Backup with OpenNebula

Anh Vu Le Quy

The goal of this testing is to find a way, how to save and restore an online virtual machine, preferably during some work load. This all within an environment of OpenNebula.

OpenNebula provides us with basically two snapshot tools – `onevm disk-snapshot` and `onevm snapshot-create`. The first one simply copies a disk image of a virtual machine and can be executed on a running machine with no constrains. Since it is a relatively simple operation, we have some concerns about what happens with data which are not yet „hardwired“ into a disk (e.g. data in the memories). The second command should be able to save the state of a disk AND also the state of the virtual machine itself. That sounds more promising, but we try the first one as well.

If you do not want to read through all the approach, but just only see the successful method, you can skip directly here „Backuping with work load revert with QEMU-monitor „.

1. Preparations

At first we're going to create a VM template representing a backup machine. All the attributes will be the same as in the original template. The only thing we have to change is a name of its disk. The disk will be a copy of a VM we want to backup. That way, we can easily instantiate a backup VM using scripts written below.

- create template

```
onetemplate clone 0 "Snapshot Restore Test"
```

- edit DISK parameter of a template XML file

```
onetemplate update "Snapshot Restore Test"
DISK=[
  IMAGE="SnapTest",
  IMAGE_UNAME="oneadmin" ]
```

The snapshoted (cloned) disk image will have to cary a name „SnapTest“ in order to be used by this template.

2. Scripts used

2.1. Script 1: This script (*snapTest.sh*) backpus a disk of a given virtual machine and creates a new one from
the cloned image using the template previously prepared. It firstly copies the OS disk, names it appropriately, then waits, till the image becomes ready and finally instantiate a new VM.

```
#!/bin/sh
# Backup script

date;

onevm disk-snapshot $1 0 "SnapTest" --live
echo "Preparing new image..."

      while [ `oneimage list | grep SnapTest | awk '{print $9}'` = "lock" ]
      do
      i=0
      done

echo "VM instantiating..."
```

```
onetemplate instantiate "Snapshot Restore Test"
echo "VM created"
```

2.2. **Script 2:** The template above can only use the image named „SnapTest“ and there could be only one image named „SnapTest“ at a time. So if we want to make another backup test, we want be able to use the same name again. Therefore if we are done working with some backup VM, the following script (*snapTestClean.sh*) takes care of terminating it and freeing the name „SnapTest“ (by deleting the outdated image).

```
#!/bin/sh
# Cleaning script

onevm shutdown $1 --hard;
echo "Waiting for VM to shut down..."

      while ! [ -z `onevm list | grep Restor | awk '{print $6}'` ]
      do
      i=0
      done

echo "Deleting SnapTest image..."
oneimage delete "SnapTest";
echo "Cleaning completed."
```

2.3. **Script 3:** The point of our testing, is to find a way to backup a running AND working machine. Preservation of data the VM is currently working with is also an aspect worth considering. This script (*dateTest.sh*) simply logs recent time into a file, wich should simulate some work load. While this script is running, we will try to backup the VM, and investigate the log file whether there are data, or not.

```
#!/bin/sh
# Working load simulation script

echo "Writing test in progress..."

      while [ 1 ]
      do
      i=0
                  while [ $i != "10000" ]
                  do
            date +"%H:%M:%S:%N" >> log.txt
                  i=$((i+1))
            done
      rm log.txt
      done
```

2.4. **Script 4:** We want services running on machines to be avaiable nearly 24/7, therefore we are backuping live (with offline VM will everything be easier...). The time, when services are not avaible has to be the smallest. Script below (*durationTest.sh*) measures, how long does it také to finish a VM-state snapshot

```
#!/bin/sh
date
onevm snapshot-create $1
      while ! [ -z `onevm list | grep snap | awk '{print $6}'` ]
      do
      i=0
      done
date
```

3. Testing

3.1. Backuping with work load (`disk-snapshot --live`)

As mentioned above, OpenNebula has two tools for snapshoting. In this section, we will test the behavior of the first one - `onevm disk-snapshot --live` during memory load. Will the data in memory be saved, or just data in a disk?

Methodology:

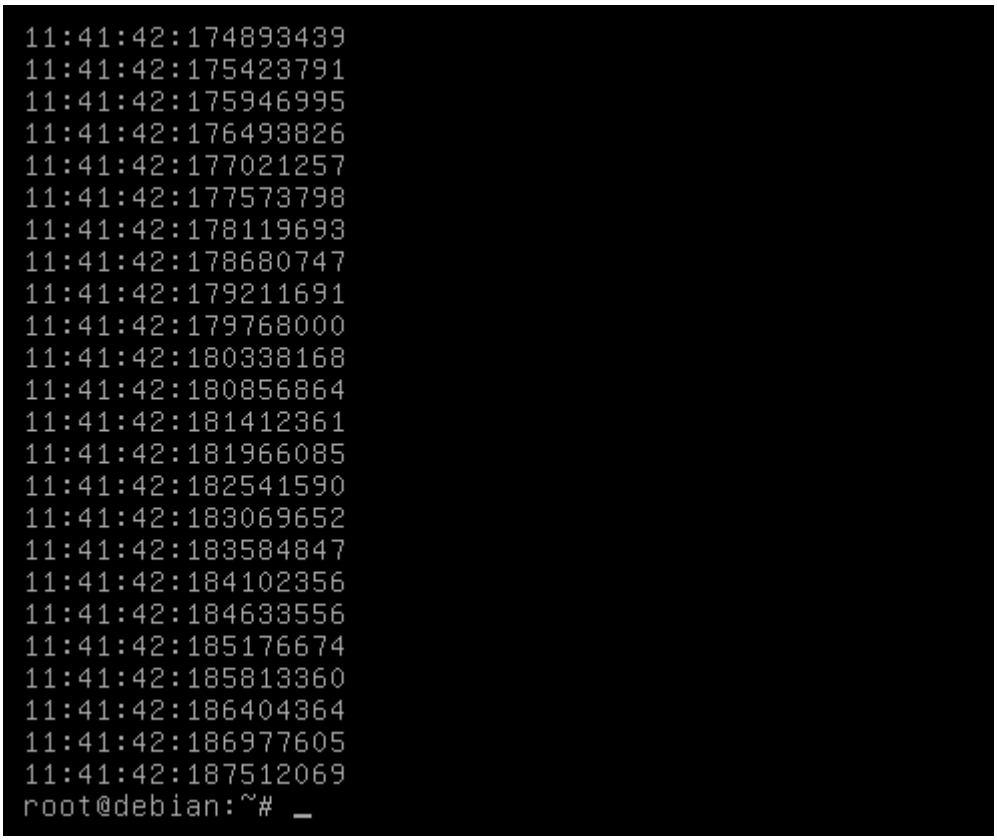
1. Run working simulation script (see above) on a reference VM
2. backup the VM using *Backup* script above
3. check *log.txt* file in a newly created VM

A virtual machine we want to backup is at the moment busy filling data into a *log.txt* file. Run the *snapTest.sh*...

```
Wed Aug 27 11:41:07 CEST 2014
Image ID: 54
Preparing new image...
VM instantiating...
VM ID: 50
VM created
```

What are the results? *log.txt* file on a backup VM was always empty, while the original log was populated with data. Anyway, see the screenshots below.

- Content of a *log.txt* file on the reference machine



- Content of a *log.txt* file on the backup machine (empty)

This test confirms our concerns, that only concrete disk data are preserved and the state of RAM or VM is not carried during this kind of snapshots. The `onevm disk-snapshot --live` snapshot does not satisfy our needs, however OpenNebula has the second snapshot command.

The `onevm snapshot-create` command is supposed to remember the state of a system as a whole, not only the disk. That includes the memories too and that is what we need – data from RAMs.

System snapshots are saved directly into the virtual machines's *.qcow2* image. So theoretically, if we copy the disk with `disk-snapshot –live` and instantiate a new machine, we should be able to get to the system snapshots on that new VM since it uses the same image as the original one. Then we can revert to the snapshot and successfully backup the machine. But get to these snapshots isn't as easy as it may seem. As you can see in next tests.

The system snapshots require to pause a machine in order to save its current state. Hence in the first place we shall measure the time when a virtual machine is in a paused state. For completeness – it was a 2GB VM image with Debian OS. The whole process lasted according to the *durationTest.sh* script...

```
oneadmin@backup1:~/datastores/0$ ./durationTest.sh 50
Wed Sep  3 00:03:39 CEST 2014
Wed Sep  3 00:15:01 CEST 2014
```

...about 12minutes. The VM was in a idle state. For a full OS in load, that would be certainly a lot more.

3.2. System snapshot preservation test (after `onevm disk-snapshot`)

Now we are going to test the approach suggested above. We will make a machine running on a disk copy and we will try to access the snapshots hidden somewhere on that image.

Methodology:

1. create a system snapshot of a reference VM
2. copy its disk and instantiate a new VM using CREATE script above
3. check for the snapshot in a newly created VM

At first, the system snapshot is created (of a machine with *ID 9* in this case)

```
oneadmin@backup1:~$ onevm snapshot-create 9
```

Then run the script to backup the VM

```
oneadmin@backup1:~$ /var/lib/one//datastores/0/snapTest.sh 9

Sat Aug 23 21:00:44 CEST 2014
Image ID: 44
Preparing new image...
./snapTest.sh: 8: [: =: unexpected operator
VM instantiating...
VM created
```

Let's take a look at current virtual machine list

```
oneadmin@backup1:~$ onevm list
    ID USER     GROUP    NAME            STAT UCPU    UMEM HOST             TIME
     8 oneadmin oneadmin Debian Install- runn    0    512M backup1     14d 00h12
     9 oneadmin oneadmin Debian Install- runn    0    512M backup1     14d 00h01
    36 oneadmin oneadmin Snapshot Restor runn    0      0K backup1      0d 00h02
```

You can see the original virtual machine with *ID 9* and its restored version with *ID 36* and name *„Snapshot Restore Test“* (the same as the template it was created from).

OpenNebula is the upper level of the virtualization. Let's take a look at the current situation in lower levels. Check through *virsh:*

```
virsh # list
 Id    Name                           State
----------------------------------------------------
 60    one-9                          running
 63    one-36                         running
```

The IDs differ from OpenNebula, but you can see the OpenNebula IDs are transfered into the name, so the machines are still distinguishable.In virsh, it is also possible to see the list of all snapshots associated with the machine. Firstly take a look at the original VM's snapshots.

```
virsh # snapshot-list 60
 Name                 Creation Time             State
------------------------------------------------------------
 1408820313           2014-08-23 20:58:33 +0200 running
```

The same with the backup machine...

```
virsh # snapshot-list 63
 Name                 Creation Time             State
------------------------------------------------------------
```

There is no snapshot listed. It probably got lost after backuping process. How disappointing...

Another level, however, seems to have a better memory, because as you see below, it is able to list all the snapshots in the VM's (respectively image's) history. *Qemu-img* saves the day...

```
oneadmin@backup1:~/datastores/0/36$ qemu-img snapshot -l disk.0
Snapshot list:
ID        TAG                 VM SIZE                DATE       VM CLOCK
4         1408615602             244M 2014-08-21 12:06:42  260:49:23.332
5         1408623515             248M 2014-08-21 14:18:35  262:31:55.415
6         1408789398             249M 2014-08-23 12:23:18  280:03:03.841
7         1408790650             120M 2014-08-23 12:44:10   00:04:07.650
8         1408820313             113M 2014-08-23 20:58:33   00:32:58.206
```

If you have a look at the *Date* column of the last snapshot, you will see it is the same as the date of the snapshot shown by virsh above.

You may wonder, what is the *disk.0* – each VM instantiated by OpenNebula has a folder in `/datastores/0/` directory. This folder is named according to the VM's ID (e.g. VM with *ID 9* would have a folder named *9*). Each folder stores a *disk.0* file, which is the OS image of the VM – the image is basically the VM itself. Every change is written to this file as well as all snapshots.

We can see now, that when we try to backup a VM, every snapshot ever made on it will be hidden. Unless *qemu-img* is used. Because it is the only tool able to see the snaphots, it will be the only one able to use those snaphots to revert the VM. But let's do some more testing.

3.3. System snapshot preservation test (after power off)

Methodology:

1. create a system snapshot
2. power off the VM and resume it again
3. check whether the snapshot is stil avaiable

The procedure is the same as above, only disk copy is replaced by powering off the machine

```
oneadmin@backup1:~$ onevm snapshot-create 8
oneadmin@backup1:~$ onevm show 8
      SNAPSHOTS
      ID         TIME NAME                                    HYPERVISOR_ID
      0  08/23  20:36 snapshot-0                               1408818981
```

List the snapshots through OpenNebula and then through *virsh*

```
virsh # snapshot-list 59
 Name                 Creation Time             State
------------------------------------------------------------
 1408818981           2014-08-23 20:36:21 +0200 running
```

...and of course through *qemu-img*

```
oneadmin@backup1:~/datastores/0/8$ qemu-img snapshot -l disk.0
Snapshot list:
ID        TAG                 VM SIZE                DATE       VM CLOCK
1         1408818981             123M 2014-08-23 20:36:21   00:19:57.301
```

Now try to poweroff the VM a see what happens

```
oneadmin@backup1:~$ onevm poweroff 8
oneadmin@backup1:~$ onevm resume 8
oneadmin@backup1:~$ onevm show 8
```

The `SNAPSHOTS` section in the listing is missing and because OpenNebula uses the *virsh* commands in its scripts, it is not surprising that *virsh* lost traces of the snaphots too.

```
virsh # list
 Id    Name                           State
----------------------------------------------------
 60    one-9                          running
 61    one-8                          running

virsh # snapshot-list 61
 Name                 Creation Time             State                  #
```

```
Snapshot disappeared as well
------------------------------------------------------------

oneadmin@backup1:~/datastores/0/8$ qemu-img snapshot -l disk.0
Snapshot list:
ID        TAG                 VM SIZE                DATE       VM CLOCK
1         1408818981             123M 2014-08-23 20:36:21   00:19:57.301
```

System snapshots provided by OpenNebula save a state of the disk and also the memories into the *.qcow2* file of the VM. That might be a drawback, since backuping will require uploading a the whole (mostly massive) image. At least the content of a memory should be saved...

3.4. Backuping with work load (snapshot-create) revert with *qemu-img*

From the test above we can see, that `disk-snapshot` does not preserve data during work load. Previous tests have also showed, that *qemu-img* can see system snapshots of a cloned disk image. So we want to know, if it is possible to obtain data in memories through it.

Methodology:

1. run working-load-simulation script above on a reference VM
2. system snapshot that VM
3. run *Backup* script to make a new VM
4. revert to the snapshot and check log.txt on the new VM

Logging script is now running on a virtual machine 9...

```
oneadmin@backup1:~$ onevm snapshot-create 9

oneadmin@backup1:~$ /var/lib/one//datastores/0/snapTest.sh 9
Sun Aug 24 13:40:06 CEST 2014
Image ID: 47
Preparing new image...
VM instantiating...
VM created

oneadmin@backup1:~$ qemu-img snapshot -l disk.0
Snapshot list:
ID        TAG                 VM SIZE                DATE       VM CLOCK
4         1408615602             244M 2014-08-21 12:06:42  260:49:23.332
5         1408623515             248M 2014-08-21 14:18:35  262:31:55.415
6         1408789398             249M 2014-08-23 12:23:18  280:03:03.841
7         1408790650             120M 2014-08-23 12:44:10   00:04:07.650
8         1408820313             113M 2014-08-23 20:58:33   00:32:58.206
9         1408823736             115M 2014-08-23 21:55:36   01:28:40.072
10        1408877157             145M 2014-08-24 12:45:57   14:24:55.013
11        1408879902             145M 2014-08-24 13:31:43   14:56:05.451
```

Power off the VM first, because following `snapshot --aply` command can be used only for offline guests. The last snapshot will be used.

```
oneadmin@backup1:~$ qemu-img snapshot -a 11 disk.0
```

```
Debian GNU/Linux 7 debian tty1

debian login: root
Password:
Last login: Sat Aug 23 12:49:14 CEST 2014 on tty1
Linux debian 3.2.0-4-amd64 #1 SMP Debian 3.2.60-1+deb7u3 x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
root@debian:~# cat log.txt
root@debian:~# _
```

The *log.txt* file is present, but empty.
Lets use OpenNebula `revert` command, to see, if we can successfuly return at least on a reference VM.

```
oneadmin@backup1:~$ onevm snapshot-revert 9 3
```

The *log.txt* file should look like this:

```
13:43:14:530172530
13:43:14:530610485
13:43:14:531039386
13:43:14:531470860
13:43:14:531901911
13:43:14:532332796
13:43:14:532764423
13:43:14:533193474
13:43:14:533626923
13:43:14:534069851
13:43:14:534603137
13:43:14:535084244
13:43:14:535559635
13:43:14:535986874
13:43:14:536415145
13:43:14:536841631
13:43:14:537298074
13:43:14:537752229
13:43:14:538183966
13:43:14:538615203
13:43:14:539044223
13:43:14:539473061
13:43:14:539903172
13:43:14:540331071
root@debian:~# _
```

From this test we can conclude, that recent data in memories are flushed using the *qemu-img* command, while OpenNebula can easily revert them using the same snapshot with no data loss.
The script for OpenNebula `snapshot-revert` command is following (only the relevant part):

```
virsh --connect $LIBVIRT_URI snapshot-revert --force $DOMAIN $NAME
```

We can see, that it is using *virsh* command `snapshot-revert --force.` That is bad news for us, because former tests have shown, that *virsh* looses all the memory of snapshots after poweroff or disk copy and therefore we can't use this feature for reverting a VM, which was created from an image copy.

The situation is now following: We are able to clone a machine we want to backup. The process however causes the data in memories to be lost. We therefore use a `system-snapshot` for RAM data saving. This checkpoint will be written on the image and therefore preserved after cloning. Unfortunately, OpenNebula and *virsh* don't see these snapshots on cloned machines and we then can't use their tools to revert virtual machine's state. Snapshots on cloned VMs can be seen by *qemu-img*, but we can not use them, since *qemu-img* tools are not able to handle memories.

This is an awkward situation, where one has an desired ability and cannot see snaphots and the second sees the snapshots but doesn't have the ability to revert them. We have to find somewhere else.

3.5. Preparation for backuping with work load and reverting with *QEMU-monitor*

Virsh allows to connect to a *QEMU-monitor* interface through `qemu-monitor-command --hmp <domain>  '<commands> [...]'`. This monitor offers a few features which might be useful for our case. Namely commands `savevm` and `loadvm`. Since QEMU remembers all taken snapshots of a given disk and *qemu-img* cannot give us what we need, lets examine this next hope:

Methodology:

1. snapshot a reference VM in a idle state
2. make some changes (e.g. delete some file) to determine a pre- and post-snapshot state
3. poweroff the VM and resume it again (*virsh* then forgets the snapshot)
4. revert it using *QEMU-monitor* commandline

At this point, steps 1. - 3. are completed...

```
virsh # qemu-monitor-command --hmp 89 'info snapshots'
ID        TAG                 VM SIZE                DATE       VM CLOCK
4         1408615602             244M 2014-08-21 12:06:42  260:49:23.332
5         1408623515             248M 2014-08-21 14:18:35  262:31:55.415
6         1408789398             249M 2014-08-23 12:23:18  280:03:03.841
7         1408790650             120M 2014-08-23 12:44:10   00:04:07.650
8         1408820313             113M 2014-08-23 20:58:33   00:32:58.206
9         1408823736             115M 2014-08-23 21:55:36   01:28:40.072
10        1408877157             145M 2014-08-24 12:45:57   14:24:55.013
11        1408879902             145M 2014-08-24 13:31:43   14:56:05.451
12        1408883181             145M 2014-08-24 14:26:21   15:43:29.607
13        1408895075             145M 2014-08-24 17:44:35   17:22:11.861
14        1408986062             119M 2014-08-25 19:01:02   00:01:23.944
```

We're going to use the last one.

```
virsh # qemu-monitor-command --hmp 89 'loadvm 14'

virsh #
virsh # resume 89
Domain 89 resumed
```

In all the tests, our VMs have stucked in the 'Paused' state and had to be resumed manually. However its filesystem remained intact a even in a correct state (a test file was present). That is a pretty possitive news. We can now proceed to testing this method with a working load on a VM.

3.6. Backuping with work load revert with *QEMU-monitor*

From the preparation for this test we can see, that (with some inconvenience) we are able to revert to the snapshot using a snapshot *virsh* no longer sees. But how will the monitor perform with some busy memories?

Methodology:

1. Run the work simulation script (see above)
2. Snapshot the loaded VM
3. run *Backup* script (see above) to get a new VM
4. revert this new VM to the snapshot via QEMU monitor

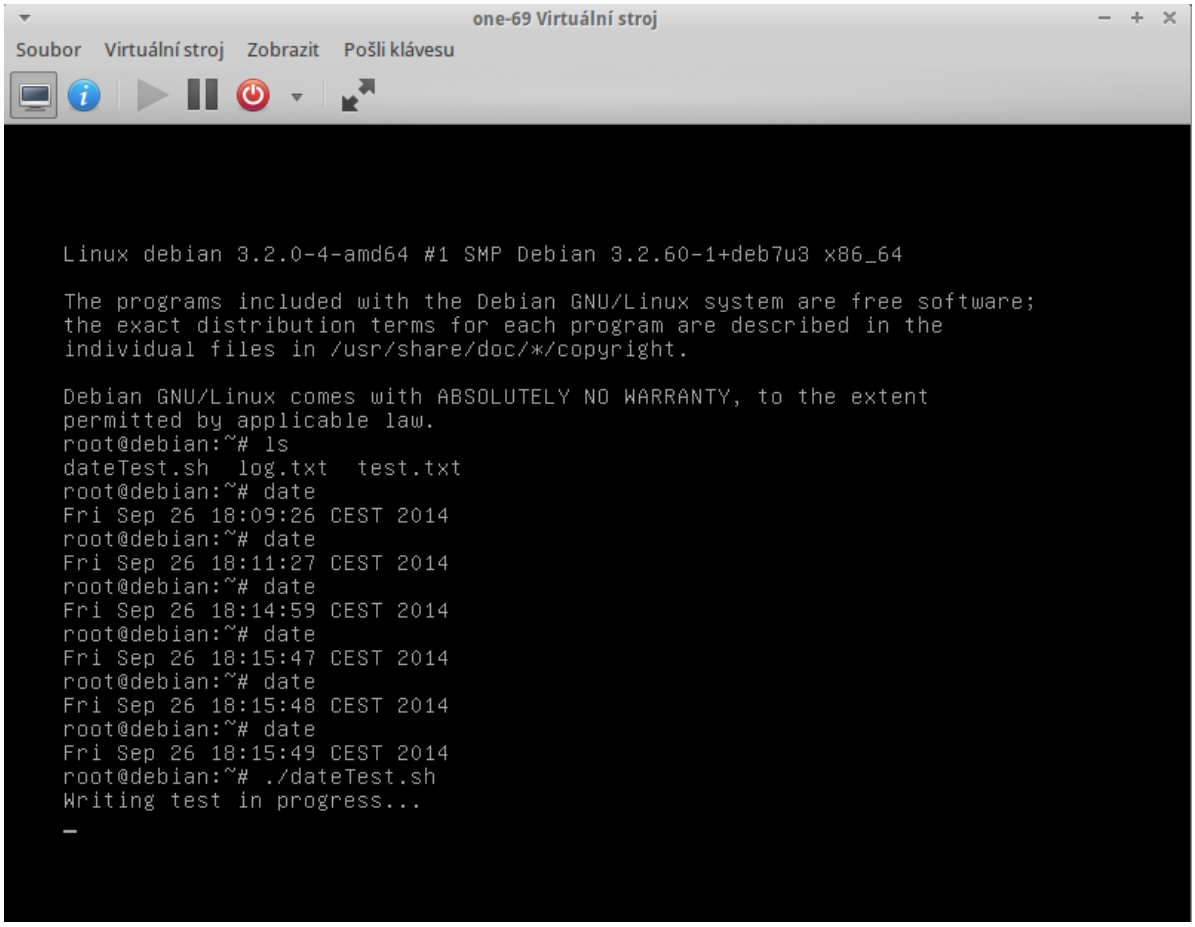
At this point, steps 1. and 2. are completed.

```
oneadmin@backup1:~/datastores/0$ ./snapTest.sh 8
Thu Oct  2 17:21:19 CEST 2014
Image ID: 71
Preparing new image...
VM instantiating...
VM ID: 69
VM created

virsh # qemu-monitor-command --hmp "one-69" 'info snapshots'
ID        TAG                 VM SIZE                DATE       VM CLOCK
1         1408818981             123M 2014-08-23 20:36:21   00:19:57.301
2         1409045206             119M 2014-08-26 11:26:46   00:18:28.026
3         snapTest               232M 2014-08-30 19:46:25  104:32:32.023
4         1409662660             137M 2014-09-02 14:57:40   17:44:03.692
5         1409663544             137M 2014-09-02 15:12:24   17:50:32.691
6         1411748161             162M 2014-09-26 18:16:01  339:17:31.021
```

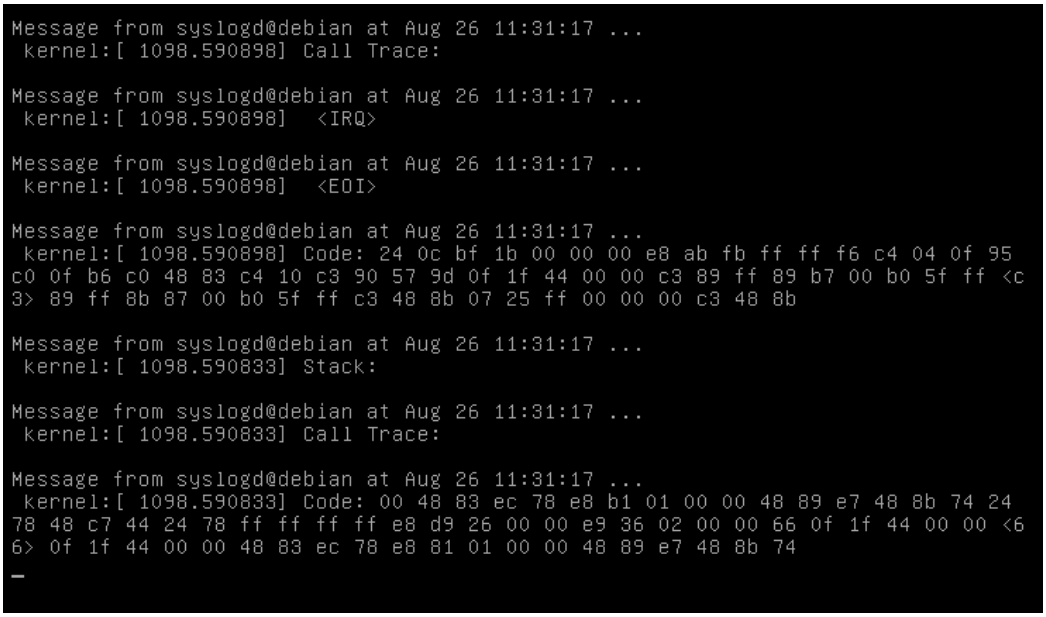
The last snapshot was taken while the load script was running. Revert to this state:

```
virsh # qemu-monitor-command --hmp "one-69" 'loadvm 6'

virsh #
virsh # resume 58
Domain 58 resumed
```

The state of the machine after resuming was very pleasing. The VM was allright even with the running scrip. Exactly as we wanted (note the „Writing test in progress“ line. This message can be seen in the script above):



We have finally found a method, that can backup a running VM and even preserve its memory. A little flaw is some error messages, that may occur. These messages could be sometimes seen after reverting:



This rarely happens after OpenNebula's `snapshot-revert` as well and it seems we can do nothing about it. In a simple OS like ours simply hit `Ctr+C` or `Enter` several times and a usual commandline appears. The question is, whether it will somehow damage the VM running on more complex, real-life OS...

3.7. Backup using virsh external snapshot

External snapshots provided by *virsh* can be very effective way to create backups. Just a brief reminder - after an external snapshots the original *.qcow2* image become read-only and the changes to a VM will be written to a new *.qcow2* image. The delta image is therefore very small compared to the original (now read-only) one. This is a great advantage, because backups won't consume much memory in a datastore. Moreover, if a chain of delta files become too long, it is possible to merge them into a singe active layer. External snapshots can be thanks to these abilities very powerful tool for VM backuping. But is this way compatible with OpenNebula?

Methodology:

1. external snapshot a VM with virsh
2. make some changes to distinguish the pre- and post-snapshot state (e.g. create a file)
3. power off and resume a VM
4. does OpenNebula know it has to read-form another file?

This long command basically makes a external snaphot of a machine 100. The snapshot will be named *„snapTest“* and stored in *„one8snap.qcow2“* file. Atomic option only ensures that it will either succeeds or fails without any influence of the image.

```
# virsh snapshot-create-as --domain 100 snapTest --disk-only --diskspec
vda,snapshot=external,file=/var/lib/one//datastores/0/one8snap.qcow2
--atomic
Domain snapshot snapTest created
```

If we list some details about the new file, we will notice its small size (200Kb). The backing (read-only) file is the well known 'disk.0' image.

```
oneadmin@backup1:~$ qemu-img info /var/lib/one//datastores/0/one8snap.qcow2
image: /var/lib/one//datastores/0/one8snap.qcow2
file format: qcow2
virtual size: 20G (21474836480 bytes)
disk size: 200K
cluster_size: 65536
backing file: /var/lib/one//datastores/0/8/disk.0
```

Now create a test file, power off and resume the VM...

A directory was found without the file created before powering off. It seems, that OpenNebula is blindly focused only on the 'disk.0' image and ignores this disk's baking files. We would have to upload the baking image each time the VM was going to be powered off, what is probably not the best way. In the other hand, with a non-stop running machine is this approach not only possible, but recommended.

3.8. Backup using virsh external snapshot (part 2 – swap names)

The test above tells us, that OpenNebula doesn't care about external snapshots a focuses only on a 'disk.0' file. We can try to trick it by giving it an external snapshot file camouflaged as 'disk.0'. If this succeeds, we would find a low bandwidth backup way compatible with OpenNebula.

Methodology:

1. external snapshot the disk
2. rename disk.0 to something else and name the external snapshot file as "disk.0"
3. make some changes (e.g. create file)

4. power off and resume the VM, check if the file is still there

```
oneadmin@backup1:~/datastores/0/8$ touch snapTest.qcow2

# virsh snapshot-create-as --domain 108 snapTest --disk-only --diskspec
vda,snapshot=external,file=/var/lib/one//datastores/0/8/snapTest.qcow2
--atomic
Domain snapshot snapTest created

oneadmin@backup1:~/datastores/0/8$ qemu-img info snapTest.qcow2
image: snapTest.qcow2
file format: qcow2
virtual size: 20G (21474836480 bytes)
disk size: 712K
cluster_size: 65536
backing file: /var/lib/one//datastores/0/8/disk.0
```

Note the 'backing file' line. Virsh requires to have this specificaly named backing image as a read-only disk. We therefore won't be able to rename the delta file to "disk.0", otherwise we would create a loop. The continuation fo the test prooves it, because we won't be able to boot the device.

What about renaming the original disk BEFORE snapshot, so the name 'disk.0' will be aviable? The methodology will be then as follows:

Methodology:

1. rename disk.0
2. external snapshot the disk with name 'disk.0'
3. make some changes (e.g. create a file)
4. power off and resume the VM, check if the file is still there

```
oneadmin@backup1:~/datastores/0/8$ mv disk.0 ./backupdisk.0
oneadmin@backup1:~/datastores/0/8$ touch snapTest.qcow2
```

The disk.0 name is now avaiable and the snapshot file prepared. Now the snapshot comes.

```
virsh # snapshot-create-as --domain 114 snapTest --disk-only --diskspec
vda,snapshot=external,file=/var/lib/one//datastores/0/8/snapTest.qcow2 –
atomic

error: internal error unable to execute QEMU command 'transaction': Could
not open '/var/lib/one//datastores/0/8/snapTest.qcow2'
```

Virsh could not access the *snapTest* file? But we had just created the file *snapTest.qcow2*. Why couldn't it reach that? Lets take a look into the file system:

```
oneadmin@backup1:~/datastores/0/8$ ls
backupdisk.0   deployment.11  deployment.15  deployment.3  deployment.7
deployment.0   deployment.12  deployment.16  deployment.4  deployment.8
deployment.1   deployment.13  deployment.17  deployment.5  deployment.9
deployment.10  deployment.14  deployment.2   deployment.6  disk.1
```

The 'disk.0' is absent as planed. But the file *snapTest.qcow2* we created is absent too. Try it one more time:

```
oneadmin@backup1:~/datastores/0/8$ touch snapTest.qcow2
oneadmin@backup1:~/datastores/0/8$ ls
backupdisk.0   deployment.12  deployment.17  deployment.6  snapTest.qcow2
```

```
deployment.0   deployment.13  deployment.2   deployment.7
deployment.1   deployment.14  deployment.3   deployment.8
deployment.10  deployment.15  deployment.4   deployment.9
deployment.11  deployment.16  deployment.5   disk.1

virsh # snapshot-create-as --domain 114 snapTest --disk-only --diskspec
vda,snapshot=external,file=/var/lib/one//datastores/0/8/snapTest.qcow2 –
atomic

error: internal error unable to execute QEMU command 'transaction': Could
not open '/var/lib/one//datastores/0/8/snapTest.qcow2'

oneadmin@backup1:~/datastores/0/8$ ls
backupdisk.0   deployment.11  deployment.15  deployment.3  deployment.7
deployment.0   deployment.12  deployment.16  deployment.4  deployment.8
deployment.1   deployment.13  deployment.17  deployment.5  deployment.9
deployment.10  deployment.14  deployment.2   deployment.6  disk.1
```

The same result. After changing the 'backupdisk.0' back to 'disk.0', everything works just fine. The renamimg process seems to have greater impact, then expected. And external-snapshoting with a offline guest is not possible because virsh can't reach them when poff/susp/stop.

3.9. Backup using virsh external snapshot (part 3 – swap names of persistent images)

Previous attempts with renaming the disk.0 file were made on non-persistent images. They can be used only on one machine at a time. In case of non-persistent images, the disk.0 file is just a link to a *.qcow2* file in `datastores/1/`. That could make it less reluctant to the renamimg (respectively „re-linking“). We will try to make the disk.0 point not to the persistent image in the datastore, but to the delta file made after a snapshot.

Methodology:

1. Snapshot a VM (VM-state snapshot)
2. Link the disk.0 to the new image
3. Create a file to determine the pre- and post-snapshot states
4. Turn off and on; see if the file is present

Below you can notice, that machine 59 is different from the ones you've seen before. The disk.0 file is a link to some read-only image

```
oneadmin@backup1:~/datastores/0/59$ ls -l
total 8
-rw-r--r-- 1 oneadmin oneadmin 728 Sep 15 16:26 deployment.0
-rw-r--r-- 1 oneadmin oneadmin 728 Sep 15 16:28 deployment.1
lrwxrwxrwx 1 oneadmin oneadmin  58 Sep 15 16:26 disk.0 ->
/var/lib/one/datastores/1/ed5d4f7f7a4821f23236489a1f4dadd1
```

The snapTest file is created a now we can proceed to the snapshoting

```
virsh # snapshot-create-as --domain 25 snapTest --disk-only --diskspec
vda,snapshot=external,file=/var/lib/one//datastores/0/59/snapTest.qcow2 –
atomic
Domain snapshot snapTest created
```

Here comes the re-linking part. We have basically two options. We either re-link the disk.0 with the guest online, or after having him powered-off. Lets examine both options.

- **Re-link while guest is online**

```
oneadmin@backup1:~/datastores/0/59$ rm disk.0
oneadmin@backup1:~/datastores/0/59$ ln -s ./snapTest.qcow2 ./disk.0

oneadmin@backup1:~/datastores/0/59$ ls -l
lrwxrwxrwx 1 oneadmin oneadmin      16 Sep 17 16:45 disk.0 ->
./snapTest.qcow2
-rw-r--r-- 1 oneadmin oneadmin 2686976 Sep 17 16:44 snapTest.qcow2
```

  Then we powered-off the machine and weren't able to resume it - the VM wasn't responding. After linking the „correct“ file, the machine starts just fine.

- **Re-link while guest is offline**
  The procedure is almost the same, only we first power-off the machine and then link the disk.0 to *snapTest.qcow2*. That didn't work either.

Pointing to a external snaphots instead of a original image isn't the solution. We assume, that for a successful launch both read-only image and external snapshot are needed, not only the snapshot. OpenNebula then thinks it deals with a full-fledged OS image, what is probably the reason the machine fails to start.

3.10. Non-opennebulian attemps for backup - XML file export

Export of a VM's XML file can be used for creating a new one. Will the snapshot survive' this process?

Methodology:

1. Snapshot a VM (VM-state snapshot)
2. Export a XML file, modify the unique attributes and instantiate a new VM
3. Check for the snapshot

The `dumpxml` command will list the XML description of a given machine to the standart output. We will override this output into a new XML file.

```
oneadmin@backup1:~$ virsh -c qemu:///system dumpxml one-9 >
/var/lib/one//datastores/0/VMtest.xml
```

Change the NAME and ID attributes, obtain a new UUID. They have to be unique. Attributes has to updated manually in the XML file. The rest items, including the image source file ('disk.0' of the original machine), stays the same.

```
oneadmin@backup1:~$ cat /proc/sys/kernel/random/uuid
195b81ec-b622-4f5a-9f86-0ef04cdda3b7
```

Now we are ready to create a new VM from the XML description file.

```
oneadmin@backup1:~$ virsh -c qemu:///system create
/var/lib/one//datastores/0/VMtest.xml
Unable to read from monitor: connection reset by peer
```

Error occured because we're trying to run 2 different VMs on the same disk image. The original one has to be therefore offline. But all of the offline states (susp/paus/poff) result in a snapshot loss. This can be fixed by the approach described above with *QEMU-monitor*, but there are more simple ways then manualy change a XML attributes...