Advanced - Creating an rpool manually

Generally, if you install OpenIndiana with the wizard, it creates your `rpool` for you, and the greatest influence you have on this process is to prepare the partition which would be wiped to create a default `rpool`.

The information and procedures below can help if you are migrating your system to a new `rpool` device, or perhaps if you are into writing new installation wizards.

The bootable "root" pools have some limitations compared to general-use "data" pools:

- In order for any part of the `rpool` to be sufficient at boot-time (which can include recovery from breakage of some devices), the pool must be a single-disk or a `mirror` device, not a `raidz` set. For the same reasons, it can not include a `log` device.
- Each `rpool` component should be a Solaris SMI slice encapsulated (for x86/x86_64) in an MBR table. This makes it tricky to create `rpools` on devices sized over 2TB which command a EFI/GPT table (shims are possible, but unsupported, to create and maintain an MBR table which addresses in primary partitions the same ranges as the GPT table, to specify the legacy OS partitions).
- In order to ensure proper alignment of the `rpool` slices on SSDs and "Advanced Format" HDDs, you might need to prepare the partition table as outlined in Advanced - Creating aligned rpool partitions.
- The illumos family of OSes are booted by a specially modified version of GRUB-0.97 bootloader provided along with the distributions. Other bootloaders, including other releases of GRUB or GRUB2, might chainload the partitions with this GRUB, but probably wont be able to boot illumos directly.
- The root of the pool should not be compressed, so as to not confuse GRUB which has a limited support of ZFS (though recent developments added support for `compression=lz4`). Certain datasets, such as `rpool/export` or components of the root filesystem hierarchy per Advanced - Split-root installation may be compressed with whatever algorithm – they are only processed and mounted by the illumos kernel with complete ZFS support.
- While it may be possible to set `dedup` on a root pool, it was advised on the mailing lists not to do so. Less complexity is better for the system data.
- The `rpool` is an important component of your system, in matters of both performance and reliability. You should keep it from overflowing (i.e. by setting up the split-root configuration and enabling quotas on potentially obese datasets – such as `/var/cores` or `/var/crash`, or store such datasets in a different pool). In order to increase space on the `rpool`, and to reduce writes to the device which houses it (in case of SSD), you might want to configure the `dump` and `swap` volumes to be stored on another pool as well. It is generally established that modifications to a filesystem (writes and deletions) are somewhat riskier operations than just reads, and given the importance of the `rpool` for booting and recovering your system, you should consider everything non-OS (including user data and local zones) on a different pool.

Now, assuming that you have already prepared an MBR partition and marked it as bootable, and prepared a Solaris SMI table in this partition, let's make some `rpool`!

```
:; zpool create -f -O checksum=sha256 -O org.openindiana.caiman:install=busy \
   -O atime=off -O compression=off -O com.sun:auto-snapshot=true \
   -o failmode=continue -o listsnapshots=off -R /a rpool c5d1s0
:; zfs umount rpool
```

- Note that with recent releases of the kernel and GRUB you may use `compression=lz4` on the `rpool`, but such pools won't be readable from older software versions.
- To get rid of writes incurred by directory reads, we set `atime=off`.
- Also we generally allow `zfs-auto-snapshot` service to make regular snapshots of datasets on this pool, whenever this service would be enabled.
- Finally, we specify an alternate root – the mountpoint under which the datasets of this pool would be auto-mounted, so that it wouldn't interfere with the running OS, and unmount the new pool's root dataset, so that it won't forbid us from mounting an OS root dataset a bit later.

Now, let's make the `dump` and `swap` volumes the way the installation wizard makes them (as of oi_151a8):

```
:; zfs create -o com.sun:auto-snapshot=false -o compression=off \
   -o primarycache=metadata -b `getconf PAGESIZE` -V 2045m rpool/swap
:; zfs create -o com.sun:auto-snapshot=false -o compression=off \
   -o dedup=off -o checksum=off -b 131072 -V 2045m rpool/dump

:; zfs create -o com.sun:auto-snapshot=false -o compression=off \
   -b 131072 -V 100m rpool/rsvd
```

- Note that the swap block size should match the system page size, which is 4096 bytes on x86_64 target platform for OI, and is generalized with g`etconf` callout above.
- In extreme low-memory situations, you might want to disable caching of even metadata in the RAM-based ARC, but note that disabling metadata in the primary cache would likely effectively disable caching anything in the secondary cache (SSD L2ARC) and would incur latency hits due to more reads from disk required to do anything with the data. That is, keep this possibility in mind, but only as a last resort and temporary workaround before you're forced to reboot a non-responsive system.
- The last command is not from the wizard, but rather a habitual "reservation" for the cases when your system did get filled up and you can't delete any files because snapshots reference them. In this case you can reduce or `zfs destroy` the `rsvd` volume and free up operational space.
- We disable automatic snapshots of the volumes, because they are pointless and will by default reserve the volume size (so that the volume contents can be fully overwritten), which is a big waste of `rpool`.

Now, let's prepare a simple dataset structure for the traditional monolithic root filesystem:

```
:; zfs create -o mountpoint=legacy -o canmount=off rpool/ROOT
:; zfs create -o mountpoint=/ -o canmount=noauto rpool/ROOT/openindiana
:; zfs set org.opensolaris.libbe:uuid=c2c6c968-9866-c662-aac1-86c6cc77c2c8 rpool/ROOT/openindiana
### UUID is random, individual for each BE; maintained by beadm in live systems
### This is where you can customize stuff for the root-BE dataset only; except compression.
:; zfs mount -O rpool/ROOT/openindiana
```

The administrative user's home also lives in `rpool`, and it can be compressed well:

```
:; zfs create -o mountpoint=/export -o compression=gzip-9 rpool/export
:; zfs create rpool/export/home
:; zfs create rpool/export/home/admin
```

To go deeper with compressing parts of the operating environment to save some space and IOPS, see Advanced - Split-root installation.

...After you've installed or migrated the OS to this new `rpool`, don't forget to make it bootable:

```
### Note the rpool disk(s):
:; zpool status rpool | grep ONLINE | awk '{print $1}' | egrep '^c.+s.$' | \
   while read SLICE; do echo "=== $SLICE"; \
     /sbin/installgrub /boot/grub/stage1 /boot/grub/stage2 "/dev/rdsk/$SLICE"; \
   done
:; zpool set failmode=continue rpool
:; zpool set bootfs=rpool/ROOT/openindiana rpool
:; zfs set org.openindiana.caiman:install=ready rpool

:; zfs set mountpoint=/rpool rpool
:; zfs mount rpool
### Just in case - should become /a/rpool
:; mkdir -p /a/rpool/boot/grub/bootsign
:; for F in capability menu.lst splash.xpm.gz; do \
   cp -pf /a/boot/grub/$F /a/rpool/boot/grub/; done

### Set up "boot signs" to point GRUB to bootable pools:
:; mkdir -p /a/rpool/etc:; touch /a/rpool/boot/grub/bootsign/pool_rpool
:; echo "pool_rpool" >> /a/rpool/etc/bootsign
:; touch /a/reconfigure
:; bootadm update-archive -R /a

### When you are ready to try the new pool...
:; init 6
```

- You can review some customizations of the installed operating environment and of the GRUB menu suggested in Advanced - Split-root installation and Advanced - Manual installation of OpenIndiana from LiveCD media; I won't repeat all those details here (and if you got here, you are probably following a setup procedure from one of those pages anyway).

Finally, as a trick that can help to quickly export and re-import the `rpool` in the Live Media or comparable environment, you can do something like this:

```
:; zpool export rpool
:; zpool import -N -R /a -f rpool && \
   zfs mount -O rpool/ROOT/openindiana && \
   zfs mount -a
```

HTH,
//Jim Klimov

See also:

- https://wiki.archlinux.org/index.php/ZFS -- many of the ZOL ideas are applicable to other ZFS branches as well... I cross-pollinated a few...