

UMBC

Office of Information Technology

AFS and Solaris 10 High(er) Availability using ZFS & Zones



Presented By:

Dale Ghent - daleg@umbc.edu

Coordinator, Storage and UNIX Systems - UMBC OIT

AFS & Kerberos Workshop 2007

May 7-11, 2007 - SLAC



First, a little background...

The UMBC.EDU cell:

- * 6 RW volume servers, 2 RO volume servers, 3 DB servers
- * 4TB total of mirrored vicesp space for hosting 47,000 volumes
- * Covers all home directories (incl. email and web), software depots, university websites, department file space

Hardware involved:

- * 6x Sun V20z and 2x Dell PE2650 servers running Solaris 10 (11/06 equiv.)
- * 2x Apple Xserve RAIDs, both with a LUN for each server. All servers mirror between the two LUNs to provide chassis-level redundancy. The PE2650s use direct-attached SCSI JBODs for RO.
- * Dual-port Qlogic FC HBAs on each server connected to separate FC switches to provide link- and switch-level redundancy (used with Solaris MPxIO multipath driver (aka scsi_vhci))

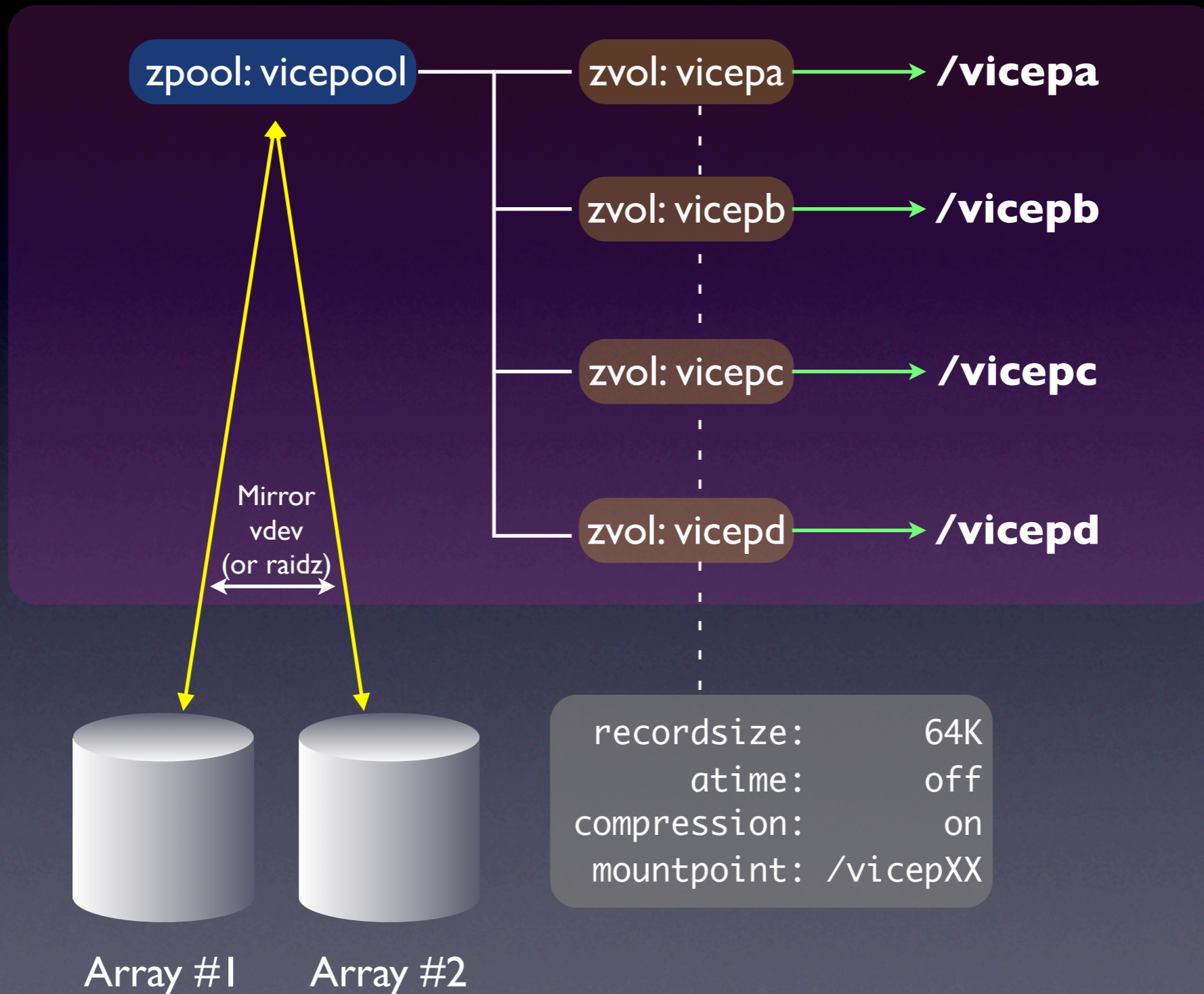
Part I

Serving AFS volumes from ZFS

Initial reasons for investigating ZFS for the vicep backing store

- * **Reliability:** Take advantage of the COW design of ZFS to enable salvageless starting of the AFS server after a crash.
(-DontSalvage)
- * **Flexibility:** Use ZFS's built-in volume and pool-based management to easily add space on the fly with little impact to server operations thus taking full advantage of our AFS server's SAN-based disks.
- * **Maintainability:** No need to fsck 556GB (in our case) of disk per server; extend existing disk space by using ZFS's built-in compression.
- * **Disaster Recovery:** Take advantage of ZFS's easy volume management (a la VxVM disk groups) to shuffle disks amongst servers. (DR with Zones)

Basic ZFS configuration for vice partitions



Basic ZFS configuration for vice partitions (cont.)

```
[daleg@hfs10]~$ zpool status
```

```
pool: hfs10
state: ONLINE
scrub: resilver completed with 0 errors on Tue Apr 17 03:35:54 2007
config:
```

NAME	STATE	READ	WRITE	CKSUM
hfs10	ONLINE	0	0	0
mirror	ONLINE	0	0	0
c4t600039300001043101000000D52AD7D9d0	ONLINE	0	0	0
c4t600039300001034A01000000D63843C6d0	ONLINE	0	0	0

```
[daleg@hfs10]~$ zfs list -r hfs10
```

NAME	USED	AVAIL	REFER	MOUNTPOINT
hfs10	298G	249G	24.5K	none
hfs10/vicepa	75.0G	61.8G	75.0G	/vicepa
hfs10/vicepb	72.3G	64.5G	72.3G	/vicepb
hfs10/vicepc	75.7G	61.1G	75.7G	/vicepc
hfs10/vicepd	75.0G	61.8G	75.0G	/vicepd

Basic ZFS configuration for vice partitions (cont.)

```
[daleg@hfs10]~$ zfs get all hfs10/vicepb
```

NAME	PROPERTY	VALUE	SOURCE
hfs10/vicepb	type	filesystem	-
hfs10/vicepb	creation	Wed Jan 10 14:21 2007	-
hfs10/vicepb	used	72.3G	-
hfs10/vicepb	available	64.5G	-
hfs10/vicepb	referenced	72.3G	-
hfs10/vicepb	compressratio	1.13x	-
hfs10/vicepb	mounted	yes	-
hfs10/vicepb	quota	137G	local
hfs10/vicepb	reservation	none	default
hfs10/vicepb	recordsize	64K	inherited from hfs10
hfs10/vicepb	mountpoint	/vicepb	local
hfs10/vicepb	sharenfs	off	default
hfs10/vicepb	checksum	on	default
hfs10/vicepb	compression	on	inherited from hfs10
hfs10/vicepb	atime	off	inherited from hfs10
hfs10/vicepb	devices	on	default
hfs10/vicepb	exec	on	default
hfs10/vicepb	setuid	on	default
hfs10/vicepb	readonly	off	default
hfs10/vicepb	zoned	off	default
hfs10/vicepb	snapdir	hidden	default
hfs10/vicepb	aclmode	groupmask	default
hfs10/vicepb	aclinherit	secure	default

Set each zvol's quota by dividing the total space of the pool by the number of configured vicep volumes, rounding down to the nearest whole.

This is important for performance reasons. AFS never writes more than 64KB at a time. Set the recordsize of the zvol to match this maximum.

The ZFS default of 128KB is far too large for this role.

All ZVOL settings should be made prior to putting data on them!

Setting AFS up for serving off of ZFS

! You must run a recent 1.4.x file and volserver in NAMEI mode!

* Patches:

- ▶ (Recommended) Rob Banz's "disable fsync()" patch:
<https://lists.openafs.org/pipermail/openafs-devel/2007-March/015030.html>

This patch *should* be in a future release, but not necessarily in the form you see here. ZFS does not appreciate tons of fsync() calls, and this patch raised our performance considerably with no found data integrity drawbacks.

- ▶ (Optional) Auto-attach to /vicepXX file systems of type ZFS:
<http://rt.central.org/rt/Ticket/Display.html?id=56531>

Otherwise, you'll need to touch AlwaysAttach in each /vicepXX. This patch will certainly be in the OpenAFS 1.4.5 or later release.

Setting AFS up for serving off of ZFS (cont.)

*Software Configuration:

✓ /usr/afs/local/BosConfig:

Add `-DontSalvage` to command line parameters for salvager:

```
parm /usr/afs/bin/salvager -tmpdir /usr/tmp -parallel all4 -DontSalvage
```

This will enable the salvageless startups. **Only do this when you have a FS that guarantees on-disk consistency!**

- ✓ If you did not apply the aforementioned ZFS/NAMEI patch, be sure to touch a zero-length file named `AlwaysAttach` in each ZFS-backed `/vicepXX` filesystem. `fileserv` will ignore them otherwise.
- ✓ Add a root cron job to run `zpool scrub <poolname>` once a month. It's reasonable ZFS housekeeping and can alert you to bad disks (in the form of checksum errors) in advance.
- ✓ For now, set a hard limit for ZFS's ARC (Adaptive Replacement Cache). Its code is a little overzealous at the moment and may cause AFS server processes to swap out if it eats up all `freemem`. Refer to the [Solaris Internals](#) wiki on how to configure this.

Observed behavior of ZFS-backed AFS servers

- * **Better overall throughput:** Across eight AFS servers, we have measured a detectable increase in IO rates due to ZFS write strategy and turning off observance of cache flush commands on our RAID arrays.
- * **So-so compression:** For vice partitions which hold only user home vols, we have seen compression ratios range from 1.1:1 to 1.3:1. While certainly not 2:1, this is a savings of 14GB - 41GB per vicep in our case. ZFS's lzjb compression is optimized for speed, not so much for space. gzip compression will be a future option.
- * **Stellar crash recovery:** In February 2007, we suffered two major power outages. During one, 80% of our AFS servers were downed ungracefully. Upon power restoration and a salvageless startup, we detected zero corrupted volumes on any of the servers.
- * **No fsck'ing around:** After the aforementioned outage, we also avoided a likely and lengthy UFS fsck at boot. This, coupled with salvageless start, decreased downtime by many hours.

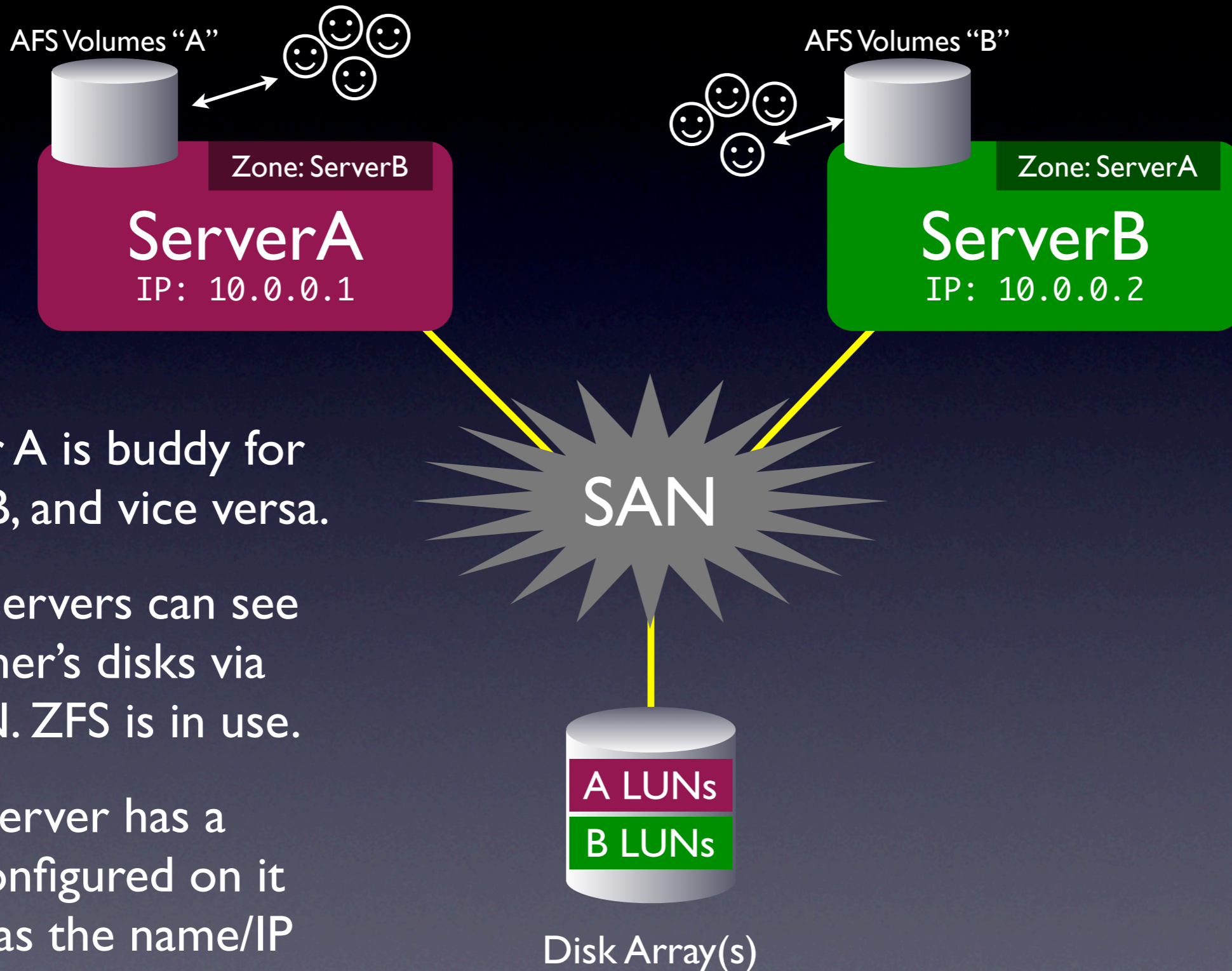
ZFS References and Resources

- * **Solaris Internals** - <http://www.solarisinternals.com/wiki/index.php?title=Category:ZFS>
- * **OpenSolaris ZFS Community** - <http://opensolaris.org/os/community/zfs/>
- * **Roch's Blog** - <http://blogs.sun.com/roch/> (and other blogs.sun.com blogs)
- * The `zpool` and `zfs` man pages
- * From the DTrace Toolkit, use the `iopattern` and `bitesize.d` scripts to gather filesystem read/write size statistics.

Part 2

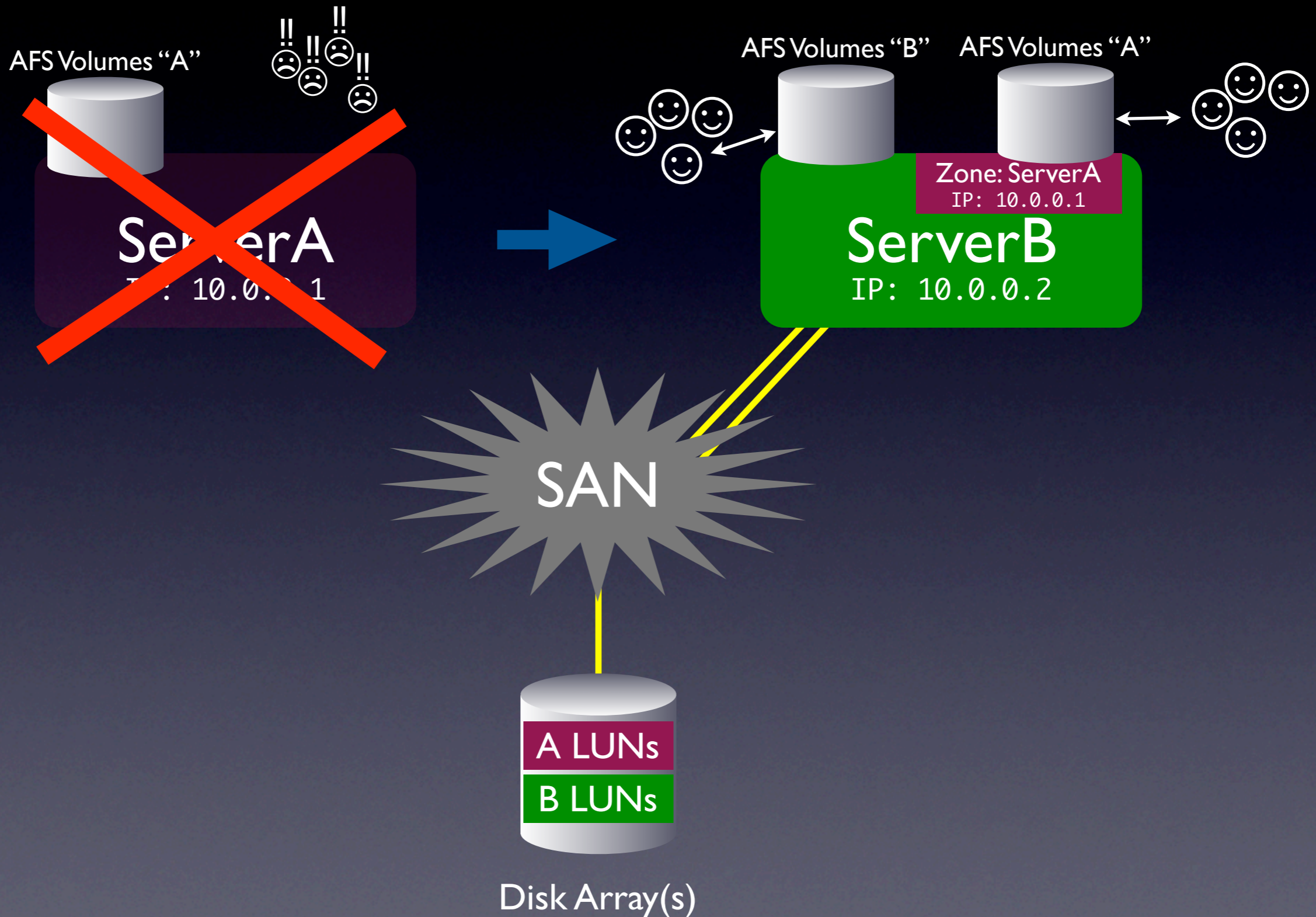
Using Zones with ZFS
for faster outage
recovery

Creating a “Buddy System”



- Server A is buddy for Server B, and vice versa.
- Both servers can see each other's disks via the SAN. ZFS is in use.
- Each server has a Zone configured on it which has the name/IP of the other.

Let's say ServerA commits seppuku. What should the result look like?



What just happened there?

- 1) ServerA died (hardware failure, kernel panic, whatever) and it is realized that ServerA will be down for a while until it's sorted. The decision is made to bring up the standby Zone for ServerA on ServerB, and ServerB will act as a surrogate for ServerA for a while to keep its AFS volumes live.
- 2) The SA springs into action on ServerB and issues a `zpool import` command with the force (`-f`) and altroot (`-R <path>`) arguments. This brings ServerA's disks into ServerB from the SAN and mounts the `/vicepXX` volumes within the root directory of ServerA's Zone.
- 3) The SA then boots the ServerA Zone with the `zoneadm` command. It takes a matter of seconds (if that) for a Zone to boot. Solaris creates a "zoned" IP interface with ServerA's IP address as this happens.
- 4) The SA may then `zlogin` into the ServerA Zone and start the AFS `fileserver/volserver` daemons, per normal procedures (that is, if they are not started automatically on Zone boot)

What does a Zone need in order to be an AFS server?

▶ Nothing special. It can be a “Sparse Zone” or a “Whole Root Zone”, but there are two important things to keep in mind when doing this:

1) The Zone’s network configuration needs to have the same IP address as the server it is surrogate for. This is a requirement of AFS, as the VLDB uses an AFS server’s IP address to keep track of what volumes are served by it.

2) If the server that hosts a Zone for another is also a AFS server, AND that AFS server runs in the Global Zone, be sure to put the IP address of the surrogate Zone in the Global Zone’s NetRestrict file. This is so the AFS server running in the Global Zone doesn’t decide to advertise the IP address of the surrogate Zone for its own volumes. That would be Really Bad.

The Future...

Right now, this “HA” process is completely manual. A human must detect a fault, and then manually start the failover process to the surrogate server. Methods are being studied to automatically detect certain failure states and appropriate parameters of them for both semi-automatic and (in some cases) fully automatic failovers which are initiated by a monitoring daemon. As these methods are tested and deliberated on, they’ll find their way into a public software release.

Zones References and Resources

- * **Solaris Blueprints for Containers** - <http://www.sun.com/blueprints/0506/819-6186.pdf>
- * **OpenSolaris Zones Community** - <http://opensolaris.org/os/community/zones/>
- * **Jeff Victor's Blog** - <http://blogs.sun.com/JeffV/>
- * **The zoneadm and zonecfg man pages**