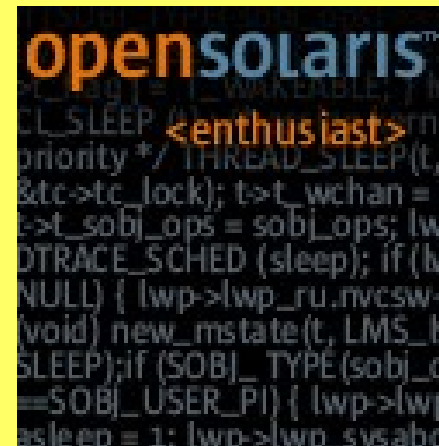


Zettabyte File System

Robert Miłkowski
System Group Manager
Wirtualna Polska



What is ZFS?

- Completely new 128-bit file system
- Integrated volume manager
- Best data integrity
- Simple management
- Great performance

Traditional file systems

- No protection for silent data corruption
 - Can't guarantee correct data
- Hard to manage
 - Partitions, slices, labels
 - Not flexible storage space management
 - Lots of limits
 - Platform dependent
- Too much complexity with FS+VM

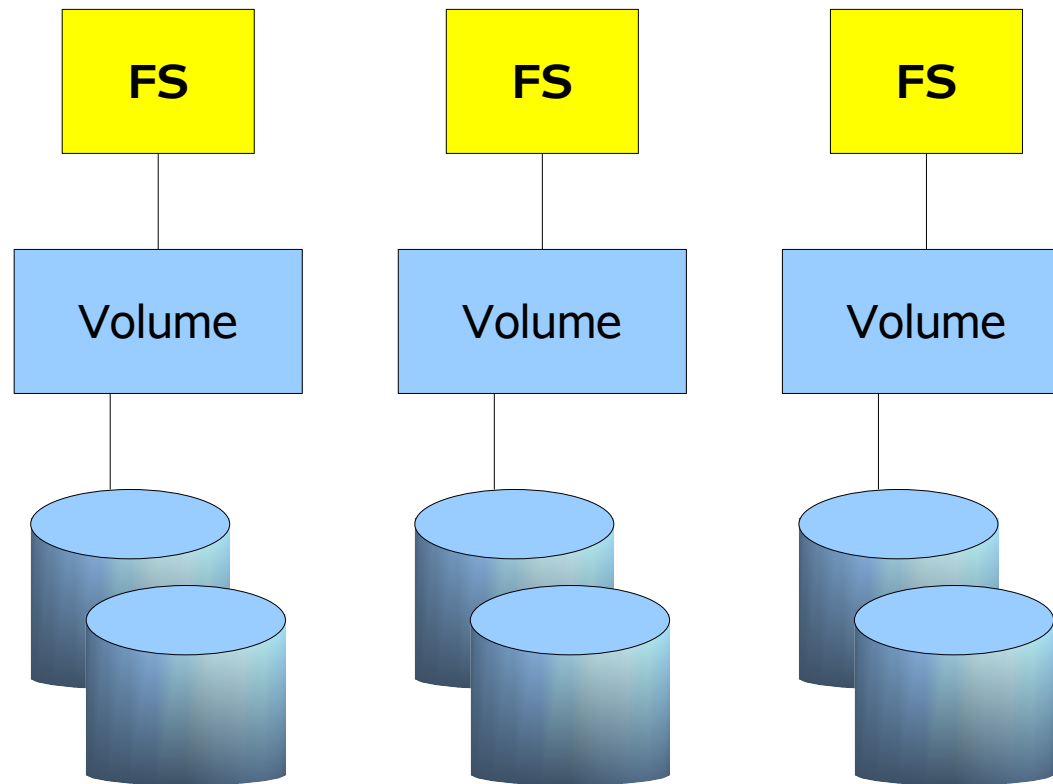
Wish list

- Make it SCALABLE
- Make it MANAGABLE
- Make it DYNAMIC
- Make it SAFE
- Make it EASIER

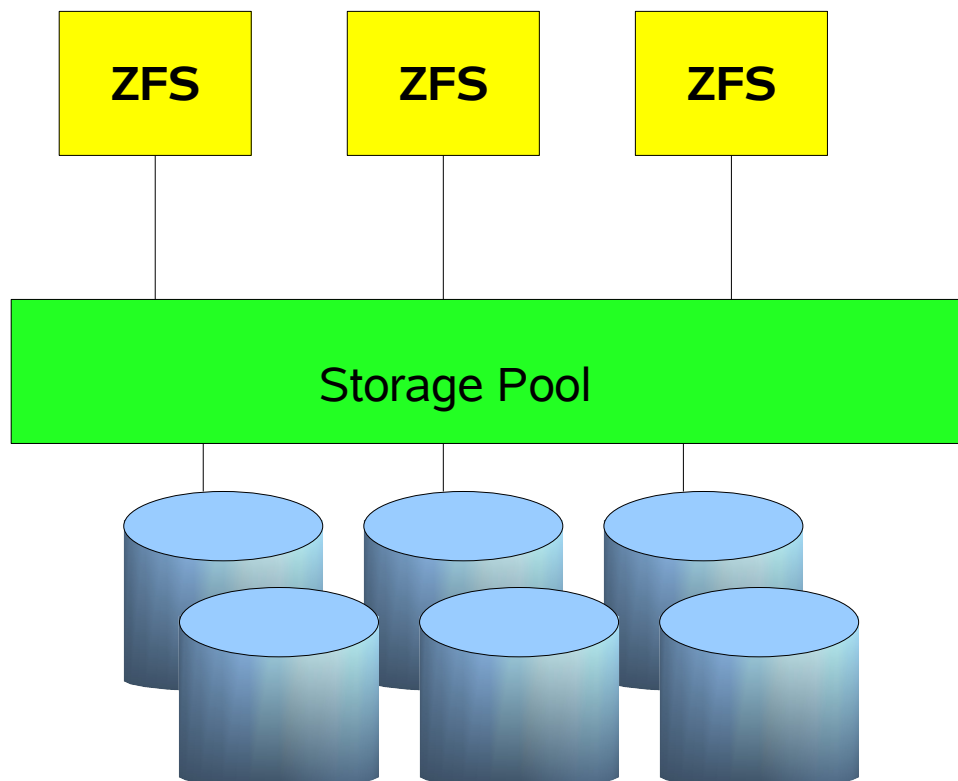
Let's do it from the scratch

- Pooled storage
 - No need for slices, partitions, volumes, etc.
 - Make as many file systems as you want for free
- End-to-end data integrity
 - Guarantee that application gets correct data
- Transactional
 - Always consistent on disk – meta and user data

Traditional FS+VM



ZFS pooled storage



ZFS pooled storage - example

```
# zpool create home c5t40d0 c5t40d1 c5t40d2 c5t40d3
# zfs create home/milek
# zfs create home/www
# zfs create home/milek/mail
# zfs set compression=on home/milek/mail
#
# zfs list
```

NAME	USED	AVAIL	REFER	MOUNTPOINT
home	72K	18.0T	10K	/home
home/milek	18.5K	18.0T	9.50K	/home/milek
home/milek/mail	9K	18.0T	9K	/home/milek/mail
home/www	9K	18.0T	9K	/home/www

```
#
```

ZFS Quotas & Reservations

- You can reserve space for a given fs
- You can set quota for a given fs
- quota/reservation for hierarchies
- No quotas for uid/gid

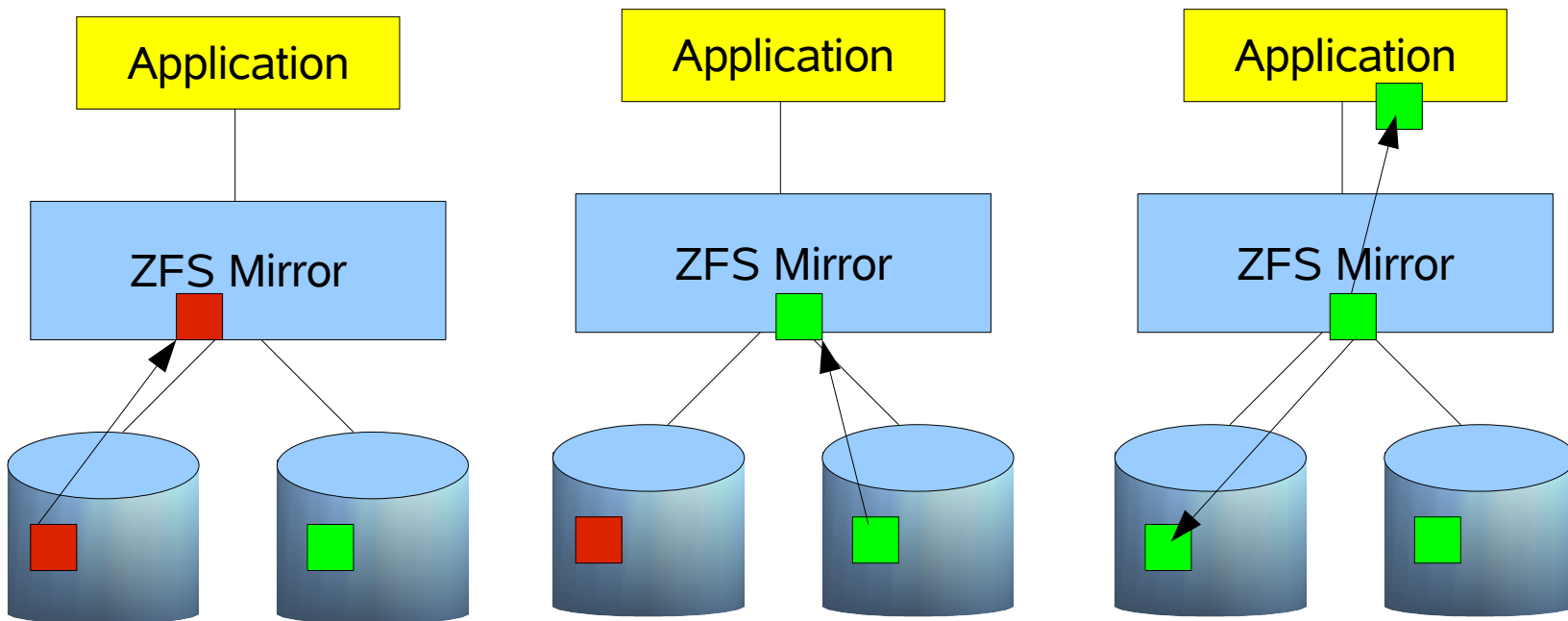
Data integrity

- Both data and meta-data are checksummed
 - No silent data corruption
- Everything is Copy-On-Write
 - Never overwrite live data
 - Always consistent on disk
 - No need for fsck-like utility

End-to-end data integrity

- Checksum checked after block is in memory
 - Whole IO path is checked
 - Corrects driver bugs, phantom writes, etc.
- Checksum and data block stored separately
 - Checksum is stored in parent block
 - Entire pool is self-validating
- Protects from accidental overwrites

ZFS Self Healing



Data integrity - example

```
# zpool create home mirror c5t40d0 c5t40d1
# cd /home/ ; ncftp sol-nv-b39-x86-dvd.iso

# digest -a md5 sol-nv-b39-x86-dvd.iso
a09a7af5ad25a980fa907b552a5bd523

# dd if=/dev/zero of=/dev/rdisk/c5t40d0s0 bs=1024k

# digest -a md5 sol-nv-b39-x86-dvd.iso
a09a7af5ad25a980fa907b552a5bd523
```

Ditto blocks

- Can't read data block – EIO
- Can't read meta-data block – part of fs is gone
- In most cases meta-data is less than 2%
- So replicate meta-data blocks
 - Additional to pool protection (mirror, raid-z)

Ditto blocks

- 3x meta-data blocks with pool information
- 2x meta-data blocks with fs information
- 1x user data blocks
 - Planned are 2x and 3x user data ditto blocks per fs
- Keep each copy on separate vdev
 - Spread block with only one drive
 - With user ditto blocks you get mirroring on single disk

RAID-5

- RAID-5 write hole – silent data corruption
 - Software workaround very slow
 - Hardware solution is expensive
- Partial-stripe writes – performance problem
 - Two synchronous reads before writes
 - Hardware write cache = expensive

RAID-Z

- Variable stripe width
- Always full stripe writes
- No RAID-5 write hole
- Meta-data based reconstruction
- Detects and corrects silent data corruption
- RAID-Z is **FAST** – no hardware needed

Disk resyncing

- Only actual data are resynced
 - If most of a pool is free this is big performance win
- During resync checksums are checked
- Offline/Online a disk
 - Only changed data resynced
- Top-down resilvering
 - Already resynced data are available
- Priority based resyncing - planned

Disk Scrubbing

- Corrects errors when still possible
- Checks all data blocks in a pool
- Automatically corrects bad data
- Works in a background with lower priority
- Checks **ONLY** actually used blocks

Snapshots

- As many snapshots as you want
 - Creating snapshot is an instant operation
 - Initially snapshots do not consume space
 - Easy rollbacks to a snapshot
 - Incremental backups
- Snapshots are persistent across reboots
- No need for dedicated space for snapshots
- Snapshots work with ZFS volumes

Clones

- As many clones as you want
 - Creating a clone is instantaneous
 - Initially clones do not consume space
- Creating a clone is instantaneous
 - Clone can be created only from a snapshot
- Clones work with ZFS volumes
- Great for development environments

Snapshots & Clones - example

```
# ls /home/  
sol-nv-b39-x86-dvd.iso  
  
# zfs snapshot home@monday  
  
# rm /home/sol-nv-b39-x86-dvd.iso  
# ls /home/.zfs/snapshot/monday/  
sol-nv-b39-x86-dvd.iso  
  
# zfs rollback home@monday  
# ls /home/  
sol-nv-b39-x86-dvd.iso  
  
# zfs clone home@monday home/home-new
```

ZFS Volumes

- You can create a volume
 - /dev/zvol/{dsk,rdsk}/path
- You can snapshot or clone a zvol

```
zfs create -V 10gb oracle/redo1
```

ZFS & Zones

- Shared storage space for many zones
 - Quotas and reservations are useful
 - Good to have separate fs for each zone
- Datasets admin delegation
- Cheap&Quick local zone cloning
- Zone snapshots – great for development

ZFS Performance

- Always Copy-On-Write
 - Random writes are sequential most of the time
- Variable stripe width
 - Increases throughput
 - Always full stripe writes
- Pipelined IO
 - Priorities, deadline scheduling, IO aggregation
- Multiple prefetch streams
 - Detects sequential access forward or backward

ZFS Data Migration

- Machine neutral on-disk format
 - You can move disks between SPARC/x86/x64
- You don't need to care about disks paths
- No need for `/etc/vfstab`

```
SPARC    #    zpool export oracle
x86-x64 #    zpool import oracle
```

Booting from ZFS?

- Not quite there yet
 - Phase 1 integrated in Nevada build 37
 - No SPARC yet
 - UFS still needed for bootstrapping
- Whole procedure is clumsy right now
- /var /opt /home – this works already perfect
 - Use legacy mountpoint

Live Upgrade on ZFS

- 1 – create a snapshot
- 2 – create a clone
- 3 – upgrade cloned system
- 4 – reboot to cloned system

- Only required space consumed
- No need for repartitioning
- Easy go-back if something goes wrong

Q&A

FSCK YOU!

{if you think ZFS is not production ready}

:)))))))))

Useful links

<http://opensolaris.org/os/community/zfs/>

http://blogs.sun.com/roller/page/bonwick?entry=zfs_end_to_end_data

http://blogs.sun.com/roller/page/bill?entry=ditto_blocks_the_amazing_tape

http://blogs.sun.com/roller/page/tabriz?entry=are_you_ready_to_rumble