ORACLE

SOLARIS

An Oracle White Paper
September 2010

Best practices for deploying Oracle RAC inside Oracle Solaris Containers

ORACLE

Executive Overview

Enterprises today are faced with increasing administration, maintenance, space and power costs due to server sprawl, as organizations typically deploy new applications on its own server. Consolidation is one approach that can help to contain server sprawl and reduce costs. At the same time, consolidation solution must also provide flexibility, performance, resource management, ease of administration and maintenance. Oracle Solaris operating system has built-in virtualization technology called Solaris Containers. Oracle Solaris Containers consists of several technologies that work together to foster improved resource management and isolate the environment from underlying OS. With the recent certification and support for Oracle RAC on Solaris Containers, enterprises can realize the benefit of deploying RAC cluster on same nodes as their applications or multiple RAC clusters on consolidated hardware for various lines of businesses while leveraging common hardware or quickly procure and deploy RAC nodes as demand picks up.

The following key areas are explored in this document:

- Oracle Real Application Cluster (RAC) and its consolidation options and best practices, to have an optimal resource utilization based on the business demands.
- Manage multiple deployments of Oracle RAC (different versions) on the same system.

This is achieved by seamless integration of Oracle RAC with Oracle Solaris Containers. Oracle RAC adjusts itself to the new resource levels to offer maximum availability and scalability within the Solaris Container environment.

Introduction

Oracle Solaris Containers

An integral part of the Oracle Solaris 10 Operating System, Oracle Solaris Containers isolate software applications and services using flexible, software-defined boundaries and allow many private execution environments to be created within a single instance of Oracle Solaris 10. Each environment has its own identity, separate from the underlying hardware. Each behaves as if it is running on its own operating system making consolidation simple, safe, and secure.

Containers can all share CPU resources, can each have dedicated CPU resources, or can each specify a guaranteed minimum amount of resources as well as a maximum. Memory can be shared among all Oracle Solaris Containers, or each can have a specified memory cap. Physical I/O resources such as disk and network can be dedicated to individual Oracle Solaris Containers, shared by some, or shared by all. Regardless of what is shared or dedicated, each virtualized environment will have isolated access to local file system and networking, as well as system and user processes.

Ideal for environments, that consolidates a number of applications on a single server. The cost and complexity of managing numerous machines make it advantageous to consolidate several applications on larger, more scalable servers. Enable more efficient resource utilization on your system. Dynamic resource reallocation permits unused resources to be shifted to other Oracle Solaris Containers as needed. Fault and security isolation mean that poorly behaved applications do not require a dedicated and under-utilized system.

There are two ways to create Oracle Solaris Containers. One of them is 'sparse root' Container, where the root file system of the Container is mounted as read-only from the global zone, occupies less space on the file system and its quick to create. Second one is 'whole root' Container, where the root file system is mounted in read-write mode, all the packages required for Container are installed inside it. The whole root Container is like a typical system. There are two types of networking available for Containers. One of which is 'shared-IP' where the NIC is shared with the global zone, with the shared-IP can not plumb IP address within Container, only at the time of booting configured IPs of Container are brought up. For 'exclusive-IP' type of network a dedicated NIC is assigned and the IP to this NIC can be configured within a Container.

Oracle Solaris Containers is supported on T-series, M-series and x86 systems running Solaris 10 operating system.

Oracle RAC database

Oracle RAC database is a RDBMS database, which is highly available and scalable. It can dynamically adjust to dynamic resource changes in the environment. Oracle RAC's data files are hosted on shared storage, be it NAS or SAN based storage connected to all the available systems. It's recommended to use low latency high throughput private interconnect for Oracle RAC inter node communication. Minimum 2 physical systems are required to provide high availability, to mitigate failure of one node. However, for increased scalability and availability add additional physical server or node.

To host Oracle RAC in Oracle Solaris Containers environment, following points to be considered,

- Provision one Oracle Solaris Container per node or server.
    - By this high availability is achieved per instance or node level, as if one node goes down the other instance is available from other node and not the same.
    - Consolidate multiple such Oracle RAC databases within one set of physical servers.
    - In addition to that Oracle Solaris Container is being highly secure and isolated environment, these environments could be assigned as independent nodes to different DBAs managing different Oracle RAC database installations with the Oracle Solaris Container root access.
- Name space isolation offers, configuration of different Containers of different Oracle RAC database to be in different time zones on a given server, however, it's practiced that the same time zone is configured across all the other nodes hosting that particular Oracle RAC database.
    - Name space isolation offers, to configure these contained environments as independent systems by configuring them to different name server, be it DNS or NIS etc.
    - Independent process tree per Container offer process level isolation, there by a processes running in one Container will not have any impact on the other

Container in the event of a crash and reboot by that process on that environment reboots other Containers are untouched.

- To host Oracle RAC database binaries use ZFS file system. Or by creating Containers environment on ZFS and host database binaries on it. Gives great advantages like:
    - Faster deployment of Oracle binaries and setting up of the Container environment by cloning the ZFS file system hosting it.
    - Dynamically increase the file system size transparently, when the file system is near full.
- To host the Oracle RAC database data files leverage ASM, as it offers
    - Greater availability of the LUNs achieved by using redundancy. ASM does mirror disks or LUNs across controllers there by the availability increases.
    - Most optimum way to retrieve and store the data, as Oracle has better understanding of data.
- Flexibility to change the CPU resources based on the requirement could be achieved by the host administrator, to accommodate workload needs.

Deploy Oracle RAC without Virtualization

To host Oracle RAC without virtualization, following is considered,

- Need minimum 2 to 3 nodes for high availability, Shared storage to host data files, and the required private and public networking for this Oracle Cluster environment.
- However there is no scope of hosting other version or different patch level of Oracle RAC on the same set of systems.
- Though the over-all CPU and other resource utilization might be just 20 to 25%. To host another setup with a different version another set of systems needs to be configured, installed and managed.

The below drawing 1 shows that,

- There are 4 * Sun SPARC T5220 servers, configured in the RAC environment, hosts Oracle 10gR2 RAC database with 2 different databases on global zone or host's operating system environment.
- There are 2 'Sun StorageTek 6140' array with redundant controllers. Each redundant controller is connected to redundant HBAs on the systems. On the system, multipath IO (MPxIO) is used to provide high availability of connectivity to the LUNs on the disk array.
  - There are 2 color codes used to show 2 different controllers,
  - From each storage controller, say storage1, cables would be connected to port1 on slot4 and slot5. Same with storage2 however port02 is used.
  - MPxIO enables HA among slot4 and slot5 and one disk name is provided instead of two.
  - Oracle Automatic Storage Management (ASM) normal disk group is used for normal redundancy among 2 LUNs of port1 and port2, it provides HA among the failure of entire storage array.

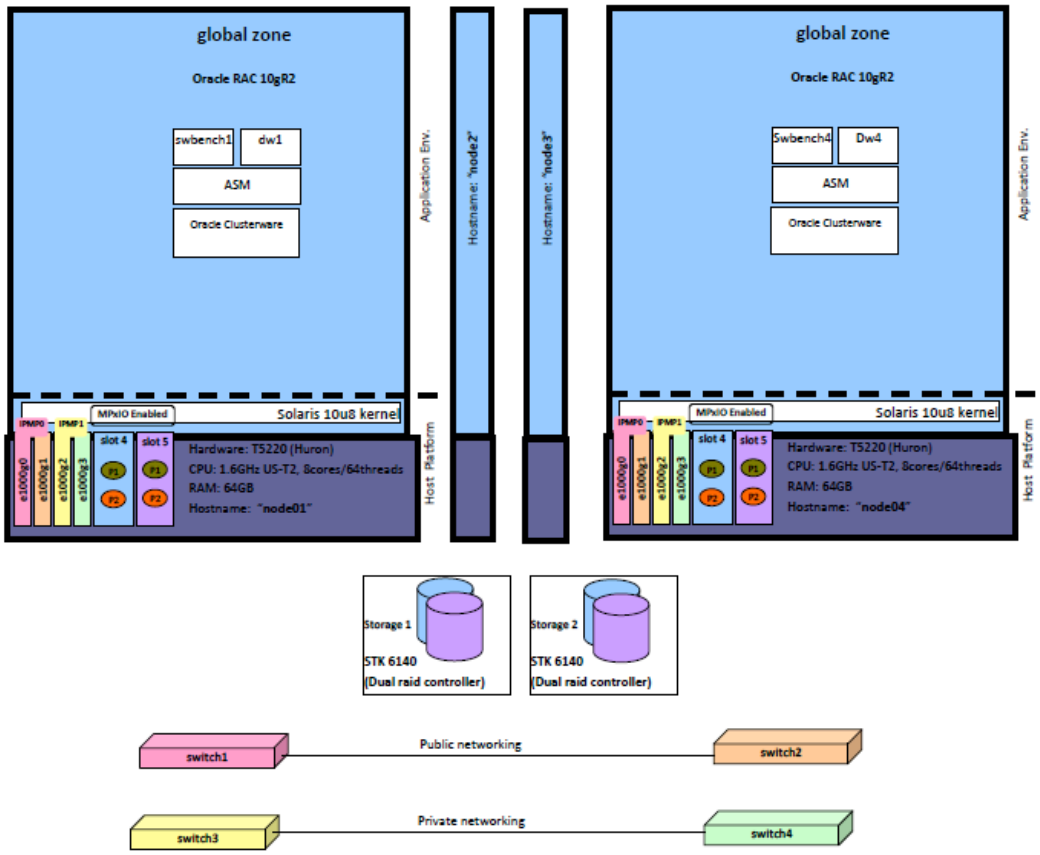

Figure 1: Oracle RAC deployment in non-virtual environment

- Public network cables are connected to switch1 and switch2 from e1000g0 and e1000g1, an IPMP group ipmp0 provides HA for the public network to hosts IP and VIP IP addresses.
  - These switches are connected to the public network,in the lab environment they can have an uplink port configured.
- Private network cables are connected to switch3 and switch4 from e1000g2 and e1000g3, an IPMP group ipmp1 provides HA for the private network on the host, and the priv IP address is configured and brought up as the host comes up, and the ipmp1 group also comes-up along with that.
  - Oracle RAC leverages this IP address as its private IP address, when the Oracle clusterware comes up.
  - These switches have an up-link port configured as trunk port to pass the traffic in case of a failure of one of the switches components the link fails-over transparently to other switch.

Oracle RAC Deployment with Virtualization

The virtualized environment is Oracle Solaris Containers, a built-in technology as explained earlier. To deploy Oracle RAC inside Oracle Solaris Containers extend the learning from the previous chapter as the high level deployment inside this virtual environment is seamless. Let us consider the following while deploying Oracle RAC inside Oracle Solaris Containers.

The below drawing 2 is a typical deployment scenario, that shows,

- There are 4 nodes, hosting 2 Container environments, running 2 different versions of Oracle RAC database 10gR2 and 11gR1 on 4 different Oracle clusterware inside Containers.
  - This shows that one container is created per node for a given Oracle clusterware environment, there by HA of instances is achieved.
  - At the same time, other container environment on a given failed node also goes down, will not have major impact as only one Container or virtual node per cluster is not available.
- These cont10g01 to cont10g04, and cont11g01 to cont11g04, are hosted on physical nodes port01 to port04 respectively.
  - Two cores are assigned (16 threads) per Container.
- The public and private networking shares the same set of hardware NICs among 2 Oracle Clusterware environments without impacting each other by using VLAN tagged NICs. Since Oracle clusterware plumbs the VIP, exclusive-IP type Containers are created and VLAN tagged NICs are assigned instead of physical NICs.
  - NICs e1000g0 and e1000g1 are connected to switch1 and switch2.
    - Ports of these switches are configured as trunk ports to allow VLAN traffic with VLAN tags 131,132 for Oracle 10gR2 and Oracle 11gR1 environments.
    - Are public NICs, VIP is hosted on it. Plumbed by vip service of Oracle clusterware.
    - For example, inside cont10g01 Container these NICs are e1000g131000 and e1000g131001, an IPMP group ipmp0 is created to provide HA at this network layer.
  - NICs e1000g2 and e1000g3 are connected to switch3 and switch4.
    - Ports of these switches are configured as trunk ports to allow VLAN traffic with VLAN tags 111,112,113 for Oracle 10gR2, Oracle 11gR1, and Oracle 11gR2 environments.
    - Private IPs are configured by Container and Oracle clusterware.
    - For example, inside cont10g01 Container these NICs are e1000g111002 and e1000g111003, an IPMP group ipmp1 is created to provide HA at this network layer.
  - For simplification, same IPMP group names could be used across all the Container environments of a given Clusterware.

- A set of dedicated Storage LUNs are assigned per Oracle clusterware environment, and the same set of LUNs are configured for other Containers across all the nodes. Physical connectivity is same as explained in the previous chapter.

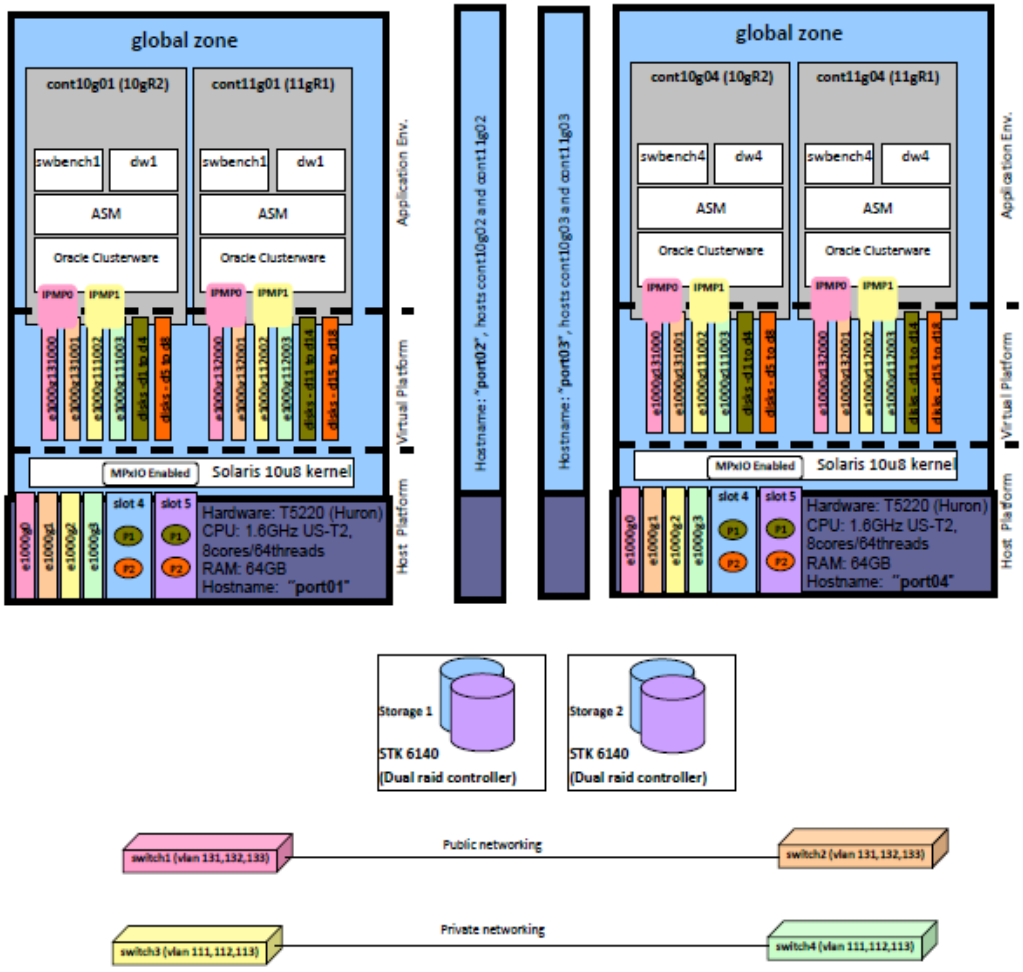

Figure 2: Oracle RAC deployment in virtual environment

This is a high level deployment environment, for more detailed planning and considerations on storage, networking, HA, etc let's look at the next chapter.

Details of Containers for Oracle RAC

To host multiple Oracle RAC clusters on Oracle Solaris Containers need careful planning, to utilize the resources optimally. Let's explore storage, networking, HA features along with instant deployment etc.

Storage

Storage Configuration

Let's look at the storage configuration details in a bit more detail, and how the HA is achieved for storage and what various components play what type of role.

The below diagram-3 shows that the storage STK 6140, has redundant controllers, and the system also has dual F-CAL HBA with dual ports per node, the physical connectivity of a given node is as follows:

- storage1 controller1 to slot4 port1
- storage1 controller2 to slot5 port1
- storage2 controller1 to slot4 port2
- storage2 controller2 to slot5 port2

The LUNs created per storage are available from both the controllers; hence failure of a HBA on slot4 or failure of a cable on that path will not have any impact on the availability of the LUN itself. This HA is achieved using MPxIO, a built in feature that comes along with Oracle Solaris. This multi-path enabled LUN has only one disk name, and this disk is provisioned to Container environment.

Assigned such dedicated LUNs for dw (data ware house) database and swingbench (swbnech) database per cluster. Configure the same set of LUNs across all the nodes and their respective Containers. When the same set of LUNs are available from all the Containers, Oracle RAC recognizes them as shared storage. With this now ASM configuration becomes seamless as that of configuring ASM disk group on the global zone environment itself.

For optimal IO performance, there could be a requirement to share all the available disks to create the LUNs on the storage box, when the workload is divided based on the time, for example when the RAC database is being accessed by 2/3 different applications, at 2/3 different timings of the day, it's good to share the disks to get higher rate of IO. However if the workload of these 2/3 applications is concurrent and heavy on IO at the same time, it's good to have LUNs created on dedicated set of disks

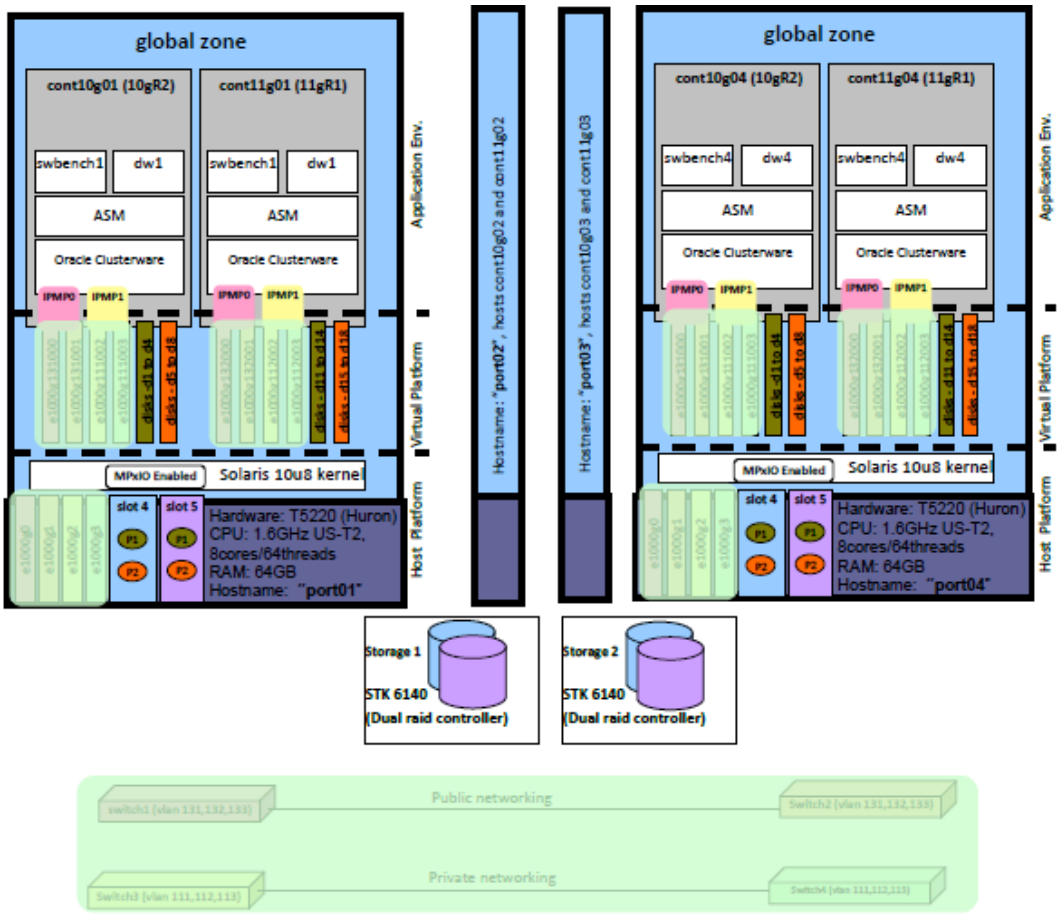

Figure 3: Storage configuration

on the storage.

ASM configuration

2 Such MPxIO enabled LUNs one from storage1 and another from storage2 are used to create ASM disk group with normal redundancy, which is mirroring. ASM's offers mirroring of disks/LUNs across controllers and enables same across nodes seamlessly as clustered volume to host Oracle RAC database. ASM comes by default with the Oracle RAC. For higher level of availability ASM could have 3 levels of redundancy configured.

Since the LUNs are dedicated to each Container's Oracle cluster environment, need to configure separate ASM instances on all clusters with its own dedicated set of disk-groups. These disk groups are created on dedicated LUNs as detailed above.

Networking

Public as well as private networking of Oracle RAC inside Container environment uses IPMP to provide high availability at the networking stack. Any accidental removal of the virtual NIC or un-plumb of the NIC will not have impact on other Container environments on the same node. IPMP monitors and takes care of seamless availability of the NIC within a Container.

The below drawing 4 shows that, there are 4 physical switches used to provide high availability at both the private and public networking layers of Oracle RAC in a typical environment be it 2 or 4 nodes or more number of nodes. 2 switches are used for public networking and 2 are used for private networking.

- Oracle CRS/Clusterware needs to have the control of public network interfaces so that it can operate the VIP resources, specifically plumb/unplumb an IP address on a network interface. That control is only possible with exclusive-IP type.
- As its virtual environment VLAN tagged NICs are created. Switches also needs to be configured to allow those VLAN tagged traffic.
- Vlan tagged NICs are used for both private as well as public networking.
  - switch1 and switch2 are used for public networking, connected to the public network. e1000g0 and e1000g1 NICs are connected to switch1 and switch2 respectively.
  - switch3 and switch4 are used for private networking, connected to the cluster's private networking. e1000g2 and e1000g3 are connected to switch3 and switch4 respectively.
- In the Containers environment, each NIC has different VLAN tags for different Container environment, for example 10g environment has e1000g131000 and e1000g131001 as public network NICs, where 131 is the VLAN and 000 and 001 is the NIC number.
- For two different environments, there are 2 VLAN tags configured for public network like 131, 132 for 10gR2 and 11gR1 respectively as public network. 111, 112 for 10gR2 and 11gR1 as private network.
- On the switches, switch1 and switch2 ports have been configured as trunk ports to allow VLAN traffic of VLAN 131, 132.
  - Look for the respective switch configuration to enable this "port trunk" feature.
  - Port trunking is enabled on switch ports where these NICs are connected so that VLAN tagged traffic is passed through.
  - On the switch configure VLAN tags with broadcasting option where no domain is configured. If VLAN domains are configured on the switch make sure the same VLAN domain is configured on both the switches.

- On the host, side just configure these VLAN tagged NICs along with Container configuration. And configure IPMP on top of it, to achieve high availability.
- IPMP can be configured in active-standby and active-active modes with probe based or link based failure detection, with or without FAILBACK option.
  - To leverage quick fail-over time, the configuration used is, link based active-active IPMP with FAILBACK option disabled.
  - Which means there is always only fail-over process that takes place every time when the VLAN tagged NIC fails hosting the IP address, and there is no FAILBACK at all.
- For more details and various options of IPMP configuration please refer to "Highly available and Scalable Oracle RAC networking with Oracle Solaris 10 IPMP".

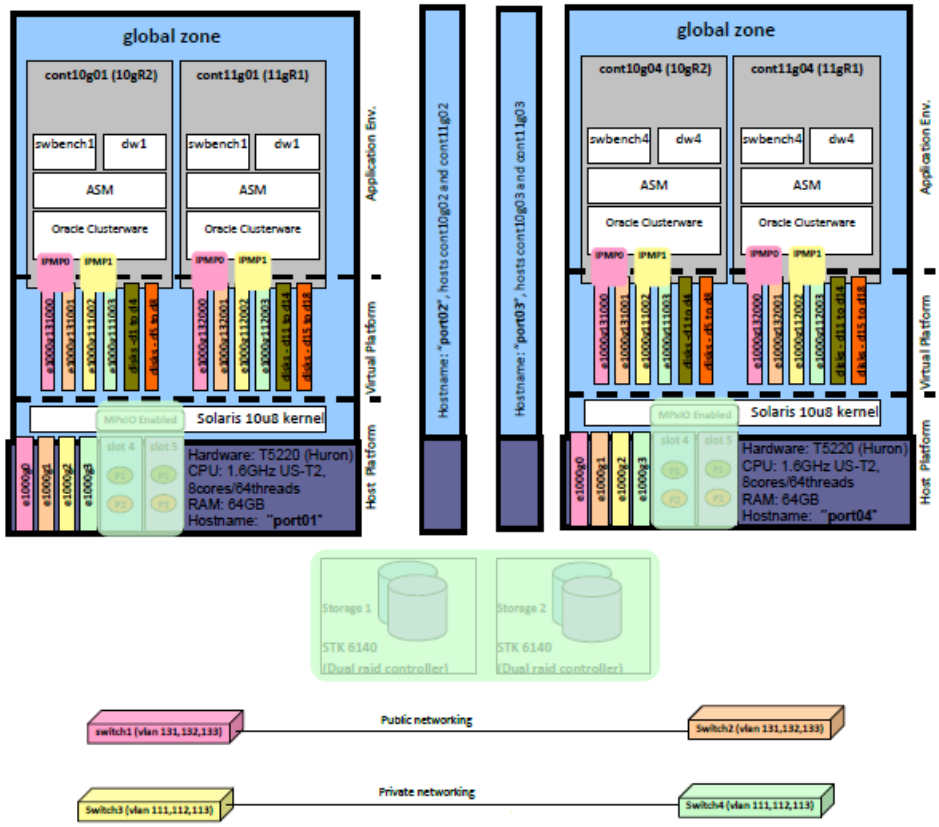

Figure 4: Network configuration

HA considerations for Oracle RAC inside Containers

For

- high availability,
- fault isolation,
- and better manageability

Think of hosting Oracle RAC on the global zone where one instance is hosted per physical server, the same approach has to be applied while consolidating the Oracle RAC using Oracle Solaris Containers too. For example, For Oracle RAC 10gR2, create one container per node on all 4 nodes as shown in the above drawing-2 cont10g01 to cont10g04. These 4 containers are used as 4 nodes to host Oracle

RAC 10gR2. To deploy such multiple clusters create different Container environment, like 11gR1 as shown in the above drawing-2. This means different Containers on the same node are part of different Oracle RAC cluster and not the same.

For different databases on the same cluster binaries, could leverage different projects to configure the various system level resources required for that database instances across all the Containers. Same binaries could be used to bring up the database. This is another level of software level of abstraction that gives even more fine grain control over the system resources, is called Solaris Resource Manager and it's explained in the subsequent chapter.

Instantly create Containers using ZFS

- Host Oracle Solaris Containers on ZFS file system to instantly create similar Container environment. It eases the creation of Container virtual environment by using snapshot feature of the ZFS file system.
- Create Containers on ZFS file system, configure the container environment to install Oracle clusterware and Oracle database binaries.
- Bring down the Container environment, take a snapshot with a name as 'pre_oracle_install'.
- Install Oracle clusterware software and Oracle database binaries only on the root file system of the Container.
- Once the binaries are installed, preserve the environment by taking another snapshot as 'port_oracle_install'. These snapshots could be taken at different stages of configuration too, to revert back to previous state, as it becomes easy to over-come any manual configuration errors.
- To create new Container environment with a different version of Oracle, leverage the existing snapshot 'pre_oracle_install', configure Container configuration template to leverage new network and disks configurations. Clone the 'pre_oracle_install' environment to create new Container with the custom configuration and place appropriate 'sysidcfg' file under /etc directory of the Container environment before booting the Container environment. This brings up the Container environment instantly ready to host any other version of Oracle Database or clusterware.
- On the other hand, to create similar Oracle version environment, create the custom container configuration file with the changed network and disk configuration. Clone the 'post_oracle_install' snapshot to instantly bring up the Oracle installed environment. Later configure the Oracle as per the new environment requirements.
- For more details on ZFS, please refer the zfs documentation on Oracle Solaris documentation.
- To limit the ZFS file system cache, edit /etc/system and provide 1GB as cache, typically ZFS would try to consume as much memory as it could and release on need basis. Instead limit the ZFS memory requirement to 1GB. Look at Appendix for the configuration details.

Container management and resource allocation

Oracle Solaris Resource Manager

Resource management functionality is a component of the Solaris Container environment. Resource management enables you to control how applications use available system resources.

- Allocate computing resources, such as processor time
- Monitor how the allocations are being used, then adjust the allocations as necessary
- Generate extended accounting information for analysis, billing, and capacity planning

Solaris resource management features enable you to treat workloads individually. You can do the following:

- Restrict access to a specific resource
- Offer resources to workloads on a preferential basis
- Isolate workloads from each another

Show dynamic resource changes

CPU

When we observe that one of the Container environment's CPU utilization is high, it is possible to dynamically allocate CPU resources to it, either by taking away CPU from underutilized Container or from the global zone. To move CPU from underutilized Container, first release those CPUs to the default pool, and later assign to the pool destination pool. Let's look at steps followed to dynamically change CPUs for a given Container:

Scenario: cont10g01 has dynamically created resource pool at the startup of the Container, which has 2 cores (16 threads) as the CPU count. It's observed that the virtual node is hitting 80% cpu on this node as well as on all other virtual nodes hosted on other physical nodes like cont10g02 to cont10g04. Let's look at how to add 1 more cores (8 threads) to this Container and do the same on other physical nodes too.

step1 – Ensure that there are 2 or more than 2 cores available on the pset_default processor set. Pset_default processor set is the default processor set. SUNWtmp_cont10g01 poolset, is created when cont10g01 zone comes up, and the pool disappears soon after the system reboots or the system goes down.

```
root@potr01:~/# psrinfo
root@potr01:~/# poolcfg -dc ''transfer 8 from pset pset_default to SUNWtmp_cont10g01''
```

step2 – Follow the above steps to dynamically change CPUs on all the other nodes too.

For temporary change of CPU, just modify the dynamic pool SUNWtmp_cont10g01 as called out above. For permanent change, change the Container configuration, so that next time Container boots, it comes up with the new CPU configuration. However the above example is interim arrangement of a CPU hungry Container environment, which would be typical run-time scenario.

There are other scenarios where a CPU share could be altered instead of dedicated CPUs on other platforms.

Conclusion

- Oracle Solaris Containers is the best choice for consolidating various Oracle RAC databases with different versions or patch levels. And could be hosted on the same operating system kernel.
- Oracle Solaris Containers enable the end users to manage resources well, pool the resources from underutilized Containers and provision the same to the required Containers.
- In addition to that the accounting feature of SRM could be leveraged to bill the end users based on the CPUs consumed, probably a grid environment could leverage this feature for appropriate billing based on resource utilization.
- Dynamic resource management like memory and CPU changes would offer greater flexibility.
- VLAN tagged NICs overcome the hard limitation of physical NICs on a server. To support this, network switches also posses this feature.
- IPMP takes care of availability at the network stack, apart from Oracle RAC's monitoring and timeouts to detect network failure.
- MPxIO offers seamless availability by offering single disk name for different paths of the same disk/LUN. It manages the availability of different paths of LUNs/disks from storage arrays to hosts.
- ASM offers the benefits of a cluster volume manager where disks/LUNs from different storage arrays could be mirrored to offer HA among two different storage arrays.

References

1. 'ZFS administration guide'
   http://docs.sun.com/app/docs/doc/819-5461
2. 'Best Practice for Running Oracle Databases in Solaris Containers'
   Roman Ivanov and Ritu Kamboj.
3. 'Highly available and Scalable Oracle RAC networking with Oracle Solaris 10 IPMP'
   John Mchugh and Mohammed Yousuf
4. Solaris Containers-Resource Management and Solaris Zones
   http://docs.sun.com/app/docs/doc/817-1592
5. 'Virtualization options for deploying Oracle Database Deployments on Sun SPARC Enterprise T-series Systems'
   Roman Ivanov and Mohammed Yousuf
6. Oracle Database Online Documentation 10g Release 2(10.2.0.4)
   http://www.oracle.com/pls/db102/homepage
7. Oracle Database Online Documentation 11g Release 1(11.1.0.7)
   http://www.oracle.com/pls/db111/homepage

Appendix-I - Sample Container configuration

Hardware:

Sun SPARC Enterprise T5220 (M-series or x86 based system could also be used to host Container virtual environment).

4 * Sun SPARC Enterprise T5220 server each configured with 1.6GHz 8 cores or 64 threads, and 64GB RAM.

Storage:

Sun StorageTek 6140 Array with dual controller.

OS:

Solaris10 10/09 SPARC with kernel patch 142900-14 and it's dependent patch 143055-01

Oracle database:

- Oracle RAC 10gR2 10.2.0.4 with latest patch set 9352164
- Oracle RAC 11gR1 11.1.0.7 with latest patch set 9207257 + 9352179

Configuration files

Host system configurations files:

1. Limit ZFS cache to 1GB

edit /etc/system # add these lines

```
* set this value for limiting zfs cache to 1G
set zfs:zfs_arc_max = 0x3E800000
```

2. To override the system wide limit of 1/4 of physical memory by default on S10, the shminfo_shmmax tunable would need to be configured in /etc/system to remove that limit. Edit /etc/system and set the value of shminfo_shmmax to the value that suites the requirement.

```
* set the max shared memory to 24G
set shmminfo_shmmax = 0x600000000
```

3. Enable MPxIO

edit /kernel/drv/fp.conf # the following entry

```
mpxio-disable="no";
```

If the entry is present, make sure it's set as mentioned above to enable the MPxIO or add the entry. For all the /etc/system values to take affect reboot the node.

Inside Containers configuration files:

1. Oracle shared memory configuration using SRM facility /etc/project.

```
root@cont11g01:/# cat /etc/project
system:0::::
user.root:1::::
noproject:2::::
default:3::::
group.staff:10::::
user.oracle:100:Oracle  project:::process.max-sem-nsems=(privileged,4096,deny);project.max-shm-
memory=(privileged,16106127360,deny)
```

2. IPMP configuration

a. Public network

```
root@cont10g01:/# cat /etc/hostname.e1000g131000
cont10g01 group pub_ipmp0
root@cont10g01:/# cat /etc/hostname.e1000g131001
group pub_ipmp0 up
root@cont10g01:/#
```

b. Private network

```
root@cont10g01:/# cat /etc/hostname.e1000g111002
cont10g01-priv group priv_ipmp0
root@cont10g01:/# cat /etc/hostname.e1000g111003
group priv_ipmp0 up
root@cont10g01:/#
```

Container configuration file:

Create a file with the following content to create the Containers.

#save the below content as config_template_to_create_cont10g01.cfg

```
create -b
set zonepath=/zonespool/cont10g01
set autoboot=false
set limitpriv=default,proc_priocntl,proc_lock_memory
set scheduling-class=TS,RT,FX
set ip-type=exclusive
add net
set physical=e1000g111002
end
add net
set physical=e1000g111003
end
add net
set physical=e1000g131000
end
add net
set physical=e1000g131001
end
add capped-memory
set physical=24G
end
add dedicated-cpu
set ncpus=16
end
add rctl
set name=zone.max-swap
add value (priv=privileged,limit=25769803776,action=deny)
end
add rctl
set name=zone.max-locked-memory
add value (priv=privileged,limit=12884901888,action=deny)
end
add device
set match=/dev/rdsk/c5t600A0B800011FC3E00000E074BBE32EAd0s6
end
add device
set match=/dev/dsk/c5t600A0B800011FC3E00000E074BBE32EAd0s6
end
add device
set match=/dev/rdsk/c5t600A0B800011FC3E00000E194BBE3514d0s6
end
add device
set match=/dev/dsk/c5t600A0B800011FC3E00000E194BBE3514d0s6
end
add device
set match=/dev/rdsk/c5t600A0B800011FC3E00000E234BBE720Cd0s6
end
add device
set match=/dev/dsk/c5t600A0B800011FC3E00000E234BBE720Cd0s6
end
```

Copy paste the above content, change the disk paths and NIC names to suite your configuration.

```
root@port01:~/# zonecfg -z cont10g01 -f config_template_to_create_cont10g01.cfg
```

This will create the zone with the name “cont10g01”.

```
root@port01:~/# zoneadm -z cont10g01 install
```

#Installation is complete, boot the Container and configure it.

```
root@port01:~/# zoneadm -z cont10g01 boot
```

#Login at the Container console to configure it for the first time, set the root passwd, configure networking, configure time zone, etc. And zone/Container will reboot.

```
root@port01:~/# zlogin –C cont10g01
```

Follow the steps on the screen to configure the zone.

Oracle is committed to developing practices and products that help protect the environment

Oracle White Paper— Best practices for deploying Oracle RAC inside Oracle Solaris Containers
September 2010
Author: Mohammed Yousuf
Reviewers: Allan Packer, Gia-Khanh Nguyen, Wataru Miyoshi

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

SOFTWARE. HARDWARE. COMPLETE.