http://www.sun.com/bigadmin/content/submitted/ha_containers_zfs.jsp
Aug 31, 2008

BigAdmin System Administration Portal

Community-Submitted Article

Installing HA Containers With ZFS Using the Solaris 10 5/08 OS and Solaris Cluster 3.2 Software

Luke Williams, July 2008

Installing high availability (HA) Containers on ZFS is pretty tricky. The documentation from Sun seems a little incomplete when going over this subject, but for the most part, it points you in the right direction.

This article covers the following topics:

Software and Hardware Environment

Here is the hardware and software I am running:

- Two Sun Fire 280R servers with 2 Gbytes of RAM and one 1.2-GHz processor. They each have two 68-Gbyte hard drives that are laid out like this:
  - `c1t0d0s0`: 14-Gbyte `/` that is mirrored to `c1t1d0s0` using Solaris Volume Manager
  - `c1t0d0s1`: 2 Gbytes of swap space
  - `c1t0d0s3`: 8-Gbyte `/var` that is mirrored to `c1t1d0s3` using Solaris Volume Manager
  - `c1t0d0s4` and `c1t0d0s5` are 10 Mbytes for the metadatabase, and the metadatabase is on `c1t0d0s4` and `c1t0d0s5`
  - `c1t0d0s6`: 512-Mbyte `/globaldevices` for clustering
  - `c1t0d0s7`: 43-Gbyte ZFS file system with `c1t1d0s7` as store with another ZFS file system called `store/home` mounted to `/export/home`
- One StorEdge 3310 SCSI array (as "just a bunch of disks" [JBOD]) with eight 68-Gbyte hard-drive ZFS file systems to one RAIDZ2 storage pool called `tank`, and one 18-Gbyte hard drive with two slices; `s0` is a 2-Gbyte quorum device and `s1` is 16 Gbytes of shared storage, yet to be used
- Solaris 10 5/08 OS for SPARC platforms (HW508)
- Solaris Cluster (formerly called Sun Cluster) 3.2 2/08 software

Installing the OS and Creating a ZFS Pool

First things first...install your OS. I set up mine as described in the Software and

Hardware Environment section because I have not yet figured out how to get SPARC to boot ZFS. Once that is available, I will be all over that. Until then, I use Solaris Volume Manager to set up my metadevices and use that to mirror my root and `/var` file systems.

Make sure both nodes look the same and have the same root password and user IDs for your users. That way, when you set up the cluster, everything will work nice and smooth.

I then created a ZFS pool called `store` for storage for everything else that I mount:

```
zpool create -f store c0t0d0s7 c1t0d0s7
zfs create store/home
zfs set mountpoint=/export/home store/home
zfs set sharenfs=rw store/home
zfs set quota=10g store/home
```

Connecting SCSI Cables and Resolving SCSI Resets

After I set up my initial ZFS pool, I ran into a couple of issues with my JBOD when I first hooked it up to the servers. One issue was that I didn't have enough SCSI cables, so I had to order those as well as another NIC for my interconnect devices.

Once I had all that set up, it was pretty straightforward. After I got my SCSI cables, I hooked them up to the JBOD, as shown in Figure 1.

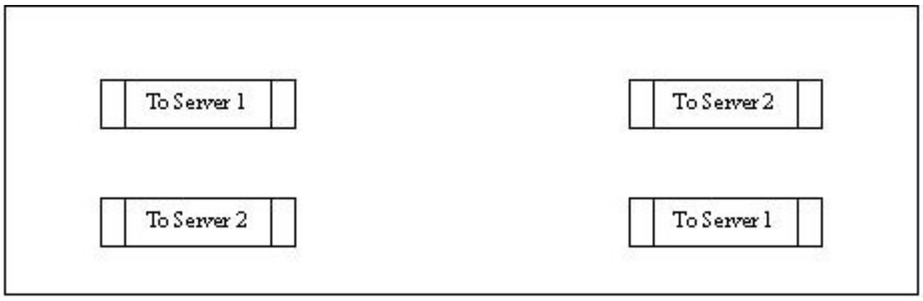

Figure 1: JBOD and SCSI Cables

I found I was getting SCSI resets on the bus. This was due to the `scsi-initiator-id` being set to 7 on both servers. I went into the OpenBoot prompt and simply used the following command on one of the servers:

```
setenv scsi-initiator-id 5
```

Then I rebooted both servers and tried to see all the disks. All of them showed up on one server, and none showed up on the other server.

So I then ran `devfsadm`, and just like magic, all my hard drives in the JBOD were there, and I was able to see the JBOD and all the disks on both servers.

Installing the Solaris Cluster 3.2 Software

I installed the Solaris Cluster 3.2 software. I made sure I installed the core and the following packages:

- `SUNW.HAStoragePlus` (HA Storage)
- `SUNWsczone` (HA Containers)

- `SUNW.nfs` (HA NFS, which is needed if you want to share your ZFS file system through NFS)
- `SUNW.gds` (HA Generic Data Services)
- `SUNW.LogicalHostname` (to give your cluster a host name that is common across the cluster)

You can install anything else you want. You just have to make sure you have these packages if you want to have high availability for your zones and ZFS file system. I recommend installing everything; you never know what your cluster will be used for in the future.

After you install the Solaris Cluster software, download all the patches and security updates from Sun, so your system is all safe and snug. Then reboot your system after installing the patches.

Creating Storage Pools for the Zones
Now we can make the storage pool for our zones. I made one giant `raidz2` pool on the JBOD by typing the following:

```
zpool create -f tank raidz2 c2t8d0 c2t9d0 c2t10d0 \
c2t11d0 c3t8d0 c3t9d0 c3t10d0 c3t11d0
zfs set mountpoint=/share tank
mkdir /share/HA
```

Note: I made the `/share/HA` directory for a later step. This is where my HA NFS settings will go. It will all make sense later on. There is a method to my madness...

Installing the Cluster
Now it is time to install the cluster:

```
/usr/cluster/bin/scinstall
```

Here is a text walk-through on how to set up your cluster. You tell it how many nodes, their names, and how the nodes will talk to each other. I don't use switches for the interconnects, I just use crossover cables, and I use the default of `/globaldevices`.

After both nodes are configured and rebooted, the system comes up in cluster mode, finds all the devices, and sets up a quorum device. At this step in the install "wizard," I tell it I will define the quorum device, which is my 18-Gbyte hard drive in the JBOD with the 2-Gbyte slice.

When I ran `scinstall` again after the reboot, it took me straight to installing the quorum device. Unfortunately, I didn't catch what the device was when the cluster attached itself to all the disks, so I had to pop over to the other node and type `scdidadm -L`.

This command allows you to see all the disks attached to both nodes. I remembered that `c2t12d0s0` was my quorum device, and it was given DID 11. So I just put in `d11`, and it found `s0`, and that was my quorum device.

My system was now a fully functioning cluster with nothing shared and monitored...yet.

Setting Up the Resources

Now we get to the fun stuff, which is actually setting up the resources. This part is the biggest headache, but luckily, Sun has made it pretty straightforward.

The first thing you are going to need is a resource group in which to home all the resources that your cluster is going to need. Make sure you are logged in either as root or as a system administrator.

First, register the services:

```
clresourcetype register SUNW.HAStoragePlus SUNW.nfs \
SUNW.gds
```

Then create the resource group, with the `PathPrefix` of `/share/HA` for HA NFS. The directory has to exist before you use this command or you can't use this path prefix. This location will be where my `dfstab` goes for all my NFS shares that will be in HA.

```
clresourcegroup create -p PathPrefix=/share/HA tank_rg
```

Here is a brief walk-through of the previous command: You are telling the cluster software to create the resource group `tank_rg`.

I like to keep the naming convention `poolname_rg` for my resource groups, `servicename_rs` for the services resource, and `hostname_lh_rs` for the host name logical host resource. It makes it easier for me to find errors in the syslogs if I use a naming standard like this.

I then set up the logical host name of the server.

Note: This name must be in the host file on both nodes, and it must be tied to an IP address.

First I set up the host name in `/etc/hosts` using vi.

```
192.168.101.5 logicalname
```

I then save the file using `:wq`, and I then create the resource:

```
Clreslogicalhostname create -g tank_rg -h logicalname \
logicalname_lh_rs
```

Setting Up the HA Storage

Now we set up the HA storage:

```
clresource create -g tank_rg -t SUNW.HAStoragePlus -p \
Zpools=tank tank_hastorageplus_rs
```

Now your storage pool is running under the Solaris Cluster software. You have to start your cluster resource group to see your storage pool again.

```
clresourcegroup online -M tank_rg
```

If you run `df -k`, you'll see your storage pool, but it will be mounted where you have your mount point set, for example:

```
tank    420037632 7730096 403406064  2%  /share
```

Now, before NFS will work, we need to set up a couple of directories and files.

Remember how we put `/share/HA` in the path prefix for our resource group? Well, NFS will use that path for its configuration and where it will keep its lock files so that your NFS share will actually have high availability. We have to add a directory to it and a copy of the `dfstab` with the resource name added, since the configuration of the HA NFS resource looks for this directory and file.

```
mkdir /share/HA/SUNW.nfs
cp /etc/dfs/dfstab /share/HA/SUNW.nfs/dfstab.tank_nfs_rs
```

After copying the file, go to the directory and use vi to edit your `dfstab`:

```
vi dfstab.tank_nfs_rs

share -F nfs -o rw -d "HA NFS File Share" /share/nfs

:wq

mkdir /share/nfs; chmod 777 /share/nfs
```

Now we create the resource for NFS:

```
clresource create -g tank_rg -t SUNW.nfs -p \
Resource_dependencies=tank_hastorageplus_rs \
tank_nfs_rs
```

Now we have NFS in HA, ZFS storage in HA, and a logical host name all ready for action.

Building the Zone

Next we build our zone. It is important to note that when you build your zone, if you share the default directories (`/usr`, `/platform`, `/sbin`, and `/lib`), they have to be identical on both nodes. The easiest way around this is to give your zone its own ZFS mount point, and then use the `create -b` option in `zonecfg` and make sure you create your ZFS pool on your newly created HA storage.

```
zfs create tank/zones
zfs create tank/zones/testzone
zfs set quota=5g tank/zones/testzone
```

This will set up the ZFS pool that our zone will live in. I used only 5 Gbytes because all that this test zone will be used for is to check other UNIX servers in my organization for faults and to email me the results. The entire Solaris 10 zone installation uses 3 Gbytes of space, which gives me 2 Gbytes for patches and any other small software I might want to install in the future.

Now we set up the zone:

```
zonecfg -z testzone
```

```
create -b
set autoboot=false
set zonepath=/share/testzone
add net
set address=192.168.101.6
set physical=eri0
end
verify
commit
exit
```

Make sure you have `autoboot` set to `false`. Our zone will be started and stopped by an HA script in the Solaris Cluster Zone package. You can't install the zone in HA if you have `autoboot` enabled.

Now we install, boot, and configure the new zone:

```
zoneadm -z testzone install
zoneadm -z testzone boot
zlogin -C testzone
```

Once you have your configuration complete, press ~. to terminate your console session on your zone. Then run the following command:

```
zoneadm -z testzone halt
```

You must halt your zone so that you can run the scripts for installing the zone in HA.

Now we have to edit some files so that we can use them to create our zone resource.

When you installed HA Containers, it was installed in `/opt/SUNWsczone`. The only directory that I am worried about in this walk-through is `/opt/SUNWsczone/sczbt/util`, which is the boot scripts for the zones.

`/sczsh` is used for a startup script after the zone is up and running, which is useful if you want to start a program after your zone is running.

`/sczsmf` is used to get a Service Management Facility (SMF) service up and running after your zone is running.

Since I only want the zone to boot up and run, I will only use `/sczbt`.

I created a `/config` directory under `/share/zones` for my scripts for creating my HA zones:

```
mkdir /share/zones/config
```

I then copied `/opt/SUNWsczone/sczbt/util/sczbt_config` to `/share/zones/config`:

```
cp /opt/SUNWsczone/sczbt/util/sczbt_config \
/share/zones/config
```

I then used vi to edit the file so that it looked like this:

```
RS=testzone_rs
```

```
RG=tank_rg
PARAMETERDIR=/share/zones/config
SC_NETWORK=false
SC_LH=
FAILOVER=true
HAS_RS=tank_hastorageplus_rs
Zonename="testzone"
Zonebrand="native"
Zonebootopt=""
Milestone="multi-user-server"
LXrunlevel="3"
SLrunlevel="3"
Mounts=""
```

This is what the previous commands mean:

- `RS` specifies the name of the resource your zone will use.
- `RG` specifies the name of the resource group your resource will fall under. This must exist before you run the script.
- `PARAMETERDIR` specifies the directory where your script and any parameters that you wish to pass on to your zone will be located.
- `SC_NETWORK` specifies how your zone gets its network ID.

  If you set it up using `zonecfg`, then use `false` here.

  If you want to set up your zone to use `logicalhosts` resource from your cluster, then enter `true`, and then put the `logicalhost` resource in `SC_LH`.

  Otherwise, leave `SC_LH` blank.
- `FAILOVER` specifies whether or not your zone can fail over. Since we want it to fail over, and it is on shared storage that runs under HA, we put `true` here and we put the HA storage name in the next line.
- `HAS_RS` specifies the HA storage resource name. This is needed if you want your zone to fail over.
- `Zonename` is pretty self explanatory; it is the name of your zone.
- `Zonebrand` specifies what type of zone you are booting.

  `native` is used by default. `lx` is for Linux containers and `solaris8` is used for Solaris 8 legacy containers.
- `Zonebootopt` specifies the zone boot options. The only option available is `-s` for single user, but because we want multiuser mode, just leave it blank.
- `Milestone` specifies what milestone level we want before the resource says it has started successfully to the cluster. The default of `multi-user-server` is good here.

- `LXrunlevel` is used only if `Zonebrand` is `lx`. You can leave the default because it is ignored if `native` is used.
- `SLrunlevel` is used only if `Zonebrand` is `solaris8`. You can leave the default because it is ignored if `native` is used.
- `Mounts` specifies any mounts that you want to have your zone load from the global zone after it is booted. Note that you need to have them able to be seen from your zone.

Now that we are all done writing our boot script, we have to register the resource:

```
clresourcetype register SUNW.gds
```

Now go to `/opt/SUNWsczone/sczbt/util` and run the `register` script:

```
/opt/SUNWsczone/sczbt/util/sczbt_register -f \
/share/zones/config/sczbt_config
```

Now you can enable the service:

```
clresource enable testzone_rs
```

Setting Up the Second Node

Now you have your zone up and running in HA. However, you're not quite done yet. You have everything set up only on one node of your cluster. If you fail it over now, everything but your zone will work.

You have two small things left to do.

There are two XML files that you need to copy to the other node. They are both located in `/etc/zones`. One is the index file and the other is the `testzone.xml` file.

Copy those files to your other node and put them in the `/etc/zones` directory. Boom...you can fail your cluster back and forth and to and fro, and you are running with no problems.

I used `rcp` to copy the files over because both nodes freely talk to each other anyway. So `rcp` seemed logical to use.

To get it working, I added each cluster node to a file called `.rhosts` in the `/` directory on the servers. I just used vi to write it.

On Node 1:

```
vi /.rhosts
node2 root

:wq
```

On Node 2:

```
vi /.rhosts
node1 root
```

```
:wq
```

Now you can use `rcp` to copy files from one server to the other. Just type the following to copy the XML files to the other node.

On Node 1:

```
rcp /etc/zones/testzone.xml node2:/etc/zones
```

On Node 2:

```
cd /etc/zones
mv index index.org
rcp node1:/etc/zones/index .
```

That's it. Your XML configuration for the zone is copied over.

You can now set up your zones to do what ever you want, and they should fail over to the other node with no issues. Enjoy.

About the Author
Lucas Williams has been a UNIX and Microsoft Windows system administrator for over 15 years. He has experience with various forms of Linux, the Solaris OS (releases 2.6 through 10), Novell Networking, and Microsoft Windows NT 3.51 through Microsoft Windows Server 2008. He writes scripts to make managing various systems easier and does beta testing of new technologies prior to deploying the technologies into a production environment.

Currently, he is working on using virtual technologies to run production-critical systems in a high availability environment.