An Oracle White Paper
Oct 2014

Hard Partitioning With Oracle Solaris Zones

Introduction

This document describes hard partitioning with Oracle Solaris Zones (also known as Oracle Solaris Containers), and how to use it to conform to the Oracle licensing policies for partitioned environments.

The approved hard partition configurations described below apply to all types of Oracle Solaris Zones, which include but are not limited to Native Zones, Kernel Zones, Oracle Solaris Legacy Containers and, Oracle Solaris 10 Zones on Oracle Solaris 11.

NOTE: the Oracle licensing document referenced above states that the acceptable way to configure Oracle Solaris Zones is using “capped Zones/Containers only”. The use of the designation “capped” is a generic way of stating the Oracle Solaris Zone has a defined upper CPU boundary which is less than all the CPUs on the system. In this instance the term “capped” refers to all valid methods of configuring Oracle Solaris Zones as hard partitions. It should not be specifically associated with the `capped-cpu` setting used by the `zonecfg` command.

CPUs, CPU Threads, CPU Cores and CPU Processors

Oracle licensing which uses a processor based metric is based on the number of cores that the applications run on. Today's servers generally contain multiple processors, which contain multiple CPU cores which themselves contain multiple CPU threads. When discussing hard partition rules it is key that these definitions are clearly understood.

Definition of Processor, Core and Thread

- Physical Processor: A physical die, chip or processor that is a single computing unit that can contain multiple cores.
- Core: Independent Central Processing Unit that can read and execute program instructions
- Thread: a hardware thread.
- Virtual processor: another name for a hardware thread.
- CPU: used in some Oracle Solaris commands this is another name for a hardware thread.

Figure 1 below shows a server with 2 processors, each with 8 cores, each core with 8 hardware threads:

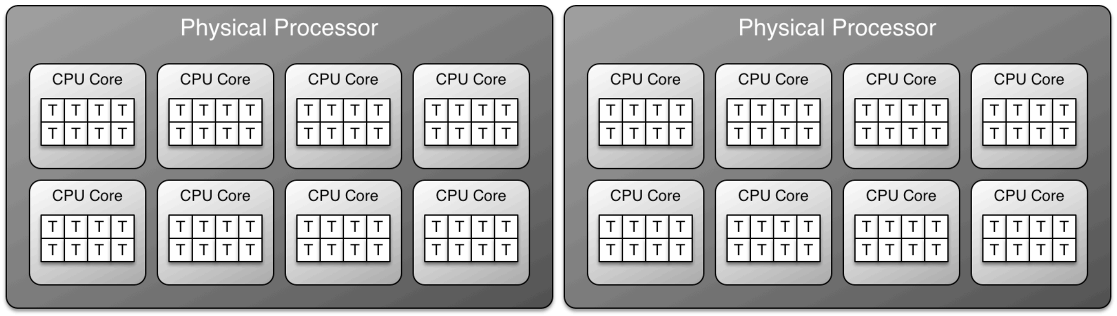

Figure 1: A system with 2 processors, 8 cores per processor and 8 threads per core

Examining the layout of your system

When implementing hard partitioning using Oracle Solaris Zones there are two key metrics it is useful to collect. The number of cores available on the server and the number of threads per core. Oracle Solaris provides tools to help get this information, the one we will examine here is `psrinfo`. Here are two examples of output from `psrinfo`, one for a SPARC server and one for an x86 server.

Below is example output from a SPARC server running Oracle Solaris 11, it shows that the server has a total of 12 cores with 8 threads per core:

```
root:~# psrinfo -pv
The physical processor has 8 cores and 64 virtual processors (0-63)
  The core has 8 virtual processors (0-7)
  The core has 8 virtual processors (8-15)
  The core has 8 virtual processors (16-23)
  The core has 8 virtual processors (24-31)
  The core has 8 virtual processors (32-39)
  The core has 8 virtual processors (40-47)
  The core has 8 virtual processors (48-55)
  The core has 8 virtual processors (56-63)
    SPARC-T4 (chipid 8, clock 2848 MHz)
The physical processor has 4 cores and 32 virtual processors (64-95)
  The core has 8 virtual processors (64-71)
  The core has 8 virtual processors (72-79)
  The core has 8 virtual processors (80-87)
  The core has 8 virtual processors (88-95)
    SPARC-T4 (chipid 65540, clock 2848 MHz)
```

Below is example output from an x86 server running Oracle Solaris 11, it shows that the server has a total of 16 cores with 2 threads per core:

```
root:~# psrinfo -pv
The physical processor has 8 cores and 16 virtual processors (0-7 16-23)
 The core has 2 virtual processors (0 16)
 The core has 2 virtual processors (1 17)
 The core has 2 virtual processors (2 18)
 The core has 2 virtual processors (3 19)
 The core has 2 virtual processors (4 20)
 The core has 2 virtual processors (5 21)
 The core has 2 virtual processors (6 22)
 The core has 2 virtual processors (7 23)
   x86 (GenuineIntel 206D7 family 6 model 45 step 7 clock 2893 MHz)
     Intel(r) Xeon(r) CPU E5-2690 0 @ 2.90GHz
The physical processor has 8 cores and 16 virtual processors (8-15 24-31)
 The core has 2 virtual processors (8 24)
 The core has 2 virtual processors (9 25)
 The core has 2 virtual processors (10 26)
 The core has 2 virtual processors (11 27)
 The core has 2 virtual processors (12 28)
 The core has 2 virtual processors (13 29)
 The core has 2 virtual processors (14 30)
 The core has 2 virtual processors (15 31)
   x86 (GenuineIntel 206D7 family 6 model 45 step 7 clock 2893 MHz)
     Intel(r) Xeon(r) CPU E5-2690 0 @ 2.90GHz
```

Creating Oracle Solaris Zones that meet hard partition requirements

Oracle Solaris Zones resource management is highly flexible and can be configured in many different ways to meet the many requirements of customer applications and infrastructure. A full discussion of the types of resource management available are beyond the scope of this document but more information can be found in the Oracle Solaris Zones documentation found here.

To meet the requirements of hard partitions three Oracle Solaris Zone resource management methods are valid. These are:

- Use of the `dedicated-cpu` setting in `zonecfg`
- Use of the `capped-cpu` setting in `zonecfg`
- Use of a resource pool with a fixed set of assigned hardware threads and adding one or more Oracle Solaris Zones to that resource pool.

The following figure illustrates these methods:

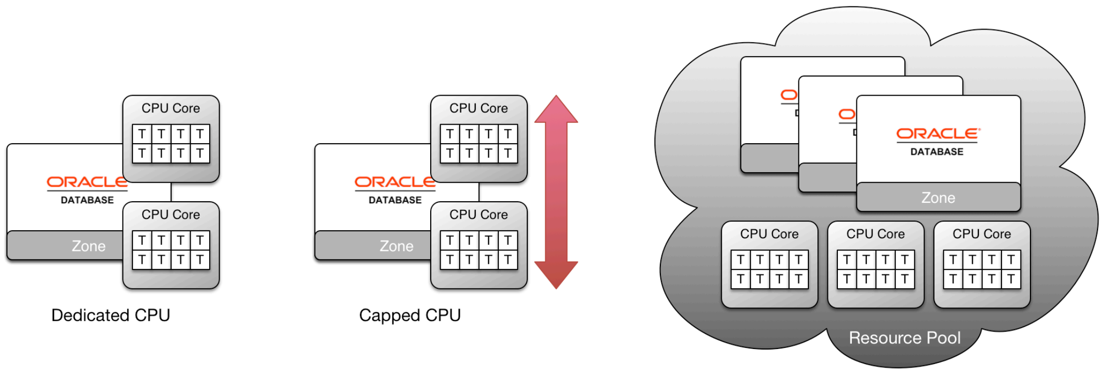

Figure 2: The three types of approved resource management for hard partitions.

Dedicated CPU assignment

The `dedicated-cpu` property of `zonecfg` defines the number of hardware threads (also known as CPUs or virtual processors) to give exclusively to a particular zone. Use `zonecfg` to set this value for a zone as follows, note in this case the zone has already been created:

```
root:~# zonecfg -z dedicated-zone
zonecfg:dedicated-zone> add dedicated-cpu
zonecfg:dedicated-zone:dedicated-cpu> set ncpus=3
zonecfg:dedicated-zone:dedicated-cpu> end
zonecfg:dedicated-zone> verify
zonecfg:dedicated-zone> commit
zonecfg:dedicated-zone> exit
```

To check or validate the number of `dedicated-cpus` assigned to the zone use the following:

```
root:~# zonecfg -z dedicated-zone info dedicated-cpu
dedicated-cpu:
         ncpus: 3
```

Note that the number of `dedicated-cpus` (actually hardware threads) is shown by the `ncpus` value, in this case it is three CPUs (hardware threads). Also note that the `ncpus` value can take a range, if a range is used then the maximum value must be used for licensing purposes.

Following the method above creates a valid Oracle Solaris Zone hard partition environment.

Capped CPU assignment

The `capped-cpu` property of `zonecfg` defines the number of hardware threads (also known as CPUs or virtual processors) which a given zone must not exceed. Use `zonecfg` to set this value for a zone as follows (note in this case the zone has already been created):

```
root:~# zonecfg -z capped-zone
zonecfg:capped-zone> add capped-cpu
zonecfg:capped-zone:capped-cpu> set ncpus=3
zonecfg:capped-zone:capped-cpu> end
zonecfg:capped-zone> verify
zonecfg:capped-zone> commit
zonecfg:capped-zone> exit
```

To check or validate the number of `capped-cpus` specified to the zone use the following:

```
root:~# zonecfg -z capped-zone info capped-cpu
capped-cpu:
          [ncpus: 3.00]
```

Note that the upper limit of `capped-cpus` (actually hardware threads) is shown by the `ncpus` value, in this case it is three CPUs (hardware threads). Also note that the `ncpus` value for a `capped-cpu` does not have to be an integer and may include a fraction of a CPU.

Following the method above creates a valid Oracle Solaris Zone hard partition environment.

Resource Pools with Oracle Solaris Zones

The final approved method to create an approved hard partition is to create a resource pool with a set number of CPUs and add one or more zones to that resource pool. This has one advantage over the previous two methods – multiple zones can effectively share a set of CPUs in a hard partition.

Perform the following commands to create a valid hard partition based on resource pools.

Enable resource pools:

```
root:~# pooladm –e
```

Create a processor set (`pset`) with the number of desired CPUs (hardware threads), in this case 16:

```
root:~# poolcfg -dc 'create pset orapset'
root:~# poolcfg -dc 'modify pset orapset (uint pset.max=16)'
root:~# poolcfg -dc 'modify pset orapset (uint pset.min=16)'
```

Create a pool to contain this pset, and associate the pset with it:

```
root:~# poolcfg -dc 'create pool orapool'
root:~# poolcfg -dc 'associate pool orapool (pset orapset)'
```

Update the server resource pool settings with this configuration:

```
root:~# pooladm -s
```

Finally associate any zones with this resource pool hard partition (in this example two zones, `orazone1` and `orazone2` which have previously been created):

```
root:~# zonecfg -z orazone1 set pool=orapool
root:~# zonecfg -z orazone2 set pool=orapool
```

Note that in this example `pset.max` and `pset.min` were set to the same value, it is permissible to have different values (i.e. a range). In the case of having a range the `pset.max` value would be used to calculate the number of CPUs (hardware threads) for licensing purposes.

To check or validate the resource pool settings do the following:

For each zone that is a hard partition examine the pool association of the zone, in the output below it is orapool:

```
root:~# zonecfg -z orazone1 info pool
pool: orapool
```

Next examine the `poolcfg` information of the `orapool` setting for `pset`, in this example it is `orapset`. By then examining `orapset` it shows that `pset.max` is set to 16, this is the number of CPUs (or hardware threads) that must be counted for licensing purposes:

```
root:~# poolcfg -c 'info pool orapool'

pool orapool
         int       pool.sys_id 2
         boolean   pool.active true
         boolean   pool.default false
         int       pool.importance 1
         string    pool.comment
         pset      orapset

         pset orapset
                   int       pset.sys_id 2
                   boolean   pset.default false
                   uint      pset.min 16
                   uint      pset.max 16
                   string    pset.units population
                   uint      pset.load 0
                   uint      pset.size 0
                   string    pset.comment
```

Following the method above creates a valid Oracle Solaris Zone hard partition environment.

Checking the number of cores used in a Oracle Solaris Zones hard partition environment

Each of the valid methods for creating a valid Oracle Solaris Zone hard partition environment assigns CPUs (hardware threads), it is necessary to convert this to cores (or in certain cases sockets) for the Oracle licensing policy.

To do this add up all the CPUs on the server that are assigned to Oracle Solaris resource pools and Oracle Solaris Zones that are to be used as hard partitions. Note when using ranges for CPU assignments always use the maximum number for license calculations. Divide this total CPU count by the number of threads per core for the system (or sockets per core for socket values). The number of threads per core can be calculated using `psrinfo –pv` (see earlier section for examples).

How to verify the number of CPUs (hardware threads) assigned is shown earlier in this document but for convenience is also summarized below:

Dedicated CPU

To check or validate the number of `dedicated-cpus` assigned to the zone use the following and note the `ncpus` setting:

```
root:~# zonecfg -z dedicated-zone info dedicated-cpu
dedicated-cpu:
         ncpus: 3
```

Capped CPU

To check or validate the number of `capped-cpus` specified to the zone use the following and note the `ncpus` setting:

```
root:~# zonecfg -z capped-zone info capped-cpu
capped-cpu:
         [ncpus: 3.00]
```

Resource Pool

For each zone that is a hard partition examine the pool association of the zone, in the output below it is orapool:

```
root:~# zonecfg -z orazone1 info pool
pool: orapool
```

Next examine the `poolcfg` information of the `orapool` setting for `pset.max` is the value to note:

```
root:~# poolcfg -c 'info pool orapool'

pool orapool
         int      pool.sys_id 2
         boolean  pool.active true
         boolean  pool.default false
         int      pool.importance 1
         string   pool.comment
         pset     orapset

         pset orapset
                  int      pset.sys_id 2
                  boolean  pset.default false
                  uint     pset.min 16
                  uint     pset.max 16
                  string   pset.units population
                  uint     pset.load 0
                  uint     pset.size 0
                  string   pset.comment
```

Conclusion

Oracle Solaris Zones can be used as valid Oracle hard partition environments when configured using the `zonecfg dedicated-cpu` setting, `zonecfg capped-cpu` setting or as part or a resource pool with specific CPUs assigned. When calculating the number of cores required to be licensed, the total number of CPUs (which equates to hardware threads) assigned to the hard partition environments is divided by the hardware thread per core ratio. This will give the number of cores that are used and need to be licensed.

For more information or in the case where there is any doubt about the proposed configuration contact the appropriate Oracle Account representative or Oracle License Management.

Oracle Hard Partitions with Oracle Solaris Zones
October 2014
Author: Duncan Hardie

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200

oracle.com