 13-1: Bedeutung der einzelnen Spalten, die in der Ausgabe von »xm list« angezeigt werden

Name	Der Name der virtuellen Maschine.
ID	Eine ID, anhand derer die virtuelle Maschine identifiziert wird. Die Domain-0 hat an dieser Stelle immer die Nummer 0. Gastsystemen wird dieser inkrementelle Zähler immer beim Systemstart dynamisch zugewiesen.
Mem(MiB)	Der maximale zugewiesene Hauptspeicher in MByte.
VCPUs	Die Anzahl der virtuellen CPUs.
State	Der Zustand, in dem sich die virtuelle Maschine befindet. Es werden folgende Zustände unterschieden: r (running): Die virtuelle Maschine wird zurzeit aktiv ausgeführt. b (blocked): Die virtuelle Maschine ist blockiert, weil sie auf eine I/O-Operation wartet, oder sie schläft, weil sie zurzeit nichts zu tun hat. p (paused): Die virtuelle Maschine wurde angehalten. In diesem durch xm pause herbeigeführten Zustand verbraucht der Gast weiterhin Hauptspeicher, bekommt jedoch keine CPU-Zyklen mehr zugewiesen. s (shutdown): Der Gast befindet sich im Shutdown-, Reboot- oder Suspend-Vorgang und wird demnächst beendet. c (crashed): Die virtuelle Maschine ist gecrasht, zum Beispiel durch eine fehlerhafte Konfiguration oder einen fehlgeschlagenen Reboot-Vorgang. d (dying): Die virtuelle Maschine »stirbt«, hat jedoch weder den Shutdown-Prozess noch einen Crash vollständig vollziehen können.
Time	Die verbrauchte CPU-Zeit in Sekunden, die Xen zugeschrieben wird.

Sollen nur Informationen über eine bestimmte Maschine angezeigt werden, kann man dem Befehl `xm list` den Namen der virtuellen Maschine wie folgt als Parameter übergeben:

```

root@knuth:~# xm list xenvm1
Name                               ID Mem(MiB) VCPUs State  Time(s)
xenvm1                             6    256     1 -b---- 262.6
  
```

Noch detailliertere Informationen liefert die Ausgabe des folgenden Befehls:

```

root@knuth:~# xm list -l
  
```

Diese Ausgabe lässt sich wegen ihres relativ einfachen Aufbaus auch leicht zur Auswertung mittels selbst entwickelter Skripte heranziehen. Eine beispielhafte Ausgabe sieht in etwa so aus:

```

root@knuth:~# xm list -l
(domain
  (domid 0)
  (uuid 00000000-0000-0000-0000-000000000000)
  (vcpus 2)
  (cpu_weight 1.0)
  (memory 1764)
  (shadow_memory 0)
  (maxmem 1764)
  (features )
)
  
```

```

(name Domain-0)
(on_poweroff destroy)
(on_reboot restart)
(on_crash restart)
(state r-----)
(shutdown_reason poweroff)
(cpu_time 602120.295495)
(online_vcpus 1)
)
(domain
  (domid 6)
  (uuid 3a26ff70-19ab-c906-dc20-82156927e2b4)
  (vcpus 1)
  (cpu_weight 1.0)
  (memory 256)
  (shadow_memory 0)
  (maxmem 256)
  (features )
  (name xenvm1)
  (on_poweroff destroy)
  (on_reboot restart)
  (on_crash restart)
  (image
    (linux
      (kernel /boot/vmlinuz-2.6.18-5-xen-686)
      (ramdisk /boot/initrd.img-2.6.18-5-xen-686)
      (root '/dev/hda1 ro')
    )
  )
  (device
    (vif
      (backend 0)
      (script vif-bridge)
      (bridge xenbr0)
      (mac 00:16:3e:70:01:04)
    )
  )
  (device
    (vbd
      (backend 0)
      (dev hda1:disk)
      (uname file:/opt/xen/xenvm1/hda1.img)
      (mode w)
    )
  )
  (device
    (vbd
      (backend 0)
      (dev hda2:disk)
      (uname file:/opt/xen/xenvm1/hda2.img)
      (mode w)
    )
  )
  (state -b----)
)

```

```
(shutdown_reason poweroff)
(cpu_time 262.669601729)
(online_vcpus 1)
(up_time 591650.532566)
(start_time 1202667105.97)
(store_mfn 464306)
(console_mfn 55134)
)
```

Das S-Expression-Format

Mit dem Begriff *S-Expression* (auch oftmals abgekürzt als *sexp*, wobei das S für »symbolic« steht) bezeichnet man ein lesbares, textbasiertes Format zur Strukturierung komplexer Datensätze.

Besonders verbreitet ist dieses Format in der Welt der von John McCarthy entwickelten Programmiersprache LISP bzw. verwandter Programmiersprachen.

Im Jahre 1997 hat der Kryptografie- und Informatik-Professor Ron Rivest, der unter anderem zu den Erfindern des RSA-Algorithmus gehört, einen Internet-Draft eingereicht, mit dem Ziel der Veröffentlichung eines RFCs, der die *S-Expression* als allgemein anerkanntes, generelles Format zur Beschreibung von Daten etablieren sollte, vergleichbar dem heutzutage verwendeten XML. Das ist jedoch gescheitert, und während vielen das XML-Format geläufig ist, wird *S-Expression* – dem subjektiven Empfinden des Autors zufolge – primär in Software aus dem akademischen Umfeld verwendet.

Rivest nennt folgende Design-Ziele für *S-Expression*:

- *Generalität*: *S-Expression* sollte gut beliebige Daten repräsentieren können.
- *Lesbarkeit*: Das Format sollte einfach zu lesen und die Struktur der Daten einfach zu erkennen sein.
- *Wirtschaftlichkeit*: *S-Expression* soll Daten auf kompakte Art und Weise darstellen können.
- *Transportierbarkeit*: *S-Expression* soll einfach über Kommunikationsmedien (wie E-Mail) transportierbar sein, die weit davon entfernt sind, perfekt zu funktionieren.
- *Flexibilität*: *S-Expression* soll relativ einfach zu modifizieren und durch weitere Datenstrukturen zu erweitern sein.
- *Kanonisierung*: Es sollte einfach sein, einen eindeutigen »Kanon« (im Sinne des englischen Begriffs *canonical*, von autorisiert bzw. anerkannt) einer *S-Expression* für den Zweck einer digitalen Signatur zu erstellen.
- *Effizienz*: *S-Expressions* sollen wenig Hauptspeicher verbrauchen und sich effektiv verarbeiten lassen.

– Fortsetzung –

Das Format besteht aus Ausdrücken, die aus Elementen bestehen können, die entweder als »Atome« oder *S-Expressions* bezeichnet werden. Im Folgenden sehen Sie Beispiele für *S-Expressions*, die aus zwei Elementen bestehen:

```
(a b)
(a (b c))
((a b) (c d))
(() ())
```

Während das erste Beispiel aus zwei atomaren Einträgen, den beiden Strings »a« und »b«, besteht, finden sich im zweiten Ausdruck sowohl ein »Atom« als auch eine *S-Expression* (die wiederum aus zwei »Atomen« besteht).

Das dritte Beispiel ist aus zwei »Atomen« zusammengesetzt, während das letzte aus zwei leeren *S-Expressions* besteht.

Siehe auch:

- <http://people.csail.mit.edu/rivest/Sexp.txt> – der von Ron Rivest verfasste Internet-Draft mit dem Ziel der Standardisierung von *S-Expression*
- <http://sexpr.sourceforge.net/> – eine kleine, schnelle *SEXP*-Bibliothek, aus deren Dokumentation auch das obige Beispiel stammt
- <http://www.clsp.jhu.edu/~edrabe/utills/> – ein Parser für *S-Expressions* in Python
- <http://blog.lab49.com/archives/2220> – eine Anleitung zur Entwicklung eines einfachen *S-Expression*-Parsers

Anzeige der Uptime von Domain-0 und den aktiven virtuellen Maschinen

Anhand der Ausgabe von `xm uptime` ist ersichtlich, dass die Domain-0 bereits seit acht Tagen, 16 Stunden, 46 Minuten und 38 Sekunden in Betrieb ist. Die virtuelle Maschine hingegen wurde erst vor sechs Tagen, 20 Stunden und 13 Minuten sowie 6 Sekunden gestartet. Soll die Information der Uptime in Verbindung mit einem aktuellen Zeitstempel ausgegeben werden, kann der Parameter `-s` mit herangezogen werden:

```
root@knuth:~# xm uptime -s
15:33:34 up 8 days, 16:55, Domain-0 (0)
15:33:34 up 6 days, 20:21, xenvm1 (6)
```

Auswertung der Auslastung in Gast und Host in Echtzeit

Die Ausgabe von `xm top` (auch unter dem Namen `xentop` bekannt) gliedert sich in einen Header, gefolgt von einer sich in regelmäßigen Intervallen aktualisierenden Tabelle:

```
xentop - 15:35:56 Xen 3.0.3-1
2 domains: 1 running, 1 blocked, 0 paused, 0 crashed, 0 dying, 0 shutdown
Mem: 2095484k total, 2094204k used, 1280k free CPUs: 2 @ 3000MHz
```

Aus dem Header geht hervor, dass wir die Ausgabe von `xentop` in der Version, die im Xen 3.0.3 enthalten ist, sehen. Aktuell werden zwei Systeme ausgeführt (neben der Domain-U zählt hierzu auch die Domain-0).

Von den zwei aktiven Systemen befindet sich eines im Modus `running`, und ein anderes ist zurzeit blockiert (`blocked`). Es ist weder eine pausierte noch eine abgestürzte noch eine tote oder sich im Shutdown-Vorgang befindliche Maschine zu finden.

Augenscheinlich stehen insgesamt 2.095.484 KByte Hauptspeicher (2046 MByte) zur Verfügung, von denen 2.094.204 KByte belegt sind und 1280 KByte zur Verfügung stehen.

Die zugrundeliegende Hardware ist dabei ein Multi-Prozessor-System mit 3000 Megahertz. Die nachfolgende Tabelle 13-2 zeigt eine Liste der Spalten in der Ausgabe von `xm top` sowie deren Bedeutung.

Tabelle 13-2: Spalten in der Ausgabe von `xm top`

Header in <code>xm top</code>	Beschreibung
NAME	Der Name des Systems
STATE	Der aktuelle Status
CPU(sec)	Die Anzahl verbrauchter CPU-Zeit in Sekunden
CPU(%)	Die Anzahl verbrauchter CPU-Zeit in Prozent, gemessen am Gesamtsystem
MEM(k)	Der verbrauchte Hauptspeicher in KByte
MEM(%)	Der verbrauchte Hauptspeicher in KByte, gemessen am Gesamtsystem
MAXMEM(k)	Wie viele KByte Hauptspeicher dieses System maximal verbrauchen darf
MAXMEM(%)	Verhältnis in Prozent zwischen dem aktuellen und dem maximal erlaubten Hauptspeicherverbrauch des Systems
VCPUS	Anzahl der virtuellen CPUs
NETS	Anzahl der virtuellen Netzwerke
NETTX(k)	Anzahl der gesendeten Netzwerkpakete in KByte
NETRX(k)	Anzahl der empfangenen Netzwerkpakete in KByte
VBDS	Virtual Blockdevices
VBD_OO	Anzahl der »Out of«-Requests eines Virtual Blockdevice
VBD_RD	Anzahl der Lesezugriffe eines Virtual Blockdevice
VBD_WR	Anzahl der Schreibzugriffe eines Virtual Blockdevice
SSID	Security-ID

Während der interaktiven Bedienung stehen noch die in Tabelle 13-3 aufgeführten Tastenkombinationen zur Verfügung.

Tabelle 13-3: Tastenkombinationen von `xm top`

Tastenkombination	Auswirkung
D	Verzögerung zwischen Update-Intervallen in Sekunden
N	Umschaltung der Netzwerkanzeige
Q oder Esc	Beenden von <code>xentop</code>
R	Spaltenüberschriften einblenden
S	Ausgabe sortieren
V	Informationen über virtuelle CPUs ein- oder ausblenden
Pfeiltasten	In der Ausgabe nach oben bzw. unten blättern

Abfrage von Informationen über das Hostsystem

Die Ausgabe von `xm info` listet die in Tabelle 13-4 aufgeführten Details auf.

Tabelle 13-4: Ausgabe des Kommandos `xm info`

Zeile	Beschreibung
<code>host</code>	Hostname
<code>release</code>	Kernel-Release (entsprechend <code>uname -r</code>)
<code>version</code>	Kernel-Version (entsprechend <code>uname -v</code>)
<code>machine</code>	Die Prozessorklasse
<code>nr_cpus</code>	Die Anzahl der CPUs
<code>nr_nodes</code>	Die Anzahl der Knoten (1 auf Nicht-NUMA-Systemen)
<code>sockets_per_node</code>	Die Anzahl der Sockets pro Knoten
<code>cores_per_socket</code>	Die Anzahl der Kerne pro Socket
<code>threads_per_core</code>	Die Anzahl der Threads, die ein Kern gleichzeitig ausführen kann
<code>cpu_mhz</code>	Der CPU-Takt in Megahertz
<code>hw_caps</code>	Ein Vektor, der Auskunft über die Hardware-Features des Prozessors gibt, wie sie auch in der <code>flags</code> -Zeile der Ausgabe von <code>/proc/cpuinfo</code> enthalten sind
<code>total_memory</code>	Größe des gesamten Hauptspeichers in MByte
<code>free_memory</code>	Größe des freien Hauptspeichers in MByte, der momentan weder von virtuellen Maschinen noch von der Domain-0 verbraucht wird
<code>xen_major</code>	Die Major-Versionsnummer von Xen (z.B. 3.x)
<code>xen_minor</code>	Die Minor-Versionsnummer (z.B. 0 im Falle von 3.0.3-1)
<code>xen_extra</code>	Die zusätzliche Versionsnummer
<code>xen_caps</code>	Die Xen-Version, die Architektur sowie Architekturdetails wie <code>x86_32</code> , <code>x86_32p</code> (mit PAE-Unterstützung) bzw. <code>x86_64</code> oder <code>ia64</code>
<code>xen_pagesize</code>	Die Größe einer Seite im Speicher
<code>platform_parameters</code>	Zusätzliche Parameter, abhängig von der Version des Hypervisors
<code>xen_changeset</code>	Die ID des <i>Mercurial-Changesets</i> – an dieser Stelle kann abgelesen werden, welche Version des Xen-Quellcodes genau herangezogen wurde, um die vorliegende Xen-Version zu kompilieren

Tabelle 13-4: Ausgabe des Kommandos `xm info` (Fortsetzung)

Zeile	Beschreibung
<code>cc_compiler</code>	Die Version des gcc-Compilers, mit der diese Xen-Version übersetzt wurde
<code>cc_compile_by</code>	Der Benutzername derjenigen Person, die diese Xen-Version kompiliert hat
<code>cc_compile_domain</code>	Der Domainname des Hosts, auf dem diese Xen-Version kompiliert wurde
<code>cc_compile_date</code>	Der Tag, an dem diese Xen-Version kompiliert wurde
<code>xend_config_format</code>	Die Nummer des Versionsformats

Auflistung der virtuellen Prozessoren

Aus der Ausgabe von `xm vcpu-list` ist ersichtlich, dass jede einer Domäne zugeordnete virtuelle CPU in einer separaten Zeile angezeigt wird. Die einzelnen Spalten der Ausgabe, die keine Übereinstimmung mit der Ausgabe von `xm info` haben, haben dabei die in Tabelle 13-5 erläuterten Bedeutungen.

Tabelle 13-5: Ausgabe des Kommandos `xm vcpu-list`

Spalte	Beschreibung
VCPU	ID der virtuellen CPU, diese ist innerhalb der Domäne eindeutig.
CPU	ID der logischen CPU, auf der diese virtuelle CPU momentan ausgeführt wird.
CPU Affinity	Gibt an, auf welchen logischen CPUs die entsprechende virtuelle CPU laufen kann. In unserem Beispiel steht dort »any cpu« – die virtuelle CPU kann also auf jeder logischen CPU ausgeführt werden.

Anzeige der Blockdevices

`xm block-list` führt eine Übersicht der Blockdevices auf, die einer virtuellen Maschine zugeordnet sind. Neben der kurzen Variante steht durch Verwendung des Parameter `-l` auch wieder eine ausführliche Ausgabe zur Verfügung:

```
root@knuth:~# xm block-list xenvm1 -l
(769
  ((backend-id 0)
    (virtual-device 769)
    (device-type disk)
    (state 4)
    (backend /local/domain/0/backend/vbd/6/769)
    (ring-ref 8)
    (event-channel 6)
  )
)
(770
  ((backend-id 0)
    (virtual-device 770)
    (device-type disk)
    (state 4)
    (backend /local/domain/0/backend/vbd/6/770)
```

```

        (ring-ref 9)
        (event-channel 7)
    )
)

```

Tabelle 13-6 zeigt eine detaillierte Erläuterung der Ausgabe von `xm block-list`.

Tabelle 13-6: Ausgabe des Kommandos `xm block-list`

Spalte	Beschreibung
backend-id	ID der Backend-Domäne
virtual-device	Gerätenummer des Frontend-Geräts
device-type	Typ des Geräts
state	Status des Geräts
backend	XenStore-Eintrag des Backend-Geräts
ring-ref	Referenz-ID des Ringpuffers für blockbasierte Anfragen
event-channel	Event-Kanal, der für diesen Ringpuffer verwendet wird

Anzeige der Netzwerkkonfiguration

Der Befehl `xm network-list` listet die virtuellen Netzwerk-Interfaces einer jeden Domain auf. Hierbei steht neben der kurzen und bündigen Version auch eine ausführliche Variante im S-Expression-Format zur Verfügung, sofern die Option `-l` oder `--long` mit angegeben wird.

Darstellung der xend-Protokolldatei

`xm log` ermöglicht die Anzeige der Xen-Protokolldatei über das Kommando `xm`. Diese Datei kann jedoch auch direkt, z.B. durch das Programm `tail`, angezeigt werden. Abhängig von der eingesetzten Distribution, variiert der Pfad zu der Logdatei zwischen `/var/log/xend.log` und `/var/log/xen/xend.log`.

Anzeige des xend-Message-Buffers (analog zum dmesg-Kommando)

Der Befehl `xm dmesg` gibt den Xen-Message-Buffer auf der Kommandozeile aus. Dabei handelt es sich um einen Speicherplatz, der Informationen, Warnungen und handfeste Fehlermeldungen im Kontext von Xen sammelt, die im Zuge des Xen-Boot-Prozesses aufgetreten sind. Dies ist vergleichbar mit dem aus Linux-Systemen bekannten `dmesg`-Programm, in dessen Speicher sich nach dem Systemstart ebenfalls Informationen über den Boot-Vorgang befinden. Durch die Angabe des Parameters `-c` (für clear) kann dieser Speicher gelöscht werden.

Auflistung der Partitionen und des verfügbaren Speicherplatzes

Der Befehl `df` (für disk free) zeigt eine Liste aller zurzeit gemounteter Dateisysteme. Der verwendete Parameter `-h` steht für human-readable. Dies ist ein Format, in dem die angezeigten Größen des gesamten, des freien und des belegten Speichers auf den Partitionen

in Formate, die für Menschen leichter lesbar sind, gerundet werden. Im obigen Beispiel sind Angaben in Kilobyte (K), Megabyte (M), sowie Gigabyte (G) zu sehen. Wird `df` ohne Parameter aufgerufen, werden Größenangaben in 1-KByte-Blöcken ausgegeben.

Siehe auch

- `DF(1)` – die Manpage des Programms `disk-free`, das Informationen zum Speicher-verbrauch von Dateisystemen ausgibt
- `DMESG(1)` – die Manpage von `dmesg`, einem Programm zur Ausgabe und Konfiguration des Kernel-Ring-Buffers
- `UPTIME(1)` – die Manpage von `uptime`, einem Programm, das anzeigt, wie lange ein System bereits läuft
- http://www.xen.org/files/xen_user_manual.pdf – der Xen-User-Guide, das offizielle Handbuch der Xen-Entwickler
- `XM(1)` – die Manpage von `xm`, der zentralen Xen-Management-Anwendung
- `XENTOP(1)` – die Manpage von `xentop`, einem Tool zur Anzeige von Informationen über die Domain-0 sowie über angeschlossene virtuelle Maschinen in Echtzeit

13.3 Virtuelle Maschinen unter Fedora Linux überwachen

Problem

Sie wollen virtuelle Maschinen mit Fedora überwachen.

Lösung

Das von der Fedora-Distribution verwendete Programm `virt-manager` ist in der Lage, Informationen über die aktuelle CPU- und Hauptspeicher-Auslastung sowohl der Domain-0 als auch der angeschlossenen virtuellen Maschinen anzuzeigen. Hierbei erfolgt auch eine grafische Visualisierung der CPU-Last eines Gastes.

Um eine Übersicht über die Auslastung der Domain-0 und aller aktiven Domain-Us zu erhalten, rufen Sie `virt-manager` auf und wählen Sie aus der Liste der Hostplattformen den Eintrag der Domain-0, über die Sie gerne Informationen abfragen möchten (siehe Abbildung 13-2). `virt-manager` ist als Client-Server-Anwendung in der Lage, auch die Verwaltung entfernter Systeme zu übernehmen.

In der Standardkonfiguration ist dort jedoch nur der eigene PC unter der Bezeichnung `localhost` eingetragen. Wählen Sie diese Zeile aus, betätigen Sie die rechte Maustaste und wählen anschließend `CONNECT`.

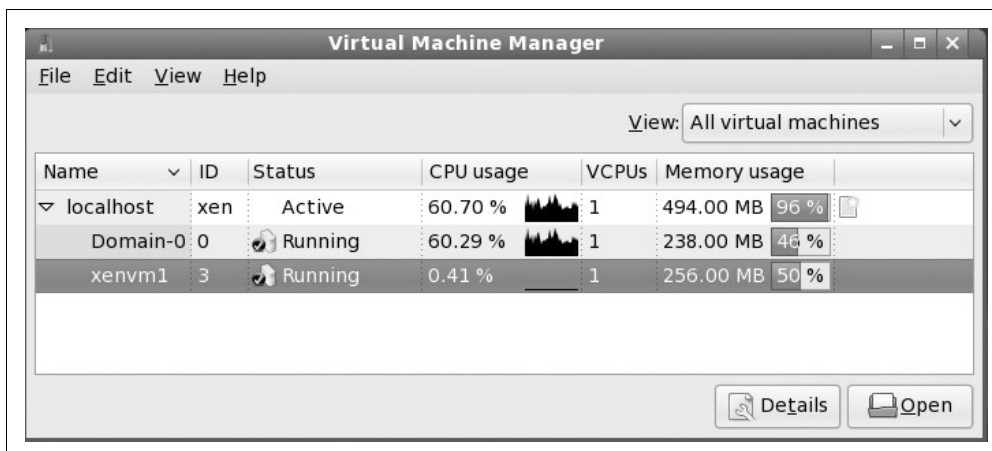


Abbildung 13-2: Exemplarische Ausgabe des Programms virt-manager

Diskussion

In der Ausgabe von `virt-manager` sind folgende Informationen verborgen: Die erste Zeile summiert die Anzahl der Ressourcen und deren aktuelle Auslastung. Dort wird also die Gesamtsituation dargestellt, bestehend aus der CPU- und Hauptspeicher-Auslastung der Domain-0 und ihrer virtuellen Maschinen.

Anschließend wird zuerst die Domain-0 alleine, gefolgt von den virtuellen Maschinen aufgelistet. Dabei gibt die Statusanzeige in der letzten Spalte an, wie viel Prozent des verfügbaren Hauptspeichers in Gebrauch sind.

Zusätzlich zum bereits erwähnten Eingangsbildschirm ist `virt-manager` in der Lage, Details über ein ausgewähltes Gastsystem anzuzeigen. Dazu muss eine Domain-U aus der Liste der aktiven Systeme ausgewählt und über die rechte Maustaste in das Menü DETAILS gebracht werden (siehe Abbildung 13-3).

Siehe auch

- http://www.techotopia.com/index.php/Managing_and_Monitoring_Fedora_based_Xen_Guest_Systems – ein Howto, das sich mit dem Management und Monitoring von Gastsystemen unter Fedora Linux beschäftigt
- VIRT-MANAGER(1) – die Manpage von `virt-manager`, einem Programm zur Verwaltung virtueller Maschinen unter Fedora Linux

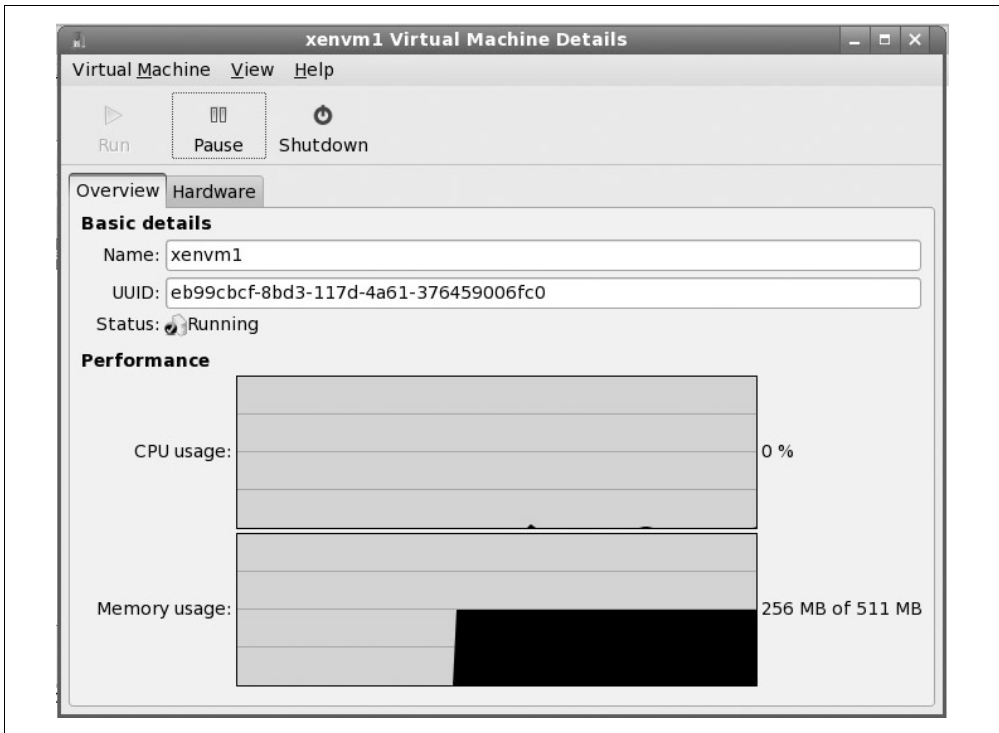


Abbildung 13-3: Anzeige von Details über ein ausgewähltes Gastssystem mittels virt-manager

Dies ist ein Auszug aus dem Buch "XEN Kochbuch", ISBN 978-3-89721-729-4
<http://www.oreilly.de/catalog/xeinckiger/>
 Dieser Auszug unterliegt dem Urheberrecht. © O'Reilly Verlag 2009

13.4 Die CPU-Auslastung mit XenStats auswerten

Problem

Sie möchten Grafiken zur Dokumentation der CPU-Auslastung virtueller Maschinen mit XenStats erstellen.

Lösung

Ein Bild sagt oft mehr als tausend Worte. Deshalb beschreiben wir im Folgenden, wie durch die beiden Skripte *xenupdate.py* und *xengraph.py* eine grafische Auswertung der CPU-Auslastung von Domain-0 sowie den aktiven Domain-Us erstellt werden kann.

Voraussetzung hierfür ist ein installierter Webserver sowie das Programmpaket RRD. Im Folgenden gehen wir davon aus, dass der Webserver bereits installiert ist.

RRD steht für Round Robin Database und ist ein Programm, das in erster Linie dafür verwendet wird, zeitabhängige Messdaten zu erfassen bzw. zu visualisieren. Für die Integration in bestehende Anwendungen stehen Sprachbindungen für die unterschiedlichsten Programmiersprachen zur Verfügung.

Die Installation von RRD gestaltet sich abhängig von der Distribution wie folgt:

Installation von RRD unter Fedora:

```
[root@postel ~]# yum install rrdtool
```

Installation von RRD unter SUSE:

```
[root@postel ~]# yast -i rrdtool
```

Installation von RRD unter Debian/Ubuntu:

```
[root@postel ~]# apt-get install rrdtool
```

Nun können wir als Nächstes *xenstats* installieren:

```
[root@postel ~]# wget http://skaya.enix.org/webs/xen/xenupdate.py
[root@postel ~]# wget http://skaya.enix.org/webs/xen/xengraph.py
[root@postel ~]# wget http://www.mathematik.uni-marburg.de/~picht/patches/xenupdate.
patch
[root@postel ~]# wget http://www.mathematik.uni-marburg.de/~picht/patches/xengraph.patch
[root@postel ~]# cat xenupdate.patch |patch -p0
[root@postel ~]# cat xengraph.patch |patch -p0
[root@postel ~]# chmod +x xen*.py
```

Nach der Vorbereitung der Installation ist jetzt im Bereich des Webservers ein Verzeichnis anzulegen, unter dem die Grafiken später angesehen werden können. Damit die URL das Format *http://domain0.xencookbook.com/xenstats/* hat, muss der Pfad je nach Distribution wie folgt gewählt werden:

Fedora:

```
[root@postel ~]# mkdir /var/www/html/xenstats
```

SUSE:

```
[root@postel ~]# mkdir /srv/www/htdocs/xenstats
```

Debian:

```
[root@postel ~]# mkdir /var/www/xenstats
```

Während im Folgenden das Skript *xenupdate.py* die Daten sammelt und in die RRD-Datenbank schreibt, wird mit dem zweiten Skript die grafische Auswertung erzeugt. Der als zweiter Parameter übergebene Pfad spezifiziert im ersten Befehl den Pfad zur RRD-Datenbank und im zweiten Fall den Ordner, in dem die Grafiken gespeichert werden:

```
[root@postel ~]# ./xenupdate.py /var/www/html/xenstats
[root@postel ~]# ./xengraph.py /var/www/html/xenstats
```

Nach einem erfolgreichen Systemtest werden die Skripte nach */usr/local/bin/* kopiert und durch einen Cron-Job in regelmäßigen Intervallen ausgeführt:

```
[root@postel ~]# cp xen*.py /usr/local/bin/
[root@postel ~]# echo -e "-*/5 * * * * \t root /usr/local/bin/xenupdate.py /var/www/html/
xenstats/" >> /etc/crontab
[root@postel ~]# echo -e "-*/5 * * * * \t root /usr/local/bin/xengraph.py /var/www/html/
xenstats/" >> /etc/crontab
```

In Abbildung 13-4 sehen Sie einen Screenshot einer exemplarischen Ausgabe von *XenStats*.

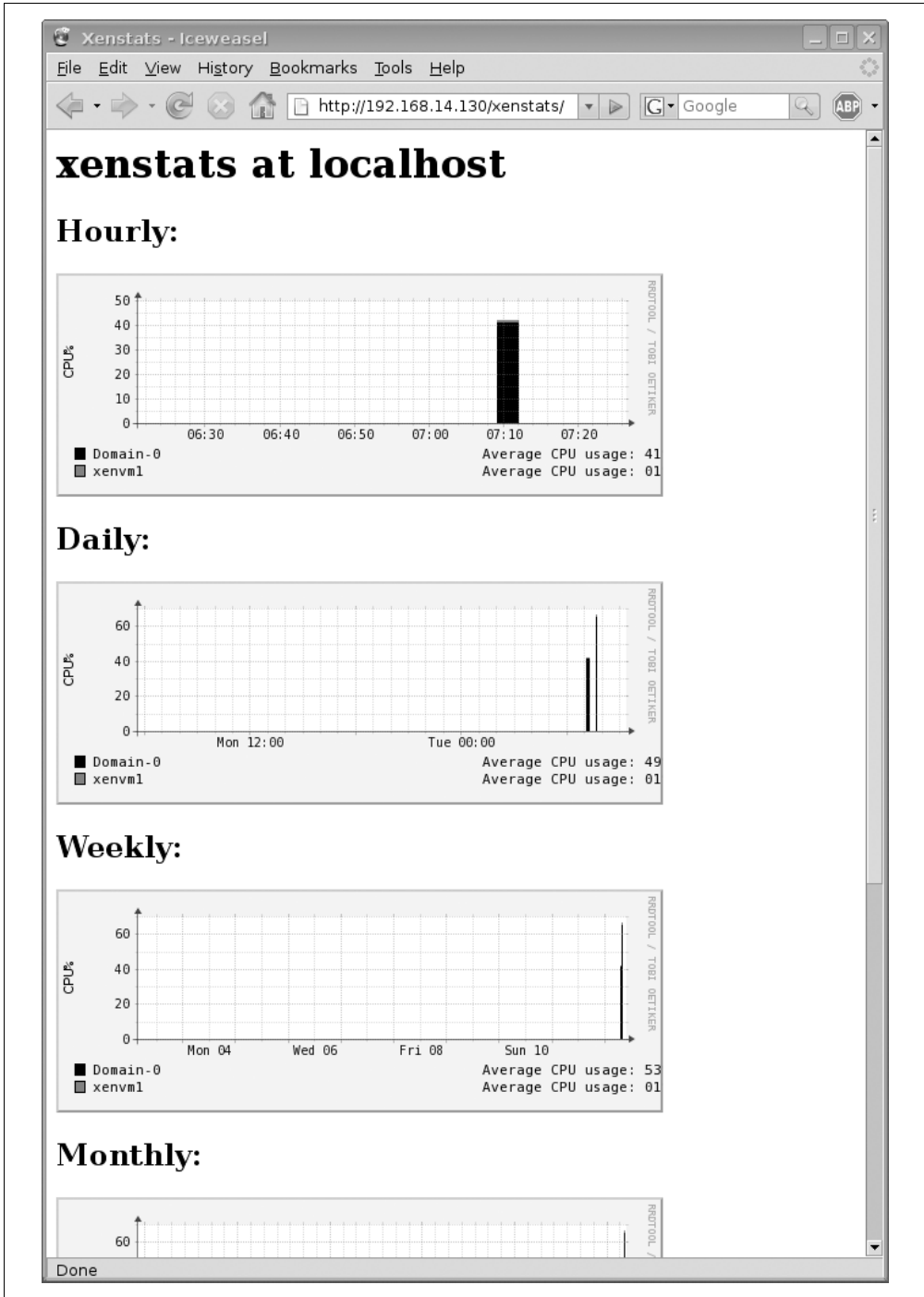


Abbildung 13-4: Eine Beispielausgabe von XenStats

Dies ist ein Auszug aus dem Buch "XEN Kochbuch", ISBN 978-3-89721-729-4
<http://www.oreilly.de/catalog/kenbkjger/>
 Dieser Auszug unterliegt dem Urheberrecht. © O'Reilly Verlag 2009

Diskussion

Nach der Installation des *rrdtool*-Pakets mit den Installationswerkzeugen der jeweiligen Distribution werden die beiden benötigten Python-Skripte mit dem Download-Manager *wget* heruntergeladen.

Da diese beiden von Jérôme Petazzoni entwickelten Skripte aus dem Jahre 2006 stammen und mit aktuellen Xen-Versionen nicht mehr einwandfrei funktionieren, hat der Autor sie entsprechend angepasst. Diese Änderungen können von der angegebenen Webseite als Patch heruntergeladen werden.

Die beiden heruntergeladenen Patch-Dateien werden mit dem vom Erfinder der Programmiersprache Perl (Larry Wall) entwickelten Programm *patch* eingespielt und verändern die im aktuellen Verzeichnis vorliegenden Dateien *xengraph.py* und *xenupdate.py*.

Im Anschluss werden die Dateien mittels *chmod* ausführbar gemacht. Nachdem ein Ordner unterhalb des Webservers angelegt wurde, kann die Software durch einen initialen Aufruf ausprobiert werden.

Tritt kein Fehler auf, werden die Dateien nach */usr/local/bin* verschoben und zur regelmäßigen Ausführung via *cron* konfiguriert. Letzteres geschieht durch eine Anpassung der Datei */etc/crontab*. Aus der weiter oben angegebenen Konfiguration geht hervor, dass die Dateien */usr/local/bin/xenupdate.py* sowie */usr/local/bin/xengraph.py* mit den Rechten des Benutzers *root* alle fünf Minuten ausgeführt werden, ohne die Ausführung in der Protokolldatei des Syslog-Daemons festzuhalten.

Eine weitere Möglichkeit zur Erstellung von Grafiken, zumindest unter Debian GNU/Linux, ist *DTC-xen*, eine Erweiterung von Domain-Technologie-Control (DTC), einer Software aus dem Bereich des Hosting-Management. Dies ist eine Webanwendung, die eigentlich zur Verwaltung virtueller Maschinen konzipiert wurde, die aber ebenfalls in der Lage ist, Informationen über die Auslastung von CPU, Speicher und Netzwerk zu sammeln und grafisch aufzubereiten.

Obwohl die Funktionalität der Erstellung von Bildern zur grafischen Auswertung nur eine Teilkomponente ist, muss die Management-Lösung dennoch komplett installiert und konfiguriert werden.

Siehe auch

- <http://skaya.enix.org/wiki/XenStats> – die Webseite des XenStats-Projekts
- <http://oss.oetiker.ch/rrdtool/> – die Homepage der RRD-Tools, einer Open Source Software zur Protokollierung und grafischen Auswertung zeitversetzter Daten
- <http://www.gplhost.com/software-dtc-xen.html> – die Webseite der Domain Technologie Control (DTC), einer Software aus dem Bereich des Hosting-Management, mit deren Hilfe unter anderem auch Monitoring-Daten für Xen erstellt werden können
- RRD-BEGINNERS(1) – eine Manpage, in der auch der »RRDtool Beginners' Guide«, eine Einführung in RRD, zu finden ist

- CRONTAB(5) – die Manpage von Crontab, einer Lösung zur zeitgesteuerten Ausführung von Linux-Kommandos
- PATCH(1) – die Manpage von patch, mit dessen Hilfe Patches für bestehenden Quellcode eingespielt werden können

13.5 Monitoring virtueller Maschinen mithilfe von Nagios – vorbereitende Schritte

Nagios ist eine Open-Source-Lösung zur Überwachung einzelner Systeme oder auch kompletter Netzwerke.

Egal, ob man lediglich gerne wissen möchte, ob in seinem lokalen »Heimnetzwerk« alle Computer fehlerfrei arbeiten, oder ob man als hauptberuflicher Administrator eine große Serverfarm zu betreuen hat – Nagios eignet sich hervorragend zur Systemüberwachung. Läuft eine bestimmte Komponente aus dem Ruder, ist sie komplett ausgefallen oder zeigt sie nur zu bestimmten Zeiten ein ungewöhnliches Verhalten? Wer ein gut gepflegtes Netz verwaltet, muss jederzeit informiert sein.

Nagios, dessen Projektwebseite unter der URL <http://www.nagios.org> zu finden ist, kann außer auf Linux auch auf einer Vielzahl weiterer Unix-Systeme betrieben werden. Durch den Einsatz des SNMP-Protokolls und bestimmter Nagios-Erweiterungen wird es sogar möglich, Windows-Server zu überwachen.

Da fast jedes Netzwerk seine eigenen Tücken und Fallstricke bietet und nicht jede Firma die gleichen Komponenten einsetzt, verfügt Nagios über eine modulare Struktur, in der die Basissoftware um Testfälle für einzelne Komponenten erweitert werden kann. Diese Testmodule werden im Nagios-Jargon als *Checks* bzw. *Plugins* bezeichnet. Neben den bereits in der Basisinstallation mitgelieferten Checks können von Webseiten wie <http://www.nagiosexchange.org> oder <http://nagiosplug.sourceforge.net/> zusätzliche Erweiterungen, unter anderem für Xen, heruntergeladen werden.

Wird von Nagios ein Plugin aufgerufen, um eine bestimmte Funktionalität zu überprüfen, so werden in der Rückgabe die folgenden Stationen ausgewertet: *OK*, *Warning*, *Critical* oder *Unknown*. Liefert ein Plugin ein negatives Ergebnis zurück, wird abhängig von der Konfiguration erst versucht, den Test zu wiederholen. Bleibt danach der Status weiterhin »rot«, hat also etwas wieder nicht funktioniert, kann eine Benachrichtigung, etwa in Form einer E-Mail oder SMS, an eine zuvor definierte Personengruppe oder funktionale User-ID geschickt werden.

Informationen über die eigenständige Entwicklung von Plugins in diversen Programmiersprachen finden Sie in den »Nagios Plugin Development Guidelines« unter <http://nagios-plug.sourceforge.net/developer-guidelines.html>.



Bei der Inbetriebnahme eines Nagios-Systems sollte man darauf achten, die Ausfallsicherheit von Nagios sicherzustellen. Es ist zwar schön, die Ausfallsicherheit von Xen-Systemen zu erhöhen, jedoch nützt einem die beste Nagios-Installation nichts, wenn die Erreichbarkeit des eigentlichen Monitoring-Tools nicht gewährleistet ist. Nähere Informationen dazu finden Sie in Kapitel 12, *Hochverfügbarkeit* bzw. in den folgenden beiden Dokumenten, die sich mit der Hochverfügbarkeit einer Nagios-Installation beschäftigen:

- http://nagios.sourceforge.net/download/contrib/documentation/misc/HighAvailability/NagiosHA_EN.pdf
- http://nagios.sourceforge.net/docs/1_0/redundancy.html

Problem

Sie möchten Ihre virtuellen Maschinen und Domain-0-Hosts durch eine zentrale Komponente in Form von Nagios überwachen und sich bei einem Problem per E-Mail, Pager oder SMS benachrichtigen lassen. Neben den meisten gängigen Serverdiensten kann Nagios auch den Xen-Daemon und innerhalb einer Domain-0 laufende Gäste überwachen.

Lösung

Zuerst muss man Nagios installieren – und zwar möglichst nicht auf den zu überwachen- den Servern. In großen Netzwerken werden hierfür dedizierte Server abgestellt, aber aufgrund der geringen Ressourcenauslastung in einem kleinen Serverpark kann Nagios auch auf dem Print- oder Mailserver »nebenher« laufen.

Vorbereitung des Nagios-Hosts

Abhängig von der eingesetzten Linux-Distribution, gestaltet sich die Installation wie folgt:

Debian: `debian:~# apt-get install nagios2 nagios-plugins nagios-nrpe-plugin`

Fedora: `[root@fedora ~]# yum install nagios nagios-plugins-nrpe nagios-plugins-ping`

SUSE: `opensuse:~ # yast -i nagios nagios-www nagios-plugins apache2 nagios-nrpe`

Bei SUSE und Fedora Linux befinden sich die Konfigurationsdateien von Nagios unterhalb von `/etc/nagios`, während Debian an dieser Stelle den Pfad `/etc/nagios2` bevorzugt. Um die Konfiguration über alle drei Distributionen synchron zu halten, legen wir einen neuen Ordner namens `xen` unterhalb des Nagios-Konfigurationsverzeichnis an.

Vorbereitung des Nagios-Hosts für Fedora

Nach dem Anlegen des Verzeichnisses für die Xen-spezifische Nagios-Konfiguration wird mithilfe des Programms `htpasswd` in der Datei `/etc/nagios/passwd` ein Passworteintrag für den Benutzer `hans` angelegt. Der an dieser Stelle hinterlegte Benutzername und die Passwortkombination werden später benötigt, um sich auf der Nagios-Weboberfläche anzumelden.

```
root@fedora ~]# mkdir -p /etc/nagios/xen ; cd /etc/nagios/xen
[root@fedora xen]# htpasswd -c /etc/nagios/passwd hans
New password:
Re-type new password:
Adding password for user hans
```

Als Nächstes wird die im Apache-Webserver hinterlegte Konfiguration für die Nagios-Webseite angepasst. Nach der Installation ist der Zugriff auf die Nagios-Weboberfläche lediglich auf den lokalen Host (127.0.0.1) beschränkt. Sofern Sie Nagios nicht auf Ihrem Desktop installiert haben, muss hier noch der entsprechende IP-Bereich (oder einzelne IP-Adressen) freigeschaltet werden:

```
[root@fedora conf.d]# vim /etc/httpd/conf.d/nagios.conf
ScriptAlias /nagios/cgi-bin/ /usr/lib/nagios/cgi-bin/
<Directory /usr/lib/nagios/cgi-bin/>
    Options ExecCGI
    order deny,allow
    deny from all
    allow from 127.0.0.1 192.168.2.0/24 192.168.94.0/24
    AuthType Basic
    AuthUserFile /etc/nagios/passwd
    AuthName "nagios"
    require valid-user
</Directory>

Alias /nagios/ /usr/share/nagios/html/
<Directory /usr/share/nagios/html/>
    Options None
    order deny,allow
    deny from all
    allow from 127.0.0.1 192.168.2.0/24 192.168.94.0/24
    AuthType Basic
    AuthUserFile /etc/nagios/passwd
    AuthName "nagios"
    require valid-user
</Directory>
```

Durch einen Neustart des Webservers werden die Änderungen aktiv. Bei dieser Gelegenheit sorgen wir durch das Kommando `chkconfig` auch gleich dafür, dass der Webserver nach einem Systemneustart automatisch gestartet wird:

```
[root@fedora conf.d]# /etc/init.d/httpd start
Starting httpd: [ OK ]
[root@fedora-nagios xen]# /sbin/chkconfig httpd on
```

Vorbereitung des Nagios-Hosts für Debian

Nach dem Anlegen des Verzeichnisses für die Xen-spezifische Nagios-Konfiguration wird mithilfe des Programms `htpasswd` in der Datei `/etc/nagios2/passwd` ein Passworteintrag für den Benutzer `hans` angelegt. Der an dieser Stelle hinterlegte Benutzername und die Passwortkombination werden später benötigt, um sich auf der Nagios-Weboberfläche anzumelden. Das Nagios-Webinterface ist später unterhalb des Pfades `/nagios2/` auf dem Webserver zu finden. In unserem Beispiel lautet die IP-Adresse `http://192.168.132.132/nagios2/`.

Als Letztes wird, um die Konfiguration bezüglich dieses Problems zwischen allen Distributionen synchron zu halten, ein Link namens `/etc/nagios` angelegt, der auf `/etc/nagios2` verweist, und es wird sichergestellt, dass der Webserver Apache nach einem Systemstart automatisch gestartet wird:

```
debian-nagios# mkdir -p /etc/nagios2/xen ; cd /etc/nagios2/xen
debian-nagios# htpasswd -c /etc/nagios2/htpasswd.users hans
debian-nagios# ln -s /etc/nagios2 /etc/nagios
debian-nagios# update-rc.d apache2 defaults
```

Vorbereitung des Nagios-Hosts für openSUSE

Nach dem Anlegen des Verzeichnisses für die Xen-spezifische Nagios-Konfiguration wird mithilfe des Programms `htpasswd` in der Datei `/etc/nagios/passwd` ein Passworteintrag für den Benutzer `hans` angelegt. Der an dieser Stelle hinterlegte Benutzername und die Passwortkombination werden später benötigt, um sich auf der Nagios-Weboberfläche anzumelden.

```
suse-nagios:/etc # mkdir -p /etc/nagios/xen ;
suse-nagios:/etc # htpasswd2 -c /etc/nagios/passwd hans
New password:
Re-type new password:
Adding password for user hans
```

Als Nächstes wird die im Apache-Webserver hinterlegte Konfiguration für die Nagios-Webseite angepasst. Im Vergleich zu anderen Distributionen kann in der Standardkonfiguration unter openSUSE jeder Nutzer auf die Nagios-Oberfläche zugreifen. Durch die folgende Anpassung wird der Zugriff auf autorisierte Benutzer beschränkt:

```
suse-nagios:/etc/nagios # cat /etc/apache2/conf.d/nagios.conf
ScriptAlias /nagios/cgi-bin /usr/lib/nagios/cgi
<Directory /usr/lib/nagios/cgi>
    Options ExecCGI
    order deny,allow
    deny from all
    allow from 192.168.2.0/24 192.168.94.0/24
    AuthType Basic
    AuthUserFile /etc/nagios/passwd
    AuthName "nagios"
    require valid-user
</Directory>
```

```

Alias /nagios /usr/share/nagios
<Directory /usr/share/nagios>
  Options None
  order deny,allow
  deny from all
  allow from 192.168.2.0/24 192.168.94.0/24
  AuthType Basic
  AuthUserFile /etc/nagios/passwd
  AuthName "nagios"
  require valid-user
</Directory>

```

Damit die Änderungen aktiv werden, muss der Webserver neu gestartet werden. Mithilfe von `insserv` wird sichergestellt, dass der Webserver zukünftig automatisch gestartet wird:

```

suse-nagios:/etc # rcapache2 restart
suse-nagios:/etc # insserv apache2
suse-nagios:/etc # rcnagios start && insserv nagios

```

Fortsetzung der Nagios-Hosts-Konfiguration

Nun ist es notwendig, dem soeben angelegten Benutzer Zugriff auf bestimmte Nagios-Ressourcen zu geben. Details zu den einzelnen Variablen finden Sie in der anschließenden Diskussion.

```

authorized_for_system_information=hans
authorized_for_configuration_information=hans
authorized_for_system_commands=hans
authorized_for_all_services=hans
authorized_for_all_hosts=hans
authorized_for_all_service_commands=hans
authorized_for_all_host_commands=hans

```

Die obigen Zeilen müssen in der Nagios-Konfigurationsdatei `cgi.cfg` hinterlegt werden. Unter Debian befindet sich diese unterhalb von `/etc/nagios2`, während Fedora und SUSE stattdessen `/etc/nagios` verwenden. Außerdem sind folgende Dateien notwendig:

```

root@ritchie:~# cd /etc/nagios/xen && ls -la
total 44
drwxr-xr-x 2 root root 4096 2008-05-22 17:50 .
drwxr-xr-x 5 root root 4096 2008-05-21 02:24 ..
-rw-r--r-- 1 root root 870 2008-05-22 17:10 commands.cfg
-rw-r--r-- 1 root root 1149 2008-05-20 00:30 xenhosts.cfg
-rw-r--r-- 1 root root 4124 2008-05-22 17:06 xenservices.cfg

```

Die erste Konfigurationsdatei `/etc/nagios/xen/commands.cfg` legt fest, welche »Kommandos« Nagios kennt und wie diese zu verwenden sind. Im Folgenden sehen wir drei definierte Befehle. Der erste überprüft, ob auf einem angegebenen Host ein SSH-Server erreichbar ist. Im vorliegenden Fall wird dabei direkt versucht, eine ssh-Verbindung vom Nagios-Host zum entsprechenden Client aufzubauen. Die folgenden beiden Einträge benötigen wir für den Nagios Remote Plug-in Executor (NRPE). Mit seiner Hilfe werden Befehle auf den zu überwachenden Servern aufgerufen. Er liegt in zwei Varianten vor, da

wir im Folgenden zwei unterschiedliche Arten von über NRPE ausgeführten Tests unterscheiden müssen: jene, die ein Kommandozeilenargument benötigen, und diejenigen, die direkt über ihren Namen identifiziert werden können.

```

root@ritchie:/etc/nagios/xen# cat commands.cfg
define command {
    command_name    check_ssh
    command_line    /usr/lib/nagios/plugins/check_ssh -t 120 -H $HOSTADDRESS$
}

# this command runs a program $ARG1$ with arguments $ARG2$
define command {
    command_name    check_nrpe_2arg
    command_line    /usr/lib/nagios/plugins/check_nrpe -t 120 -H $HOSTADDRESS$ -
c $ARG1$ -a $ARG2$
}

# this command runs a program $ARG1$ with no arguments
define command {
    command_name    check_nrpe_1arg
    command_line    /usr/lib/nagios/plugins/check_nrpe -t 120 -H $HOSTADDRESS$ -c
$ARG1$
}
  
```

Die Datei `/etc/nagios/xen/xenhosts.cfg` enthält eine Liste der zu überwachenden Server. In diesem Beispiel handelt es sich dabei um vier Systeme, von denen drei einer Domain-0 entsprechen, das vierte einer virtuellen Maschine. Am Ende der Datei wird ein Template namens `xen-server` spezifiziert, das von den konfigurierten Hosts `fedora-dom0`, `debian-dom0`, `suse-dom0` und `debian-domU` verwendet wird:

```

root@ritchie:/etc/nagios/xen# cat xenhosts.cfg
define host{
    host_name        fedora-dom0
    alias            fedora-dom0.xencookbook.com
    address          192.168.94.129
    use              xen-server
}

define host{
    host_name        debian-dom0
    alias            debian-dom0.xencookbook.com
    address          192.168.94.133
    use              xen-server
}

define host{
    host_name        suse-dom0
    alias            suse-dom0.xencookbook.com
    address          192.168.94.130
    use              xen-server
}
  
```

```

define host{
    host_name          debian-domU
    alias              debian-domU.xencookbook.com
    address            192.168.94.135
    use                xen-server
}

#####

define host{
    name              xen-server
    use               generic-host
    check_period      24x7
    max_check_attempts 10
    check_command     check-host-alive
    notification_period 24x7
    notification_interval 120
    notification_options d,u,r
    contact_groups    admins
    register          0
}
  
```

Unterhalb von `/etc/nagios/xen/xenservices.cfg` befindet sich die eigentliche Konfiguration der zu überwachenden Systeme. In diesem Beispiel sind sie nach Servern sortiert. Für den Server `fedora-dom0.xencookbook.com` werden drei Dinge überprüft:

1. Läuft auf diesem Server der Xen-Daemon `xend`?
2. Läuft auf diesem Server eine virtuelle Maschine namens `fedora-domU1`?
3. Ist das System per `ping` erreichbar?

Nach einer vergleichbaren Konfiguration für eine exemplarische Debian- und SUSE-Maschine wird an späterer Stelle noch eine virtuelle Maschine namens `debian-domU` detaillierter überwacht. Bei ihr wird überprüft, ob der `ssh`- und der `http`-Dienst erreichbar sind und ob auf der `root`-Partition noch mehr als 10 bis 20 Prozent Speicher zur Verfügung stehen. Am Ende der Konfigurationsdatei befindet sich dann noch die Beschreibung eines Server-Templates. Dies ist nicht zwingend erforderlich, wird jedoch an dieser Stelle verwendet, um sicherzustellen, dass dieses Beispiel ohne weitere Anpassungen sofort auf Debian, Fedora und SUSE einsetzbar ist. Weiterhin wird ein Kontakt definiert und einer Gruppe hinzugefügt, die im Falle eines Problems per E-Mail informiert werden soll. Hierfür muss zudem sichergestellt sein, dass der Nagios-Server in der Lage ist, E-Mail zu versenden. Die hier verwendete E-Mail-Adresse `hans@xencookbook.com` müssen Sie natürlich entsprechend anpassen.

Im Folgenden sehen Sie eine exemplarische Konfigurationsdatei `/etc/nagios/xen/xenservices.cfg`:

```

define service {
    host_name          fedora-dom0
    service_description Xen Virtual Machine Monitor Fedora Domain-0
    use                xen-service
}
  
```

```

        check_command          check_nrpe_1arg!check_xen_dom0
        notification_interval  0
    }
    define service {
        host_name              fedora-dom0
        service_description    Xen Virtual Machine Monitor Fedora Domain-U
        use                    xen-service
        check_command          check_nrpe_2arg!check_xen_domU!fedora-domU1
        notification_interval  0
    }
    define service {
        host_name              fedora-dom0
        service_description    Xen Virtual Machine Monitor Fedora Domain-0 ping
        check_command          check_ping!100.0,20%!500.0,60%
        use                    xen-service
        notification_interval  0
    }
    #####
    define service {
        host_name              debian-dom0
        service_description    Xen Virtual Machine Monitor Debian Domain-0
        use                    xen-service
        check_command          check_nrpe_1arg!check_xen_dom0
        notification_interval  0
    }
    define service {
        host_name              debian-dom0
        service_description    Xen Virtual Machine Monitor Debian Domain-U
        use                    xen-service
        check_command          check_nrpe_2arg!check_xen_domU!debian-domU
        notification_interval  0
    }
    define service {
        host_name              debian-dom0
        service_description    Xen Virtual Machine Monitor Debian Domain-0 ping
        check_command          check_ping!100.0,20%!500.0,60%
        use                    xen-service
        notification_interval  0
    }
    #####
    define service {
        host_name              suse-dom0
        service_description    Xen Virtual Machine Monitor SUSE Domain-0
        use                    xen-service
        check_command          check_nrpe_1arg!check_xen_dom0
        notification_interval  0
    }
    define service {
        host_name              suse-dom0
        service_description    Xen Virtual Machine Monitor SUSE Domain-0 ping
        check_command          check_ping!100.0,20%!500.0,60%
        use                    xen-service
        notification_interval  0
    }
}

```

```
#####
define service {
    host_name                debian-domU
    service_description      Xen Virtual Machine Monitor Debian Domain-U ssh
    check_command            check_ssh
    use                      xen-service
    notification_interval    0
}
define service {
    host_name                debian-domU
    service_description      Xen Virtual Machine Monitor Debian Domain-U http
    check_command            check_http
    use                      xen-service
    notification_interval    0
}
define service {
    host_name                debian-domU
    service_description      Xen Virtual Machine Monitor Debian Domain-U disk
    check_command            check_nrpe!check_root_partition
    use                      xen-service
    notification_interval    0
}
#####

define contact{
    contact_name            xen-admin
    alias                   Nagios Admin
    service_notification_period 24x7
    host_notification_period 24x7
    service_notification_options w,u,c,r
    host_notification_options d,r
    service_notification_commands notify-by-email
    host_notification_commands host-notify-by-email
    email                   hans@xencookbook.com
}
define host{
    name                    xen-server
    use                     generic-host
    check_period            24x7
    max_check_attempts      10
    check_command            check-host-alive
    notification_period     24x7
    notification_interval   120
    notification_options    d,u,r
    contact_groups          xen-admins
    register                 0
}
define contactgroup{
    contactgroup_name      xen-admins
    alias                   Administrators
    members                 xen-admin
}

```

Damit sich Nagios der neu hinzugewonnenen Konfigurationsdateien unterhalb von */etc/nagios/xen* bewusst wird, muss die zentrale Nagios-Konfigurationsdatei *nagios.cfg* hierüber informiert werden. Der folgende Eintrag sorgt dafür, dass Nagios alle in dem angegebenen Verzeichnis enthaltenen Dateien zu seiner Konfiguration dazuzählt:

```
root@ritchie:~# echo "cfg_dir=/etc/nagios/xen" >> /etc/nagios/nagios.cfg
```

Unter Debian müssen noch zwei Dateien entfernt werden, da diese Überschneidungen mit unserem vorliegenden Beispiel-Setup aufweisen:

```
debian-nagios # rm /etc/nagios-plugins/config/check_nrpe.cfg
debian-nagios # rm /etc/nagios-plugins/config/ssh.cfg
```

Ab jetzt kann Nagios gestartet werden, wie bei der jeweiligen Distribution üblich:

```
[root@fedora nagios]# /etc/init.d/nagios start
debian-nagios:/etc/nagios/xen# /etc/init.d/nagios2 restart
suse-nagios:/etc/nagios/xen # rcnagios restart
```

Folgende Befehle stellen obendrein sicher, dass Nagios in Zukunft automatisch gestartet wird:

```
[root@fedora ~]# /sbin/chkconfig nagios on
suse-nagios:/etc/nagios/xen # chkconfig nagios on
debian-nagios:/etc/nagios2/xen # update-rc.d nagios2 defaults
```

Diskussion

Zurzeit existiert eine Vielzahl an quelloffenen Lösungen zur Überwachung und Sicherung von Systemen oder ganzen Netzwerken. Zu den populärsten Monitoring-Werkzeugen zählt zweifellos Nagios (<http://www.nagios.org>). In diesem Abschnitt gehen wir lediglich auf die Xen-Spezifika ein. Für eine allgemeine Einführung zum Thema Nagios sei das Buch *Nagios* von Wolfgang Barth empfohlen. Bei dem vorliegenden Setup handelt es sich also um eine minimalistische Lösung, mit der virtuelle Xen-Systeme überwacht werden können und die das Potenzial hat, zu einer ausgereiften Nagios-Installation ausgebaut zu werden.

Nagios-Authentifizierung

Über die Konfigurationsdatei */etc/nagios/cgi.cfg* (bzw. bei Debian GNU/Linux und darauf aufbauenden Derivaten mithilfe von */etc/nagios2/cgi.cfg*) legt Nagios verschiedenste Einstellungen rund um die Nagios-Weboberfläche fest. Ein wichtiger Punkt hierbei ist die Authentifizierung. Direkt nach der Installation wird bei allen in diesem Buch primär besprochenen Linux-Distributionen der Zugriff per »Authentifizierung« (use_authentication=1) verwendet. Aus Sicherheitsgründen ist es keine gute Idee, dieses Sicherheitsmerkmal zu deaktivieren. Im Kommentar der Beispielkonfigurationsdatei schreiben die Nagios-Entwickler hierzu Folgendes:

```
# NOTE: It is a really *bad* idea to disable authorization, unless
# you plan on removing the command CGI (cmd.cgi)! Failure to do
# so will leave you wide open to kiddies messing with Nagios and
```

```
# possibly hitting you with a denial of service attack by filling up
# your drive by continuously writing to your command file!
```

Tabelle 13-7 gibt Aufschluss über die Verwendung der wichtigsten Authentifizierungs-Variablen der Konfigurationsdatei *cgi.cfg*.

Tabelle 13-7: Authentifizierungs-Variablen der Nagios-Konfigurationsdatei *cgi.cfg*

Variable	Bedeutung
<code>authorized_for_system_information</code>	Kommaseparierte Liste von Benutzernamen, die Zugriff auf die Nagios-Prozessinformationen erhalten.
<code>authorized_for_configuration_information</code>	Kommaseparierte Liste von Benutzernamen, die alle Konfigurationsoptionen (Host, Kommandos, usw.) einsehen können.
<code>authorized_for_system_commands</code>	Diese Option erwartet ebenfalls eine durch Kommas getrennte Liste von Benutzernamen, denen erlaubt wird, über die Nagios-Weboberfläche die Kommandos <code>shutdown</code> , <code>restart</code> und <code>standby</code> auszuführen. Mithilfe dieser Befehle ist es möglich, Nagios über die Weboberfläche zu beenden, neu zu starten oder zu pausieren.
<code>authorized_for_all_services</code> <code>authorized_for_all_hosts</code>	Diese beiden Variablen enthalten – ebenso wie alle vorausgegangenen – eine Benutzerliste, die durch Kommata getrennt ist. Die vorliegenden Optionen regeln den Zugriff auf Informationen für alle Services und Hosts, die von Nagios überwacht werden. Normalerweise können Benutzer nur Informationen der Hosts bzw. Services einsehen, bei denen sie in der Konfiguration als Kontakt hinterlegt sind. An dieser Stelle besteht zusätzlich die Möglichkeit, dem Wert der Variablen ein * zuzuweisen. Damit erhalten <i>alle</i> über den Webserver authentifizierten Benutzer Zugriff.
<code>authorized_for_all_service_commands</code> <code>authorized_for_all_host_commands</code>	Diese beiden Optionen, die eine durch Kommata getrennte Aufzählung von Benutzernamen enthalten können, legen fest, welchen Benutzern der Zugriff auf Informationen über alle Hosts bzw. Services vergönnt ist. In der Standardeinstellung kann ein Benutzer nur Informationen der Hosts bzw. Services einsehen, bei denen er als Kontakt hinterlegt ist. An dieser Stelle besteht zusätzlich die Möglichkeit, dem Wert der Variablen ein * zuzuweisen. Damit erhalten <i>alle</i> über den Webserver authentifizierten Benutzer Zugriff.

Nagios-Basiskonfiguration

Im Folgenden gehen wir anhand ausgewählter Anweisungen näher auf die einzelnen Nagios-Konfigurationsanweisungen ein. Exemplarisch sehen wir an dieser Stelle eine Kommandodefinition:

```
define command {
    command_name    check_nrpe_2arg
    command_line    /usr/lib/nagios/plugins/check_nrpe -t 120 -H $HOSTADDRESS$ -c
$ARG1$ -a $ARG2$
}
```

Es wird ein Befehl namens `check_nrpe_2arg` festgelegt, der mithilfe von NRPE Befehle auf dem zu überwachenden Knoten ausführt. Dazu werden `check_nrpe` verschiedene Argumente übergeben. Während `-t` einen Timeout in Sekunden spezifiziert, wird durch den Parameter `-H` die IP-Adresse des zu überwachenden Systems angegeben. Darüber hinaus existieren zwei Parameter. Der erste legt fest, was ausgeführt wird (`$ARG1$`), und nach `-a` wird ein zweiter Parameter (`$ARGS2`) angegeben. Dieser sorgt dafür, dass bei einem Aufruf von `check_xen_domU!debian_domU`, wie wir ihn weiter oben in der Konfiguration finden, die Kontrollroutine `check_xen_domU` über NRPE mit dem Namen einer virtuellen Maschine als Parameter gestartet wird. Im Hintergrund führt Nagios also etwa folgenden Befehl aus:

```
/usr/lib/nagios/plugins/check_nrpe -t 120 -H 192.168.94.133 -c check_xen_domU -a debian-domU
```

Die Definition einzelner Befehle wird später mit bestimmten Host-Objekten verbunden. Ein solches Host-Objekt kann wie folgt aussehen:

```
define host{
    host_name          fedora-dom0
    alias              fedora-dom0.xencookbook.com
    address            192.168.94.129
    use                xen-server
}
```

Hier sehen wir einen Eintrag für ein System namens `fedora-dom0` mit einer bestimmten IP-Adresse und einem DNS-Namen. Letzterer dient lediglich als Beschreibung zum Host oder zur Vergabe eines längeren Namens, der benutzt wird, um den Eintrag später wieder identifizieren zu können. Zum Schluss ist in der Zeile `use` noch der Name eines Templates angegeben, das an dieser Stelle zum Einsatz kommt.

```
define host{
    name              xen-server
    use              generic-host
    check_period      24x7
    max_check_attempts 10
    check_command     check-host-alive
    notification_period 24x7
    notification_interval 120
    notification_options d,u,r
    contact_groups    xen-admins
    register          0
}
```

Das Template legt ein Host-Objekt namens `xen-server` mit einer Reihe von Eigenschaften an, von denen es bereits einige von einer bereits existierenden Vorlage namens `generic-host` erbt. Hosts vom Typ `xen-server` zeichnen sich dadurch aus, dass sie rund um die Uhr überwacht werden.

Darauf folgt die Direktive `max_check_attempts`, die festlegt, wie oft Nagios versucht, ein bestimmtes Check-Kommando abzusetzen, nachdem ein anderes Ergebnis als OK zurückgegeben wurde.

Mithilfe von `check_command` wird ein Kommando angegeben, mit dem die Erreichbarkeit eines bestimmten Hosts überprüft werden soll.

Die Variable `notification_period` legt einen Zeitraum fest, nach dem der Administrator oder ein Team von Administratoren informiert werden soll. Fällt ein System aus, kommt aber innerhalb der in dieser Variable angegebenen Zeit in Sekunden wieder zurück in einen erreichbaren Zustand, wird keine Benachrichtigung gesendet.

Der darauf folgende Eintrag `notification_options` legt fest, in welchen Situationen Benachrichtigungen über einen Host verschickt werden sollen. Die Parameter werden in Form von einzelnen Zeichen angegeben, die auch miteinander kombiniert werden können und folgende Bedeutung haben:

- d = Verschicke eine Benachrichtigung, wenn sich das System im Status »down« befindet.
- u = Verschicke eine Benachrichtigung, wenn sich das System im Status »unreachable« befindet.
- r = Verschicke eine Benachrichtigung, wenn das System wieder erreichbar ist (»recovery«).
- f = Verschicke eine Benachrichtigung, wenn sich das System im Status »flapping« befindet.



Im Nagios-Jargon versteht man unter »Flapping« die Tatsache, dass ein System seinen Zustand permanent ändert. Dies kann beispielsweise so aussehen, dass es eine Minute erreichbar ist, für eine Minute ausfällt und dann wieder erreichbar wird. Weitere Details hierzu finden Sie unter http://nagios.sourceforge.net/docs/2_0/flapping.html.

Obendrein kann an dieser Stelle die Option `n` (für »none«) angegeben werden, dann werden keine Benachrichtigungen verschickt.

Mithilfe von `register` kann Nagios unterscheiden, ob es sich bei dem vorliegenden Host-Eintrag um eine Vorlage oder einen tatsächlich zu überwachenden Rechner handelt.

`contact_groups` enthält den Namen einer Kontaktgruppe oder mehrere durch Kommata getrennte Gruppen. In Kontaktgruppen sind einzelne Administratoren zusammengefasst, die bei einem Problem informiert werden sollen.

```
define contactgroup{
    contactgroup_name    xen-admins
    alias                 Administrators
    members              xen-admin
}
```

Die obige Kontaktgruppe mit dem Namen `xen-admins` besteht aus einem Mitglied namens `xen-admin`. Bei `alias` kann ein mehr oder weniger aussagekräftiger Kommentar hinterlegt werden. Ein `contact`-Objekt, das einen Administrator repräsentiert, könnte beispielsweise folgendes Format haben:

```

define contact{
    contact_name          xen-admin
    alias                 Nagios Admin
    service_notification_period 24x7
    host_notification_period 24x7
    service_notification_options w,u,c,r
    host_notification_options d,r
    service_notification_commands notify-by-email
    host_notification_commands host-notify-by-email
    email                hans@xencookbook.com
}

```

Über den Eintrag hinter `contact_name` wird das vorhandene `contact`-Objekt identifiziert. `alias` enthält wie bereits zuvor eine Beschreibung. Danach folgt mit `service_notification_period` ein Eintrag, der festlegt, zu welcher Zeit der angegebene Administrator erreichbar ist und Serviceberichte erhalten möchte.

Möchte man davon absehen, seinen Administrator nachts zu wecken, kann die Vorgabe eines festen Zeitfensters wie folgt aussehen:

```

define timeperiod{
    timeperiod_name workhours
    alias           "Normal" Working Hours
    monday         09:00-17:00
    tuesday        09:00-17:00
    wednesday     09:00-17:00
    thursday      09:00-17:00
    friday         09:00-17:00
}

```

Damit der Administrator nur in der Zeit von 9 bis 17 Uhr informiert wird, wird der Variable `service_notification_period` der Wert `workhours` übergeben.

Analog zur `service_notification_period` kann der Variable `host_notification_period` ebenfalls ein definiertes Zeitfenster oder `24x7` übergeben werden. Dadurch wird festgelegt, in welchem Zeitraum der Administrator über Probleme oder die Rückkehr einer bestimmten Komponente zum Normalbetrieb informiert werden soll.

`host_notification_options` legt fest, über welche Zustände der Administrator informiert werden soll. Dabei wird wie im vorangegangenen Beispiel zwischen `d` (»down«), `u` (»unreachable«), `r` (»recovery«) sowie `f` (»flapping«) unterschieden.

Über `service_notification_commands` wird festgelegt, wie ein Administrator über Service-Probleme informiert werden soll, während sich `host_notification_commands` um die Benachrichtigung über die Nichterreichbarkeit eines ganzen Hosts kümmert.

In der Variable `email` wird die E-Mail-Adresse des Administrators hinterlegt.

Nagios-Servicekonfiguration

Kommen wir nun zur eigentlichen Konfiguration einzelner Dienste (Services), die überwacht werden sollen. Auch hierfür wurde ein Template angelegt. Dieses unterscheidet

sich kaum von einer generischen Nagios-Vorlage, wurde aber auch an dieser Stelle wieder benötigt, um einen Betrieb derselben Konfiguration unter Debian, Fedora und SUSE zu gewährleisten.

```
define service{
    name                xen-service
    use                 generic-service
    check_period        24x7
    max_check_attempts 10
    normal_check_interval 5
    retry_check_interval 1
    contact_groups      xen-admins
    notification_options w,u,c,r
    notification_interval 60
    notification_period 24x7
    register            0
}
```

Nach der Vergabe eines eindeutigen Namens und einer vorgefertigten Nagios-Definition (use), die benutzt werden soll, wird das Zeitintervall angegeben, in dem ein zukünftiger Dienst vom Typ `xen-service` überwacht werden soll. Dabei soll zehnmal versucht werden, das Check-Kommando abzusetzen, nachdem ein vorangegangener Test ein Ergebnis zurückgeliefert hat, das nicht dem Status »OK« entspricht. Jeweils im Abstand von einer Minute soll dann versucht werden, den Check erneut auszuführen (`retry_check_interval`). Über einen Ausfall werden Mitglieder der Gruppe `xen-admins` unterrichtet, und zwar beim Wechsel in einen der folgenden Zustände: »warning«, »unknown«, »critical« oder »recovery«. Dabei werden die Administratoren jede Stunde (alle 60 Minuten) erneut über das Problem informiert, und zwar rund um die Uhr. Zuletzt wird mithilfe der Variable `register` festgelegt, dass es sich hierbei um eine Vorlage und keinen konkreten zu überwachenden Dienst handelt.

Im Folgenden sehen wir vier exemplarische Services, die von Nagios in regelmäßigen Abständen überwacht werden sollen. Während der erste prüft, ob in einer angegebenen Domain-0 der Xen-Daemon (`xend`) aktiv ist, prüft der zweite, ob kurzzeitig eine virtuelle Maschine namens `fedora-domU1` ausgeführt wird. Weiterhin wird überprüft, ob eine Domain-0 per `ping` erreichbar ist und ob in einer virtuellen Maschine der Webserver auf Port 80 läuft.

```
define service {
    host_name                fedora-dom0
    service_description      Xen Virtual Machine Monitor Fedora Domain-0
    use                     xen-service
    check_command            check_nrpe_1arg!check_xen_dom0
    notification_interval    0
}
```

Die einzelnen Zeilen des obigen Eintrags haben folgende Bedeutung: An der Stelle `host_name` wird auf einen Eintrag vom Typ `Host` mit dem Namen `fedora-dom0` verwiesen. Dieser wurde in der vorangegangenen Konfiguration bereits erstellt und sieht folgendermaßen aus:

```

define host{
    host_name          fedora-dom0
    alias              fedora-dom0.xencookbook.com
    address            192.168.94.129
    use                xen-server
}

```

Über diesen Eintrag wird der Name `fedora-dom0` der IP-Adresse `192.168.94.129` zugeordnet. Während das Feld `service_description` lediglich beschreibenden Charakter hat und für Kommentare gut ist, wird in der mit `use` beginnenden Zeile ein Verweis zum Service-Template hergestellt. Dann wird das Kommando konfiguriert, mit dem überprüft wird, ob der Server `fedora-dom0` zurzeit einen Xen-Daemon ausführt. Dazu ruft Nagios im Hintergrund den folgenden Befehl auf:

```
/usr/lib/nagios/plugins/check_nrpe -H 192.168.94.129 -c check_xen_dom0
```

Dieser baut eine Verbindung zum NRPE-Server an der angegebenen IP-Adresse auf und führt, sofern das System erreichbar und NRPE dort erfolgreich konfiguriert ist, einen Check namens `check_xen_dom0` durch.

An letzter Stelle befindet sich die Variable `notification_interval`, der der Wert `0` zugeordnet ist. `notification_interval` legt fest, wie oft ein Administrator über den Ausfall eines Services informiert werden soll. Ist der Wert dieser Variable `0`, wird der Administrator lediglich einmal und nicht in regelmäßigen Abständen über das Problem informiert.

```

define service {
    host_name          fedora-dom0
    service_description Xen Virtual Machine Monitor Fedora Domain-U
    use                xen-service
    check_command      check_nrpe_2arg!check_xen_domU!fedora-domU1
    notification_interval 0
}

```

Der obige Service unterscheidet sich (abgesehen von der Beschreibung und dem Check-Kommando) kaum vom letzten Beispiel. Dabei wird jedoch ein anderer Befehl von Nagios ausgeführt, der wie folgt aussieht:

```
/usr/lib/nagios/plugins/check_nrpe -H 192.168.94.129 -c check_xen_domU -a fedora-domU1
```

Auch hier kommt NRPE zum Einsatz, um auf dem Server mit der IP-Adresse `192.168.94.129` einen Check namens `check_xen_domU` mit dem Parameter `fedora-domU1` aufzurufen. Dieser Test überprüft, ob momentan eine virtuelle Maschine namens `fedora-domU1` aktiv ist.

```

define service {
    host_name          fedora-dom0
    service_description Xen Virtual Machine Monitor Fedora Domain-0 ping
    check_command      check_ping!100.0,20%!500.0,60%
    use                xen-service
    notification_interval 0
}

```

Ein weiterer Test ist der auf die Erreichbarkeit von `fedora-dom0.xencookbook.com`. Dazu wird das Nagios-Plugin `check_ping` verwendet. Die durch Ausrufezeichen getrennten Parameter haben dabei folgende Bedeutung: Wenn die Antwortzeit eines ICMP-Paketes über 100 Millisekunden beträgt und über 20 Prozent der Pakete verloren gehen, wird der Dienst in den Status »warning« versetzt. Bei einer Antwortzeit von mehr als 500 Millisekunden und einem Paketverlust von 60 Prozent findet ein Übergang in den Status »critical« statt.

```
define service {
    host_name                debian-domU
    service_description      Xen Virtual Machine Monitor Debian Domain-U http
    check_command            check_http
    use                      xen-service
    notification_interval    0
}
```

Die Überprüfung, ob ein Webserver auf Port 80 erreichbar ist, wird durch das Nagios-Plugin `check_http` abgedeckt, das ohne Argumente gestartet werden kann.

Siehe auch

- <http://www.novell.com/communities/node/2640/xen-virtual-machine-monitor-plugin-nagios> – die Beschreibung eines Nagios-Plugins und dessen Integration in eine SUSE-Linux-Installation
- <http://www.novell.com/coololutions/feature/19490.html> – ein Howto zur Überwachung einer Xen-Installation durch Nagios
- <http://beta.perseverantia.com/devel/?project=nagiosplug> – die Webseite der Entwickler eines Nagios-Plugins zum Monitoring von Xen
- <http://www.rrze.uni-erlangen.de/dienste/arbeiten-rechnen/linux/projekte/nagios.shtml> – eine exemplarische dokumentierte Nagios-Konfigurationsdatei des Regionalen Rechenzentrums Erlangen (RRZE)
- Wolfgang Barth: *Nagios – System- und Netzwerk-Monitoring*, Open Source Press (2005)
- <http://www.heise.de/netze/Netz-Controllerti-/artikel/81238/5> – ein einführender Artikel zum Netzwerk-Monitoring mithilfe von Nagios
- <http://www.nagiosexchange.org> – eine Plattform zum Austausch von Nagios-Plugins, einer Ansammlung von Nagios-Erweiterungen zur Überwachung spezifischer Komponenten
- <http://www.nagios.org> – die Webseite des Nagios-Projekts

13.6 Monitoring von Domain-0s und ihren virtuellen Maschinen mit Nagios

Problem

Sie haben im vorangegangenen Rezept die Grundlagen einer Nagios-Installation geschaffen und möchten nun die notwendigen Anpassungen vornehmen, um eine Domain-0 beziehungsweise eine Domain-U zu überwachen.

Lösung

Anpassung der zu überwachenden Domain-0

Nagios bringt einen eigenen Mechanismus mit, um entfernte Server zu überwachen – den Nagios Remote Plug-in Executor (NRPE). Dabei handelt es sich um einen Daemon, der auf Anfrage des Nagios-Servers ein lokales Plugin ausführt und das Ergebnis an den Nagios-Monitoring-Host zurückgibt.

Zunächst müssen auf dem zu überwachenden Host NRPE und die benötigten Plugins installiert werden. Hierfür stehen je nach verwendeter Linux-Distribution folgende Mechanismen zur Verfügung:

Debian:

```
debian:~# apt-get install nagios-nrpe-server lsof sudo
```

Fedora:

```
[root@postel ~]# yum install nrpe lsof sudo
```

SUSE:

```
linux-kpej:~ # yast -i nagios-nrpe lsof sudo
```

Als Nächstes wird das Nagios-Plugin für Xen (`check_xen`) heruntergeladen und an der dafür vorgesehenen Stelle gespeichert:

```
root@ritchie:~# mkdir -p /usr/lib/nagios/plugins
root@ritchie:~# cd /usr/lib/nagios/plugins
root@ritchie:/usr/lib/nagios/plugins# wget "http://www.nagiosexchange.org/cgi-bin/jump.cgi?ID=2272&view=File1;d=1" -O check_xen
root@ritchie:/usr/lib/nagios/plugins# chmod +x check_xen
```

Für Fedora verwenden Sie bitte diese wget-Kommandozeile:

```
root@ritchie:~# wget "http://fedorapeople.org/~hans/check_xen_fedora" -O check_xen
```

Danach wird NRPE über die Konfigurationsdatei `/etc/nagios/nrpe.cfg` konfiguriert. Eine exemplarische minimale Konfigurationsdatei ohne Kommentare kann für Debian wie folgt aussehen:

```
pid_file=/var/run/nrpe.pid
server_port=5666
nrpe_user=nagios
nrpe_group=nagios
allowed_hosts=192.168.94.128,192.168.94.130,192.168.94.132
dont_blame_nrpe=1
debug=0
command_timeout=60
connection_timeout=300
command[check_xen_dom0]=/usr/lib/nagios/plugins/check_xen dom0
command[check_xen_domU]=/usr/lib/nagios/plugins/check_xen domU $ARG1$
```

Fedora hat durch einen anderen Benutzernamen für NRPE eine leicht abgewandelte Version, die sich lediglich in den Variablen `nrpe_user` und `nrpe_group` unterscheidet:

```
pid_file=/var/run/nrpe.pid
server_port=5666
nrpe_user=nrpe
nrpe_group=nrpe
allowed_hosts=192.168.94.128,192.168.94.130,192.168.94.132
dont_blame_nrpe=1
debug=0
command_timeout=60
connection_timeout=300
command[check_xen_dom0]=/usr/lib/nagios/plugins/check_xen dom0
command[check_xen_domU]=/usr/lib/nagios/plugins/check_xen domU $ARG1$
```

Selbiges gilt für SUSE, wo die beiden folgenden Zeilen unterschiedlich sind:

```
nrpe_user=nobody
nrpe_group=nobody
```

Wichtig ist an dieser Stelle vor allem die Variable `allowed_hosts`, in der – durch Kommata getrennt – eine Liste von IP-Adressen angegeben werden kann, die sich mit dem NRPE-Daemon verbinden dürfen. Damit die Änderungen aktiv werden, muss der NRPE-Server gestartet werden.

Debian:

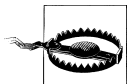
```
root@ritchie:~# /etc/init.d/nagios-nrpe-server start
```

Fedora und SUSE:

```
root@ritchie:~# /etc/init.d/nrpe start
```

Da die von NRPE ausgeführten Checks mit den Rechten des unprivilegierten Benutzers `nagios` laufen, muss das System so konfiguriert werden, dass der Benutzer `nagios` die Befehle `lsof` und `xm` mit root-Rechten ausführen kann. Hierfür starten wir den Befehl `visudo` und fügen am Ende der angezeigten Konfigurationsdatei unter Debian folgende Zeilen hinzu:

```
nagios          ALL=NOPASSWD: /usr/bin/lsof
nagios          ALL=NOPASSWD: /usr/sbin/xm
```



Unter Fedora befindet sich in der sudo-Konfigurationsdatei */etc/sudoers* ein Eintrag namens *requiretty*. Dieser verhindert, dass Prozesse, die sich über ein virtuelles Terminal am System angemeldet haben, *sudo* verwenden dürfen. Möchten Sie Nagios in Verbindung mit *nrpe* auf Fedora verwenden, muss diese *requiretty*-Anweisung mit einer Raute auskommentiert werden.

Außerdem weichen bei Fedora der Pfad zum Programm *lsuf* sowie der verwendete Benutzername von anderen Distributionen ab. Unter Fedora muss der Eintrag in der Datei */etc/sudoers* wie folgt lauten:

```
nrpe    ALL=NOPASSWD: /usr/sbin/xm
nrpe    ALL=NOPASSWD: /usr/sbin/lsuf
```

Bei SUSE läuft das *nrpe*-Programm mit den Rechten des unprivilegierten Benutzers *nobody*, weshalb der Eintrag für die Datei */etc/sudoers* wie folgt aussehen muss:

```
nobody    ALL=NOPASSWD: /usr/bin/lsuf
nobody    ALL=NOPASSWD: /usr/sbin/xm
```

Anpassung der zu überwachenden Domain-U

Neben der Überwachung des *xend* innerhalb der Domain-0 kann auch eine Domain-U wie ein ganz »gewöhnlicher« nicht virtualisierter Server durch Nagios überwacht werden. Analog zur vorangegangenen Konfiguration von *nrpe* innerhalb einer Domain-0 kann *nrpe* in einer Domain-U installiert werden. Dabei ändern sich lediglich die bisherigen letzten zwei Zeilen der Konfigurationsdatei */etc/nagios/nrpe.cfg*. Zusätzlich kann es nötig werden, weitere Nagios-Pakete der verwendeten Distribution zu installieren, um weitere Plugins zu erhalten. Das folgende Beispiel erfordert das installierte *check_disk*-Nagios-Plugin. Dieses befindet sich unter Debian im Paket *nagios-plugins*, unter Fedora in *nagios-plugins-disk* und unter SUSE ebenfalls in *nagios-plugins*. Damit kann der verbleibende freie Speicherplatz auf einer Partition überwacht werden. Sobald nur noch 20 Prozent zur Verfügung stehen, wird eine Warnung ausgegeben, und ab 10 Prozent wird der Zustand bei Nagios als kritisch eingestuft.

```
command[check_root_disk]=usr/lib/nagios/plugins/check_disk -w 20 -c 10 -p /dev/hda1
```



Sollten Sie LVM verwenden, muss anstelle von *check_disk* das an folgender URL hinterlegte Nagios-Plugin verwendet werden: <http://www.nagios-exchange.org/cgi-bin/page.cgi?g=Detailed%2F2103.html;d=1>.

Aufruf der Weboberfläche

Nach der erfolgreichen Konfiguration und einer kleinen Wartezeit, in der Nagios die einzelnen Plugins der Reihe nach ausführt, kann man sich anhand der Weboberfläche einen Überblick über die aktuell überwachte Systemlandschaft verschaffen. Während unter Fedora und SUSE hierzu der Browser auf die IP-Adresse des Nagios-Hosts unterhalb der Direktive */nagios/* geleitet wird, verwenden Debian-Benutzer hingegen folgende Notation: *http://IP-ADRESSE-DES-NAGIOS-HOSTS/nagios2/*.

Nach der Eingabe des zuvor angelegten Benutzers kann über den Link SERVICE DETAILS der aktuelle Statusbericht aufgerufen werden (siehe Abbildung 13-5).

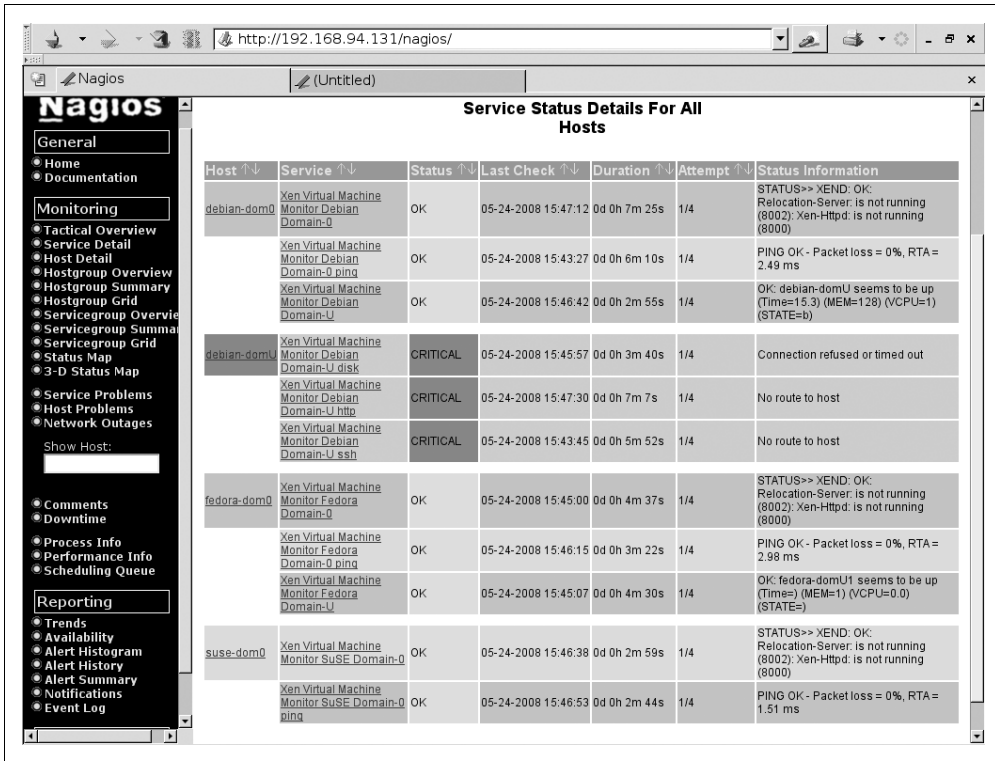


Abbildung 13-5: Screenshot der Nagios-Weboberfläche

Der Screenshot zeigt, dass zurzeit drei Xen-Dom0s überwacht werden. Namentlich sind dies debian-dom0, fedora-dom0 sowie suse-dom0. Außerdem wird eine virtuelle Maschine namens debian-domU überwacht. Für diese wird der Festplattenplatz überwacht, und es wird kontrolliert, ob sowohl der ssh- als auch der http-Service laufen. Leider ist keiner dieser drei Tests erfolgreich, da die virtuelle Maschine zurzeit ein Netzwerkproblem hat.

Diskussion

NRPE-Konfiguration

Nagios bringt einen eigenen Mechanismus mit, um entfernte Server zu überwachen – den Nagios Remote Plug-in Executor (NRPE). Hierbei handelt es sich um einen Daemon, der auf Anfrage des Nagios-Servers ein lokales Plugin ausführt und das Ergebnis an den Nagios-Monitoring-Host zurückgibt (siehe Abbildung 13-6).

Dies ist ein Auszug aus dem Buch "XEN Kochbuch", ISBN 978-3-89721-729-4
http://www.o'reilly.de/catalog/kenbkjger/
Dieser Auszug unterliegt dem Urheberrecht. © O'Reilly Verlag 2009

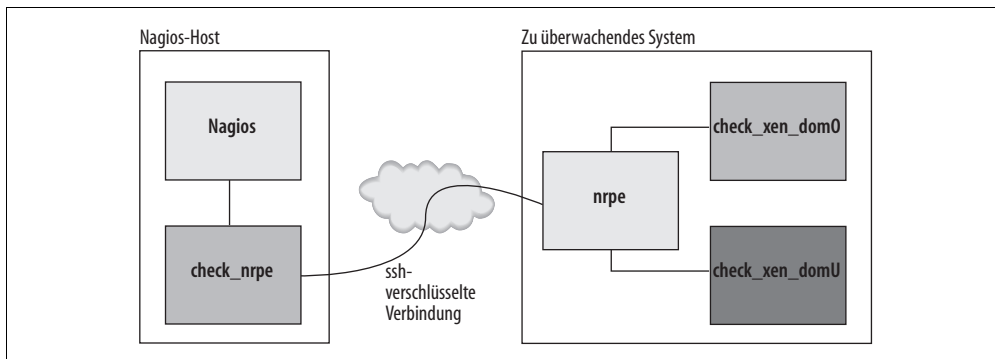


Abbildung 13-6: Schematische Darstellung der Kommunikation zwischen Nagios und NRPE



Alternativ besteht die Möglichkeit, Befehle über `ssh` auszuführen. Hierbei kommt eine Host-Key-Authentifizierung zum Einsatz, die ein Login ohne Passwort ermöglicht. Diese Variante wird hier nicht behandelt, aber weitergehende Informationen finden Sie unter http://nagiosplugins.org/man/check_by_ssh.

Wie in Abbildung 13-6 zu sehen ist, kommuniziert Nagios mit dem zu überwachenden Server mittels `nrpe` über eine verschlüsselte Verbindung. Hierzu werden in der Nagios-Konfiguration Services angelegt, die mit `check_nrpe` überwacht werden sollen. Neben der Konfiguration des Nagios-Servers ist es notwendig, die Client-Seite zu konfigurieren, auf der ein NRPE-Dienst ausgeführt wird. Für das Monitoring Xen-spezifischer Parameter werden die Checks `check_xen_dom0` und `check_xen_domU` verwendet. Diese müssen auf dem zu überwachenden Server installiert werden. Außerdem benötigt der NRPE-Dienst selbst eine Konfiguration, die in der Datei `/etc/nagios/nrpe.cfg` gespeichert wird. Die folgende Tabelle 13-8 liefert eine kurze Beschreibung der Variablen in dieser Konfigurationsdatei.

Tabelle 13-8: Variablen der NRPE-Konfigurationsdatei und ihre Bedeutung

Variable	Bedeutung
<code>server_port</code>	Der Port, auf dem NRPE aktive Netzwerkverbindungen entgegennimmt
<code>nrpe_user</code> <code>nrpe_group</code>	Der Benutzer und die Gruppe, mit deren Rechten der NRPE-Serverprozess ausgeführt wird
<code>allowed_hosts</code>	Eine durch Kommata getrennte Liste von IP-Adressen, von denen NRPE Anfragen entgegennimmt
<code>dont_blame_nrpe</code>	Eine Variable, die kontrolliert, ob NRPE Checks ausführen darf, denen ein Kommandozeilenargument übergeben werden kann
<code>debug</code>	Eine Boolesche Variable, die die Werte 0 und 1 entgegennimmt, um damit zu kontrollieren, ob NRPE Debug-Meldungen an den Syslog-Daemon weitergeben soll
<code>command_timeout</code>	Ein Timeout in Sekunden, der festlegt, nach welcher Zeitspanne die Ausführung eines check-Kommandos abgebrochen wird

Tabelle 13-8: Variablen der NRPE-Konfigurationsdatei und ihre Bedeutung

Variable	Bedeutung
connection_timeout	Ein Timeout, nach dessen Ablauf Verbindungen zum Nagios-Server beendet werden
command[check_xen_dom0]=usr/lib/nagios/plugins/check_xen_dom0 command[check_xen_domU]=usr/lib/nagios/plugins/check_xen_domU \$ARG1\$	Zwei exemplarische von NRPE aufzurufende Checks. Der erste, in eckigen Klammern angegebene Name ist dabei der Name, den der Nagios-Server bei seinem Aufruf verwendet. Auf der rechten Seite des Gleichheitszeichens steht hingegen der Pfad zu dem tatsächlich aufzurufenden Programm oder Skript. Werden diesem Argumente übergeben, haben sie das Format \$ARG1\$, \$ARG2\$ usw.

Die Verwendung der Option `dont_blame_nrpe` ist nicht ganz unproblematisch. Damit können NRPE Argumente übergeben werden, die unter Umständen sicherheitskritische Implikationen haben können, wie folgende von Nagios ausführbare Kommandozeile zeigt:

```
root@ritchie:~# /usr/lib/nagios/plugins/check_nrpe -H 192.168.94.129 -c check_xen_domU -a fedora-domU && rm -rf /1
```

Dabei sollten mindestens zwei Dinge beachtet werden: In der Variable `allowed_hosts` sollte nur die IP-Adresse des Nagios-Servers hinterlegt werden. Zweitens ist `nrpe` mit einem unprivilegierten Benutzer wie `nrpe` oder `nobody` auszuführen.

Wir haben uns im vorliegenden Fall für die Verwendung dieser sicherheitskritischen Option entschieden, da sie die Konfiguration verkürzt. Ein sicherer Ansatz besteht darin, keine Argumente zu verwenden. Welche Auswirkungen hat dies auf die bisher gezeigte Konfiguration? Betrachten wir einen in Nagios konfigurierten Server, der folgende Argumente verwendet:

Nagios-Konfiguration:

```
define service {
    host_name                fedora-dom0
    service_description      Xen Virtual Machine Monitor Fedora Domain-U
    use                      xen-service
    check_command            check_nrpe!check_xen_domU!fedora-domU1
    notification_interval    0
}
```

NRPE-Konfiguration:

```
command[check_xen_domU]=usr/lib/nagios/plugins/check_xen_domU $ARG1$
```

Um das Argument aus der Konfiguration zu entfernen, muss für jede in der obigen Konfiguration zu überwachende Domain-U ein eigener Eintrag in der Konfigurationsdatei `nrpe.cfg` hinterlegt werden. Um zu überprüfen, ob auf einer Domain-0 eine virtuelle Maschine namens `fedora-domU1` läuft, muss der obige Eintrag wie folgt abgeändert werden:

Aufseiten von NRPE:

```
command[check_xen_domU_fedora-domU1]=usr/lib/nagios/plugins/check_xen_domU fedora-domU1
```

Auf dem Nagios-Server:

```

define service {
    host_name                fedora-dom0
    service_description      Xen Virtual Machine Monitor Fedora Domain-U
    use                      xen-service
    check_command            check_nrpe_2arg!check_xen_domU_fedora-domU1
    notification_interval    0
}
  
```

Anstelle von variablen Argumenten ist der Name der virtuellen Maschine jetzt hart codiert und kann nun ohne die Option `dont_blame_nrpe` verwendet werden. Nach einem Update der Konfiguration kann der Variable unter NRPE der Wert 0 zugewiesen werden. Wie bei Unix-Diensten üblich, muss die Konfigurationsdatei im Anschluss neu eingelesen werden, damit die Änderungen aktiv werden.

Fehlersuche bei der Nagios-Konfiguration

Mit Hilfe des Kommandos

```
root@ritchie:~# nagios -v /etc/nagios/nagios.cfg
```

ist es möglich, die Konfiguration von Nagios auf Fehler zu überprüfen. Dies ist insbesondere dann hilfreich, wenn beim Versuch, Nagios über das init-Skript `/etc/init.d/nagios` zu starten, dieser Vorgang lediglich mit einer knappen Fehlermeldung gewürdigt und abgebrochen wird.

Fehlersuche im Zusammenhang mit NRPE

Bleibt der Status der Xen-Systeme in Nagios aufgrund eines Fehlers auf »rot«, ist dies oftmals auf ein Kommunikationsproblem zwischen Nagios und dem NRPE-Daemon auf der zu überwachenden Domain-0 zurückzuführen. Der folgende Befehl kontrolliert in der Domain-0, ob der NRPE-Daemon aktiv und für Anfragen aus dem Netzwerk bereit ist:

```

root@ritchie:~# netstat -an|grep -i nrpe
tcp        0      0 0.0.0.0:5666          0.0.0.0:*            LISTEN     31940/nrpe
unix      2      [ ]          DGRAM              193173    31940/nrpe
  
```

Außerdem ist es möglich, auf dem Nagios-Host über die Kommandozeile eine Anfrage analog zum folgenden Beispiel zu stellen, in dem der Parameter `-H` die IP-Adresse der Domain-0 enthält:

```
root@ritchie:~# /usr/lib/nagios/plugins/check_nrpe -H 192.168.1.102 -c check_xen_dom0
```



Außerdem kann es nützlich sein, den NRPE-Daemon im Debugging-Modus zu betreiben. Dabei wird in der Konfigurationsdatei `/etc/nagios/nrpe.cfg` der Wert der Variable `debug` auf 1 gesetzt. Dadurch werden alle Anfragen an `nrpe` in der Protokolldatei des `syslogd` erfasst (je nach Distribution `/var/log/messages` oder `/var/log/syslog`).

```
debug=1
```

Die Nagios-Grundlagen gehen weit über das hier Gezeigte hinaus und können an dieser Stelle nicht weiter vertieft werden. Neben den bisher gesehenen Modulen steht noch eine Vielzahl von Plugins für die Überwachung aller erdenklichen Systemkonfigurationen zur Verfügung. NRPE gibt es beispielsweise auch für Windows-Systeme.

Betreibt man mehrere Server mit einer vergleichbaren Konfiguration, ist es ratsam, diese in der Nagios-Konfiguration in Gruppen zusammenzufassen und anschließend konfigurierte Services nicht mit einzelnen Hosts, sondern mit aus mehreren Servern bestehenden Hostgruppen zu assoziieren.



Besonders ans Herz gelegt sei dem zukünftigen Nagios-Benutzer die Nagios-Exchange-Webseite <http://www.nagiosexchange.org>, auf der man eine reichhaltige Sammlung von Plugins mit mehr oder weniger ausführlichen Installations- und Konfigurationsbeispielen finden kann.

Siehe auch

- <http://www.nagiosexchange.org> – eine Plattform zum Austausch von Nagios-Plugins, einer Ansammlung von Nagios-Erweiterungen zur Überwachung spezifischer Komponenten
- <http://www.nagios.org> – die Webseite des Nagios-Projekts
- <http://nagios.sourceforge.net/docs/nrpe/NRPE.pdf> – die offizielle Dokumentation von NRPE
- <http://www.nagios-wiki.de/nagios/howtos/nrpe?s=nrpe> – ein Artikel des deutschsprachigen Nagios-Wikis zur Konfiguration von NRPE
- <http://www-user.tu-chemnitz.de/~volz/nagios/> – ein deutschsprachiges Howto zur Installation und Konfiguration von Nagios und NRPE

13.7 Alternative Lösungen

Problem

Ihnen ist Nagios zu komplex und Sie interessieren sich für Alternativen. Im Folgenden gehen wir auf verschiedene vergleichsweise einfach einzurichtende Monitoring-Lösungen ein.

Lösung

Argo

Argo ist ein Framework zum Monitoring virtueller Maschinen auf der Basis von Xen. Zur Zeit stagniert das Projekt und wird momentan nicht weiterentwickelt, wohl aufgrund einer intensiveren Redesign-Phase. Weitere Informationen finden Sie auf der Projekt-Webseite unter <http://www.steve.org.uk/Software/argo/>.

Xenoprof

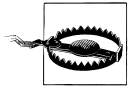
Xenoprof ist ein sogenannter Profiler für virtuelle Maschinen auf der Basis von Xen und ist in der Lage, Profiling-Informationen von den Betriebssystemen und Applikationen aufzuzeichnen, die innerhalb von Domain-Us ausgeführt werden. Dazu gehören unter anderem auch folgende Hardware-Events: clock cycles, instruction execution und TLB cache misses.

Xenoprof wurde in den HP Labs als Erweiterung des existierenden Linux-Profilers Oprofile entwickelt.

Folgende Programme befinden sich bereits im Lieferumfang von Xen:

Xentrace

Xentrace ist ein Programm, das verwendet werden kann, um bestimmte Events im Xen-Hypervisor aufzuspüren und ihr Vorkommen sowie die Dauer ihrer Ausführung zu protokollieren.



Unter Debian GNU/Linux 4.0 ist die Ausführung von xentrace leider nicht fehlerfrei möglich, bevor nicht zuerst folgende Kommandozeile eingegeben wurde:

```
echo "/usr/lib/xen-3.0.3-1/lib/" > /etc/ld.so.conf.d/xen.conf && ldconfig
```

Xenbackend

Oft sind die von xentrace aufgezeichneten Protokolle alleine nicht sehr nützlich. Hier kommt mit xenbackend ein Programm ins Spiel, das aus den xentrace-Protokollen beispielsweise Informationen über den Zustand der virtuellen Maschine (*Wake Ups*, *Sleep*- oder *Blocked*-Status) zusammenstellt.

Außerdem exportiert xenbackend die verarbeiteten Daten in einem Text-Format, so dass es leicht möglich ist, ein Frontend zur Darstellung der Daten in einer beliebigen Programmiersprache zu entwickeln (z.B. xenmon).

XenMon (Xen-Performance-Monitor)

Das Programm xenmon ist ein Frontend für bestehende Xen-Tracing-Mechanismen wie xentrace und xenbackend. Es ist Teil der Toolsuite, die zum Software-Umfang von Xen gehört. Im Kommentar des Quellcodes des *XenMon*-Programms weisen die Autoren der Software darauf hin, dass das xenmon-Skript als Beispiel zu sehen ist, wie Xen-Performance- und -Monitoring-Daten dargestellt werden können.



Die Software wurde im Zuge einer Fallstudie erstellt, die zu dem nachfolgenden Bericht der Linux-Abteilung von HP geführt hat: <http://www.hpl.hp.com/techreports/2005/HPL-2005-187.pdf>.

xenperf

xenperf.c ist eine kleine C-Anwendung, die von Rolf Neugebauer bei Intel Research Cambridge entwickelt wurde. Das Programm erstellt Histogramme und verwaltet einen dazugehörigen Zähler zur Ausgabe der verwendeten Hypercalls.

libxenstat

libxenstat ist eine Bibliothek, die verwendet werden kann, um statistische Informationen über den Xen-Daemon abzufragen. Zur Zeit existiert eine Anbindung von *libxenstat* an die Programmiersprache Python. Diese Bibliothek wird unter anderem von dem Programm `xm top` (auch bekannt als `xentop`) verwendet.

xenaccess

xenaccess ist eine an der Universität Georgia Tech entwickelte Bibliothek, die es ermöglicht, *Memory Introspection* durchzuführen, ohne sich mit den Low-level-Implementierungsdetails zu beschäftigen. Mit dieser Bibliothek kann man Informationen über den Hauptspeicher virtueller Maschinen abfragen und auf Ressourcen wie Kernel-Symbole sowie virtuelle und physikalische Adressen zugreifen.

Diskussion

Abgesehen von Argo, dem Framework zum Monitoring von virtuellen Maschinen auf Basis von Xen, das sich zurzeit in einer Umbruchphase befindet, kann keines der an dieser Stelle vorgestellten Programme direkt mit Nagios verglichen werden. Ein Großteil der in diesem Problem vorgestellten Lösungen fällt im Kontext von Monitoring in die Unterkategorie des Profiling. Dabei wird das Laufzeitverhalten virtueller Maschinen gemessen und protokolliert. Ein Teil dieser unter Umständen wenig Endbenutzer-freundlichen Programme stammt direkt aus dem Umfeld der Xen-Entwickler und wurde unter anderem zur Messung der Xen-Performance entwickelt. Viele erlauben eine hochgradig detaillierte Protokollierung, von deren Auswertung jedoch oft nur der intime Kenner der Xen-Implementierung profitiert.

Auch wenn die initiale Inbetriebnahme von Nagios einmalig mit einem vergleichsweise hohen Aufwand verbunden ist, gestaltet sich die anschließende »einfache Überwachung« jedoch weitestgehend problemlos, wenn der Administrator primär daran interessiert ist festzustellen, ob ein Host und die darauf ausgeführten Gastssysteme noch erreichbar sind oder sich innerhalb eines vorgegebenen Toleranz-Rahmens bewegen.

Siehe auch

- <http://www.steve.org.uk/Software/argo/> – die Webseite des Argo-Projekts, eines Monitoring-Frameworks für Xen-basierte Systeme
- <http://xenoprof.sourceforge.net/> – die Webseite der xenoprof-Entwickler bei HP

- http://xenoprof.sourceforge.net/xenoprof_2.0.txt – der offizielle Xenoprof-Userguide
- <http://oprofile.sourceforge.net> – die Webseite des Oprofile-Projekts, das sich mit dem Profiling Linux-basierter Installationen beschäftigt
- XENTRACE(8) – die Manpage von xentrace, einem Programm zur Sammlung von Xen-Event-Daten
- XENTRACE_FORMAT(8) – eine Manpage des Programms, das die binäre Ausgabe von xentrace formatiert ausgeben kann
- http://kibab.homeip.net/hw/vienna_hw6/tutorial.html – ein Tutorial, das die Monitoring-Tools beschreibt, die in einer Xen-Basis-Installation mitgeliefert werden
- <http://www.hpl.hp.com/techreports/2005/HPL-2005-187.pdf> – ein Technical Report der HP Laboratories in Palo Alto zum Thema »XenMon: QoS Monitoring and Performance Profiling Tool«
- <http://lxr.xensource.com/lxr/source/tools/xenmon/> – das Top-level-Verzeichnis im Xen-Quellcode-Repository, in dem sich alle für xenmon relevanten Dateien inklusive README befinden
- <http://lxr.xensource.com/lxr/source/tools/misc/xenperf.c> – der Quellcode des xenperf-Programms
- <http://lxr.xensource.com/lxr/source/tools/xenstat/libxenstat/> – die libxenstat-Bibliothek, die von xm top verwendet wird
- <http://xenaccess.sourceforge.net/doc/> – die offizielle Dokumentation des xenaccess-Projekts