SAP HANA on KVM

Best Practices

Resource Guide

Markus Koch (Red Hat)
Dave Dumas (Red Hat)
Sherry Yu (Red Hat)

Version 2.0
June 2017

100 East Davie Street
Raleigh NC 27601 USA
Phone: +1 919 754 3700
Phone: 888 733 4281
Fax: +1 919 754 3701
PO Box 13588
Research Triangle Park NC 27709 USA

The GPG fingerprint of the security@redhat.com key is:
CA 20 86 86 2B D6 9D FC 65 F6 EC C4 21 91 80 CD DB 42 A6 0E

Table of Contents

Introduction

To run SAP HANA virtualized on the KVM (Kernel-based Virtual Machine) hypervisor in a certified production environment., you need to follow the instructions in this guide to configuring, deploying, and sizing SAP HANA on Red Hat® Virtualization (RHV) that is built on KVM hypervisor technology.

SAP HANA Hardware Requirements

SAP HANA must be deployed on certified SAP HANA hardware, either running on bare metal or virtualization. Before deploying HANA, always check the current list of certified SAP HANA Appliances as well as certified Enterprise Storage and Entry Level Systems.

Certified SAP HANA Hardware Directory

RHV Hypervisor System Requirements

RHV Hypervisor Installation

Please follow the instructions in the RHV 4.1 installation guide at https://access.redhat.com/documentation/en/red-hat-virtualization/4.1/paged/installation-guide/

KVM Version Requirements

To run SAP HANA a qemu-kvm-rhev version 2.6 or higher is required. These packages are currently only available as part of Red Hat Virtualization with the Hypervisor on Red Hat Enterprise Linux® 7.3 or later. It is not possible to run SAP HANA on RHEL 7 with the co-packaged qemu-kvm version 1.5.

Make sure the following packages are installed on your RHEL hypervisors:

qemu-img-rhev	**2.6.0-28.el7_3.8 or later**
qemu-kvm-common-rhev	**2.6.0-28.el7_3.8 or later**
qemu-kvm-rhev	**2.6.0-28.el7_3.8 or later**
qemu-kvm-tools-rhev	**2.6.0-28.el7_3.8 or later**

Please note that the guest OS in the virtual machines is completely independent of the OS on the hypervisor. So the first supported hypervisor is based on RHEL 7.3 on which it is possible to run SAP HANA on RHEL 7.2 EUS, which is a current supported release for SAP HANA.

RHV Hypervisor System Requirements

On the RHV hypervisor system, besides the resources allocated to virtual machines, there should be enough disk space and RAM dedicated to the KVM hypervisor. Please ensure the minimum requirements are met.

System Requirements – Host system requirements

Virtualization Extensions

Virtualization extensions are required for full virtualization. To determine whether the system has the hardware virtualization extensions, and that they are enabled, run the check as outlined.

System Requirements – KVM hypervisor requirements

TUNED Profile on Host

Tuned is a tuning profile delivery mechanism that adapts Red Hat Enterprise Linux for certain workload characteristics, such as requirements for CPU-intensive tasks, or storage/network throughput responsiveness. A number of tuning profiles are pre-configured to enhance performance and reduce power consumption in a number of specific use cases.

By default in a Red Hat Enterprise Linux 7 installation, the tuned package is installed and the tuned service is enabled.

The "virtual-host" profile is based on the "throughput-performance" profile, and also decreases the swappiness of virtual memory and enables more aggressive writeback of dirty pages. This profile is the recommended profile for virtualization hosts. Use tuned-adm to check whether the profile 'virtual-host' is set:

```
# tuned-adm active
Current active profile: virtual-host
```

If the current profile is not virtual-host, activate it with the following command:

```
# tuned-adm profile virtual-host
```

Further details on setting the profile can be found below.

TUNED and TUNED-ADM

Kernel Boot Options

The SAP HANA guest needs to be put into 1GB hugepages. To configure this you need to determine the size of your HANA guest to figure out how many hugepages are required.

Then add the following parameters to the kernel bootline:

- **default_hugepagesz=1GB**
- **hugepagesz=1GB**
- **hugepages=[# hugepages]**
- **intel_idle.max_cstate=1**

To calculate the number of hugepages, take the amount of memory you want to spend for the vm (e.g. 131072 MB = 128 GB)

Then add them to the variable GRUB_CMDLINE_LINUX in /etc/sysconfig/grub and update the grub configuration file.

Finally /etc/sysconfig grub should look similar to this (look for the bold entries):

```
~]# cat /etc/sysconfig/grub
GRUB_TIMEOUT=5
GRUB_DISTRIBUTOR="$(sed 's, release .*$,,g' /etc/system-release)"
GRUB_DEFAULT=saved
GRUB_DISABLE_SUBMENU=true
GRUB_TERMINAL_OUTPUT="console"
GRUB_CMDLINE_LINUX="crashkernel=auto rd.lvm.lv=rhel_inf21/root
rd.lvm.lv=rhel_inf21/swap rhgb quiet default_hugepagesz=1GB hugepagesz=1GB
hugepages=128 intel_idle.max_cstate=1 numa_balancing=disable"
GRUB_DISABLE_RECOVERY="true"
```

On BIOS-based machines, issue the following command as `root`:

```
~]# grub2-mkconfig -o /boot/grub2/grub.cfg
```

On UEFI-based machines, issue the following command as `root`:

```
~]# grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
```

See RHEL 7 Administration Guide for details.
A reboot is needed for the changes to take effect.

Storage Pool Setup

Select direct LUNS or disks for /hana from an iScsi or FC storage pool, so that they are mapped as type=raw, cache=none ,io=native.
Alternatively you can choose to use NFS-shares for /hana.

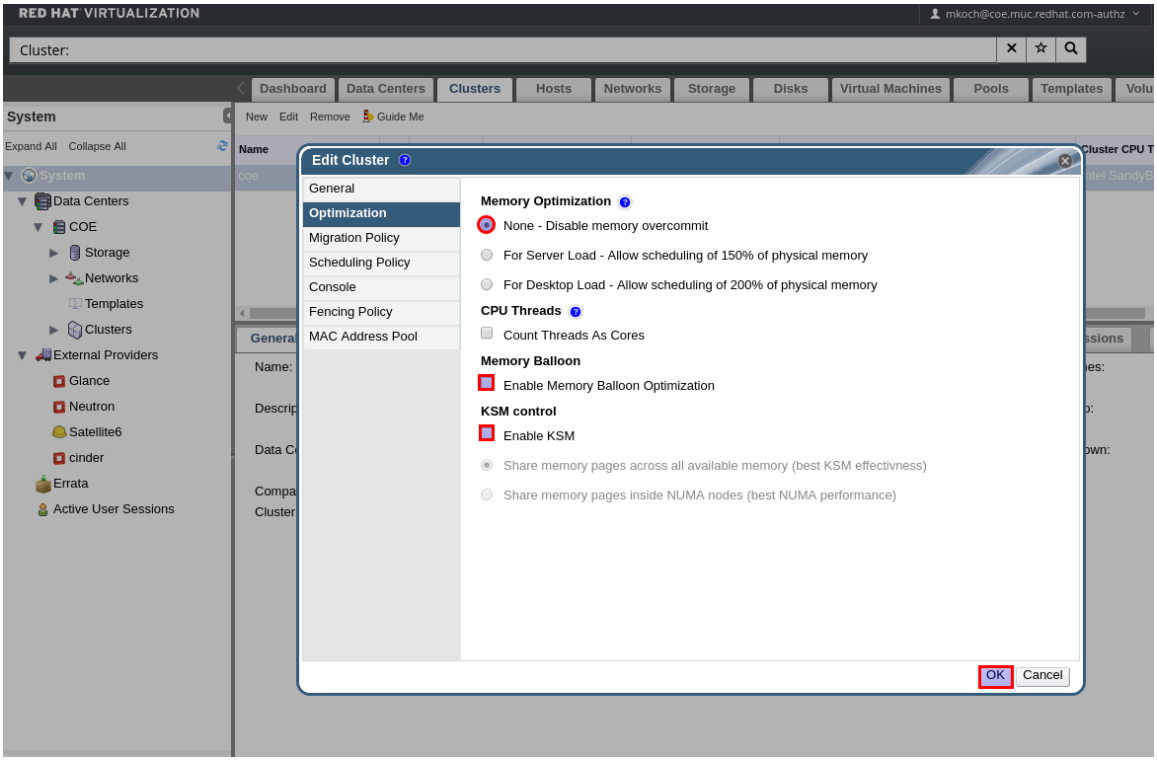
No matter what you choose, you need to ensure that these disks meet the performance requirements (KPIs) as described in [SAP Note 1943937](#).

Set Optimizations in the RHV-M Cluster

Edit the cluster, where you want to host the high performance VMs to make sure the following changes are applied:

- **disable memory overcommit**
- **Disable Memory Balloon Optimization**
- **Disable KSM**

Click on Clusters, select your high performance cluster, and click on "Edit" -> "Optimization" and make the appropriate changes:



The VMs need to use 1GB hugepages in the NUMA nodes. The current version of RHV-Manager is not able to recognise that the VM will try to grab hugepages to run, so that memory checking for newly launched VMs needs to be disabled in the cluster where the SAP HANA VMs should be hosted.

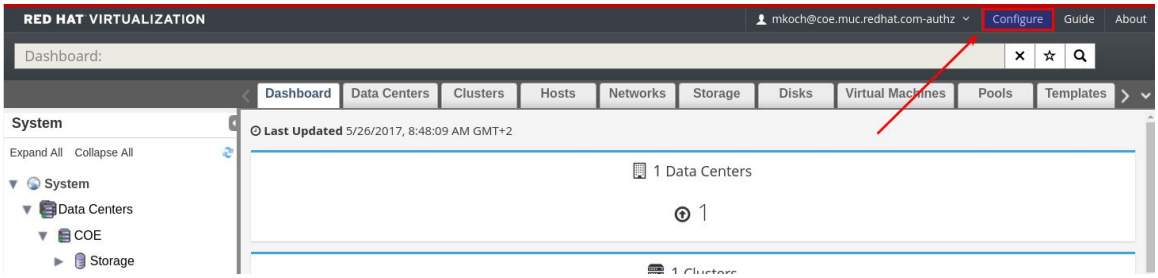
The filter is responsible for checking three things currently (based on RHV 4.1):

- **free reserved memory - all started VMs must fit into (physical memory - host overhead) * overcommit ratio**

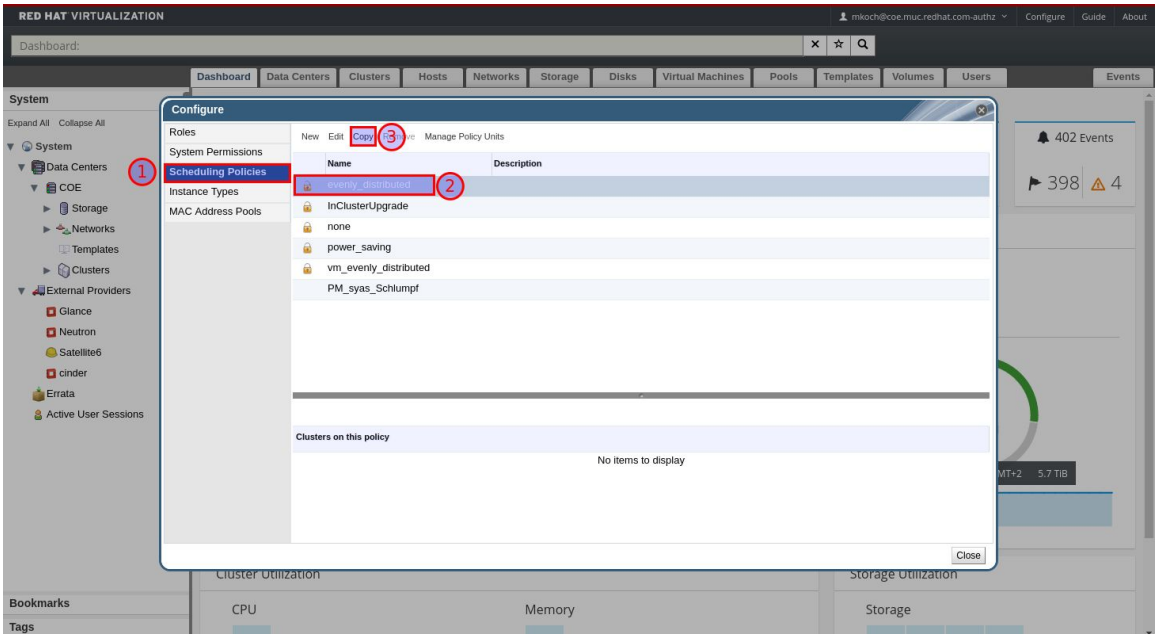
- **free physical memory - QEMU needs to be able to call malloc with the full requested memory**
- **swap usage - if a host is swapping too much it will not be used**

So disabling the filter removes those three checks from the scheduler.

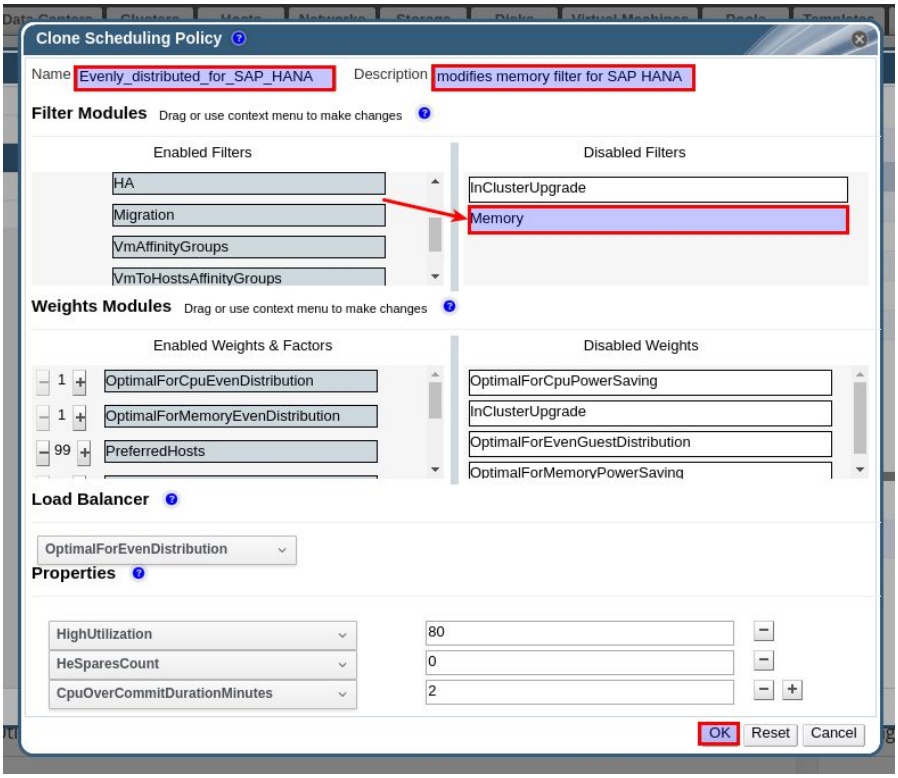
To disable the memory filter click “Configure” in the top right corner.



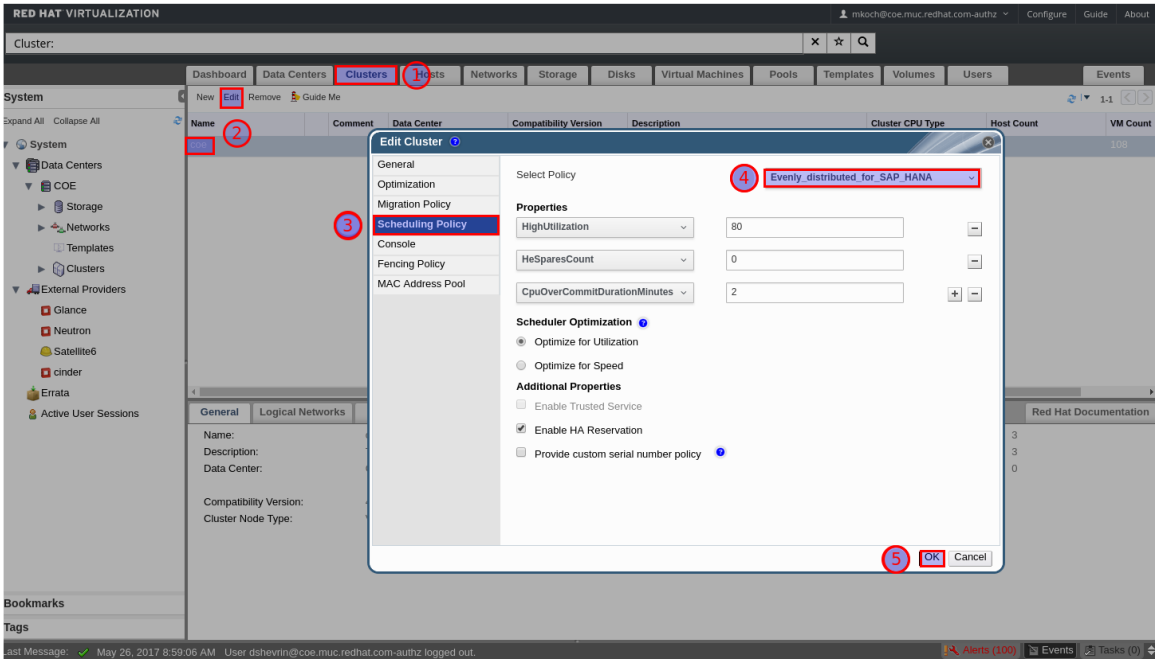
Now select “Scheduling Policies”, select the current policy you use (typically evenly_distributed), and click on “Copy”.



A window with now editable parameters of the scheduling policy opens.
Give it a telling name and optionally a description to identify it later on (e.g. Evenly_distributed_for_SAP_HANA) .
In the “Filter Modules” stanza check the “Enabled Filters” area, locate the “memory” filter and drag it to the “Disabled Filters” area.



Then go to Clusters(1) and select the cluster your HANA systems will be running in. Select Edit(2), then Scheduling Policy(3). In the Drop down menu in the top right corner select the cloned scheduling policy(4) you edited and finish with OK(5).



More information on scheduling policies can be found in the [Administration guide]

Disable KSM on the Host Manually

KSM stands for Kernel Samepage Merging. It is used by the KVM hypervisor, allowing KVM guests to share identical memory pages.

While KSM should already be deactivated in the RHV-M Clustering setup, KSM must be manually deactivated by stopping the ksmtuned and the ksm service on the hypervisor. The reason is that if MOM (Memory Overcommit Manager) detects memory utilization at 95% or more, it will turn KSM on regardless if it was deactivated in RHV-M clustering setup. It should also be noted that when KSM is disabled, any memory pages that were shared prior to deactivating KSM are still shared. To deactivate KSM, run the following in a terminal as root:

```
# systemctl stop ksmtuned
Stopping ksmtuned:                                         [  OK  ]
# systemctl stop ksm
Stopping ksm:                                              [  OK  ]

# systemctl disable ksm
# systemctl disable ksmtuned
```

To delete all of the Page KSM in the system, use the following command in a terminal as root:

```
# echo 2 >/sys/kernel/mm/ksm/run
```

Install Required Hooks for the Virtual Guests

Some configurations that are required for setting up a high performance VM like in RHV 4.1 environments need the following hook.

Log on to your hypervisor, change to the hooks before_vm_start directory and create the following file:

```
# cd /usr/libexec/vdsm/hooks/before_vm_start
# cat > 50_highperf << EOF
#!/usr/bin/python2

import os
import sys
import traceback

import hooking

'''
Syntax:
highperf=1 (value doesn't matter)
```

```
The 1GB hugepages must be already defined during boot-time of the
hypervisor, e.g. like
"default_hugepagesz=1GB hugepagesz=1GB hugepages=[# hugepages needed]"

The VM also needs to have iothreads enabled in the RHV-M Web-UI.
The number of threads need to be set to "1"

As invariant tsc is needed, this flag is explicitly passed through to
the guest. Therefore CPU hostpassthrough needs to be enabled in the
RHV-M Web-UI.
'''


if 'highperf' in os.environ:
    try:
        domxml = hooking.read_domxml()
        domain = domxml.getElementsByTagName('domain')[0]
        if len(domain.getElementsByTagName('memoryBacking')):
            sys.stderr.write('hugepages: VM already have hugepages\n')
            sys.exit(0)

        memoryBacking = domxml.createElement('memoryBacking')
        hugepages = domxml.createElement('hugepages')
        page = domxml.createElement('page')
        page.setAttribute('size', '1048576')
        hugepages.appendChild(page)
        memoryBacking.appendChild(hugepages)
        domain.appendChild(memoryBacking)

        sys.stderr.write('hugepages: adding hugepages tag\n')

        if len(domain.getElementsByTagName('iothreads')):
            iothreadids = domxml.createElement('iothreadids')
            ids = domxml.createElement('iothread')
            ids.setAttribute('id', '1')
            iothreadids.appendChild(ids)
            domain.appendChild(iothreadids)

            if len(domain.getElementsByTagName('cputune')):
                cputune = domain.getElementsByTagName('cputune')[0]
            else:
                cputune = domxml.createElement('cputune')

            iothreadpin = domxml.createElement('iothreadpin')
            iothreadpin.setAttribute('iothread', '1')
            iothreadpin.setAttribute('cpuset', '0')
            emulatorpin = domxml.createElement('emulatorpin')
            emulatorpin.setAttribute('cpuset', '0')
            cputune.appendChild(iothreadpin)
            cputune.appendChild(emulatorpin)
            if not len(domain.getElementsByTagName('cputune')):
                domain.appendChild(cputune)

        if len(domain.getElementsByTagName('cpu')):
            cpu = domain.getElementsByTagName('cpu')[0]
            feature_tsc = domxml.createElement('feature')
            feature_tsc.setAttribute('policy', 'require')
            feature_tsc.setAttribute('name', 'invtsc')
            feature_rdt = domxml.createElement('feature')
```

```
            feature_rdt.setAttribute('policy', 'require')
            feature_rdt.setAttribute('name', 'rdtscp')
            cpu.appendChild(feature_tsc)
            cpu.appendChild(feature_rdt)

        hooking.write_domxml(domxml)
    except Exception:
        sys.stderr.write('highperf hook: [unexpected error]: %s\n' %
                         traceback.format_exc())
        sys.exit(2)
EOF
```

Now login to Rhv-Manager host and define the additional variable that triggers this hook:

Get the current properties:

```
[root@rhv-m ~]# engine-config -g UserDefinedVMProperties
UserDefinedVMProperties:  version: 3.6
UserDefinedVMProperties:  version: 4.0
UserDefinedVMProperties:  version: 4.1
```

Set the highperformance property:

```
[root@rhv-m ~]# engine-config -m UserDefinedVMProperties='highperf=^[0-9]+$'
--cver=4.1
```

Check that the property is set:

```
[root@rhv-m ~]# engine-config -g UserDefinedVMProperties
UserDefinedVMProperties:  version: 3.6
UserDefinedVMProperties:  version: 4.0
UserDefinedVMProperties: highperf=^[0-9]+$ version: 4.1
```

Restart ovirt engine for the changes to take effect:

```
[root@rhv-m ~] /bin/systemctl restart ovirt-engine.service
```

Virtual Machine Requirements

NOTE: Memory or CPU overcommitting is NOT allowed for SAP HANA deployment.

When you create an new virtual machine for SAP HANA, you need to set the following parameters:

- **Enable CPU pinning**
- **Set correct NUMA topology**
- **Enable memory backing with hugepages**
- **Enable CPU passthrough**
- **Enable iothread pinning**

The following settings are recommended for console use:

- **Use the headless option with serial console or**
- **Use VNC instead of SPICE for screen redirection**
- **Disable sound card**

Creating HANA VM via RHV-GUI

This chapter shows how to setup a HANA VM on a hypervisor with the RHV admin GUI. Log into your hypervisor and run "lscpu" and "numactl --hardware". Note the following parameters from the output:

- **Thread(s) per core**
- **Core(s) per socket**
- **Number of Sockets**
- **CPUs/Memory per Numa Node**

The following is an example of a Server with 4 NUMA Nodes and 64 GiB Memory per NUMA node.

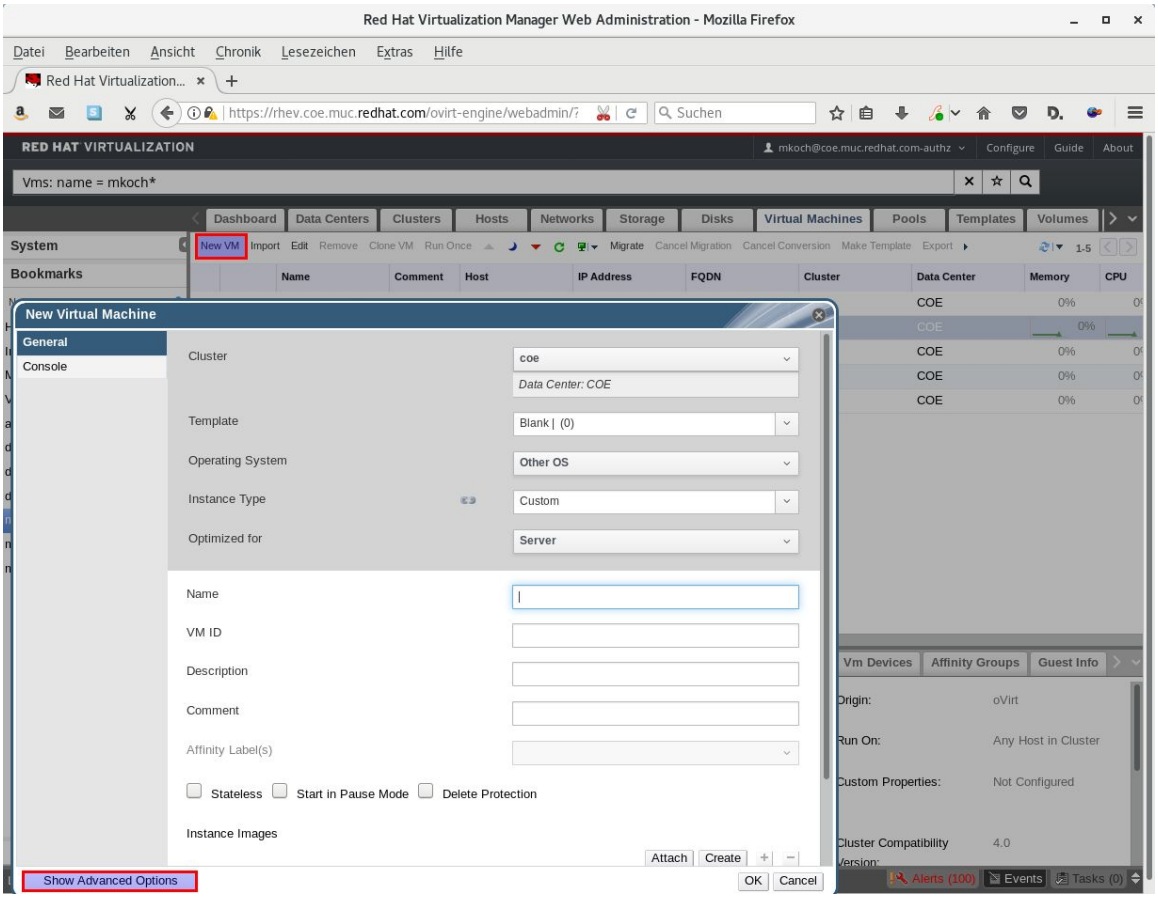
```
[root@inf21 ~]# lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                64
On-line CPU(s) list:   0-63
Thread(s) per core:    2
Core(s) per socket:    8
Socket(s):             4
NUMA node(s):          4
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 45
```

```
Model name:            Intel(R) Xeon(R) CPU E5-4650 0 @ 2.70GHz
Stepping:              7
CPU MHz:               3034.652
BogoMIPS:              5398.30
Virtualization:        VT-x
L1d cache:             32K
L1i cache:             32K
L2 cache:              256K
L3 cache:              20480K
NUMA node0 CPU(s):     0-7,32-39
NUMA node1 CPU(s):     8-15,40-47
NUMA node2 CPU(s):     16-23,48-55
NUMA node3 CPU(s):     24-31,56-63

[root@inf21 ~]# numactl --hardware
available: 4 nodes (0-3)
node 0 cpus: 0 1 2 3 4 5 6 7 32 33 34 35 36 37 38 39
node 0 size: 65459 MB
node 0 free: 4860 MB
node 1 cpus: 8 9 10 11 12 13 14 15 40 41 42 43 44 45 46 47
node 1 size: 65536 MB
node 1 free: 57720 MB
node 2 cpus: 16 17 18 19 20 21 22 23 48 49 50 51 52 53 54 55
node 2 size: 65536 MB
node 2 free: 17605 MB
node 3 cpus: 24 25 26 27 28 29 30 31 56 57 58 59 60 61 62 63
node 3 size: 65536 MB
node 3 free: 59280 MB
node distances:
node   0   1   2   3
  0:  10  21  21  21
  1:  21  10  21  21
  2:  21  21  10  21
  3:  21  21  21  10
```

Please note that the output of the lscpu command can vary depending on your hardware vendor and you need to adapt the output appropriately.

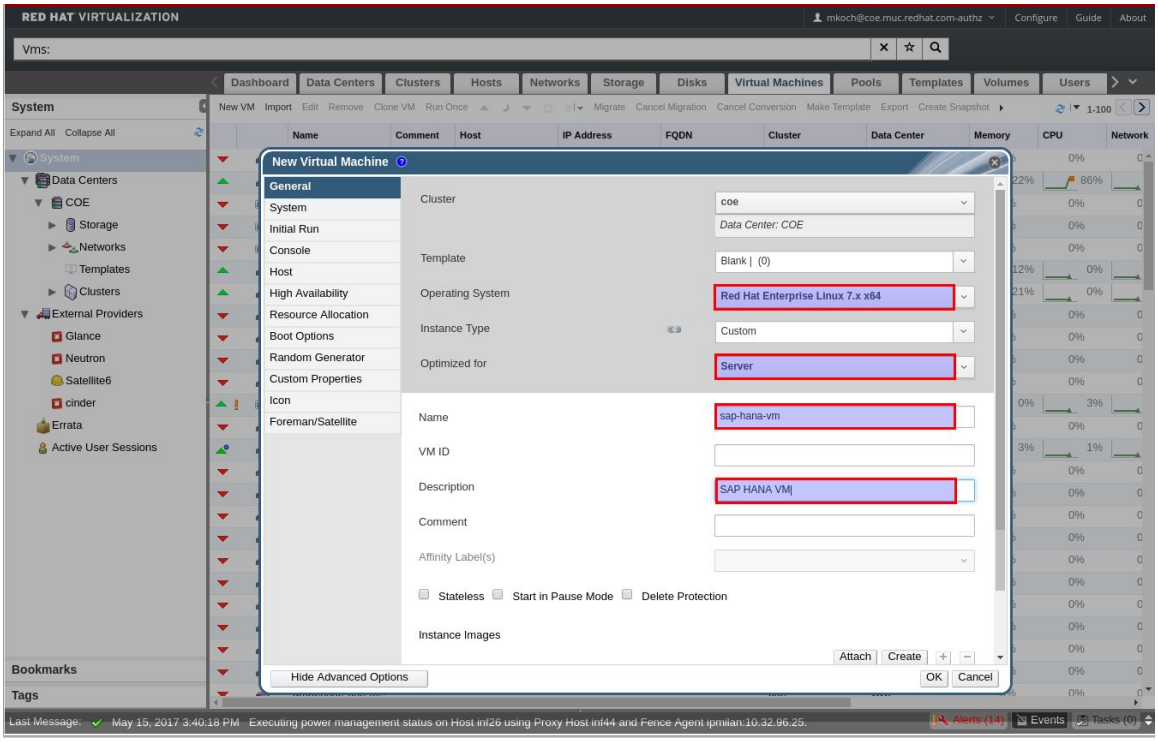
To install a new virtual machine for SAP HANA log into the Administration console of RHEV and click “New VM”.
In the window that opens, click on Advanced Options.



In the general Tab set the following parameters:

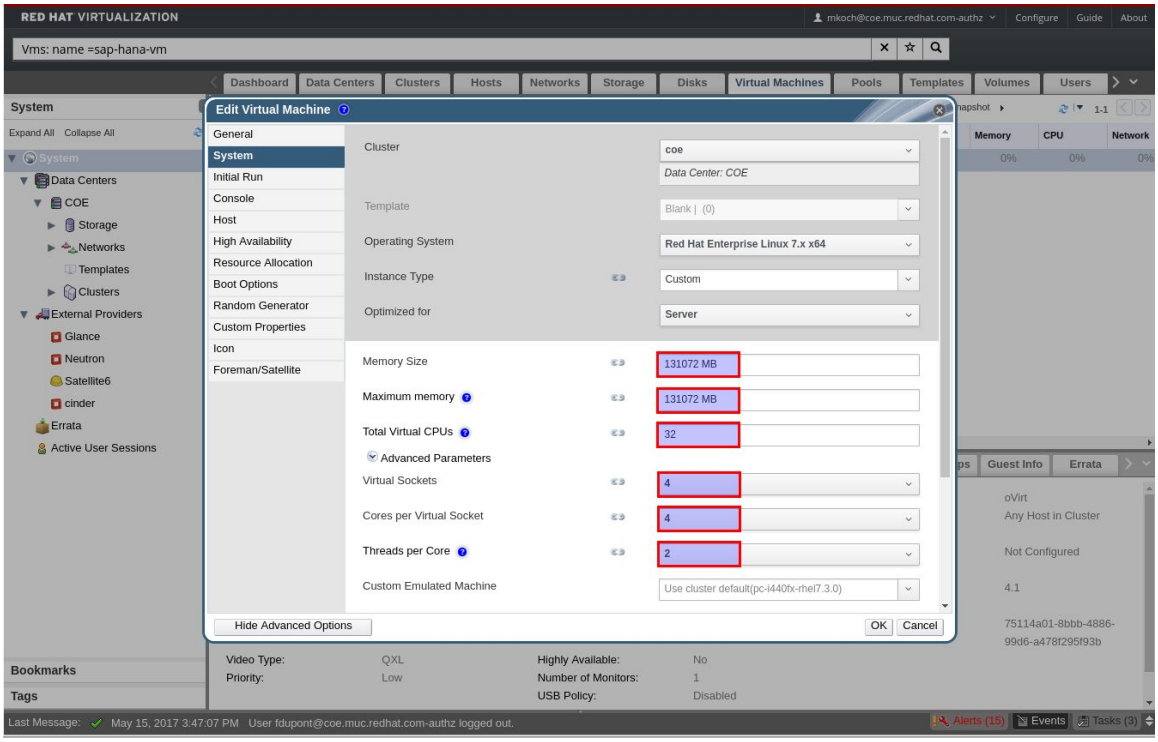
- **Select "Red Hat Enterprise Linux 7.x x64" as Operating System**
- **Instance Type: Custom**
- **Optimized for: Server**
- **Name: Name of the VM**
- **Description: Free Choice**
- **Select: Start in Pause Mode (to control your settings in the end**
- **Scroll down to define your NICs**

Chose Template Blank if you want to PXE-boot the server. If you instantiate templates choose a standard RHEL or an already prepared SAP HANA template. The purpose of this document is not to give an Install Guide for the guest OS of a SAP HANA VM, because this covered in SAP Note 2009879.
The purpose of this guide is to create the virtual machine, so that the guest has optimal performance.



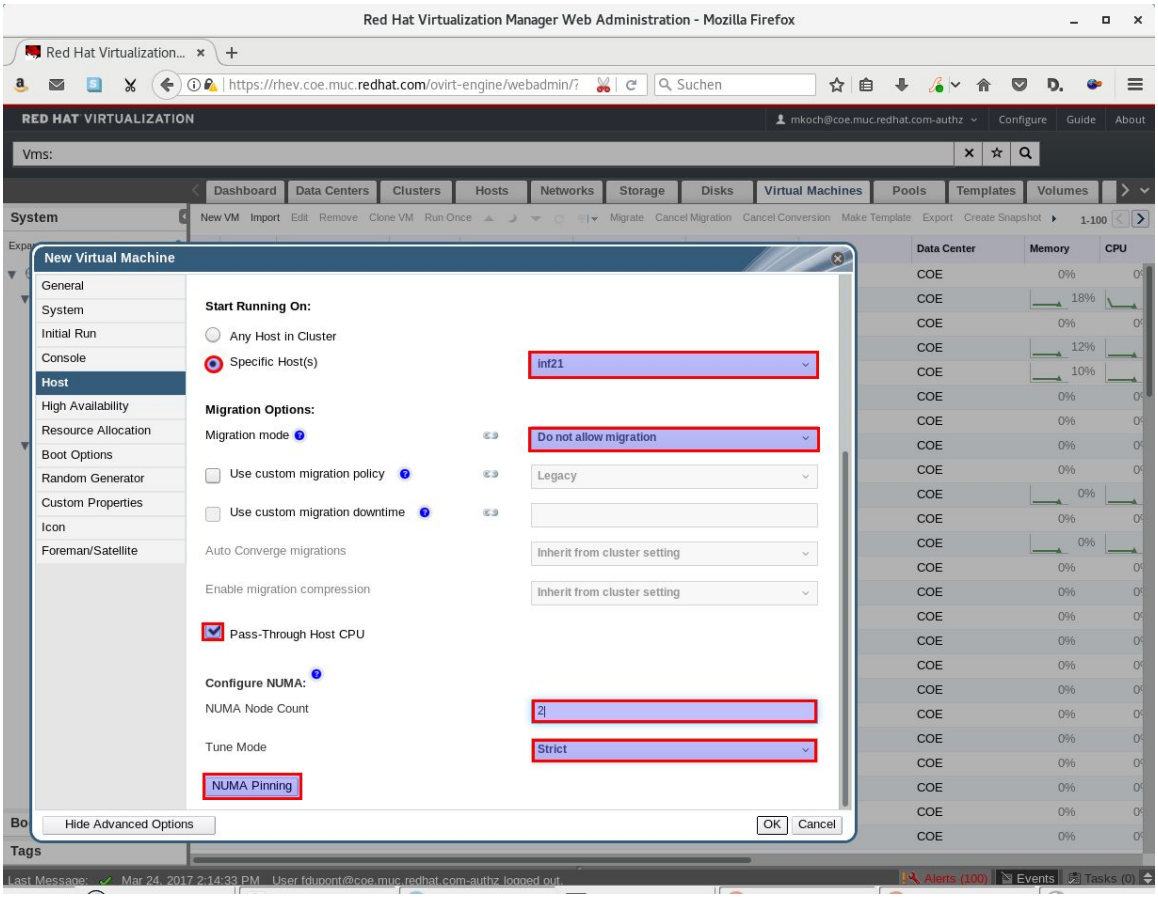
Now click on the System Tab. Fill Out the following fields:

- **Memory Size: #of Numa nodes * memory you want to use per Numa node (e.g. four NUMA Nodes with 128 GiB in total).**
  Note: The hugepages you allocated on the hypervisor will be distributed equally across the server's NUMA Nodes, so you have create your HANA VM across all Numa Nodes.
- **Maximum Memory: equals Memory Size**
- **Total Virtual CPUS: Number of CPUs, make sure that you setup your CPUs identically to the hypervisor (Click "Advanced Parameters"). The number of sockets and hyperthreads needs to match the original hardware settings. If you want all CPUs for you HANA VM make sure Sockets, Cores, and Threads match your hardware topology.**

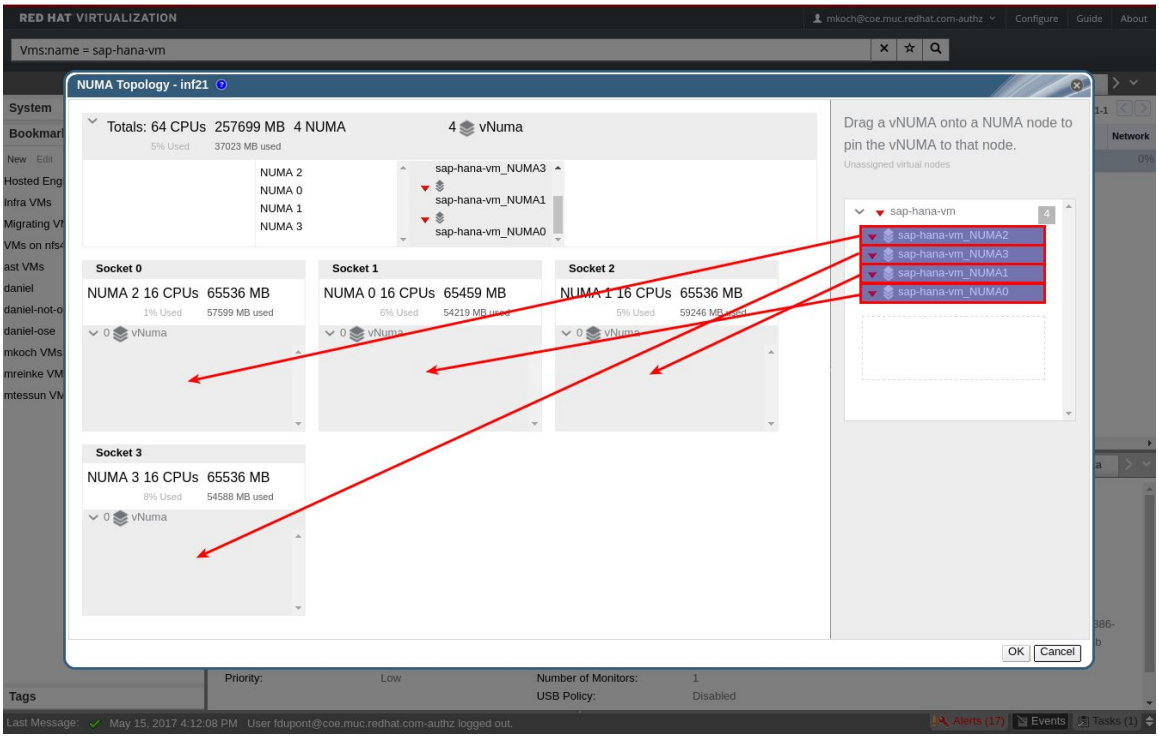


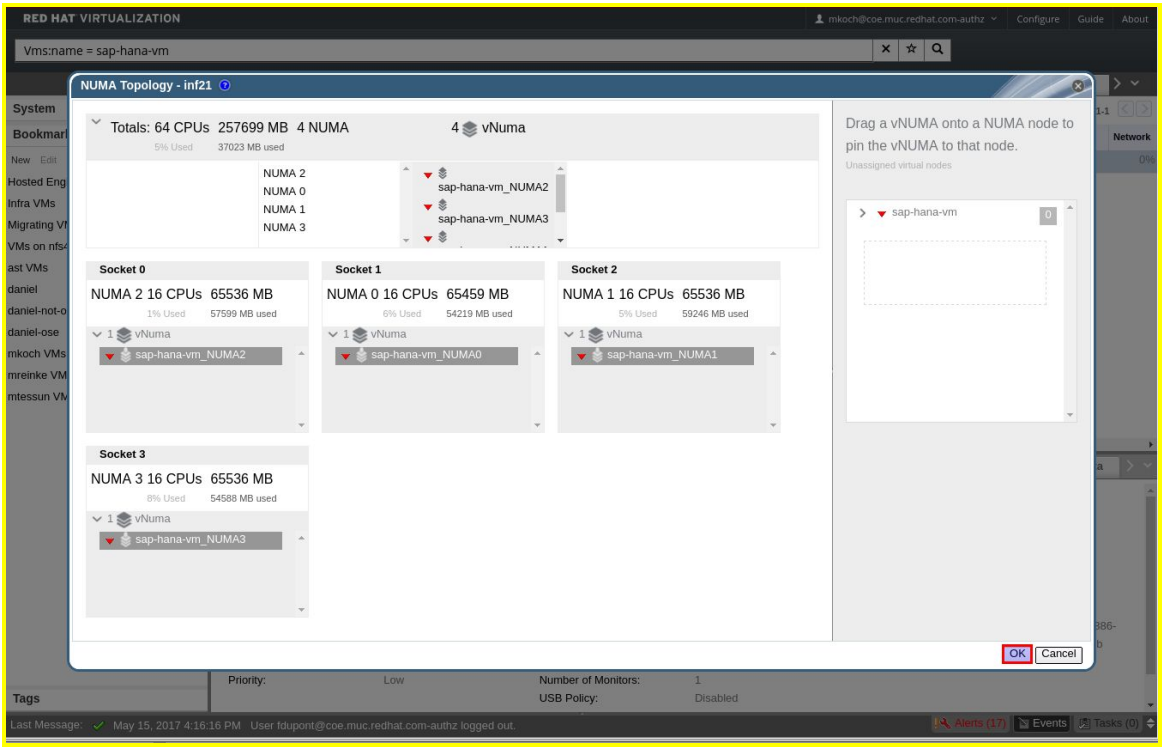
Now click on the Host tab. Make sure the following parameters are set:

- **Start running on: Specific Host (required to pass through CPU flags)**
- **Migration mode: “Do not allow migration”**
- **Select Pass-Through Host CPU**
- **NUMA Node Count: Enter number of Numa Nodes the VM should run on (in our example 4)**
- **Tune Mode: Strict**


Red Hat Virtualization Manager Web Administration - Mozilla Firefox
Red Hat Virtualization...
https://rhev.coe.muc.redhat.com/ovirt-engine/webadmin/?
Suchen
RED HAT VIRTUALIZATION
mkoch@coe.muc.redhat.com-authz
Configure
Guide
About
Vms:
Dashboard
Data Centers
Clusters
Hosts
Networks
Storage
Disks
Virtual Machines
Pools
Templates
Volumes
System
New VM
Import
Edit
Remove
Clone VM
Run Once
Migrate
Cancel Migration
Cancel Conversion
Make Template
Export
Create Snapshot
1-100
Data Center
Memory
CPU
COE
New Virtual Machine
General
System
Initial Run
Console
Host
High Availability
Resource Allocation
Boot Options
Random Generator
Custom Properties
Icon
Foreman/Satellite
Start Running On:
Any Host in Cluster
Specific Host(s)
inf21
Migration Options:
Migration mode
Do not allow migration
Use custom migration policy
Legacy
Use custom migration downtime
Auto Converge migrations
Inherit from cluster setting
Enable migration compression
Inherit from cluster setting
Pass-Through Host CPU
Configure NUMA:
NUMA Node Count
2
Tune Mode
Strict
NUMA Pinning
Hide Advanced Options
OK
Cancel
Tags
Last Message:
Mar 24, 2017 2:14:33 PM
User fdupont@coe.muc.redhat.com-authz logged out.
Alerts (100)
Events
Tasks (0)

Now select “NUMA Pinning” and distribute your NUMA nodes across the hypervisor host. Place the virtual NUMA nodes according to the physical NUMA Nodes and click OK.

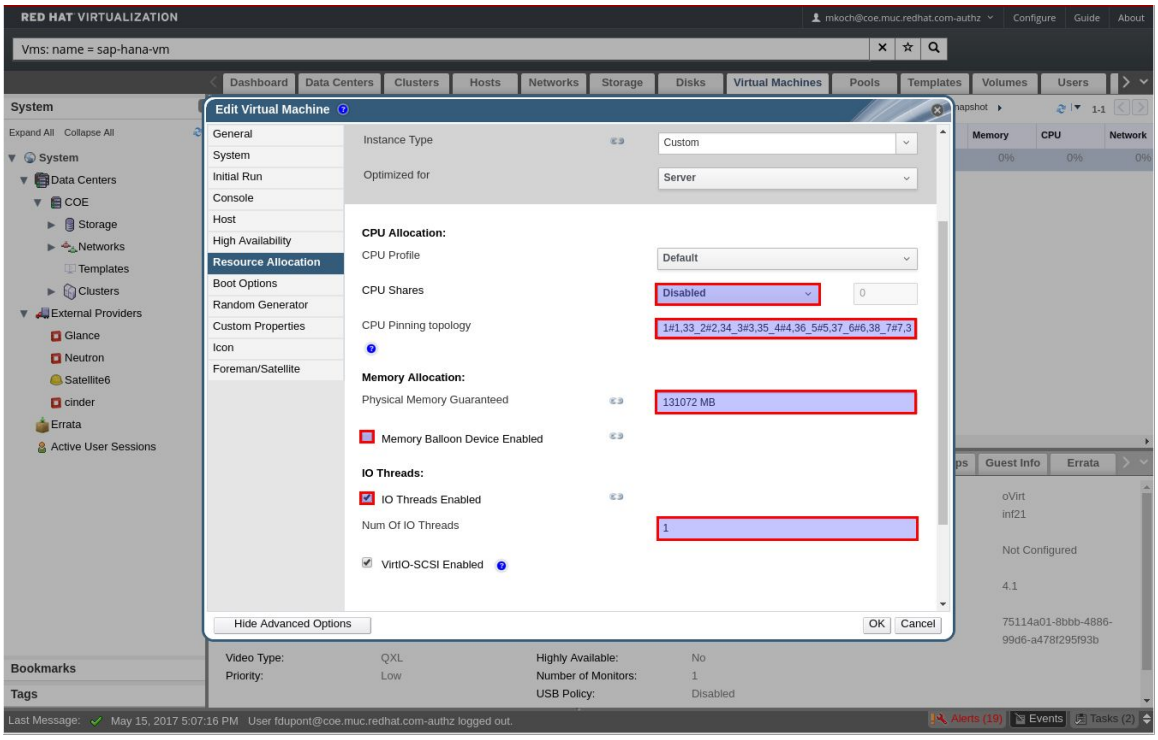




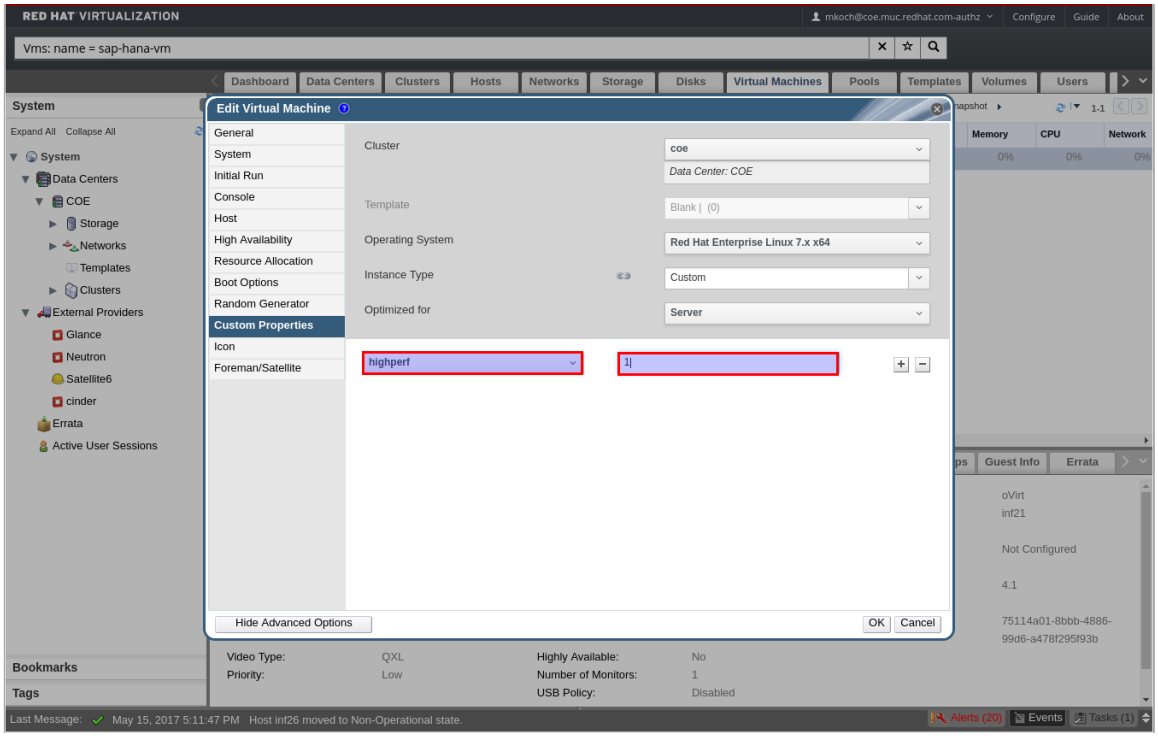
Now click on “Resource Allocation” and

- **disable CPU shares**
- **create a static vCPU pinning. The appropriate syntax has to dedicate each core with its hyperthreads to a vCPU. The generic syntax for this is:**
  vcpu1#core1,thread1_vcpu2#core2,thread2 **etc.**
  Example:
  In our example “numactl --hardware” gives the following output for node 0:
  `node 0 cpus: 0 1 2 3 4 5 6 7 32 33 34 35 36 37 38 39`

  So the appropriate entry for node 0 would be:
  1#1,33_2#2,34_ …. As we want to leave CPU 0 for the hypervisor
  See also Appendix “Calculate CPU pinning”
- **Guarantee the whole required Memory to the VM**
- **Enable the IO Threads**



Now Click on Custom Properties to enable the high performance hook. Click on the drop-down menu, select highperf, and enter 1.



Now continue to add boot disk, network interfaces as usual.

After creating the virtual machine according to the above steps, please verify with virsh -r dumpxml that these settings are correct (need a hook to verify settings). The following example shows a possible output:

```
# virsh -r dumpxml vm
[...]
```

Verify your HANA disk settings

If you have set up your direct attached LUNs correctly, the disks selected for /hana should be mapped as type=raw,cache=none,io=native.
Ensure that these disks meet Hana performance requirements as described here: SAP Note 1943937 - Hardware Configuration Check Tool - Central Note Zentraler Hinweis für das Tool zur Prüfung der Hardwarekonfiguration

You can verify your setup with virsh -r dumpxml command like in the following example:

```
# virsh -r dumpxml vm
[...]
    <disk type='block' device='disk'>
      <driver name='qemu' type='raw' cache='none' io='native' iothread='1' />
      <source dev='/dev/mapper/vg01-lv_hanav_data'/>
      <target dev='vdb' bus='virtio'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x08'
function='0x0'/>
    </disk>
[...]
```

Verify memory backing via hugepages.

You need to check that your guest uses the hugepages that were created during boot time on your hypervisor:

```
# virsh -r dumpxml vm
[...]
       <memoryBacking>
          <hugepages>
             <page size='1048576' unit='KiB'/>
          </hugepages>
       </memoryBacking>

[...]
```

Verify correct NUMA topology

Verify that the NUMA topology that is defined in the xml file matches the topology given by lscpu. For example:

When lscpu will show you:

```
# lscpu
[...]
Thread(s) per core:    2
Core(s) per socket:    18
Socket(s):             4
[...]
```

This should be in the xml-file:

```
# virsh -r dumpxml vm
[...]
```

```
            <topology sockets='4' cores='8' threads='2'/>
          <numatune>
            <memory mode='strict' nodeset='0-3'/>
            <memnode cellid='0' mode='strict' nodeset='0'/>
            <memnode cellid='1' mode='strict' nodeset='1'/>
            <memnode cellid='2' mode='strict' nodeset='2'/>
            <memnode cellid='3' mode='strict' nodeset='3'/>
          </numatune>

            <numa>
              <cell id='0' cpus='0-33' memory='123731968' unit='KiB'/>
              <cell id='1' cpus='34-67' memory='123731968' unit='KiB'/>
              <cell id='2' cpus='68-101' memory='123731968' unit='KiB'/>
              <cell id='3' cpus='102-135' memory='123731968' unit='KiB'/>
            </numa>

[...]
```

Please note: If you do not use all hypervisor CPUs in your VM the unused cores show up as sockets due to the hot-add-CPUmemory functionality of RHV. Make sure that cores and threads match your setup.

Verify correct vCPU pinning

Understanding lscpu output from the host is key to getting this correct. lscpu output will look different for different hardware vendors. The BIOS sets the layout.

The following is an example based on socket 0 of a Dell Broadwell server:

Partial lscpu output:

```
NUMA node0 CPU(s):
0,4,8,12,16,20,24,28,32,36,40,44,48,52,56,60,64,68,72,76,80,84,88,92,96,100,1
04,108,112,116,120,124,128,132,136,140
NUMA node1 CPU(s):
1,5,9,13,17,21,25,29,33,37,41,45,49,53,57,61,65,69,73,77,81,85,89,93,97,101,1
05,109,113,117,121,125,129,133,137,141
NUMA node2 CPU(s):
2,6,10,14,18,22,26,30,34,38,42,46,50,54,58,62,66,70,74,78,82,86,90,94,98,102,
106,110,114,118,122,126,130,134,138,142
NUMA node3 CPU(s):
3,7,11,15,19,23,27,31,35,39,43,47,51,55,59,63,67,71,75,79,83,87,91,95,99,103,
107,111,115,119,123,127,131,135,139,143
```

This needs to be reflected in the vm definition as follows - Please note it is required to leave CPU#0 for the iothread. In the example, CPU 0,72 is not used and as such is available for iothreads.

```
[...]
        <cputune>
```

```
[... socket 0 begins Note that 0,72 is not assigned ...]
          <vcpupin vcpu='0' cpuset='4,76'/>
          <vcpupin vcpu='1' cpuset='4,76'/>
          <vcpupin vcpu='2' cpuset='8,80'/>
          <vcpupin vcpu='3' cpuset='8,80'/>
          <vcpupin vcpu='4' cpuset='12,84'/>
          <vcpupin vcpu='5' cpuset='12,84'/>
          <vcpupin vcpu='6' cpuset='16,88'/>
          <vcpupin vcpu='7' cpuset='16,88'/>
          <vcpupin vcpu='8' cpuset='20,92'/>
          <vcpupin vcpu='9' cpuset='20,92'/>
          <vcpupin vcpu='10' cpuset='24,96'/>
          <vcpupin vcpu='11' cpuset='24,96'/>
          <vcpupin vcpu='12' cpuset='28,100'/>
          <vcpupin vcpu='13' cpuset='28,100'/>
          <vcpupin vcpu='14' cpuset='32,104'/>
          <vcpupin vcpu='15' cpuset='32,104'/>
          <vcpupin vcpu='16' cpuset='36,108'/>
          <vcpupin vcpu='17' cpuset='36,108'/>
          <vcpupin vcpu='18' cpuset='40,112'/>
          <vcpupin vcpu='19' cpuset='40,112'/>
          <vcpupin vcpu='20' cpuset='44,116'/>
          <vcpupin vcpu='21' cpuset='44,116'/>
          <vcpupin vcpu='22' cpuset='48,120'/>
          <vcpupin vcpu='23' cpuset='48,120'/>
          <vcpupin vcpu='24' cpuset='52,124'/>
          <vcpupin vcpu='25' cpuset='52,124'/>
          <vcpupin vcpu='26' cpuset='56,128'/>
          <vcpupin vcpu='27' cpuset='56,128'/>
          <vcpupin vcpu='28' cpuset='60,132'/>
          <vcpupin vcpu='29' cpuset='60,132'/>
          <vcpupin vcpu='30' cpuset='64,136'/>
          <vcpupin vcpu='31' cpuset='64,136'/>
          <vcpupin vcpu='32' cpuset='68,140'/>
          <vcpupin vcpu='33' cpuset='68,140'/>
[... socket 1 begins ...]
           <vcpupin vcpu='34' cpuset='1,73'/>
           <vcpupin vcpu='35' cpuset='4,76'/>
          <vcpupin vcpu='36' cpuset='9,81'/>
           [...]
```

Verify CPU mode is set to host-passthrough

Certain CPU properties are crucial for SAP HANA, so check that they are required. HANA checks for these and if they are not found, HANA will drop back to making system calls which will result in a performance penalty.
Check CPU properties on the guest (/proc/cpuinfo) for 'rdtscp' and 'nonstop_tsc'. Also check HANA startup message file to verify the timer is found and being used:

```
<cpu mode='host-passthrough'>
   <feature policy='require' name='invtsc'/>
   <feature policy='require' name='rdtscp'/>
```

Verify that a dedicated iothread is pinned to a CPU

The iothread should be pinned to the physical socket where I/O device is located. Please check in your vm configuration that this is the case.

Definition of dedicated iothread:

```
<iothreads>1</iothreads>
  <iothreadiiothreadds>
    < id='1'/>
  </iothreadids>
```

Make sure it is attached to the appropriate cpuset in the cputune section:

```
[...]
         <cputune>
           <iothreadpin iothread='1' cpuset='0,72'/>
           <emulatorpin cpuset="0,72"/>
[...]
```

KVM RHEL Guest Installation Requirements

In the previous chapter you have created a virtual machine and installed it with base RHEL. This chapter describes how to configure RHEL for SAP HANA.

SAP HANA Installation on RHEL

Read the required documentation carefully before starting a SAP HANA deployment. The documentation contains information pertaining to supportability.

SAP Notes

- **SAP Note 2009879 - SAP HANA Guidelines for Red Hat Enterprise Linux RHEL) Operating System (includes Installation Guide)**
- **SAP Note 2292690 - SAP HANA DB: Recommended OS settings for RHEL 7.2**
- **SAP Note 1943937 - Hardware Configuration Check Tool - Central Note (contains the user guide for HWCCT)**
- **SAP Note 1788665 - SAP HANA Support for virtualized / partitioned (multi-tenant) environments**
- **SAP Note 1943937 - Hardware Configuration Check Tool - Central Note Zentraler Hinweis für das Tool zur Prüfung der Hardwarekonfiguration**

Red Hat Support Articles

- What's Red Hat Enterprise Linux for SAP HANA Subscription?
- Red Hat Enterprise Linux for SAP HANA: system update and supportability
- RHEL 7.2 and higher: How to subscribe a RHEL 7 system to RHEL for SAP HANA child channel?
- How to subscribe RHEL 7 SAP HANA system to Extended Update Support (EUS) channel?

Additional Installation Notes for the Guest

Make sure to install the virtual machine with SAP HANA according to SAP Note 2009879. Different from and in addition to this configuration guide, the following settings need to be done.

Install and enable the qemu guest agent on the SAP HANA guest from the rhel-7-server-rh-common-rpms repository:

```
~]# yum -y install rhevm-guest-agent-common qemu-guest-agent
~]# systemctl start qemu-guest-agent
~]# systemctl enable qemu-guest-agent
```

Add the following parameters to the kernel boot line:

```
selinux=0
nohz=full
numa_balancing=disable
```

These parameters disable SELinux and numa_balancing in the guest and force the guest to use the hardware timer that is passed through.

To do this, add them to the variable GRUB_CMDLINE_LINUX in /etc/sysconfig/grub and update the grub configuration file.
On BIOS-based machines, issue the following command as `root`:

```
~]# grub2-mkconfig -o /boot/grub2/grub.cfg
```

On UEFI-based machines, issue the following command as `root`:

```
~]# grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
```

See RHEL 7 Administration Guide for details.
Reboot for the changes to take effect.

TUNED Profile on Guest

“Virtual-guest” tuned profile is based on the “throughput-performance” profile and also decreases the swappiness of virtual memory.

The virtual-guest profile is automatically selected when creating a Red Hat Enterprise Linux 7 guest virtual machine. It is the generally recommended profile for virtual machines, but for SAP HANA you need to use the sap-hana profile instead.

```
# tuned-adm profile sap-hana
```

Further details on setting the profile can be found below.

TUNED and TUNED-ADM

KVM Guest Timing Management

Verify that HANA uses the correct timing function by checking HANA startup log, if RDTSC timer is used:

Switch to the following directory */hana/shared/***<SID>/HDB00/<systemname>/trace and look in the indexserver trace file for the timer. See the following example (the numbers will vary):**

```
# grep Timer.cpp indexserver_<system name>.30003.001.trc
[4330]{-1}[-1/-1] 2017-04-18 10:50:59.068073 w Basis Timer.cpp(00718) :
Fallback to system call for HR timer
```

The above indicates fallback to system calls--at least one CPU flag is missing. nonstop_tsc not being there will cause this. This output from the grep command is what you want to see:

```
# grep Timer.cpp indexserver_<system name>.30003.001.trc
[4548]{-1}[-1/-1] 2017-03-16 08:43:36.137096 i Basis Timer.cpp(00642) : Using
RDTSC for HR timer
```

For more details on the guest timing management, please see below.

KVM Guest Timing Management

Verification of CPU/NUMA Settings

Your CPU/NUMA settings are correct if all of the following commands deliver the same CPU/NUMA topology:

- **In host and guest compare the outputs of**
  - **numactl -H**

    - lscpu
- Compare this OS NUMA topology with the topology HANA figured out:
    - As user <sidadm> run the following command: “hdbcons 'jexec info'”

Further Considerations

Virtualization Limits for KVM and RHEV

When it comes to sizing, the following limits apply to KVM and RHV respectively. When planning the resources, please make sure that enough memory is dedicated to the hypervisor host, according to the requirements outlined in the previous section as well as the below mentioned links.

Virtualization limits for Red Hat Enterprise Linux with KVM

Virtualization limits for Red Hat Enterprise Virtualization

High Availability

VMs as Highly Available Resources

Both the RHEL High Availability Add-On and Red Hat Virtualization provide mechanisms for highly available KVM virtual machines.

The smallest RHV solution currently requires 4 nodes: 2 to provide HA for the RHV-M server (either externally or hosted engine) and 2 to act as VM hosts for HANA VM.

High Availability can be achieved by having 2 VMs that run in System Replication mode clustered with the HA Add-On.

See also: IMPROVING UPTIME WITH VIRTUAL MACHINE HIGH AVAILABILITY

Live Migration

Live Migration is currently not available due to the specific needed settings for the VM and will not be supported in the first release.

APPENDIX

Put guest in 1GB huge pages

Example of how to determine the number of 1GB hugepages required to get the entire guest in huge pages.

Find out the size of memory for your guest:
Look for these values in your vm config:

```
# virsh -r dumpxml vmname)
[...]
<memory unit='KiB'>494927872</memory>
<currentMemory unit='KiB'>494927872</currentMemory>
[...]
```

In this case our unit is KiB: 494927872 KiB / 1024 / 1024 = 472 Gb

Note: Make sure the memory size that you're allocating in your .xml file is an exact multiple of 1Gb.
If it isn't, round up or down. If making a change, you'll need to modify the "memory unit" and "current Memory unit" fields, as well as the "cell id" memory values for each NUMA node.
Add the following to the kernel boot line:

```
default_hugepagesz=1GB hugepagesz=1GB hugepages=[# hugepages]
```

Calculate CPU Pinning

The following script helps to print out the appropriate CPU pinning entry. Copy and paste the output to the CPU pinning field in the resource allocation tab.

```
#!/bin/bash

usage() {
cat << EOR
Usage: $0 [num_vcpu]

This script creates CPU pinning according to the host CPU topology up to the max
number of CPUs and always leaving cpu core0 and according threads to the hypervisor
for each numa node

EOR
exit 1
}

num_sockets=$(lscpu | awk -F: '/^Socket\(s\)/ { print $2}' | tr -d " ")
num_threads=$(lscpu | awk -F: '/^Thread\(s\) per core/ { print $2}'| tr -d " ")
if [ -n "$1" ]; then
  max_cnt=$(( $1 / $num_sockets / $num_threads ));
  PARM="-v CNT=$max_cnt"
fi

numactl --hardware | awk -F: -v THREADS=$num_threads $PARM '
       BEGIN { num_threads=THREADS;
               vcpu=1
               if ( CNT ) { cnt = CNT;
                            print ("limiting to "cnt++" cpus/numa node. " );
                            }
               else {
                            cnt = 0;
                            print ("Using all cpus/numa node. " );
                    }
              }
       /node [0-9]+ cpus/ {
```

```
            # split $2 by " " in array cores
            num_vcpu=split($2,threadnum," ");
            num_core=num_vcpu / num_threads;
            if ( cnt == 0 ) { cnt=num_core }
            for (i=2;i<=cnt ;i++) {
                   printf (vcpu++"#"threadnum[i]","threadnum[i+num_core]"_");
            }
           }
      END { printf("\n"); }' |  sed 's/.$//'
```

Log into the hypervisor where your HANA-VM will run and check the output of lscpu. In our example we have 4 CPUs with 8 cores and 2 threads. Sample output of the script looks like this:
All CPUs in use for HANA VM:

```
# ./print_cpu_pinning
Using all cpus/numa node.
1#1,33_2#2,34_3#3,35_4#4,36_5#5,37_6#6,38_7#7,39_8#9,41_9#10,42_10#11,43_11#1
2,44_12#13,45_13#14,46_14#15,47_15#17,49_16#18,50_17#19,51_18#20,52_19#21,53_
20#22,54_21#23,55_22#25,57_23#26,58_24#27,59_25#28,60_26#29,61_27#30,62_28#31
,63
```

In case you only want to use 32 vCPUs instead of all 64 run this:

```
./print_cpu_pinning 32
limiting to 4 cpus/numa node.
1#1,33_2#2,34_3#3,35_4#4,36_5#9,41_6#10,42_7#11,43_8#12,44_9#17,49_10#18,50_1
1#19,51_12#20,52_13#25,57_14#26,58_15#27,59_16#28,60
```

Please note that the first physical CPU core is not pinned and as such is left for the hypervisor and the iothreads.