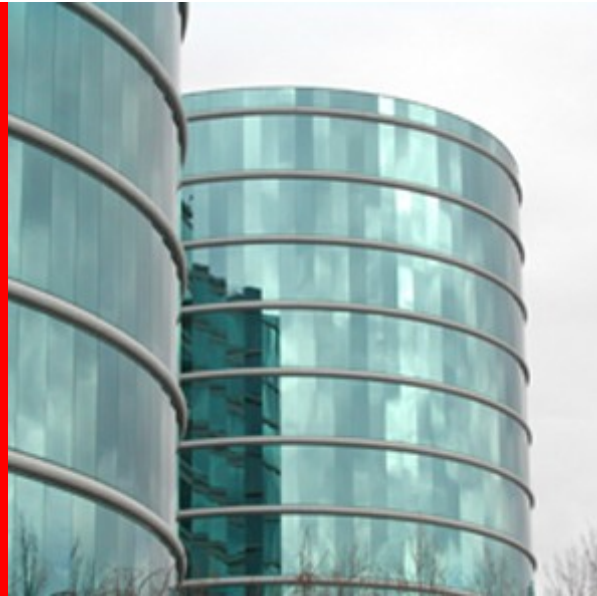


**ORACLE®**



**ORACLE<sup>®</sup>**

## **From Nuts to Soup: What's Cooking in Oracle Solaris 11 Express**

Jason Schroeder  
Principal Sales Consultant  
February 2011  
[jason.schroeder@oracle.com](mailto:jason.schroeder@oracle.com)



# So... What's in a NAME?

**ORACLE®**

**SOLARIS**

**Oracle Solaris Express**

OpenSolaris

Solaris Express Developer Edition

Solaris Express Community Edition

**Solaris 10**

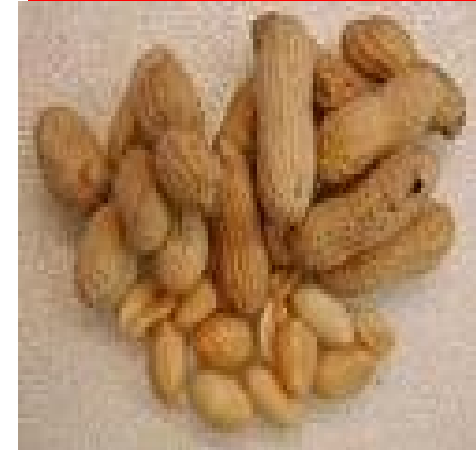
**Solaris 9**

**Solaris 8**



**ORACLE®**

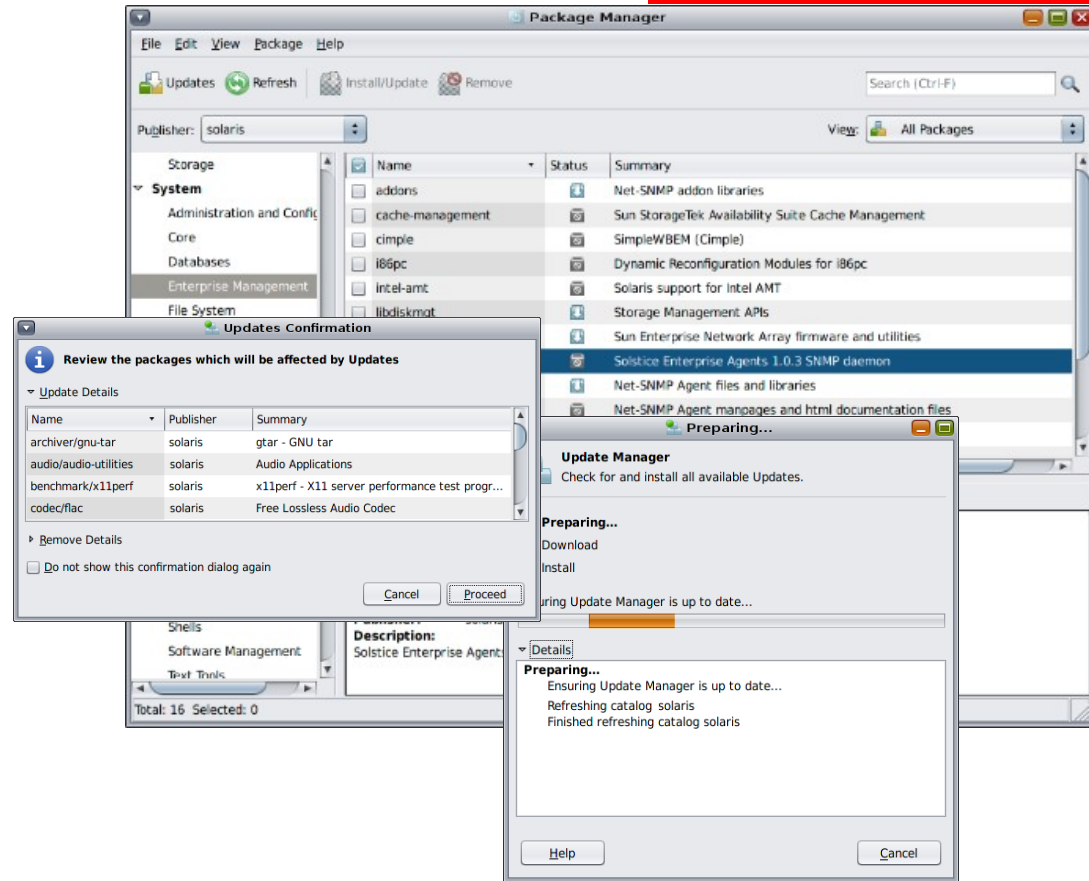
# So... Elephant in the Room?



# So... What else should we talk about?

- Oracle Solaris 11 is (by far) the longest release we've ever done
  - S8: 38 builds
  - S9: 58 builds
  - S10: 74 builds
  - S11: 156 and counting...
- That should give us some ideas, no?

- Property value ordering (95)
- Template extensions (102)
- Early manifest import (137)
- Networking type extensions (141)
- FMA integration (146)



# Image Packaging System

# IPS Cascades Quite a Waterfall

Solaris 10	Solaris 11 Express
SVR4 Packages	IPS Packages
Install DVD	Install CD + pkg repository
Live Upgrade	Boot Environments
Upgrade from installer	pkg(1), Update Manager
JumpStart	Automated Installer(AI)
JumpStart Profiles	AI manifests
Flash Install	No equivalent yet
Blueprints for custom DVD's	Distribution Constructor

# Boot Environments

- Make updates safe, reliable, recoverable
- Different from, and simpler than, Solaris 10 Live Upgrade
  - ZFS only
- Managed by **beadm(1M)**, functionality includes:
  - List
  - Activate
  - Create, Destroy, Rename
  - Mount, Unmount
- BE's are “free”; use liberally as an administrative safety net

# Interactive Installation

- Text-based UI for server systems (SPARC & x86)
- GUI for x86 desktop/laptop systems
- Principle: Install fixed software payload with basic configuration, customize afterwards
- Static networking configuration planned for text installer
- Both provide configuration of initial user account, with administrative privileges via `sudo`
  - Root configured as a role

# Automated Installation (AI)

- Lower up-front and ongoing costs of deploying Solaris-based software stack
- Leverages ZFS, SMF, IPS features to provide enhanced features vs. JumpStart
  - Reduces need for third-party or customer-developed extensions
  - Most scripting moved to first-boot SMF services
- Integrated, seamless Zones deployment (future)
- WAN-capable design provides operational flexibility
- Designed to be manageable and observable
  - `installadm(1M)` provides one-stop management interface

# Three Steps to Deploy Automated Installation

```
# wget <download server>/ai_sparc_image.iso
# pkg install installadm
DOWNLOAD                PKGS          FILES          XFER (MB)
Completed                10/10         1128/1128      6.30/6.30

PHASE                    ACTIONS
Install Phase            1509/1509
PHASE                    ITEMS
Reading Existing Index   8/8
Indexing Packages        10/10

# installadm create-service -a sparc -n solaris_11 \
-i 192.168.1.10 -c 3 -s ai_sparc_image.iso \
/export/ai/solaris_11
```

# Distribution Constructor (DC)

- Tool to easily construct installation images and virtual machine images
  - Used by Solaris engineering to build the product
- Use DC to build AI (or interactive install) images customized with additional drivers or services
- XML manifest (similar to AI) specifies construction
- Checkpoint/resume feature nicely leverages ZFS!
- Fully extensible – plug your own customizations into build process
- See `distro_const(1M)` for more information

## Principle features

- Primary means of software distribution is networked – HTTP[S] or NFS.
- Package installation automatically includes closure of any missing dependencies.
- Package updating (“patching/upgrade”) results in downloading and only installing differences between installed and desired versions.
- Full cryptographic signature support is present to insure end to end package and catalog integrity.
- IPS leverages ZFS snapshot and clone mechanisms to automatically apply changes in cloned boot environments, if needed.

## Packaging basics, continued.

- A package is specified by an *FMRI*.
- Package FMRI consists of the following parts:

`pkg://{publisher}/{package name}@{version}`

- Publisher represents who published the package
- The package name consists of a hierarchical name space separated by '/' characters of arbitrary depth. Package versions consist of four sequences of numbers of the form `{component version },{build version}-{branch version}:{time stamp}`

where the component, build and branch versions consist of an arbitrary number of integer elements separated by dots, and the time stamp is the time of publication in %Y%m%dT%H%M%SZ format in UTC.

`pkg://solaris/package/pkg@0.5.11,5.11-0.151:20101027T054323Z`

## Packaging basics, continued.

- The contents of a package are specified by a *manifest*, which contains *actions*: files, dirs, links, dependencies, users, groups, etc:

```
dir group=sys mode=0755 owner=root path=etc
```

```
user gcos-field="pkg(5) server UID" group=pkg5srv uid=97 username=pkg5srv
```

- Actions can have additional attributes of the form name=value added; name may be repeated.
- Current action types include:
  - Files, directories, symbolic links, hard links (ref counted).
  - Devices, users, groups
  - Set (generic package meta data), legacy (SVR4 compatibility information)
  - Dependencies
  - Signatures

## Packaging basics (cont.)

- Packages can be installed, uninstalled, or be updated or downgraded. If needed, they can also be checked and repaired should accidental damage occur.
- Modifying several packages is done as a single operation with the blended differences between all packages. This implies:
  - No ordering for update
  - All side effects wait until all packages are installed
  - Cycles in dependency graph are permissible
  - Arbitrary exchange of content, including editable files, is supported across upgrade; downloads are optimized accordingly.

## Packaging basics (cont.)

- Install-time side effects on live images are provided via *actuators*. Current actuators include:
  - **disable\_fmri, refresh\_fmri, restart\_fmri, suspend\_fmri**  
these tags cause the specified SMF service instance to be disabled, restarted, etc. when this action changes state.
  - **Reboot-needed=true**  
This tag marks the action as requiring a reboot when modified on a live image; as a result, if this tag is present on any modified actions the packaging transition is applied to a cloned boot environment rather than the live image.

# Practical examples

- Installing a package:

```
barts@cyber[18]; sudo pkg install gimp
```

```
    Packages to install:      1
  Create boot environment:    No
    Services to restart:     2
```

```
DOWNLOAD                                PKGS      FILES      XFER (MB)
Completed                               1/1      1713/1713  10.0/10.0
```

```
PHASE                                    ACTIONS
Install Phase                           1896/1896
```

```
PHASE                                    ITEMS
Package State Update Phase              1/1
Image State Update Phase                  2/2
```

```
PHASE                                    ITEMS
Reading Existing Index                    8/8
Indexing Packages                          1/1
```

# Practical examples

- Updating a package:

```
barts@cyber[18]; sudo pkg update pkg
```

```
          Packages to update:      1  
          Create boot environment: No
```

```
DOWNLOAD          PKGS          FILES          XFER (MB)  
Completed          1/1          118/118          0.6/0.6
```

```
PHASE              ACTIONS  
Update Phase      240/240
```

```
PHASE              ITEMS  
Package State Update Phase      2/2  
Package Cache Update Phase      1/1  
Image State Update Phase        2/2
```

```
PHASE              ITEMS  
Reading Existing Index          8/8  
Indexing Packages               1/1
```

# Practical examples

- Updating all the software:

```
barts@cyber[27]; sudo pkg update
```

```
    Packages to install:      1
    Packages to update:     795
    Create boot environment:  Yes
```

```
DOWNLOAD                                PKGS           FILES          XFER (MB)
Completed                               796/796        4754/4754      205.2/205.2
```

```
PHASE                                ACTIONS
Removal Phase                         2561/2561
Install Phase                          3967/3967
Update Phase                           6277/6277
```

```
...
```

```
A clone of opensolaris-39 exists and has been updated and activated.
On the next boot the Boot Environment opensolaris-40 will be mounted on
'/'.
```

```
Reboot when ready to switch to this updated BE.
```

# Practical examples

- Searching for Software:

```
barts@cyber[30]; pkg search /usr/bin/nmap
```

INDEX	ACTION	VALUE	PACKAGE
path	file	usr/bin/nmap	pkg:/diagnostic/nmap@5.21-0.151
path	file	usr/bin/nmap	pkg:/diagnostic/nmap@5.21-0.151.0.1

- Listing installed packages

```
barts@cyber[32]; pkg list snort
```

```
pkg list: no packages matching 'snort' installed
```

```
barts@cyber[33]; pkg list nmap
```

NAME (PUBLISHER)	VERSION	STATE	UFOXI
diagnostic/nmap (solaris)	5.21-0.151	installed	u----

```
barts@cyber[34]; pkg list \*osnet*\*
```

NAME (PUBLISHER)	VERSION	STATE	UFOXI
consolidation/osnet/osnet-incorporation	0.5.11-0.153	installed	u----
developer/opensolaris/osnet	0.5.11-0.153	installed	u----

# Package publication tools

- Pkgrepo – create and manage repositories
- Pkgsend – build a package
  - Generate
    - build a new package manifest from directory, tarfile, SVR4 pkg, etc.
  - Publish
    - Send a manifest & file contents to a HTTP or file repository.
- Pkgrecv – retrieve contents from a repository into another repository
  - Handy for complete repo copies, merging repositories.
  - Accepts SSL keys and certs...

## Package publication tools (continued)

- Pkgmogrify – transform package manifests
  - Expand macros
  - Add/remove/modify metadata using regexps.
  - Check manifest contents.
- Pkgdepend – generate and resolve dependencies
  - Automated analysis of ELF, #! interpreters, python, hard links and SMF manifests.
  - More coming....
  - Adds dependency actions to manifests
- pkg.depotd – provide a packaging repository via HTTP/HTTPS.

# Package from a Directory

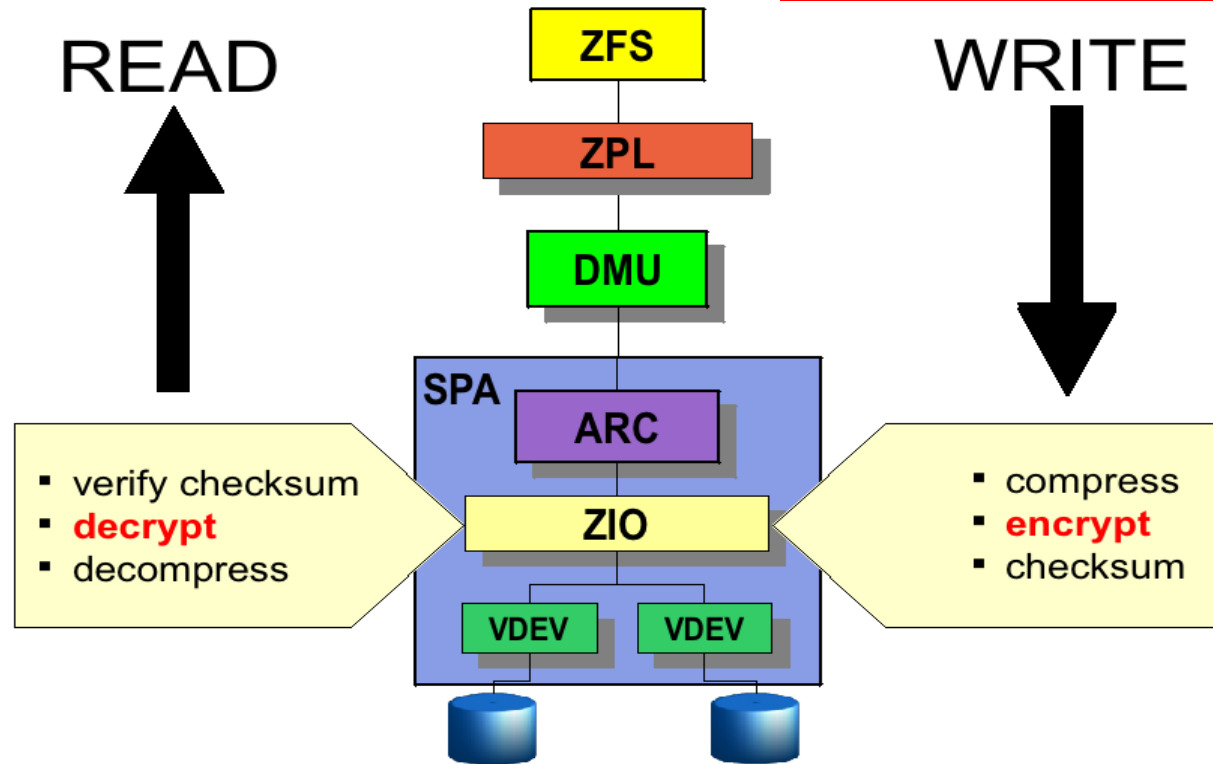
- Easy to package existing software

```
$ pkgrepo -s file:/tmp/test-repo create
$ pkgrepo -s file:/tmp/test-repo set publisher/prefix=lianep.oracle.com
$ pkgsend -s file:/tmp/test-repo open directory-test@1.0
$ export PKG_TRANS_ID=...
$ pkgsend -s file:/tmp/test-repo import directory-test
$ pkgsend -s file:/tmp/test-repo close
pkg://lianep.oracle.com/directory-test@1.0,5.11:20101109T012101Z
PUBLISHED
```

- Or ask pkg(5) to emit a manifest

```
$ pkgsend generate directory-test
dir group=bin mode=0755 owner=root path=opt timestamp=20101109T051058Z
dir group=bin mode=0755 owner=root path=opt/app timestamp=20101109T051110Z
file opt/app/app-bin group=bin mode=0555 owner=root path=opt/app/app-bin pkg.size=48088
file opt/app/app.conf group=bin mode=0644 owner=root path=opt/app/app.conf pkg.size=267
```

- Initial integration (27)
- FMA (36)
- Zones (39)
- Hot spares (42)
- Clone promotion (42)
- Bootable datasets (62)
- Hotplug (68)
- L2ARC (78)
- Boot support (88)
- Deduplication (128)
- Crypto (149)



## ZFS New Features

# ZFS Root Pool Enhancements

- ZFS root/boot/install/update features:
  - The beadm interface replaces `lu*` commands.
  - New auto installation provides these features:
    - Mirrored root pool creation
    - Swap and dump resizing
  - LiveCD and text-mode installer do not provide mirrored pool creation or swap and dump resizing.
  - With IPS, BEs are updated with latest build that includes bug fixes and features. No need to patch separately.
  - The `pkg image-update` or `pkg update` commands replace the need to create separate boot environments for patching and upgrades.
  - The `pkg update` process creates a new BE automatically.

# ZFS Root Pool – BE Upgrade Example

- After updating your ZFS BE with `pkg update`, boot to the new BE, `zfsBE-1`.

```
# pkg update ...
```

```
A clone of zfsBE exists and has been updated and activated.
```

```
On the next boot the Boot Environment zfsBE-1 will be mounted on '/'.  
Reboot when ready to switch to this updated BE.
```

```
# init 6 ...
```

```
# beadm list
```

BE	Active	Mountpoint	Space	Policy	Created
--	-----	-----	-----	-----	-----
zfsBE	-	-	9.38M	static	2010-10-15 09:18
zfsBE-1	NR	/	10.76G	static	2010-11-05 09:57

# ZFS Deduplication

- ZFS deduplication synchronously removes redundant data blocks, but is disabled by default.
- Determine if data would benefit from dedup:
  - Use `zdb -S pool-name` to simulate the potential space savings.
  - If the estimated dedup ratio is above 1.0, then you might see dedup space savings.
- Determine dedup memory requirements:
  - Use the `zdb -S` output to determine the dedup table sizing.
  - Each in-core dedup table entry is approximately 250 bytes.
  - Multiply the number of allocated blocks by 250.

# ZFS Deduplication - Configuration

- The `dedup` property is set on a ZFS file system.

```
# zfs set dedup=on tank/home
```

- Duplicate data blocks are removed synchronously. Only unique data is stored and common components are shared between files.

- A file system property, but the scope is pool-wide. For example, you can identify the dedup ratio as follows:

```
# zpool list tank
```

```
NAME SIZE  ALLOC FREE   CAP  DEDUP HEALTH ALTROOT
tank 136G  55.2G 80.8G 40%   2.30x ONLINE  -
```

- The `zfs list` output might not display accurate space accounting if `dedup` is enabled.

# ZFS Encryption - Configuration

- Encryption means that data is encoded for privacy. The data owner uses a key to access encoded data.
- Enable the `encryption` property when a file system is created. You are prompted for passphrase:

```
# zfs create -o encryption=on tank/home/darren
Enter passphrase for 'tank/home/darren': xxxxxxxx
Enter again: xxxxxxxx
```

- Default encryption algorithm is `aes-128-ccm` when a file system's `encryption` property value is `on`.
- A wrapping key encrypts the data encryption keys. The wrapping key is passed from the `zfs` command to the kernel. A wrapping key is either in a file (in raw or hex format) or it is derived from a *passphrase*.

# ZFS Encryption - Configuration

- Encrypt a file system with a raw key from a file. Use `pktool` to generate a raw key to a file and then identify the `keysource` property value as a file.

```
# pktool genkey keystore=file outkey=/dmkey.file keytype=aes  
keylen=256
```

```
# zfs create -o encryption=aes-256-ccm -o  
keysource=raw,file:///dmkey.file tank/home/darren
```

- A file system's encryption policy is inherited by descendent file systems and cannot be removed.

```
# zfs clone tank/home/darren@now tank/home/darren-new
```

```
Enter passphrase for 'tank/home/darren-new': xxxxxxxx
```

```
Enter again: xxxxxxxx
```

```
# zfs set encryption=off tank/home/darren-new
```

```
cannot set property for 'tank/home/darren-new': 'encryption'  
is readonly
```

# What is encrypted on Disk by ZFS ?

## Encrypted

- All “application” data
- POSIX layer data
- Permissions/ACL, owner
- Directory structure
- All ZVOL data
- All the above in a snapshot
- All the above in a clone

## In the clear

- Pool metadata
- Pool history
- Raid (vdev) config, etc.
- Dataset properties
- Dataset names
- Dataset user properties

# ZFS Performance

- Not all sync I/O benefits from lower latency
  - When latency isn't critical, it's cheaper to go to disk
  - With many disks, available bandwidth is far higher
- Welcome logbias
  - Allow per-dataset log bias for latency vs throughput
  - For Oracle redo logs:
    - Zfs set logbias=latency
  - For Oracle data files:
    - Zfs set logbias=throughput

# ZFS Synchronous Behavior - Tuning

- The `sync` property value descriptions:
  - `standard` – synchronous file system transactions, `fsync`, `O_DSYNC`, `O_SYNC`, are written to the intent log. The default value.
  - `always` – ensures that every file system transaction is written and flushed to stable storage by a returning system call. This value has a significant performance penalty.
  - `disabled` – synchronous requests are disabled. File system transactions only commit to stable storage on the next transaction group commit, which might be after many seconds. This value gives the best performance, with no risk of corrupting the pool. However, this value is very dangerous because ZFS is ignoring the synchronous transaction demands of applications, such as databases or NFS operations.

# ZFS Snapshot Differences (`zfs diff`)

- Use the `zfs diff` command to determine snapshot differences.

```
$ ls /tank/home/timh
```

```
fileA
```

```
$ zfs snapshot tank/home/timh@old
```

```
$ ls /tank/home/timh
```

```
fileA fileB
```

```
$ zfs snapshot tank/home/timh@new
```

```
$ zfs diff tank/home/timh@old tank/home/timh@new
```

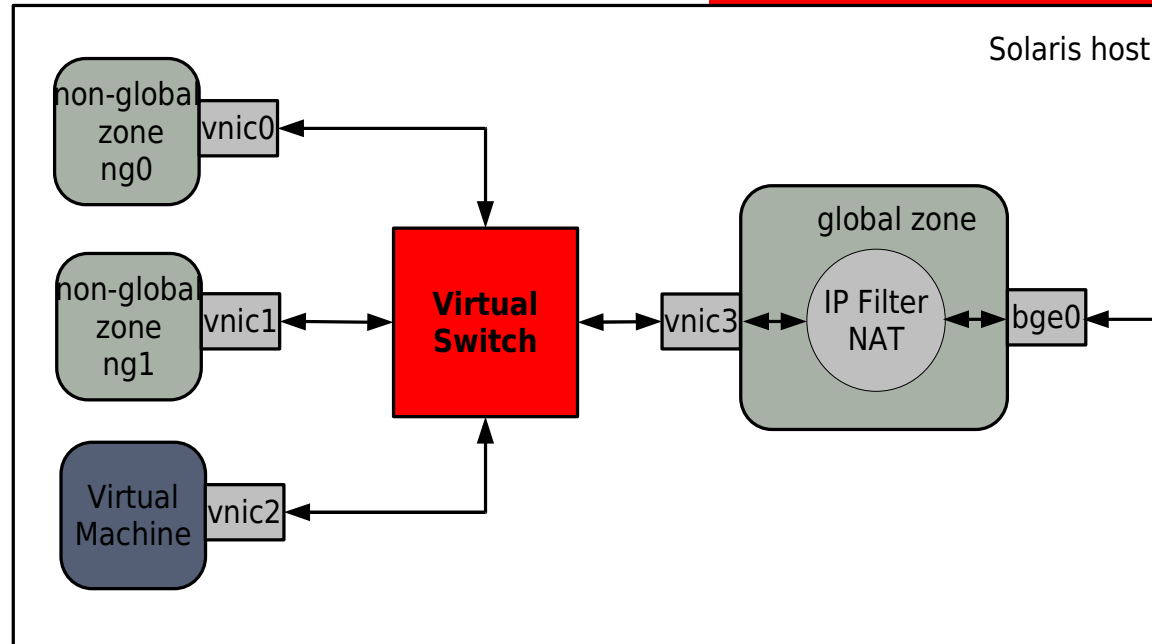
```
M      /tank/home/timh/
```

```
+      /tank/home/timh/fileB
```

# Additional ZFS Features

- Dynamic LUN expansion (`autoexpand` property)
- Split a mirrored storage pool with `zpool split`
- Triple parity RAID-Z (`raidz3`) support
- ZFS automatic snapshots
- User or group quotas
- Pool recovery (disks that silently ignore write ordering)
- Send/recv support for NDMP streams
- `zpool import -o readonly=on tank`
- Bootblocks applied automatically (new mirror or spare)
- Mirroring of some key raidZ metadata
  - Improves directory scans and the like

- GLDv3 aka Nemo (12)
- IP Instances (57)
- NWAM (62, 100, 134)
- Vanity naming (83)
- Tunnel reform (53)
- IPMP Rearchitecture (107)
- Enhanced dladm
- Virtualization and resource management (105, 136, 154)
- Low latency socket framework (106)
- lpadm (137)
- IP tunneling (125)



## Networking Meets Crossbow

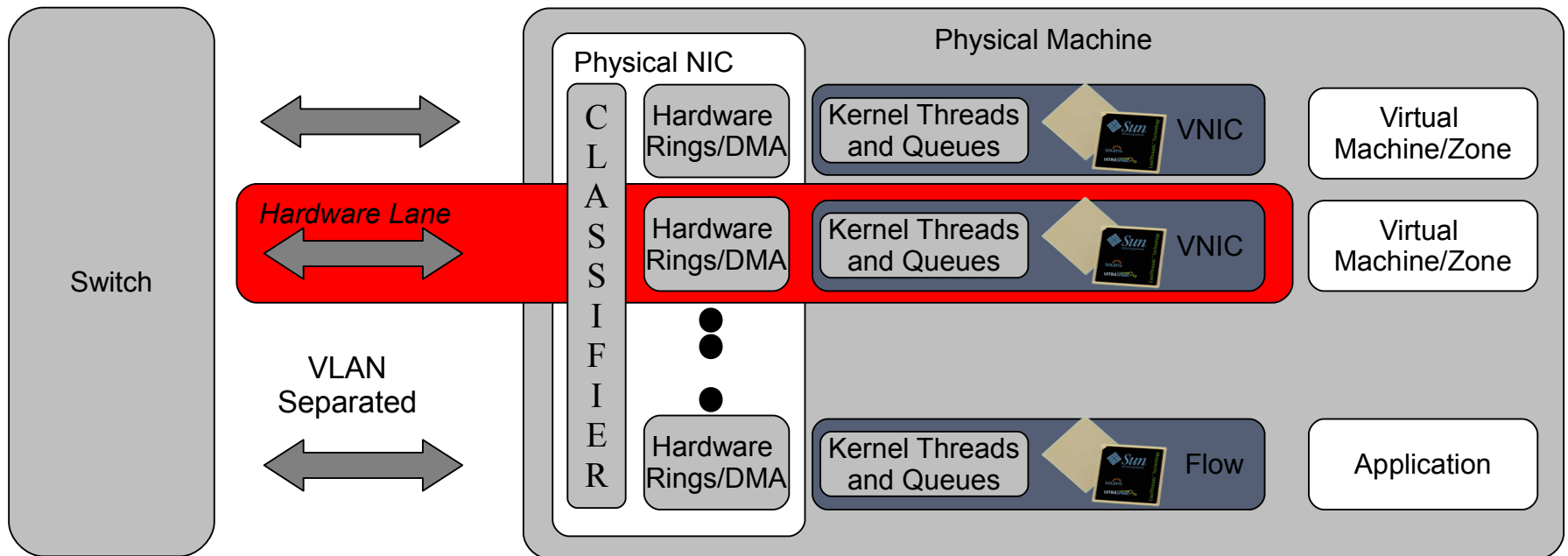
# Crossbow Features Overview

- NIC Virtualization
- IP Instances for Zones
- Bandwidth Partitioning
  - Associate priority and bandwidth limits to VNICs, services, protocols, connections.
- Scalability
  - Polling, fanout to multiple hardware and software threads
  - Exploits advanced NIC features
- Resource control
  - Constraint the CPUs used by NICs and VNICs
  - Tight integration with CPU pools
- Real time usage and history

# Crossbow Hardware Lanes

Ground Up Design for multi-core and multi-10GigE

- Linear Scalability by using Hardware Lanes with dedicated resources
- Network Virtualization and QoS designed in the stack
- More Efficiency due to dynamic polling and packet chaining



# Crossbow Virtual NICs (VNICs)

- Pseudo MAC instances
  - Can be managed as if they were physical NICs
  - Per VNICs stats, reuse existing management tools
  - Link speed derived from configured bandwidth limit
  - High-Availability by creating VNICs on link aggregations or combining VNICs in IPMP groups
- Dedicated MAC address
  - Random, Value, or factory (on NICs which provide multiple MAC addresses)
- Dedicated per-VNIC hardware and kernel resources
- Integrated in core network stack
  - Data path pass-through, no bump in the stack
  - Closely works with bandwidth control for efficient QoS
- Standard-based End-to-End Network Virtualization
  - VLAN tags and Priority Flow Control (PFC) assigned to VNIC extend Hardware Lanes to Switch

# Crossbow Virtual Switching

- A virtual switch is created implicitly each time 2 or more VNICs with the same VLAN id are created on a data link
- The MAC layer provides packet switching semantics equivalent to an Ethernet switch
  - Data path between VNICs created on top of the same data link
  - Connectivity between VNICs and physical network
  - Per VLAN broadcast domain, isolation between VLANs
- VNICs can be created on etherstub to create virtual switches independent from hardware
  - Many etherstubs and VNICs can be created to build arbitrary virtual network topologies
- Datapath between VNICs is implemented in software, no roundtrip to hardware required for better performance

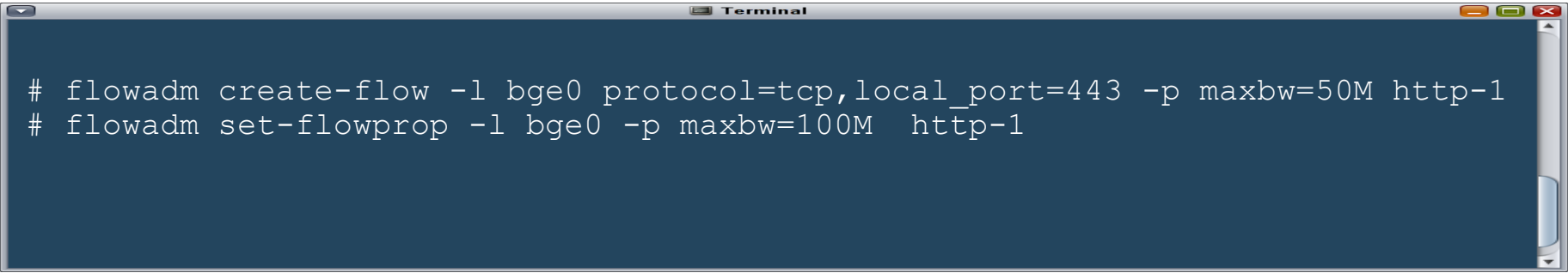
# Virtual NIC & Virtual Switch Usage

```
Terminal
# dladm create-vnic -l bge1 vnic1
# dladm create-vnic -l bge1 -m random -p maxbw=100M -p cpus=4,5,6 vnic2
# dladm create-etherstub vswitch1
# dladm show-etherstub
LINK
vswitch1
# dladm create-vnic -l vswitch1 -p maxbw=1000M vnic3
# dladm show-vnic
LINK          OVER      MACTYPE    MACVALUE    BANDWIDTH    CPUS
vnic1         bge1     factory    0:1:2:3:4:5  -            -
vnic2         bge1     random     2:5:6:7:8:9  max=100M     4,5,6
vnic3         vswitch1 random     4:3:4:7:0:1  max=1000M    -

# dladm create-vnic -l ixgbe0 -v 1055 -p maxbw=500M -p cpus=1,2 vnic9
```

# Crossbow Flows

- Crossbow flows based on the following attributes
  - Services (protocol + remote/local ports)
  - Transport (TCP, UDP, SCTP, iSCSI, etc)
  - IP addresses and IP subnets
  - DSCP labels
- Each flow can be assigned its own bandwidth limit
- Flows maintain their own statistics
  - Displayed in real-time through flowstat(1M)
  - History through extended accounting



```
Terminal
# flowadm create-flow -l bge0 protocol=tcp,local_port=443 -p maxbw=50M http-1
# flowadm set-flowprop -l bge0 -p maxbw=100M http-1
```

# Solaris 11 Network Configuration Overview

- Expand on the use of dladm for data link configuration
  - Physical NICs, VLANs, VNICs, etherstubs, link aggregations, WiFi, Bridges, IP tunnels
  - Persistent as well as temporary configuration
  - Unified properties
- Introduce ipadm to provide stable interface for IP configuration
  - Replacement for configuration via /etc/hostname\*
  - Configuration of persistent properties
- NWAM for automated IP configuration
  - Automatic configuration of IP services out of the box
  - Allows the user to switch between multiple locations with different security and name services configuration
  - Allows the invocation of user-specified hooks manually (CLI or GUI), or automatically (for example switch of locations)
  - Prioritized list of favorite Wireless LANs

# Data link Unification and Vanity Naming

- Data link Unification
  - GLDv3 support for GLDv2 and legacy DLPI drivers
- Vanity Naming
  - Flexible names for data links
  - Configured using `dladm(1M)` and `libdladm`
  - `/dev/net` file system for data links
  - Allows us to move from `e1000g0` style names to `net0` style names
- Proper VLAN management using `dladm` instead of “PPA hack” (`VID*1000+instance PPA`)

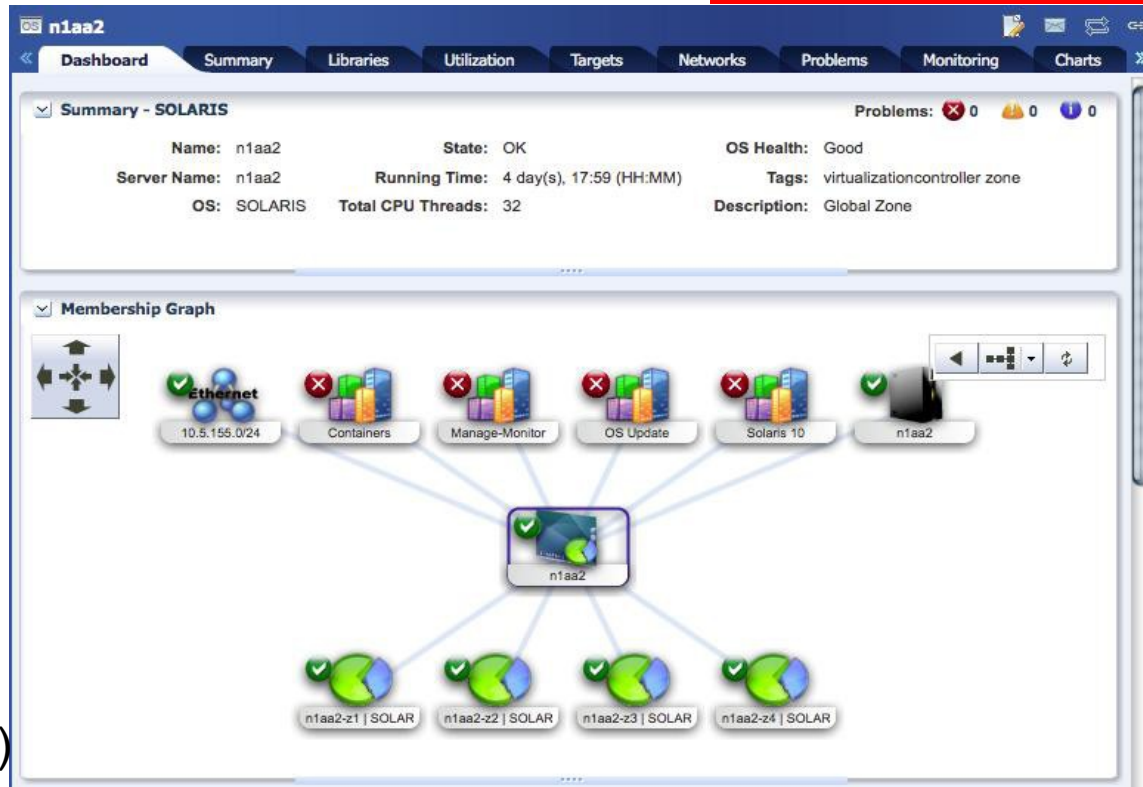
# Crossbow and IP over Infiniband (IPoIB)

- IP over IB (IPoIB) implemented by pseudo driver, registering with the Crossbow MAC layer, and layered on top of the IB verb layer
- dladm CLI was extended to be IB-aware
  - Physical HCA ports are represented by physical IB data links
  - IB partitions are managed through a set of {create,show,delete}-part subcommands
  - IB partitions data links for a specified P\_Key can be created on a physical IB data link
  - The resulting IB partition data link are plumbed by IP
    - Can be assigned to a non-global zone
  - dladm properties can be applied to IB data links

# Integration with Resource Pools

- Allows restricting the CPUs to be used for processing the network traffic for a data link (VNIC, physical NIC, aggregation, etc)
  - kernel threads
  - interrupts
- Configured through “pools” data link property
  - Alternative to “cpus” property
- CPUs are derived from the actual pool configuration, and kept up to date as CPUs migrate between pools
- Integrated with Zones dynamic pools to automatically bind/unbind a data link to the pool of a zone as it is booted/shutdown

- Rename (24)
- Upgrade (27, 53)
- Move & clone (33)
- Migration (36)
- Configurable privs (37)
- ZFS (39)
- System V resource controls (48)
- Update on attach (82)



## Zone New Features

# Shared-IP Zones: IPMP

- Address moves during failover/failback plagued the old model:
  - Zone would have ce0:2 one moment, ce1:1 the next
  - Zone administrator had no fixed point of control for the zone's IP addresses, nor any clues that the names might change
- With New IPMP:
  - IPMP no longer affects the zone – e.g., ipmp0:2 remains ipmp0:2 over the lifetime of the zone
  - Should the zone administrator be interested, the IFF\_IPMP flags will make it clear that the address is highly available

# Networking: Exclusive-IP Zones

- Extend and improve exclusive IP stack:
  - New `allowed-address` property constrains which IP addresses zone can use (via in-kernel L2/L3 protection)
  - `defrouter` property now supported for exclusive-IP zones

```
example# zonecfg -z myzone
zonecfg:myzone> set ip-type=exclusive
zonecfg:myzone> add net
zonecfg:myzone:net> set allowed-address=11.1.1.32/24
zonecfg:myzone:net> set physical=vnic0
zonecfg:myzone:net> set defrouter=11.1.1.1
zonecfg:myzone:net> end
zonecfg:myzone> commit
```

- Result: Shared-Stack style control over IP addresses with exclusive-stack features.

# Networking: Exclusive-IP Zones (2)

- Most of the benefits of network administrative rework accrue to zones: dladm, ipadm, IP tunnels, IPMP, etc.
- Crossbow and Zones work well together:
  - Create arbitrary numbers of vnics and assign them to zones, lifting S10 limitations for exclusive stack
  - Zones work seamlessly with crossbow virtual networks
  - Resource management and flow controls
- Layer 2 & 3 Networking Protections
  - Prevents exclusive stack zones from emitting various forms of mischevious traffic (MAC spoofing, IP spoofing, et cetera)
  - (Shared stack zones cannot by definition do these things)

# Introducing zonestat(1m)

```
$ zonestat 5
```

```
...
```

```
SUMMARY                Cpus/Online: 32/32    Physical: 32.0G    Virtual: 47.9G
```

```
-----CPU-----  ----PHYSICAL-----  -----VIRTUAL-----
```

```
ZONE  USED  %PART  %CAP  %SHRU  USED  PCT  %CAP  USED  PCT  %CAP
```

```
[total]  1.57  4.92%    -    -  5660M  17.2%    -  9.9G  20.6%    -
```

```
[system]  0.09  0.28%    -    -  5086M  15.5%    -  9275M  18.8%    -
```

```
kodiak-dp  1.00  100%    -  100%  46.0M  0.14%  4.49%  36.2M  0.07%  1.17%
```

```
  global  0.48  1.56%    -  1.56%  419M  1.27%    -  673M  1.37%    -
```

```
kodiak-ab  0.00  0.00%    -  0.01%  67.0M  0.20%    -  115M  0.23%    -
```

```
kodiak-rie  0.00  0.00%    -  0.02%  41.6M  0.12%    -  62.4M  0.12%    -
```

# Introducing zonestat(1m)

- `zonestatd` Daemon performs monitoring
  - Allows non-root users and non-global zones to see (some of) the information
- Zonestat can monitor:
  - virtual-memory, physical-memory, locked-memory, pool-psets, lwps, processes, shm-memory, shm-ids, sem-ids, msg-ids
- Limit output to specific zones
- Sort by various columns
- Machine parseable output mode
- End-of-run reporting for average, high, total usage.
- Drill down by resource type

# Introducing zonestat(1m)

- Example: Monitor lwps & processes:

```
$ zonestat -r processes,lwps 5
PROCESSES          SYSTEM LIMIT
system-limit      292K
                  ZONE  USED   PCT   CAP  %CAP
                  [total] 191  0.63%  -    -
                  [system]  0  0.00%  -    -
                  global  167  0.55%  -    -
                  foo    24  0.08%  300  8.00%

LWPS              SYSTEM LIMIT
system-limit      2047M
                  ZONE  USED   PCT   CAP  %CAP
                  [total] 713  0.00%  -    -
                  [system]  0  0.00%  -    -
                  global  618  0.00%  -    -
                  foo    95  0.00%  1000 9.50%
```

# Resource Management

- New max-processes resource control

```
example# zonecfg -z myzone
zonecfg:myzone> set max-processes=300
```

- prctl(1) now shows resource utilization:

```
example# prctl -i zone foo
zone: 4: foo
NAME      PRIVILEGE          VALUE      FLAG      ACTION
zone.max-lofi
  usage           0
  system          18.4E      max      deny
zone.max-swap
  usage           28.3MB
  privileged      3.00GB     -       deny
  system          16.0EB     max      deny
```

# Zones Security

- Delegated administration (via RBAC authorizations)
  - Authorizations can be configured directly in zonecfg(1m):

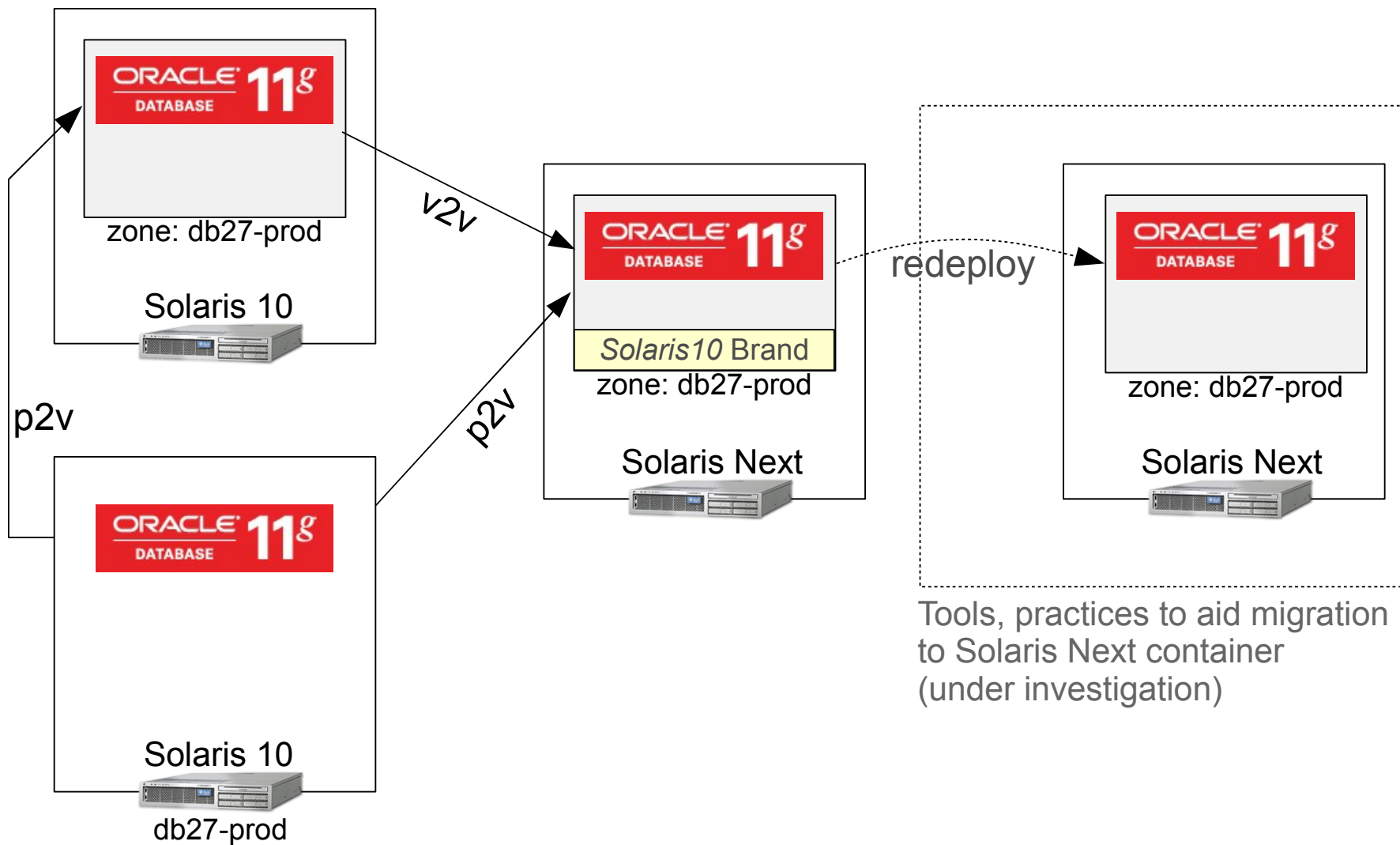
```
example# zonecfg -z myzone
zonecfg:myzone> add admin
zonecfg:myzone:admin> set user=jack
zonecfg:myzone:admin> set auths=login,manage
zonecfg:myzone:admin> end
zonecfg:myzone> commit
```

- Authorizations are implemented via /etc/user\_attr and synced there by zonecfg.

# Solaris 10 Containers on Solaris Next

- *solaris10* branded zone
  - Similar to existing solaris8 and solaris9 brands on S10
- Adoption and compatibility aid for Solaris Next
  - Protect investment in S10 (infrastructure, training, support)
  - Leverage new technology in an S10 context
    - e.g. Crossbow for Solaris 10
  - Avoid required application recertification
- p2v installation process
  - Also v2v for moving Solaris 10 native zones
- Supports Solaris 10 10/09 or later within the zone

# Expected Application Migration Path



- Sun4v (13)
- T3 (131)
- Nahalem & Westmere (97)
- Fast reboot (x86: 100, by default: 112, sparc 136)
- Fast crash dump (127)
- Root alternate home directory (87)
- Next Gen Audio (115)
- Modernization (136)
  - bash, ksh93, GNU patch, vim, etc
- Trusted extensions (37)
- Secure by default (42)
- IPSec Tunnel reform (53)
- Packet capture (125)
- ipmpstat
- L3/L4 Integrated load balance
- dlstat and flowstat
- Snoop Io0, intra zone
- Dtrace IP provider
- Getvmusage improvements
- Zoneadm attach -U ( > -u )
- Lofiadm in zones
- Root is a role (as it should)
- tpmadm
- NSA Suite B
- ikeadm
- Oracle Key Management
  - pkcs\_kms
- No reboot audit enable
- FCoE
- IPoIB
- EoIB
- NUMA I/O

## What's Left?

# Uh, What is the Soup Du Jour?



**Mmm. That sounds good. I'll have that.**

**SOFTWARE. HARDWARE. COMPLETE.**

**ORACLE®**