


Slide 1



What has ZFS ever done for us?

ZFS boot in the Mission Critical Environment

Philip Scarlett
<http://www.linkedin.com/in/philipscarlett>

I'm Philip Scarlett and have been working with Solaris in mission critical environments on and off for about 15 years.

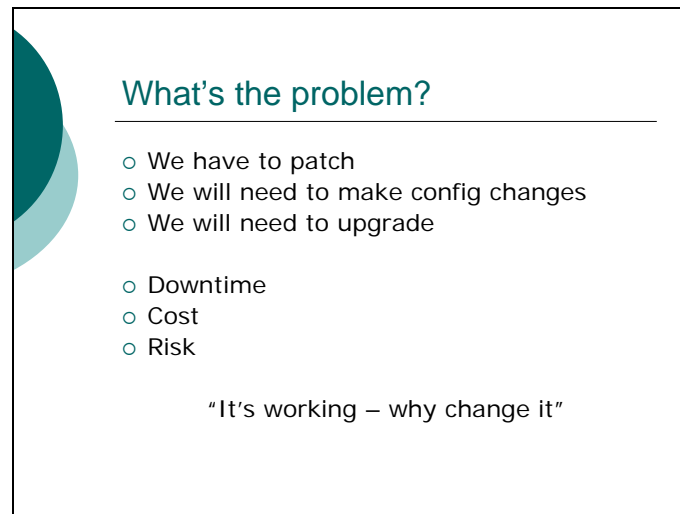
No matter what job titles I've had I still consider myself to be a Sys Admin. The main reason for this is that I'm still keen to automate and reduce duplication of effort wherever I can. And I'm sure that ZFS is going to make my life a whole lot easier. In fact it has already has!

Part of the glorious life of a Sys Admin is to interface technology with the business, and this is what I hope to achieve with my presentation tonight.

I've already used ZFS for my shared data volumes in a recent Sun Cluster build and it really did solve many problems we were facing with Solaris Volume Manager.

My next goal is get ZFS boot adopted as a standard so we can transform the management of Solaris boot in mission critical environments.

Slide 2



What's the problem?

- We have to patch
- We will need to make config changes
- We will need to upgrade

- Downtime
- Cost
- Risk

"It's working – why change it"

(XXX)

So what's the problem with our current Solaris Volume Manager or Veritas Volume Manager boot environments?

Is there a problem?

After all they've been good enough for the past decade - haven't they?

Well - probably not. It's just been what was available to us and as Sys Admins we came up with some very clever ways to work with them.

So let's consider the most common planned changes to systems. If we were to consider the unplanned ones we'd be here all night!!

(XXX)

Patching

Nothing new here, we all patch twice a year don't we? Possibly more often if we have to apply regular security patches.

Configuration changes

These range from small, single file changes to larger, say Cluster, reconfigurations.

Upgrades

Our developers and application teams may want Java updates, Perl updates etc.

In our mission critical environments all these changes will be subject to a defined change window and will require proven back out plans. This is where it starts to get a little bit more complicated than our home or test systems.

(XXX)

So what restraints are we working under to achieve these tasks?

In mission critical environments the business will always want minimum downtime to their systems. When trading systems aren't trading they are usually running reconciliation tasks or batch jobs.

Change windows are small and most likely on pre-planned weekends throughout the year.

This means we may not even have the luxury of preparing our systems for the changes as we can only touch systems during the change window. This severely hampers existing tools such as Live Upgrade or scripted solutions that may create a 3rd mirror.

Cost is always a concern. Most changes are out-of-hours which costs money, not just in overtime but also possibly over-running changes that may delay the service coming back online.

And Risk!! This is the big one. In most mission critical environments there will be some form of a change management process that will scrutinize everything we do as Sys Admins. This process will require cast iron guarantees and back out plans for all eventualities.

ZFS boot can reduce all of the above. Downtime, Cost and Risk.

But how do we convince the less than technical people around us? Usually the business groups or the service owners.

(XXX)

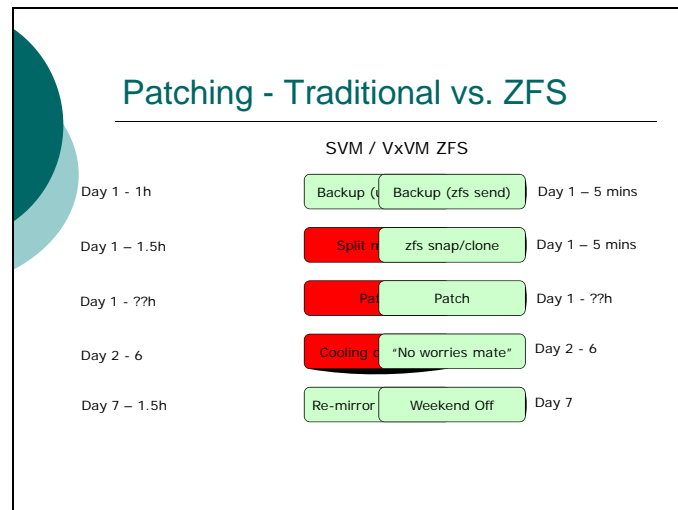
Their usual response is – “It’s working – why change it”

All of us here probably appreciate how wrong this statement is. However when you are trying to convince a change board or a business group in a major financial institution, it’s a common response.

Believe me, I’ve been there.

So when we talk about how ZFS boot can reduce Downtime, Cost and Risk we can represent it all in a very basic process for Patching. Patching is one of those relentless Sys Admin tasks that consumes an amazing amount of time and money – especially when you scale it to a thousand servers or more.

Slide 3



(XXX)

So let's look at a typical patch process for our traditional volume management products, VxVM or SVM. Remember that preparation isn't always possible as we often can't touch systems until the change window begins. This may give us only a few hours over a weekend in which we have to complete the whole change.

I've tried to convey an elapsed time down the side with the number of hours for each task. Please remember this is just an example.

So, all Sys Admins should insist on a consistent, restorable backup. Most daily enterprise backups will be incremental and may be split over several tapes, and may even be offsite. So we look to ufsdump – it's been a standard for many years and we know it will give us a reliable backup.

Now we come to splitting our root mirrors. There are so many methods for this depending upon which Volume Manager you are using. They all take time and effort and introduce risk, even if scripted, maybe especially if scripted, but it will give us a more-or-less instant fail-back

I've allowed 1½ hours for this which sounds a lot but it's no good splitting our root mirrors if we don't successfully test boot both sides.

Then we move onto patching. There are so many ways to do this – manual patching, smpatch, OpsWare, xVM, SunConnection.

This is where you will be glad that you took a backup and split your mirrors.

It's been rumoured that some patches can break things, or more than likely not reading the patch READMEs will help YOU break things.

In short this is a risky business so you need to be covered.

Assuming all went well, 2 hours or so later you should have a patched system, all rebooted and ready to go.

Now, our application teams, the business and our risk managers will usually insist on a cooling off period after a major change such as patching. They will want a guaranteed, rapid fail-back method for a period of time if the patching proves problematic. So we leave our systems in an un-mirrored state.

Then, probably on the next weekend, when all has been proven OK we get the go ahead from the business and the change board to re-mirror and tidy up. We will be allowed a small change window do this work.

Approximately an hour and a half for this work especially as disks have to re-sync.

You've probably all guessed that the red shows us when we are running our systems at risk. Production running on a single disk for a whole week! Best be checking out the MTBF figures of those disks. It does really happen out there!!

Please note that this is just a basic example. In reality there are many different ways of doing this but whatever the way it will take time, cost money and introduce risk.

So let's compare this traditional approach to patching to how we would perform the same tasks with a ZFS boot environment. Here we will see we not only reduce downtime, but reduce cost and reduce RISK.

(XXX)

So instead of a ufsdump we perform a ZFS send/receive of our latest incremental snapshot. Here we really start to see how ZFS will transform our boot environments. If you were here in July, Sally Houghton gave a presentation on a ZFS OpenBackup solution that she's now successfully implemented. This means that we only ever have to take a ZFS snapshot and send it to our ZFS backup server to have a consistent, full, restorable backup.

Time for this - I would say 5 mins is playing it really safe. I would expect sub minute backup time for a root filesystem.

Now instead of slitting mirrors we perform a ZFS snapshot and clone. Here we really reap the benefits of ZFS. A ZFS snapshot and clone will give us a complete boot environment we can revert to if our patching doesn't go to plan.

Once again 5 mins is really pessimistic and we would probably script it as part of our pre-patch process and it will be 5 seconds!


So we are 10 minutes into our change and we have a full root system backup and a recoverable boot environment on the **still mirrored boot disks**.

Patching more or less remains the same. But what happens if I lose a disk while patching? For the traditional method, depending on which disk fails, either my patching will fail, and the system will most likely panic, or my fallback boot environment will be no more. With ZFS all we'll have to do is replace a disk.

Now my next comment sounds a bit flippant but you're not going to have to explain to the business what went wrong if you lose your single boot disk during that week!!

Do we really get a day off? No, probably not, as Sys Admins we don't have a social life so we will be using all that spare time to update our firmware or patching more servers.

So how do we communicate this wonderful news to the business and get their buy-in? Money obviously.



The Money Slide

Savings of ZFS over traditional patching

- 250 servers
- Patch twice a year
- Saving of 4 hours per server
- £50/hour

Total = $250 * 2 * 4 * 50 = \text{£}100,000$

(XXX)

This is a very simple but effective slide that I've used on management to get my way and move to ZFS boot. Unfortunately part of a Sys Admin's job is to interface the technology with reality. As much as we'd love to promote all good technical advancements in Solaris sometimes it just comes down to money. The good news is that ZFS boot is an easy sell.

(XXX) Let's assume we have 250 servers to patch. This represents a small environment in terms of some Solaris estates out there.

(XXX) Well we all say, or like to believe, we patch twice a year.

(XXX) And from the previous slide a saving of around 4 hours per server could be achieved with ZFS boot.

(XXX) Another large assumption is the paper cost of Sys Admin time – please don't quote this as it's for accounting purposes only. In truth it's probably a very conservative figure.

(XXX) So by some simple calculations we can save £100,000 a year – and that's in patching time alone. If we added in all the unplanned outages which we can also protect against using ZFS boot this figure could be a lot higher.

Oh! And don't forget the cost savings for all of those Veritas licenses.

Slide 5



(XXX)

Now in no way am I advocating this is money that should be taken away from your department's budget. It's money that might be spent on other worthwhile tasks such as firmware patching or sorting out your other issues such as.....

(XXX)

We all have "housekeeping" tasks that never seem to get done. Hopefully ZFS Boot will help us focus on other areas that require our attention.

So what is this ZFS magic we're talking about? The previous slides are really the management pitch, not much for Sys Admins.

The reality is there isn't that much for the Sys Admin to do with ZFS boot. It's all been built into ZFS.

So let's run through a real example that will give us an effective full backup and a quick and easy failback Boot Environment.

Slide 6

```
ZFS pre-change process

# zfs snapshot rpool/ROOT/std_s10u8@CR54321
# zfs send <snapshot> | ssh <backup_server> zfs recv <backup_pool>/<target>
# zfs clone rpool/ROOT/std_s10u8@CR54321 rpool/ROOT/std_s10u8_CR54321
# zfs set canmount=noauto rpool/ROOT/std_s10u8_CR54321
# zfs set mountpoint=/ rpool/ROOT/std_s10u8_CR54321

# echo "
#----- Pre-Patch CR54321 -----
#
title Solaris 10 - Pre-Patch CR54321
bootfs rpool/ROOT/std_s10u8_CR54321
kernel$ /platform/i86pc/multiboot -B $ZFS-BOOTFS
module /platform/i86pc/boot_archive
#
#----- Pre-Patch CR54321 Failsafe -----
#
title Solaris 10 - Pre-Patch CR54321 - Failsafe
bootfs rpool/ROOT/std_s10u8_CR54321
kernel /boot/multiboot -s -B console=ttya
module /boot/amd64/x86.miniroot/safe
#
#----- END Pre-Patch CR54321 -----
" >> /rpool/boot/grub/menu.lst
```

(XXX)

I hope you can all see this.....

- We take our snapshot giving it a real, sensible name. Here I've used a change record number.
- We send it to our ZFS OpenBackup Server which by magic will give us a full backup off host. The actual syntax will probably be different to this as we'll usually be sending just incremental changes but it wouldn't all fit on the slide 😊
- Now we clone our snapshot and make a couple of property changes then update our GRUB menu adding an entry for our cloned Boot Environment as well as it's failsafe environment. Remember some patches will update our failsafe environments so it's important we have a failback for this as well.

This is all I need to do prior to any change. Pretty easy to script as part of our process. This process is so lightweight in space and time that there's no excuse not to do it.

So, say we make some radical system changes – and it fails miserably. System won't boot. How do we dig ourselves out of this hole?

Slide 7

ZFS fail-back process

- Boot failsafe (either original or cloned version)
- Don't elect to mount any discovered root filesystem

```
# zpool import -f rpool
# zfs promote rpool/ROOT/std_s10u8_CR54321
# zfs destroy -r rpool/ROOT/std_s10u8
# zfs rename rpool/ROOT/std_s10u8_CR54321 rpool/ROOT/std_s10u8
# reboot
```

- Once system is up, update the `/rpool/boot/grub/menu.lst` appropriately

Note: - This maintains any snapshots prior to cloning

(XXX)

The first step is to boot in failsafe mode. Just pick the one that works. No DVDs or Boot Net required. Believe me this is a real bonus at 3am in the morning.

Don't let Failsafe mount anything when it boots.

Then we can follow these five simple commands and we are right back where we started. OK – we may have lost our changes but we are back in a production ready state in 10-15mins. No restores required or changes to manually back out. More importantly anyone can perform the backout plan – we don't have to have prior knowledge of what occurred during the change.

If we fail-back to our renamed clone we can even be assured that our snapshots are still present and in sync with our ZFS Open Backup solution.

To back out minor changes we could use ZFS rollback of the associated snapshot. However with ZFS rollback I'm not sure how stable the system will be after major changes such as kernel patches or system libraries have been changed – This is one to look into and test.

So in addition to major system changes we now have a mechanism where every time we log in as a privileged user we can reduce risk.

Consider the ever so common admin mistake, `chmod 777` in the wrong directory.

If we can incorporate a zfs snapshot and clone routine into our patch tools, our login mechanisms and any other process that may put a system at risk then we may just save ourselves a whole heap of trouble.

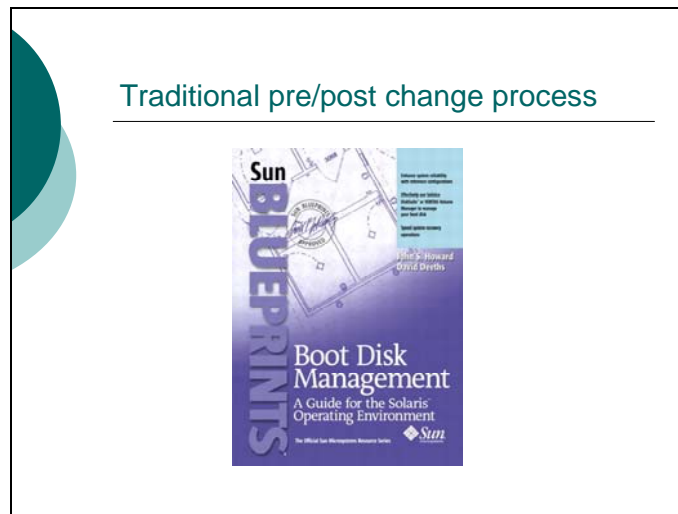
You may ask why I'm not using beadm or Live Update. I'm not knocking these tools – it's just a Sys Admin thing to give a common solution across OpenSolaris and Solaris that can easily be incorporated into our existing processes.

Whilst this process won't stop human error it gives you that instant recovery point. Our scripts could even send a syslog message or e-mail when a login snapshot is taken so everyone knows where to recover to.

Now if we would like this functionality in our traditional SVM or VxVM environments we may consider.....

(click)

Slide 7



All 180 pages of it.

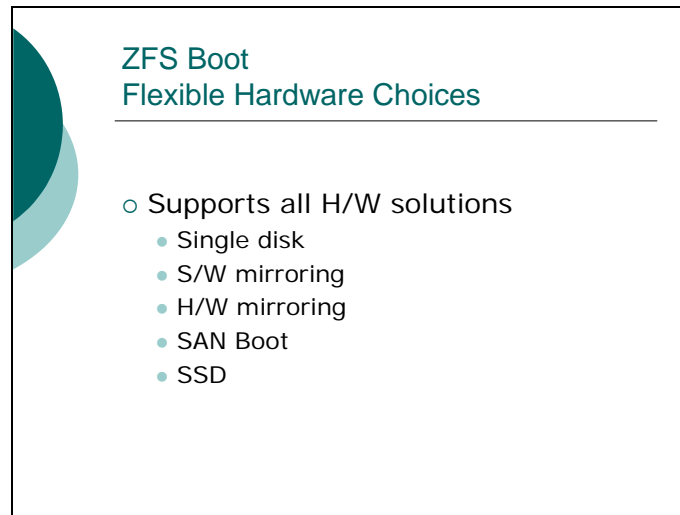
This is an excellent book and I would highly recommend it to anyone still managing SVM or VxVM.

I've used this over the years as a reference and as a foundation for building Jumpstart and scripted solutions.

If you've worked with or delivered solutions similar to those detailed in this book then you will really appreciate ZFS boot.

So now we have a low maintenance, low cost, low risk boot environment -- what else can ZFS do for us?

Slide 8



ZFS Boot
Flexible Hardware Choices

- Supports all H/W solutions
 - Single disk
 - S/W mirroring
 - H/W mirroring
 - SAN Boot
 - SSD

(XXX)


It levels the playing field when it comes to different hardware types.

No matter whether we have single disk, s/w mirroring, h/w mirroring, SAN boot or SSD it's all the same to us (more or less). Our management above the hardware layer is exactly the same.

After all, it's pretty difficult to split mirrors on a single-disk system.

We're not even touching on the advantages of using ZFS on SSDs. I'm sure Joy can get someone else to talk on that soon!

So, apart from all this, what else can ZFS boot do for us?



More ZFS Boot Features

- Compression
- Swap & Dump management
- ZFS Flash install in U8 (10/09)
- ZFS Supported in Jumpstart
- ZFS Backup Solutions

- *De-dupe?*
- *zpool split/join?*

(XXX)

We can apply lzjb compression to our boot file systems. No more gzip'ing log files when they get a bit big. It's not the best compression ZFS offers but it's better than nothing and the only currently supported option for root filesystems.

We can now, more or less, dynamically manage swap and dump space with the use of zvols.

ZFS Flash, Jumpstart and ZFS Open Backup Solutions

These three really complete the ZFS boot picture for me. We saw in the ZFS OpenBackup presentation back in July how we'll never need to do full backups again.

Now combine this with the ability to create ZFS Flash Archives on your ZFS OpenBackup server ready to deploy new ZFS boot'd servers via Jumpstart.


OK – it will need some work to put it all together but nothing compared to what we've had to do with SVM and VxVM in the past.

(XXX)

I've added these two options as futures enhancements I'm looking forward to.

De-dupe will save us so much space on our ZFS OpenBackup server.
Especially if we have a thousand ZFS boot images.

Zpool split I think is coming. I know I can just pull a ZFS disk but it would be nice to have a more controlled way to do it that keeps the pool consistent.



Summary

- UFS has served us well
- Other VMs are “heavyweight” and/or costly
- UNIX skill sets are changing
- Unified System Administration

(XXX)

So, in summary, UFS has served us well...so well, it's lasted for 25 years. It's time for change and I only hope that ZFS will last the same amount of time.

All other volume managers I've known require a lot of maintenance and specialist knowledge. ZFS is not only free but is comprised of just two commands – zpool and zfs. For an aging population of Sys Admins such as me, this is much better than remembering all the various Vx commands and where they live!

UNIX skill sets are changing – try finding someone with an in-depth knowledge of fixing corrupted Veritas private regions! It's even worse with all the convoluted, custom script solutions out there – who maintains those when people move on?

For all our Solaris systems, whether they be SPARC or x86, we can now have a simple, unified approach to managing our boot environments.

Unfortunately, it can't help our linux systems, or any other unix systems come to that. Maybe one day eh!

So That's it - apart from the obligatory links page

(XXX)

Slide 11



Handy Links

Big Admin
<http://www.sun.com/bigadmin/topics/zfs/>

Boot Disk Management – John Howard, David Deeths
<http://www.sun.com/books/catalog/howard.xml>

OpenSolaris ZFS Resources
<http://www.opensolaris.org/os/community/zfs/>

Solaris 10 – 10/09
<http://docs.sun.com/app/docs/coll/1236.11?en>

Lori Alt – Staff Engineer at Sun
http://blogs.sun.com/storage/entry/what_we_re_watching_lori

OpenBackups using ZFS – LOSUG Presentation - Sally Houghton
<http://opensolaris.org/os/project/losug/files/June2009/>

Solaris Internals Wiki – ZFS Root Pool Info
http://www.solarisinternals.com/wiki/index.php/ZFS_Troubleshooting_Guide

Here we have the usually culprits but of special note are

- Lori Alt's ZFS boot presentation
- The July LOSUG presentation on ZFS Open Backups

Any questions is what I should say now.....