

ORACLE®

VM

Hard Partitioning with Oracle VM Server for x86

ORACLE WHITE PAPER | JULY 2016



ORACLE®

Introduction

This document describes hard partitioning with Oracle VM Server for x86, and how to use it to conform to the [Oracle licensing policies for partitioned environments](#).

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 as CPUs: the guest schedules applications on these vCPUs, and the hypervisor schedules these vCPUs over the physical CPU cores or threads. All vCPUs from a guest are symmetrical. Oracle VM Server treats these equally, as long as scheduling parameters such as using CPU pinning have not changed.

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

Oracle Hard Partition Licensing

To conform to the Oracle hard partition licensing requirement, you must follow the instructions described in this white paper to bind vCPUs to physical CPU threads or cores.

Live migration of CPU pinned virtual machines to another Oracle VM Server is not permitted under the terms of the hard partitioning license. Consequently, for Oracle VM Release 3, any servers running CPU pinned guests must not be included in DRS (Distributed Resource Scheduler) and DPM (Distributed Power Management) policies.

When live migration is used in an Oracle VM server pool, hard partition licensing 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 pool. For example, if a customer has a server pool with 32 servers and 20 virtual machines running Oracle Software within the server pool, the customer must license the 20 largest physical servers in the pool. If the customer is running 50 virtual machines with Oracle Software in a pool of 32 physical servers, they need only to license the 32 physical servers in the pool.

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

“Trusted Partitions” allow subset licensing without limitation on live migration, but only available on the approved Oracle Engineered Systems listed on [Oracle licensing policies for partitioned environments](#).

Understanding CPU Topology in Oracle VM

Get a Summary of the Server Hardware

On an Oracle VM Server, you can run the **xm info** 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.

```
# xm info
...
nr_cpus           : 8
nr_nodes          : 1
cores_per_socket  : 4
threads_per_core  : 2
cpu_mhz           : 3200
...
```

This server has a single socket with 4 cores and 2 threads per core. Total of 8 "CPUs". So CPU 0..7 is really thread 0..7.

```
# xm info
...
nr_cpus           : 12
nr_nodes          : 1
cores_per_socket  : 6
threads_per_core  : 2
...
```

This server has a single socket with 6 cores and 2 threads per core, thus there are total of 12 "CPUs".

Get the CPU Topology

The **xenpm** command prints out the thread/core/socket topology on a given server:

```
# xenpm get-cpu-topology
CPU    core    socket  node
CPU0   0        0       0
CPU1   0        0       0
CPU2   1        0       0
CPU3   1        0       0
```

The above examples show a single socket machine with 2 cores and 2 threads per core. CPU 0 is thread 0 of core 0, CPU 1 is thread 1 of core 0, CPU 2 is thread 0 of core 1, and CPU 3 is thread 1 of core 1. `cpus="0,1"` in the virtual machine configuration file (`vm.cfg`), would be running the VM on core 0. `cpus="0-3"` in the `vm.cfg`, would actually run a virtual machine on both cores.

```
# xenpm get-cpu-topology
CPU    core    socket  node
CPU0   0        0       0
CPU1   0        0       0
CPU2   1        0       0
CPU3   1        0       0
CPU4   2        0       0
CPU5   2        0       0
CPU6   3        0       0
CPU7   3        0       0
```

In the above example, you have a single socket server with 4 cores and 2 threads per core. CPU 0 maps to the thread 0 of core 0, CPU1 maps to the thread 1 of core 0, and so on. `cpus="4-7"` in the `vm.cfg` file would run the virtual machine on cores 2 and 3.

```
# xenpm get-cpu-topology
CPU      core    socket  node
CPU0     0       0       0
CPU1     0       0       0
CPU2     1       0       0
CPU3     1       0       0
CPU4     2       0       0
CPU5     2       0       0
CPU6     8       0       0
CPU7     8       0       0
CPU8     9       0       0
CPU9     9       0       0
CPU10    10      0       0
CPU11    10      0       0
```

In the above example, you see a single socket server with 6 cores with hyperthreading enabled.

Get the CPU Topology for vCPU Bindings to Physical CPUs

The **xm vcpu-list** command shows a summary of which virtual CPUs are running on which physical CPUs.

```
# xm vcpu-list 1
Name                                     ID  VCPU  CPU State  Time(s) CPU Affinity
0004fb00000600007c351fa24276c63f      1   0    5  -b-    4673.6 5-6
0004fb00000600007c351fa24276c63f      1   1    5  -b-    4534.0 5-6
```

If you add the virtual machine or domain ID to the command **xm vcpu-list 1**, you get the information for just that guest. In the above example, you have a guest with 2 virtual CPUs both running, at this time, on physical CPU (thread in this case) 5. The column **CPU Affinity** shows 5-6, which means that both virtual CPUs could be running on either thread 5 or 6. This shows that the guest is pinned on those 2 threads. Combined with the information of **xenpm get-cpu-topology** you can then see that in this case, CPU 5 is thread 1 of core 2, and CPU 6 is thread 0 of core 8. So this 2 vCPU guest is pinned to 2 separate physical cores.

```
# xm vcpu-list
Name                                     ID  VCPU  CPU State  Time(s) CPU Affinity
0004fb00000600007c351fa24276c63f      1   0    5  -b-    4676.8 5-6
0004fb00000600007c351fa24276c63f      1   1    5  -b-    4537.0 5-6
Domain-0                                0   0    0  -b-     932.1 any cpu
Domain-0                                0   1    6  -b-   1168.0 any cpu
Domain-0                                0   2    7  -b-   1010.8 any cpu
Domain-0                                0   3   11  -b-    903.0 any cpu
Domain-0                                0   4    8  -b-    494.2 any cpu
Domain-0                                0   5    9  r--    773.8 any cpu
Domain-0                                0   6    1  -b-    522.7 any cpu
Domain-0                                0   7    2  -b-    785.1 any cpu
Domain-0                                0   8    4  -b-    473.8 any cpu
Domain-0                                0   9    3  -b-    728.1 any cpu
Domain-0                                0  10   10  -b-    490.8 any cpu
Domain-0                                0  11    0  r--   1219.6 any cpu
```

This is the same system, but **xm vcpu-list** without the argument. It also shows the dom0 guest. As you can see in this example, dom0 can run on any physical thread and the CPU Affinity is **any cpu**, which implies any virtual CPU can be scheduled on any physical thread, so there is no pinning or partitioning.

Oracle VM 3: Configuring Hard Partitioning

While using Oracle VM 3, you can use the Oracle VM Utilities (ovm_vmcontrol) to set hard partitioning.

Setting Hard Partitioning Using Oracle VM 3 Utilities

You can use the Oracle VM 3 Utilities (ovm_vmcontrol) to set and get the CPU/vCPU bindings for a virtual machine through Oracle VM Manager 3.

The Oracle VM 3 Utilities are a collection of command line scripts that allow you to perform a set of basic management tasks. The Oracle VM Utilities are available for download as a .zip file from [My Oracle Support](#), search for patch ID [13602094](#). Please review the patch readme file as versions of Oracle VM utilities relate to versions of Oracle VM. Also read *Administrator's Guide* in the [Oracle VM Documentation](#).

```
# ./ovm_vmcontrol -u admin -p Manager1 -h oracle_vm_manager_hostname -v apitest -c
  getvcpu
Oracle VM VM Control utility 2.1.
Connecting to OVM Manager using Web Service.
Connected.
OVM Manager version: 3.4.1.1369
Command : getvcpu
Getting pinned CPU list...
Current pinned CPU: 5,6
```

In the above example, a virtual machine named **apitest** accepts the action **getvcpu** to show that virtual CPUs of this guest are bound to threads 5 and 6.

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

```
# ./ovm_vmcontrol -u admin -p Manager1 -h oracle_vm_manager_hostname -v apitest -c
  setvcpu -s 0
Oracle VM VM Control utility 2.1.
Connecting to OVM Manager using Web Service.
Connected.
OVM Manager version: 3.4.1.1369
Command : setvcpu
Pinning vCPU '0' to VM 'apitest'
Pinning vCPU succeed.
```

Once you have configured a virtual machine for CPU pinning, you may need to stop the virtual machine and then start it again before the CPU pinning can take effect. Now, running the **xm vcpu-list** command to find out the CPU pinning status:

```
# xm vcpu-list 1
Name                                     ID  VCPU  CPU State  Time(s) CPU Affinity
0004fb00000600007c351fa24276c63f      1   0    0  -b-    4687.6 0
0004fb00000600007c351fa24276c63f      1   1    0  -b-    4547.2 0
```

The VM now has **CPU Affinity 0** for both virtual CPUs.

In the above example we were running the 2.1ovm-utilities from an Oracle Linux host external to the Oracle VM Manager which is possible with Oracle VM 3.4 as it is fully web services based.

In the next section, we'll show how to set hard partitioning by modifying the virtual machine configuration file (vm.cfg) for Oracle VM 2.

Oracle VM 2: Configuring Hard Partitioning

Locate the virtual machine in the storage repository. For example, the virtual machine is stored in `/OVS/running_pool/directory-to-virtual-machine`. Next, modify the `vm.cfg` file for the corresponding guest or virtual machine:

```
cpus = '0-3'
```

or

```
cpus = '0,1'
```

In the first example, only CPUs 0, 1, 2, and 3 can be used for the guest. In the second example, CPUs 0 and 1 are used.

If you have a guest that has 4 vCPUs with `cpus = '0'` in the `vm.cfg` file, all 4 vCPUs will be scheduled on the same physical CPU. If you have a guest that has 4 vCPUs and you want to use 2 CPUs, then add `cpus = '0,1'` in this configuration on an 8-CPU system :

```
# xm vcpu-list guest1
Name  ID VCPU CPU State Time(s) CPU Affinity
guest1 4 0 4 -b- 8645.7 any cpu
guest1 4 1 4 -b- 9843.6 any cpu
```

The virtual machine **guest1** has 2 vCPUs and they can run on any of the 8 CPUs.

```
# xm vcpu-list guest1
Name  ID VCPU CPU State Time(s) CPU Affinity
guest1 26 0 0 -b- 8646.6 0
guest1 26 1 0 -b- 9844.3 0
```

The virtual machine `guest1` has 2 vCPUs and they can only run on physical CPU 0.

```
# xm vcpu-list guest1
Name  ID VCPU CPU State Time(s) CPU Affinity
guest1 26 0 0 -b- 8647.8 0
guest1 26 1 1 -b- 9845.0 1
```

The virtual machine `guest 1` has 2 vCPUs and they can only run on physical CPU 0, 1.

Conclusion

With Oracle VM Server for x86, 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 product software licensing also adds some restrictions such as live migration, DRS and DPM.

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






Oracle Corporation, World Headquarters

500 Oracle Parkway
Redwood Shores, CA 94065, USA

Worldwide Inquiries

Phone: +1.650.506.7000
Fax: +1.650.506.7200

CONNECT WITH US

-  blogs.oracle.com/virtualization
-  facebook.com/oracleVirtualization
-  twitter.com/ORCL_Virtualize
-  oracle.com/virtualization

Integrated Cloud Applications & Platform Services

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. This document is provided for information purposes only, and the contents hereof are subject to change without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document, and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group. 0116

Hard Partitioning with Oracle VM Server for x86
July 2016