ORACLE

Hard Partitioning with Oracle Linux KVM

July, 2024

Introduction

This document describes hard partitioning with Oracle Linux KVM in conjunction with Oracle Linux Virtualization Manager, and how to use it to conform to the Oracle Partitioning Policy.

CPU Cores and CPU Threads

On an x86-based system, a CPU core (no hyperthreading enabled) or a CPU thread (hyperthreading enabled) within a core is presented as a physical CPU by the hypervisor or the bare metal operating system. vCPUs (virtual CPUs) are exposed to the guest virtual machine (VM) as CPUs. The guest VM schedules applications on these vCPUs, and the hypervisor schedules these vCPUs over the physical CPU cores or threads.

Oracle Linux KVM offers an advanced feature for hard partitioning, also known as CPU pinning. Hard partitioning means binding vCPUs to physical CPU threads or cores, and prevents these vCPUs from being scheduled on physical CPUs, threads or cores, other than the ones specified.

Oracle Hard Partitioning Policy

To conform to the Oracle Partitioning Policy for hard partitioning, follow the instructions described in this paper to bind vCPUs to physical CPU threads or cores.

Live migration of CPU pinned virtual machines to another Oracle Linux KVM node is not permitted under the terms of the hard partitioning policy. Consequently, the cluster, that is a pool of Oracle Linux KVM nodes with shared storage, must not be configured with any scheduling policy available on Oracle Linux Virtualization Manager.

When live migration is used with pinned virtual machines running Oracle software in an Oracle Linux KVM cluster, hard partition licensing for Oracle software is not applicable. You must determine the number of virtual machines running the Oracle software and then license the same number of physical servers (starting with the largest servers based on the CPU core count) up to the total number of the physical servers in the cluster. For example, if a customer has a cluster with 32 servers and 20 virtual machines running Oracle software within the cluster, the customer must license the 20 largest physical servers in the cluster. If the customer is running 50 virtual machines with Oracle software in a cluster of 32 physical servers, they need only to license the 32 physical servers in the cluster.

Live migration of other virtual machines with non-Oracle software within the server pool is not relevant to Oracle software hard partitioning and has no impact to how Oracle software licensing is calculated.

"Trusted partitions" allow subset licensing without limitation on live migration, but is only available on the approved Oracle Engineered Systems listed in the Oracle Partitioning Policy.

ORACLE

Understanding CPU topology in Oracle Linux KVM

On a server running Oracle Linux KVM, you can run the `lscpu` command to print out the basic CPU configuration of the server hardware. Look for the lines below in the output for detail on your system's CPUs.

```
# lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                104
On-line CPU(s) list:   0-103
Thread(s) per core:    2
Core(s) per socket:    26
Socket(s):             2
NUMA node(s):          2
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 85
Model name:            Intel(R) Xeon(R) Platinum 8167M CPU @ 2.00GHz
Stepping:              4
CPU MHz:               1820.519
CPU max MHz:           2400.0000
CPU min MHz:           1000.0000
BogoMIPS:              4000.00
Virtualization:        VT-x
L1d cache:             32K
L1i cache:             32K
L2 cache:              1024K
L3 cache:              36608K
NUMA node0 CPU(s):     0-25,52-77
NUMA node1 CPU(s):     26-51,78-103
```

Some considerations on the output above:

- This server has 2 sockets with 26 cores and 2 threads per core. Total of 104 "CPUs"
- CPU 0..103 is really thread 0..103
- NUMA node0 includes threads 0-25 and 52-77
- NUMA node1 includes threads 26-51 and 78-103

Get the CPU Topology for vCPU Bindings to Physical CPUs

The command `virsh`, available on an Oracle Linux KVM Server managed by Oracle Linux Virtualization Manager, can be executed in read-only mode. This command generates the list of active virtual machines running locally on the server:

```
[root@kvm03: ~]# virsh --readonly list
 Id   Name              State
--------------------------------
 1    ocne-control01    running
 2    ocne-operator     running
 3    ocne03            running
 5    ol8vm             running
```

With the same `virsh` command, it is easy to check if the CPUs for a virtual machine are pinned to a physical thread/core:

```
# virsh --readonly vcpuinfo ol8vm --pretty
VCPU:           0
CPU:            2
State:          running
CPU time:       680.5s
CPU Affinity:   0-11 (out of 12)

VCPU:           1
```

```
CPU:            7
State:          running
CPU time:       596.9s
CPU Affinity:   0-11 (out of 12)
```

In the above example, the virtual machine is configured with 2 vCPUs and both can span from thread/core 0 until thread/core 11 (out of 12); so, the virtual CPUs for this virtual machine are not pinned to physical threads/cores.

Virtual machines with vCPUs pinned to physical threads/cores, can be easily identified by the limited "CPU Affinity" compared to the total number of physical threads/cores (out of number):

```
# virsh --readonly vcpuinfo ol8vm --pretty
VCPU:           0
CPU:            0
State:          running
CPU time:       5.4s
CPU Affinity:   0-1 (out of 12)

VCPU:           1
CPU:            1
State:          running
CPU time:       2.9s
CPU Affinity:   0-1 (out of 12)
```

Configuring Hard Partitioning

Setting Hard Partitioning using the Oracle Linux Virtualization Manager Utility

The server running Oracle Linux KVM must meet the requirements as an Oracle Linux KVM compute host defined in the Oracle Linux Virtualization Manager Architecture and Planning Guide. The `olvm-vmcontrol` utility is required to set and get the CPU/vCPU bindings for a virtual machine running on Oracle Linux KVM through the Oracle Linux Virtualization Manager.

The `olvm-vmcontrol` utility is available for Oracle Linux 8 (ovirt 4.5) channels from Oracle Linux yum server or the Unbreakable Linux Network (ULN). Please review the `olvm-vmcontrol` utility man page file in the RPM for additional information on the utility and the commands that can be executed:

```
# man olvm-vmcontrol
```

For further details please refer to Oracle Linux Virtualization Manager Documentation.

The `olvm-vmcontrol` utility can run on the host running Oracle Linux Virtualization Manager or a separate Oracle Linux host that has connectivity to the Oracle Linux Virtualization Manager. Before running the utility, make sure that the utility version matches the Oracle Linux Virtualization Manager version.

If the utility has to be installed on a separate Oracle Linux host, the following is required:

- The external host has to run the same Oracle Linux major release running on the Oracle Linux Virtualization Manager host (Oracle Linux 8).
- The relevant "oVirt" yum channels must be enabled on the external host system.

[OPTIONAL]

To enable the "oVirt" Yum channels on an external host, proceed with the following steps:

- Oracle Linux 8

```
# dnf install oracle-ovirt-release-45-el8 -y
```

[OPTIONAL END]

To install the CPU pinning utility, you first need to install the required RPM:

```
# dnf install olvm-vmcontrol -y
```

Once installed the `olvm-vmcontrol` utility is available on the Linux system path:

```
# which olvm-vmcontrol
/usr/bin/olvm-vmcontrol
```

The `olvm-vmcontrol` utility requires authentication to the Oracle Linux Virtualization Manager for each operation executed. If a large number of operations have to be executed and if you are running the `olvm-vmcontrol` utility on the same Oracle Linux Virtualization Manager host, it's suggested that you setup an environment variable dedicated to the password:

```
# export OLVMUTIL_PASS=<password>
```

By specifying the "**-e**" option and with this environment setting in place, the execution of `olvm_vmcontrol` binary will not require authentication.

In the example below, a virtual machine named **ol8vm** accepts the action **getvcpu** to show that virtual CPUs of this guest are not bound to any specific thread or core.

- Syntax example

```
# olvm-vmcontrol -m <FQDN-OLVM-HOST> -u <USER-ID> -v <VM-NAME> -c <COMMAND> -e
```

- Execution example

```
# olvm-vmcontrol -m ovirt.it.oracle.com -u admin@internal -v ol8vm -c getvcpu -e
/usr/lib/jvm/java-11-openjdk-11.0.14.0.9-2.el8_5.x86_64/bin/java has not been configured as
    an alternative for java
Oracle Linux Virtualization Manager VM Control Utility 4.4.1-1.1
OLVMUTIL_PASS environment variable not found.
Password:
Connected to Oracle Linux Virtualization Manager 4.4.8.6-1.0.11.el8
Getting vcpu pinning ...
No CPU pinning is configured
```

Let's bind the vCPUs to core 0 by running the following command

- Syntax example

```
# olvm-vmcontrol -m <FQDN-OLVM-HOST> -u <USER-ID> -v <VM-NAME> -c <COMMAND> -s <CPULIST> -e
```

- Execution example on a demo system

```
# olvm-vmcontrol -m ovirt.it.oracle.com -u admin@internal -v ol8vm -c setvcpu -s 0,1 -e
```

```
/usr/lib/jvm/java-11-openjdk-11.0.14.0.9-2.el8_5.x86_64/bin/java has not been configured as
    an alternative for java
Oracle Linux Virtualization Manager VM Control Utility 4.4.1-1.1
OLVMUTIL_PASS environment variable not found.
Password:
Connected to Oracle Linux Virtualization Manager 4.4.8.6-1.0.11.el8
Setting vcpu pinning ...
Setting VM's "Start Running On: Specific Host(s)" to host kvm03.it.oracle.com
Trying to pin virtual cpu # 0
Trying to pin virtual cpu # 1
Retrieving vcpu pinning to confirm it has been set...
No CPU pinning is configured

NOTE: if the VM is running you must now stop and then start the VM from the Oracle Linux
    Virtualization Manager in order for CPU pinning changes to take effect.
NOTE: a restart or a reboot of the VM is not sufficient to put CPU pinning changes into
    effect.
```

Once you have configured a virtual machine for CPU pinning, you need to stop the virtual machine and then start it again before the CPU pinning can take effect. With the virtual machine properly restarted, you can verify that the proper vCPU pinning is in place.

- Syntax example

```
# olvm-vmcontrol -m <FQDN-OLVM-HOST> -u <USER-ID> -v <VM-NAME> -c <COMMAND> -e
```

- Execution example on a demo system

```
[root@ovirt: ~]# olvm-vmcontrol -m ovirt.it.oracle.com -u admin@internal -v ol8vm -c getvcpu
    -e
/usr/lib/jvm/java-11-openjdk-11.0.14.0.9-2.el8_5.x86_64/bin/java has not been configured as
    an alternative for java
Oracle Linux Virtualization Manager VM Control Utility 4.4.1-1.1
OLVMUTIL_PASS environment variable not found.
Password:
Connected to Oracle Linux Virtualization Manager 4.4.8.6-1.0.11.el8
Getting vcpu pinning ...
vcpu 0 pinned to cpuSet[0,1]
vcpu 1 pinned to cpuSet[0,1]
```

On the server running Oracle Linux KVM, you can also execute the `virsh` command to find out the CPU pinning status:

```
# virsh --readonly vcpuinfo ol8vm --pretty
VCPU:           0
CPU:            0
State:          running
CPU time:       5.4s
CPU Affinity:   0-1 (out of 12)

VCPU:           1
CPU:            1
State:          running
CPU time:       2.9s
CPU Affinity:   0-1 (out of 12)
```

The virtual machine now has **CPU Affinity** 0-1 for both virtual CPUs configured.

ORACLE

Conclusion

With Oracle Linux Virtualization Manager and Oracle Linux KVM, to conform to the Oracle hard partition licensing requirement, you must bind a virtual machine to physical CPUs or cores. This prevents the software from running on physical cores other than the ones specified. In such a case, virtual machines are configured with dedicated CPU resources instead of the default of resource scheduling, which is to use all available CPUs of the server. Using hard partitioning to limit Oracle software licensing also adds some restrictions to activities such as live migration and scheduling policies available on Oracle Linux Virtualization Manager.

For more information about Oracle's virtualization solutions, visit oracle.com/virtualization.

Connect with us

Call **+1.800.ORACLE1** or visit **oracle.com/linux**. Outside North America, find your local office at: **oracle.com/contact**.

blogs.oracle.com/linux facebook.com/oraclelinux twitter.com/oraclelinux