

# Peta ... Exa ... Zeta - Looking at 18+ months of ZFS experience



Thomas Nau

Thomas.Nau@uni-ulm.de

kiz

University of Ulm



# Overview

- quick introduction
- ZFS basics
- the gory details
- lots of slides with examples
- ZFS versus UFS: a performance comparison
- summary of our experience



# Infrastructure Department

- 25 people in 2 teams running
  - cluster based university wide backup-, mail-, directory-, portal-, fileservices, databases and more
  - compute services for Universities in Ulm and Stuttgart
    - provided by SunFire 6800 and 4200 systems
    - additional funding granted to us about a week ago
  - LAN, MAN and WLAN installations
  - phone system including large VoIP installations
  - shop selling DVDs, paper, wireless phones, cables, ...



## ZFS Design Principals

- end-to-end data integrity; checksums everywhere
- close to zero-administration; self-adapting
- built-in volume manager
- almost no limits
- high performance
- platform-endianness independent
- 20+ years lifetime



## A Word About Lifetime

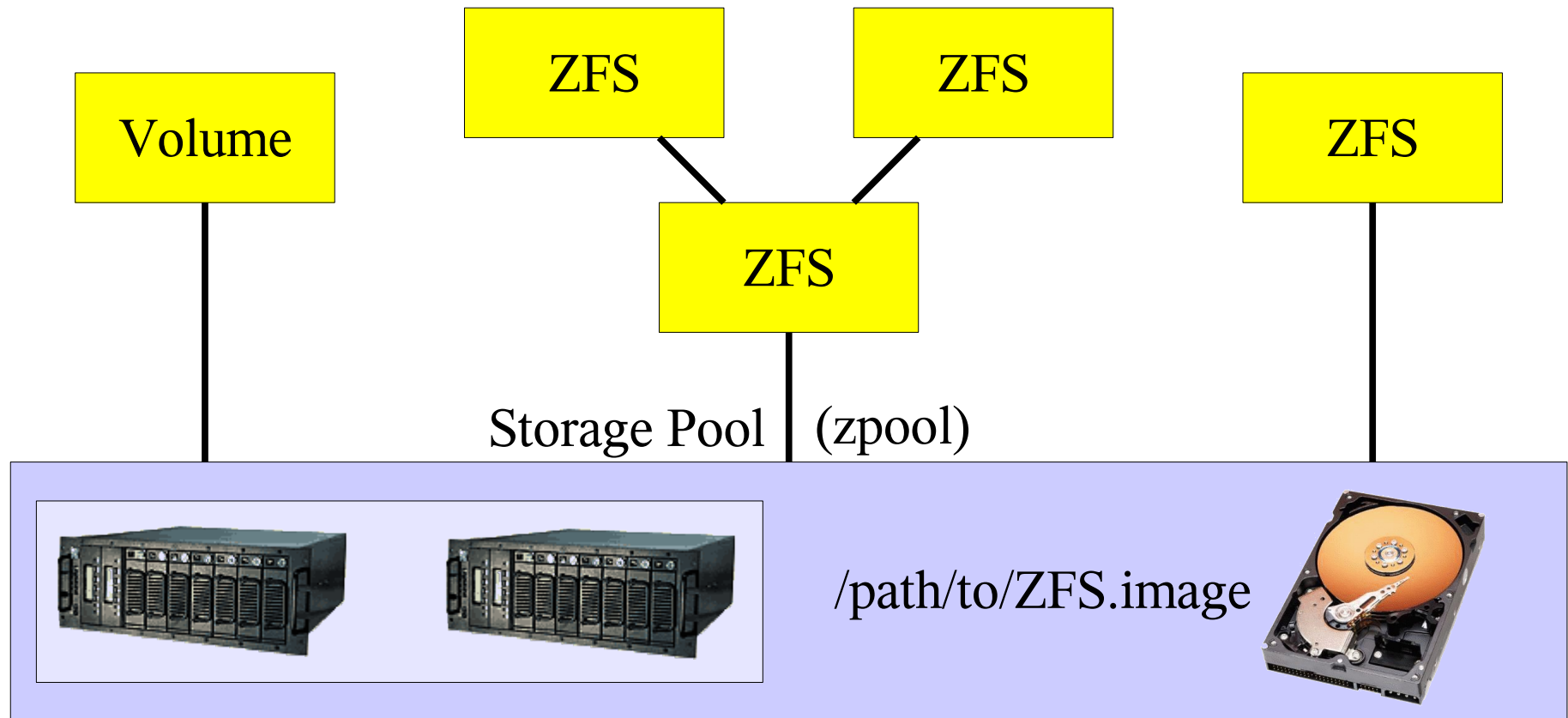
- UFS predecessors are 20 years old
- in the late 70s the famous PDP-11 used 5MB removable disks with a form factor of approximately 12" (30cm)
- today we are talking 500GB and pretty soon 1TB using 3.5" drives
- assuming a similar growth 64 bits might not be sufficient for the next 20 years; ZFS team decided to use 128bits



## ZFS Storage Pools (SPs)

- from the manual: “a storage pool is collection of devices providing physical storage and data replication for ZFS datasets”
- all datasets in a pool share the same storage
- virtual devices (vdevs) are the basic building blocks for storage pools
- data is striped over vdevs which cannot be nested
- Storage Pool itself is accessible as ZFS filesystem

# Storage Design





## Virtual Devices

- disks: either block devices, slices or partitions;  
**hint**: use EFI labels for endianness independence
- ordinary files residing on UFS, NFS or even ZFS filesystems
  - very handy for testing
- mirrors: a mirror of 2+ block devices, slices, partitions or even files
- RAID-Z: a variation of RAID-5 with 3+ block devices elimination “write-hole” problem, ...



# Virtual Devices

- vdevs in identical configuration (mirror, ...) can be added to the pool at any time
- ZFS will automatically make use of the newly available storage
- removing, not replacing, devices is on the wishlist
- build-in support for removable devices



## *zpool(1m)*

- single interface to administer storage pools
- sub-command examples
  - status and maintenance commands such as scrubbing
  - on- and off-lining devices
  - export- and importing of pools
- **caution:** the destroy sub-command will discard all data residing in the pool without asking for confirmation



# Example: Pool Creation

```
obi-wan# mkfile -v 128m vdev1 vdev2 vdev3 vdev4 vdev5
obi-wan# zpool create pool mirror /var/tmp/vdev1 /var/tmp/vdev2 \
        mirror /var/tmp/vdev3 /var/tmp/vdev4
```

```
obi-wan# zpool list
```

NAME	SIZE	USED	AVAIL	CAP	HEALTH	ALTROOT
pool	246M	164K	246M	0%	ONLINE	-

```
obi-wan# zpool status
```

```
pool: pool
state: ONLINE
scrub: none requested
config:
```

NAME	STATE	READ	WRITE	CKSUM
pool	ONLINE	0	0	0
mirror	ONLINE	0	0	0
/var/tmp/vdev1	ONLINE	0	0	0
/var/tmp/vdev2	ONLINE	0	0	0
mirror	ONLINE	0	0	0
/var/tmp/vdev3	ONLINE	0	0	0
/var/tmp/vdev4	ONLINE	0	0	0



# Example: Replacing Devices

```
obi-wan# zpool replace pool /var/tmp/vdev2 /var/tmp/vdev5
```

```
obi-wan# zpool status pool
```

```
pool: pool
state: ONLINE
status: One or more devices is currently being resilvered. The pool will
        continue to function, possibly in a degraded state.
action: Wait for the resilver to complete.
scrub: resilver in progress, 99.97% done, 0h0m to go
config:
```

NAME	STATE	READ	WRITE	CKSUM	
pool	ONLINE	0	0	0	
mirror	ONLINE	0	0	0	
/var/tmp/vdev1	ONLINE	0	0	0	
replacing	ONLINE	0	0	0	
/var/tmp/vdev2	ONLINE	0	0	0	
/var/tmp/vdev5	ONLINE	0	0	0	1.14M resilvered



## Example: Maintenance

```
obi-wan# zpool create pool mirror /var/tmp/vdev1 /var/tmp/vdev2
obi-wan# rsync -aH /opt /pool/
obi-wan# dd if=/dev/zero of=vdev2 oseek=268640 count=100000 >& /dev/null
```

```
obi-wan# zpool scrub pool
obi-wan# zpool status -v
```

```
pool: pool
state: ONLINE
status: One or more devices has experienced an unrecoverable error. An
attempt was made to correct the error.
```

...

```
scrub: scrub in progress, 17.99% done, 0h0m to go
config:
```

NAME	STATE	READ	WRITE	CKSUM
pool	ONLINE	0	0	0
mirror	ONLINE	0	0	0
/var/tmp/vdev1	ONLINE	0	0	0
/var/tmp/vdev2	ONLINE	0	0	80 519K repaired



## SAN Support Features

- ZFS by itself is endianness independent
  - you have to use EFI labels for the disks;  
ZFS will do this for you if you want it to
- data is always written in native byte order and swapped on demand while being read
  - no performance penalty as it is only required for metadata access
- export/import subcommands allow you to move ZFS internal information between systems



# Example: Endianness Independence

```
obi-wan# rsync -aH /opt /pool/  
obi-wan# zpool export pool  
obi-wan# scp vdev yedi:/var/tmp/
```

```
# NOTE: while 'obi-wan' is an AMD64 based Galaxy, 'yedi' is an  
#       UltraSPARC-T1 based T2000 with different endianness
```

```
yedi# zpool import -d /var/tmp pool  
yedi# zpool list
```

NAME	SIZE	USED	AVAIL	CAP	HEALTH	ALTROOT
pool	250M	154M	95.9M	61%	ONLINE	-

```
yedi# ls -l /pool/opt
```

drwxr-xr-x	3	root	bin	3 Feb 11 18:30	SUNWits
drwxr-xr-x	4	root	bin	5 Feb 11 18:35	SUNWmlib
drwxr-xr-x	11	root	sys	11 Feb 11 18:41	SUNWspro

# Peta ... Exa ... Zeta - Looking at 18+ months of ZFS experience



## The Gory Details



## *zfs(1m)*

- single interface to administer datasets within storage pools
- currently supported datasets
  - filesystem
    - zone support as dataset or filesystem
  - volume (block devices)
    - can be used as swap device, not as dump device
    - not available for local zones
  - snapshots and clones



# ZFS Properties

- *zfs(1m)* sets or gets properties of datasets such as
  - compression: on/off, algorithm and current ratio
  - quota and reservation
  - mountpoint and NFS export
- can be switched on/off as required
  - e.g. enable compression when filesystem fills up
- most writable ones are inheritable
  - exceptions: quota, reservation and volume size



# Example: Compression

```
obi-wan# zfs create pool/fs
obi-wan# zfs set compression=on pool/fs
obi-wan# rsync -aH /opt /pool/fs/
obi-wan# zfs get all pool/fs
```

NAME	PROPERTY	VALUE	SOURCE
pool/fs	type	filesystem	-
pool/fs	creation	Thu Feb 23 13:58 2006	-
pool/fs	used	66.8M	-
pool/fs	available	87.7G	-
pool/fs	referenced	66.8M	-
pool/fs	<b>compressratio</b>	<b>2.29x</b>	-
pool/fs	mounted	yes	-
pool/fs	quota	none	default
pool/fs	reservation	none	default
pool/fs	recordsize	128K	default
pool/fs	mountpoint	/pool/fs	default
...			



## Quotas and Reservations

- quotas are handled on a per filesystem basis not on per user basis
- we prefer group quotas anyway so we may use one filesystem per research group in the future (can also easily and fast be created on the fly)
- improves handling of batch jobs which require large amounts of scratch space
  - reservations guarantee that batch jobs get the disk space in a new ZFS they require or don't start at all



# Example: Quota And Reservations

```
obi-wan# zfs create pool/fs
```

```
obi-wan# zfs set quota=10g pool/fs
```

```
obi-wan# zfs set reservation=5g pool/fs
```

```
obi-wan# zfs list
```

NAME	USED	AVAIL	REFER	MOUNTPOINT
pool	5.00G	82.8G	99K	/pool
pool/fs	98.5K	10.0G	98.5K	/pool/fs

reservation →  
quota →



# Snapshots

- read-only copy of a filesystem or volume created in almost no-time
- consumes almost no initial space (copy on write)
- filesystem snapshots can be accessed independently through *.zfs/snapshot*
  - visibility of *.zfs* is controlled by property
- volume snapshots only support roll-back and cloning



## Example: Snapshots

```
obi-wan# zfs create pool/fs
obi-wan# rsync -aH /opt /pool/fs/
obi-wan# zfs snapshot pool/fs@`date +%d%m%y`
obi-wan# ls -l /pool/fs/.zfs/snapshot/
total 2
drwxr-xr-x  3 root      sys           3 Feb 26 10:53 260206
```

```
obi-wan# zfs list
```

NAME	USED	AVAIL	REFER	MOUNTPOINT
pool	154M	87.7G	99K	/pool
pool/fs	154M	87.7G	154M	/pool/fs
pool/fs@260206	0	-	154M	-

```
obi-wan# rm -rf /pool/fs/opt
obi-wan# rsync -aH /opt /pool/fs/
obi-wan# zfs list
```

NAME	USED	AVAIL	REFER	MOUNTPOINT
pool	309M	87.5G	99K	/pool
pool/fs	308M	87.5G	155M	/pool/fs
pool/fs@260206	154M	-	154M	-



# Clones

- writable copies of filesystems or volumes
- always tied to a snapshot
  - created in almost no-time
  - consumes almost no initial space (copy on write)
- cloning creates a dependency between the snapshot and the clone



## Example: Clones

```
obi-wan# zfs clone pool/fs@260206 pool/fsclone
```

```
obi-wan# zfs list
```

NAME	USED	AVAIL	REFER	MOUNTPOINT
pool	309M	87.5G	99.5K	/pool
pool/fs	308M	87.5G	155M	/pool/fs
pool/fs@260206	154M	-	154M	-
pool/fsclone	0	87.5G	154M	/pool/fsclone

```
obi-wan# zfs get all pool/fsclone
```

NAME	PROPERTY	VALUE	SOURCE
pool/fsclone	type	filesystem	-
pool/fsclone	creation	Fri Feb 26 12:55 2006	-
pool/fsclone	used	0	-
pool/fsclone	available	87.5G	-
pool/fsclone	referenced	154M	-
pool/fsclone	compressratio	1.00x	-
pool/fsclone	origin	pool/fs@260206	-
pool/fsclone	recordsize	128K	default
pool/fsclone	mountpoint	/pool/fsclone	default

...



## Snapshot and Clone Usage

- in combination with roll-back very useful for test installations
- to back up large filesystems as consistent as possible
- user accessible short term, e.g. daily, backups like *Plan9* does

# Peta ... Exa ... Zeta - Looking at 18+ months of ZFS experience



## Our Experience



## Test Restrictions


- we got a dominant hint at the very beginning:  
on-disk format may change any time
- this restricted ZFS use for the most time to
  - data which can easily be backed up and restored
  - “volatile” data
    - scratch areas for scientific applications
    - caches for backup system
    - web-caches



## Testbed

- v240 built-in disks for software evaluation and development
- total of 1.5TB scratch spaces located on striped T3 arrays attached to 6800er compute nodes
- 10TB 3<sup>rd</sup> party fibre channel RAID arrays attached to v440 and used as disk cache for Tivoli Storage Manager
- other systems for performance comparison: T2000, 4200, X1, ...

## Note

- all our experience and tests are based on beta releases up to the latest S10 update 2 beta
- syntax, performance and anything else may still change (even so I do not believe in that)
- functionality is still added while we talk and will first show up in  **openSolaris™**  
<enthusiast>
- ZFS is planned to be part of the upcoming 6/06 Solaris update



## ZFS versus SVM/UFS

- ZFS has proven to be more flexible and much easier to handle in many areas as for example
  - setting up a SVM mirrored volume plus creating and mounting 3 UFS filesystems takes 20-25 commands
  - only 4 with ZFS
  - increasing sizes with SVM/UFS is a pain
  - time comparison is an even bigger plus for ZFS



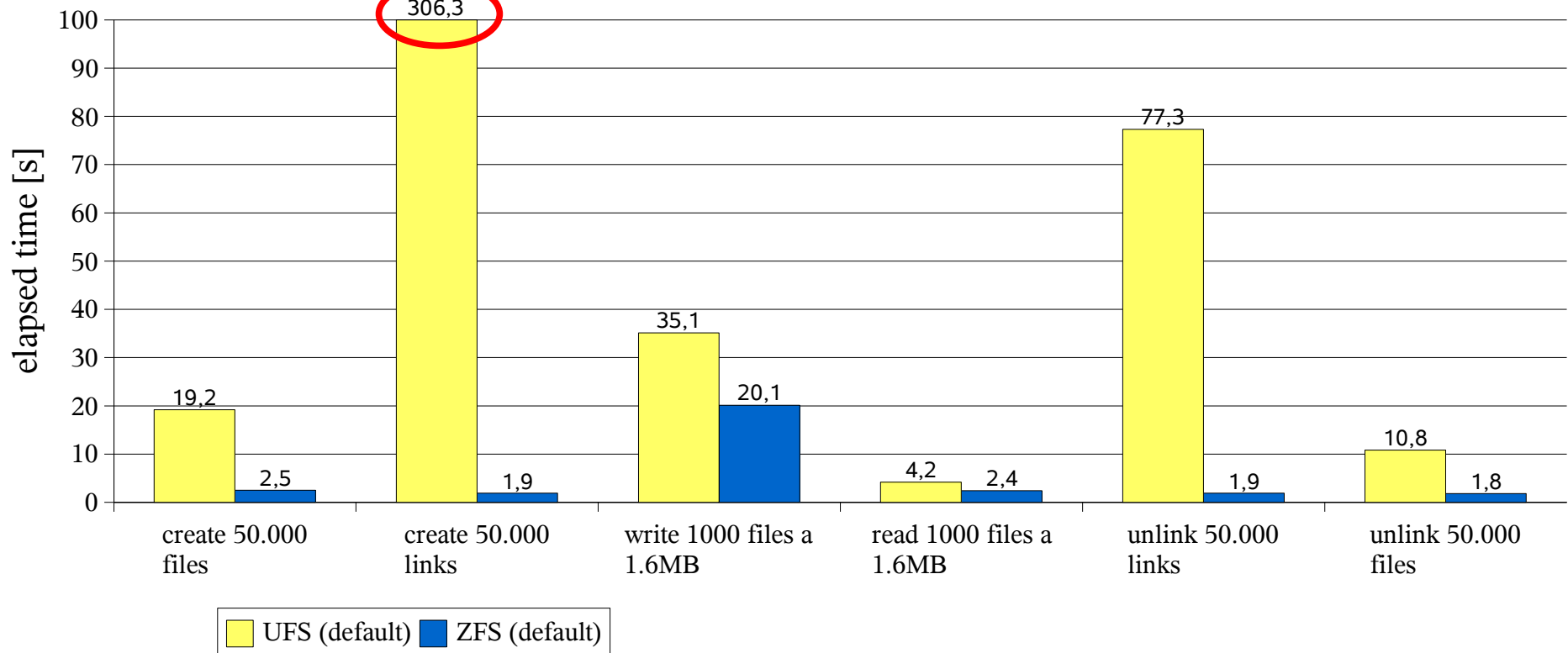
## ZFS versus SVM/UFS

- SVM re-sync algorithm doesn't deal well with large busy filesystems such as mail spool
- breaks 1TB barrier
  - multi terabyte UFS isn't usable at our site as it dramatically limits the number of inodes
- no need to think about disk space distribution or about partition setup
- **missing ZFS feature:** filesystems on SVM sub-mirrors are still mountable after detaching



## File Operations UFS versus ZFS

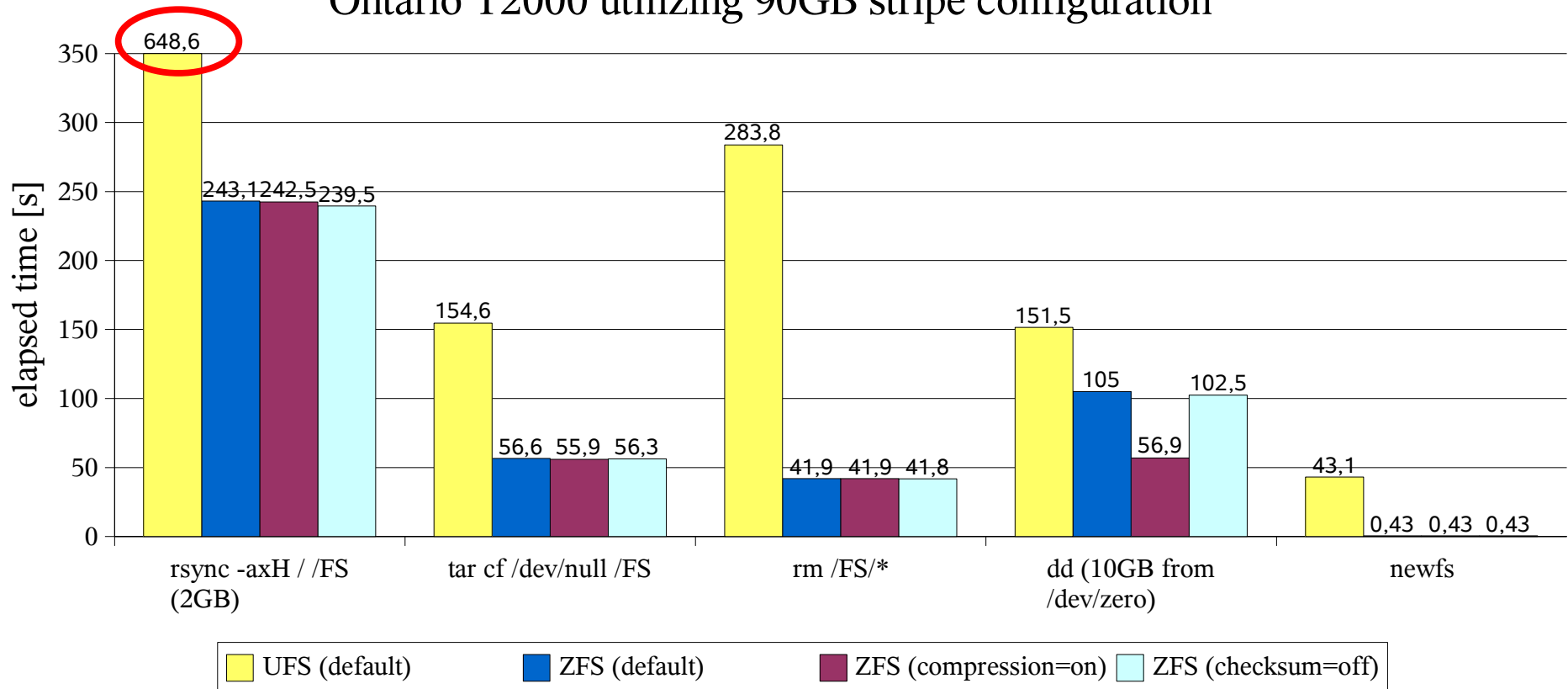
Galaxy 4200 utilizing 90GB stripe configuration





## Real World UFS versus ZFS

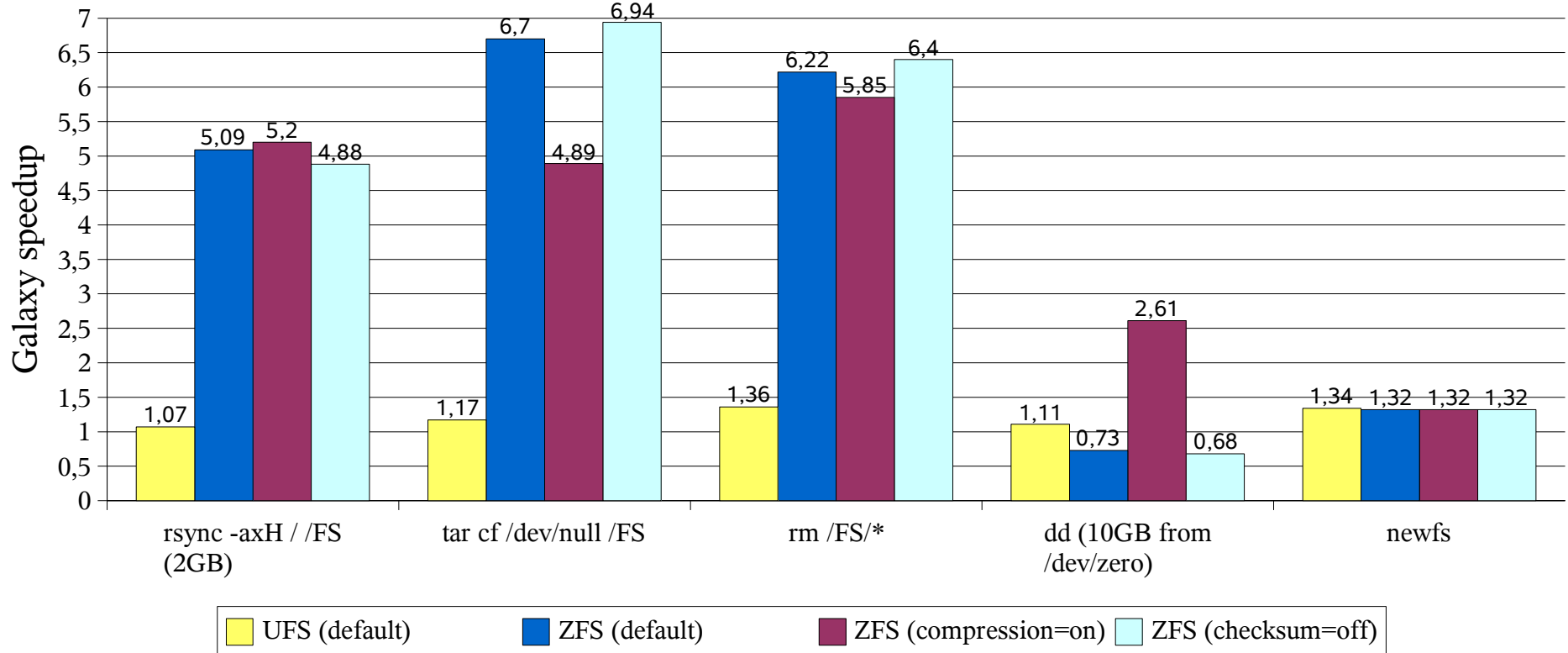
Ontario T2000 utilizing 90GB stripe configuration





# Real World UFS versus ZFS

Galaxy 4200 vs Ontario T2000





# Dealing With Data Corruption

- assuming non-redundant setup
- damage file data
  - file can be removed and restored from backup
- damaged file- or directory metadata
  - objects can only be moved;  
enhancements like hiding from namespace in work
- corrupted pool/filesystem metadata
  - sneak out and run



# Our Experience With Data Corruption

- had a single problem (didn't use redundant setup but stripes)
- power problems of FC-AL attached storage device caused data corruption
- ZFS alarmed us about whereas UFS might have run into a “silent data corruption”
  - you don't want to have this happen to your data



## Summary

- **important note: ZFS is not a cluster filesystem**
- for many months now, ZFS provided us with the flexibility we need to get our job done easier
- it excels in ease of administration
  - creation and size flexibility (pool concept)
  - data integrity; no penalty for checksums
- high performance especially for metadata operations and initial sync or rebuild of mirrors



## Summary

- open issues and questions
  - currently lacks native backup client support
  - are tools ready for 1000+ filesystems?
- administrators have to alter the way they deal with storage and filesystems

*Use The Force!*

