

Virtualization

Running virtual machines on compute nodes

Introduction

There are situations when Anselm's environment is not suitable for user needs.

- Application requires different operating system (e.g Windows), application is not available for Linux
- Application requires different versions of base system libraries and tools
- Application requires specific setup (installation, configuration) of complex software stack
- Application requires privileged access to operating system
- ... and combinations of above cases

We offer solution for these cases - **virtualization**. Anselm's environment gives the possibility to run virtual machines on compute nodes. Users can create their own images of operating system with specific software stack and run instances of these images as virtual machines on compute nodes. Run of virtual machines is provided by standard mechanism of Resource Allocation and Job Execution.

Solution is based on QEMU-KVM software stack and provides hardware-assisted x86 virtualization.

Limitations

Anselm's infrastructure was not designed for virtualization. Anselm's environment is not intended primary for virtualization, compute nodes, storages and all infrastructure of Anselm is intended and optimized for running HPC jobs, this implies suboptimal configuration of virtualization and limitations.

Anselm's virtualization does not provide performance and all features of native environment. There is significant performance hit (degradation) in I/O performance (storage, network). Anselm's virtualization is not suitable for I/O (disk, network) intensive workloads.

Virtualization has also some drawbacks, it is not so easy to setup efficient solution.

Solution described in chapter HOWTO is suitable for single node tasks, does not introduce virtual machine clustering.

Please consider virtualization as last resort solution for your needs.

Please consult use of virtualization with IT4Innovation's support.

For running Windows application (when source code and Linux native application are not available) consider use of Wine, Windows compatibility layer. Many Windows applications can be run using Wine with less effort and better performance than when using virtualization.

Licensing

IT4Innovations does not provide any licenses for operating systems and software of virtual machines. Users are (> in accordance with Acceptable use policy document) fully responsible for licensing all software running in virtual machines on Anselm. Be aware of complex conditions of licensing software in virtual environments.

Users are responsible for licensing OS e.g. MS Windows and all software running in their virtual machines.

HOWTO

Virtual Machine Job Workflow

We propose this job workflow:

Workflow](virtualization-job-workflow “Virtualization Job Workflow”)

Our recommended solution is that job script creates distinct shared job directory, which makes a central point for data exchange between Anselm’s environment, compute node (host) (e.g HOME, SCRATCH, local scratch and other local or cluster filesystems) and virtual machine (guest). Job script links or copies input data and instructions what to do (run script) for virtual machine to job directory and virtual machine process input data according instructions in job directory and store output back to job directory. We recommend, that virtual machine is running in so called snapshot mode, image is immutable - image does not change, so one image can be used for many concurrent jobs.

Procedure

1. Prepare image of your virtual machine
2. Optimize image of your virtual machine for Anselm’s virtualization
3. Modify your image for running jobs
4. Create job script for executing virtual machine
5. Run jobs

Prepare image of your virtual machine

You can either use your existing image or create new image from scratch.

QEMU currently supports these image types or formats:

- raw
- cloop
- cow
- qcow
- qcow2
- vmdk - VMware 3 & 4, or 6 image format, for exchanging images with that product
- vdi - VirtualBox 1.1 compatible image format, for exchanging images with VirtualBox.

You can convert your existing image using `qemu-img convert` command. Supported formats of this command are: `blkdebug` `blkverify` `bochs` `cloop` `cow` `dmg` `file` `ftp` `ftps` `host_cdrom` `host_device` `host_floppy` `http` `https` `nbd` `parallels` `qcow` `qcow2` `qed` `raw` `sheepdog` `tftp` `vdi` `vhdx` `vmdk` `vpc` `vvfat`.

We recommend using advanced QEMU native image format `qcow2`.

More about QEMU Images

Optimize image of your virtual machine

Use virtio devices (for disk/drive and network adapter) and install virtio drivers (paravirtualized drivers) into virtual machine. There is significant performance gain when using virtio drivers. For more information see Virtio Linux and Virtio Windows.

Disable all unnecessary services and tasks. Restrict all unnecessary operating system operations.

Remove all unnecessary software and files.

Remove all paging space, swap files, partitions, etc.

Shrink your image. (It is recommended to zero all free space and reconvert image using `qemu-img`.)

Modify your image for running jobs

Your image should run some kind of operating system startup script. Startup script should run application and when application exits run shutdown or quit virtual machine.

We recommend, that startup script

maps Job Directory from host (from compute node) runs script (we call it “run script”) from Job Directory and waits for application’s exit - for management purposes if run script does not exist wait for some time period (few minutes)

shutdowns/quits OS For Windows operating systems we suggest using Local Group Policy Startup script, for Linux operating systems rc.local, runlevel init script or similar service.

Example startup script for Windows virtual machine:

```
@echo off
set LOG=c:startup.log
set MAPDRIVE=z:
set SCRIPT=%MAPDRIVE%run.bat
set TIMEOUT=300

echo %DATE% %TIME% Running startup script>%LOG%

rem Mount share
echo %DATE% %TIME% Mounting shared drive>%LOG%
net use z: 10.0.2.4qemu >%LOG% 2>&1
dir z: >%LOG% 2>&1
echo. >%LOG%

if exist %MAPDRIVE% (
    echo %DATE% %TIME% The drive "%MAPDRIVE%" exists>%LOG%

    if exist %SCRIPT% (
        echo %DATE% %TIME% The script file "%SCRIPT%"exists>%LOG%
        echo %DATE% %TIME% Running script %SCRIPT%>%LOG%
        set TIMEOUT=0
        call %SCRIPT%
    ) else (
        echo %DATE% %TIME% The script file "%SCRIPT%"does not exist>%LOG%
    )

) else (
    echo %DATE% %TIME% The drive "%MAPDRIVE%" does not exist>%LOG%
)
echo. >%LOG%

timeout /T %TIMEOUT%

echo %DATE% %TIME% Shut down>%LOG%
shutdown /s /t 0
```

Example startup script maps shared job script as drive z: and looks for run script called run.bat. If run script is found it is run else wait for 5 minutes, then shutdown virtual machine.

Create job script for executing virtual machine

Create job script according recommended

Virtual Machine Job Workflow.

Example job for Windows virtual machine:

```
#!/bin/sh

JOB_DIR=/scratch/$USER/win/${PBS_JOBID}

#Virtual machine settings
VM_IMAGE=~/.work/img/win.img
VM_MEMORY=49152
VM_SMP=16

# Prepare job dir
mkdir -p ${JOB_DIR} && cd ${JOB_DIR} || exit 1
ln -s ~/.work/win .
ln -s /scratch/$USER/data .
ln -s ~/.work/win/script/run/run-appl.bat run.bat

# Run virtual machine
export TMPDIR=/lscratch/${PBS_JOBID}
module add qemu
qemu-system-x86_64
    -enable-kvm
    -cpu host
    -smp ${VM_SMP}
    -m ${VM_MEMORY}
    -vga std
    -localtime
    -usb -usbdevice tablet
    -device virtio-net-pci,netdev=net0
    -netdev user,id=net0,smb=${JOB_DIR},hostfwd=tcp::3389-:3389
    -drive file=${VM_IMAGE},media=disk,if=virtio
    -snapshot
    -nographic
```

Job script links application data (win), input data (data) and run script (run.bat) into job directory and runs virtual machine.

Example run script (run.bat) for Windows virtual machine:

```
z:
cd winappl
call application.bat z:data z:output
```

Run script runs application from shared job directory (mapped as drive z:), process input data (z:data) from job directory and store output to job directory (z:output).

Run jobs

Run jobs as usual, see Resource Allocation and Job Execution. Use only full node allocation for virtualization jobs.

Running Virtual Machines

Virtualization is enabled only on compute nodes, virtualization does not work on login nodes.

Load QEMU environment module:

```
$ module add qemu
```

Get help

```
$ man qemu
```

Run virtual machine (simple)

```
$ qemu-system-x86_64 -hda linux.img -enable-kvm -cpu host -smp 16 -m 32768 -vga std -vnc :0
```

```
$ qemu-system-x86_64 -hda win.img -enable-kvm -cpu host -smp 16 -m 32768 -vga std -localtime -u
```

You can access virtual machine by VNC viewer (option -vnc) connecting to IP address of compute node. For VNC you must use VPN network.

Install virtual machine from iso file

```
$ qemu-system-x86_64 -hda linux.img -enable-kvm -cpu host -smp 16 -m 32768 -vga std -cdrom linux
```

```
$ qemu-system-x86_64 -hda win.img -enable-kvm -cpu host -smp 16 -m 32768 -vga std -localtime -u
```

Run virtual machine using optimized devices, user network backend with sharing and port forwarding, in snapshot mode

```
$ qemu-system-x86_64 -drive file=linux.img,media=disk,if=virtio -enable-kvm -cpu host -smp 16 -m
```

```
$ qemu-system-x86_64 -drive file=win.img,media=disk,if=virtio -enable-kvm -cpu host -smp 16 -m
```

Thanks to port forwarding you can access virtual machine via SSH (Linux) or RDP (Windows) connecting to IP address of compute node (and port 2222 for SSH). You must use VPN network.

Keep in mind, that if you use virtio devices, you must have virtio drivers installed on your virtual machine.

Networking and data sharing

For networking virtual machine we suggest to use (default) user network backend (sometimes called slirp). This network backend NATs virtual machines and provides useful services for virtual machines as DHCP, DNS, SMB sharing, port forwarding.

In default configuration IP network 10.0.2.0/24 is used, host has IP address 10.0.2.2, DNS server 10.0.2.3, SMB server 10.0.2.4 and virtual machines obtain address from range 10.0.2.15-10.0.2.31. Virtual machines have access to Anselm's network via NAT on compute node (host).

Simple network setup

```
$ qemu-system-x86_64 ... -net nic -net user
```

(It is default when no -net options are given.)

Simple network setup with sharing and port forwarding (obsolete but simpler syntax, lower performance)

```
$ qemu-system-x86_64 ... -net nic -net user,smb=/scratch/$USER/tmp,hostfwd=tcp::3389-:3389
```

Optimized network setup with sharing and port forwarding

```
$ qemu-system-x86_64 ... -device virtio-net-pci,netdev=net0 -netdev user,id=net0,smb=/scratch
```

Advanced networking

Internet access**

Sometime your virtual machine needs access to internet (install software, updates, software activation, etc). We suggest solution using Virtual Distributed Ethernet (VDE) enabled QEMU with SLIRP running on login node tunnelled to compute node. Be aware, this setup has very low performance, the worst performance of all described solutions.

Load VDE enabled QEMU environment module (unload standard QEMU module first if necessary).

```
$ module add qemu/2.1.2-vde2
```

Create virtual network switch.

```
$ vde_switch -sock /tmp/sw0 -mgmt /tmp/sw0.mgmt -daemon
```

Run SLIRP daemon over SSH tunnel on login node and connect it to virtual network switch.

```
$ dpipe vde_plug /tmp/sw0 = ssh login1 $VDE2_DIR/bin/slirpvde -s - --dhcp &
```

Run qemu using vde network backend, connect to created virtual switch.

Basic setup (obsolete syntax)

```
$ qemu-system-x86_64 ... -net nic -net vde,sock=/tmp/sw0
```

Setup using virtio device (obsolete syntax)

```
$ qemu-system-x86_64 ... -net nic,model=virtio -net vde,sock=/tmp/sw0
```

Optimized setup

```
$ qemu-system-x86_64 ... -device virtio-net-pci,netdev=net0 -netdev vde,id=net0,sock=/tmp/sw0
```

TAP interconnect**

Both user and vde network backend have low performance. For fast interconnect (10Gbps and more) of compute node (host) and virtual machine (guest) we suggest using Linux kernel TAP device.

Cluster Anselm provides TAP device tap0 for your job. TAP interconnect does not provide any services (like NAT, DHCP, DNS, SMB, etc.) just raw networking, so you should provide your services if you need them.

Run qemu with TAP network backend:

```
$ qemu-system-x86_64 ... -device virtio-net-pci,netdev=net1  
                        -netdev tap,id=net1,ifname=tap0,script=no,downscript=no
```

Interface tap0 has IP address 192.168.1.1 and network mask 255.255.255.0 (/24). In virtual machine use IP address from range 192.168.1.2-192.168.1.254. For your convenience some ports on tap0 interface are redirected to higher numbered ports, so you as non-privileged user can provide services on these ports.

Redirected ports:

- DNS udp/53->udp/3053, tcp/53->tcp3053
- DHCP udp/67->udp3067
- SMB tcp/139->tcp3139, tcp/445->tcp3445).

You can configure IP address of virtual machine statically or dynamically. For dynamic addressing provide your DHCP server on port 3067 of tap0 interface, you can also provide your DNS server on port 3053 of tap0 interface for example:

```
$ dnsmasq --interface tap0 --bind-interfaces -p 3053 --dhcp-alternate-port=3067,68 --dhcp-rang
```

You can also provide your SMB services (on ports 3139, 3445) to obtain high performance data sharing.

Example smb.conf (not optimized)

```
[global]
socket address=192.168.1.1
smb ports = 3445 3139

private dir=/tmp/qemu-smb
pid directory=/tmp/qemu-smb
lock directory=/tmp/qemu-smb
state directory=/tmp/qemu-smb
ncalrpc dir=/tmp/qemu-smb/ncalrpc
log file=/tmp/qemu-smb/log.smbd
smb passwd file=/tmp/qemu-smb/smbpasswd
security = user
map to guest = Bad User
unix extensions = no
load printers = no
printing = bsd
printcap name = /dev/null
disable spoolss = yes
log level = 1
guest account = USER
[qemu]
path=/scratch/USER/tmp
read only=no
guest ok=yes
writable=yes
follow symlinks=yes
wide links=yes
force user=USER
```

(Replace USER with your login name.)

Run SMB services

```
smbd -s /tmp/qemu-smb/smb.conf
```

Virtual machine can of course have more than one network interface controller, virtual machine can use more than one network backend. So, you can combine for example use network backend and TAP interconnect.

Snapshot mode

In snapshot mode image is not written, changes are written to temporary file (and discarded after virtual machine exits). **It is strongly recommended**

mode for running your jobs. Set TMPDIR environment variable to local scratch directory for placement temporary files.

```
$ export TMPDIR=/lscratch/${PBS_JOBID}
$ qemu-system-x86_64 ... -snapshot
```

Windows guests

For Windows guests we recommend these options, life will be easier:

```
$ qemu-system-x86_64 ... -localtime -usb -usbdevice tablet
```