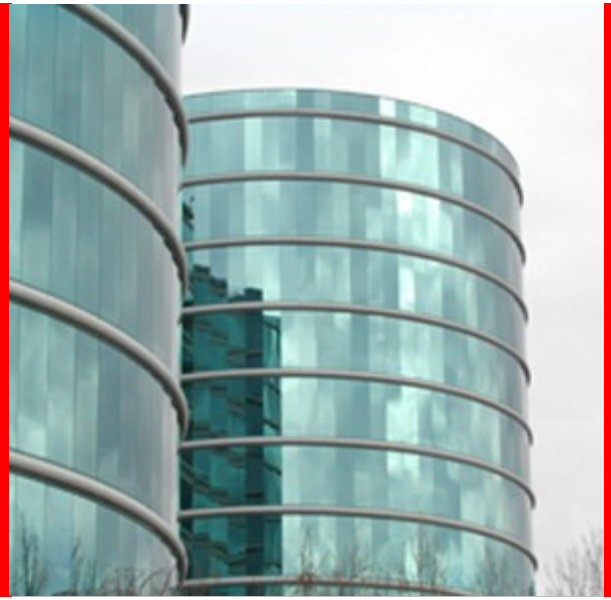

ORACLE®



ORACLE

Oracle Solaris 11 Express ZFS

Darren J Moffat
Oracle Solaris Core Technologies - Security

General OS 11/ZFS Feature Recap

- ZFS root/boot/install and patch/update features
- Read-only pool import
- Pool import with missing log device
- `zpool status` and `zpool list` improvements
- Deduplication
- Encryption
- `zfs send` and `zfs receive` backup options
- `zfs diff` to compare snapshot differences
- ACL interoperability improvements
- Performance improvements

ZFS Root/Boot/Install/Update

- ZFS is default root file system, UFS is not an option
- The beadm interface replaces `lu*` commands
- New auto installation keywords provide swap and dump volume resizing
- LiveCD and text-mode installer do not allow swap and dump resizing
- Mirrored root pool improvements:
  - CR 6668666 is fixed: If you attach a disk to create a root pool mirror, the bootblocks are automatically applied.
  - The `autoreplace` property does not reformat and apply bootblocks on a replaced root pool disk.

ZFS Root/Boot/Install/Update

- Patch and update improvements:
  - The `pkg image-update` or `pkg update` commands replace the need to create separate boot environments for patching and upgrades.
  - With IPS, no need exists to update and patch BEs. We update BEs with latest build that includes bug fixes and features.

ZFS Pool Import Recovery Features

- Attempt to import a pool with a missing log.

```
# zpool import dozer
The devices below are missing, use '-m' to import the pool anyway:
                c3t3d0 [log]
cannot import 'dozer': one or more devices is currently unavailable
```

- Import the pool with the missing log device.
- After the pool is imported, attach the missing log device. Similar recovery can be done with a mirrored log device. Just reattach both devices after the pool import.
- Run the `zpool clear` command to clear the pool errors.

ZFS Pool Import Recovery Features

- Importing a Pool in Read-Only Mode
  - If a pool is so damaged that it cannot be accessed, this feature might allow you to recover the pool's data:

```
# zpool import -o readonly=on tank
# zpool scrub tank
cannot scrub tank: pool is read-only
```

  - All file systems and volumes are mounted in read-only mode.
  - Pool transaction processing is disabled. In addition, any pending synchronous writes in the intent log are not played until the pool is imported read-write.
  - Attempts to set a pool property during the read-only import are ignored.
  - A read-only pool can be set back to read-write mode by exporting and importing the pool.

Improved ZFS Pool Messages

- The `zpool status` command has more scrub and resilver information.
  - Resilver in-progress report

```
 scan: resilver in progress since Thu Oct 28 14:08:04 2010
    24.3M scanned out of 12.2G at 3.03M/s, 1h8m to go
    24.0M resilvered, 0.19% done
```

  - Resilver completion message

```
scan: resilvered 12.2G in 0h14m with 0 errors on Thu Oct 28..
```

  - Scrub in-progress report

```
scan: scrub in progress since Thu Oct 28 13:50:22 2010
    98.6M scanned out of 13.3G at 16.4M/s, 0h13m to go
    0 repaired, 0.72% done
```

  - Scrub completion message

```
scan: scrub repaired 512B in 1h2m with 0 errors on ...
```

Improved ZFS Pool Messages

- Scrub and resilver completion messages persist across system reboots.
- The `zpool status` command includes an interval and count argument like the `zpool iostat` command.
- Both the `zpool list` and the `zpool status` commands include a `-T d` or `-T u` for a date and time stamp.
  - The `-T d` option includes date and time in `date.1` format:

```
# zpool list -T d
Thu Oct 28 15:04:07 MDT 2010
```

  - The `-T u` option includes date and time in `time.2` format:

```
# zpool status -T u
1288300086
```

ZFS Dedup - Recommendations

- ZFS deduplication synchronously removes redundant data blocks, but is disabled by default.
- Determine if data would benefit from dedup by simulating the potential space savings.

```
# zdb -S pool-name
```

  - If the estimated dedup ratio is above 1.0, then you might see dedup space savings.
- Determine the memory requirements as follows:
  - Use the `zdb -S` output to determine the dedup table sizing.
  - Each in-core dedup table entry is approximately 320 bytes.
  - Multiply the number of allocated blocks by 320. For example:

```
in-core DDT size = 3.75MB x 320 = 1200 MB
```

ZFS Dedup – Limits and Restrictions

- 20 TB of unique data stored in 128K records requires about 32 GB of physical memory.
- Performance is best when the dedup (DDT) table fits into memory. If the DDT table is written to disk, then performance decreases. Removing a large file system severely impacts performance if the system doesn't have enough memory.
- Writing the same data using different compression algorithms will result in data that cannot be deduped.
- Encrypted data can be deduped but only with in a "clone family" that shares data encryption keys.

ZFS Deduplication - Configuration

- The `dedup` property is set on a ZFS file system.

```
# zfs set dedup=on tank/home
```

- When enabled, only unique data is stored and common components are shared between files.
- A file system property, but the scope is pool-wide.

```
# zpool list tank
NAME SIZE  ALLOC FREE  CAP  DEDUP HEALTH ALTROOT
tank 136G  55.2G 80.8G 40%  2.30x ONLINE -
```

- The `zfs list` output might not display accurate space accounting if `dedup` is enabled.
- The process is compression -> encryption -> dedup, if all 3 properties are configured.

ZFS Dedup - Troubleshooting

- Troubleshooting tools
  - Use `zdb -DD` to display the size of the DDT.
  - DDT is considered metadata. Up to 25% of memory (`zfs_arc_meta_limit`) can be used to store metadata.
  - Monitor size of ZFS memory cache in bytes:

```
# kstat zfs::arcstats:size
```

- Review the ZFS Dedup FAQ for current issues and steps for determining whether a system has enough memory to support dedup.

ZFS Encryption - Configuration

- Encryption means that data is encoded for privacy. The data owner uses a key to access encoded data.
  - The `encryption` property is enabled when a file system is created and you are prompted for passphrase by default:

```
# zfs create -o encryption=on tank/home/darren
Enter passphrase for 'tank/home/darren': xxxxxxxx
Enter again: xxxxxxx
```

  - Default encryption algorithm is `aes-128-ccm` when a file system's `encryption` property value is `on`.
  - A wrapping key encrypts the data encryption keys. The wrapping key is passed from the `zfs` command when the encrypted file system is created to the kernel. A wrapping key is either in a file as raw or hex or derived fro a passphrase, or prompted for interactively.

ZFS Encryption - Configuration

- Encrypt a file system with a raw key from a file. Use `pktool` to generate a raw key to a file and then identify the `keysource` property value as a `file`.

```
# pktool genkey keystore=file outkey=/dmkey.file keytype=aes
keylen=256
# zfs create -o encryption=aes-256-ccm -o
keysource=raw,file:///dmkey.file tank/home/darren
```

- A file system's encryption policy is inherited by descendent file systems and cannot be removed.

```
# zfs clone tank/home/darren@now tank/home/darren-new
 Enter passphrase for 'tank/home/darren-new': xxxxxxxx
 Enter again: xxxxxxx
# zfs set encryption=off tank/home/darren-new
cannot set property for 'tank/home/darren-new': 'encryption'
is readonly
```

ZFS Encryption - Configuration

- Changing an encrypted ZFS file system's keys:
  - The existing wrapping key must be loaded first, either at boot time or by explicitly loading the filesystem key (`zfs key -l`) or by mounting the file system (`zfs mount` *filesystem*).

```
# zfs key -c tank/home/darren
Enter new passphrase for 'tank/home/darren': xxxxxxxx
Enter again: xxxxxxxx
```

  - The data encryption key for an encrypted file system is changed by using the `zfs key -K` command, but the new encryption key is only used for newly written data.

```
# zfs key -K tank/home/darren
```

  - The data encryption key is not visible nor is it directly managed. You need the `keychange` delegation to perform a key change operation.

ZFS Encryption - Configuration

- The `keysource` property identifies the format and location of the key that wraps the file system's data encryption keys.
- The ZFS `rekeydate` property identifies the last `zfs key -K` operation date.
- If an encrypted file system's `creation` and `rekeydate` properties have the same value, the file system has never been rekeyed by a `zfs key -K` operation.
- Loading/unloading a dataset key with the `zfs key -l` and `zfs key -u` commands require the key permission. The mount permission is generally needed as well.
- Changing a dataset key with the `zfs key -c` and `zfs key -K` commands require the `keychange` permission.

ZFS Encryption - Limits/Restrictions

- You can’t boot from an encrypted file system.
- Migrating encrypted or unencrypted ZFS file systems:
  - You cannot send an unencrypted dataset stream and receive it as an encrypted stream even if the receiving pool's dataset has encryption enabled.
  - Use `cp -r`, `find | cpio`, `tar`, or `rync` to migrate unencrypted data to a pool/dataset with encryption enabled.
  - You can send an encrypted dataset stream and receive it as an encrypted stream but it is not encrypted during the send process.

ZFS Send Stream Enhancements

- You can send a ZFS snapshot stream with a certain file system property value, but you can specify a different local property value.
- You can also specify that the original file system property value is used when the snapshot stream is received to recreate the original file system.
- You can also disable a file system property when the snapshot stream is received.

ZFS Send Stream Enhancements - Examples

- The `tank/data` file system has the `compression` property disabled. A `tank/data` snapshot stream is sent to a backup pool and is received with the `compression` property enabled.

```
# zfs get compression tank/data
NAME        PROPERTY     VALUE     SOURCE
tank/data   compression  off       default
# zfs send -p tank/data@snap1 | zfs recv -o compression=on -d
bpool
# zfs get -o all compression bpool/data
NAME        PROPERTY     VALUE  RECEIVED  SOURCE
bpool/data   compression  on    off       local
```

ZFS Send Stream Enhancements - Examples

- If this snapshot stream is sent to a new pool to recreate the original `tank/data` file system, the original `compression` property value is applied by using the `zfs send -b` option.

```
# zfs send -b bpool/data@snap1 | zfs recv -d restorepool
# zfs get -o all compression restorepool/data
NAME              PROPERTY     VALUE     RECEIVED  SOURCE
restorepool/data  compression  off       off       received
```

ZFS Send Stream Enhancements - Examples

- If you want to disable a local property when it is received, use the `zfs receive -x` command.
- Send a recursive snapshot stream of home directory file systems with all file system properties reserved to a backup pool, but without the quota property values.

```
# zfs send -R tank/home@1020 | zfs recv -x quota bpool/home
# zfs get -r quota bpool/home
NAME                      PROPERTY  VALUE  SOURCE
bpool/home                quota     none   default
bpool/home@1020           quota     -      -
bpool/home/cindys         quota     none   local
bpool/home/cindys@1020    quota     -      -
bpool/home/tom            quota     none   local
bpool/home/tom@1020       quota     -      -
```

ZFS Snapshot Differences (`zfs diff`)

- Use the `zfs diff` command to determine snapshot differences.

```
$ ls /tank/home/timh
fileA
$ zfs snapshot tank/home/timh@old
$ ls /tank/home/timh
fileA fileB
$ zfs snapshot tank/home/timh@new
$ zfs diff tank/home/timh@old tank/home/timh@new
M       /tank/home/timh/
+       /tank/home/timh/fileB
```

ZFS Snapshot Differences (`zfs diff`)

- Snapshot differences are indicated as follows:
  - `M` indicates file or directory is modified or file or directory link count changed
  - `-` indicates file or directory is present in the older snapshot but not in the newer snapshot
  - `+` indicates file or directory is present in the newer snapshot but not in the older snapshot
  - `R` indicates file or directory is renamed

Misc newish “small features”

- rstchown per dataset instead of /etc/system
- ACL interoperability improvements
- logbias=latecy | throughput
- sync=standard|always|disable
- primarycache=none|all|metadata
- secondarycache=none|all|metadata
- mlslabel – for Trusted Extensions zones
- Remember quotas are per dataset & per user/group

ZFS Performance Improvements

- Better, faster block allocator
- Scrub prefetch
- Raw scrub/resilver
- Zero-copy I/O – used by NFS & CIFS
- Explicit sync mode control (sync property)
- RAID-Z/mirror hybrid allocation


ORACLE®