Dokumentation des Wettkampfsystems der Software-Challenge

25. Oktober 2010

Inhaltsverzeichnis

Kapitel I

Einführung

Die Software-Challenge ist ein Schulprojekt, das durch das Institut für Informatik der CAU Kiel veranstaltet wird und das durch zahlreiche Unternehmen, die Prof. Dr. Werner Petersen-Stiftung sowie das Ministerium für Wissenschaft, Wirtschaft und Verkehr des Landes Schleswig-Holstein finanziert wird. In Zusammenarbeit mit den Gymnasien und Gesamtschulen und dem Institut für Informatik sowie den Firmen soll in diesem Projekt der Informatik-Unterricht in praxisbezogener Weise mitgestaltet und dadurch aufgewertet werden. Gegenstand der Software-Challenge ist ein Programmierwettbewerb, der während des gesamten Schuljahres läuft, und der den Schülerinnen und Schülern die Möglichkeit bietet, mit Spaß und Spannung sowie mit kompetenter Begleitung in die Welt der Informatik einzusteigen. Für die Verwaltung der Teams, die von den Schülern entwickelten Computerspieler (Clients) und die Abwicklung der Spieltage wird ein Online-Wettkampfsystem benutzt, das einen termingerechten und regelkonformen Ablauf der Spiele sicherstellt.

In dieser Dokumentation soll von technischer Seite das Zusammenspiel der Anwendungen und Prozesse des Wettkampfsystems erläutert werden. Dazu gehören der Webserver, der Spielserver und die virtuellen Maschinen zur Clientausführung. Weiterhin soll hier noch auf die Datensicherung eingegangen werden. Für Informationen zur Software-Challenge im Allgemeinen, dem Wettkampfverlauf und der Anwendersoftware steht unter http://sc-doku.gfxpro.eu/wiki/Hauptseite ein Wiki zur Verfügung.

Kapitel II

Numerobis

Auf Numerobis laufen die virtuellen Maschinen (VMs), auf denen die meisten Anwendungen der Software-Challenge ausgeführt werden. Zusätzlich gibt es einige Programme und Scripte, die helfen, die VMs zu steuern.

1 Hardware und OS

Numerobis besitzt 16 Xeon-Kerne mit jeweils 2,27GHz. Weiterhin verfügt er über 24GB RAM. Als Betriebssystem wird SunOS genutzt, genauer:

```
> uname -a
SunOS numerobis 5.10 Generic_142901-04 i86pc i386 i86pc
```

2 Netzplan

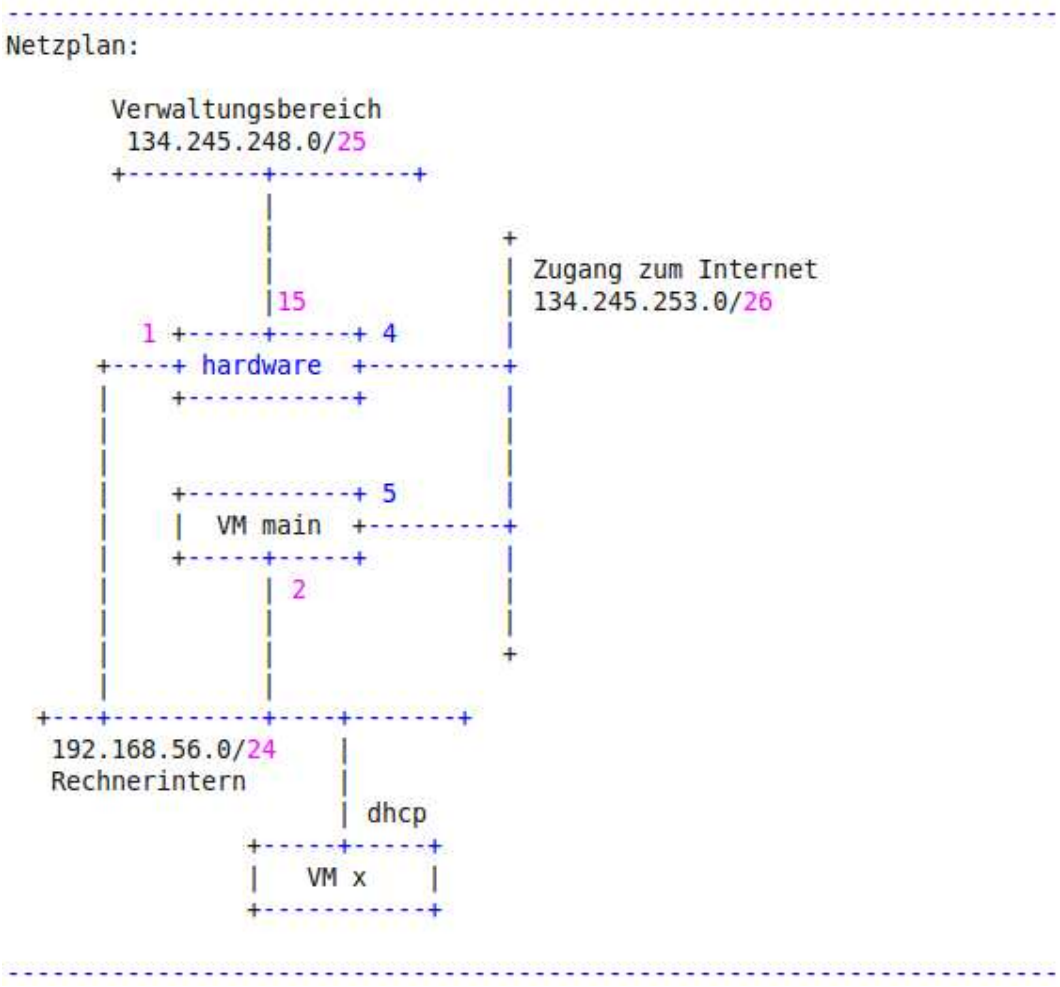

Abbildung II.1: Netzplan von Numerobis

3 Software

3.1 VirtualBox

VirtualBox wird genutzt um virtuelle Maschinen bereitzustellen, auf denen das Websystem und die Computerspieler ausgeführt werden.

3.1.1 Steuern der VMs

Hier ein paar grundlegende Befehle und Scripte zur VM-Steuerung. Für alles Weitere sind die Hilfe des Befehls VBoxManage und Dokumentationen zu VirtualBox zu Rate zu ziehen.

Informationsgewinnung

```
VBoxManage list runningvms
VBoxManage list vms
VBoxManage list hdds
```

Listet alle

- laufenden VMs
- registrierten VMs
- registrierten Festplatten

auf.

```
VBoxManage showvminfo <VM>
VBoxManage guestproperty enumerate
```

Zeigt an:

- Hardwareinformationen der angegebenen VM
- Informationen des (laufenden) Gastsystems

Modifizierung

```
VBoxManage createvm -name <VM> -register --ostype <OS>
VBoxManage unregistervm <VM> [--delete]
```

- Erstellt und registriert eine neue VM mit dem Namen VM und dem angegebenen Betriebssystem (z.B. Ubuntu)
- Hebt die Registrierung der VM auf [und löscht diese]

```
VBoxManage modifyvm <VM> --memory <MEM> --cpus <CPUS>
VBoxManage modifyvm <VM> --nic1 hostonly --nictype1 82540EM --cableconnected1 on
    --hostonlyadapter1 <IF> --macaddress1 auto
VBoxManage modifyvm <VM> --hda ``/path/to/image.vdi"
VBoxManage modifyvm $vmname --hda none
```

Ändert die Hardware der angegebenen VM:

- Setzt den RAM auf MEM MB und die Anzahl der CPUs auf CPUS.
- Richtet einen Netzwerkadapter für das Interface IF ein. Neben hostonly ist zum Beispiel auch bridged möglich, um der VM die Benutzung eines Host-Netzwerkinterfaces zu ermöglichen.
- Setzt die Festplatte hda der VM auf das angegebene Image.
- Entfernt die Festplatte hda aus der VM.

```
VBoxManage internalcommands sethduuid ``/path/to/image.vdi''
VBoxManage closemedium disk ``/path/to/image.vdi''
```

- Ändert die UUID des angegebenen Disk-Images
- Hebt die Registrierung des angegebenen Festplattenimages auf

Steuerung

```
VBoxManage startvm <VM> --type headless
VBoxManage startvm <VM> --type vrdp
```

Startet die VM mit dem Namen <VM>

- ohne grafische Oberfläche
- mit VRDP Schnittstelle zur Benutzung eines Remote Desktops.

```
VBoxManage controlvm <VM> poweroff
```

Stoppt die VM mit dem Namen <VM>

```
~/kill_vms.sh
```

Beendet alle laufenden Client VMs und die dazugehörigen Startscripte. Dieses Script sollte mit Vorsicht benutzt werden, da es die VM-Prozesse rücksichtslos beendet ohne die VM oder deren HDD zu deregistrieren und zu löschen.

3.1.2 Die VMMain starten

Die VMMain muss zum Beispiel nach einem Neustart von Numerobis gestartet werden. Dies geschieht mit dem Befehl

```
VBoxManage startvm vmmain --type headless
```

3.1.3 Eine geklonte Client-VM starten

Da jeder Client auf einer eigenen Version der Client-VM gestartet werden soll, muss für jeden Client zunächst ein Klon der Client-VM erstellt und gestartet werden. Auf Numerobis übernimmt dies das Script /home/vbox/-startVM.sh (siehe Anhang 1), trotzdem sollen hier im Einzelnen die nötigen Befehle erläutert werden. Zunächst müssen einige Dinge vorbereitet werden. Dazu gehören ein eindeutiger Name (z.B. der aktuelle Timestamp) des neuen Klons und der (ZFS-)Pfad zum Klon:

```
# Unique number for the VM
vmnr=`bin/date +%m%d%H%M%S`
# Path for VM harddisk
zfspath=``zpool1/vbox/harddisks/vmclient-$vmnr''
vmpath=``/home/vbox/harddisks/vmclient-$vmnr''
```

Nun muss die Image-Vorlage der Client-VM vom Snapshot geklont werden. Dies geschieht folgendermaßen:

```
zfs clone zpool1/vbox/harddisks/vmclient@kopie $zfspath
```

Die geklonte HDD braucht unbedingt eine eigene UUID, damit VirtualBox alles auseinanderhalten kann:

```
cd $vmpath
VBoxManage internalcommands sethduuid vmclient.vdi
```

Nun kann eine neue VM erstellt werden. Dann muss die Hardware, Netzwerkverbindung und Festplatte hinzugefügt werden:

```
VBoxManage createvm -name $vmname -register --ostype Ubuntu
VBoxManage modifyvm $vmname --memory 1536 --cpus 1
VBoxManage modifyvm $vmname --nic1 hostonly --nictype1 82540EM --cableconnected1
    on --hostonlyadapter1 vboxnet0 --macaddress1 auto
VBoxManage modifyvm $vmname --hda ``$vmpath/vmclient.vdi"
```

Die erstellte VM kann nun gestartet und benutzt werden:

```
VBoxManage startvm $vmname --type headless
```

Den Klon wieder vernichten

Wird der Klon nicht mehr benötigt, sollte dieser wieder gelöscht werden. Dazu muss man die geklonte Client-VM zunächst wieder beenden. Dann wird die Festplatte entfernt und die Maschine gelöscht. Schließlich muss die HDD noch ausgeworfen werden.

```
VBoxManage controlvm $vmname poweroff
VBoxManage modifyvm $vmname --hda none
VBoxManage unregistervm $vmname --delete
VBoxManage closemedium disk ``$vmpath/vmclient.vdi''
```

Zuletzt noch das Festplatten-Image aus dem Dateisystem löschen:

```
zfs destroy -f $zfspath
```

3.2 Der Consumer

Der Consumer läuft ständig im Hintergrund. In regelmäßigen Abständen prüft er, ob eine neue Anfrage zum Starten einer Client-VM vorliegt. Dazu pollt er eine Queue, die mit "vm-queue" bezeichnet ist (mehr zu den Queues bei 3.3). Liegt eine solche Anfrage vor, startet der Consumer einen neuen Klon der Client-VM. Dazu benutzt er das Startscript /startVM.sh (siehe **??**).

Consumer starten

Der Consumer kann über das Script /home/vbox/startVMConsumer.sh gestartet werden. Dies muss nach einem Neustart von Numerobis manuell durchgeführt werden. Die Startparameter für den Consumer (/home/vbox/daemonconsumer.jar) sind:

- -c: Der Befehl oder das Script der/das ausgeführt werden soll.
- -b: Pfad zur Bash
- -q: Name der Queue, die überprüft werden soll
- -h: Host, auf dem die Queue läuft
- -m: Maximale Anzahl an Aufträgen, die gleichzeitig bearbeitet werden dürfen
- -i: Intervall in Sekunden, in dem die Queue überprüft werden soll
- -d: Debug-Modus, erzeugt erweiterte Ausgaben

3.3 Das ZFS Dateisystem

Zur Verwaltung der Disk-Images für die VMs wird das ZFS Dateisystem und dessen Features zum Klonen genutzt. Das Ausgangsdateisystem für alle Disk-Images ist /home/vbox/harddisks. Ab hier muss für jedes Diskimage ein eigenes weiteres Filesystem angelegt werden. Diese Images sollen NIE direkt benutzt werden. Stattdessen sollte immer erst ein Snapshot/Clone angelegt werden. Das hat den Vorteil, dass das Original immer als Fallback zur Verfügung steht.

Beispiel für OpenSolaris

Zuerst muss ein neues Filesystem angelegt werden.

```
% zfs create zpool1/vbox/harddisks/opensolaris
% zfs list -r zpool1
NAME                                  USED  AVAIL  REFER  MOUNTPOINT
zpool1                               13.2G   121G  6.97G  /zpool1
zpool1/vbox                          6.24G   121G  4.19G  /zpool1/vbox
zpool1/vbox/harddisks                2.05G   121G    23K  /zpool1/vbox/
    harddisks
zpool1/vbox/harddisks/opensolaris    2.05G   121G  2.05G  /zpool1/vbox/
    harddisks/opensolaris
```

Dann erst kann das Image dort abgelegt werden.

```
% cd ~/harddisks/opensolaris
% cp <irgendwoher>/OpenSolaris.vdi .
```

ACHTUNG: Jedes Image braucht eine eigene UUID! Sonst bringt die VirtualBox u.U. die Images durcheinander.

```
% VBoxManage internalcommands sethduuid OpenSolaris.vdi
VirtualBox Command Line Management Interface Version 3.0.8
(C) 2005-2009 Sun Microsystems, Inc.
All rights reserved.

UUID changed to: cd6f93b1-cefd-405a-88ba-b55661928bc3
```

Jetzt wird zuerst ein Snapshot des Originals angelegt. Das muss sein, da Klone nur von Snapshots erzeugt werden können.

```
% zfs snapshot zpool1/vbox/harddisks/opensolaris@kopie
% zfs list -r zpool1
NAME                                  USED  AVAIL  REFER  MOUNTPOINT
zpool1                               13.2G   121G  6.97G  /zpool1
zpool1/vbox                          6.24G   121G  4.19G  /zpool1/vbox
zpool1/vbox/harddisks                2.05G   121G    23K  /zpool1/vbox/
    harddisks
zpool1/vbox/harddisks/opensolaris    2.05G   121G  2.05G  /zpool1/vbox/
    harddisks/opensolaris
zpool1/vbox/harddisks/opensolaris@kopie      0      -  2.05G  -
```

Und dann den Clone erzeugen:

```
% zfs clone zpool1/vbox/harddisks/opensolaris@kopie zpool1/vbox/harddisks/
    opensolaris-1
% zfs list -r zpool1
```

```
NAME                                        USED  AVAIL  REFER  MOUNTPOINT
zpool1                                     13.2G   121G  6.97G  /zpool1
zpool1/vbox                                6.24G   121G  4.19G  /zpool1/vbox
zpool1/vbox/harddisks                      2.05G   121G    23K  /zpool1/vbox/
    harddisks
zpool1/vbox/harddisks/opensolaris          2.05G   121G  2.05G  /zpool1/vbox/
    harddisks/opensolaris
zpool1/vbox/harddisks/opensolaris@kopie        0      -  2.05G  -
zpool1/vbox/harddisks/opensolaris-1            0   121G  2.05G  /zpool1/vbox/
    harddisks/opensolaris-1
```

Dieser Clone kann beschrieben werden, und sollte sofort mit einer eigenen UUID versorgt werden.

```
%  cd ../*1
/home/vbox/harddisks/opensolaris-1

% ls -l
-rw-------   1 vbox     vbox     3845210624 Oct 15 12:42 OpenSolaris.vdi

% VBoxManage internalcommands sethduuid OpenSolaris.vdi
VirtualBox Command Line Management Interface Version 3.0.8
(C) 2005-2009 Sun Microsystems, Inc.
All rights reserved.

UUID changed to: fcafca0b-3e77-4259-a954-01d4ab5e4954

% zfs list -r zpool1
NAME                                        USED  AVAIL  REFER  MOUNTPOINT
zpool1                                     13.2G   121G  6.97G  /zpool1
zpool1/vbox                                6.24G   121G  4.19G  /zpool1/vbox
zpool1/vbox/harddisks                      2.05G   121G    24K  /zpool1/vbox/
    harddisks
zpool1/vbox/harddisks/opensolaris          2.05G   121G  2.05G  /zpool1/vbox/
    harddisks/opensolaris
zpool1/vbox/harddisks/opensolaris@kopie        0      -  2.05G  -
zpool1/vbox/harddisks/opensolaris-1         54.5K   121G  2.05G  /zpool1/vbox/
    harddisks/opensolaris-1
```

Wie man gut erkennen kann, belegt der Clone statt 2GB nur 55KB und kann trotzdem völlig unabhängig vom Original genutzt werden. Bei Bedarf ist schnell eine zweite Kopie erstellt:

```
% zfs clone zpool1/vbox/harddisks/opensolaris@kopie zpool1/vbox/harddisks/
    opensolaris-2
% cd ../*2
% VBoxManage internalcommands sethduuid OpenSolaris.vdi
VirtualBox Command Line Management Interface Version 3.0.8
(C) 2005-2009 Sun Microsystems, Inc.
All rights reserved.

UUID changed to: 55cec833-3685-483b-9a29-e90cf9b33800

% zfs list -r zpool1
NAME                                        USED  AVAIL  REFER  MOUNTPOINT
```

```
zpool1                                    13.2G   121G  6.97G  /zpool1
zpool1/vbox                               6.24G   121G  4.19G  /zpool1/vbox
zpool1/vbox/harddisks                     2.05G   121G    25K  /zpool1/vbox/
    harddisks
zpool1/vbox/harddisks/opensolaris         2.05G   121G  2.05G  /zpool1/vbox/
    harddisks/opensolaris
zpool1/vbox/harddisks/opensolaris@kopie      0      -  2.05G  -
zpool1/vbox/harddisks/opensolaris-1       54.5K   121G  2.05G  /zpool1/vbox/
    harddisks/opensolaris-1
zpool1/vbox/harddisks/opensolaris-2       54.5K   121G  2.05G  /zpool1/vbox/
    harddisks/opensolaris-2
```

Kapitel III

Die Client-VM

Die Client-VMs dienen dazu, die Computerspieler in einer isolierten und genau festgelegten Umgebung auszuführen. Für jeden auszuführenden Computerspieler wird ein neuer Klon der Client-VM erstellt und gestartet.

1 Konzept

Nach dem Boot der Client-VM wird auf ihr automatisch ein weiterer Consumer (/home/scadmin/swcconsumer.jar) gestartet, der einen Eintrag aus der Queue "swc-job-queue" ausliest (mehr zu den Queues bei 3.3). Dieser Eintrag enthält einen SCP-Befehl, der den vorbereiteten Client von der VMMain auf die Client-VM kopiert. Dort wird der Client entpackt und ausgeführt.

2 Hardware

Jede Client-VM läuft mit einer Xeon 2,27GHz (1 Kern) CPU und 1,5 GB RAM. Als Betriebssystem wird ein 32-bit Ubuntu benutzt, genauer:

```
> uname -a
Linux base 2.6.31-14-generic #48-Ubuntu SMP Fri Oct 16 14:04:26 UTC 2009 i686
    GNU/Linux
```

3 Software

Ein Eintrag in /etc/rc.local startet nach dem Booten den Consumer:

```
/usr/bin/java -jar /home/scadmin/swcconsumer.jar 192.168.56.2
```

Dieser liest aus der "swc-job-queue" den nächsten Eintrag (einen SCP-Befehl) aus und übergibt diesen an das Consumer-Script /home/scadmin/consume.sh. Dieses bereitet zunächst den Ordner vor, in dem der Client später ausgeführt werden soll:

```
/bin/mkdir $clientdir
/bin/chown clientexec:clientexec $clientdir
/bin/chmod 777 $clientdir
```

Dann führt es den übergebenen SCP-Befehl aus und kopiert damit das Client-Archiv von der VMMain in den richtigen Ordner auf der Client-VM:

```
`$scpcommand $zipfile` >> $log 2>&1
Führt zu:
/usr/bin/scp -i /home/scadmin/id_rsa scadmin@192.168.56.2:/home/scadmin/tmp
    /1286904225_400_gymnasium-elmschenhagen_634.zip /home/clientexec/client/
    client.zip
```

Danach löscht das Script das Client-Archiv auf der VMMain:

```
sudo -u scadmin ssh -l scadmin 192.168.56.2 rm /home/scadmin/tmp/${file} >> $log
    2>&1 --
```

Jetzt wird das Client-Archiv entpackt und dem Benutzer "clientexec" gegeben, unter dem der Client ausgeführt werden soll. Außerdem muss noch das Startscript des Clients (startup.sh) ausführbar gemacht werden:

```
cd $clientdir
/usr/bin/unzip $zipfile >> $log 2>&1
/bin/chown -R clientexec:clientexec .
/bin/chmod +x $startup
```

Schließlich wird der Client mit dem Benutzer "clientexec" ausgeführt. Der Benutzerwechsel stellt sicher, dass der Client keinen Zugang zu sensiblen Daten oder gar SSH-Zugang zur VMMain erhält.

```
sudo -Hu clientexec /bin/bash +x $startup >> $log 2>&1 --
```

4 Die Client-VM verändern

Möchte man die Vorlage der Client-VM verändern, muss man die Client-VM selbst starten (auf Numerobis):

```
VBoxManage startvm vmclient --type headless
```

Nun braucht die VM einige Zeit zum Hochfahren. Ist die VM hochgefahren, kann man die IP der VM auslesen:

```
VBoxManage guestproperty enumerate vmclient
```

Entscheidend ist folgende Zeile:

```
Name: /VirtualBox/GuestInfo/Net/0/V4/IP, value: 192.168.56.56, timestamp:
    1286982333177594000, flags:
```

Ist diese Zeile noch nicht in der Ausgabe vorhanden, ist die VM noch nicht vollständig hochgefahren. Nun kann man sich (am besten von der VMMain aus) per SSH auf die VMClient verbinden und alle nötigen Änderungen vornehmen. Danach sollte man die VMClient wieder herunterfahren. Vorzugsweise nicht einfach abschalten, sondern richtig Herunterfahren:

```
sudo shutdown -P now
```

Nun muss noch die Klonvorlage im ZFS-Dateisystem aktualisiert werden. Dazu muss zunächst der bereits vorhandene Snapshot gelöscht werden:

```
zfs destroy -R zpool1/vbox/harddisks/vmclient@kopie
```

Dann wird das HDD-Image der ClientVM in das ZFS-Dateisystem kopiert:

```
cd /home/vbox/harddisks/vmclient
cp /home/vbox/harddisks/vm/vmclient.vdi .
```

Und zuletzt ein neuer Snapshot erstellt:

```
zfs snapshot zpool1/vbox/harddisks/vmclient@kopie
```

Das Aktualisieren des Snapshots wird komplett auch vom Script "/home/vbox/updateVMClient.sh" (siehe Anhang 2) übernommen.

Kapitel IV

Die VMMain

1 Konzept

Auf der VMMain läuft die Rails-Webanwendung des Wettkampfsystems sowie der Software-Challenge Server, auf dem die Spiele ausgetragen werden.

2 Hardware

Die VMMain verfügt über eine Xeon CPU mit 2,27GHz und 3,5GB RAM. Als Betriebssystem wird Ubuntu verwendet, genauer:

```
> uname -a
Linux base 2.6.32-25-generic #45-Ubuntu SMP Sat Oct 16 19:48:22 UTC 2010 i686
    GNU/Linux
```

3 Software

3.1 Die WebApp

Die Webanwendung (WebApp) ist eine Rails-Anwendung, die über den Apache Webserver bei Bedarf automatisch gestartet wird. Die WebApp wird allerdings durch einige Anwendungen ergänzt, die gesondert gestartet werden müssen.

3.1.1 Der Job-Worker

Zur geplanten Abarbeitung von Aufgaben im Hintergrund (zum Beispiel das Durchführen von Spielen) wird der DelayedJob-Worker von Rails genutzt. Damit dieser seine Aufgaben auch erfüllt, müssen die Worker-Prozesse laufen. Nach einem Neustart der VMMain müssen diese evtl. manuell gestartet werden. Dies geschieht folgendermaßen:

```
cd /home/scadmin/rails-workspace
cap delayed_job:start
```

3.1.2 Daemons

Es gibt einige Daemons, die ständig im Hintergrund parallel zur WebApp laufen, z.B. um den Server zu Überwachen und automatische Abläufe in der WebApp zu steuern. Diese Daemons müssen evtl. nach einem Neustart der VMMain manuell gestartet werden. Dies geschieht folgendermaßen:

```
cd /home/scadmin/rails-workspace
cap daemons:start
```

3.2 Der Spielserver

Der Spielserver ist eine Java-Anwendung, die ständig läuft. Sie nimmt Verbindungen von der WebApp entgegen, um neue Spiele zu öffnen, auf die sich dann die auf den Client-VMs ausgeführten Computerspieler verbinden können. Der Spielserver wird automatisch zusammen mit den Daemons gestartet. Die Anwendung befindet sich in /home/scadmin/rails-deployment/current/public/server/.

3.3 Die Queues

Eine Warteschlange (RabbitMQ) wird genutzt, um Aufträge für zu startende VMs zu verwalten. Auf der VMMain werden Einträge in die Warteschlange eingefügt, die auf Numerobis und den Client-VMs ausgelesen werden. Ein Eintrag in die Warteschlange "vm-queue" hat zur Folge, dass eine neue Client-VM auf Numerobis gestartet wird. Der Inhalt des Eintrags ist dabei egal. Gleichzeitig sollte in die Warteschlange "swc-job-queue" ein Eintrag eingetragen werden, der einen SCP-Befehl zum Kopieren des vorbereiteten Clients auf die Client-VM enthält. Dieser sieht z.B. so aus:

```
/usr/bin/scp -i /home/scadmin/id_rsa scadmin@192.168.56.2:/home/scadmin/tmp
    /1286904225_400_gymnasium-elmschenhagen_634.zip
```

Das Eintragen der entsprechenden Elemente in die Queues wird auf der VMMain vom Producer (s. 3.4) übernommen.

3.4 Der Producer

Der Producer überwacht ein angegebenes Verzeichnis. Findet er in diesem einen vorbereiteten Client (ZIP-Archiv), verschiebt er diesen in einen spezifizierten Ordner und erstellt jeweils einen Eintrag in die Queues "vm-queue" und "swc-job-queue". Derzeit ist der überwachte Ordner /home/scadmin/clients/ und der Ordner, in dem die Ordner auf die VM warten ist /home/scadmin/tmp.

Producer starten

Der Producer kann über das Script /startProducer.sh gestartet werden. Dieses Script ruft god auf, ein Prozessüberwachungstool, das sicherstellt, das der Producer ständig läuft und bei einem Absturz neugestartet wird. Der Aufruf sieht folgendermaßen aus (einmal mit god und einmal direkt)

```
god -c ~/monitoring/producer.conf # Aufruf mit god
java -jar /home/scadmin/producer.jar <Startparameter> # Direkter Aufruf,
    unüberwacht
```

Der Producer muss zum Beispiel nach einem Neustart der VMMain evtl. manuell gestartet werden. Die Startparameter für den Producer (producer.jar) sind:

- -w: Verzeichnis, das überwacht werden soll
- -s: Der SCP-Befehl zum Kopieren des Clients auf die Client-VM
- -h: Host auf dem die RabbitMQ läuft
- -t: Das Verzeichnis, in das der behandelte Client verschoben werden soll.

Kapitel V

Anhang

1 startVM.sh

```
#!/bin/bash

message=$1
log="/home/vbox/logs/script.log"
/bin/echo "Starting a new VM at \frac{bin/date`" >> $log

# Unique number for the VM
vmnr=\frac{bin/date +%m%d%H%M%S`
# Path for VM harddisk
zfspath="zpool1/vbox/harddisks/vmclient-$vmnr"
vmpath="/home/vbox/harddisks/vmclient-$vmnr"

# Clone the VM HDD template
echo "creas sethduuid vmclient.vdi
cd

# Create and start new VM using the cloned HDD
/bin/echo "creating vm"
vmname="vmclient-$vmnr"
VBoxManage createvm -name $vmname -register --ostype Ubuntu
VBoxManage modifyvm $vmname --memory 1536 --cpus 1
VBoxManage modifyvm $vmname --nic1 hostonly --nictype1 82540EM --cableconnected1
    on --hostonlyadapter1 vboxnet0 --macaddress1 auto
VBoxManage modifyvm $vmname --hda "$vmpath/vmclient.vdi"

/bin/echo "starting vm"
VBoxManage startvm $vmname --type headless

# Give the VM 5 minutes to run before killing it again
/bin/echo "5"
sleep 60
/bin/echo "4"
sleep 60
/bin/echo "3"
```

```
sleep 60
/bin/echo "2"
sleep 60
/bin/echo "1"
sleep 60
/bin/echo "0"

# Parse the VM's ip and call the log copy script on VMMain
echo "saving log file"
echo \varepsiloncho $vmname`
echo `VBoxManage guestproperty get $vmname /VirtualBox/GuestInfo/Net/0/V4/IP`
vmip=`VBoxManage guestproperty get $vmname /VirtualBox/GuestInfo/Net/0/V4/IP |
    grep 'Value:' | sed 's poweroff
VBoxManage modifyvm $vmname --hda none
VBoxManage unregistervm $vmname --delete

# Destroy the HDD clone
zfs destroy -f $zfspath

exit 0
```

2 updateVMClient.sh

```
#!/bin/bash
# Update the hdd clone
echo ``removing snapshot''
zfs destroy -R zpool1/vbox/harddisks/vmclient@kopie

echo ``copying new .vdi''
cd ~/harddisks/vmclient
cp ~/harddisks/vm/vmclient.vdi .

echo ``creating snapshot''
zfs snapshot zpool1/vbox/harddisks/vmclient@kopie

exit 0
```