Resizing NMIS VMs

It **is always advisable to make a backup of the target VM first**, ensuring you can recover your original VM should things go wrong !

Introduction

Generally you should only be needing to resize the partiton at the **/data** mountpoint on an NMIS VM as this partition contains the following directories:

- nmis9/
  - database/
  - var/
  - backups
- omk/
  - var/
- mongo/
  - (**mongo/** is the directory set as the **storage.dbPath** set in **/etc/mongod.conf**)

If it should become necessary to expand the storage space for the partition at the **/data** mountpoint, the following set of instructions should help you to perform that change with minimal NMIS downtime.

First, determine whether the NMIS VMs' partitions are using Logical Volume Manager (LVM)

First, determine whether the NMIS VM is using LVM:

If the **sudo lsblk** command produces partition entries in the TYPE column of type **lvm**, then that partition is using LVM.

```
sudo lsblk

NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda      8:0    0  120G  0 disk
sda1   8:1    0    1G  0 part /boot
sda2   8:2    0   15G  0 part /
sda3   8:3    0    1G  0 part [SWAP]
sda4   8:4    0  103G  0 part /var
sdb      8:16   0  120G  0 disk
sdb1   8:17   0  120G  0 part /data
```

In the example command above, using a recent release of the NMIS VM that does not use LVM, we have **disks /dev/sda (disk 1) and /dev/sdb (disk 2)**.

Disk 1 (/dev/sda) has partitions /dev/sda1 to /dev/sda4 and all partitions are TYPE **part**
Disk 2 (/dev/sdb) has partition /dev/sdb1 of TYPE **part**

If your NMIS VM is using partition of type **lvm** for partition at mountpoint **/data**, then proceed to the paragraph further below **NMIS VMs' using Logical Voume Manager (LVM)**
Otherwise, continue with the next paragraph **NMIS VMs' using Traditional Disk Partitions**

NMIS VMs' using Traditional Disk Partitions

First Step, determine the current state

```
df -h

Filesystem      Size  Used Avail Use% Mounted on
...
/dev/sdb1        40G  8.5G   29G  23% /data
...
```

df -h command returns this information for /data partition:

Partition: **/dev/sdb1**
Partition Size: **40G**
Partition Used: 8.5G
Partition Available: 29G
Partition Used %: 23%
Partition Mount Point: /data

It is recommended that you rerun the **df -h** command at the end to verify that the resizing has worked.

Second Step, Resize Storage Hardware

At this point it'll be simplest to open your vmware client, **resize the VM's disk 2 to the desired new size and then reboot the VM** to make it pick up the size change.

If you would like to avoid any downtime and not reboot, that's also possible if your virtualisation system allows adding or resizing disks "on the go": Virtualbox doesn't allow that, Vmware does. Please note that Vmware doesn't let you resize disks if there are any snapshots present; if that is the case, then these snapshots must be removed before the resize can occur.

Third Step, Informing the OS and resizing the file system

Install **growpart**:

```
# centos|rhel
sudo yum update
sudo yum install -y cloud-utils-growpart
# debian|ubuntu
sudo apt update
sudo apt install -y cloud-guest-utils
```

Grow the partition at the **/data** mountpoint, which we now know from the **df -h** commands above is **/dev/sdb1**:

```
# note there is a space between '/dev/sdb' and '1':
sudo growpart /dev/sdb 1

CHANGED: partition=1 start=2048 old: size=83884032 end=83886080 new: size=251656159 end=251658207
```

Resize the filesystem, which will take a bit of time and eventuallly tell you that it has resized the file system for the new extended disk size:

```
# note there is NOT a space between '/dev/sdb' and '1':
sudo resize2fs /dev/sdb1

resize2fs 1.42.9 (28-Dec-2013)
Filesystem at /dev/sdb1 is mounted on /data; on-line resizing required
old_desc_blocks = 5, new_desc_blocks = 15
The filesystem on /dev/sdb1 is now 31457019 blocks long.
```

Finally check that /dev/sdb1 has been resized as expected:

```
df -h

Filesystem      Size  Used Avail Use% Mounted on
...
/dev/sdb1       118G  8.5G  105G   8% /data
...
```

Moving and Resizing partitions /dev/sda1 to /dev/sda4 as needed where NMIS VM is using Traditional Partitions

This should not be necessary with regards to the NMIS VM and requires more thought, effort and skill to achieve.

For example, one could consider adding additional disks to the NMIS VM and moving the /home directory and|or any other directory consuming huge disk space to its own mountpoint on the additional disks as an alternative to growing partitions on disk 1 (/dev/sda).

One should also keep in mind that moving or resizing the /boot partition [fortunately at partition 1 on disk 1 (/dev/sda1)] can cause the VM not to boot afterwards.

Since disk 1 (/dev/sda) has more than 1 partition, GParted is probably a useful tool for this task:
https://gparted.org/display-doc.php%3Fname%3Dmoving-space-between-partitions

NMIS VMs' using Logical Volume Manager (LVM)

The resizing procedure is quite simple, for size increases at least. The two required steps are:

- adding extra storage 'hardware' to the VM
- informing the operating system of the extra storage, attaching the storage and resizing the active file systems.

This is possible because the Opmantek Virtual appliance makes use of Linux's excellent LVM support (Logical Volume Management, described in more details here).

First Step, determine the current state

login as as root to the VM and run

```
pvs
```

which will show you something like this

```
/dev/sdb   vg_nmis64_data  lvm2 a-   100.00g    0
```

meaning 100gb are allocated and vg_nmis64_data is using it. Note that on some older VMs the volume group is called vg_data instead; the resizing process can be performed as long as you remember to change the volume group name in the command invocations.

Next run

```
lvs
```

and it'll tell you

```
lv_data vg_nmis64_data -wi-ao 100.00g
```

that the lv_data logical volume is using all of vg_nmis64_data's space, and a final

```
df -h
```

will tell you that

```
 /dev/mapper/vg_nmis64_data-lv_data 99G 39G 56G 42% /data
```

 /data is the filesystem on top of the logical volume, which has a size of 99GB (plus spare change), of which 39BG are used. It is recommended that you rerun these commands at the end to verify that the resizing has worked.

Second Step, Resize Storage Hardware

At this point it'll be simplest to open your vmware client, **resize the VM's disk 2 to the desired new size and then reboot the VM** to make it pick up the size change.

If you would like to avoid any downtime and not reboot, that's also possible if your virtualisation system allows adding or resizing disks "on the go": Virtualbox doesn't allow that, Vmware does. Please note that Vmware doesn't let you resize disks if there are any snapshots present; if that is the case, then these snapshots must be removed before the resize can occur.

In many cases it will be simpler to add another 'disk' to the system rather than resizing either of the existing 'disks'.

Third Step, Informing the OS and resizing the file system

When the VM boots the newly resized disk 2 (aka /dev/sdb) will be detected but volume group and logical volume still need to be told about the change of 'hardware'.

If you added a new disk

If you did add disks instead of resizing existing ones, the OS should have picked them up dynamically and they should show up as /dev/sdc, sdd etc. In this case they must be registered as physical volumes (optionally after partitioning), and added to the volume group.

To do so, login as root first. Then check under what id your new disk has shown up, and add it as a whole as a new physical volume:

```
cat /proc/scsi/scsi
...
Host: scsi2 Channel: 00 Id: 01 Lun:  # the '2' indicates /dev/sdc is the device file
# this marks the whole disk as physical volume
pvcreate /dev/sdc
# this adds the pv to the volume group
vgextend vg_nmis64_data /dev/sdc
```

If you resized an existing disk that was partitioned (e.g. the FIRST harddisk):

The resized disk will have grown, but the partition table on it won't reflect the new size yet, and thus the extra space needs claiming before it can be accessed.

To do that, login as root, then add a new partition to the existing disk, make it use all the new space and give it type 8E (= Linux LVM), reboot to ensure the partition table is actually reread (this doesn't always work on-the-go); when that is done, create a new physical volume on that extra partition.

```
# assuming the modified disk is the first one, ie. /dev/sda
cfdisk /dev/sda # but you may prefer to use fdisk or gdisk instead
#...go through dialog to claim the extra space, let's assume we create /dev/sda4
reboot
pvcreate /dev/sda4
```

If you resized an existing disk that was used without partitions, just as phyiscal LVM volume (e.g. 2nd or 3rd harddisk):

In this case we just tell the LVM subsystem that there's now more space available in a particular physical volume. Login as root, then run

```
# assuming that the second disk was resized
pvresize /dev/sdb
```

which will tell you that sdb is now as large as the disk2 size you configured in your virtualisation tool.

New or resized disk

Finally we're ready to add the new space to the logical volume we need it in; To do that, login as root and run

```
lvextend -l 100%VG /dev/vg_nmis64_data/lv_data
```

to perform the resizing for the logical volume, follwed by

```
resize2fs /dev/vg_nmis64_data/lv_data
```

which will take a bit of time and eventuallly tell you that it has resized the file system for the new extended disk size. A final

```
df -h
```

should show that /data is now larger than before.