Virtualbox

Issues

Discard machine "starting" state

```
VBoxManage list vms
VBoxManage discardstate aab57b2d-566b-4d63-bca1-2a8ae74cd213

VBoxManage: error: The machine '...' is already locked for a session (or
being unlocked)
VBoxManage: error: Details: code VBOX_E_INVALID_OBJECT_STATE (0x80bb0007),
component MachineWrap, interface IMachine, callee nsISupports
VBoxManage: error: Context: "LockMachine(a->session, LockType_Write)" at
line 1247 of file VBoxManageMisc.cpp

rm /tmp/.vbox-john-ipc/lock
```

Edit .vbox file and remove `aborted="true"`.

Error -610 in supR3HardenedMainInitRuntime!

cannot run virtualbox on ubuntu 16.04

Windows 10 guest - RDP login freeze

Introduced in Windows 10 version 1903 Solution: deativate the WDDM-Display Driver by group policy.

```
Local Computer Policy
 -> Computer Configuration
  -> Administrative Templates
   -> Windows Components
    -> Remote Desktop Service
     -> Remote Desktop Session Host
      -> Remote Session Environment
       -> Use WDDM graphics display driver for Remote Desktop Connections
```

Tips & Tricks

Virtualization in VBox

TAG: nested virtualization VBox doesn't support nester virtualization.

Boot from USB

There is no option to boot from USB. You need to use helper bootloader. Download plop and extract ISO from zip.

optimize for Intel

```
VBoxManage modifyvm Windows --vtxvpid on --largepages on
```

Autostarting

https://askubuntu.com/questions/57220/start-vboxheadless-vm-at-startup

```
$ VBoxManage modifyvm <vmname> --autostart-enabled on
$ VBoxManage modifyvm <vmname> --autostop-type acpishutdown
```

Systemd based way

Classic way

There is a script `/usr/lib/virtualbox/vboxautostart-service.sh` which is called by `/lib/systemd/system/vboxautostart-service.service` Script is calling binary `VBoxAutostart`. Script is sourcing following configration files:

- `/etc/vbox/vbox.cfg`
- `/etc/default/virtualbox`

And using following variables:

- `$VBOXAUTOSTART_DB`
- `$VBOXAUTOSTART_CONFIG`

```
systemctl enable vboxautostart-service.service
```

Single service per VM

etc/systemd/system/vboxvmservice@.service

```
[Unit]
Description=VBox Virtual Machine %i Service
Requires=systemd-modules-load.service
After=systemd-modules-load.service

[Service]
```

```
User=user <-- modify
Group=vboxusers
ExecStart=/usr/bin/VBoxHeadless -s %i
ExecStop=/usr/bin/VBoxManage controlvm %i savestate
TimeoutSec=5min
TimeoutStopSec=5min


[Install]
WantedBy=multi-user.target
```

```
systemctl enable vboxvmservice@vm_name.service
systemctl start vboxvmservice@vm_name.service
```

Unlock "Guru Meditation"

Find PID of VirtualBox process and kill it.

Compacting VDIs

On guest system fill all empty space with zeros:

- Windows guest:
  - use “sdelete -z” (download from: SDelete). Make sure you are using sdelete version 1.6 or 2.02. Version 2 is buggy and can stuck at 100%.
  - zero swap file at shutdown. Modify windows reg:

zeroswap.reg

```
Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session
Manager\Memory Management]
"ClearPageFileAtShutdown"=dword:00000001
```

- Start –> secpol.msc –> Local Policies –> Security Options
  - “Shutdown: Clear virtual memory pagefile entry”
- Linux guest:

```
dd if=/dev/zero of=/bigfile; sync; rm -f /bigfile
```

Shutdown guest system and compact .vdi image:

- Windows:

```
c:\Program Files\Oracle\VirtualBox\VBoxManage.exe" modifyhd --compact
```

```
mydisk.vdi
```

- Linux:

```
VBoxManage modifyhd --compact mydisk.vdi
```

Resizing VDIs

To resize disc image (size is in MB!):

- Windows Host:

```
c:\Program Files\Oracle\VirtualBox\VBoxManage.exe" modifyhd --resize
15000 mydisk.vdi
```

- Linux Host:

```
vboxmanage modifyhd hdd_c.vdi --resize 20000
```

Resizing VDIs with snapshot is tricky:

> You can resize a VM which has snapshots, but you must resize the right thing.
> Resizing the first snapshot (more accurately the base image) only affects that one. It doesn't magically change anything else.
> So what you need to do is to identify the diff image which is associated with “Current State”, i.e. what is attached to the VM. This image needs to be resized.

To resize all snapshoots from Linux:

```
for x in Snapshots/*.vdi; do vboxmanage modifyhd $x --resize 20000; done
```

Enable SSD and trim

When TRIM is issued to vdi disc, disc size is reduced. It is very slow but it seems to be only one method to compact snapshots' VDIs.

Enabling from commandline:

```
VBoxManage storageattach "Debian clean" --storagectl "SATA" --port 0 --
discard on --nonrotational on
VBoxManage storageattach "Debian clean" --storagectl "SATA" --port 1 --
discard on --nonrotational on
```

or manually in config file (but can be overwritten by Virtualbox manager):

machine.vbox

```
<AttachedDevice nonrotational="true" discard="true" type="HardDisk"
```

```
port="0" device="0">
   <Image uuid="{fdd4f1a4-36ff-4944-aee8-e35b5503e87c}"/>
</AttachedDevice>
<AttachedDevice nonrotational="true" discard="true" type="HardDisk"
port="0" device="1">
   <Image uuid="{95baed2d-0aa7-437d-9bff-534af4036ce3}"/>
</AttachedDevice>
```

NOTE: order of XML attribs are importand. Add `discard="true"` after "nonrotational="true"''

Additionally increase SATA timeout:

```
echo -n 180 > /sys/block/sda/device/timeout
echo -n 180 > /sys/block/sdb/device/timeout
echo -n 180 > /sys/block/sdc/device/timeout
```

Disabling TRIM support from commandline:

```
VBoxManage storageattach "Windows10" --storagectl "SATA" --port 0 --discard
off
```

NOTE: Enabling TRIM support will kill your rotational drive :). Every data delete will cause VDI image to trim to its data size (like compacting) and it is very slow!

To disable TRIM in Windows10, run administator shell (WIN+X) and:

```
fsutil behavior query disabledeletenotify
fsutil behavior set DisableDeleteNotify 1
```

Mount shared folder

/etc/fstab

```
sharedfolder        /mnt/myfolder    vboxsf
noauto,uid=1000,gid=1000    0    0
```

change UUID of VDI

```
VBoxManage internalcommands sethduuid virtualdisk.vdi
```

convert RAW image to VDI

Convert raw disc image (like iso,bin, raw) into vdi:

```
# from RAW to VDI
VBoxManage convertdd bootable.bin testdisc.vdi

# From VDI to RAW
VBoxManage convertdd bootable.vdi testdisc.bin
```

draft

Add your user to following groups:

- disk
- vboxuser

To access real hard drive from VirtualBox, it is needed to crate raw VMDK:

```
VBoxManage internalcommands createrawvmdk -filename ./raw_dev_sdc.vmdk -
rawdisk /dev/sdc
```

It is possible to emulate SSD drives and emulate TRIM support:

```
VBoxManage storageattach My_Linux_VM --storagectl "SATA" --port 3 --discard
on --nonrotational on --type hdd
VBoxManage modifyhd raw_dev_sdc.vmdk settype writethrough
```

REMARK: Enabling virtual disc to be detected as SSD disc has nothing to real SSD installed in host system. With TRIM support enabled the dynamic VDI file is shrink when TRIM command is issued to controller.

Windows XP

change IDE to AHCI

First install drivers. Do not touch existing IDE controllers.Add new AHCI controller with some temporary disc.

- Start Windows.
- Download Intel ICH8 driver (INF Update Utility - it work with driver 8.9.0.1023 from Intel-Sata-Matrix-Manager-IATA89ENU.exe) and install it.
- Shutdown Windows
- Edit virtual machine and detach VDI from IDE controller and attach to SATA controller.
- Start Windows from AHCI

change HAL to multiprocessor

```
rundll32 syssetup,SetupInfObjectInstallAction ACPIAPIC_MP_HAL 128
%windir%\inf\hal.inf
```

move real machine into vbox

On Windows XP - create new hardware profile:

- Unordered List ItemStart–>Control Panel–>System
- on hardware tab choose “Hardware Profiles”
- Copy current profile to new one e.g. Hardware
- Rename current profile to e.g. VirtualBox

On Windows XP - change IDE controller to generic one:

- Device Manager–>IDE Controler–>Update Driver
- Select Standard Dual channel PCI IDE controller

NTFS compression

VDI was compacted. VDI files with +c attribute (BTRFS compression). Tests on C:\WINDOWS folder. Windows shows 2,46 GB data and 1.65GB on disc.

- VDI size with C attribute: 9.0 GB
  - BTRFS subvolume data 26.96GB
- VDI size after removing C attribute: 10.0 GB
- VDI size after zeroing free space and compacting: 9.9
  - BTRFS subvolume data 27,80GB
  - BTRFS subvolume data 26,97 GB - after recompression with ZLIB

2nd test:

- NTFS Compressed folder 5,86 GB / 2,72 GB, VDI = 9,9 GB, BTRFS = 26.97 GB
- NTFS Decompressed folder 5,86 GB, VDI = 11 GB, BTRFS = 32,30 GB
- NTFS Decompressed folder 5,86 GB, VDI = 9.3 GB, BTRFS = 32,31 GB + VDI compacted
- NTFS Decompressed folder 5,86 GB, VDI = 9.3 GB, BTRFS = 32,26 GB + BTRFS ZLIB recompressed
- NTFS Compressed folder 2.72 GB, VDI = 9.3 GB, BTRFS = 35,49 GB + NTFS compression
- NTFS Compressed folder 2.72 GB, VDI = 9.3 GB, BTRFS = 38,68 GB + NTFS zero free space
- NTFS Compressed folder 2.72 GB, VDI = 9.3 GB, BTRFS = 32,29 GB + BTRFS ZLIB recompressed
- NTFS Compressed folder 2.72 GB, VDI = 9.3 GB, BTRFS = 27,12 GB + BTRFS ZLIB recompressed mount compress-force=zlib
- NTFS Uncompressed folder 5.86 GB, VDI = 11 GB, BTRFS = 26,47 GB + BTRFS ZLIB recompressed mount compress-force=zlib

3rd test:

- Uncompressed compacted windows, 6.0 GB, BTRFS recompressed 32,09 GB, previously copied from 2nd disc with compress-force
- Compressed compacted windows, 6.0 GB, BTRFS recompressed 32,65 GB, mount compress-force=zlib

4rd test:

- 39GB usage - Windows C drive compressed (C attribute)

- 42GB usage - Windows C drive uncompressed (no C attribute)
- 33GB usage - Windows C drive uncompressed + vdi file manually compressed by BTRFS (forced by btrfs defrag)

Conclusions

Better is to use BTRFS compression - periodically recompress .vdi file.

- (+) NTFS compression is not so bad. It is still a bit worse than BTRFS ZLIB.
- (-) BTRFS compression only works for whole data if mounted with **compress-force=zlib**.
- (+) VDI file on BTRFS can be compressed using BTRFS compression when machine is online.
- (-) NTFS compression is using guest CPU. Where more power is often available at host (many BTRFS threads)