ORACLE
SOLARIS

An Oracle Technical White Paper
October 2011

Minimizing Downtime While Updating Oracle Solaris Containers in Clustered and Non-Clustered Environments

ORACLE

Introduction

Application downtime, whether planned or unplanned, is not an option today. But patching or upgrading an OS or an application still needs downtime. In highly consolidated environments it can become nearly impossible to agree on a common downtime - even a very short one for a single reboot - which would be needed to activate patches or upgrades. There are already a number of technologies available that make it possible to minimize downtime like Live Upgrade ([1]). This paper goes a bit further and shows how to combine Oracle Solaris 10 technologies to update Oracle Solaris Containers individually while minimizing downtime in clustered and non-clustered environments.

An Oracle Solaris Container, also known as an Oracle Solaris Zone, is a complete runtime environment for applications. Zones allow application components to be isolated from one another even though the zones share a single instance of the Oracle Solaris Operating System.

With the capability to create up to more than eight thousand Zones in one Oracle Solaris instance, the adoption of Oracle Solaris Containers allows for the consolidation of very large environments. These environments need to be operated very efficient. Lifecycle management, e.g. installing patches and performing upgrades is one of the most important tasks of service operations.
(In the following the term “update” will be used to include patching and updating.)

The more zones that are installed on one physical system, the more important is the careful planning to update the zones. If for example a system has 30 production zones running, there is much effort to agree on a common downtime. So during the planning process the following questions typically come up:

- How can a large number of zones be updated with minimal downtime ?
- Can the update be performed on a zone-by-zone basis ?
- What happens with the application data between the updates ?
- How can such a process be standardized ?
- Can such a process be made easy and safe to be used by the operating staff ?
- Is there a way to fallback safely, e.g. if an update fails ?

Applying Updates on an Oracle Solaris System with Zones Installed

Software packages in the global zone and in non-global zones should be in sync to the maximum extent possible. The Oracle Solaris package tools enforce this (See [2] for more details). All patches applied at the global zone level are applied across all zones. When a non-global zone is installed, it is at the same patch level as the global zone. When the global zone is patched, all non-global zones are similarly patched. When migrating a non-global zone from a system with an older software and patch level to a system with newer software, the non-global zone would have to be updated.

This process can be automated by a mechanism introduced with a later release of Oracle Solaris called "update-on-attach." This technology compares the software and patch level of a non-global zone with that of its global zone and lifts this up to the level of the global zone as part of the `zoneadm attach -U` process. It is important to understand that the "update-on-attach" mechanism is not capable of performing downgrades of software or patches.

Alternatives for Updating Oracle Solaris Containers

There are two fundamentally different approaches to update zones – besides the “normal” updating process that updates a zone during a full downtime:

- Updating a copy of a zone and
- Creating a new zone

The first variant has two subclasses:

- Using Oracle Solaris Live Upgrade or
- Manually cloning and updating a zone.

The second variant also has two subclasses:

- Create a zone from a reference zone and update it, or
- Create a personalized zone on an updated system.

With all methods, application data should reside on a separate filesystem, so that it is a) not touched during a snapshot or upgrade process and b) can be simply remounted once the updated zone boots up again and becomes productive.

Update a Copy of a Zone

- **Using Oracle Solaris Live Upgrade**: The idea of Live Upgrade is to create a copy of the active filesystems of all zones while the system is running in production. The copy will either be placed onto a separate file system or be created as a ZFS snapshot (if ZFS is used) on the production system. Then the update is applied to this copy, still while the system is up and running. In the final step the copy will become active after the next reboot of the system with all its zones.
- **Manually clone and update a zone**: The production zone is cloned in a way that the clone contains all customizations of the production zone. The cloned zone will then be attached to a second system, running a more recent software level in the global zone by using the "update-on-attach" feature. This process will update the clone to this newer software level on the second system, while the production zone is up and running.

Use a new Zone, Created from a Reference

- **Update a copy of a reference zone**: A new zone will be created based on a reference zone that contains all customizations that have been done to the production zone in the past. The new zone will then be attached to a second system, running a more recent software level in the global zone. By using the "update-on-attach" feature, the zone will be updated to this newer software level. The update process of the zone will happen on this second system, while the production zone is up and running. (Finally the old zone has to be shutdown, the data volumes be migrated to the new server, and the new zone be started.)
- **Create a personalized zone on an updated system**: A new zone will be created by running a script. This script adds all the necessary customizations, to let the new zone look like the production zone. If the new zone is created on a system running a more recent software level in the global zone, the zone adopts the new software level. No update is necessary for this zone. Again, to enable this zone as the production zone, the old production zone has to be stopped, the data volumes be migrated and then the new zone has to be started using the production data. The process of creating the new zone happened on this second system, while the production zone was up and running on another system.

Comparing Oracle Solaris Container Update Procedures

To evaluate these four ways to update zones, the following questions will be compared in a table:

- Because a running zone can not be updated, a copy is needed to perform the update. How much effort is required to create this copy ?
- During the lifetime of the production zone a lot of customizations may have been made to the zones. How can these customizations survive the update process ?

- If multiple zones exist on the production system, it is very unlikely to get a common downtime for all zones to perform an update. Live Upgrade also does not help, as this would require one reboot, which would affect all zones simultaneously. How can zones be updated on a zone-by-zone basis ?
- When a system is updated, the zones on it are updated too - either during a common update downtime or during production, when Live Upgrade is used. The added I/O-burden during the update process needs to be considered. If a second system is available to perform the update, the running zones would not be affected at all. Is a second system used during the update process ?

TABLE 1: COMPARING UPDATE PROCDURES OF ZONES

UPDATE PROCESS	EFFORT TO COPY THE ZONE	SURVIVAL OF ZONE-CUSTOMIZATIONS	ZONE-BY-ZONE UPDATE POSSIBLE	SECOND SYSTEM USED
Live Upgrade	little, lucreate command	copy through Live Upgrade	no	no
Clone and update the zone	little, clone the zone	clone customizations	yes	yes
Update a copy of a reference zone	medium, maintain sample zone, copy sample zone	maintain in sample zone	yes	yes
Create a personalized zone on updated system	high, maintain the script, create the zone by script	maintain by updating the script	yes	yes

Based on Table 1 the process to update a cloned zone looks well-suited to update zones with minimal downtime, little effort and on a zone-by-zone basis.

- The effort to create the copy of the zone is very small, efficient and independent of used storage devices, if ZFS is used for cloning.
- All customizations, done to the zone will survive during the update process, because they will just be cloned.
- If “update-on-attach” will be used for updating a zone, the zones can be updated on a zone-by-zone basis. A separate zone will be cloned and updated on a second system. The production zone will be up and running until the update process of the cloned zone is done. Then the cloned zone will take-over the role of the production zone and the former production zone will be retired for fallback reasons. This is in contrast to using Live Upgrade, where all zones together would be copied and updated together with the global zone.
- Mission critical environments use multiple similar systems to guarantee highest availability. These systems can be used during the update process on a zone-by-zone basis.
- The retired zone and data remain available as fallback solution.

We will explain in the following pages how to update Oracle Solaris Containers in non-clustered environments and then extend the procedure to update zones under Oracle Solaris Cluster control.

Prerequisites

To explain the process, we will define a set of requirements. To have consistent naming we have defined some naming conventions that will be used in this paper.

Caution

All of the code samples have been extensively tested. Nevertheless we cannot guarantee that they work flawlessly under all circumstances.

It is advised to have current backups available of the systems being upgraded. It is also a best practice to test the proposed update procedures in test environment.

Naming Conventions

The two systems involved in the update process will be named TOM and JERRY, where JERRY is always the system with the more recent software level.

The production zone is named ora_zone - as runtime environment of service ora.

ZFS will be used as the root filesystem for the zone. That makes cloning of the zone very easy and reliable with little effort.

The zpools used by the zone, will be named ora_zpool and ora_dpool, where ora_zpool is the root pool of the zone and ora_dpool is the pool for the data.

The names of the zones and the zpools are modified in two ways.

- A zone or a zpool that has been cloned, will be named <name>-cloned.
- A production zone or zpool that will no longer be used in production, but might be used later on for fallback, will be named <name>-retired.

Structure of the Zones Environment

The zone root of ora_zone is placed into the data set ora_zpool/ora_zone. This zone root data set is mounted on /zones/ora_zone.

There are two possibilities to deploy applications:

- In the root filesystem of the zone and
- On the data file system which is on a separate pool.

Installing application binaries on the data pool makes the update more difficult. Deploying the applications in the root file system of the zone makes it possible to update the application at the same time, because the zone is updated without affecting the application data.

All application data of the application called ora and also all logdata of ora_zone are located on a separate zpool ora_dpool.

This zpool is located on SAN-storage and can be imported by TOM or JERRY. With this, the application data can be moved independently between TOM and JERRY through `zpool export` and `zpool import`. And so the data can stay and be productive on the production zone until the new zone has been created and updated to the new software level. Only for the last step, the move of the data into the new zone, the application needs to be taken down. This can only be done, if the data are independent from the zones root filesystem. The mountpoint for ora_dpool depends on the requirements of the application that is being used in the zone.

If logfiles should be carried between the zones and updates, they can also be placed into ora_dpool. If this is not possible, an additional synchronization step has to be added to the updating process – just like Live Upgrade does this.

TOM and JERRY are running Oracle Solaris 10 9/10 or above. This Oracle Solaris Update includes the most recent "update-on-attach" functionality, which will be used for updating the zones.

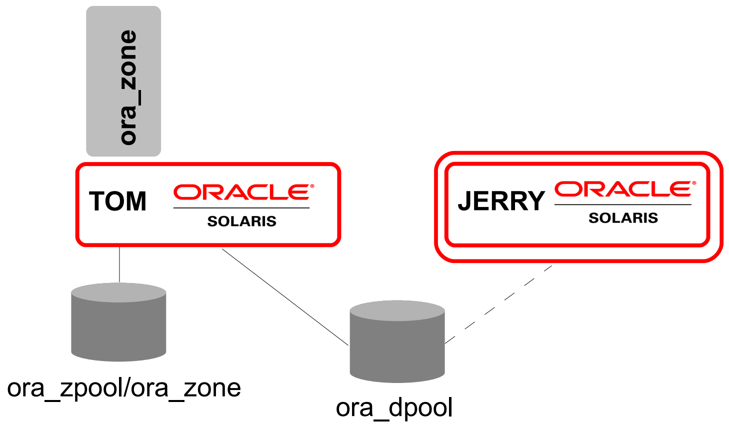


Figure 1: Basic Prerequisites: Oracle Solaris Containers and Zpools

Zones and Zpool Configuration

The following example shows the configuration of ora_zone and the creation of the zpools on TOM.

```
[TOM:root] zonecfg  -z ora_zone info
zonename: ora_zone
zonepath: /zones/ora_zone
...
dataset:
        name: ora_dpool
[TOM:root] zpool create -m none ora_zpool c3t600A0B8000347ECF00000C014DE35887d0
[TOM:root] zfs create -o mointpoint=/zones/ora_zone ora_zpool/ora_zone
[TOM:root] chmod 700 /zones/ora_zone
[TOM:root] zpool create -m none ora_dpool c3t600A0B8000347ECF00000C044DE358A9d0
```

Advanced Updating of Non-Clustered Oracle Solaris Containers

The following chapter will explain how to use “update-on-attach” with minimal downtime in a non-clustered environment. This process is very similar to the process that we generally described and compared in table 1 as “Clone and update the zone”.

Process Overview

We clone the production zone ora_zone by creating a snapshot of the ZFS zone root, while the zone is up and running. Using this snapshot, the cloned zone root is created locally on the production system (TOM) into a new created zpool. Then we move this new zpool over to JERRY, who runs already a more recent OS level. Now it is possible to create on Jerry the cloned zone through “update-on-attach” of the cloned zone root, which also exists in the moved zpool. This will update the cloned zone to the newer software level of JERRY. We can perform the update and test process of the cloned zone on JERRY, while the production zone is up and running on TOM with the application data.

Later, we shutdown the production zone, move the application data over to JERRY and then the former cloned zone will become the production zone. The previous production zone will be retired and stays intact as fallback solution.

The process is shown by the following figures in five steps.

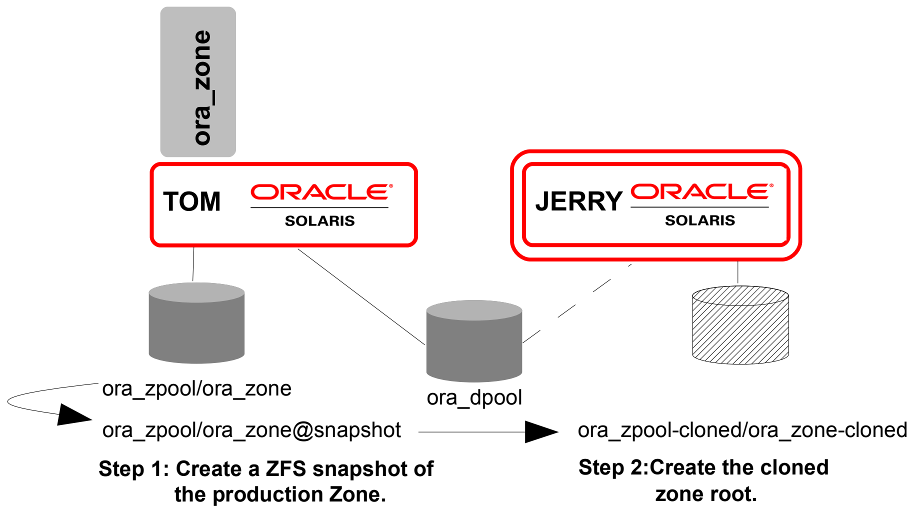
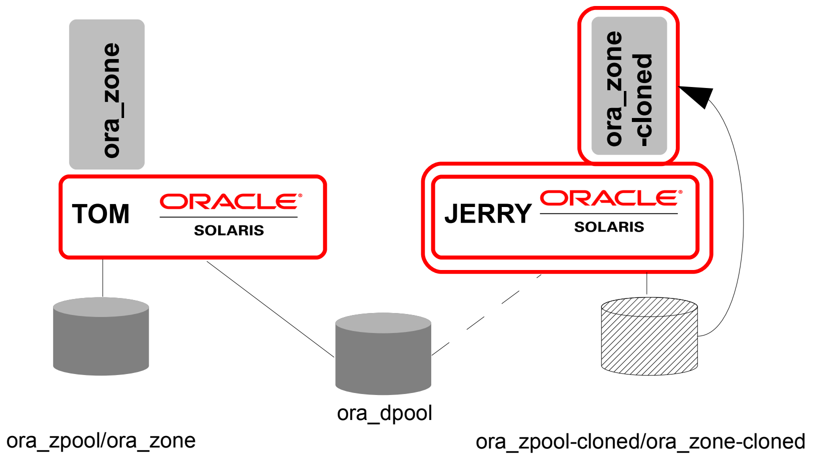

ora_zone
TOM
ORACLE
SOLARIS
JERRY
ORACLE
SOLARIS
ora_zpool/ora_zone
ora_dpool
ora_zpool/ora_zone@snapshot
ora_zpool-cloned/ora_zone-cloned
Step 1: Create a ZFS snapshot of the production Zone.
Step 2:Create the cloned zone root.

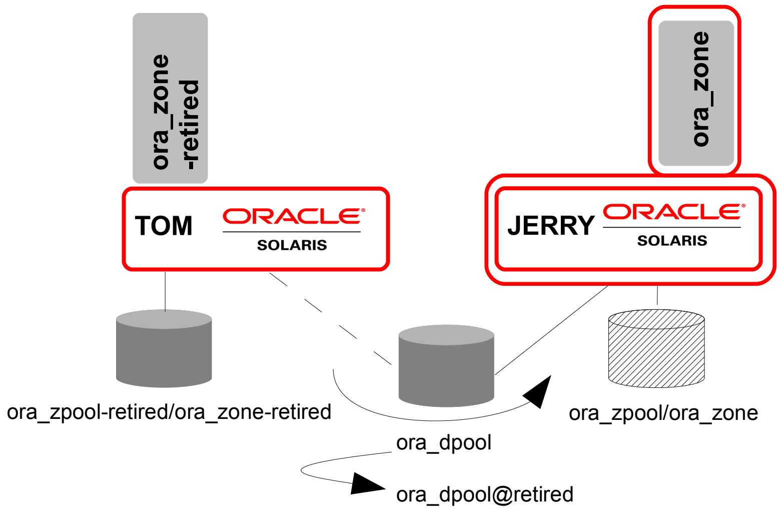
Figure 2: Updating Non-Clustered Zones: Step 1 and Step 2



Step 3: Perform the update process.

Step 4: Boot and test the cloned zone.

Figure 3: Updating Non-Clustered Zones: Step 3 and Step 4



Step 5: Change the roles of the Zones.
- Create a Snapshot of the production data
- Rename the zpools
- Rename the ora_zone data sets
- Move ora_dpool
- Boot the production Zone

Figure 4: Updating Non-Clustered Zones: Step 5

Advanced Updating Zones in five Steps

Step 1: Create a ZFS Snapshot of the Production Zone.

First generate the list of installed packages and patches (SUNWdetached.xml) of ora_zone. This list will be used later to configure the cloned zone root on JERRY.

To snapshot the production zone during runtime, we use the `zfs snapshot` command. This creates a consistent copy of ora_zpool/ora_zone. This ZFS snapshot will later be used to clone ora_zone. Cloning the zone root with ZFS results in a real identical clone of the running zone. (`zoneadm clone` can not be used here, because this command requires the zone to be halted during the clone. Additionally the cloned zone would have been set to a neutral configuration state - no hostname, standard timezone, removed nameservices, etc.- which would not lead to an identical zone copy.)

Note: All subsequent changes to the production zone, like configuration changes or package installations, will be not available in the cloned zone. Change requests to the zone should be also made to the clone and become active after making the clone active.

To cleanup remove SUNWdetached.xml from ora_zpool/ora_zone.

```
[TOM:root] zoneadm -z ora_zone detach -n > /zones/ora_zone/SUNWdetached.xml
[TOM:root] zfs snapshot ora_zpool/ora_zone@snapshot
[TOM:root] rm /zones/ora_zone/SUNWdetached.xml
```

Step 2: Create the Cloned Zone Root.

To be efficient, perform the creation of the clone on TOM. Use the ZFS snapshot to create the clone of ora_zone's zone root by `zfs send` and `zfs receive`. The result is a new data set that will later be used by JERRY. To make it obvious that this is a clone and to have distinct names, name the new zpool and data set ...-cloned. We create the new data set in a new zpool (ora_zpool-cloned) that can later easily be transferred to JERRY.

`zfs send -p` will make sure that all ZFS properties of ora_zpool/ora_zone are transferred. While ora_zone-cloned will also receive the pathname of ora_zone, `zfs receive -u` will safeguard that ora_zone-cloned will not be automatically mounted by TOM, because the production zone is still up and running and mounted on /zones/ora_zone.

After that, we delete the snapshots on ora_zpool and ora_zpool-cloned, because they are no longer needed.

Then we make the zpool available on JERRY by running `zpool export` on TOM and running `zpool import` on JERRY. The mountpoint of ora_zpool-cloned/ora_zone-cloned stays on /zones/ora_zone.

As an alternative, we could have transfered the snapshot to JERRY over the network with `zfs send` and `zfs receive` into an existing ora_zpool-cloned on JERRY. This transfer would take more time and consume additional network bandwidth, but it would not require shared disk storage for ora_zpool-cloned.

Note: Check if ora_zpool has additional zpool properties set, that need to be adopted by ora_zpool-cloned.

```
[TOM:root] zpool create -m none ora_zpool-cloned
c3t600A0B8000347ED600000C654DE3598d0
[TOM:root] zfs send -p ora_zpool/ora_zone@snapshot | \
           zfs receive -u ora_zpool-cloned/ora_zone-cloned
[TOM:root] zfs destroy ora_zpool/ora_zone@snapshot
[TOM:root] zfs destroy ora_zpool-cloned/ora_zone-cloned@snapshot
[TOM:root] zpool export ora_zpool-cloned


[JERRY:root] zpool import ora_zpool-cloned
```

Step 3: Perform the Update Process.

If the zone configuration does not exist, we configure now the zone on JERRY from the SUNWdetached.xml file that is located in /zones/ora_zone. We call the zone different than ora_zone, to make it obvious, that this is not the production zone.

By using “update-on-attach”, the zone could now be updated to the software level of the global zone. This attach process would not work here, because the data set ora_dpool is configured for ora_zone, but is missing on JERRY.

There are three alternatives to solve this situation:

- Create an empty local data set to perform the attach operation
- Remove the data set from the zone configuration
- Create a snapshot of the production ora_dpool and work with this snapshot

The first two alternatives could not work if an application within the zone would automatically be started and would require their data to be available. In that case the usage of a data snapshot would be mandatory.

In this example an empty local data set in a file is created. After the empty ora_dpool data set has been created, the update process can be performed.

```
[JERRY:root] zonecfg -z ora_zone-cloned create -a /zones/ora_zone
[JERRY:root] mkfile 64M /tmp/tmp_dpool
[JERRY:root] zpool create -m none ora_dpool /tmp/tmp_dpool
[JERRY:root] zoneadm -z ora_zone-cloned attach -U
```

Step 4: Boot and Test the Cloned Zone.

Prior to making the cloned zone the production zone, we need to test the zone. One potential difficulty could be the zone's the network configuration. This could contain IP addresses , that are still in use by the production zone. There are three alternatives to solve this:

- Temporary remove the IP-configuration from the zone configuration
- Change the IP-configuration of the zone
- Run the zone during the test in an isolated network, where a conflict with the IP-configuration of the production zone is not possible.

Because in our example no network is configured, we can skip this problem.

Now we can boot and test ora_zone-cloned. After successful testing, we shutdown the zone, clear the configuration changes and remove the temporary ora_dpool.

```
[JERRY:root] zoneadm -z ora_zone-cloned boot

[JERRY:root] zlogin ora_zone-cloned init 5

[JERRY:root] zpool destroy ora_dpool
[JERRY:root] rm /tmp/tmp_dpool
```

Step 5: Change the Roles of the Zones.

Now is the time for ora_zone and ora_zone-cloned to change their roles. That means that ora_zone-cloned will become the production zone and the former production zone will be retired - maybe for possible later fallback reasons (See the "Performing a Fallback (The Extra Step)" section).

To be prepared for a fallback of the whole process, we create a snapshot of the application data prior to changing the roles. A recursive snapshot for all included data sets can be created by `zfs snapshot -r` on ora_dpool. To create a consistent snapshot of the application data, we have to make sure that all data that may have been cached by the application, have been synced onto disk. We achieve this by stopping the zone. The downtime for the application ora begins.

Now we need the production data in ora_dpool on JERRY. ora_dpool is located on shared SAN storage. To be efficient and fast to limit the downtime, we make ora_dpool to JERRY available by running `zpool export` on TOM and `zpool import` on JERRY.

As an alternative, the same procedure as used for cloning ora_zone's zone root could be used to clone the ora_dpool. Then we would use the recursive snapshots of ora_dpool to create new data sets in a new created dpool (called ora_dpool-cloned). We would then move the whole new zpool ora_dpool-cloned to JERRY, rename it there and use it for production. While this alternative creates a complete independent dpool, the copying of the snapshots would take much more time, consumes more disk throughput and would lengthen the required downtime.

To enable a transparent change, we now change all the names for the zones and the zpools to have obvious names for the roles of the zones. There must never be the situation in the network or on the storage where two zones or two zpools have the same name.

We achieve the renaming of a zpool by exporting it and importing it with a different name. ZFS data sets can be renamed with `zfs rename` and zones can be renamed with `zonecfg`.

Note: The mountpoint of our zone can not be changed while it is in installed state and stays on /zones/ora_zone.

After the name changes we now boot the cloned zone on JERRY. The downtime for the application ora ends.

```
[TOM:root] zlogin ora_zone init 5

[TOM:root] zfs snapshot -r ora_dpool@retired
[TOM:root] zpool export ora_dpool
[TOM:root] zfs rename ora_zpool/ora_zone ora_zpool/ora_zone-retired
[TOM:root] zpool export ora_zpool
[TOM:root] zpool import ora_zpool ora_zpool-retired
[TOM:root] zonecfg -z ora_zone set zonename=ora_zone-retired

[JERRY:root] zonecfg -z ora_zone-cloned set zonename=ora_zone
[JERRY:root] zpool export ora_zpool-cloned
[JERRY:root] zpool import ora_zpool-cloned ora_zpool
[JERRY:root] zfs rename ora_zpool/ora_zone-cloned ora_zpool/ora_zone
[JERRY:root] zpool import ora_dpool
[JERRY:root] zoneadm -z ora_zone boot
```

After this step, the update process is done. The production zone runs the software level of JERRY and is using the current application data of ora_dpool. A backup copy of the application data prior to the update exists in the snapshot ora_dpool@retired. As required, the downtime was very short. The only needed downtime for the application ora during the update process was the time needed in step 5.

Note: The retired zone must not be booted later on by accident and may only be used for fallback reasons with the old data. To assure this, the retired zone could be detached from TOM.

Performing a Fallback (The Extra Step)

In most cases the update process will only be performed, if a fallback solution has been prepared. A fallback is required if the update process fails or the zone needs to be reverted back to its previous configuration for other reasons.

The fallback with this procedure is very easy. It is just turning back step 5. The retired zone has not been altered and the retired data in ora_dpool is still in sync with the retired zone.

We need to rename back the ora_zpools, the ora_zone data sets and the zonenames.

Then we rollback the snapshot of the application data and move ora_dpool back to TOM. Because there is no command to recursively roll back a recursive snapshot, you need to roll back all existing snapshots on ora_dpool step by step.

Then we can boot the old production zone on TOM.

```
[JERRY:root] zlogin ora_zone init 5

[JERRY:root] zpool export ora_dpool
[JERRY:root] zfs rename ora_zpool/ora_zone ora_zpool/ora_zone-cloned
[JERRY:root] zpool export ora_zpool
[JERRY:root] zpool import ora_zpool ora_zpool-cloned
[JERRY:root] zonecfg -z ora_zone set zonename=ora_zone-cloned

[TOM:root] zonecfg -z ora_zone-retired set zonename=ora_zone
[TOM:root] zpool export ora_zpool-retired
[TOM:root] zpool import ora_zpool-retired ora_zpool
[TOM:root] zfs rename ora_zpool/ora_zone-retired ora_zpool/ora_zone
[TOM:root] zpool import ora_dpool
[TOM:root] zfs rollback ora_dpool@retired
[TOM:root] zfs destroy ora_dpool@retired

[TOM:root] zoneadm -z ora_zone boot
```

Now the production zone runs again on TOM with the old software level and the old application data. We only rolled back step 5. Because we were well prepared, the required downtime to fallback was again very short.

Advanced Updating of Failover Zones in Oracle Solaris Cluster

In this chapter we want to extend the update process, described so far for non clustered environments, to Oracle Solaris Containers that are under Oracle Solaris Cluster control, a so called failover zone, sometimes also called a “flying container”. A failover zone is a zone that is under Oracle Solaris Cluster control, and can be failed over manually using the Oracle Solaris Cluster CLI or which is failed over automatically by the cluster in certain failure conditions.

The agent, implementing the integration of highly available failover containers into the cluster, is called “HA Container Agent” [3]. Failover zones under cluster control are very popular with customers, as their management is similar to the management of a complete OS. In addition, this type of zones offers a way to delegate the administration of a zone to its owner, which is a benefit in strictly separated administrative organizational structures.

Differences Between Updating Clustered and Non-Clustered Zones

There are a few differences, that make it worth to describe this scenario in detail as well.

- A failover zone is in the `installed` state on all nodes of the cluster on which the zone could be started by the cluster (as defined by the nodelist property of a resource group). Therefore, you must not create the zone on the node where the “update-on-attach” will take place, because the right zone configuration is already in place.
- Cluster failover zones must be prevented from failing over to a node with a newer software and patch status.
- The integration of the failover zone resources makes some extra steps necessary, on the other hand, facilitate the fallback process.
- Finally, non-clustered nodes may run different software versions for an indefinite amount of time. But clustered nodes are allowed to differ in version, software, or patch level only during the update process itself. This is a clearly documented restriction. The cluster must be back into its normal state as soon as possible.

Creating the Zone Resource Group

In order to use a real life example in the following paragraphs, we will briefly describe the process to create a cluster resource group (RG) controlling the failover container first. The following chapters will then be using this sample RG and demonstrate all of the steps required for updating the zone with detailed commands and explanations.

The zone RG (ora_zone-rg) will contain three resources:

- Logical IP address
- A storage resource of type HAStoragePlus (ora_zone-hasp), that makes sure that the zpools, needed for the container to run, are imported before the zone boots.
- A container resource of type GDS (Generic Data Service), that controls the zone itself.

None of the following commands returns any message text as long as no error occurs! All cluster commands can be run on any of the cluster nodes.

```
[TOM:root] clrg create ora_zone-rg
[TOM:root] clrslh create -g ora_zone-rg hazone
[TOM:root] clrs create -t HAStoragePlus -g ora_zone-rg \
           -p Zpools=ora_zpool,ora_dpool ora_zone-hasp
[TOM:root] clrg online -M ora_zone-rg
[TOM:root] clrt register SUNW.gds
```

With this first set of commands we create an empty RG and add two resources to it, the IP address and the storage resource. Both of these resources are still in the state `disabled`. The SUNW.gds resource type only has to be registered, if this has not been done already.

We can only create the zone resource if all of the corresponding storage resources are available, i.e. its zpools are imported. Otherwise the validation process that is part of the resource creation fails and the zone cannot be created. Therefore we have to start the still incomplete RG , which enables the two resources and makes the storage resource import the zpools.

In order to create an HA container resource, a configuration file must be used. Usually one copies the reference file in /SUNWsczone/sczbt/util/sczbt_config to enter the relevant information into this copy. For this example the following settings will be used. Only the relevant part of this file is shown. And all of the variables are clearly explained in the reference file.

```
RS=ora_zone-rs
RG=ora_zone-rg
PARAMETERDIR=/zones/ora_zone
SC_NETWORK=true
SC_LH=hazone
FAILOVER=true
HAS_RS=ora_zone-hasp
Zonename="ora_zone"
Zonebrand="native"
Zonebootopt=""
Milestone="multi-user-server"
LXrunlevel="3"
SLrunlevel="3"
Mounts=""
```

In the same directory a registration script is available, that creates the HA container resource by reading the configuration file (in this case the script is called sczbt_config_ora_zone).

```
[TOM:root] ./sczbt_register -f ./sczbt_config-ora_zone
sourcing ./sczbt_config-ora_zone
Registration of resource ora_zone-rs succeeded.
Validation of resource ora_zone-rs succeeded.
```

The resource is now created, but not yet started. To start the zone, the resource has to be `enabled`. Using the following command we boot the zone „within“ the zone resource group ora_zone-rg. Remember: the other two resources are already in `enabled` state through the onlining of the RG.

Note: Make sure, that the zone is fully configured before the `clrs enable` command is issued. If your zone has no proper system configuration (sysidcfg file) it must be booted at least once manually, to complete the initial installation process. Otherwise the boot process initiated by the cluster will fail.

```
[TOM:root] clrs enable ora_zone-rs
[TOM:root] zoneadm list -icv
  ID NAME             STATUS     PATH                   BRAND    IP
   0 global           running    /                      native   shared
  14 ora_zone         running    /zones/ora_zone        native   shared
```

Now, this RG can be switched to another cluster node by using the simple `clrg switch` command. The switch subcommand will stop the RG with all its resources; i.e. the zone will be shutdown, the IP address unconfigured and the zpools exported. On the other node all of the resources will be started in reverse order to establish a running zone with all its resources being present.

How to Patch one Node of a Cluster (Rolling Upgrade)

As we depend for the update example on cluster nodes with different patch levels, we will briefly describe how updating a cluster node can be done.

There are three procedures available to update a cluster

- Standard upgrade, which updates all nodes of a cluster while the cluster is down
- Rolling upgrade, which updates a set of nodes at a time and lets the nodes rejoin the cluster, so that it runs with mixed versions and
- Dual partition upgrade, which also updates a set of nodes at a time, but then starts a new cluster with the updated nodes and switches the HA services to the new cluster during this process.

To minimize the downtime and to minimize the time a cluster runs without proper redundancy (in the case of the rolling and dual partition upgrade) the Live Upgrade technology can be used. It creates an alternate boot environment (ABE) and updates this while the system is running. A simple reboot then enables this ABE. Live Upgrade can be combined with all three of the procedures presented.

It is important to understand, that a rolling cluster upgrade is only possible between minor updates, e.g. from Oracle Solaris Cluster 3.2 2/08 to Oracle Solaris Cluster 3.2 11/09, or between minor Oracle Solaris updates. If a major update of Oracle Solaris or Oracle Solaris Cluster has to be done, only the standard upgrade or the dual partition upgrade are an option.

It is also important to understand, that in the context of this paper, the dual partition upgrade can not be used to update a system with failover containers. This is a conceptual problem, as the prerequisite for a dual partition upgrade is that shared storage is not part of the update. But, as a failover container is based on a zonepath which has to be located on a shared storage, and which has to be updated as part of any update, this just does not fit the model.

It is left as an exercise for the reader to think about combining the procedures described in this paper with a dual partition upgrade.

Restrictions for Rolling Cluster Upgrades

Be aware that in an Oracle Solaris Cluster environment the nodes of a cluster must always run with the same version and patch levels of Oracle Solaris and Oracle Solaris Cluster. The only exception is for the purpose of a rolling upgrade. It is obvious that under this rule, having two cluster nodes running different versions for a week or even longer would be unsupported. Please contact your local support engineers to discuss more details.

Advanced Approach for Updating a Zone Under Cluster Control

In the following paragraphs we'll execute the same five steps described in "Advanced Updating Zones in five Steps". They will be slightly changed, as the prerequisites in a cluster environment offer more flexibility. They also will be enhanced with the additional steps to take care of the cluster integration.

As we are not dealing with a simple zone any more but with a cluster resource group, some more care has to be taken. As we are now working in a cluster environment with nodes at different patch levels, a zone running on TOM must not be switched to JERRY, not even in case of a node failure. So, the first additional step is to set the nodelist property to contain only TOM which prevents this from happening.

```
[TOM:root] clrg set -p nodelist=TOM ora_zone-rg
```

As described in "Advanced Updating of Non-Clustered Oracle Solaris Containers" the preparation for the update process contains steps 1 to 3:

1. Taking a snapshot of the running zone
2. Copying the snapshot into a separate zpool and
3. "configure" the cloned zone on JERRY and perform the update-on-attach.

All of these actions are being done, while the original zone (ora_zone) is up and providing services. The fact that the zone is running under cluster control is not relevant yet.

Step 1: Creating the Snapshot

The following command shows how the snapshot is taken:

```
[TOM:root] zfs snapshot ora_zpool/ora_zone@snapshot
```

Step 2: Transferring the Snapshot

This sequence of commands is still independent of the cluster controlling the container. It copies the snapshot to a new pool into a new filesystem, that uses a new data set name to make it obvious that this is a clone: ora_zone-cloned. This zpool can now be exported to JERRY.

```
[TOM:root] zpool create -m none ora_zpool-cloned c3t600A0B8000347ED600000C654DE359d0
[TOM:root] zfs send -p ora_zpool/ora_zone@snapshot | \
           zfs receive -u ora_zpool-cloned/ora_zone-cloned
[TOM:root] zfs destroy ora_zpool-cloned/ora_zone-cloned@snapshot
[TOM:root] zfs destroy ora_zpool/ora_zone@snapshot
```

As an alternative, one could copy the snapshot directly to a zpool owned by the other node, but doing the copy locally and switching the zpool then in the SAN is the more efficient way.

We now export the clone, which is stored in the zpool ora_zpool-cloned and import it on the updated node. Do not try to rename the pool to the original name, as this will later lead to confusion as you'll have two pools with the same name on shared storage!

```
[TOM:root] zpool export ora_zpool-cloned

[JERRY:root] zpool import ora_zpool-cloned
[JERRY:root] zfs list
NAME                              USED  AVAIL  REFER  MOUNTPOINT
ora_zpool-cloned                 4.41G  5.38G    23K  none
ora_zpool-cloned/ora_zone-cloned 4.41G  5.38G  4.41G  /zones/ora_zone
```

We now have imported on the updated cluster node a ZFS data set and a zpool that have different names, but exactly the same mountpoints as the originals.

Step 3: Performing the Update Process

In contrast to the first example in "Advanced Updating Zones in five Steps," the zone configuration for this zone already exists on the second node. Because this is a cluster and the zone is configured with the HA Container agent, this is a prerequisite. We also prevented the zone from failing over to JERRY by setting the nodelist property appropriately. So, it is now safe to use the same pathnames, but not the same pool and filesystem names, because the original pool is still using them. Before we can perform the "update-on-attach", we have to bring the zone into the `configured` state, because it is still in `installed` state, as required by the HA Container agent. We do this by detaching it.

One important consideration at this step is how to deal with the data set for the data and the application. In this example we have not created a snapshot for them, but the `zoneadm attach` operation requires the data set to be present. As already discussed in "Step 3: Perform the Update Process." we create an empty ora_dpool.

```
[JERRY:root] mkfile 100M /tmp/zpool_tmp
[JERRY:root] zpool create -m none ora_dpool /tmp/zpool_tmp
[JERRY:root] zfs create ora_dpool
[JERRY:root] zoneadm -z ora_zone detach
[JERRY:root] zoneadm -z ora_zone attach -U
Getting the list of files to remove
Removing 5 files
Remove 13 of 13 packages
Installing 8 files
Add 13 of 13 packages
Updating editable files
The file </var/sadm/system/logs/update_log> within the zone contains a log of the
zone update.
```

Then the "update-on-attach" is performed, which brings our non-global zone to the level of its global zone and into `installed` state.

Step 4: Booting the Updated Zone Once

Prior to re-integrating the updated zone into the cluster, it has to be booted once, manually. The reason is, that due to new packages and services, there might be some extra work to be done as part of the boot process. If this would be done under cluster control, the boot process could easily run out of the START_TIMEOUT and be stopped by the cluster.

One difficulty during this boot process lies in the fact that the zone might have one or more IP addresses associated with it, that must not be started because they are still in use in production with the ora_zone on node TOM. If these IP addresses are configured as part of the cluster RG, nothing can happen during the zone boot process, because the cluster is not involved here. But if the IP addresses are part of the zone configuration, they either have to be changed to non-conflicting addresses or temporarily be removed. An alternative would be to boot the zone in a sandbox network environment.

When you login to the booted zone now, the zone prints "ORA-zone" which is correct, because it is a clone of the original ora_zone. Be aware that ora_zone is the zone name that is used from the global zone, whereas ORA-zone is the hostname of the zone that was given to it during its creation – hostnames must not contain the character '_'!

```
[JERRY:root] zlogin -C ora_zone
[Connected to zone 'ora_zone' console]

ORA-zone console login: root
Password:
Jun  9 11:04:11 ora-zone login: ROOT LOGIN /dev/console
Last login: Wed Jun  8 11:37:21 on console
Oracle Corporation      SunOS 5.10      Generic Patch   January 2005
```

Step 5: Re-Integrating the Updated Zone into the Cluster RG

We can now integrate the zone, that we updated on the new node, back into the cluster. A little care has to be taken because we have one RG that must use different storage resources on the two nodes. This looks complicated and risky, but is rather not.

Because we are re-using the existing cluster RG, we only have to be careful with changes made to the resources that are used. The IP address has not been changed and thus the LogicalHostname resource does not need to be changed either. The zone resource has not been changed, except for the patch update. Only the underlying storage zpool has changed, which is irrelevant for the zone itself. It only needs a valid zone configuration and a valid zonepath. The name and technology of the underlying filesystem are irrelevant.

So, what has to be changed is the HAStoragePlus resource, because it still points to the old zpool holding the old zonepath. We'll stop the zone resource on the old node TOM, change the nodelist property of the RG, change the HAStoragePlus resource and finally switch the RG to the new node and boot the zone by enabling the zone resource.

In case you used a temporary data pool, it must be removed because otherwise the cluster will find duplicate names. If you changed your zone configuration, you must revert these changes. You should also shut down the updated zone, so that it can reboot under cluster control.

The following steps are the ones that control the duration of the service outage. The main part of the time will be spent in shutting down the zone which might include an application shutdown, plus the restart of the zone including its applications on the other node.

```
[JERRY:root] clrg set -p nodelist=TOM,JERRY ora_zone-rg

[JERRY:root] clrs disable ora_zone-rs
[JERRY:root] clrs set -p Zpools=ora_zpool-cloned,ora_dpool ora_zone-hasp
[JERRY:root] clrg switch -n JERRY ora_zone-rg
[JERRY:root] clrg set -p nodelist=JERRY ora_zone-rg
[JERRY:root] clrs enable ora_zone-rs
```

To allow for a switchover to the updated node, we have to set the nodelist property back to include both cluster nodes. Then we stop ora_zone, by disabling its resource. This is where the downtime of the application service begins. Now we can change the storage resource.

We then switch the RG to JERRY, set the nodelist to contain only JERRY and boot the zone by enabling the zone resource again. Because we have booted the upgraded zone once, we can be sure, that this will also work under cluster control. The downtime for the application is now over.

We are done now, and can upgrade TOM to the latest version, which is a standard procedure and does not need any additional explanation.

To have a clear naming convention, it makes sense to rename the old zpool by importing it with a different name.

```
[TOM:root]  zpool import ora_zpool ora_zpool-retired
```

One could also rename the cloned zpool to have the original name. Whether this is needed and how the additional steps are integrated into the example, is left to the reader.

Summary of the Update Process for Failover Zones

The process performed with these steps is very elegant. Once the other, old node is updated as well, ora_zone could again be switched between the cluster nodes, although the old ora_zone might still exist on the other zpool which holds its zonepath. Because there is no direct relationship between a zone and its underlying storage (zpool), but only between the zone and its zonepath, the difference between the two zones is only visible at the zpool layer, which is represented in the HAStoragePlus resource.

Trying to retain the old RG and at the same time creating a new one is not a good solution. Because the IP address and the zone resource are identical, they could only be part of one RG at a time. In order to have two RGs, one would have to create a brand new zone with a new IP address and put these two resources together with the storage resource in a new RG. It is a lot more work, creates a lot more resources and does not lead to a clearer configuration.

Performing a Fallback of a Failover Zone

The elegance of the process described above shows its beauty also in its fallback scenario. We just have to revert the failover process to change the HAStoragePlus resource to point to the old pool with the old zonepath and switch the RG back.

Remember that cluster commands can be issued on any cluster node.

```
[JERRY:root] clrs disable ora_zone-rs
[JERRY:root] clrg set -p nodelist=TOM,JERRY ora_zone-rg
[JERRY:root] clrs set -p Zpools=ora_zpool-retired,ora_dpool ora_zone-hasp
[JERRY:root] clrg switch -n TOM ora_zone-rg
[JERRY:root] clrg set -p nodelist=TOM ora_zone-rg
[JERRY:root] clrs enable ora_zone-rs
```

In short, we stop the zone, set the nodelist so that we can switch the RG so that the IP address and storage resource will be online on the old node. We change the storage resource to include the old pool, which we renamed to ora_zpool-retired, set the nodelist back to the TOM and put the zone resource online.

If we use a snapshot of the data pools as well, we can create an even better fallback solution because the data at the time of the old zone shutdown would still be available. Integrating a cloned data pool into this example is straight forward. Integrating it into the cluster RG would just mean to change the name of the ora_dpool to ora_dpool-cloned in step 5. The other steps would be identical to the ones in “Performing a Fallback (The Extra Step)“.

Conclusion

We have shown how a highly consolidated infrastructure, based on Oracle Solaris Containers can be patched or updated independently with minimal downtime. ZFS and its snapshot and cloning features are the base for this new approach. Update-on-attach delivers the update of the zones as part of the attach process. Using this approach in environments based on Oracle Solaris Cluster demonstrates the same kind of downtime minimization.

Acknowledgments

Many thanks to Heiko Stein from etomer GmbH and Frank Batschulat from Oracle Solaris Zones Engineering for their comments and advice. We also would like to thank Stephanie Choyer, Enda O'Connor, Penny Cotton, Eve Kleinknecht, Karen Perkins and Kemer Thomson.

Bibliography

[1] How to Upgrade and Patch with Oracle Solaris Live Upgrade, Oracle Whitepaper, May 2010 by Jeff McMeekin
http://www.oracle.com/technetwork/server-storage/solaris/solaris-live-upgrade-wp-167900.pdf

[2] System Administration Guide: Oracle Solaris Containers-Resource Management and Oracle Solaris Zones
http://download.oracle.com/docs/cd/E18752_01/html/817-1592

[3] Oracle Solaris Cluster Data Service for Solaris Containers Guide
http://download.oracle.com/docs/cd/E18728_01/html/821-2677/

[4] Maintaining Solaris with Live Upgrade and Update On Attach; Sun Microsystems Blueprint, September 2009 by Hartmut Streppel, Dirk Augustin and Martin Müller
http://www.oracle.com/technetwork/server-storage/archive/a11-028-sol-live-upgrade-455915.pdf

[5] Using Live Upgrade in complex environments, Enda O'Connor
http://blogs.oracle.com/patch/entry/using_live_upgrade_in_complex

[6] ZFS Best Practices Guide
http://www.solarisinternals.com/wiki/index.php/ZFS_Best_Practices_Guide

[7] Solaris 10 Container Guide, November 2009 by Detlef Drewanz, Ulrich Gräf, et al.
http://blogs.oracle.com/solarium/resource/solaris-container-guide-en-v3.1.pdf

[8] Oracle Solaris ZFS Administration Guide
http://download.oracle.com/docs/cd/E18752_01/html/819-5461

Minimizing Downtime While Updating Oracle Solaris Containers in Clustered and Non-Clustered Environments
October 2011, Version 1.0
Author: Hartmut Streppel, Detlef Drewanz

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

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

Hardware and Software, Engineered to Work Together