Solaris 10 OS Bootcamp

Linda Kateley
OS Ambassador
Data Center Practice
Sun Microsystems, Inc.

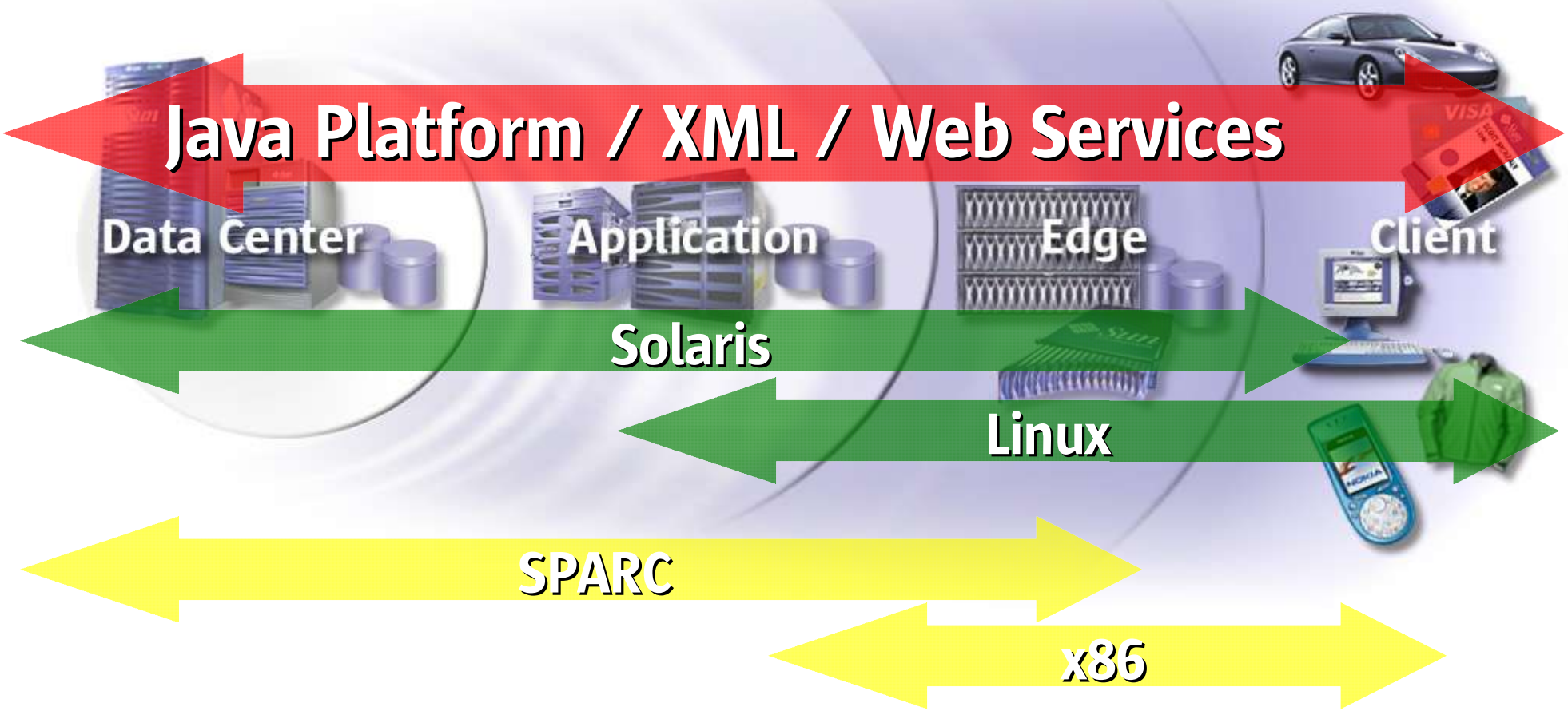
Sun microsystems

Agenda

- Sun's OS strategy
- Solaris 10 design goals
- Dtrace
  - Dtrace demo
- Zones
  - Zone demo
- Smurfs(SMF) and FMA
- ZFS and priv

Providing the Right Choices

End-to-End Services Delivery



Solaris 9 OS : The Services Platform

Manageability
- Resource Management
- Data management
- Provisioning

Security
- Fine-grained access control
- Secure remote mgt
- Strong authentication

Availability
- RAS profile
- Patch manager
- Sun Fire RAS

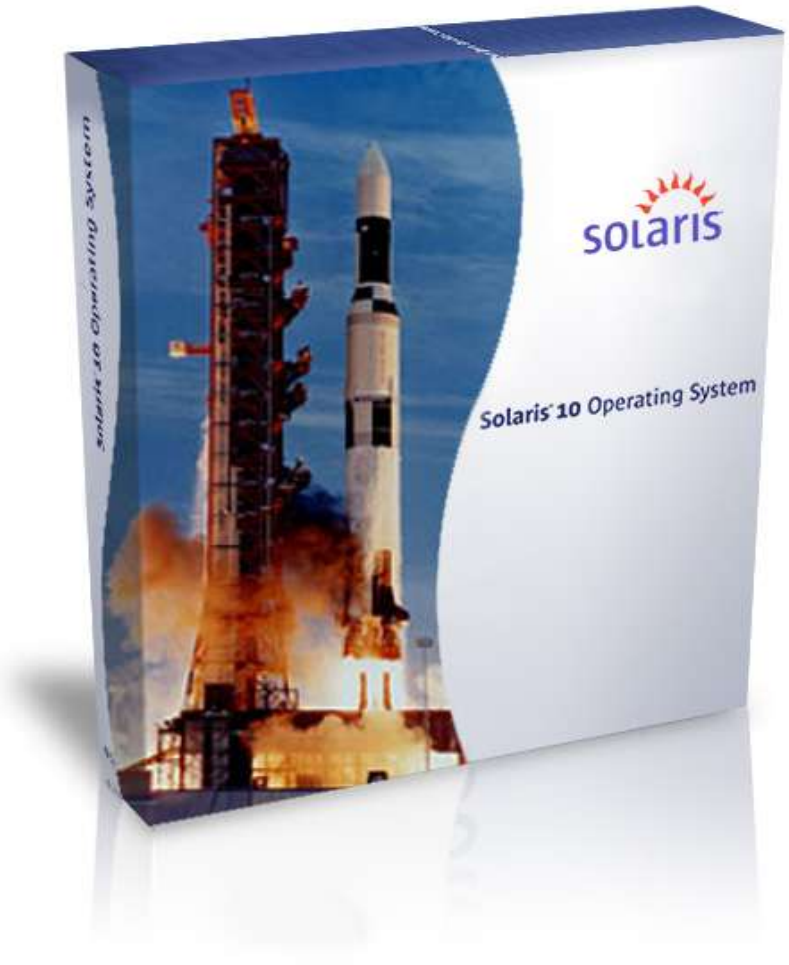
Scalability
- Threads
- Memory
- Data

Applications
- Cost savings
- Compatibility

Solaris 10 Milestones

(Future dates subject to change)

- Build 1: January 2002
- Production servers within Sun: April 2002
- First Solaris Express build available: Summer 2003
- Beta ship: Q1CY04
- Released Jan 31st 8:37 PM CST



Solaris 10

900,000+ Installs

400+ Supported Systems

400+ New ISVs

1,100+ x86 Applications

40+ Solaris OEMs

Extreme System Performance

Focus on Latency Reduction

- Portable microbenchmarks (libMicro) used to compare, tune OS performance
- Result: faster syscalls (25%+):
  - dup, fcntl, flock, getsockname, getpeername, gettimeofday, lseek, select, semop, setcontext, setsockopt, sigaction, siglongjmp, signal, sigprocmask, socket, time, times
- ...faster library functions (400%+):
  - strftime, mktime, localtime, getenv, SPARC str*

7x faster

3-5x faster

31x faster!

Extreme Performance

Source: internal Sun benchmarks

Subscription-based Service Plans for Solaris 10

Subscription Pricing

	Free	Basic	Standard	Premium
Solaris 10 OS security fixes	■	■	■	■
Regular Solaris 10 OS update releases	■	■	■	■
Solaris 10 OS overview Web training course	■	■	■	■
Sun Update Connection Web training course		■	■	■
Real time access to patches/fixes		■	■	■
System Edition of Sun Update Connection		■	■	■
Skills self-assessment		■	■	■
One Web course			■	■
Optional training credits			■	■
5 x 12 telephone support			■	■
7 x 24 telephone support				■
Interoperability services				■
U.S. $ Price/Socket/Year	**$0**	**$120**	**$240**	**$360**

Solaris 10—Design Principles

- Compatibility is paramount
  - Continued commitment to binary compatibility
  - Increased Linux compatibility
  - Developers can use SolCAT (and LinCAT) today to ensure compatibility of applications

Solaris 10—Design Principles

- Availability is critical
  - Provide a robust operating platform designed to continue operation in the face of hardware failures from below and software failures from above
- Security—everywhere, all the time
  - Ensure that Solaris systems are secure from the moment of installation
  - Allow for finer-grained delegation of security privileges (as with Trusted Solaris)
  - Enable highly secure containers

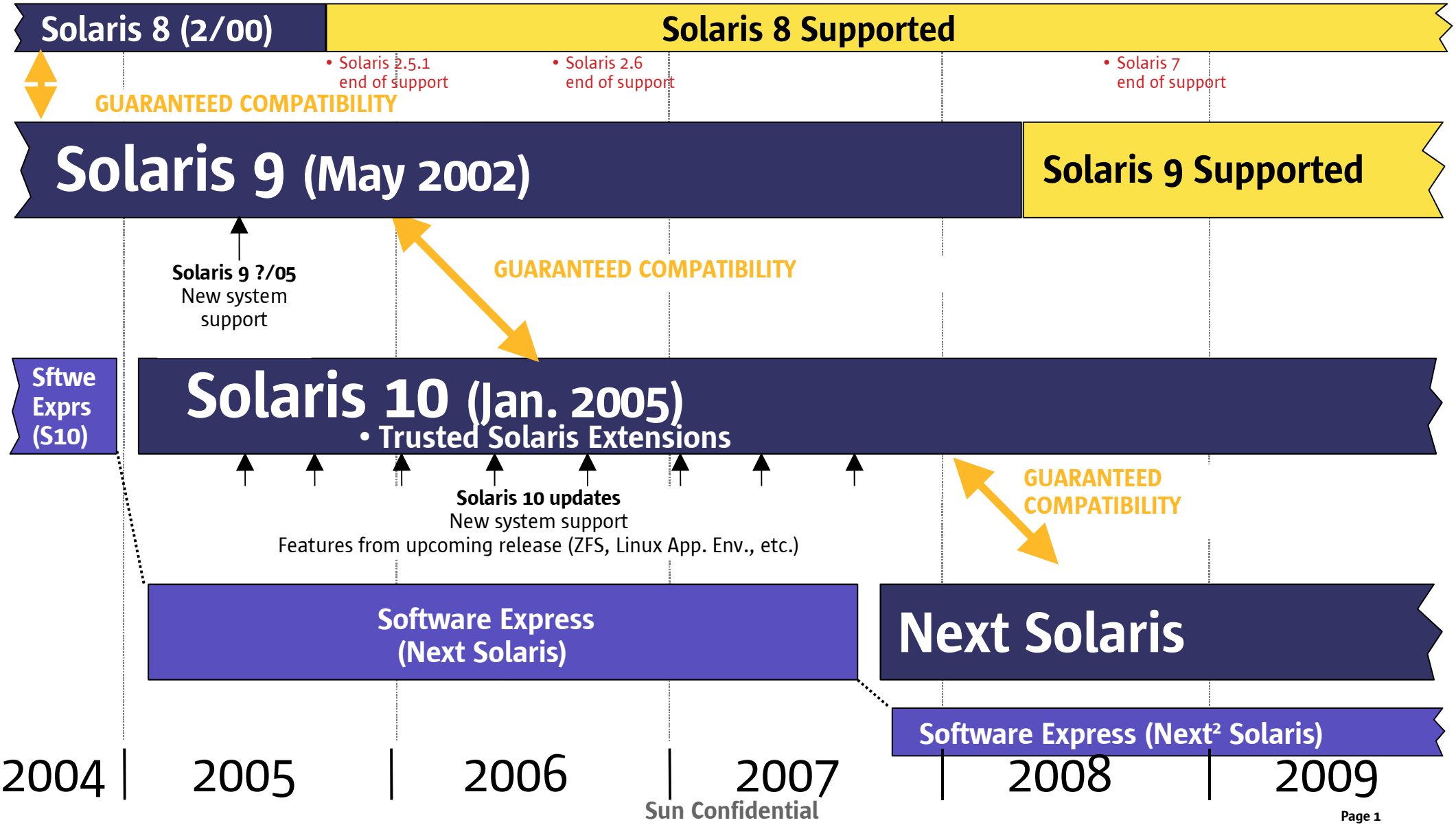
Solaris 10—Design Principles

- Innovation & integration
  - Bring innovation from within Sun & outside into Solaris and deliver as integrated whole
- Drive down total cost
  - Deliver Solaris, SunPlex, N1, JES products

    Enable massive horizontal and vertical scaling at minimal overall cost
  - Deliver total system virtualization for workload consolidation

sun.com/solaris/lifecycle.html

Solaris Roadmap, Oct. '04 - Sept. '09



Solaris Express

- Monthly snapshot of next Solaris version
- Protected web site
  - Free OS next CD images download
  - Detailed program info/instructions/FAQs
  - Discussion forums
  - Online support (only) with $99 year's subscription
- Community initiative rather than profit center
- Targeting developers, early adopters, community users

Solaris 10 OS Projects

Manageability
- Solaris Containers
  - Management interface
  - Hardware fault isolation
  - Security isolation

Security
- Trusted Solaris extensions
- Intrinsic firewall
- Least privilege

Scalability
- Datapath enhancements
- Network performance
- Small system optimization
- Chip Multi-Threading

Availability
- Self healing
- First-time fix
- Fault management
- Single-node failover
- Dynamic tracing

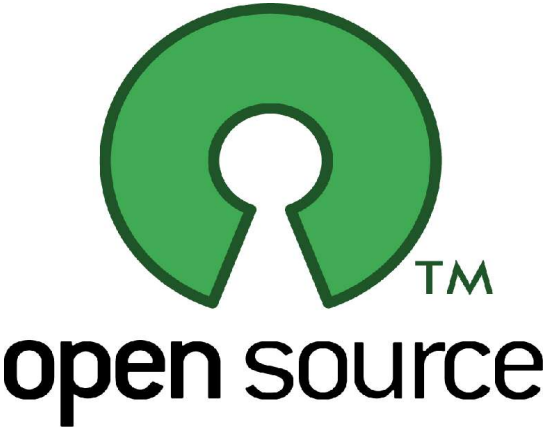
Applications
- Extended compatibility verification
- Enhanced Linux compatibility

Linux Compatibility (SOLARIS 10 UPDATE)

Linux Application Environment

- Lets current Linux applications take advantage of Solaris performance, scalability and security—using the same unchanged RPMs
- Next-generation *lxrun* replacement
  - lxrun = user-space emulation layer
  - Linux Application Environment = direct kernel support of Linux system calls

Platform Choice



opensolaris™

Solaris Source Code

OSI Approved License

Buildable Source
Q2CY2005

dtrace:

- Improved system observability
  - better debugging and performance tuning
- Dynamically instrument and trace the Solaris kernel
  - continuous “black box” recording
- Examine “live” systems and crash dumps
  - reduce time to resolutions

Ideal Software Observability

What is causing the cross calls?

The X servers.

What are the X servers doing to cause the cross calls?

They're mapping and unmapping "/dev/null".

Why are they doing that?

They're creating and destroying Pixmaps.

Who is asking them to do that?

Several instances of a stock ticker application.

How often is each stock ticker making this request?

100 times per second.

Why is the application doing that?

It was written by 10,000 monkeys at 10,000 keyboards.

Actual ~~Ideal~~ Software Observability

What is causing the cross calls?

The X servers.

What are the X servers doing to cause the cross calls?

They're mapping and unmapping "/dev/null".

Why are they doing that?

They're creating and destroying Pixmaps.

Who is asking them to do that?

Several instances of a stock ticker application.

How often is each stock ticker making this request?

100 times per second.

Why is the application doing that?

It was written by 10,000 monkeys at 10,000 keyboards.

BugID 4770261

Black Box Dynamic Tracing

- Software observability in *production*
  - *Zero probe effect* when not specifically enabled
    - Concise answers to arbitrary questions
  - Allow users to:
    - dynamically enable thousands of probes
    - dynamically associate predicates and actions with probes
    - dynamically manage trace buffers and probe overhead
    - examine trace data from a live system or crash dump

Introducing DTrace

- Dynamic tracing framework introduced in Solaris Express 9/03
- Available on stock systems – typical system has more than 35,000 probes
- Dynamically interpreted language allows for arbitrary actions and predicates
- Can instrument at both user-level and kernel-level

Introducing DTrace, cont.

- Powerful data management primitives eliminate need for most postprocessing
- Unwanted data is pruned as close to the source as possible
- Mechanism to trace during boot
- Mechanism to retrieve all data from a kernel crash dump
- Much more...

Probes

- A *probe* is a point of instrumentation
- A probe is made available by a *provider*
- Each probe identifies the *module* and *function* that it instruments
- Each probe has a *name*
- These four attributes define a tuple that uniquely identifies each probe
- Each probe is assigned an integer identifier

Listing probes

- Probes can be listed with the “-l” option to dtrace (1M)
- For each probe, provider, module, function and name are displayed

```
# dtrace -l
  ID   PROVIDER            MODULE                  FUNCTION     NAME
   1     dtrace                                                 BEGIN
   2     dtrace                                                   END
   3     dtrace                                                 ERROR
   4   fasttrap                                   fasttrap fasttrap
  ...
34611        fbt               zmod                 z_strerror return
34612        fbt               zmod               z_uncompress entry
34613        fbt               zmod              z_uncompress return
```

D Scripts, cont.

- For example, a script to trace the executable name upon entry of each system call:

```
#!/usr/sbin/dtrace -s


syscall:::entry
{
        trace(execname);
}
```

Predicates, cont.

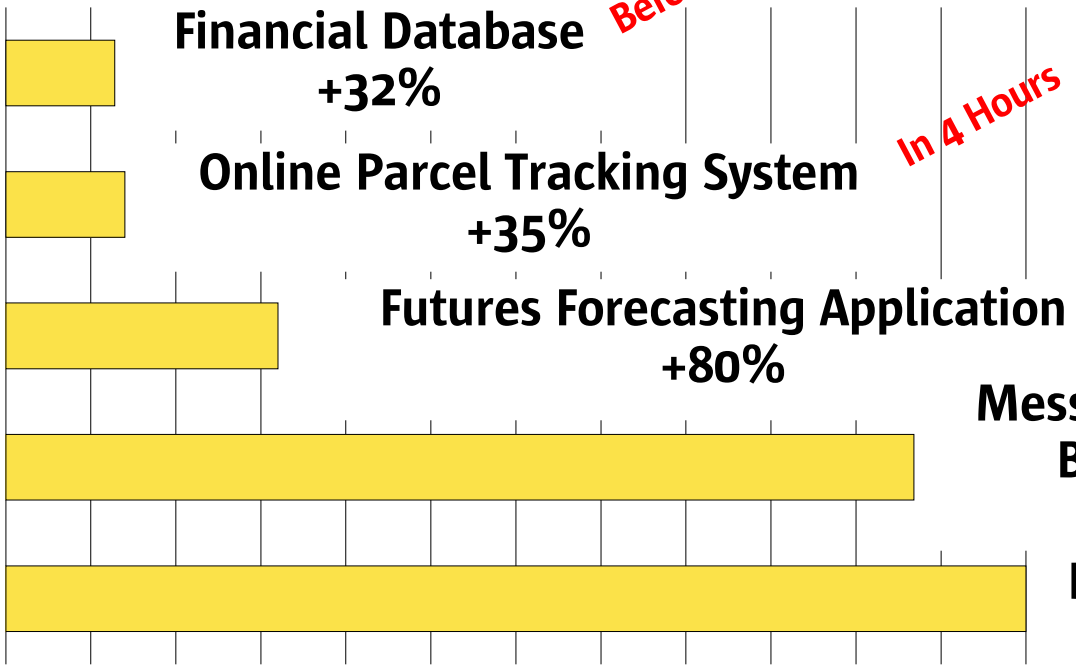
- For example, tracing the pid of every process named “date” that performs any system call:

```
#!/usr/sbin/dtrace -s


syscall::entry
/execname == “date”/
{
        trace(pid);
}
```

Latest DTrace Wins

Simple Tool, Extreme Performance



Deploy

Maintain

Solaris 2.5
Solaris 2.6
Solaris 7
Solaris 8
Solaris 9
Solaris 10

Extreme Performance

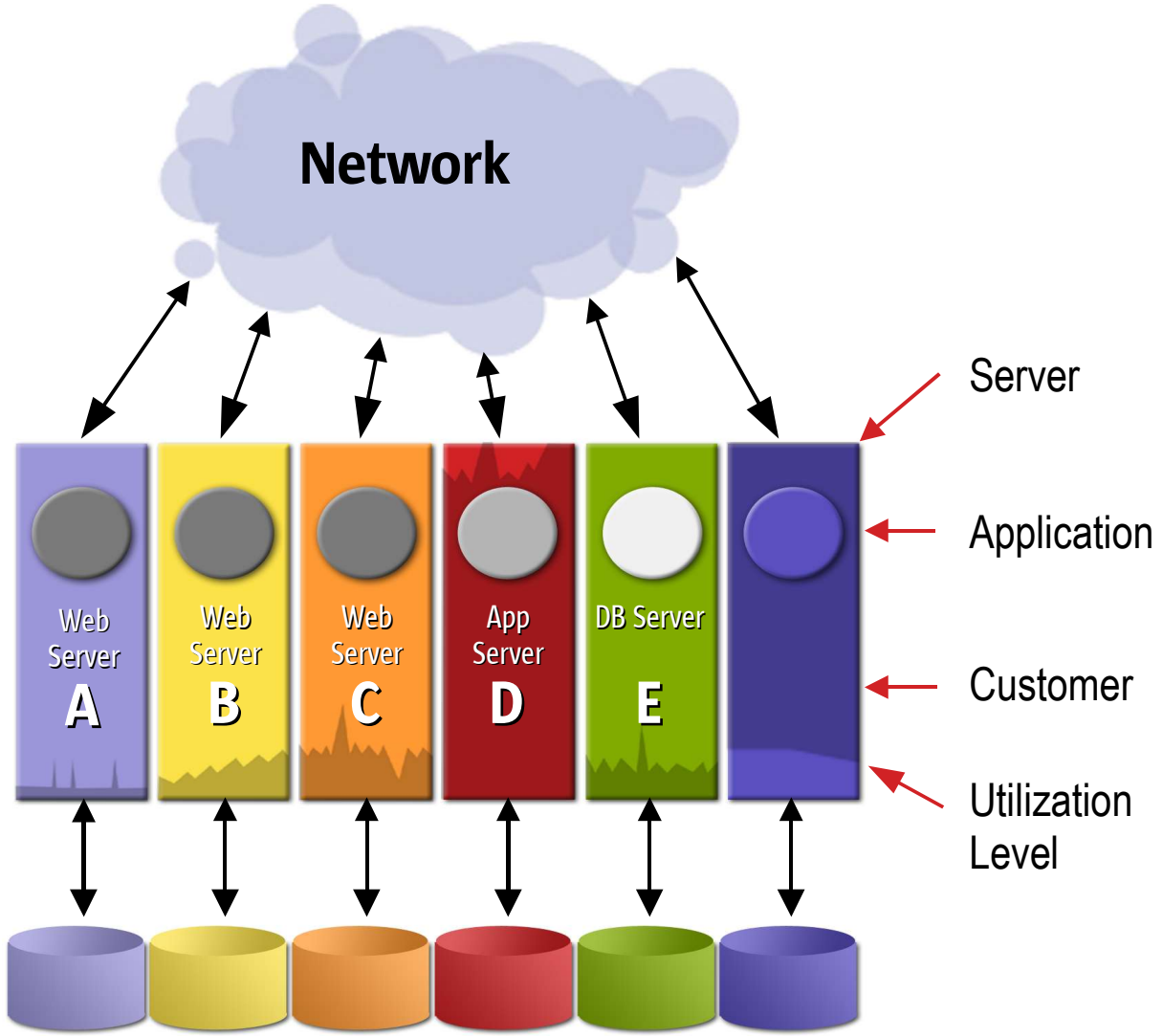
demo

Introduction to zones/containers

- Customers are interested in improving the utilization of their computing resources.
- One method for doing this is via server consolidation.
- At the same time, customers want to be able to partition and isolate various workloads on the consolidated server.

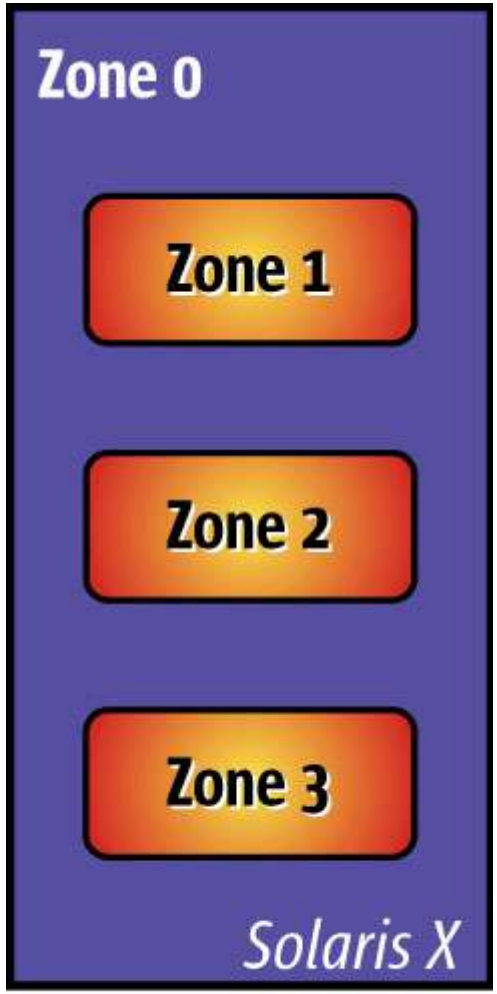
Traditional Resource Management

- One application per server
- One or more servers per customer
- Every server sized for peak workload
- Low average utilization rates

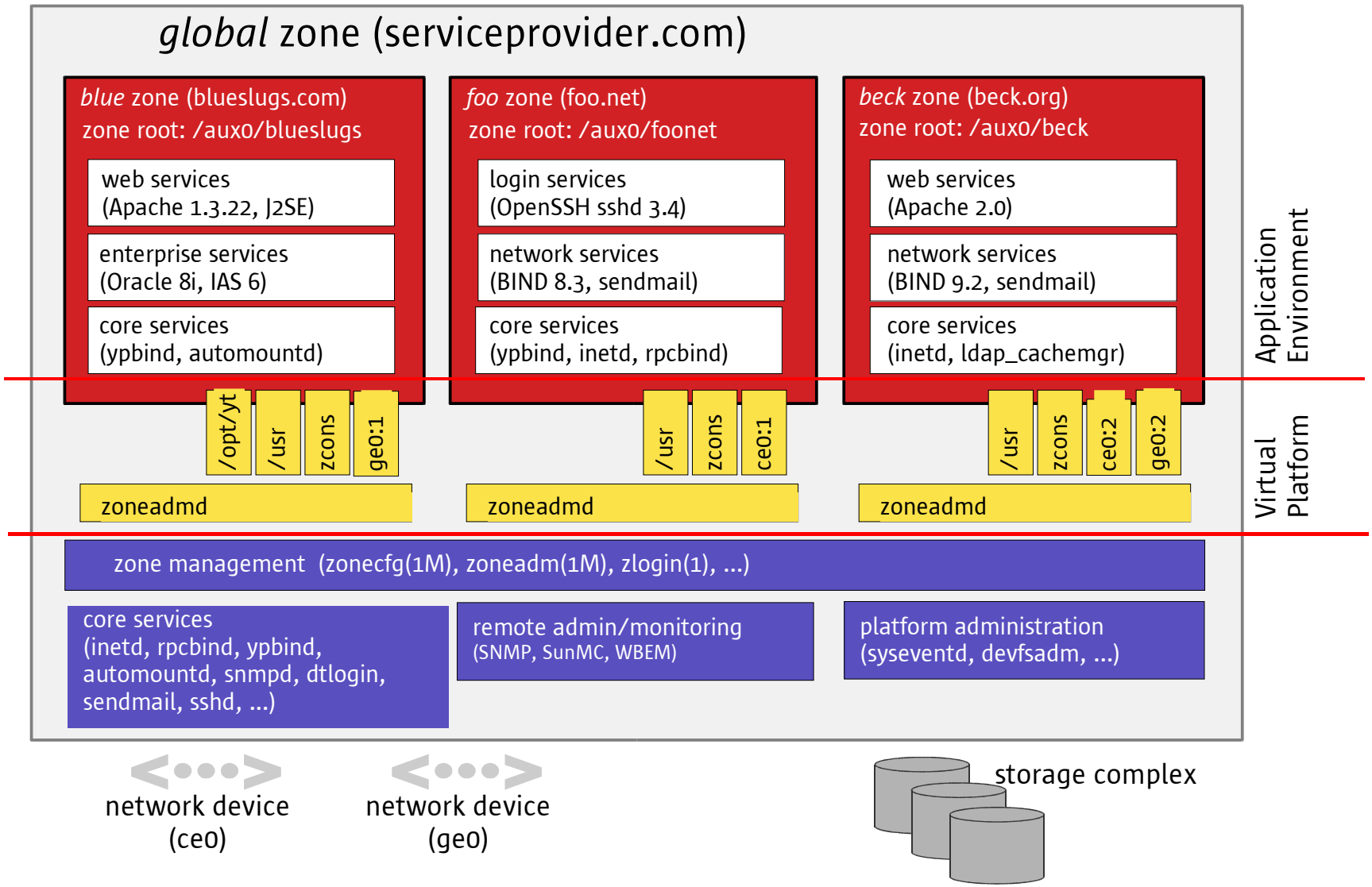


Zones

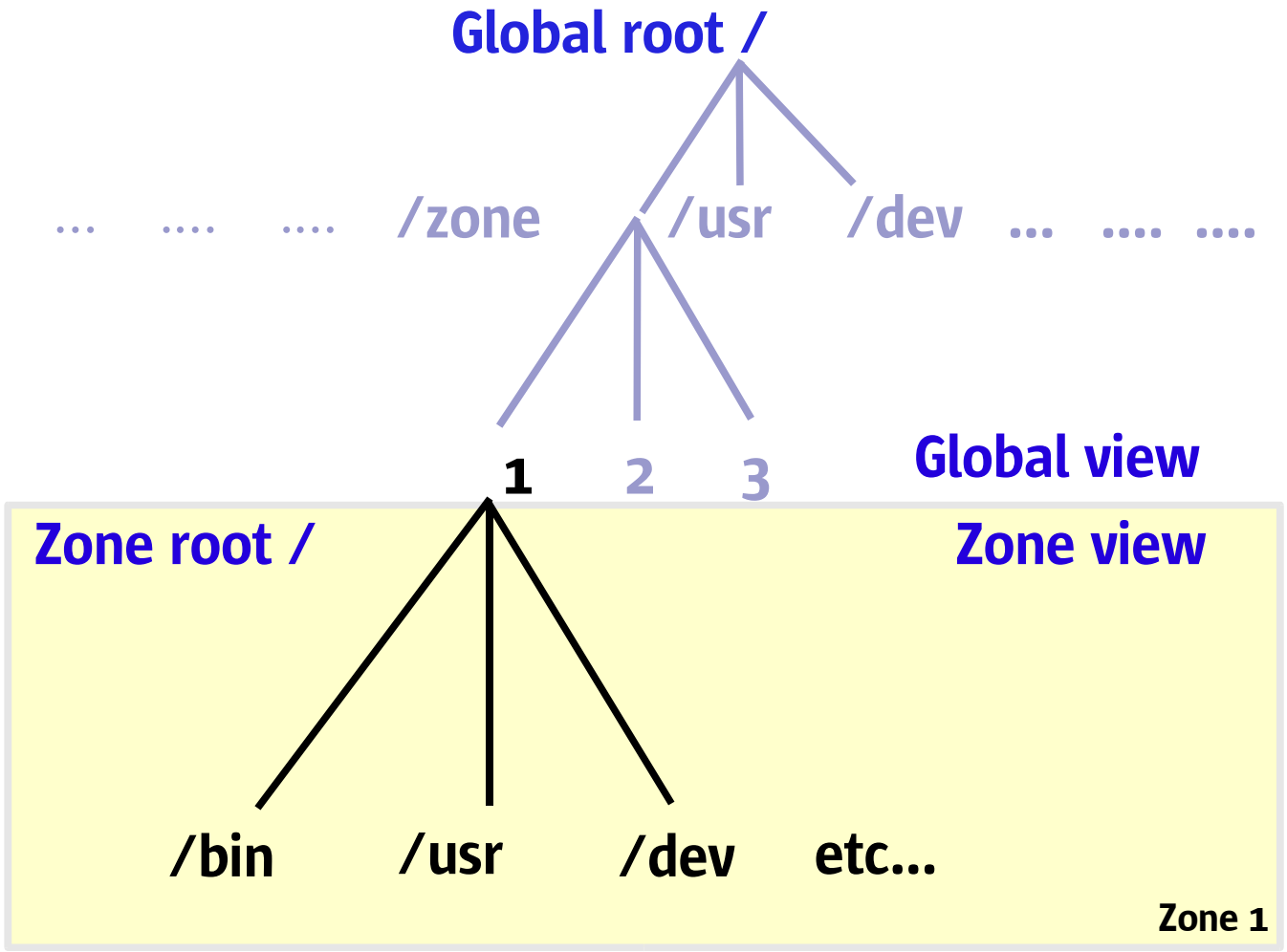
- Allow one or more processes to run in isolation from other system activities
- Each zone has access to
  - Network interface(s)
  - Storage
  - Solaris
- Very similar concept to BSD Jails
  - Network isolation vs server consolidation



Zones Block Diagram



Zone File Systems



Zone Features

- Virtualisation
  - Complete Solaris environment
    - In appearance...
    - Separate zone “root” passwds
  - Restricted global system state
    - kmem, lockstat, trapstat, cpc, ksyms
  - Hides
    - Physical devices
    - IP addresses/hostnames of other zones

Zone Features

- Granularity
  - No dedicated physical devices
  - Multiplexed resources
  - Arbitrary granularity
  - 100+ Zones on a single 1 CPU system
    - Throttle: disk space for unique zone files
- Transparency
  - Standard Solaris interfaces (SysV IPC shared memory segment)
  - ps -ef shows only current zone

Zone Features

- Security
  - No access to other Zones
  - Restricted root access
  - Functions not allowed include
    - Reboot or shutdown of entire system
    - Kernel memory through /dev/kmem
    - No access to objects in other zones
- Isolation
  - FS restriction similar to chroot
  - Shared network port (can't view other traffic)

Zone Features

- Compatibility
  - Global Zone will run apps without modification
  - Local Zones same, unless:
    - Load custom kernel modules
    - Use physical network interfaces
    - Tested apps include iAS, iDS, Apache, Oracle, sendmail, DNS
- Resource Management
  - Controlled by Global Zone
  - Local Zones cannot
    - Assign processes to RT scheduling class
    - Create processor sets or lock down memory

Creating a zone

```
global# zonecfg -z zone1
zone1: No such zone configured
Use 'create' to begin configuring a new zone.
zonecfg:zone1> create
```

Setting's for the zone

```
zonecfg:zone1> set zonepath=/zoneroots/zone1
zonecfg:zone1> set autoboot=true
zonecfg:zone1> add net
zonecfg:zone1:net> set address=192.9.200.67
zonecfg:zone1:net> set physical=hme0
zonecfg:zone1:net> end
zonecfg:zone1> ^D
```

```
#zoneadm list -c
```

Installing the zone

```
global# zoneadm -z zone1 install
Constructing zone at /zoneroot/zone1/root
Creating dev directories
Creating dev links
Copying packages and creating contents file
Copying files and directories
Setting up /etc/motd
Setting up /etc/inittab
Setting up /etc/vfstab
Setting up /var/yp/aliases
Configuring files
```

boot the zone

global# zoneadm -z zone1 boot

- Took about 30 seconds for first boot on Ultra10.

- global# zlogin -C zone1
- [Connected to zone 'mydesktop' console]
- <Run through sysid tools as usual to do initial customization>

demo

Example: zones and fs

```
#zonecfg -z name
zonecfg:zone1> add fs
zonecfg:my-zone:fs> set dir=/opt/local
zonecfg:my-zone:fs> set special=/local
zonecfg:my-zone:fs> set type=lofs
zonecfg:my-zone:fs>end
zonecfg:zone1> verify
zonecfg:zone1> commit
zonecfg:zone1> ^D
```

this will mount the /local directory from the global to a mount point of /opt/local in the zone

Example: RM+zones

```
#zonecfg -z name
zonecfg:zone1> add rctl
zonecfg:zone1:rctl> set name=zone.cpu-shares
zonecfg:zone1:rctl> add value \ (priv=privileged,limit=10,action=none)
zonecfg:zone1:rctl> end
zonecfg:zone1> verify
zonecfg:zone1> commit
zonecfg:zone1> ^D

#prctl -n zone.cpu-shares -r -v 25 -i zone zonename
```

Example: zones and fs

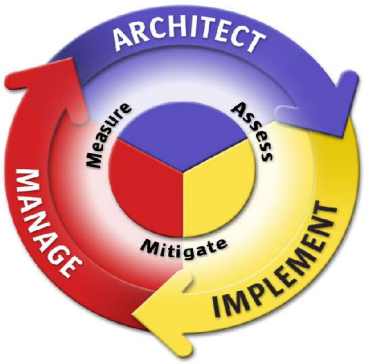
```
#zonecfg -z name
zonecfg:my-zone> add device
zonecfg:my-zone:fs> set match=/dev/dsk/c0t0d0s0
zonecfg:zone1> verify
zonecfg:zone1> commit
zonecfg:zone1> ^D
```

this will give that zone full permission to the device

Service Management Facility

Mission

- Supply a mechanism to formalize relationships between services
- Provide a unified repository for configuration of service startup behavior
- Allow Solaris to start and restart services automatically over the lifetime of a Solaris instance

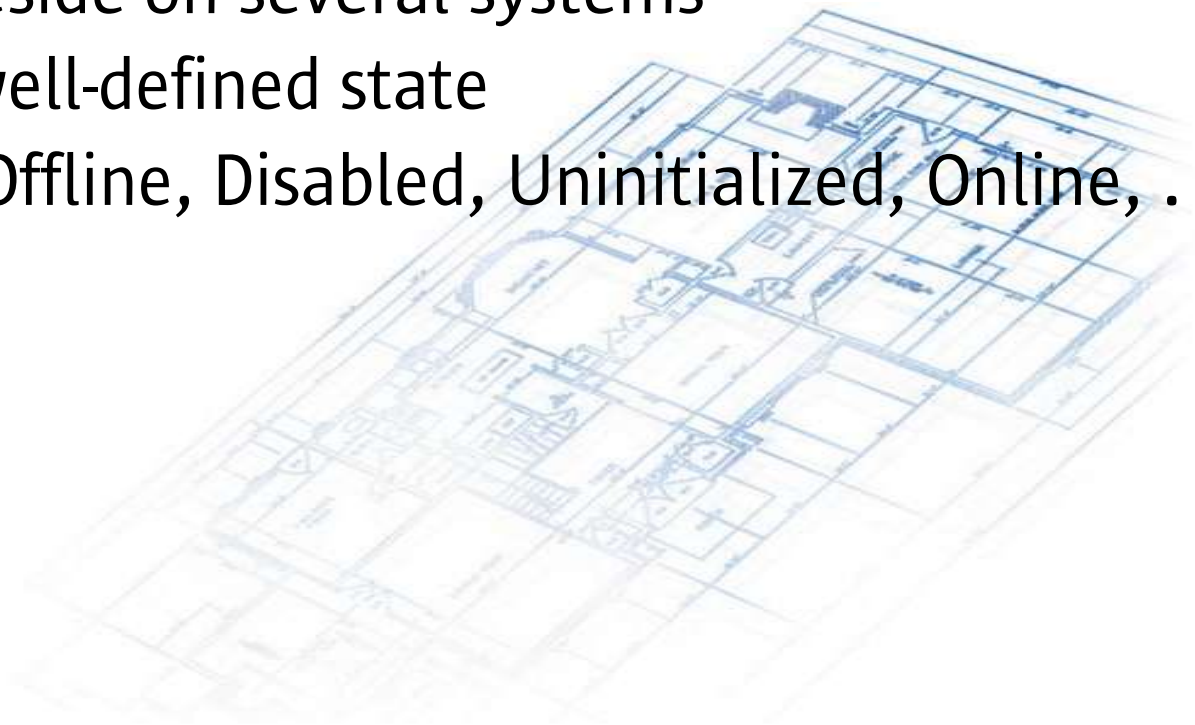


Service Management Facility

Services

Service Definition:

- Abstract description of a long-lived software object
- Object that may reside on several systems
- Application with well-defined state
  - Maintenance, Offline, Disabled, Uninitialized, Online, . . .



Service Management Facility

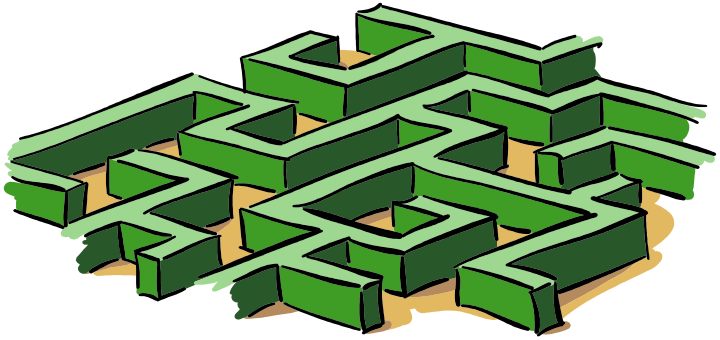
New Daemons & Commands

svc.configd(1M)

- repository cache daemon

svc.startd(1M)

- master restarter daemon



svcs(1) – lists all available services with states

svcadm(1M) – sets service states

svccfg(1M) – imports/exports service descriptions

svcprop(1) – retrieves properties from the repository

How are Services started today ?

- rc scripts in /etc/init.d
  - Long time running or one time initializations
- inetd as defined by inetd.conf
  - Short-lived to provide transient network functions
- /etc/inittab
  - Restartable or one time functions

Problem statement – Why?

- System and application services have all been administrated in different ways adding confusion and lack of organization of services/daemons.
- Dependencies are not unified and often unknown
- Services are administrated through a number of techniques
- N1 vision transportability of services is required for long-term goals
- A new system is required

Solution – SMF

- All services now have a common framework
- Boot methods run in parallel
- Command framework for information and dependencies
- Recoverable database
- All services have the same interface

SMF Identifiers

FMRI – Fault Management Resource ID

svc://localhost/network/login:rlogin

SMF Identifiers

aliases and examples

svc://localhost/network/login:rlogin

svc:/network/login:rlogin

network/login:rlogin

rlogin

svc://localhost/system/system-log:default

svc:/system/system-log:default

SMF Identifiers

functional categories

- Application – general application
- Device – useful for dependencies
- Milestone – similar to SVR4 run levels
- Network – inetd converted services
- Platform – platform specific services
- System – platform independent system services
- Site – reserved for a future use

Service States

online – The service instance is enabled and has successfully started.

offline – The service instance is enabled, but the service is not yet running or available to run.

disabled – The service instance is not enabled and is not running.

maintenance – The service instance has encountered an error that must be resolved by the administrator.

Service States

legacy_run – The legacy service is not managed by SMF, but the service can be observed. This state is only used by legacy services.

degraded – The service instance is enabled, but is running at a limited capacity.

uninitialized – This state is the initial state for all services before their configuration has been read.

SMF Manifests

An SMF manifest is an XML file that contains a complete set of properties that are associated with a service or a service instance.

To incorporate information from the manifest into the repository, you must either run svccfg import or allow the service to import the information during a system boot.

See the service_bundle(4) man page for a complete description of the contents of the SMF manifests.

SMF Compatibility

While many standard Solaris services are now managed by SMF, the scripts placed in /etc/rc*.d continue to be executed on run-level transitions.

Most of the /etc/rc*.d scripts that were included in previous Solaris releases have been removed as part of SMF.

The ability to continue to run the remaining scripts, allows for third-party applications to be added without having to convert the services to use SMF.

Boot Process

run levels and milestones

SVR4 Run Level	SMF Milestone
s, S	single-user
2	multi-user
3	multi-user-server

Basic Commands

svcs

Gives detailed views of the service state of all service instances in the service configuration repository

svcadm

Provides the ability to perform common service management tasks, such as enabling, disabling, or restarting service instances

Managing services

How to List the Status of a Service

svcs -l <fmri>

svcs -l ssh

```
fmri         svc:/network/ssh:default
name         Secure Shell
enabled      true
state        online
next_state   none
restarter    svc:/system/svc/restarter:default
contract_id  24
dependency   require_all/restart
file://localhost/etc/ssh/sshd_config (-)
dependency   require_all/none svc:/system/cryptosvc (online)
dependency   require_all/none svc:/network/loopback (online)
dependency   require_all/none svc:/system/filesystem/usr:default
(online)
```

Monitoring services

- How to disable a service
  - Become superuser or assume a role that includes the Service Management Profile..
  - Check the dependents of the service you want to disable

    # svcs -D [fmri]

    # **svcs -D network/login:rlogin**

    # **svcadm disable network/login:rlogin**

    # **svcs network/login:rlogin**

```
STATE    STIME  FMRI
disabled   11:17:24  svc:/network/login:rlogin
```

Monitoring services

- How to enable a service

 # svcadm enable network/login:rlogin

 # svcs -l network/login:rlogin

```
fmri         svc:/network/login:rlogin
name         The remote login service.
enabled      true
state        online
next_state   none
restarter    svc:/network/inetd:default
```

Evolution

Basic

ASR – Automatic System Recovery

- Configures around failed components
- Restores system to operation asap
- Minimizes need for manual intervention

"I get knocked down, but I get up again."

- Chumbawamba

Evolution

Improved

Auto Diagnosis and Recovery

Big Iron crash avoidance patch

- Detects & isolates faulty hardware
- Detects & recovers from hung domains
- Automatically deconfigures faulty components

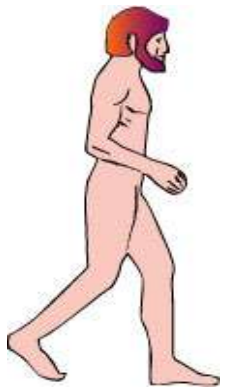
Evolution

Complete

Predictive Self-Healing

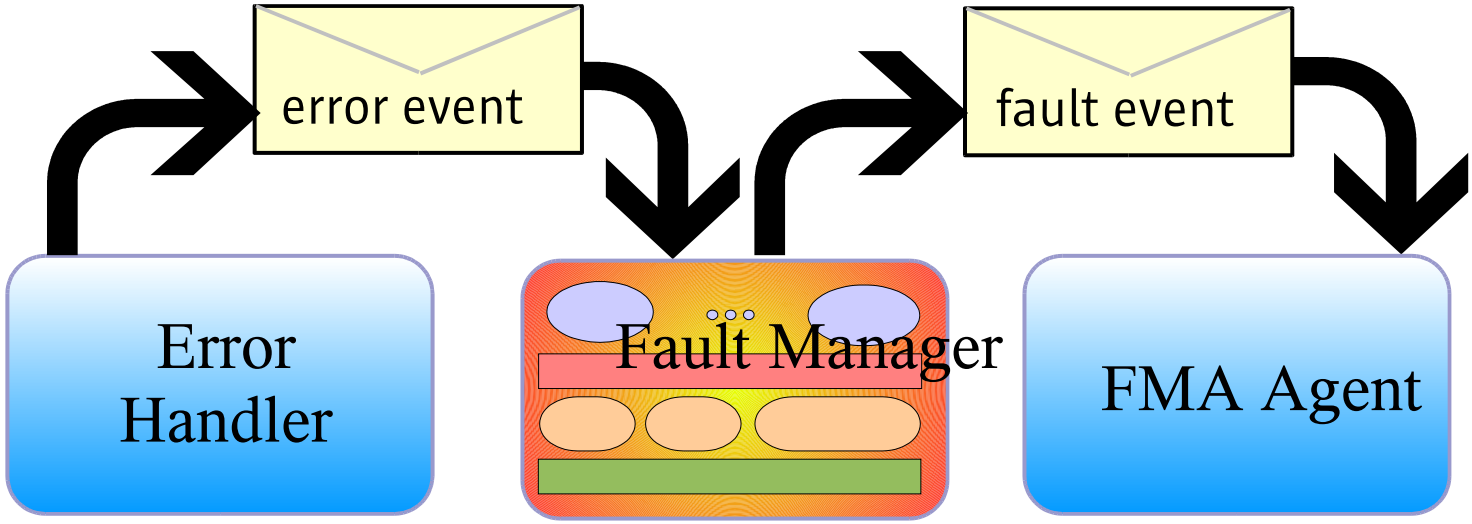
Complete extensible framework

- *Available for the Entire Product Line*
- *Result of 10+ projects from software and hardware product groups, delivered at Solaris 10*



Fault Management Architecture

Fault Management Flow



- error detection
- data gathering
- error handling

- diagnosis
- event recording

- action

Fault Management Architecture

Fault Management Tools

fmadm(1M) - view/manage activities

- View, load, unload, and reset modules
- View list of faulty resources (and more)

fmdump(1M) - view log files

- View time, UUID, and Message ID for fault log
- View time and Private event detail for error log
- Match events by UUID, class, time range

fmstat(1M) - view statistics

- View time and event statistics for all modules
- View Private bean counters for a given module

Fault Management Architecture

Coming Soon...

sun.com/msg/*(message code)*

- Customer web-site will provide latest repair procedures for each diagnosis
- Links to information on latest FMA capabilities, updates, and plans
- No passwords – totally free access

sun.com/msg/SF20000-W84N-KP3A-TF

SUNW-MSG-ID: SF20000-W84N-KP3A-TF; TYPE: Fault, VER: 1, SEVERITY: Minor

AUTO-RESPONSE: Removal of the faulty memory resources has been initiated

Next-Gen Filesystem: Update 2 or 3

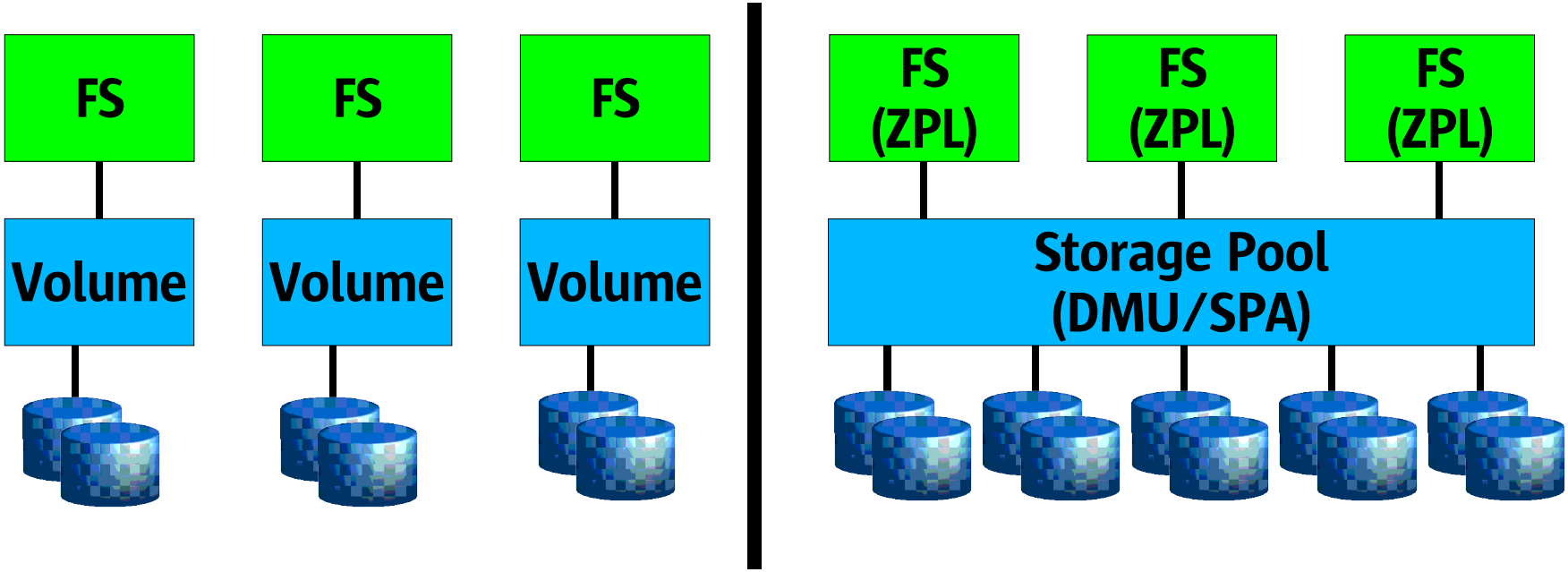
- Existing file systems:
  - No defense against data corruption
  - Lots of Limits: size, # of files, etc.
  - Difficult to manage:
    - fsck, /etc/fstab, partitions, volumes
    - too many things to tune
- Design Principles:
  - End-to-End data integrity
  - Lots of capacity (128-bit)
  - Simple to administer

Next Generation File System

- Many file systems based on old assumptions
  - Enter ZFS
- Scalable, Easy to Manage, Fast, Data Integrity
  - EoM very important considering infrequent use
- Z = “Zettabyte” (128-bit)
  - UFS = 16 TB
  - VxFS = 32 TB
  - QFS = 252 TB
- Customers have Petabytes today, Exabytes not far

Allocating Storage

- File system is meant to be logical grouping for data
  - Today we have to find physical disk to put it on (and fsck it, and edit vfstab etc.)



Administration

- Task: given two disks, create mirrored file systems for Ann, Bob and Sue. Later, add more space

Traditional FS Management

```
# format
... (long interactive session omitted)

# metadb -a -f disk1:slice0 disk2:slice0

# metainit d10 1 1 disk1:slice1
d10: Concat/Stripe is setup
# metainit d11 1 1 disk2:slice1
d11: Concat/Stripe is setup
# metainit d20 -m d10
d20: Mirror is setup
# metattach d20 d11
d20: submirror d11 is attached

# metainit d12 1 1 disk1:slice2
d12: Concat/Stripe is setup
# metainit d13 1 1 disk2:slice2
d13: Concat/Stripe is setup
# metainit d21 -m d12
d21: Mirror is setup
# metattach d21 d13
d21: submirror d13 is attached
# metainit d14 1 1 disk1:slice3
d14: Concat/Stripe is setup
# metainit d15 1 1 disk2:slice3
d15: Concat/Stripe is setup
# metainit d22 -m d14
d22: Mirror is setup
# metattach d22 d15
d22: submirror d15 is attached
```

```
# newfs /dev/md/rdsk/d20
newfs: construct a new file system /dev/md/rdsk/d20: (y/n)? y
... (many pages of 'superblock backup' output omitted)
# mount /dev/md/dsk/d20 /export/home/ann
# vi /etc/vfstab  ... while in 'vi', type this exactly:
/dev/md/dsk/d20 /dev/md/rdsk/d20 /export/home/ann ufs 2 yes -

# newfs /dev/md/rdsk/d21
newfs: construct a new file system /dev/md/rdsk/d21: (y/n)? y
... (many pages of 'superblock backup' output omitted)
# mount /dev/md/dsk/d21 /export/home/ann
# vi /etc/vfstab  ... while in 'vi', type this exactly:
/dev/md/dsk/d21 /dev/md/rdsk/d21 /export/home/bob ufs 2 yes -

# newfs /dev/md/rdsk/d22
newfs: construct a new file system /dev/md/rdsk/d22: (y/n)? y
... (many pages of 'superblock backup' output omitted)
# mount /dev/md/dsk/d22 /export/home/sue
# vi /etc/vfstab  ... while in 'vi', type this exactly:
/dev/md/dsk/d22 /dev/md/rdsk/d22 /export/home/sue ufs 2 yes -

# format
... (long interactive session omitted)
# metattach d12 disk3:slice1
d12: component is attached
# metattach d13 disk4:slice1
d13: component is attached
# metattach d21
# growfs -M /export/home/bob /dev/md/rdsk/d21
/dev/md/rdsk/d21:
... (many pages of 'superblock backup' output omitted)
```

End the Suffering

- **Create a storage pool named “home”**
- `# zpool create "home" mirror(disk1,disk2)`

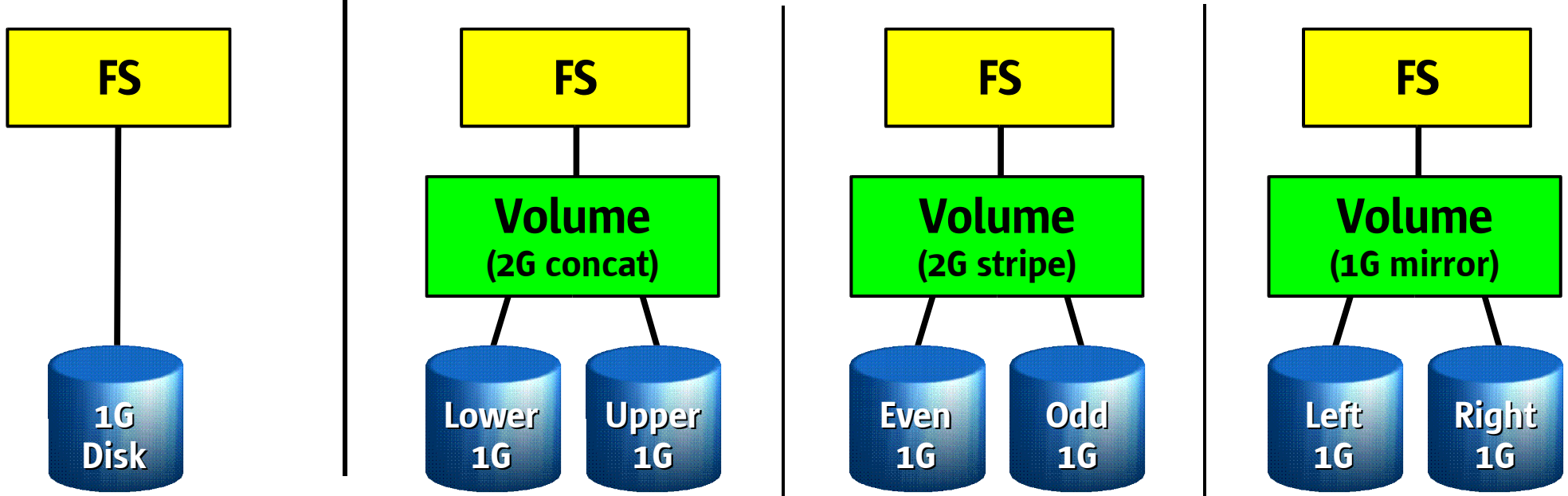
- **Create filesystems “ann”, “bob”, “sue”**
- `# zfs mount -c home/ann /export/home/ann`
- `# zfs mount -c home/bob /export/home/bob`
- `# zfs mount -c home/sue /export/home/sue`

- **Later, add space to the “home” pool**
- `# zpool add "home" mirror(disk3,disk4)`

Background: Why Volumes Exist

In the beginning, each filesystem managed a single disk.

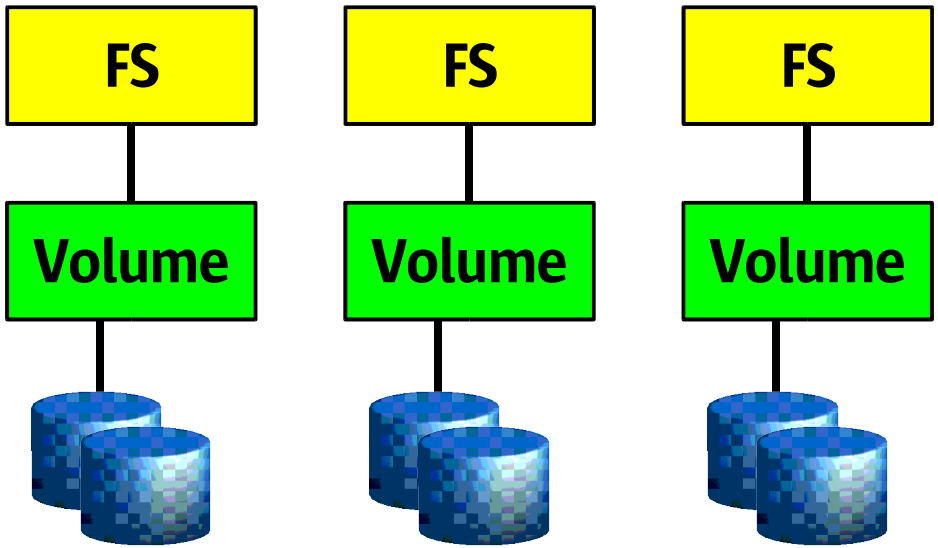
- Customers wanted more space, bandwidth, reliability
  - Rewrite filesystems to handle many disks: hard
  - Insert a little shim ("volume") to cobble disks together: easy
- An industry grew up around the FS/volume model
  - Filesystems, volume managers sold as separate products
  - Inherent problems in FS/volume interface can't be fixed



FS/Volume Model vs. ZFS

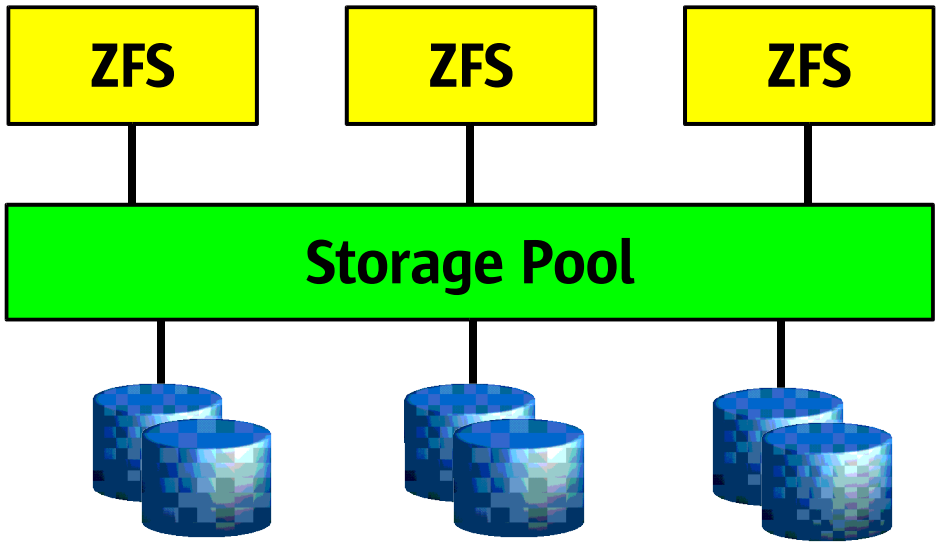
Traditional Volumes

- Abstraction: virtual disk
- Partition/volume for each FS
- Grow/shrink by hand
- Each FS has limited bandwidth
- Storage is fragmented, stranded



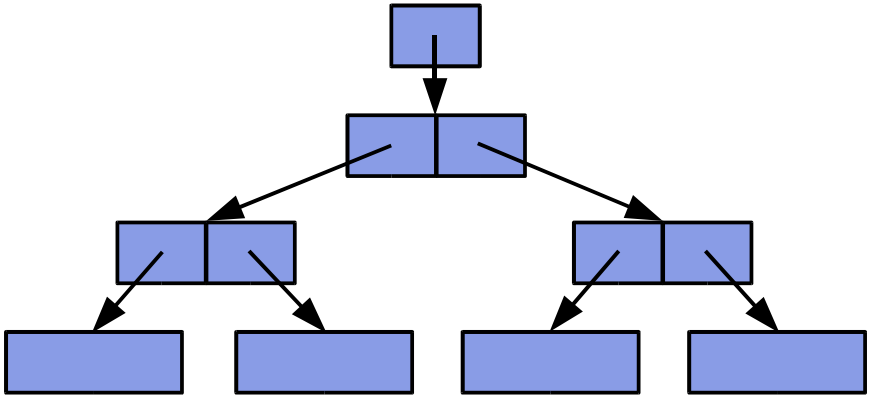
ZFS Pooled Storage

- Abstraction: malloc/free
- No partitions to manage
- Grow/shrink automatically
- All bandwidth always available
- Pool allows space to be shared

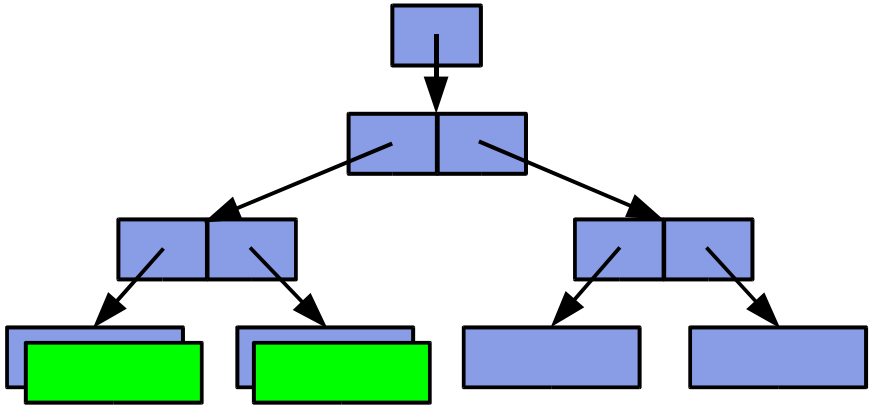


Copy-On-Write Transactions

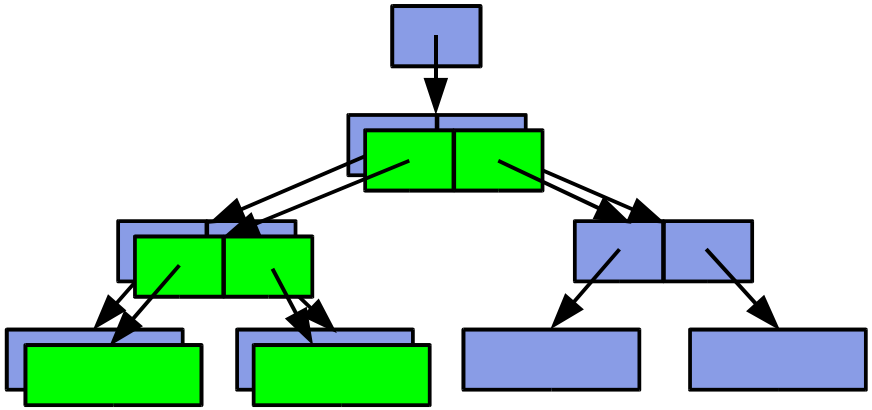
1. Initial block tree



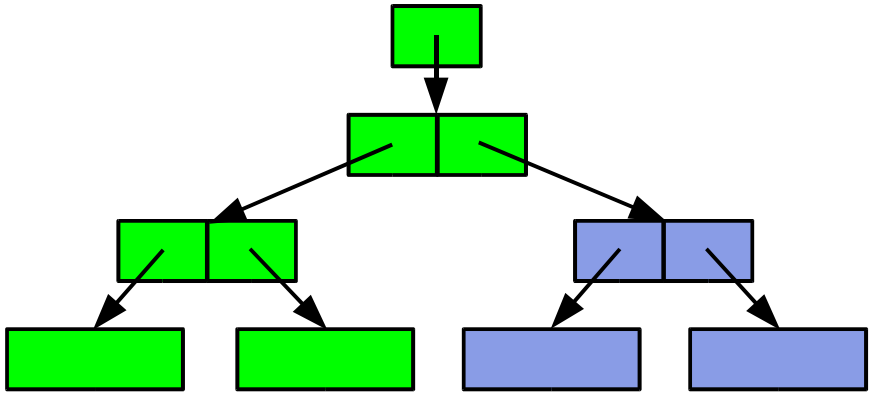
2. COW some blocks



3. COW indirect blocks

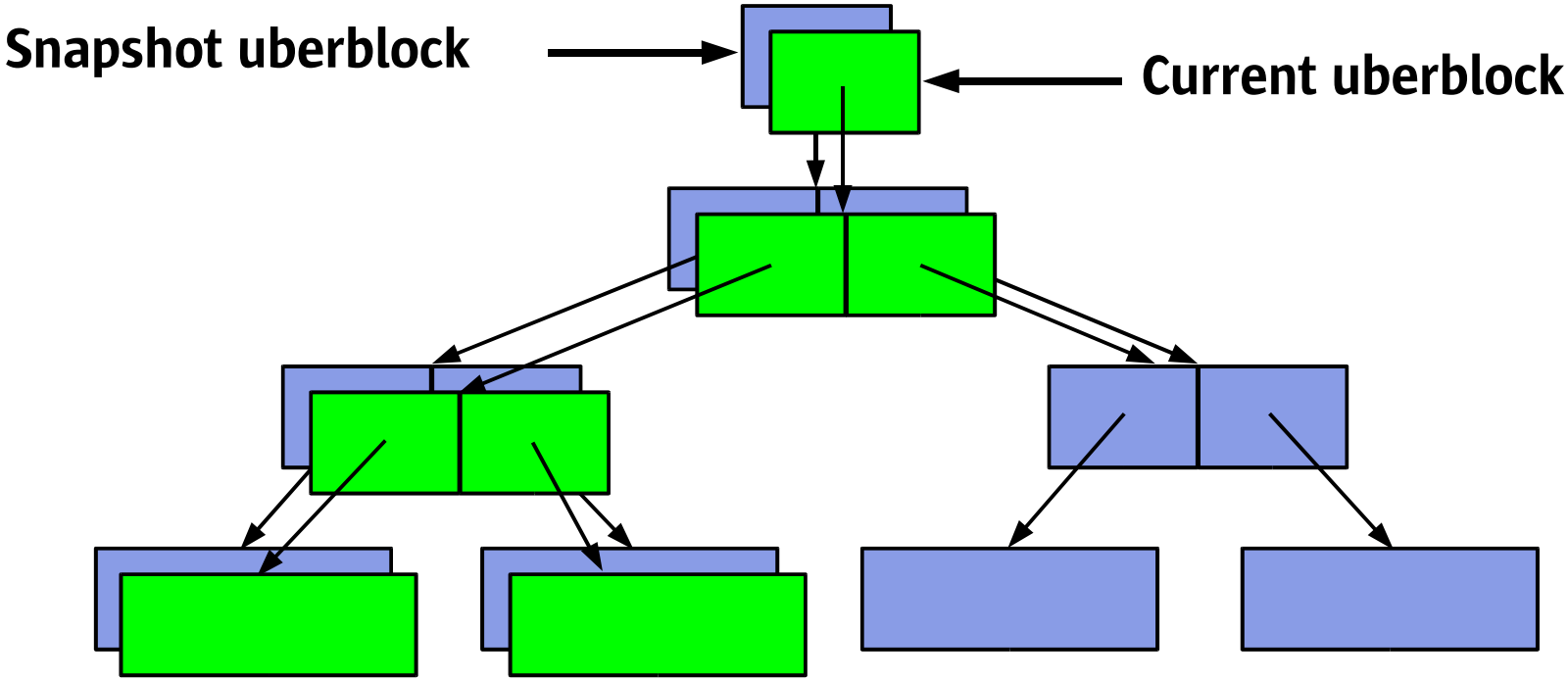


4. Rewrite uberblock (atomic)



Bonus: Constant-Time Snapshots

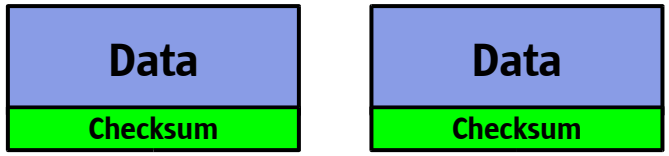
- At end of TX group, don't free COWed blocks
  - Actually cheaper to take a snapshot than not!



End-to-End Checksums

Disk Block Checksums

- Checksum stored with data block
- Any self-consistent block will pass
- Can't even detect stray writes
- Inherent FS/volume interface limitation

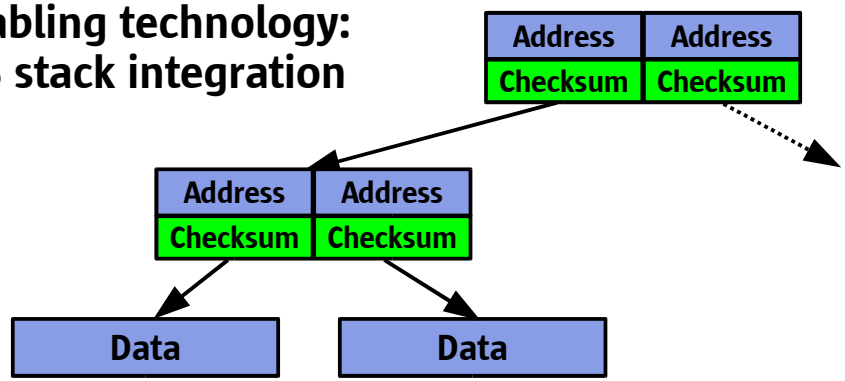


Only validates the media

✔	Bit rot
✗	Phantom writes
✗	Misdirected reads and writes
✗	DMA parity errors
✗	Driver bugs
✗	Accidental overwrite

ZFS Checksum Trees

- Checksum stored in parent block pointer
- Fault isolation between data and checksum
- Entire pool (block tree) is self-validating
- Enabling technology:
  ZFS stack integration

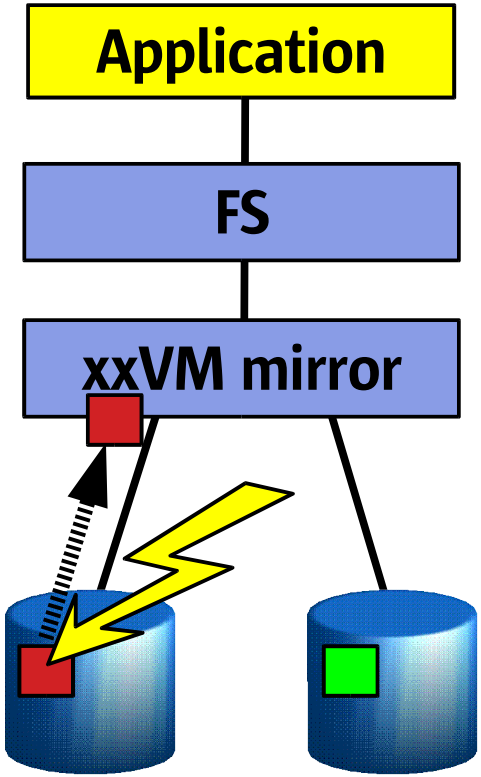


Validates the entire I/O path

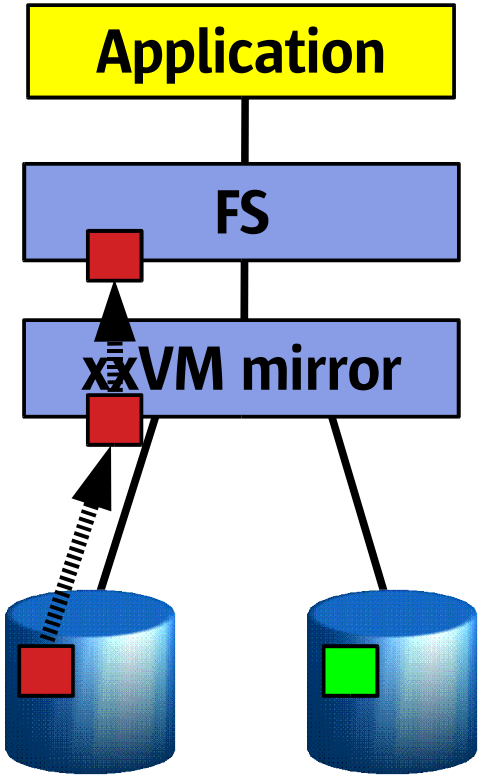
✔	Bit rot
✔	Phantom writes
✔	Misdirected reads and writes
✔	DMA parity errors
✔	Driver bugs
✔	Accidental overwrite

Traditional Mirroring

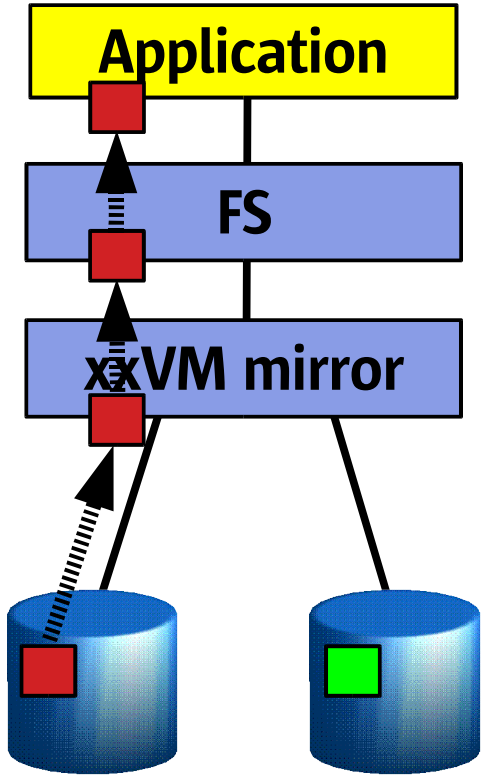
1. Application issues a read.
Mirror reads the first disk, which has a corrupt block.
It can't tell.



2. Volume manager passes bad block up to filesystem.
If it's a metadata block, the filesystem panics. If not...

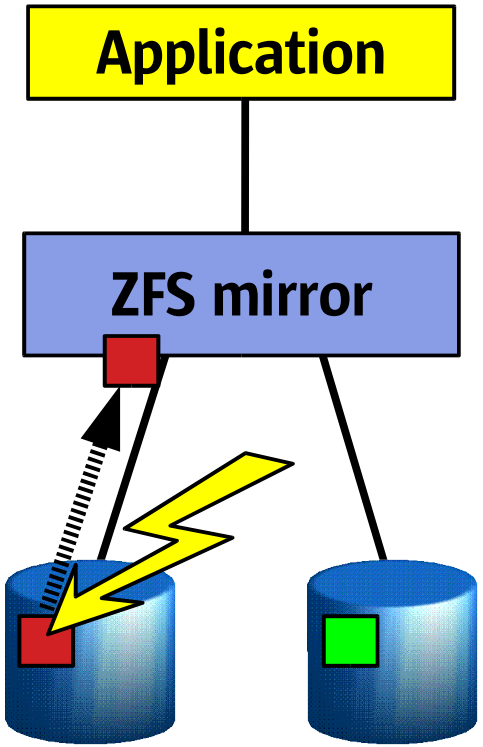


3. Filesystem returns bad data to the application.

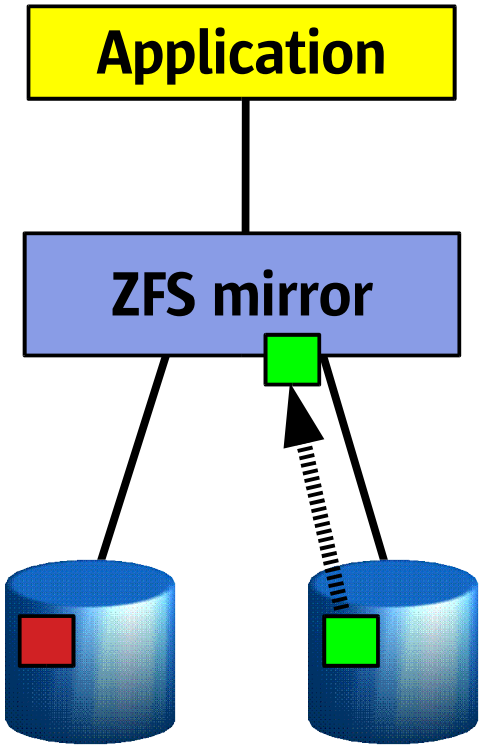


Self-Healing Data in ZFS

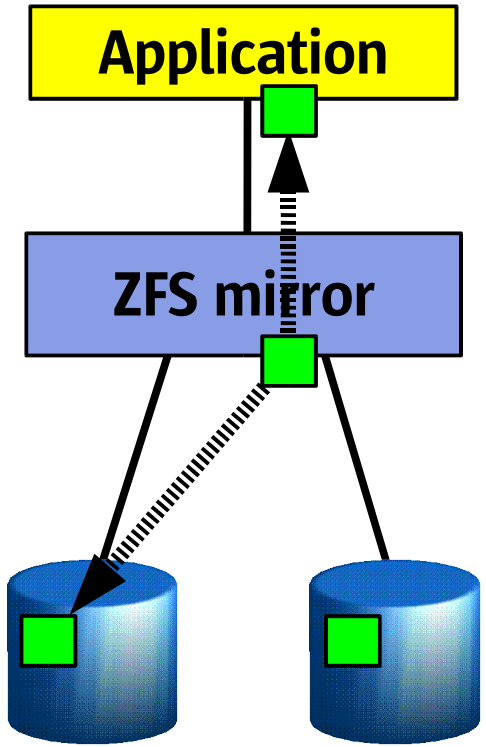
1. Application issues a read. ZFS mirror tries the first disk. Checksum reveals that the block is corrupt on disk.



2. ZFS tries the second disk. Checksum indicates that the block is good.



3. ZFS returns good data to the application and repairs the damaged block.



Self-Healing Data in Action

```
# dd if=/dev/zero of=/dev/dsk/c2d9d0s0 bs=128k ... count=12
# ... read the affected file ... no problem!
# zpool iostat home
                              capacity     operations    bandwidth
vdev   description           used avail   read write   read write   err
   1   mirror(2,3)            305M  136G    167     0 21.0M     0  0/0
   2     /dev/dsk/c2t8d0s0   -----  ----     88     0 11.0M     0  0/0
   3     /dev/dsk/c3t8d0s0   -----  ----     79     0  9.9M     0  0/0
   4   mirror(5,6)            256M  136G    168     0 21.0M     0  12/12
   5     /dev/dsk/c2t9d0s0   -----  ----     86     0 10.8M     0  12/0
   6     /dev/dsk/c3t9d0s0   -----  ----     81     0 10.2M     0  0/0
   7   mirror(8,9)            258M  136G    169     0 21.2M     0  0/0
   8     /dev/dsk/c2t10d0s0  -----  ----     93     0 11.7M     0  0/0
   9     /dev/dsk/c3t10d0s0  -----  ----     76     0 9.45M     0  0/0
  10   mirror(11,12)          257M  136G    176     0 22.1M     0  0/0
  11     /dev/dsk/c2t11d0s0  -----  ----     85     0 10.7M     0  0/0
  12     /dev/dsk/c3t11d0s0  -----  ----     91     0 11.3M     0  0/0
```

ZFS Administration

- Create a storage pool named “home”

```
# zpool create home mirror disk1 disk2
```

- Create filesystems “ann”, “bob”, “sue”

```
# zfs create home/ann /export/home/ann
# zfs create home/bob /export/home/bob
# zfs create home/sue /export/home/sue
```

- Add more space to the “home” pool

```
# zpool add home mirror disk3 disk4
```

ZFS Admin: Cool Features

- Turn on compression for Ann's data

```
# zfs setprop home/ann compression=on
```

- Limit Bob to a quota of 10G

```
# zfs setprop home/bob quota=10g
```

- Guarantee Sue a reservation of 20G

```
# zfs setprop home/sue reservation=20g
```

- Take a snapshot of Ann's filesystem

```
# zfs takesnap home/ann/tuesday-lunch
```

ZFS: Summary

- Simple administration
  - Concisely expresses the user's intent
- Provable data integrity
  - Detects and corrects silent data corruption
- Immense capacity
  - The world's first 128-bit filesystem
- Smokin' performance
  - Already #1 on several benchmarks, sometimes by multiples

...so, what about a GUI?

ZFS Graphical User Interface (Prototype)

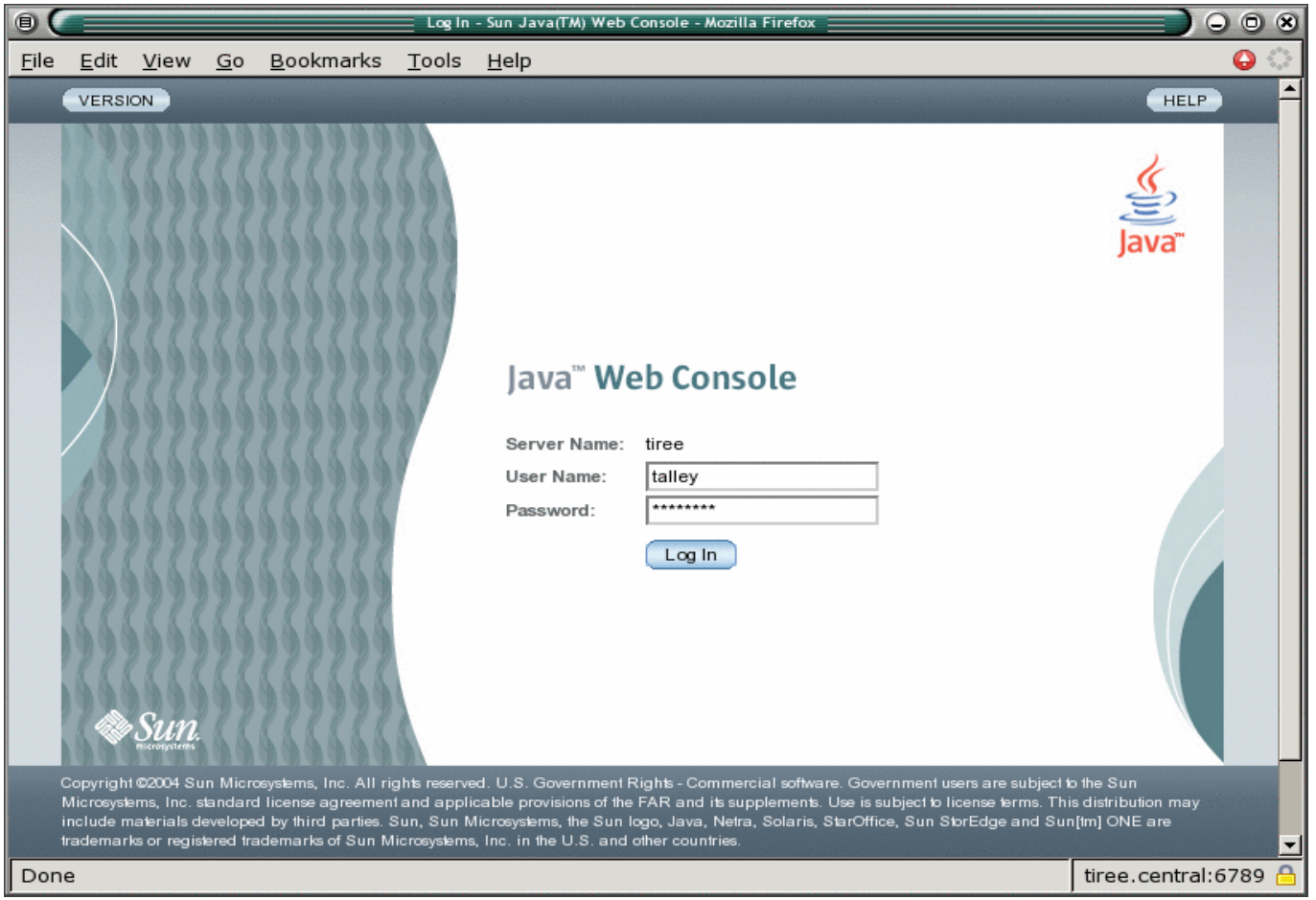
Login to the web console

Drill down to ZFS

View by File Systems

View by Devices

Create a Storage Pool



ZFS Graphical User Interface

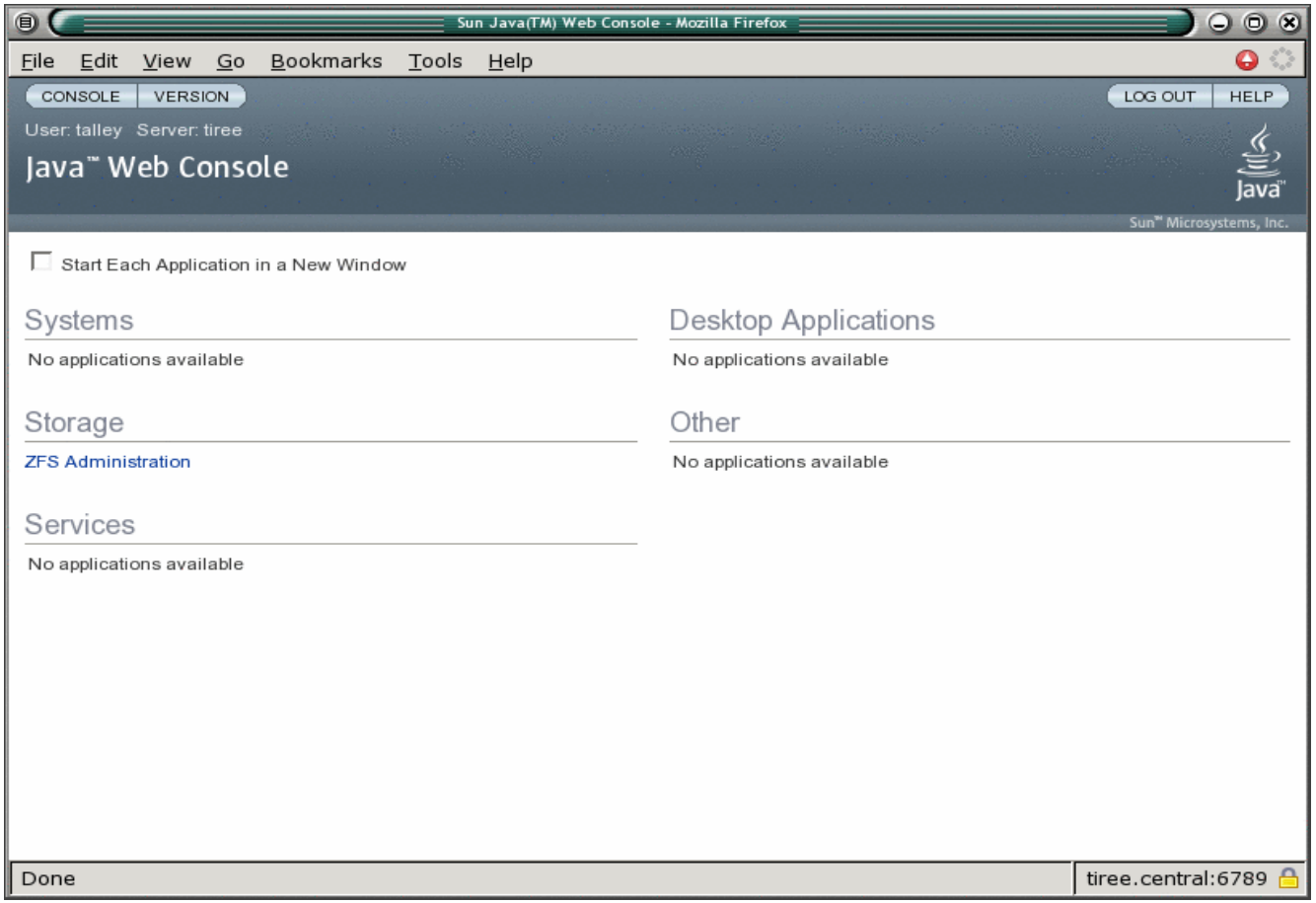
Login to the web console

Drill down to ZFS

View by File Systems

View by Devices

Create a Storage Pool



ZFS Graphical User Interface

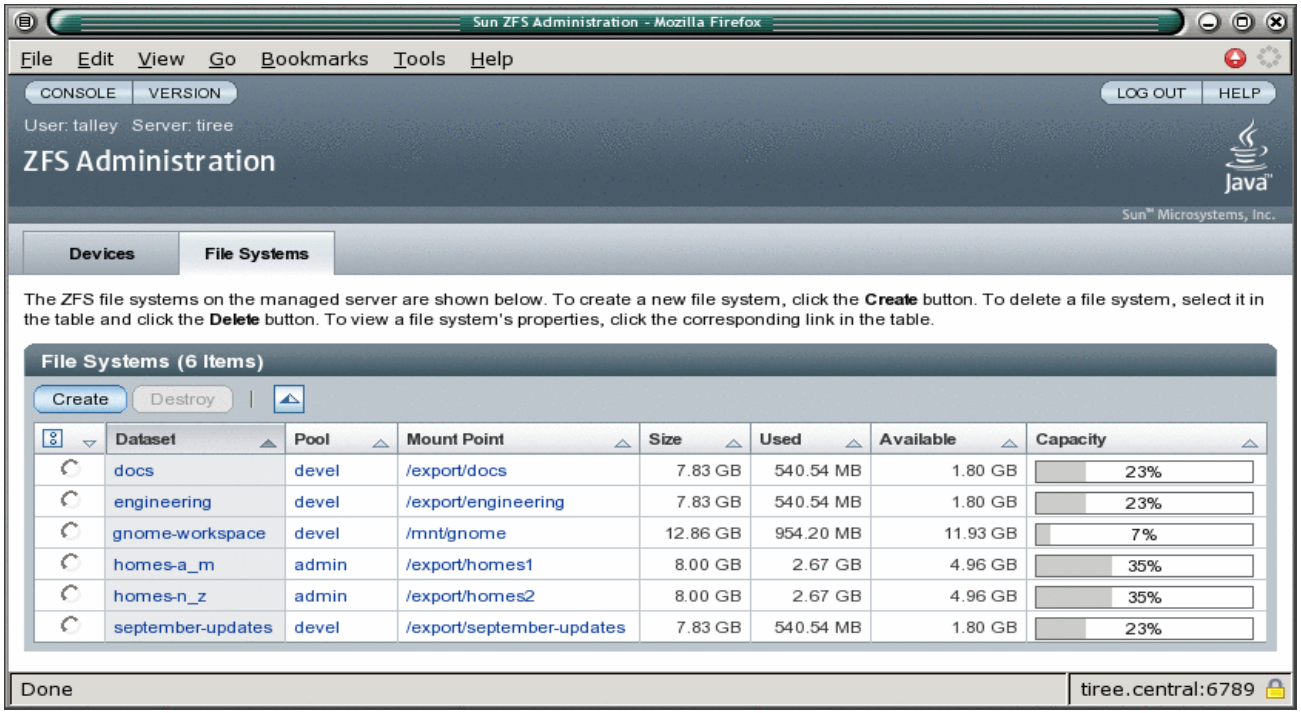
Login to the web console

Drill down to ZFS

View by File Systems

View by Devices

Create a Storage Pool



ZFS Graphical User Interface

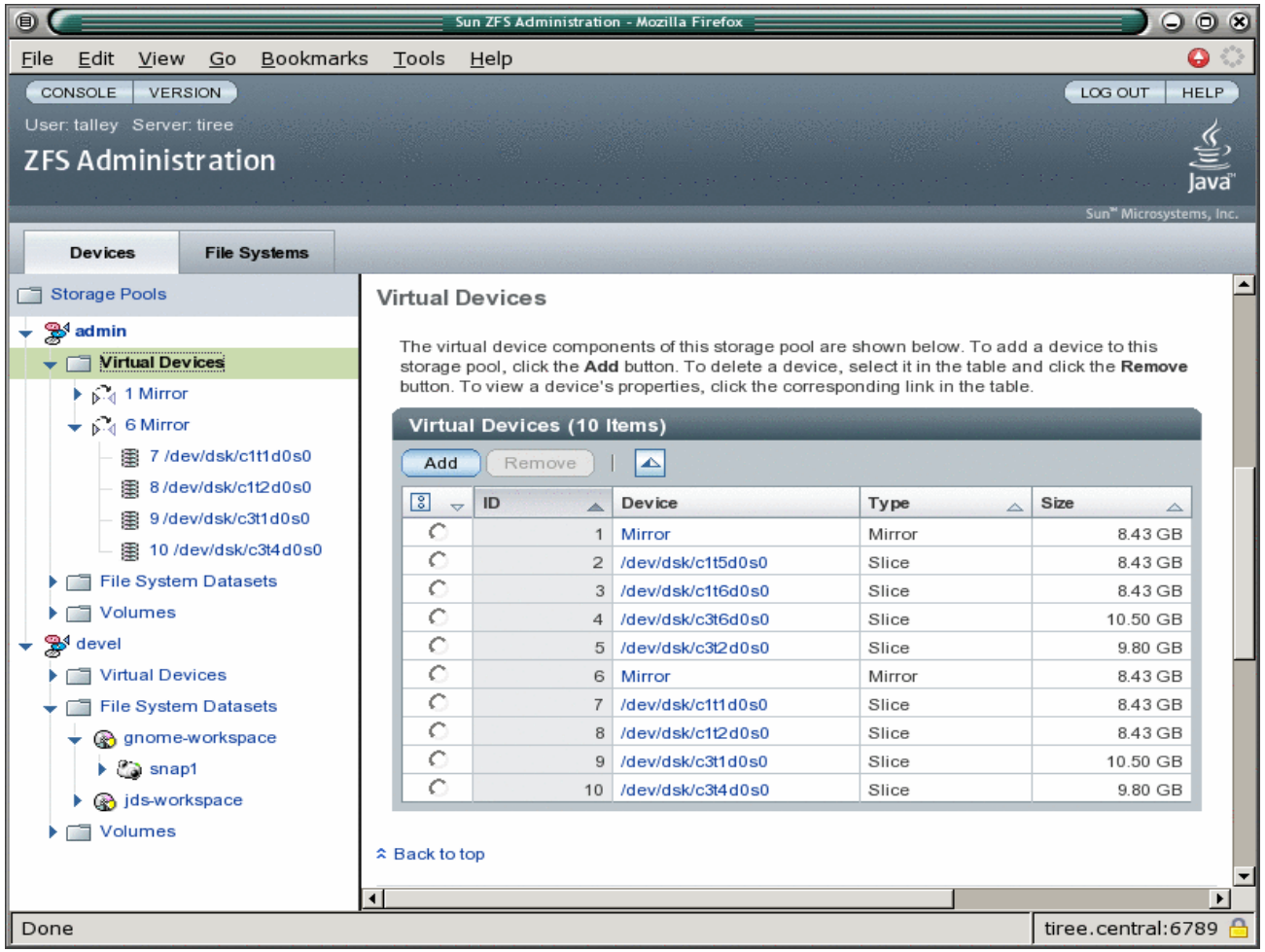
Login to the web console

Drill down to ZFS

View by File Systems

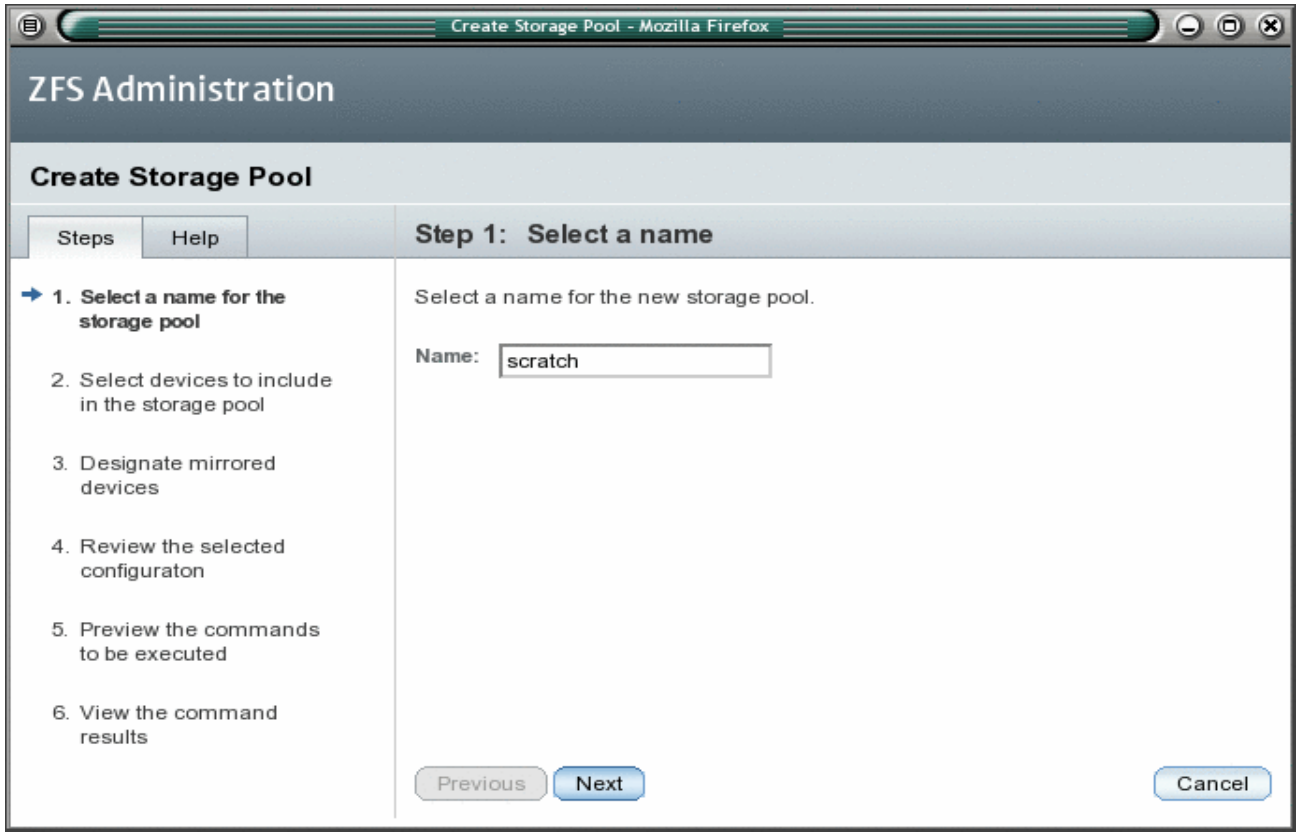
View by Devices

Create a Storage Pool



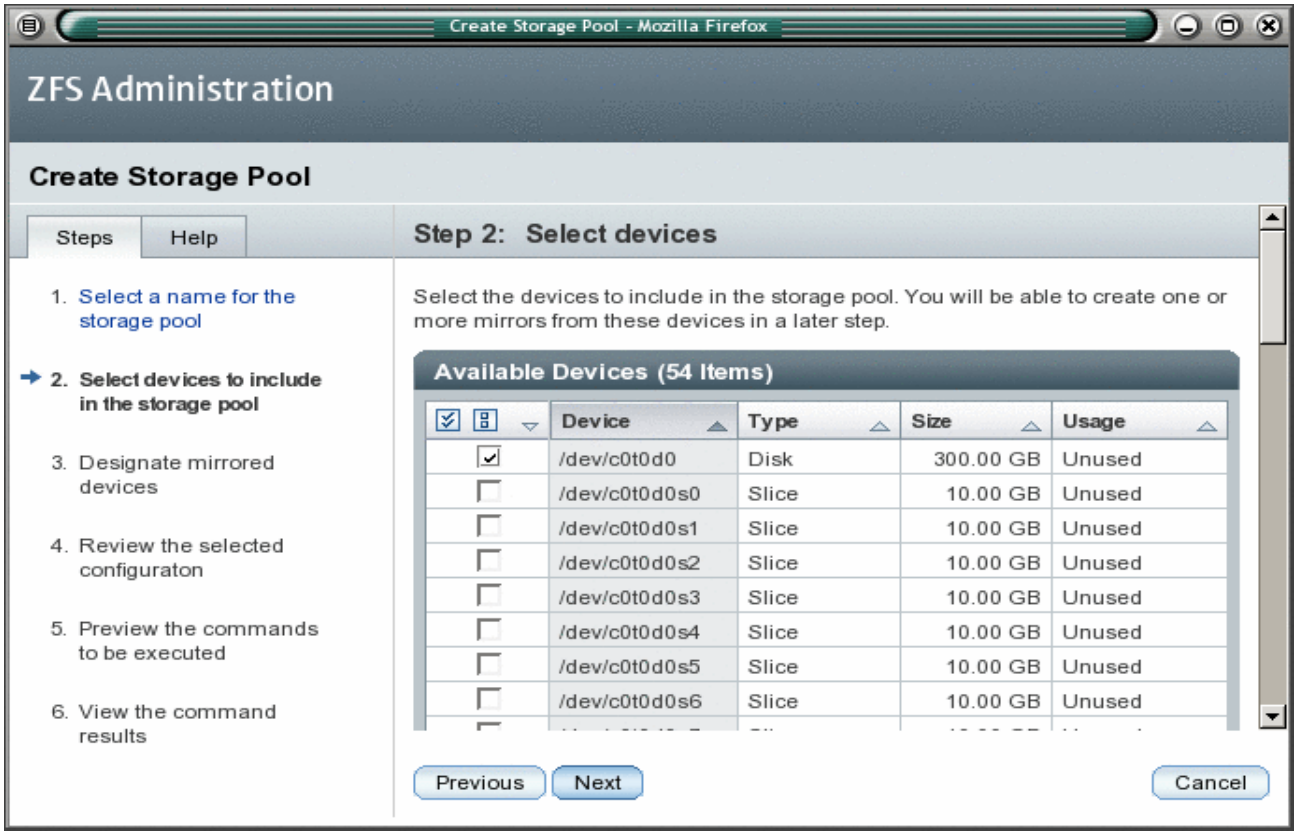
ZFS Graphical User Interface

- Login to the web console
- Drill down to ZFS
- View by File Systems
- View by Devices
- Create a Storage Pool
  - Name
  - Devices
  - Mirrored?
  - Review
  - View Commands



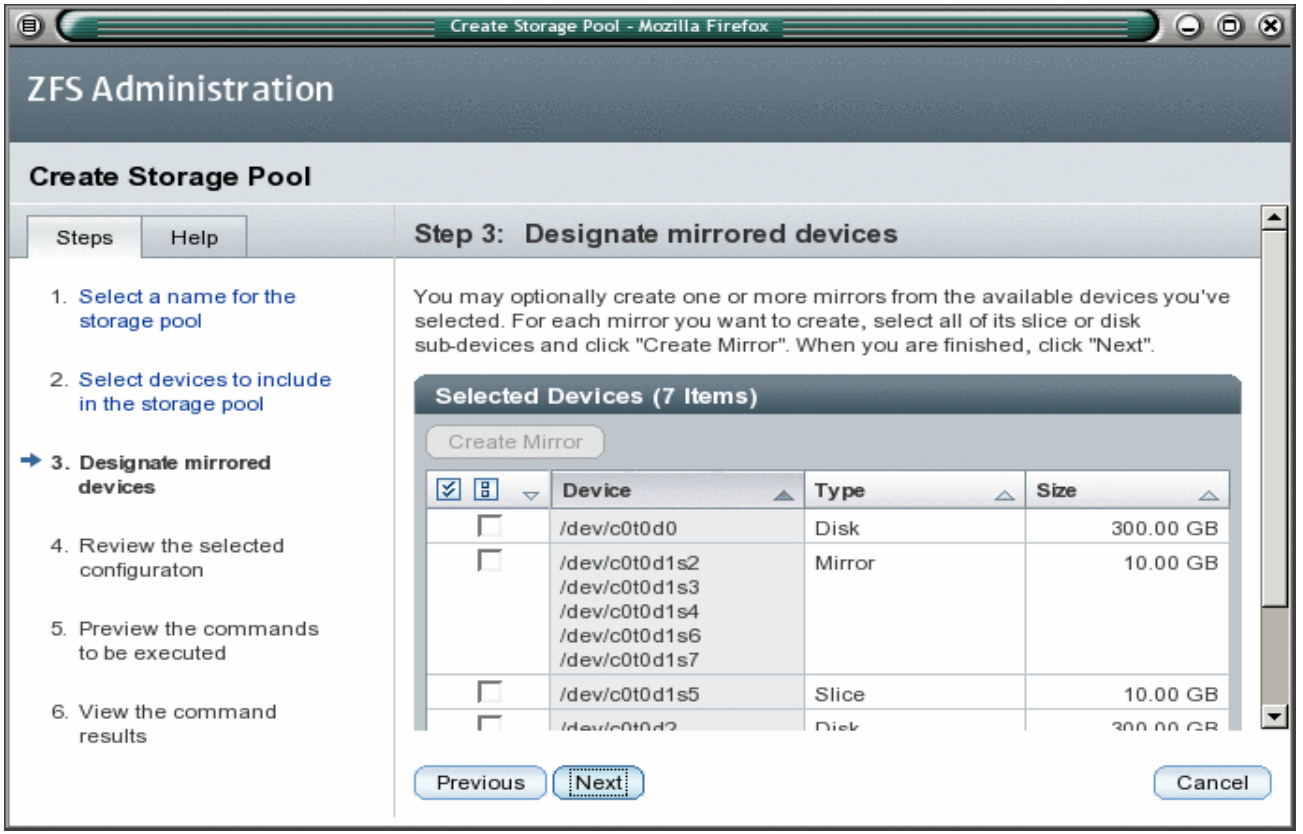
ZFS Graphical User Interface

- Login to the web console
- Drill down to ZFS
- View by File Systems
- View by Devices
- Create a Storage Pool
  - Name
  - Devices
  - Mirrored?
  - Review
  - View Commands



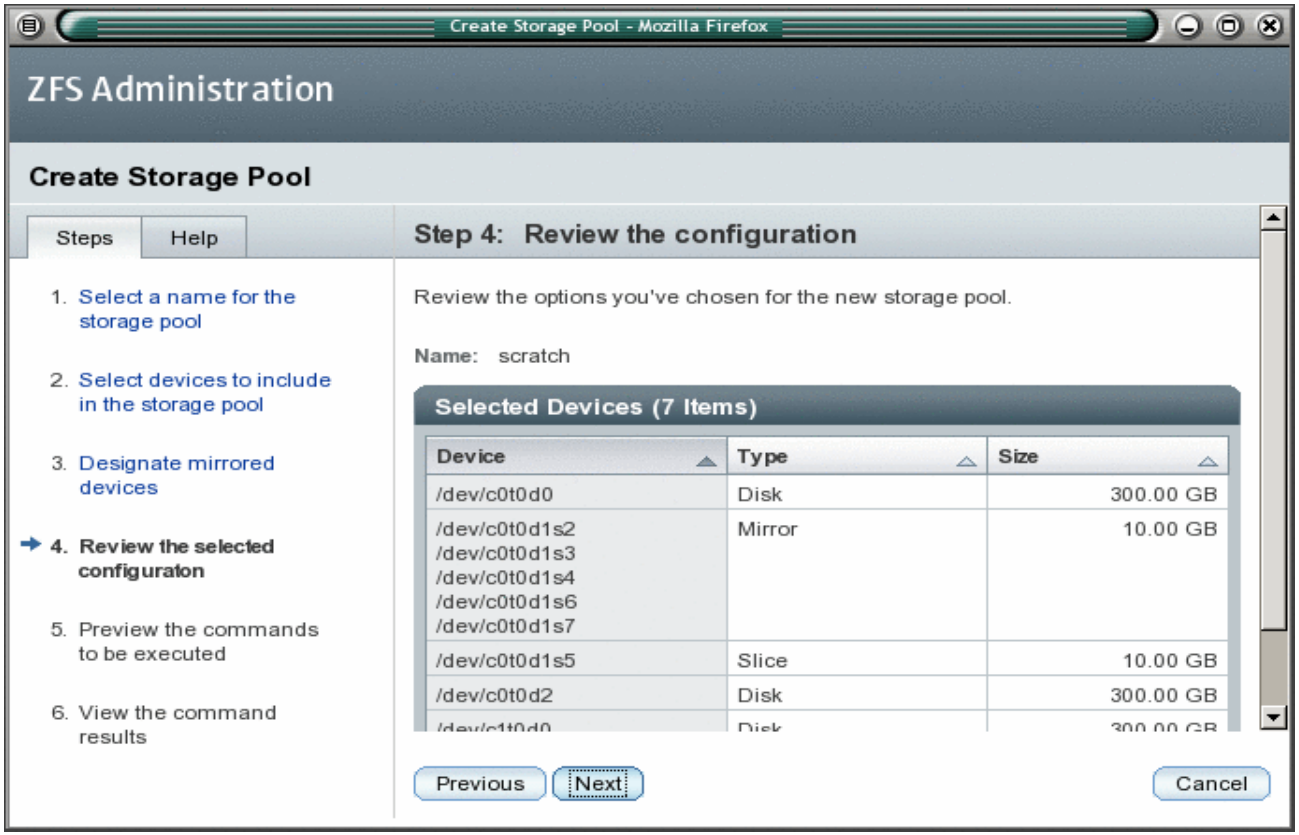
ZFS Graphical User Interface

- Login to the web console
- Drill down to ZFS
- View by File Systems
- View by Devices
- Create a Storage Pool
  - Name
  - Devices
  - Mirrored?
  - Review
  - View Commands



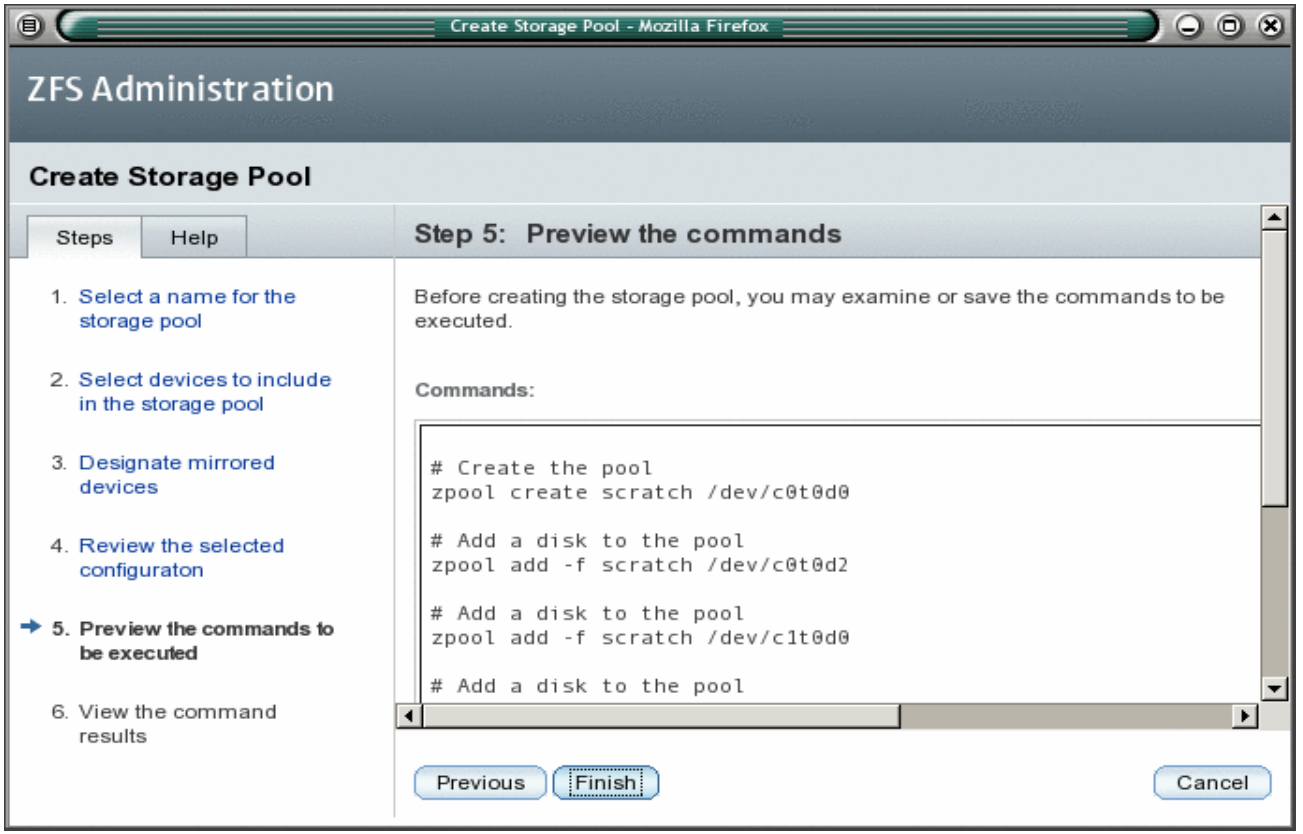
ZFS Graphical User Interface

- Login to the web console
- Drill down to ZFS
- View by File Systems
- View by Devices
- Create a Storage Pool
  - Name
  - Devices
  - Mirrored?
  - Review
  - View Commands



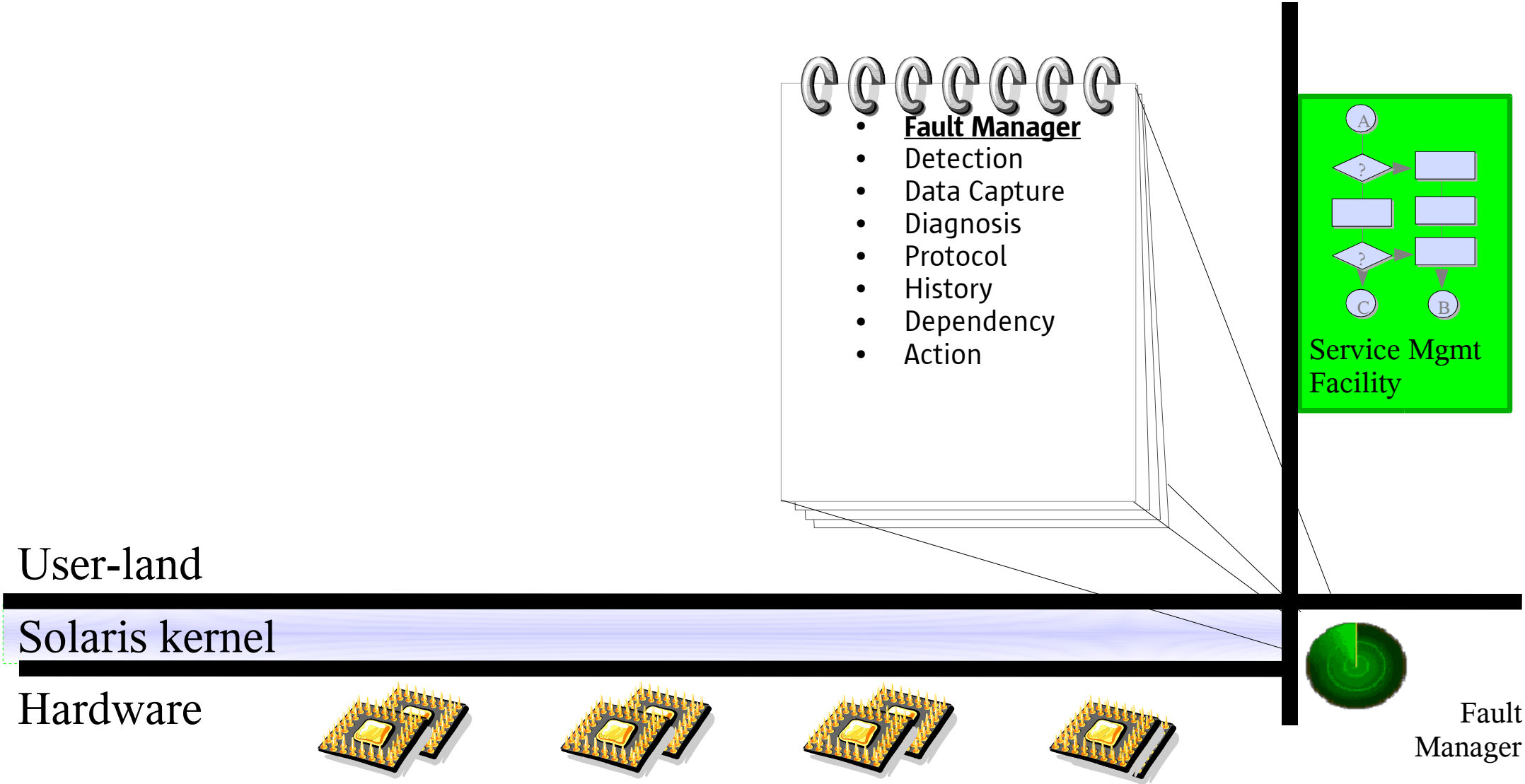
ZFS Graphical User Interface

- Login to the web console
- Drill down to ZFS
- View by File Systems
- View by Devices
- Create a Storage Pool
  - Name
  - Devices
  - Mirrored?
  - Review
  - View Commands



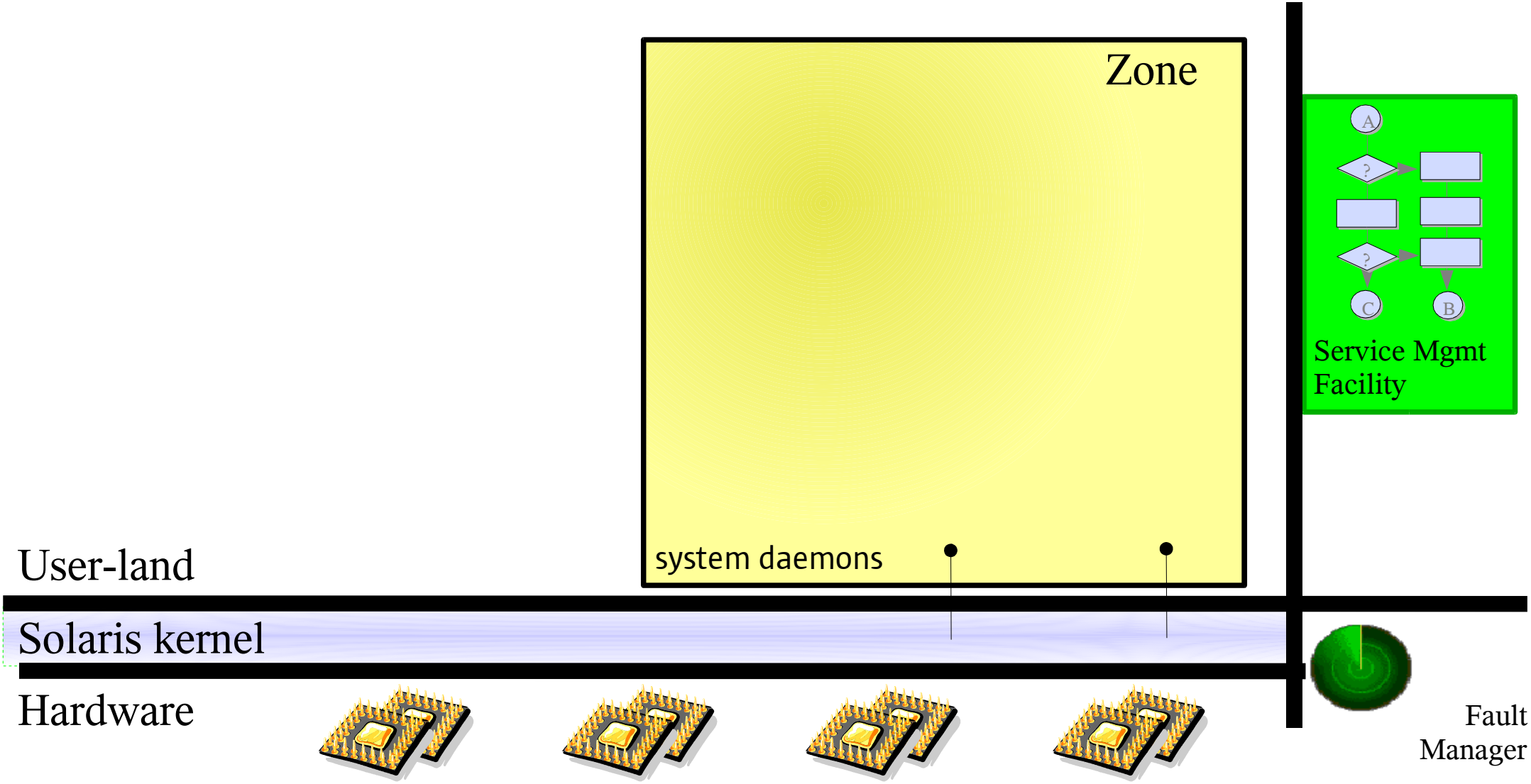
The Complete Picture

Fresh Solaris 10 Load



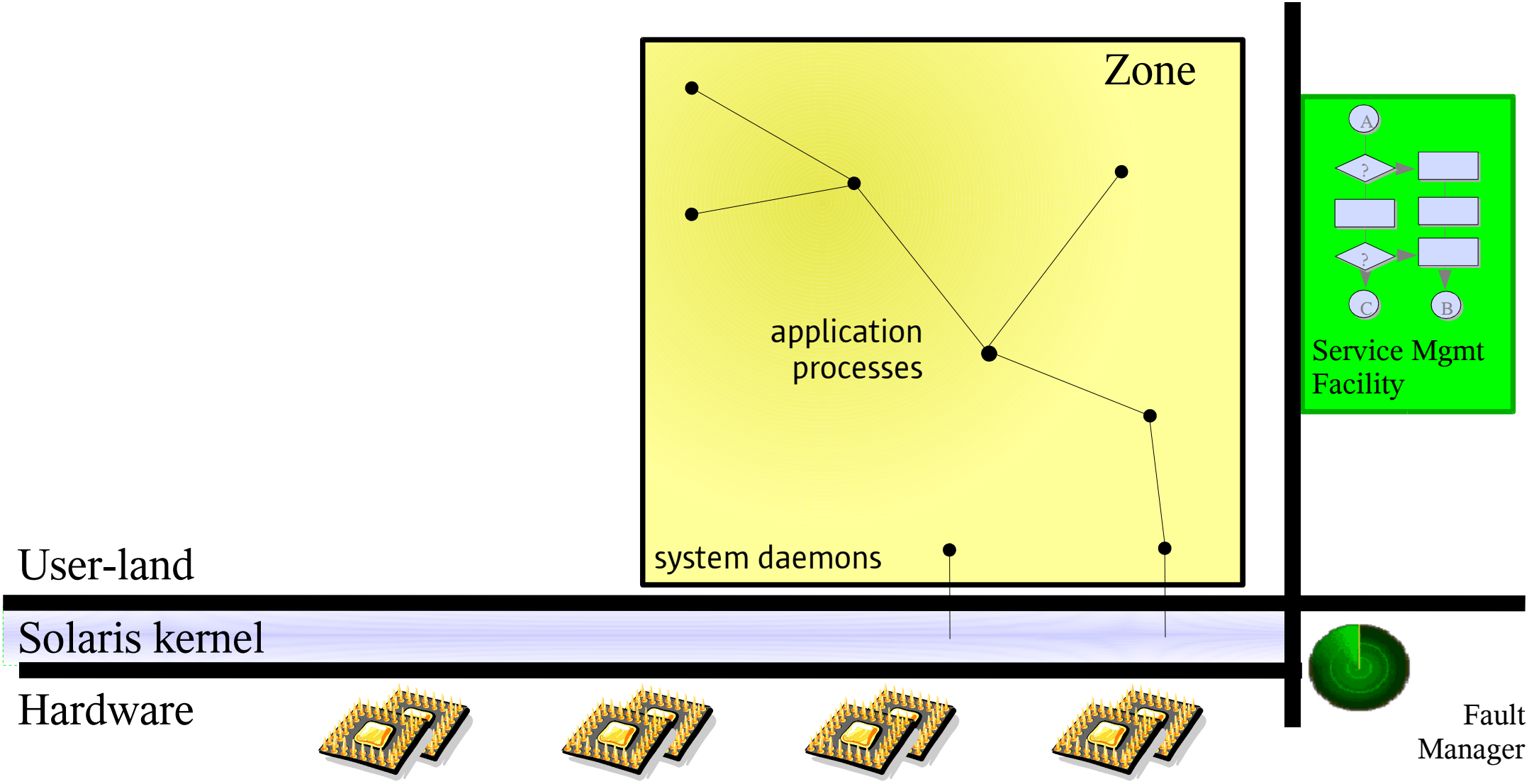
The Complete Picture

N1 Grid Container



The Complete Picture

Application

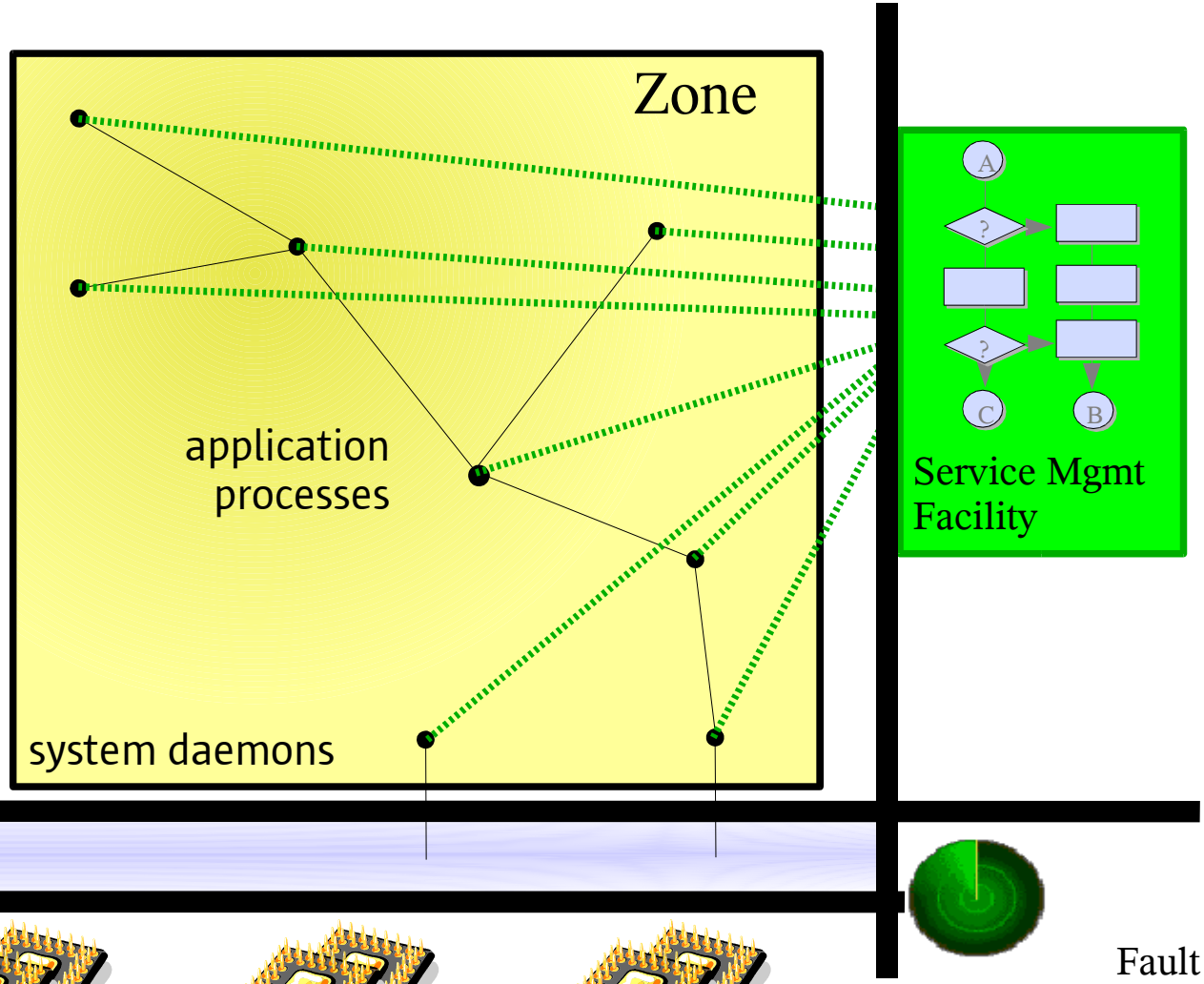


The Complete Picture

SMF Registration

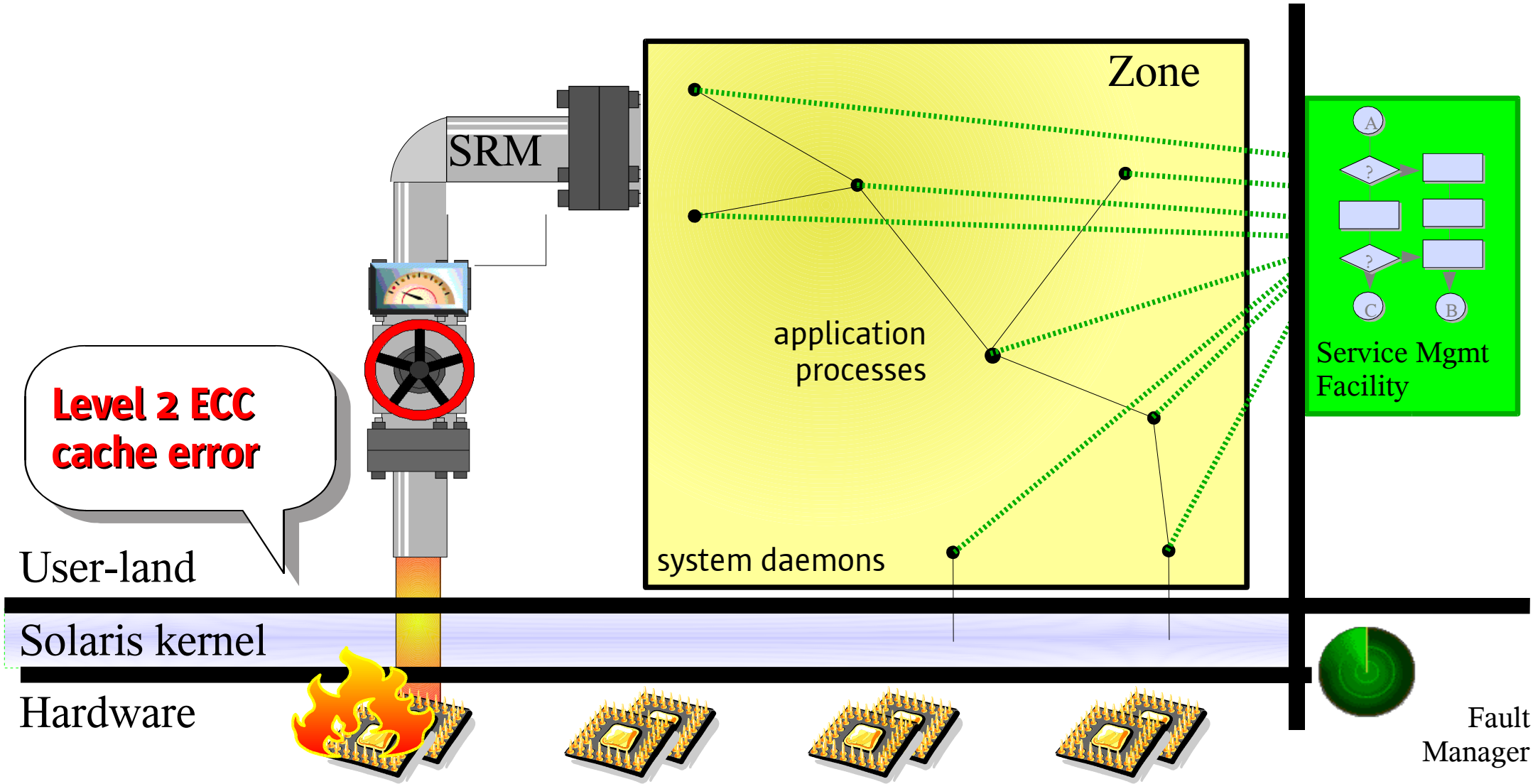
Service Mgmt

- How to start X
- Can I start X
- Should I start X
- How to restart X
- What to restart after X has failed



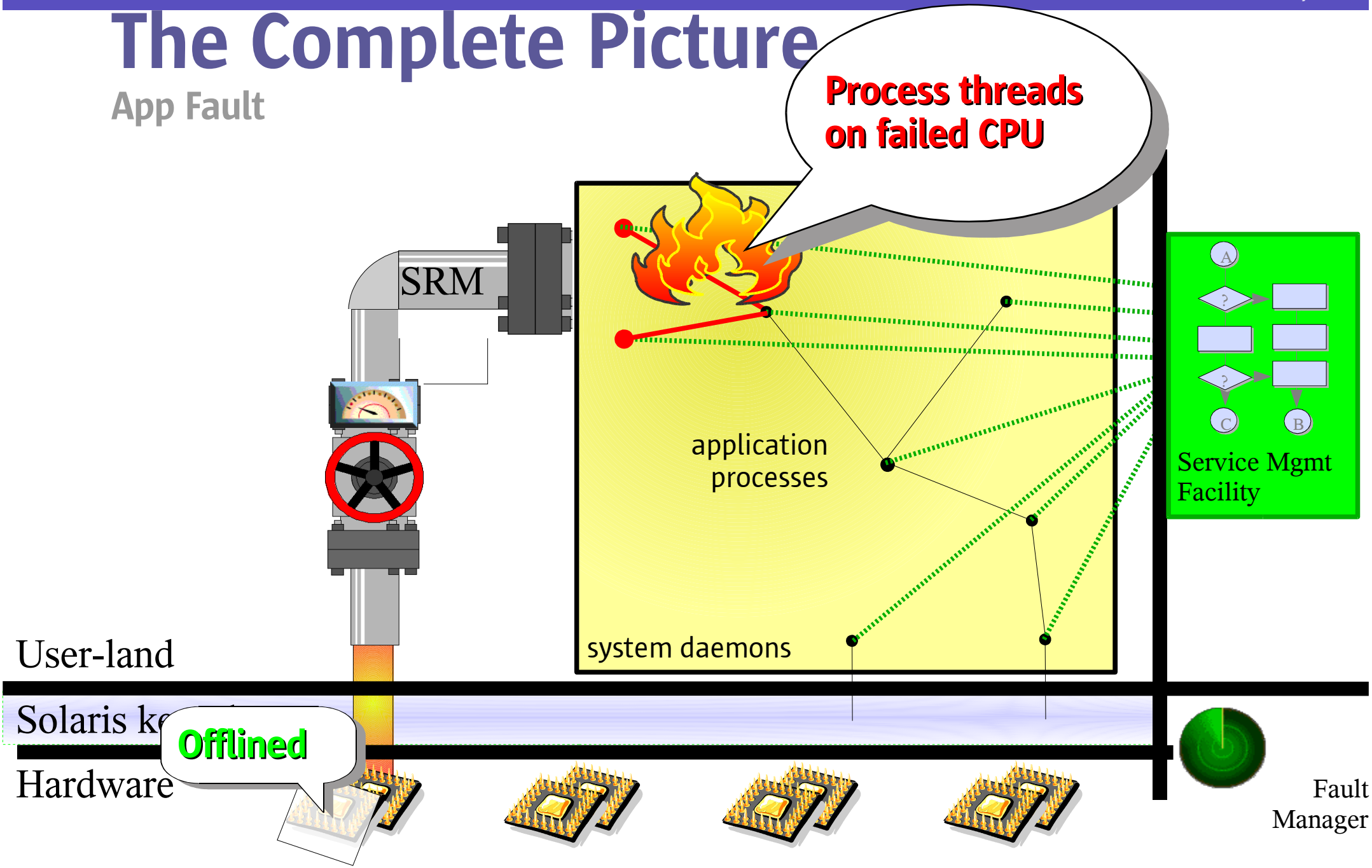
The Complete Picture

CPU Fault



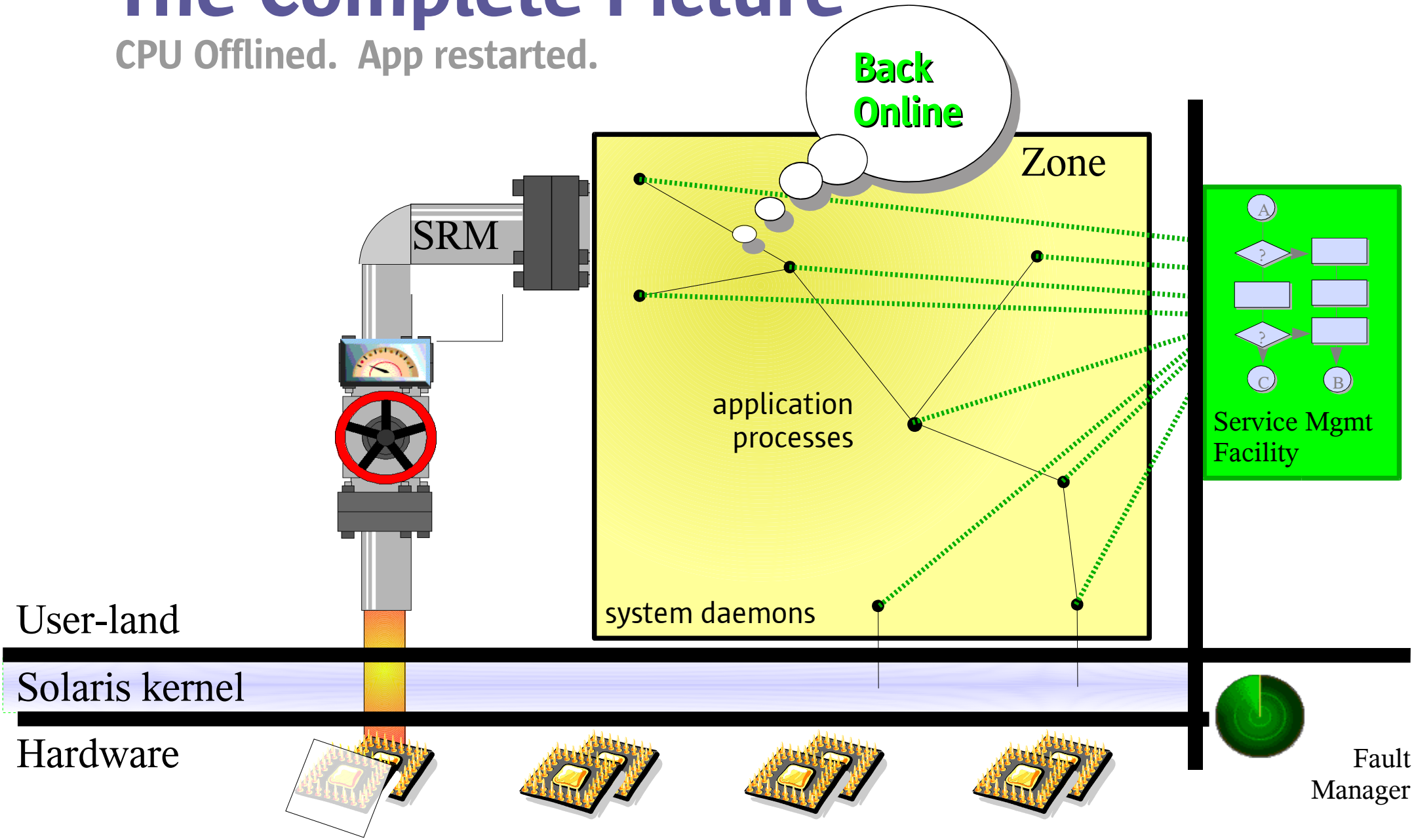
The Complete Picture

App Fault



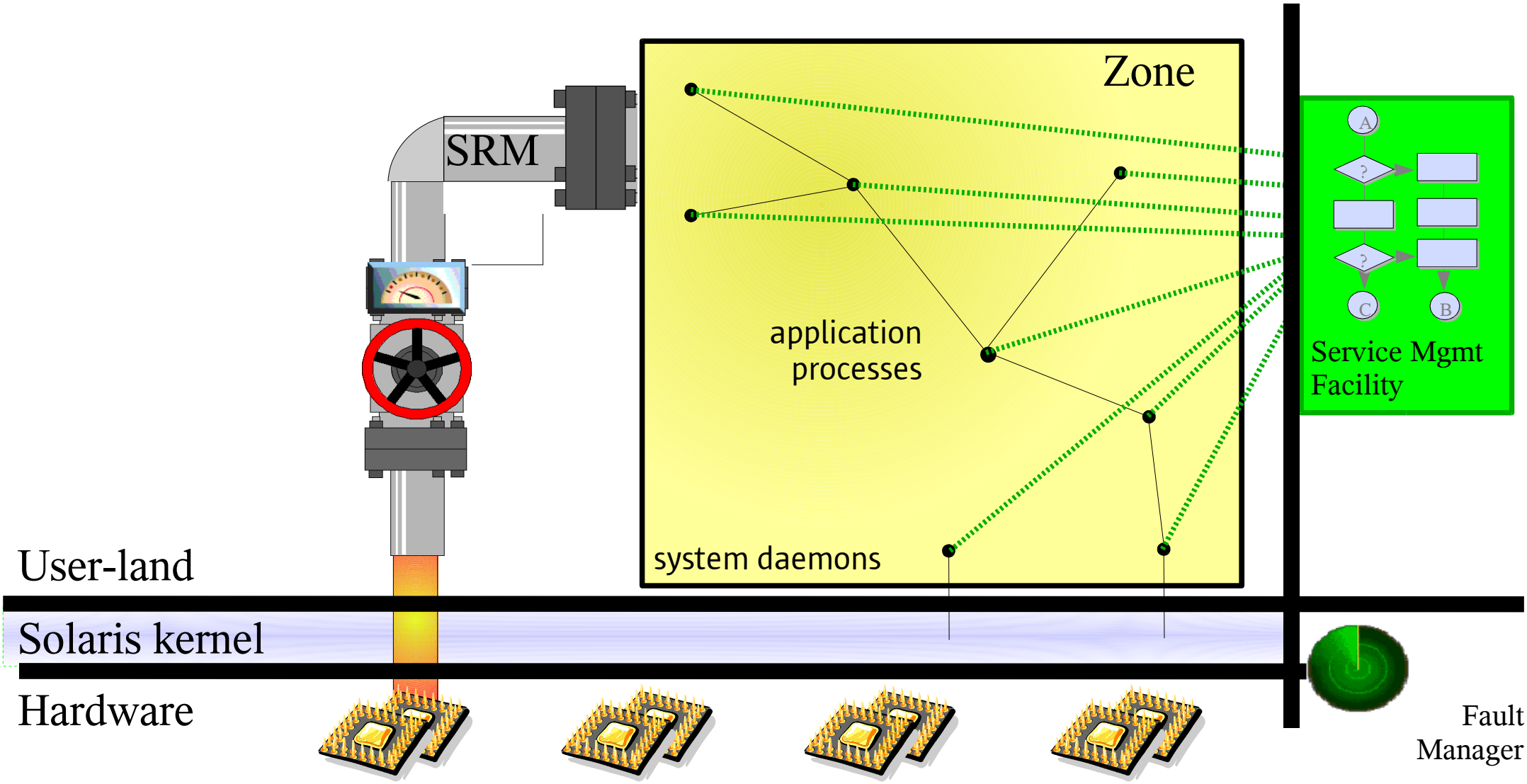
The Complete Picture

CPU Offlined. App restarted.



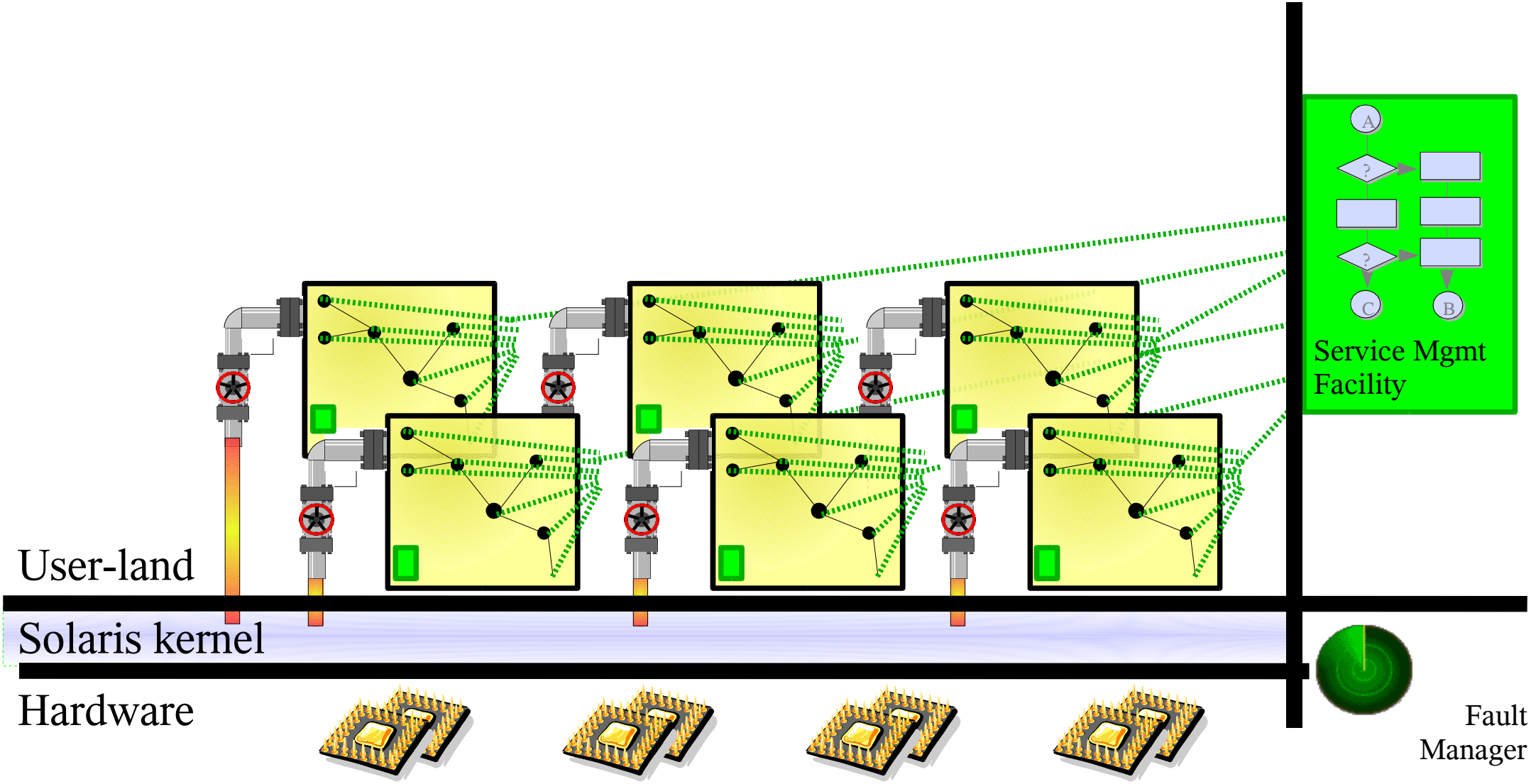
The Complete Picture

FMA + SMF + Zones + SRM



Consolidation View

Build Out with other Applications



Solaris 10 Operating System

linda.kateley@sun.com
blogs.sun.com/lkateley

Sun microsystems