STORAGE DEVELOPER CONFERENCE
Where The Storage Development Community Connects
2007

ZFS as a Root File System

Lori Alt
Sun Microsystems, Inc.

What Does It Take to be a Root File System?

- Boot capability
- Robustness characteristics (such as mirroring)
- Installation support
- Swap and dump support
- Ongoing management capabilities (upgrade, patching, snapshots, etc.)

Why use ZFS as a Root File System?

- There is a benefit to having only one file system type to understand and manage (assuming ZFS is already in use for data).
- ZFS's features make it an excellent root file system with many management advantages.
- At least for Solaris, it's the coming thing. New installation and management features will depend on it.

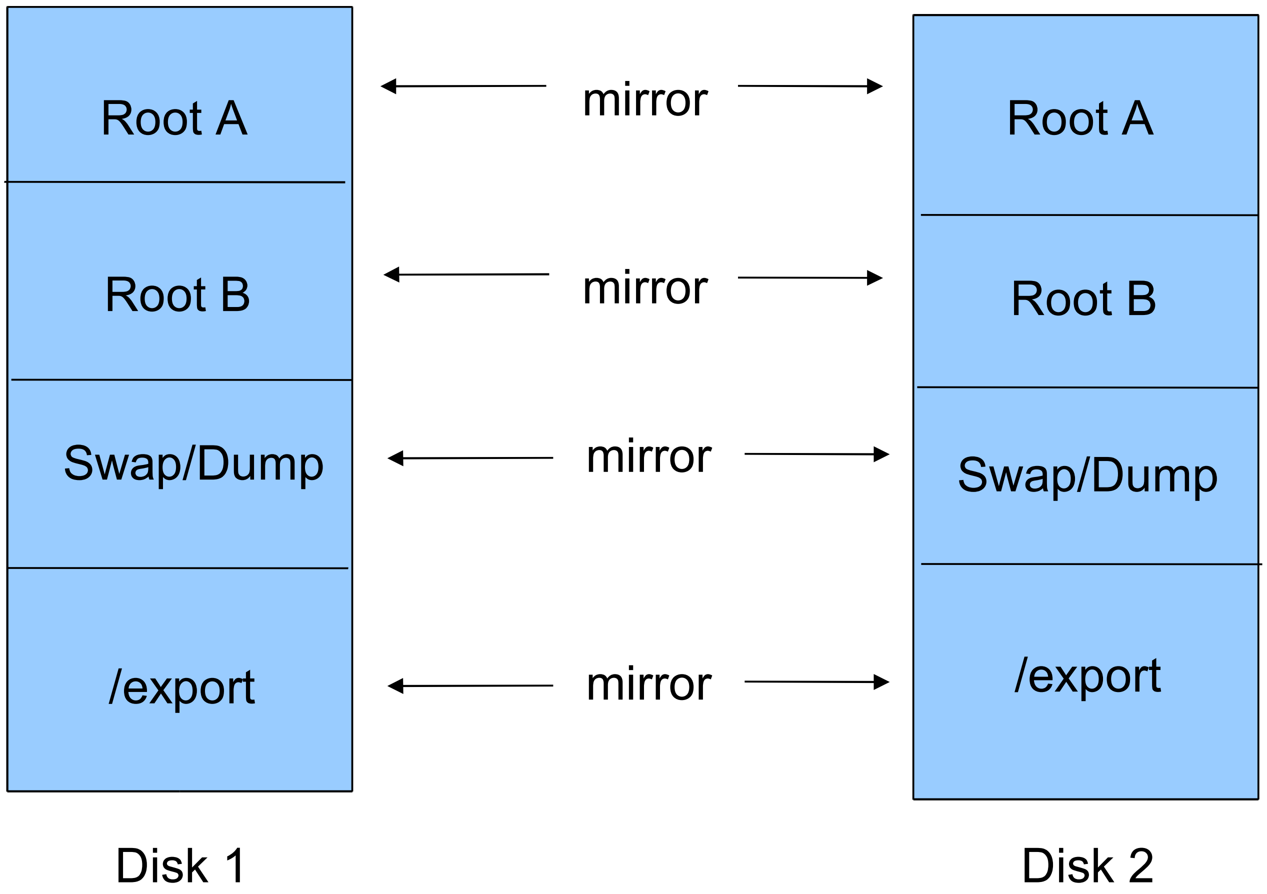
ZFS Features that Matter (for Root File Systems)

- Pooled Storage – No need to preallocate volumes. File systems only use as much space as they need
- Built-in redundancy capabilities (such as mirroring) at the pool level.
- Unparalleled data integrity features. On-disk consistency always maintained—no fsck.

More ZFS Features that Matter (for Root File Systems)

- Snapshots and clones (writable snapshots)– instantaneous, nearly free, persistent, and unlimited in size and number (except by the size of the pool)
- ZFS volumes (zvols) can be used for in-pool swap and dump areas (no need for a swap/dump slice). One pool does it all.

Storage Layout for System Software with Traditional File Systems



Root A ← mirror → Root A

Root B ← mirror → Root B

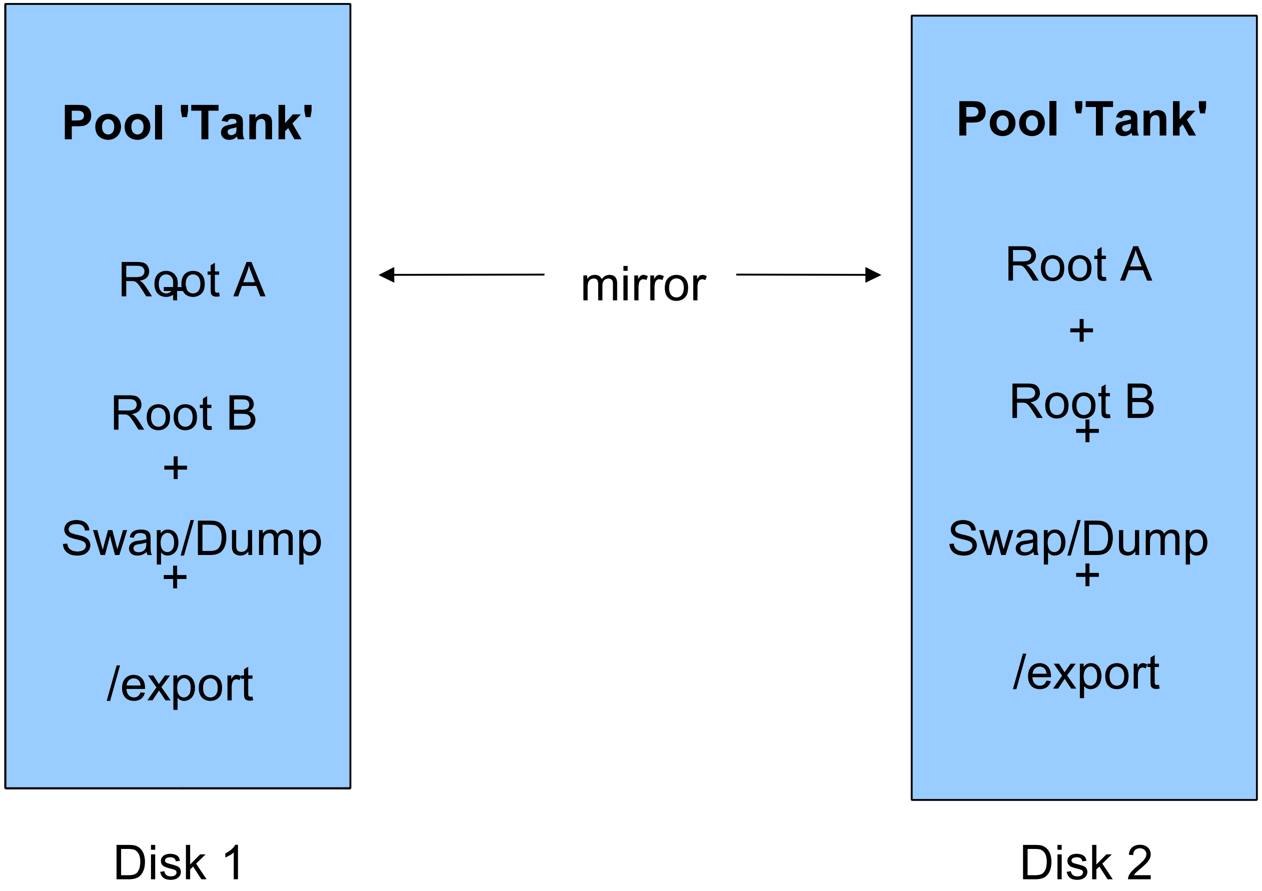
Swap/Dump ← mirror → Swap/Dump

/export ← mirror → /export

Disk 1

Disk 2

Storage Layout for System Software with a ZFS Storage Pool



Pool 'Tank'

Root A
+
Root B
+
Swap/Dump
+
/export

Disk 1

mirror

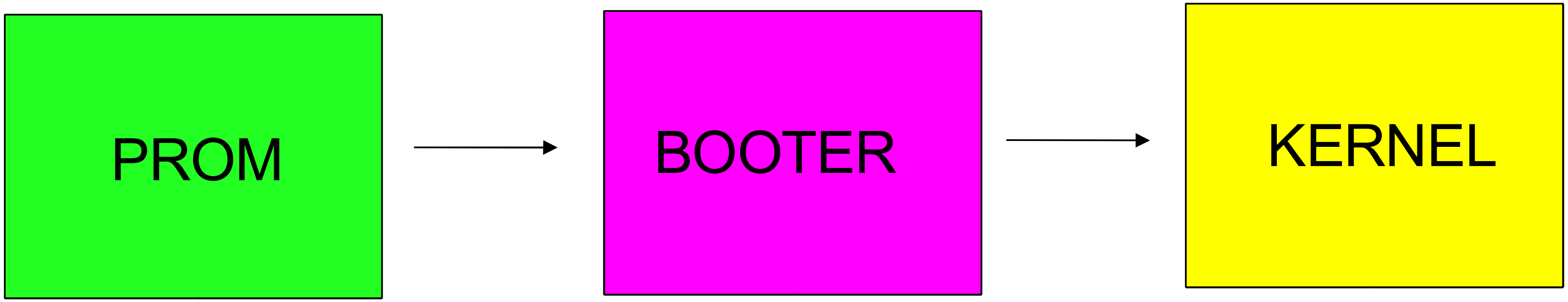
Pool 'Tank'

Root A
+
Root B
+
Swap/Dump
+
/export

Disk 2

A Short Primer on Booting Solaris

Three Phases:



PROM → BOOTER → KERNEL

Booting Solaris – PROM phase

1) The PROM (BIOS on x86, Open-Boot Prom on SPARC) identifies a boot device.
2) The PROM loads and executes a booter from the boot device.

Booting Solaris – Booter phase

1) The booter selects a root file system.
2) The booter loads one or more files from the root file system into memory and executes one of them. The executable file is either part of the Solaris kernel, or a program that knows how to load the Solaris kernel.

Booting Solaris – Kernel phase

1) The kernel uses the I/O facilities provided by the booter to load the necessary kernel modules and files (drivers, file system, and some control files) in order to do its own I/O and mount the root file system.
2) The root file system is mounted and system initialization is performed.

Booting from ZFS – PROM phase

- At the PROM stage, booting ZFS is essentially the same as booting any other file system type.
- The boot device identifies a storage pool, **not** a root file system.
- At this time, the booter which gets loaded is GRUB 0.95 on x86 platforms, and is a standalone ZFS reader on SPARC platforms.

Booting from ZFS – Booter phase

- With ZFS, there is no one-to-one correspondence between boot device and root file system. A boot device identifies a storage pool, not a file system. Storage pools can contain multiple root file systems.
- Thus, the booter phase must have a way to select among the available root file systems in the pool.
- The booter must have a way of identifying the default root file system to be booted, and also must provide a way for a user to override the default.

Booting from ZFS – Booter phase, Root File System Selection

- Root pools have a “bootfs” property that identifies the default root file system.
- We need a control file that lists all of the available root file systems, but in which file system do we store it? (we don't want to keep it in any particular root file system).
- Answer: keep it in the “pool dataset”, which is the dataset at the root of the dataset hierarchy. There's only one of them per pool and it's guaranteed to be there.

Booting from ZFS – Booter phase, Root File System Selection - x86

- On x86 platforms, the GRUB menu provides a way to list alternate root file systems.
- One of the GRUB menu entries is designated as the default.
- This default entry (or any other, for that matter) can be set up to mount the pool's default root file system (indicated by the pool's “bootfs” property).

Booting from ZFS – Booter phase, Root File System Selection - SPARC

- On SPARC platforms, a control file will list the available root file systems.
- A simple “boot” or “boot disk” command at the OBP prompt will boot whatever root file system is identified by the “bootfs” pool property.
- There will also be a standalone program available which presents a list of the available root file systems and allows the user to select one of the roots for booting.

Booting from ZFS – Booter phase, Loading the Kernel

- Once the root file system is identified, the paths to the files needed for booting are resolved **in that root file system's name space.**
- The booter loads the kernel's initial executable file (and other files, as necessary) and executes the kernel.

Booting from ZFS – Kernel phase

- The booter has passed (1) the device identifier of the boot device, and (2) the name and type of the root file system as arguments to the kernel.
- Because the root file system is ZFS, the ZFS file system module is loaded and its “mountroot” function is called.
- The ZFS mountroot function reads the pool metadata from the boot device, initializes the pool, and mounts the designated dataset as root.

Boot Environments

- A **boot environment** is a root file system, plus all of its subordinate file systems (i.e., the file systems that are mounted under it)
- There is a one-to-one correspondence between boot environments and root file systems.
- A **boot environment** (sometimes abbreviated as a **BE**) is a fundamental object in Solaris system software management.

Using Boot Environments

- There can be multiple boot environments on a system, varying by version, patch level, or configuration.
- Boot environments can be related (for example, one BE might be a modified copy of another BE).
- Multiple BEs allow for safe application and testing of configuration changes.

The “Clone and Modify” Model of System Updates

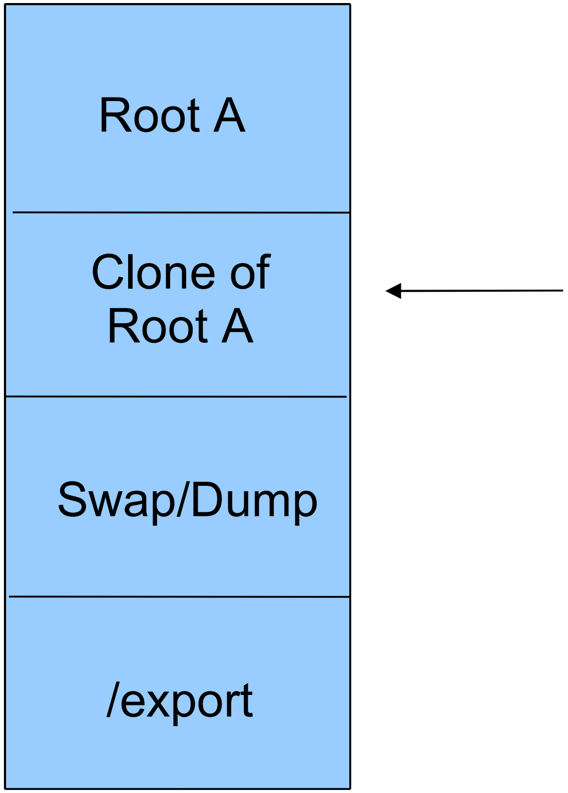
In-place updates of boot environments can be risky and time-consuming. A safer model is to do the following:

- Make a new boot environment which is a clone of the current active boot environment.
- Update the clone (upgrade, patch, or reconfigure)
- Boot the updated clone BE.
- If the clone is acceptable, make it the new active BE. If not, leave the old one active.

“Clone and Modify” Tools

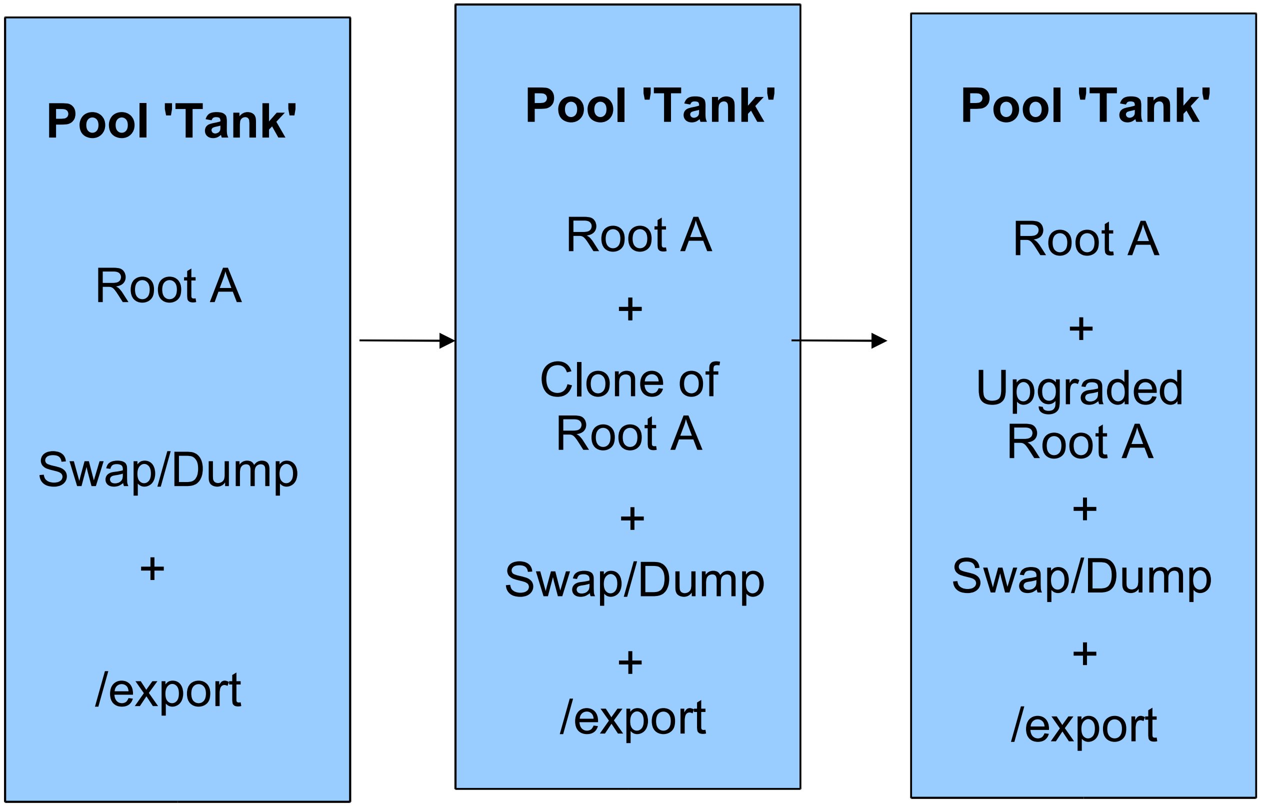
- Solaris supports a set of tools calls “LiveUpgrade”, which do cloning of boot environments for the purpose of safe upgrades and patching
- New install technology under development will support this also.
- ZFS is ideally suited to making “clone and modify” fast, easy, and space-efficient. Both “clone and modify” tools will work **much** better if your root file system is ZFS. (The new install tool will require it for some features.)

Clone and Modify with Traditional File Systems



Then upgrade this root file system

Clone and Modify with a ZFS Storage Pool



Pool 'Tank'

Root A

Swap/Dump

+

/export

Pool 'Tank'

Root A

+

Clone of Root A

+

Swap/Dump

+

/export

Pool 'Tank'

Root A

+

Upgraded Root A

+

Swap/Dump

+

/export

Initial State

After Clone

After Upgrade

Boot Environment Management with ZFS

- Boot environments can be composed of multiple datasets, with exactly one root file system.
- Regardless of how many datasets compose the boot environment, the “clone and modify” tools will treat the boot environment as a single manageable object.

The ZFS “Safe” Upgrade

The low-risk, almost-no-down-time system upgrade (using LiveUpgrade):

```
# lucreate -n S10_U6
# luupgrade -n S10_U6 -s \
      /cdrom/Solaris_10_U6
# luactivate S10_U6
[  reboot  ]
```

What Happens During the ZFS "Safe" Upgrade

lucreate

- Does a ZFS snapshot of the datasets in the current Boot Environment, and then clones them to create writable copies
- Requires almost no additional disk space and occurs almost instantaneously (because ZFS cloning works by copy-on-write).

What Happens During the ZFS “Safe” Upgrade

luupgrade

- The system remains “live” (still running the original root) during the upgrade of the clone.
- The upgrade gradually increases the amount of disk space used as copy-on-write takes place. New space is required only for files that are modified by the upgrade.

What Happens During the ZFS "Safe" Upgrade

luactivate

- Make the specified boot environment the new active BE. Both the old and the new BE are available from the boot menu (but the new one is the default).

<reboot>

- User can select either the old or the new BE. If the new BE fails for some reason, the system can be booted from the old BE.

What Happens During the ZFS “Safe” Upgrade

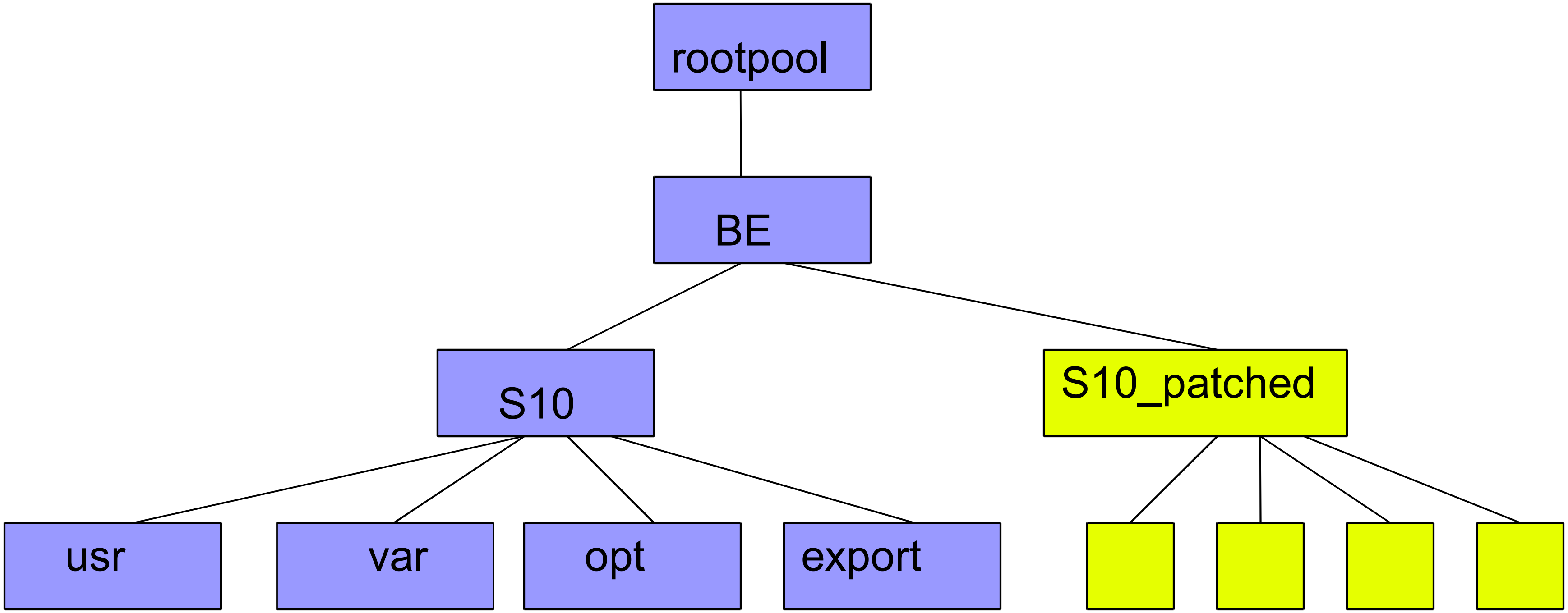
ludestroy

- At some point, the old BE can be destroyed.

Boot Environment Management with ZFS

- Boot environments will typically be composed of multiple datasets.
- The recommended configuration will be to have separate datasets for root, /usr, /var, /opt, /export and any optional directories placed under root (such as a /zoneroots directory, for example).

Boot Environment Dataset Hierarchy



Boot Environments Composed of Multiple Datasets

Why do this? Why split out /usr and so on?

- Keeps the root file system small (critical for eventually supporting boot from RAID-Z devices, because root will have to be replicated on all devices in a pool.)
- Allows parts of the Solaris name space to have different kinds of storage characteristics (such as compression).
- Allows a directory such as /var/log to be shared between boot environments

Boot Environments Composed of Multiple Datasets

Why **not** split out /usr and other directories?

- ZFS file system are more like directories than traditional file systems. Why not use them that way when it helps administration?
- Pooled storage means that the file systems don't have to have to be pre-allocated.
- About the only capability you lose if /usr is a separate file system is the ability to establish hard links between root and /usr.

ZFS Boot Limitations

- Currently, root pools can be n-way mirrors only (no striping or RAID-Z). We hope to relax this restriction in the next release.
- On Solaris, root pools cannot have EFI labels (the boot firmware doesn't support booting from them).

Installation – Near Term

- The existing Solaris install software is being adapted to set up a root pool and a root dataset and install Solaris into the root dataset (and its subordinate datasets).
- This will work with both the interactive install and the profile-driven install (Jumpstart).
- Customization features will be limited.

Installation – Future

- New installation software is currently under development which will leverage ZFS's capabilities from the outset.
- Installation will be much easier with ZFS: no need to slice up a disk into separate volumes for root, swap, /export, and so on.
- See:

http://opensolaris.org/os/project/caiman

Further Information

- Check out:

  http://opensolaris.org/os/community/zfs/boot

- We welcome ideas for how to use ZFS to manage software.