Utilizing Solaris Containers with the Oracle Database

Björn Rost
portrix systems GmbH
Hamburg

Keywords:

Solaris Container, Database, virtualization

Introduction

Virtualization, green computing, and server consolidation have been major trends for datacenters in recent years. This paper explains the reasons and motivation for using virtualization and explores technology and implementation strategies for deploying applications in Oracle Solaris Containers or zones. It focuses especially on running Oracle Database inside containers, including how to make use of hard partitioning to save initial costs while retaining the option to scale when needed. Other topics include zone migration and upgrades.

Motivation

Traditionally, applications have been deployed on a single or multiple servers. When installing multiple products or application tiers on a single server, extra care had to be taken to make sure that all software components are compatible and do not interfere with each other. Quite often, this led to each tier of an application (database, application server, webserver...) being deployed on a different machine.
Over time, the evolution of hardware has outgrown software demands in many cases, resulting in machines that are running with very low utilisation. Together with increasing energy costs to power and cool those machines this is driving the need to consolidate different services or applications on fewer servers. Today, lots of virtualization solutions exist and it is predicted that by 2012 about 50% of all application workloads will be running on virtualized environments.
An additional incentive to use virtualisation exists in terms of Oracle licensing costs. With Oracle Enterprise Edition, one has to license according to the number of CPUs or Cores in the database server. Hardware manufacturers have been increasing the number of cores in a single CPU to six or eight cores per CPU. Quad core servers are still available in 2010 but might not be in the future. So licensing Oracle EE on a single server will be more and more expensive as manufacturers manage to put even more cores on chips. One solution to this problem is to use a virtualisation technology that is recognized by Oracle as „hard partitioning“, enabling you to bind only a subset of CPUs to a virtual environment and paying license fees only for actually used CPUs[1].
Running the database, application server and webserver on seperate servers can also reduce the overall availability of a platform because an outage of a single component will make the whole system unusable. There also exists the opportunity to take advantage of virtual machine backup and recovery for desaster scenarios. It is usually easier to restore a virtual machine image on another server than to perform a bare metal recovery of any given system.

1 http://www.oracle.com/corporate/pricing/partitioning.pdf

Overview of solutions

There are many different vendors and solutions around server virtualization. VMware propably has the greatest market share but when running Oracle workloads on it one should know that Oracle does not certify its products against VMware and requires that customers proove that new bugs are also reproducable on a not virtualized platform.[2] This has created some confusion and uncertainty but the bigger problem with Oracle on VMware machines is that they are not recognised as hard partitioning which means that customers have to pay for all cores or CPUs in the VMware server even if only a subset is actually used for or assigned to the specific virtual machine running the Oracle database.
Oracle is pushing its own product Oracle VM which is a hypervisor solution based on XEN. You have a broad choice of different guest operating systems and it can be configured to be recognised as a hard partitioning virtualization solution. It also has many other useful features like GUI support for deployment and management of your machines and a solution called omotion that allows you to transfer a running virtual machine between servers without any noticable downtime. A downside to Oracle VM is that it requires you to have a dedicated machine for the VM manager which makes this solution unsuitable for very small deployments.
Oracle Solaris Containers is another lightweight virtualization solution that fits very well into a complete Oracle stack. This architecture does not require extra software or a special hypervisor. Each container is merely another execution environment under a single kernel. But each zone features its own process and namespace, IP-address and root account and filesystem. Solaris also has a lot of mechanisms to control the allocation of system resources like CPU und memory between these zones.

Theory of operation

A few terms and concepts need explanation to understand the fundamentals of Oracle Solaris Zones and Containers. There are a many features and options. I am going to mention most of these but focus on the most common setup.
Even a fresh Solaris install will feature a zone, the so called global zone. This is the default or master zone with access rights to all other zones. All other zones are called local zones and this will be just referenced as 'zone' later on. They are also sometimes referred to as non-global zones. There is also a third type of zone called 'branded zones'. This type allows you to run an older version of Oracle Solaris or Linux inside a virtualization layer.
During zone installation, you can choose between a sparse root or a whole root system. The difference is that with sparse root zones, certain file system branches (like /usr and /lib) are mounted read-only from the global zone into the local zone. This saves disk space since most binaries will be redundant between the global and local zones anyway. Full root zones have own copies of all files. They take up more space on disk but add the flexibility to install software to directories that would otherwise be read-only and to apply some patches only to a local zone.
The term 'container' is often interchanged with 'zone' but this is not totally accurate. A container is a local zone that has some resource manager restrictions active. One can limit the number of CPUs or maximum memory that can be assigned to any given zone and that is what makes a zone a container. This is very important when relying on resource management to limit CPU cores for licensing issues.
A zone can and will be in a number of different states during its lifetime. After the first setup with 'zonecfg', it will be in the 'configured' state. After this, the command 'zoneadm' will be used to transfer the zone to other states. The next logical one being 'installed' from which the zone can be booted (running) or halted (ready). Zones in the ready state can be cloned to create another zone on the same server. A zone can also get detached which means that a file containing zone descriptions is added to the zone's filesystem. It can then be transferred to another Solaris system by simply copying the filesystem. On the new system, the zone can easily be attached and booted. The only requirement for

2 See metalink note #249212.1

this form of zone migration is that the new server must have the same or newer Solaris version and patches.
Using ZFS as a root filesystem enables one to make use of even more features. With this setup, each zone has it's own ZFS filesystem. There have been numerous problems with this in early Solaris releases, especially with updating and upgrading but the technology has matured a lot and can now be considered to be safe to use for production systems.
Using ZFS as the root filesystem for both the global and non-global zones enables you to perform live-upgrade to safely patch your system with minumum downtime and minumum risk because you will generate a clone of your root filesystems, patch these while the system is still online and then enable all patches with a simple reboot. If anything broke during patching, the old image can still be rebooted to recover from faults. Also, when cloning a zone with the zonepath being a ZFS filesystem, cloning operations are actually ZFS clones instead of filesystem copies. This means a huge improvement both on the time it takes to clone and the space used.
ZFS snapshots can also be used for backing up zones. A snapshot is a very light weight method to freeze the state of a filesystem at one point in time. This can be used for logical backups but can also written to other servers or tape. This can also complement your Database backup strategy if you are willing to put your datafiles on ZFS. This is especially useful in test and development databases and can be a cheap and easy way to implement features similar to Oracle's flashback database with no additional licensing costs. If I/O performance is a concern, raw partitions or LUNs can also be mapped to local zones to set up ASM on them.
The last option you have when configuring zones is the ip setup. The traditional method is called shared IP. With this model, the IP stack of the global zone is also used in the local zone, just with another virtual IP address on the same interface as the global zone. The downside to this model is that it does not allow you to use different routing than the global zone, perform snooping on the interface or be used as a DHCP server.
The alternative is called 'private IP'. With this model, a local zone is assigned a full (spare) network interface. This way, the local zone can have it's own network stack, routing table and deliver services that were otherwise not possible in a local zone.

With all these options explained, this are the options most widely used for oracle databases at portrix:

- Oracle Database, app-server, webserver each in their own local zone
- sparse root zones with the default read-only filesystem (install oracle to /opt)
- use resource manager to cap number of CPU cores
- everything on ZFS, except for I/O performance hungry applications
- shared IP

setup/example

This is some example code that will set up a zone from scratch to get you ready to install the Oracle RDBMS in it. The first step is using zonecfg to set all options and the zonepath. In this example, a zone calles 'ora11g' is created, the zonepath set to /zp03/zones/ora11g (a zfs filesystem on my system), auto-boot enabled, limited to one CPU core (this makes the zone a container) and an IP address.

```
root@hermes:~# zonecfg -z ora11g
ora11g: No such zone configured
Use 'create' to begin configuring a new zone.
zonecfg:ora11g> create
zonecfg:ora11g> set zonepath=/zp03/zones/ora11g
zonecfg:ora11g> set autoboot=true
zonecfg:ora11g> add capped-cpu
```

```
zonecfg:ora11g:capped-cpu> set ncpus=1
zonecfg:ora11g:capped-cpu> end
zonecfg:ora11g> add net
zonecfg:ora11g:net> set address=192.168.42.79
zonecfg:ora11g:net> set physical=e1000g0
zonecfg:ora11g:net> end
zonecfg:ora11g> info
zonename: ora11g
zonepath: /zp03/zones/ora11g
brand: native
autoboot: true
bootargs:
pool:
limitpriv:
scheduling-class:
ip-type: shared
inherit-pkg-dir:
        dir: /lib
inherit-pkg-dir:
        dir: /platform
inherit-pkg-dir:
        dir: /sbin
inherit-pkg-dir:
        dir: /usr
net:
        address: 192.168.42.79
        physical: e1000g0
        defrouter not specified
capped-cpu:
        [ncpus: 1.00]
rctl:
        name: zone.cpu-cap
        value: (priv=privileged,limit=100,action=deny)
zonecfg:ora11g> verify
zonecfg:ora11g> commit
zonecfg:ora11g> exit
```

The zone is now configured but the zonepath is empty (it actually does not even exist yet). The next step populates the zonepath with all the needed files.

```
root@hermes:~# zoneadm -z ora11g verify
WARNING: /zp03/zones/ora11g does not exist, so it could not be verified.
When 'zoneadm install' is run, 'install' will try to create
/zp03/zones/ora11g, and 'verify' will be tried again,
but the 'verify' may fail if:
the parent directory of /zp03/zones/ora11g is group- or other-writable
or /zp03/zones/ora11g overlaps with any other installed zones.
```

```
root@hermes:~# zoneadm -z ora11g install
A ZFS file system has been created for this zone.
Preparing to install zone <ora11g>.
Creating list of files to copy from the global zone.
Copying <147436> files to the zone.
Initializing zone product registry.
Determining zone package initialization order.
Preparing to initialize <1504> packages on the zone.
Initialized <1504> packages on zone.
Zone <ora11g> is initialized.
Installation of these packages generated errors: <SUNWapbas>
Installation of <20> packages was skipped.
Installation of these packages generated warnings: <STATsrv>
The     file     </zp03/zones/ora11g/root/var/sadm/system/logs/install_log>
contains a log of the zone installation.
```

The zone is now installed and ready to be booted. After the first boot, you can log on to the zone with the zlogin command.

```
root@hermes:~# zoneadm -z ora11g boot
root@hermes:~#
root@hermes:~#
root@hermes:~# zlogin -C ora11g
```

Now, you get all the normal postinstall questions including languages, hostname, root password, timezone and so on. After this, your zone is ready to go and will in most ways behave like any other solaris installation.

Known issues

Zones are not inheriting the netservices setting of the global zone. We usually go with the limited services option which leaves only the SSH service enabled. But on a freshly installed local zone, many services are enabled, including telnet. Use the command 'netservices limited' to switch things back to normal. Also, the webconsole service can likely be turned off for local zones to save resources.

The Oracle RDBMS requires some kernel or system parameters to be set. This poses some problems with zones since local zones do not have their own kernels. So all system parameters must be set in the global zone to take effect. Still, the Orace installer will check for these parameters in the /etc/system file of the zone. It is safe to just copy all settings to that file, it will be ignored by Solaris and the database installer will be happy to find the 'right' settings there.

The root.sh script will try to copy some binaries to /usr/local/bin. Since /usr is read-only for sparse-root zones, you have to choose a different directory for these files like /opt/oracle/bin or similar.

When running multiple zones on a system there are a few things to think about. First of all, there will be an additional need for memory with each running zone. Other than that, one should be aware of I/O contention and the interaction between zones. Ideally, Oracle datafiles should be on a disk system that is independent from the disks of other zones. Otherwise, tracking down performance issues can get very complicated.

Conclusion

Oracle solaris zones and containers are easy to configure and lightweight virtualization tools. They allow for different parts of an application to run in seperate security context and provide a useful means for resource management both for licensing issues and performance management. The ZFS filesystem further enables you to perform very fast and deduplicated clones and snapshots that can be used for cloning whole systems or rolling back all changes made within a zone to a previous snapshot. While other virtualization solutions might offer the same or even more features, zones do not require any extra software installation or support contracts since it is a part of Oracle Solaris.

Kontaktadresse:

Björn Rost
portrix systems GmbH
Stresemannstr. 375
D-22761 Hamburg

Telefon:	+49 (0) 40-398 05319
Fax:	+49 (0) 40-398 05329
E-Mail	b.rost@portrix-systems.de
Internet:	www.portrix-systems.de