ZFS / OpenStorage

Lee Marzke

IT Consultant - 4AERO

ZFS / OpenStorage

- ***About the Zettabyte Filesystem***
  - Developed by Sun Microsystems (now Oracle)
    - ”The final word in Filesystems”
  - License
  - Quick Overview of Nexenta GUI / CLI
  - New Terms / Definitions
  - Features and Limitations
  - Platforms and Products
- ***Demo***

ZFS / OpenStorage

- ***Why ?***
  - Replace Expensive Proprietary Storage (SAN)
    - Second Tier storage
    - Primary NetApp / EMC / Equallogic / HP ?
  - Use off-the-shelf 64bit hardware
  - Proveable integrety (top to bottom checksums)
  - Transactional – always consistant on disk
  - Unlimited file size and numbers

ZFS / OpenStorage

- ***ZFS***
  - ZFS 1995: The last word in Filesystems (Sept 2008) Jeff Bonwick and Bill Moore [15] (3 hour video)
- ***Open Storage***
  - As a general term, open storage refers to storage systems built with an open architecture using industry-standard hardware and open-source software.
  - Fishworks: Now it can be told: [12],[13,17] 2006/2008 Development of 1st NAS applianceon ZFS at Sun

ZFS / OpenStorage

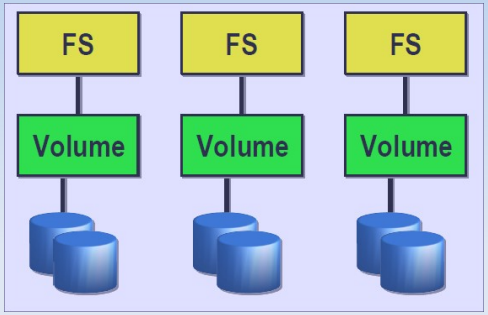
- ***ZFS License***
  - Open-Source CDDL (not compatible with GPL)
  - Linux userland (via FUSE module) implemented
    Read performance 50%
    Write performance 50% to 5% (random worse)
  - Linux kernel implementation not possible with current license. But this may change. See [1] (LLNL)

ZFS / OpenStorage

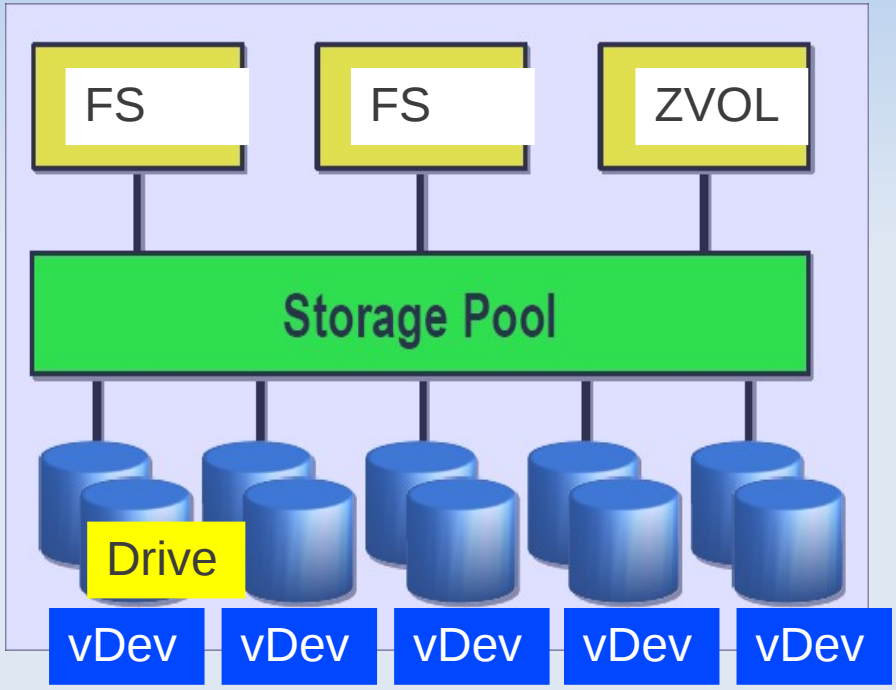
- ***Nexenta Overview***
  - Nexenta Core (Solaris Kernel + Ubuntu userland)
    - Open source – free (CLI only)
  - NexentaStor (Commmerical GUI + support)
    - Build on top of Nexenta Core
    - Free for up to 18TB used storage
    - Commercial: Approx $100/TB - $200/TB

ZFS / OpenStorage

- ***Terms***
  - No Volumes!
  - Drive
  - vDev
  - Zpool
  - DataSet
    - FileSystem
    - Clone / Snapshot
    - Zvol



Traditional



ZFS

ZFS / OpenStorage

- ***Nexenta Overview***
  - Demo appliance
  - (1) zPool - consisting of 8 devices
    - (3) mirrored vDevs
    - (1) cache SSD (read cache)
    - (1) log SSD (write log)
  - SSD are optional to speed up read/write

ZFS / OpenStorage

- ***Terms***
  - RaidZ – Parity drives added to survive failure
    Stripe width variable
    - RaidZ1 – one parity device
    - RaidZ2 – two partity
    - RaidZ3 – three parity
  - Resilver – process of creating/recreating a mirror device (mirror resync) or failed data or parity device

ZFS / OpenStorage

- ***Terms***
  - Cache Device – improves reads
    - ARC in RAM
    - L2ARC in SSD
  - Log Device – copy of transaction, speeds writes
    - ZFS Intent Log (ZIL) on SSD
  - MLC – multi-level cell SSD
    cheeper, best for reading, 10,000 writes life
  - SLC – single-level cell SSD
    best write performance, 100,000 writes life

ZFS / OpenStorage

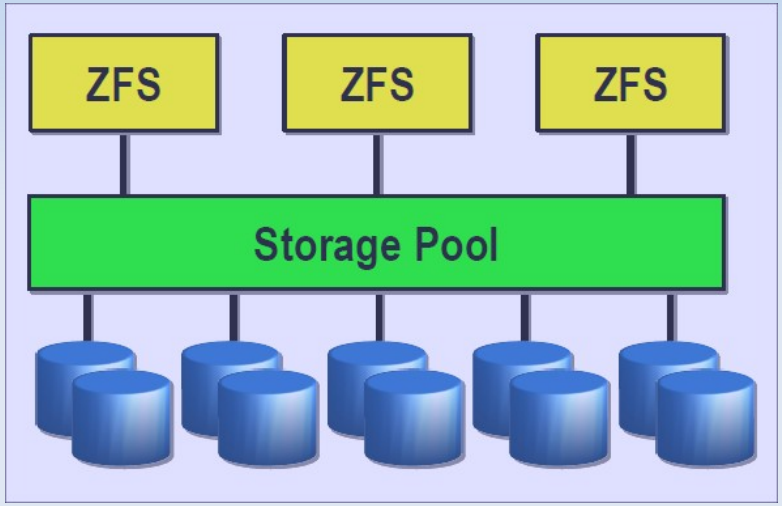
- ***Terms***
  - Hybrid Storage Management (HSM) Pool
    - Slow Primary disk
    - Medium performanc SSD cache (L2ARC)
    - Fast RAM cache (ARC)
  - ZFS Intent Log (ZIL)
    - ZFS writes are transactions
    - Write metadata and data to log
    - Commit log to main disk
    - For performance put ZIL on separate fast disk

ZFS / OpenStorage

- ***Development of ZFS at Sun***
  - ZFS: The last word in Filesystems (Sept 2008)
    Jeff Bonwick and Bill Moore [15] (3 hour video)
  - Fishworks: Now it can be told: [12]
    Development of 1st NAS appliance at Sun
    to make use of free ZFS stack on OpenSolaris

ZFS / OpenStorage

- ***WHY ZFS: Pooled Storage***
  - No volumes to resize/manage
  - Pools made up of 1 or more Vdev's
  - Vdev's in pool are striped
  - Vdev's can't be removed with present ZFS version



ZFS / OpenStorage

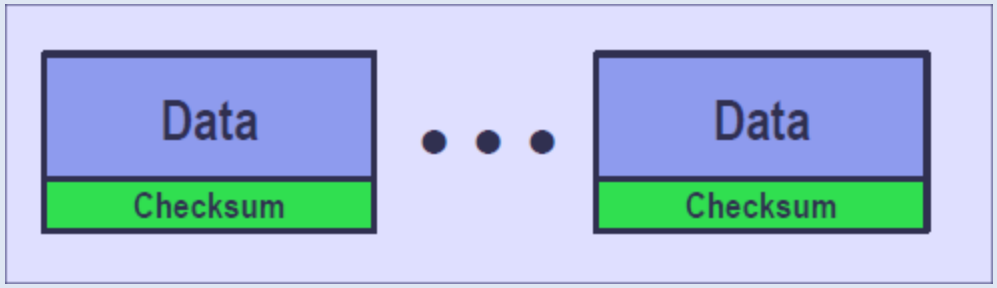
- ***Why ZFS: Transactions***
  - Transactions (meta-data and data)
    - Data first written to ZFS intent log (ZIL)
    - Then ZIL flushed to Disk
    - ZIL recommended to use dedicated SSD or Disk
  - Data on disk is ALWAYS consistant !
    - Except for TLER misfeature.
    - Disk sectors remapped – disk offline for seconds to possibly minutes

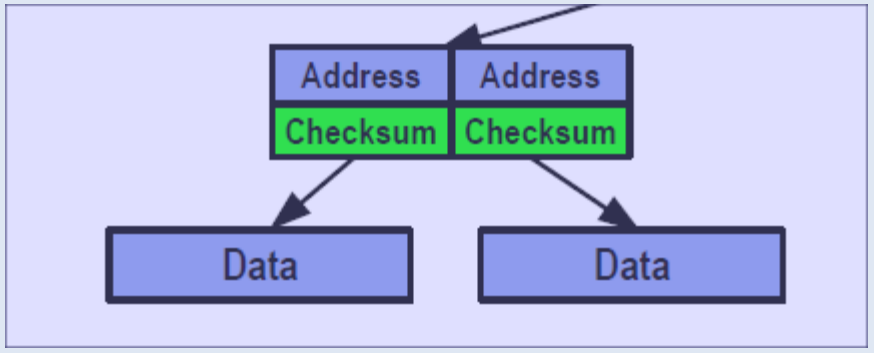
ZFS / OpenStorage

- ***Why ZFS: Transactions***
  - How 2 writes for each block can be faster:
    - Random writes are slowest
    - ZIL lives on a fast SSD device so random
      - writes to ZIL are fast
    - ZIL flushed each 5 seconds
      - ZIL flushed sequentially to disk

ZFS / OpenStorage

- ***Why ZFS: Checkums***
  - Traditional – checksum stored with block
  - ZFS – block checksum stored in meta-data
  - Each directory checksums contents below it (Merkle tree)





ZFS / OpenStorage

- ***Why ZFS***
  - Online Scrub instead of fsck
  - Scrub detects silent data corruption and repairs it.
    - Scrub reads all blocks, all parity, all meta-data.
    - Verify agains 256 bit checksum / repair as needed
    - Runs at low priority in background.
  - Recommended Scrub
    - Scrub 1/wk for Consumer drives
    - Scrub 1/month for Enterprise drives

ZFS / OpenStorage

- ***Why ZFS: Data Integrity***
  - Some blocks are more important (root meta-data)
  - ZFS adds Ditto Blocks [4]
    - Multiple copies on disk (not across stripes)
      User data – one copy
      FS meta data – two copies
      Global meta data – three copies

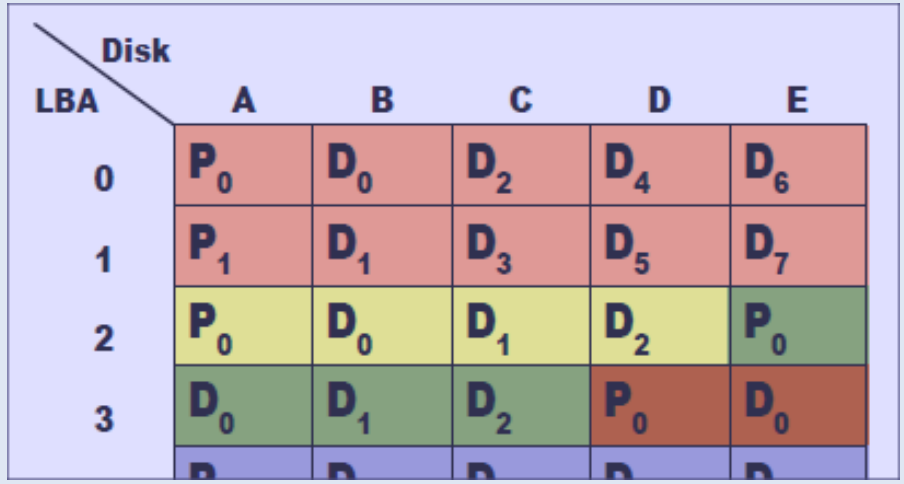
ZFS / OpenStorage

- ***Why ZFS: COW (Copy – on - Write)***
  - For RaidZx - no edit in place
    NO RAID5 write hole!
  - New copy of Entire stripe (data + parity) written
  - Variable width stripes
  - Because of ZIL
    everyting written 2X
  - This isn't BAD ! Why ?

LBA \ Disk	A	B	C	D	E
0	P_0	D_0	D_2	D_4	D_6
1	P_1	D_1	D_3	D_5	D_7
2	P_0	D_0	D_1	D_2	P_0
3	D_0	D_1	D_2	P_0	D_0

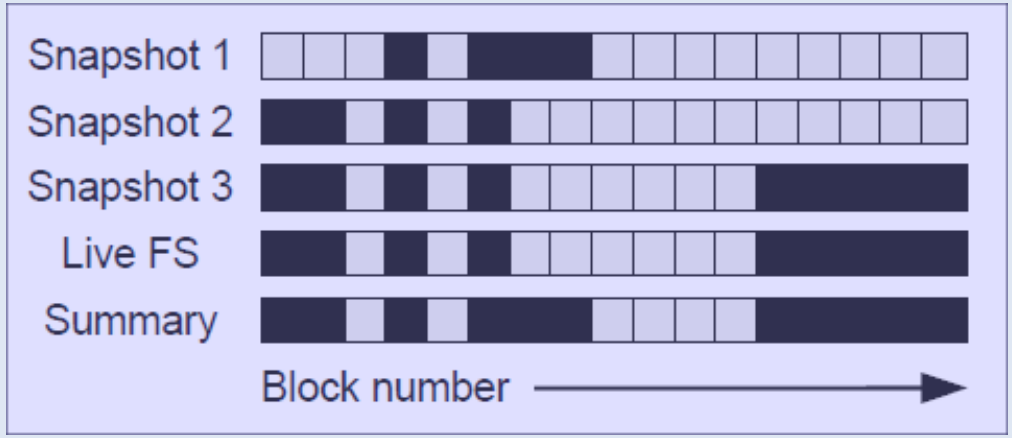
ZFS / OpenStorage

- ***Why ZFS: COW (Copy – on - Write)***
  - Most SAN writes are highly random
  - COW writes to ZIL on very fast SSD
  - After 5 seconds
    ZIL flushed ”sequentially”
    to main disk.
  - This turns random writes into sequential.



ZFS / OpenStorage

- ***Why ZFS: Snapshots***
  - Other FS: Bitmap per snaphot [15] p11
    - O(N) space, O(N) time (create, destroy)

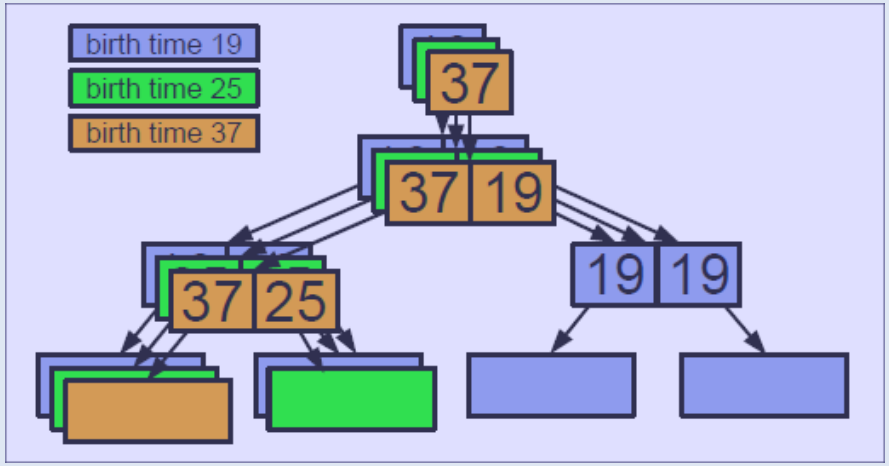


BITMAPS

Pointer + Birthtimes

ZFS / OpenStorage

- ***Why ZFS: Snapshots***
  - ZFS: Pointer with checksums, birthtimes (COW)
    - O(1) space, O(1) create time
    - O(Δ) delete (birthtime tree-walk)



BITMAPS

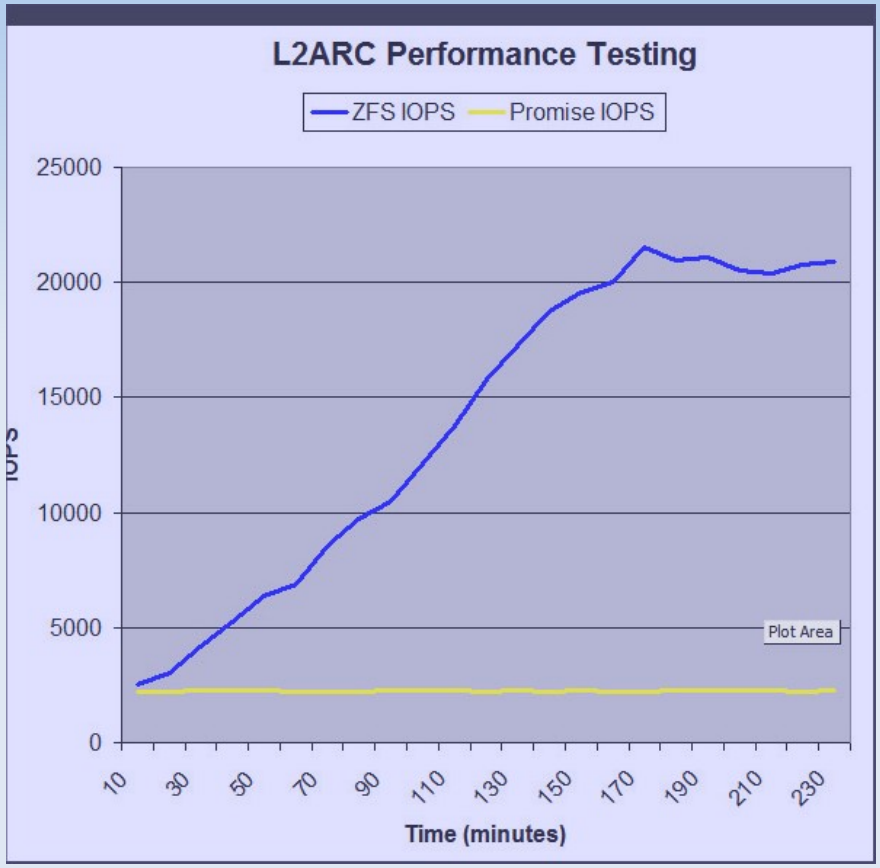
Pointer + Birthtimes

ZFS / OpenStorage

- ***Pooled HSM Storage (with cache+log)***
  - Typical Cache Setup for 10TB pool
    - 32GB RAM – 4GB for OS leaves 28GB ARC
    - (1) 500GB MLC SSD
    - Total of 0.5TB of cache
    - Use arc_summary for info
  - Typical ZIL Setup
    - Need to store (2) transaction groups of 5 seconds of write data (10 sec x 170Mb/s ~ 2GB required) ZIL Mirror recommended.
    - Use zilstat to monotor performance

ZFS / OpenStorage

- ***L2ARC test [10]***
  - Promise 610i
    16 x SATA
    -vs- ZFS
  - 100% random read
  - Cache population rate limited to slow wear on SSD's



ZFS / OpenStorage

- ***Compression***
  - Built in LZJB and GZIP
  - Encryption TBD
- ***Dedup***
  - Deduplication recently added
  - Dedup lookup table should fit in RAM
    - Requires lots of RAM and ZIL on SSD
- ***Send / Recieve***
  - ZFS snapshot Δ send/recieve (for replication)

ZFS / OpenStorage

- ***Limitations***
  - Can't remove vDev's from pool
  - Lots of RAM, and SSD's needed for good performance
  - ZFS capacity: 256 quadrillion ZB (1ZB = 1 billion TB)

ZFS / OpenStorage

- ***Sample Commands***
  - `zpool create tank mirror ctd0 c3d0`
  - `zfs create tank/home`
  - `zfs create tank/home/lee    (fs)`
  - `zfs set mountpoint=/export/home/lee tank/home/lee`
    `no need to manage /etc/exports, etc.`
  - `zfs snapshot tank/home/lee@tuesday`
  - `Zfs clone tank/app1 tank/testing/fixapp1`

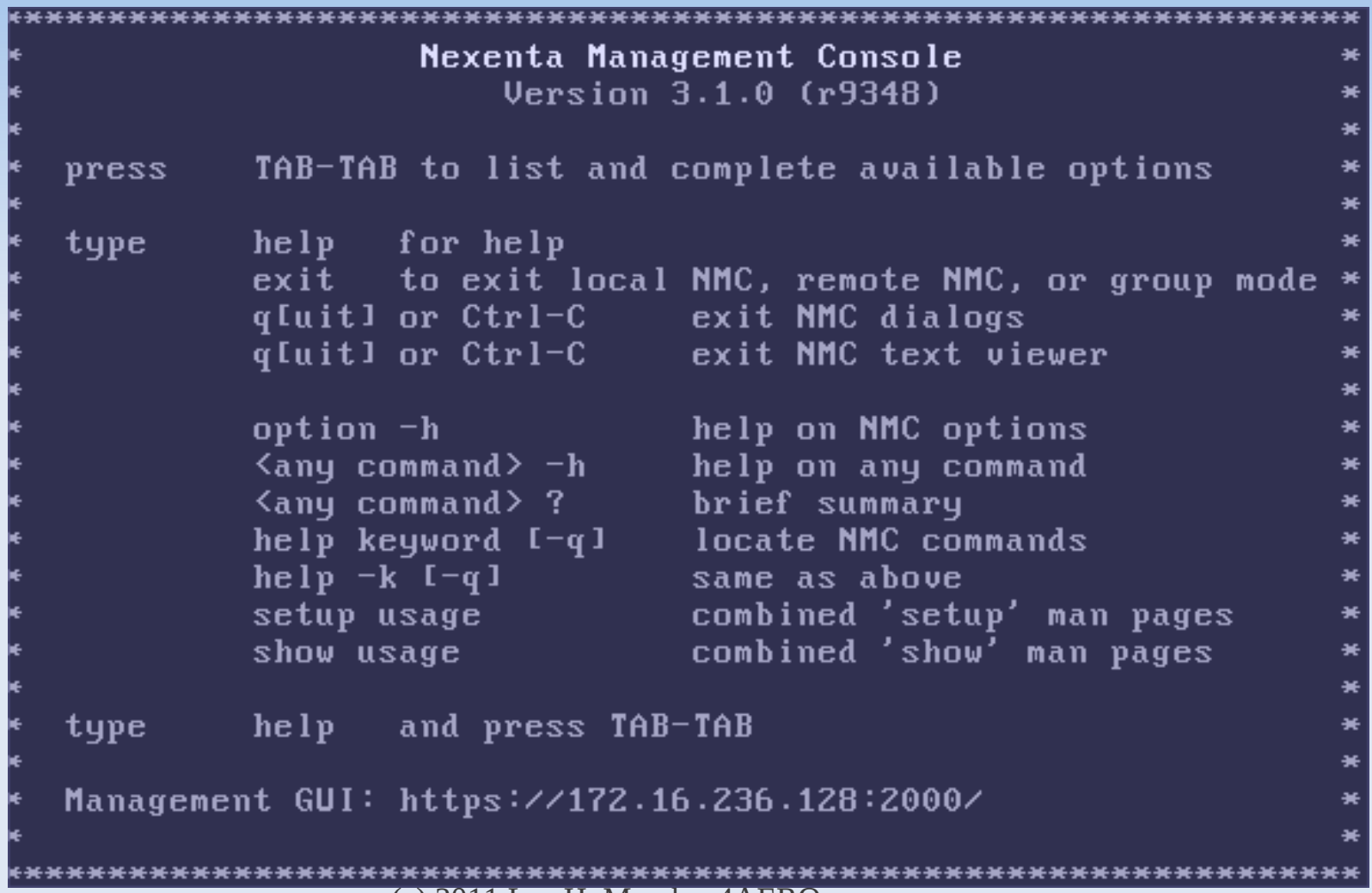
ZFS / OpenStorage

- Practical Implementations
  - OpenSolaris (discontinued by Oracle)
  - Nexenta OS (OpenSolaris kernel , Ubuntu userland)
  - NexentaStor
    - NexentaOS + Commerial GUI + Support
    - Free for first 18TB
    - Licensed by TB of storage thereafter
  - OpenIndiana
    - IllumOS Kernel - Fork of OpenSolaris
    - Many vendors migrating here, not in production

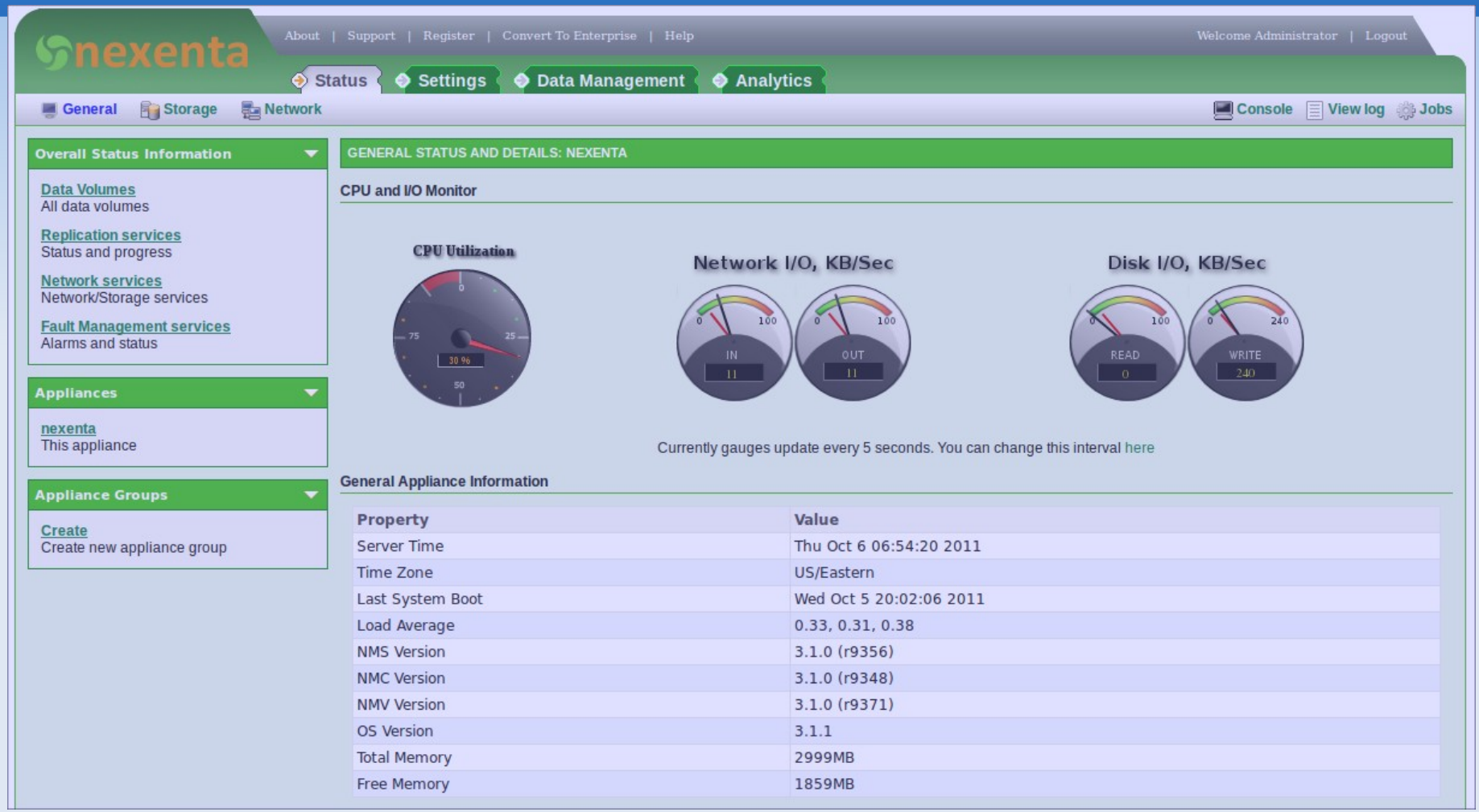
ZFS / OpenStorage

- Demo
  - NexentaStor (Free version for 18TB)
    - Running on OpenSource NexentaOS
      Unlimited use – CL only

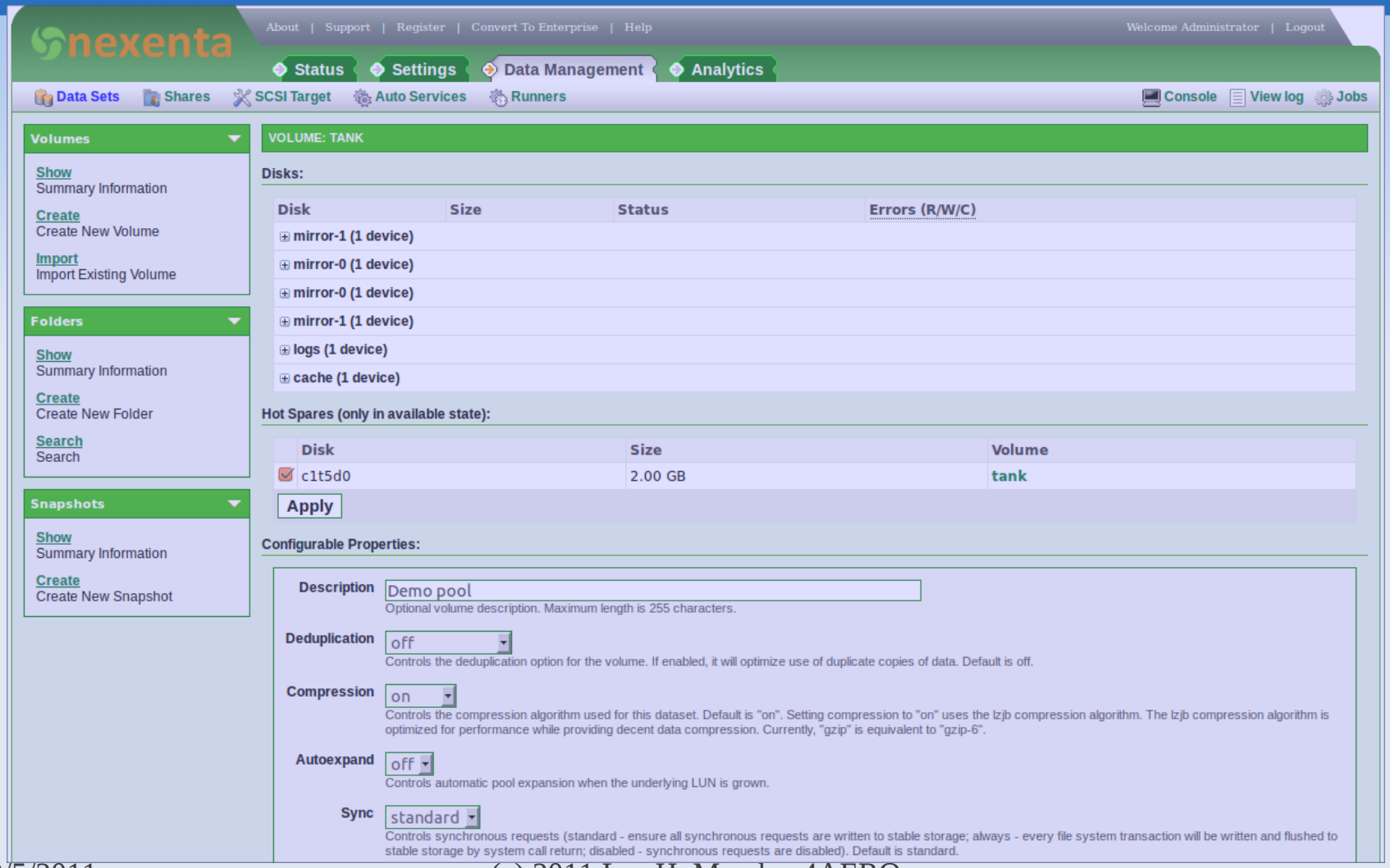
ZFS / OpenStorage



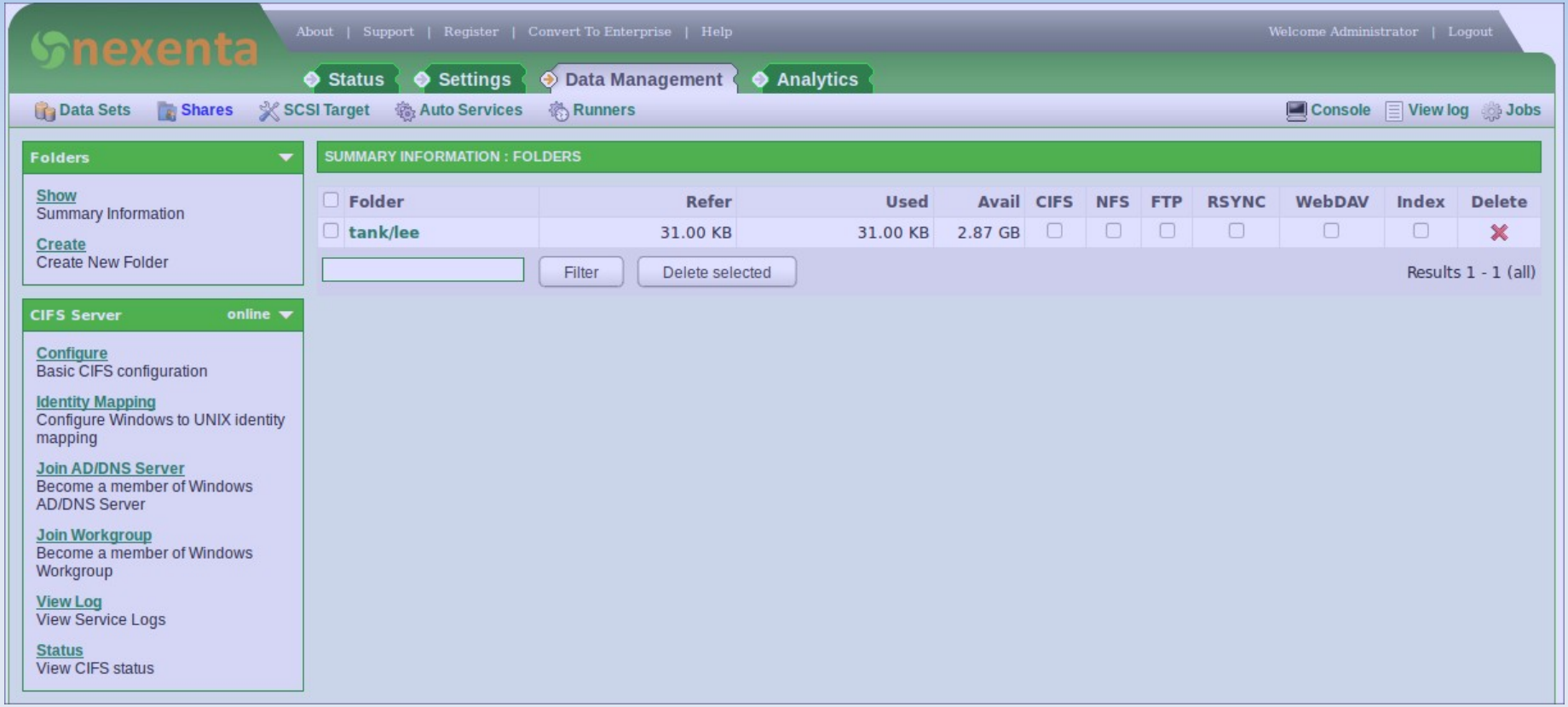
ZFS / OpenStorage



ZFS / OpenStorage



ZFS / OpenStorage



ZFS / OpenStorage

1. *http://www.phoronix.com/scan.php?page=article&item=zfs_fuse_performance*
2. *http://www.solarisinternals.com/wiki/index.php/ZFS_Best_Practices_Guide*
3. *http://zfs-on-fuse.blogspot.com/2007/04/zfs-in-linux-kernel.html*
4. *http://blogs.oracle.com/bill/entry/ditto_blocks_the_amazing_tape*
5. *http://hub.opensolaris.org/bin/view/Community+Group+zfs/basics*
6. *http://www.opensolaris.org/os/community/zfs/docs/zfs_last.pdf*
7. *http://blogs.oracle.com/roch/entry/when_to_and_not_to*
8. *http://constantin.glez.de/blog/2010/06/closer-look-zfs-vdevs-and-performance#vd*
9. *http://www.zfsbuild.com/2010/04/15/explanation-of-arc-and-l2arc/*
10. *http://www.zfsbuild.com/2010/07/30/testing-the-l2arc/*

ZFS / OpenStorage

11. http://blogs.oracle.com/openstorage/entry/fishworks_virtualbox_tutorial
12. http://blogs.oracle.com/bmc/entry/fishworks_now_it_can_be
13. http://blogs.oracle.com/bmc/resource/fishy-redacted.pdf
14. http://blogs.oracle.com/bonwick/entry/zfs_dedup
15. http://blogs.oracle.com/video/entry/zfs_the_last_word_in
16. ZFS Admin: http://download.oracle.com/docs/cd/E19082-01/817-2271/
17. http://www.informationweek.com/news/storage/systems/212001591