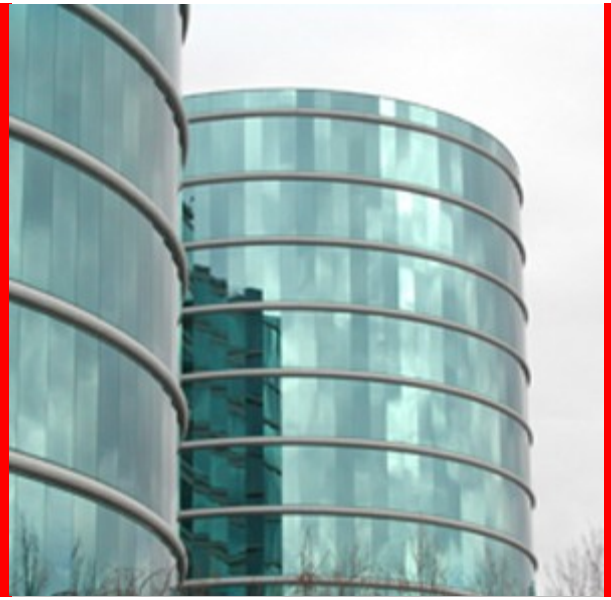


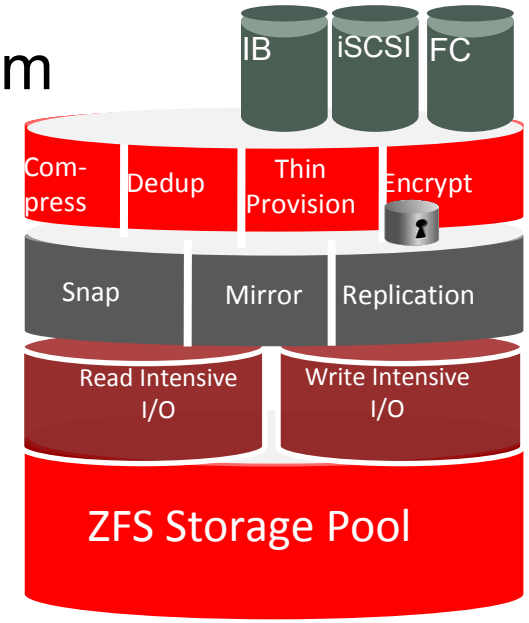
ORACLE

Data Management With ZFS

Rich Morris/Mark Musante
ZFS Team

ZFS: Simple, Powerful Data Management

- Oracle Solaris 11 default file system
  - Pooled storage
  - Data protection
  - Easy administration
- Features
  - End-to-end checksums
  - Built-in compression and encryption
  - Copy-on-write
  - Transactional semantics
  - Unlimited file systems, clones, and snapshots
  - Pool and file system properties, quotas, reservations, delegated admin, and so on



ZFS Features in Oracle Solaris 11

- Major performance improvements
- User quotas
- Root pool enhancements
- Pool recovery improvements
- Triple-parity RAID-Z
- Deduplication*
- Encryption*
- SMB sharing*
- Snapshot differences (`zfs diff`)
- Send stream enhancements (`zfs send/recv`)
- Shadow migration*

*New feature in Oracle Solaris 11

ZFS Performance Enhancements

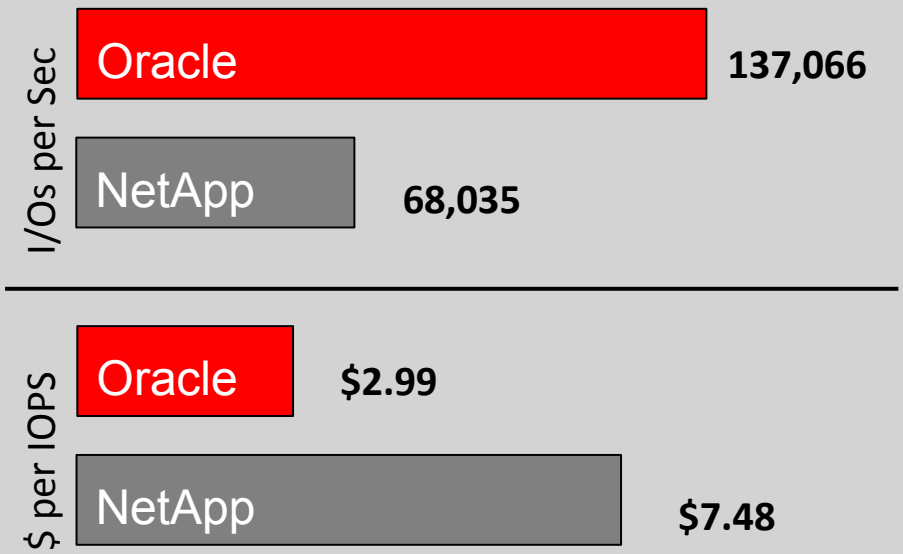
- Better, faster block allocator
- Scrub prefetch
- Raw scrub/resilver
- Zero-copy I/O
- Native iSCSI
- Hybrid storage pools
- Duty-cycle scheduler
- Intent log latency/throughput control
- Explicit sync mode control
- RAID-Z/mirror hybrid allocation
- Fast data encryption

Performance: ZFS Storage Appliances

2x faster and ½ the price of NetApp

SPC-1 Benchmark

SPC-1 is a benchmark based on a common Oracle Database workload. This benchmark demonstrates ZFS Storage advantage over NetApp in Oracle environments.



Note: Above numbers have been audited by SPC.

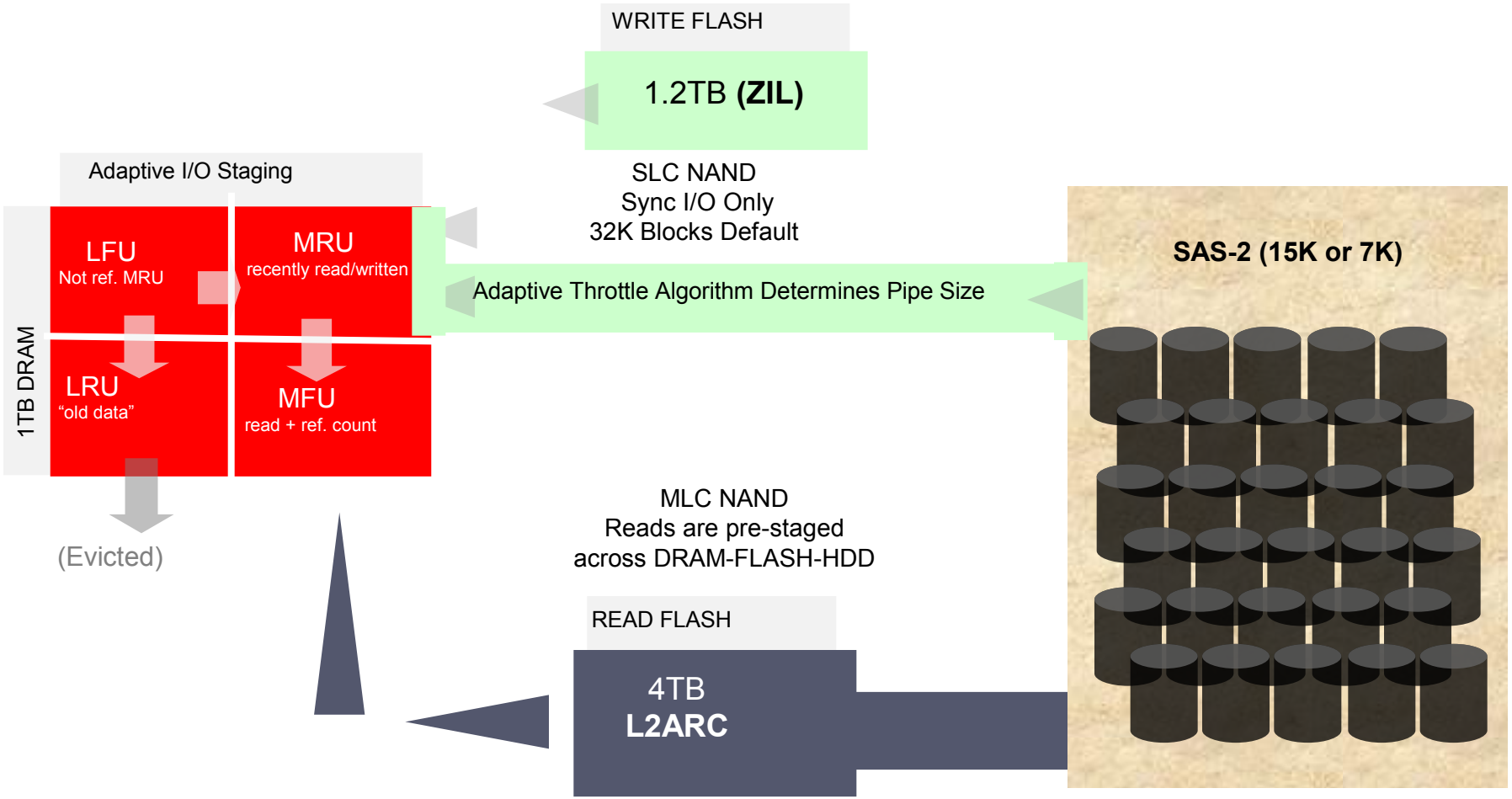
Note: SPC-2 Benchmark

Also Looking good….7GB/sec – Industry leader for dual controller array

SPC-2 is a benchmark based on a streaming workload such as VOD. It is a sequential read and write workload. This is not an audited number but coming soon…

Performance: ZFS Storage Appliances

- Engineered for performance and adaptive for next generation HW



Performance: logbias property

- Problem
  - Separate log (slog) devices are a limited resource
    - Synchronous bulk I/O crowds out latency-sensitive I/O
  - Not all sync I/O benefits from lower latency
    - When latency isn't critical, it's cheaper to go to disk
    - With many disks, available bandwidth is far higher
- Solution
  - Allow per-dataset log bias for latency vs. throughput
  - For Oracle redo logs:
    - zfs set logbias=latency
  - For Oracle data files:
    - zfs set logbias=throughput
  - Over 30% performance improvement

Performance: sync property

- Per-dataset control over synchronous semantics
  - Standard: follows the usual POSIX rules
    - Calls to fsync(), O_DSYNC write(), etc. commit to stable storage before returning
  - Always: makes every transaction synchronous
    - Explicit syncs not needed
    - Actually a performance win for some combinations of hardware and workload
  - Disabled: makes everything asynchronous
    - Huge performance win on systems without dedicated log devices when sync semantics aren't required

Performance: RAID-Z / Mirror Hybrid Allocator

- Use one group of disks as both RAID-Z and mirror
  - RAID-Z for user data and most metadata
    - Provides greatest capacity and highest write throughput
  - Mirror for latency-sensitive, dittoed metadata
    - Provides most read IOPs and lowest read latency
    - Ideal for small, randomly accessed bits of metadata
      - dnodes / indirect blocks
      - directories
      - dedup tables
- Some real-world workloads up to 4x faster
  - Copying large files with deduped blocks
  - rm -rf of giant directory

Root Pool: Install/Update Features

- The `beadm` interface replaces `lu*` commands.
- New auto installation provides these features:
  - Mirrored root pool creation
  - Separate `/var` file system
  - Swap and dump resizing
- The `pkg update` command replaces need to create separate boot environments for patching and upgrades.
  - Creates new BE automatically
  - No need to patch separately

Root Pool: Other Enhancements

- Bootblocks automatically applied at mirror attach
  - Works for spares
- ZFS send stream enhancements
  - Provide better management of root pool file system properties

Pool Recovery: The Problem

- ZFS pool integrity depends on explicit write ordering
  - Some cheap disks and bridges silently ignore it!
  - Result: uberblock written before data it points to
  - Power loss can lead to complete pool failure

Pool Recovery: The Solution

- Recover pool even if devices ignore write barriers
  - Verify integrity and completeness of last transaction group during pool open
  - If damaged, roll back to previous transaction group
  - Rollback made reliable by deferred block reuse
    - Recently freed blocks are never immediately used
    - Safe to assume they are still valid

```
# zpool import tank
cannot import 'tank': I/O error
Recovery is possible, but will result in some data loss.
Returning the pool to its state as of Wed Nov 30 16:27:23
MST 2011 should correct the problem Approximately 5 seconds
of data must be discarded, irreversibly.  Recovery can be
attempted by executing 'zpool import -F tank'.  A scrub of
the pool is strongly recommended after recovery.
```

Pool Recovery: Missing Logs

```
# 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 using `-m` flag
  - Pool will import in degraded mode
- Can re-attach the missing log device after import
  - Then, use `zpool online` to online the missing device
  - Works for mirrored logs too
- Run the `zpool clear` command to clear pool errors

Pool Recovery: Read-Only Mode

```
# zpool import -o readonly=on tank
```

- All file systems and volumes are mounted read-only
- Pool transaction processing is disabled
  - Intent logs are not played
- Attempts to set properties are ignored
- Can be set back to read-write mode by exporting and importing the pool.
- Use when pool is damaged
  - Writes fail
  - Allow recovery of pool data

Triple-Parity RAID-Z (RAIDZ3)

- Capacities of hard drives outpace their throughput
- When the time needed to rebuild a large disk increases so does the likelihood of data loss
- Larger disks mean longer rebuild times so we need more redundancy
- A RAIDZ3 configuration can survive the failure of 3 disks, or the failure of 2 disks plus occasional bad reads

ZFS Deduplication

- Only store one copy of identical data blocks
  - Three parts: on-disk, in-core, and over-the-wire (`zfs send/recv`)
- Key applications:
  - Virtualization
  - Backup servers
  - Build environments
- Fully integrated with ZFS (not an add-on)
  - No special hardware
  - No limits on capacity
  - Performance improvements in recent builds, but consider best performance is when DDT fits into memory

ZFS Deduplication

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

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

- Scope is pool-wide so use `zpool list` to identify deduped space reporting:
  - All datasets with dedup enabled share common blocks
  - Can identify the dedup ratio as follows:

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

- Using `zfs list` can be misleading when `dedup` is enabled because:
  - Does not account for deduped data blocks in space reporting
  - Can report more space used than available in pool

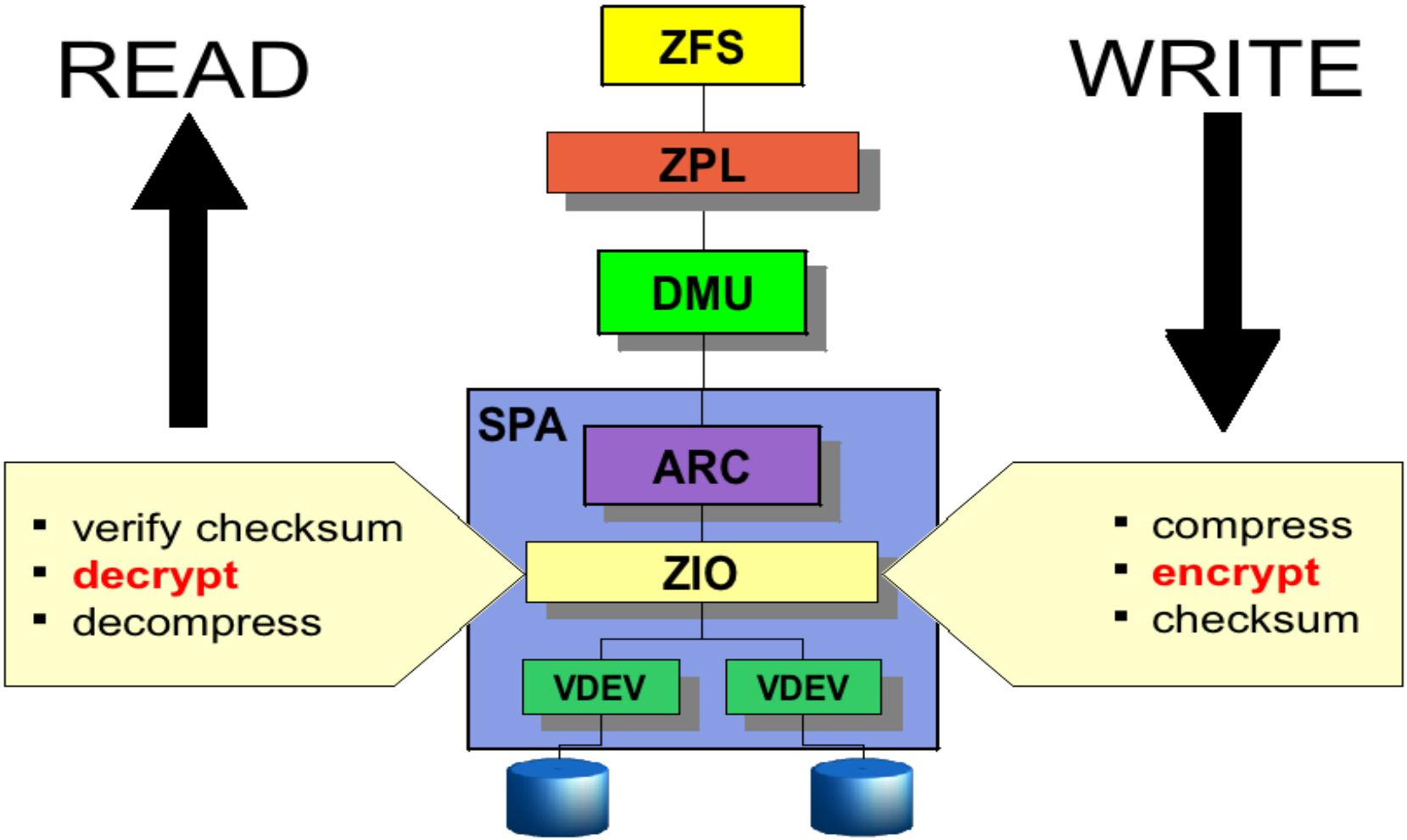
ZFS Data Encryption

- Encrypted data is encoded for privacy. The data owner uses a key to access encoded data.
- Enable the `encryption` property when a file system is created. You are prompted for passphrase:

```
# 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`.
- A wrapping key encrypts the data encryption keys.
- Key source can be raw or hex bytes or passphrase
- Key location include prompt, file, PKCS#11 token, or secure https location

ZFS Data Encryption



ZFS User and Group Quotas

- For enterprise customers:
  - Finer grained answer to: “Where did my space go?”
- For education customers:
  - Have many uncontrolled users, so want quota per user
- Deferred enforcement
  - User may overshoot quota (by one transaction group)
  - Once over quota, returns EDQUOT until back under quota
- Supports both SMB SIDs and POSIX UIDs/GIDs
- Set with `userquota` and `groupquota` properties:

```
# zfs set userquota@amy=10G pond/amy
# zfs set groupquota@staff=20GB students/staff
```

ZFS User and Group Quotas

- Display quota information with `zfs userspace` and `zfs groupspace` commands

```
# zfs userspace pond/amy
TYPE        NAME           USED  QUOTA
POSIX User  amy             14M     1G
POSIX User  rory           258M   none
SMB User    mark@oracle    103M     5G
# zfs groupspace students/staff
TYPE         NAME          USED  QUOTA
POSIX Group  staff        4.00G    20G
POSIX Group  root            3K   none
```

ZFS Snapshot Differences (`zfs diff`)

- Efficiently generate changes between snapshots
  - Show all files that have been added, removed, modified, or renamed

```
# zfs create tank/jeff
# cd /tank/jeff
# echo hello >hello.txt
# echo world >world.txt
# zfs snapshot tank/jeff@hello-world
# echo kitty >kitty.txt
# mv hello.txt goodbye.txt
# rm world.txt
# ls
goodbye.txt   kitty.txt
# zfs diff tank/jeff@hello-world
M   /tank/jeff/
R   /tank/jeff/hello.txt -> /tank/jeff/goodbye.txt
-   /tank/jeff/world.txt
+   /tank/jeff/kitty.txt
```

ZFS Snapshot Differences (`zfs diff`)

- Differences are indicated between snapshots or a snapshot and a file system:

```
# zfs diff tank/home/cindy@now tank/home/cindy
M       /tank/home/cindy/
+       /tank/home/cindy/file.1
```

- Differences are indicated as follows:
  - `M` indicates file or directory is modified or 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

ZFS Send Stream Enhancements

- Send a ZFS snapshot stream and specify a different local property value when received.
- Specify that the original property value should be used when the snapshot stream is used to recreate the original file system.
- Disable a file system property when the snapshot stream is received.

ZFS Send Stream Enhancements

- 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 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     -         local
```

ZFS Send Stream Enhancements

- If this snapshot stream is sent back to replace the original `tank/data` file system, the original `compression` property value is applied.

```
# zfs send bpool/data@snap1 | zfs recv tank/data
# zfs get compression tank/data
NAME       PROPERTY     VALUE   SOURCE
tank/data  compression  off     default
```

- If you send a snapshot stream back to replace the original file system, but you prefer to have the received property value override the local property value, you can use the `zfs send -b` command.

ZFS Send Stream Enhancements

- The `bpool/data` file system has the `compression` property enabled. The `bpool/data` snapshot is sent back to replace the original `tank/data` file system, and uses the received `compression` property value.

```
# zfs get -o all compression bpool/data
NAME        PROPERTY     VALUE RECEIVED SOURCE
bpool/data  compression  on    -        local
# zfs send -b bpool/data@snap1 | zfs recv tank/data
# zfs get compression tank/data
NAME        PROPERTY     VALUE     SOURCE
tank/data   compression  on        received
```

- Note that the `compression` source value for the `tank/data` file system is `received`.

ZFS Shadow Migration

- Migrates UFS or ZFS on local or remote systems
  - Mount old file system read-only
  - New file system will “shadow” the old one
- Files and directories immediately available
  - No need to wait for complete migration before using files
  - Renames and rewrites are allowed while migrating
- New file system may be shared while migrating
- User access and background threads work in parallel to progress migration
- Live data migration tasks
  - Migrate data from a *ABC* VM configuration to a ZFS pool
  - Migrate ZFS data from a RAIDZ pool to a mirrored pool

ZFS Shadow Migration

- Install the package. Start the shadow daemon.

```
# pkg install shadow-migration
# svcadm enable shadowd
```

- Set the local or remote file system to be migrated to read-only.

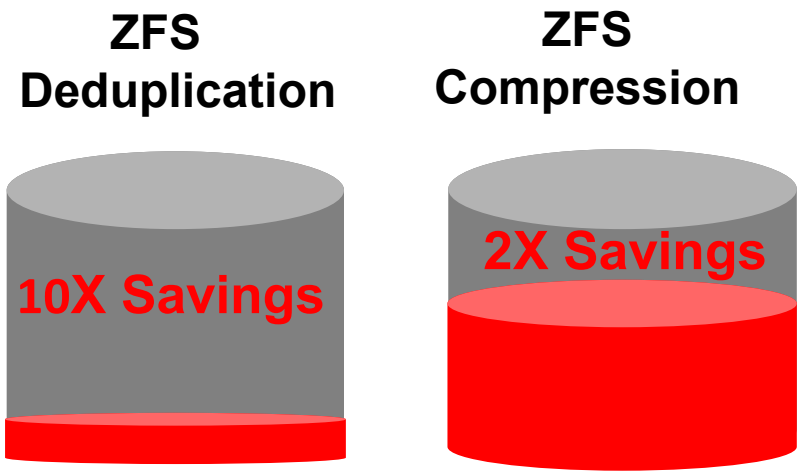
```
pond# zfs set readonly=on pool/ufsfs
```

- Create a ZFS file system with the `shadow` property set to the file system to be migrated.

```
drw# zfs create -o shadow=nfs://pond/pool/ufsfs
tank/zfsfs
```

ZFS: Simple, Powerful Data Management Summary

- Simply administration
- Built-in data services
  - **Deduplication, compression, encryption, replication, migration, snapshots, clones**
- Fully virtualized
  - **Pooled storage**
  - **Fully integrated with zones**
- Reduces storage consumption



ZFS Resources

- Oracle Solaris ZFS Administration Guide
  - http://docs.oracle.com/cd/E23824_01/html/821-1448/index.html
- ZFS Dedup and Encryption FAQ:
  - http://hub.opensolaris.org/bin/view/Community+Group+zfs/faq
- Join the conversation in zfs-discuss:
  - http://mail.opensolaris.org/mailman/listinfo/zfs-discuss


ORACLE®