SOLARIS 10 BOOTCAMP

ADELPHI, MD - 2/27/2007

Christine Tran
Solution Architect
Solaris Adoption, US Datacenter
Sun Microsystems

AGENDA

8:30 – 9:00	Introducing Solaris 10, features, release schedule
9:00 – 9:45	Containers, zones, resource manager
9:45 – 10:30	Solaris Management Facility (SMF)
10:30 – 10:45	Break
10:45 – 11:00	Fault Management Architecture (FMA)
11:00 – 11:45	Security Features
11:45 – 12:00	Q&A and wrap-up for Morning Session
12:00 – 1:00	Lunch
1:00 – 1:45	ZFS
1:45 – 2:30	DTrace
2:30 – 2:45	Break
2:45 – 3:15	Migration path and challenges
3:15 – 3:30	OpenSolaris
3:30 – 3:45	Q&A and wrap-up for Afternoon Session
3:45 – 4:00	Closing

Introducing Solaris 10: Features, Release Schedule

Solaris 10

- FCS release January 2005
- The most feature-full OS release since the leap from SunOS to Solaris
- Runs on SPARC, Intel, AMD and compatibles
- There is a hardware compatibility list
- Current release is Update 2 (or 06/06)
- What's with all the release nomenclature?

Solaris 10

- Open sourced, open development
- Binary compatibility guaranteed (with some qualifications)
- Free end-user license
- Pretty much all Sun software is now free
- Free security and kernel patches
- Support costs a little, some patches require support contract.

Major Features

- Zones
- Predictive Self-Healing: SMF and FMA
- ZFS
- DTrace
- Many security enhancements
- New and improved TCP/IP stack
- Many MANY other features and fixes, too many to list.

Solaris 10 1/06

Download: December 21, 2005
Media: January 23, 2006

- Integrated support for new systems
  - Includes those previously supported in HW updates
- GRUB-based boot for x86/x64
- Sun Update Connection
- iSCSI support
- Delivered as part of Solaris Enterprise System
- Improved network performance
- Improved memory management
- Integrated fixes

Solaris 10 6/06

Download: June 26, 2006
Media: July 14, 2006

- ZFS integration
- Fault Manager support for AMD64 processors
- Fault Manager enhancements for midrange and high-end SPARC systems
- PostgreSQL for Solaris
- Multimedia and desktop enhancements
- New system and driver support
- Improved UDP performance
- x86/x64 storage performance enhancements
- Improved SSL performance
- Integrated fixes

Solaris 10 11/06

Download: December 11, 2006
Media: December 19, 2006

- Solaris Trusted Extensions
- Secure by Default
- Logical Domains
- New Solaris Containers functionality
  - Attach/detach
  - Clone
  - Configurable privileges
- Support for new systems
- Integrated fixes

Solaris 10 6/07*

Download: June 4, 2007*
Media: June 18, 2007*

- Network virtualization (Crossbow)
- Solaris Containers for Linux Applications (BrandZ)
- Solaris Containers
  - > Physical memory control
  - > Live Upgrade support
- x64 power management

*Tentative name/release dates

Additional Planned Update Features

Extending Software Reach

Solaris Containers for Linux Applications

- Run unmodified Red Hat applications
- Extension of Solaris 10 Container technology
- Designed to support different “brands”
- Leverages Solaris security / administrative advantages
- Combines the best of virtualization, resource management and OS flexibility

Solaris Containers for Linux Applications

- Released to community: Q4CY05
  - Supports RHAS 3 and CentOS applications and libraries
  - http://opensolaris.org/os/community/brandz
- OpenSolaris integration: September, 2006
  - 32-bit Linux apps only
  - DTrace: Linux PID and syscall provider
- Planned integration in future Solaris 10 update

Solaris 9

- Solaris 9 9/05
  - > Final Solaris 9 generic update
  - > Adds support for UltraSPARC IV+ systems
  - > Integrated bug fixes
- Solaris 9 9/05 HW
  - > Solaris 9
    platform-specific release
  - > Additional support for new SPARC-based systems

Download: Oct. 9, 2006
Media: Oct. 23, 2006

Solaris 8 Transition

- Announcement: August 15, 2006
- Last order date: November 16, 2006
- Last ship date: February 17, 2007
- No additional license required for install on new Sun SPARC systems until August 15, 2007
- Support for five years after last ship
  - > Two years “Vintage Support Phase 1” (ends Q1CY09)
  - > Three years “Vintage Support Phase 2” (ends Q1CY12)
- Trusted Solaris 8 not affected

Next Solaris Release

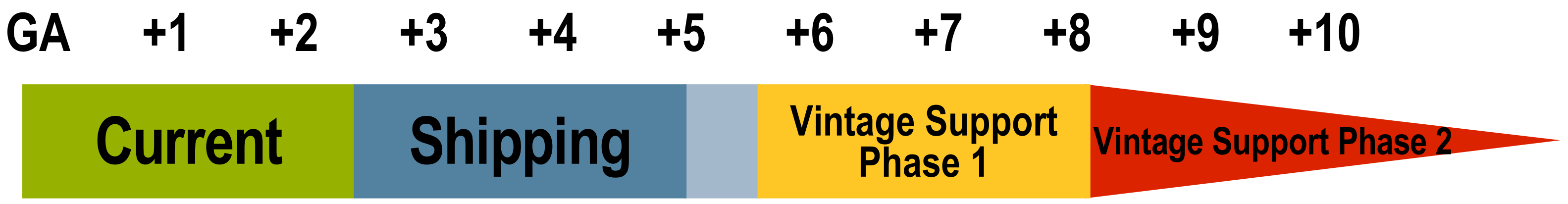
- Development builds available now via Solaris Express program (http://sun.com/solaris-express), updated each month
- Goals include continued focus on ease-of-use, enhanced availability, performance/scalability

Solaris Support Stages

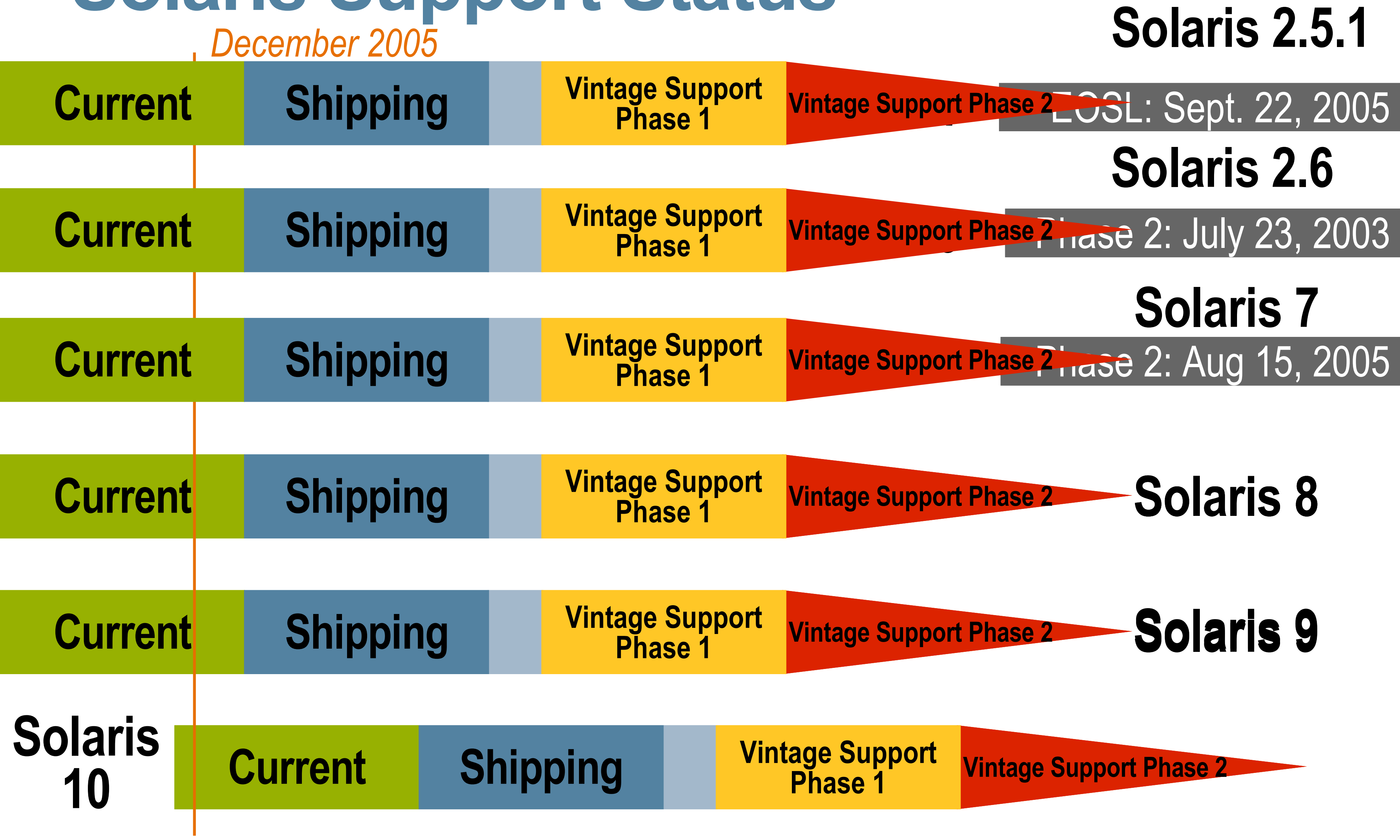
- Current release – GA to EOL
  - > Full support – new bugs logged, bugs fixed, patches released, escalations available, update releases
- Vintage Support Phase 1 – 2 years after EOL
  - > Full support – new bugs logged, bugs fixed, patches released, escalations available, no update releases
- Vintage Support Phase 2 – 3 years after Phase 1
  - > No new patches, no new bugs logged, no escalations
  - > Existing patches still available
- See InfoDocs 10012, 70063

Solaris Release Roadmap

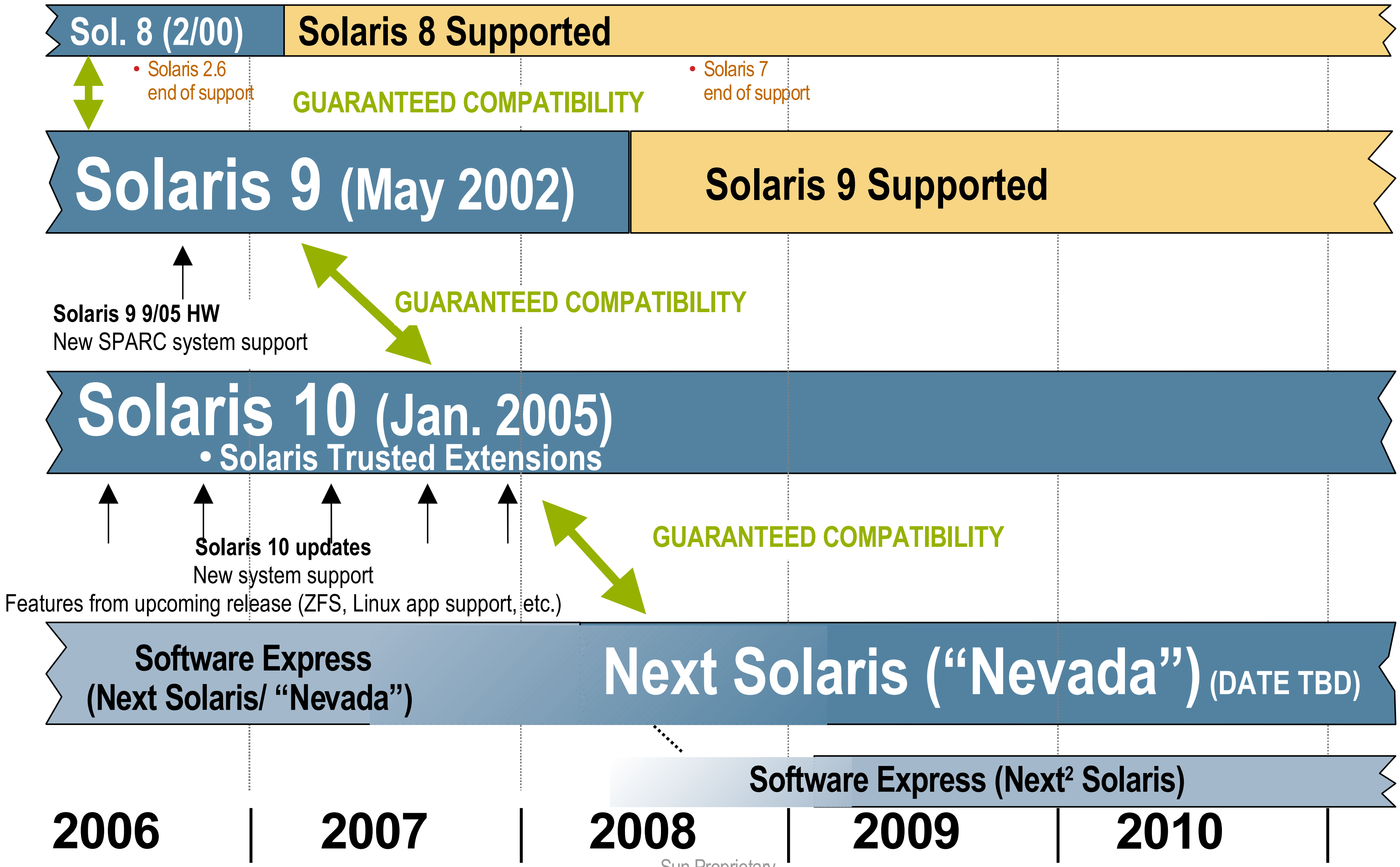
- Solaris release every 24-36 months
  - > Guaranteed application compatibility
  - > Update releases provide new features
- Shipping life: 4-6 years (or more)
- Support life: 10-11 years (or more)
- Support for the OS 5 years after it stops shipping
  - > http://sun.com/solaris/fcc/lifecycle.html



Solaris Support Status



Solaris Roadmap, Apr 2006 – Mar 2010



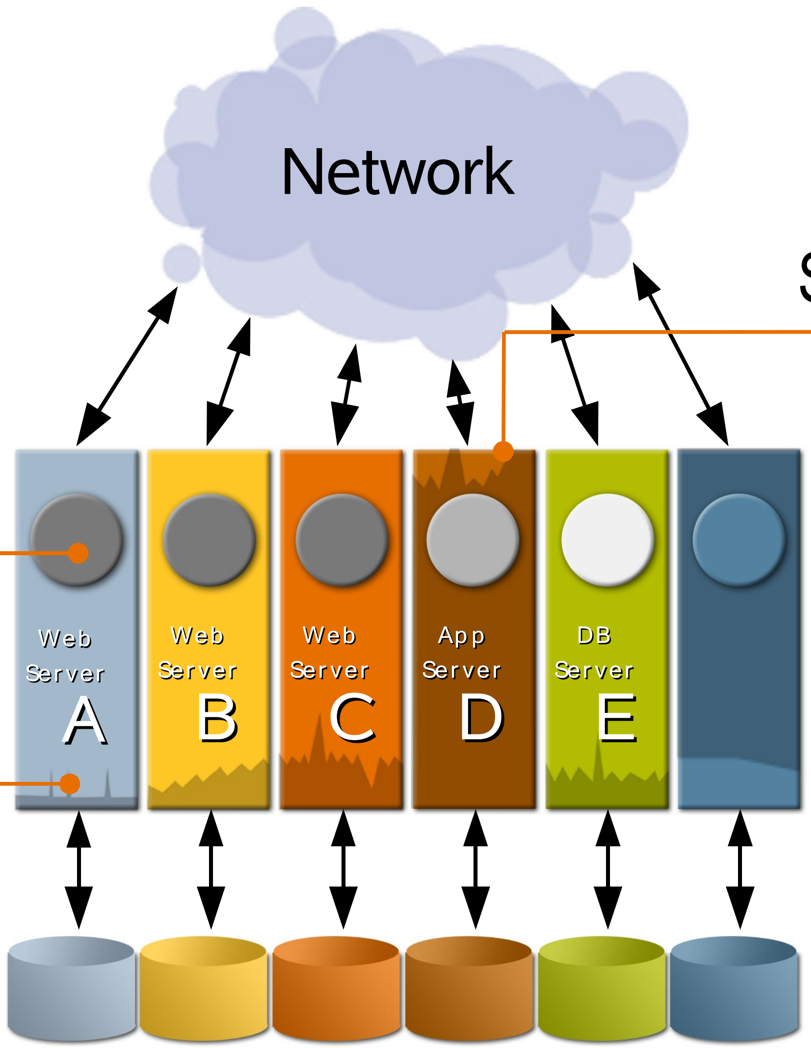
Subscription-based Service Plans for Solaris 10

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				■
US$ Price/Physical Processor/Year, 1-8	**$0**	**$120**	**$240**	**$360**
9+	**$0**	**--**	**$480**	**$600**

Containers, Zones, and the Resource Manager

Traditional Resource Management



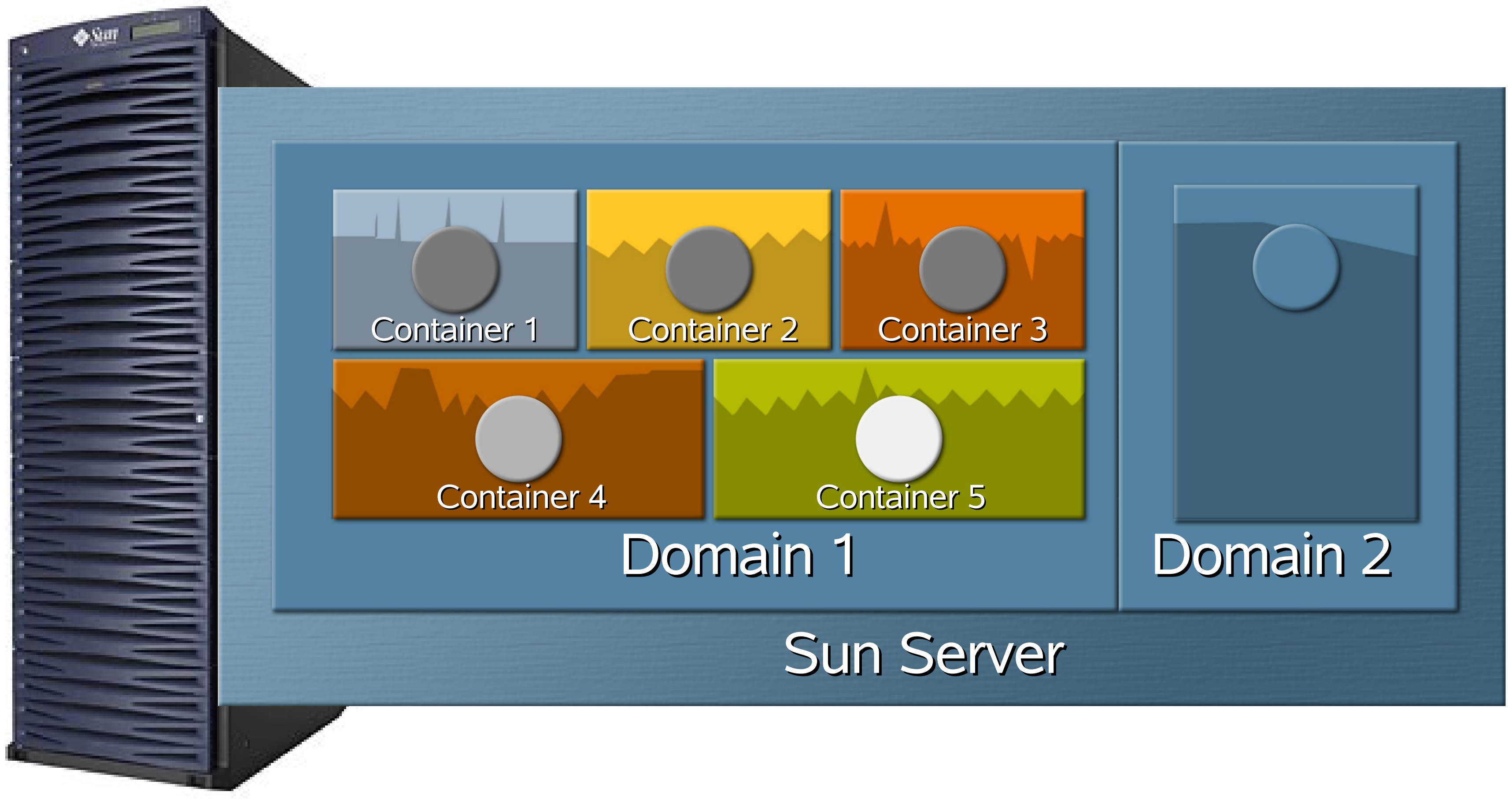
One application per server

Size every server for the peak

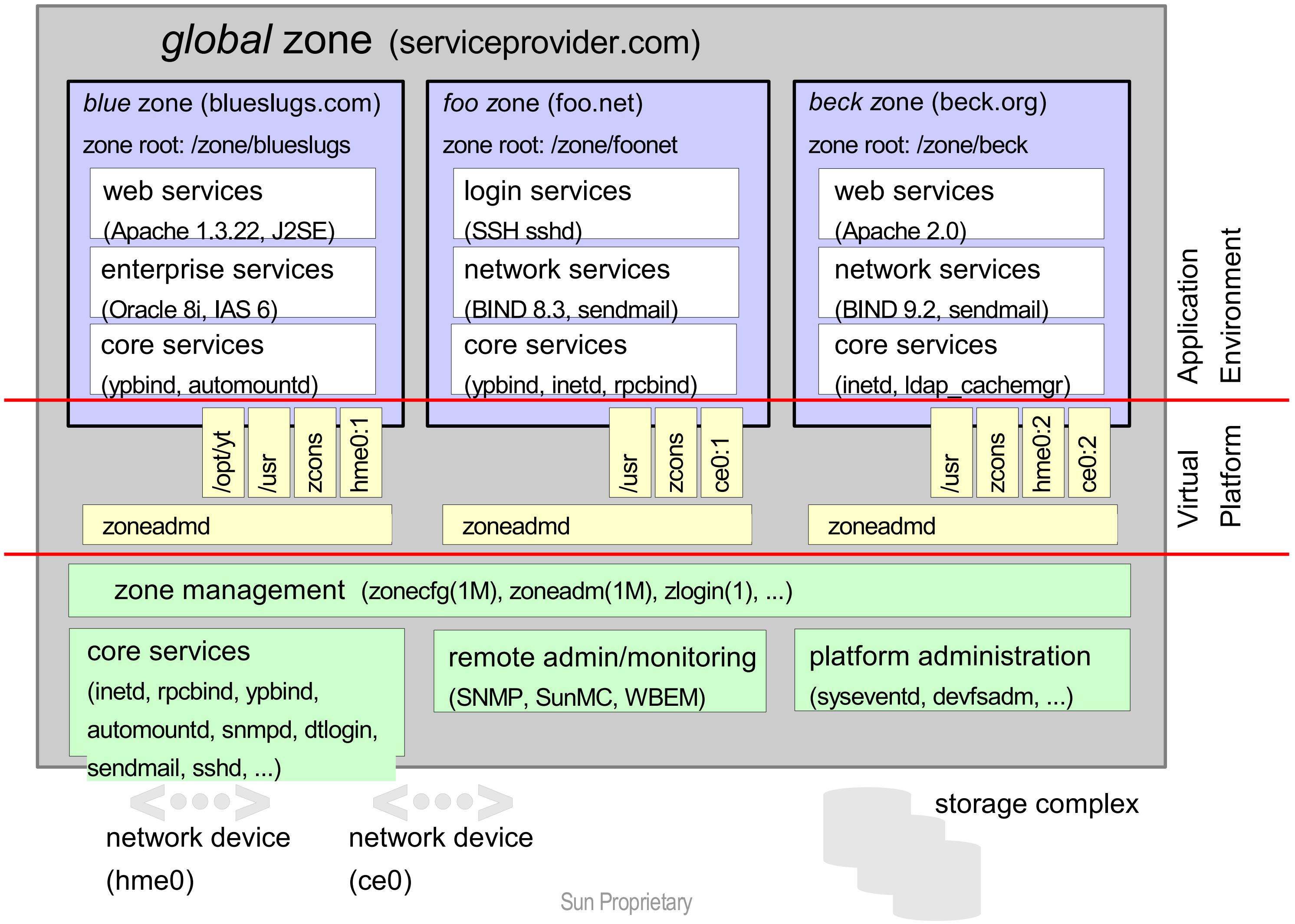
Avg. utilization rate is 20%-30%

Server Virtualization

Solaris Containers and Solaris Dynamic System Domains



Zones representation



Introducing Zones

- Zone is an an OS abstraction for partitioning systems.
- It is *not* an instance of Solaris.
- Each zone appears as an independent OS,with its own configuration, for example:
  - hostname, IP, users, process, package db, filesystems
- Zones share the same physical hardware.
- The real, unabstracted OS is called the global zone
- The number of zones you can have on a system is constrained only by resources.

Zones Isolation

- Zones are isolated from other zones
- This isolation prevents processes in one zone from affecting processes running in other zones, seeing each other's data, or manipulating the underlying resource. Each zone can be rebooted independently without affecting other zones.
- Because zones are virtualized OS, they do not POST hardware and can boot in seconds.
- Isolation also provides an added security benefit in that a compromised zone cannot affect other zones, or the real un-abstracted OS.

Zones Security

- Each zone has a security boundary around it.
- Zones run with reduced privilege.
- A compromised zone is not able to escalate its privileges; thus it cannot compromise the whole system or another zone.

Zones Identity

- Each zone controls its node name, RPC domain name, time zone and locale
- Each zone can use a different naming service such as DNS, LDAP and NIS
- Separate /etc/passwd files means that root can be delegated to the zone
- User ids may map to different names when domains differ (as with NFS now)

Zones resource control

- Resources are allocated per zone by the privileged user of the global zone.
- Among the types of resources that can be allocated are disks, filesystems, network interface, maximum number of process than can run in a zone, a guaranteed minimum share of CPU resource.
- Resource pools can also be bound to zones.
- We are working on mem-set and swap-set

Zones vs. containers

- Zones is a Solaris 10 feature.
- Containers is a generic term, meaning a virtual operating environment with a limited operating resource.
- Back in Solaris 9, you could use Resource Manager to group your work into projects, put CPUs into a pool, and bind projects to pools.
- You could also use rcap to regulate memory use.
- In Solaris 10 the boundary is formalized.
- Zones and Containers are now used interchangeably.

Demo

- Zones demo

Basic Zones Commands

- zonecfg(1M) – configure a zone
- zoneadm(1M) – basic zone administration
  - Install or remove a zone
  - View status of a zone
  - Boot a zone
- zlogin(1) – enter a zone
  - Interactive
  - Non-interactive
  - Console
- zoneadmd
  - Manages the virtual platform
  - One per local zone

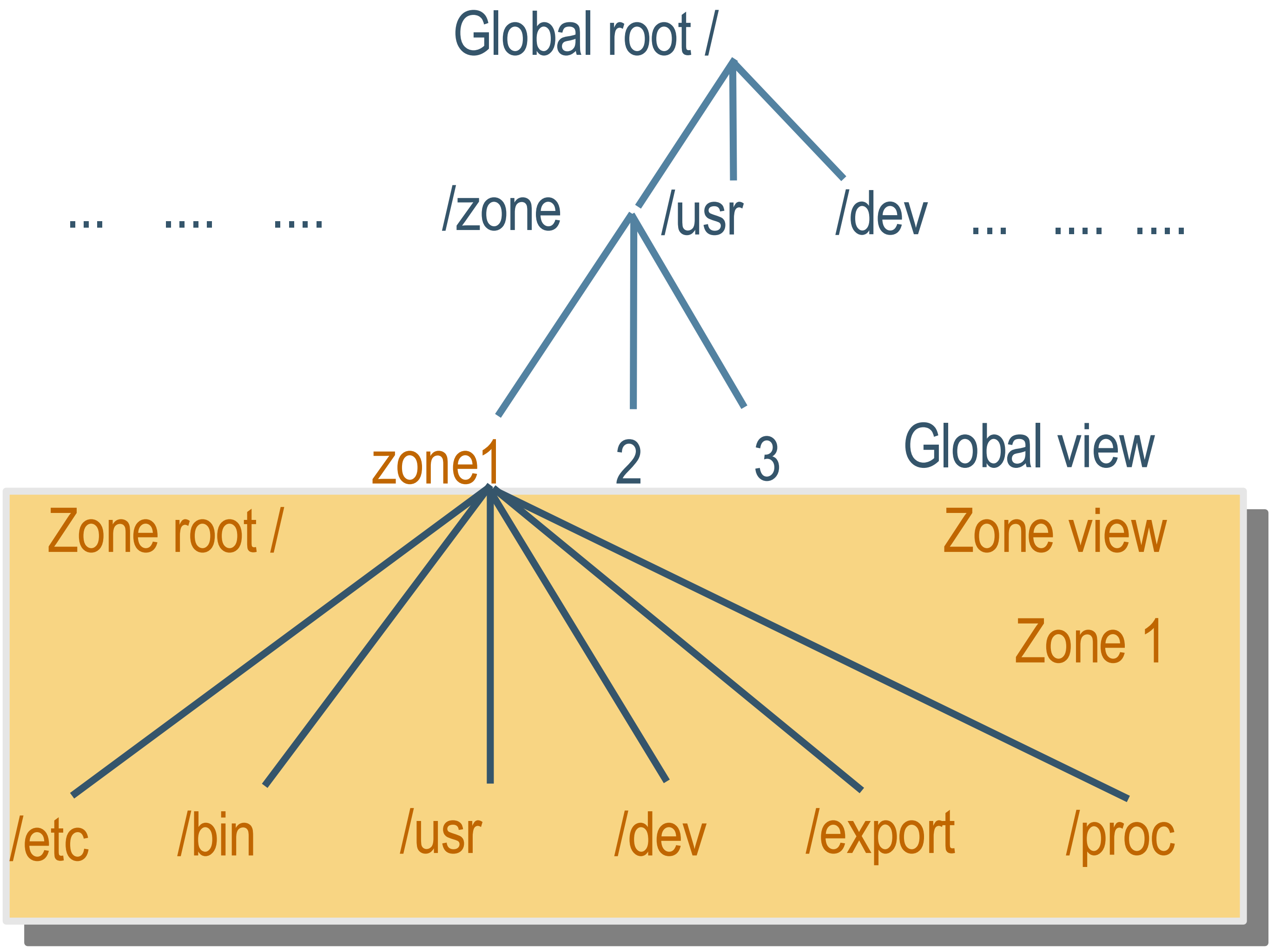
zonecfg -z <zonename> <cmd>

- create – create a zone configuration
- delete – delete a zone configuration
- cancel – cancel current operation
- commit – save configuration on stable storage
- verify – validate the zone configuration
- add <resource>
- remove <resource>
- info
- help

Primary Zone States

- Configured: Configuration completely specified and committed to stable storage
- Incomplete: Configured, installation started but not complete
- Installed: Packages have been installed under the zone's root file system
- Ready: Virtual platform has been established
- Running: User processes are executing in the zone application environment

Zone Filesystem



Inherited Package Directories

- Four default `inherit-pkg-dir` resources provided
  - `/lib`, `/platform`, `/sbin`, `/usr`
- Implemented via a read-only loopback file system mount which provides security as well as storage and virtual memory efficiencies
- `/opt` is good to add to this list, unless it will be configured differently than in the global zone

Configuration files

- /etc/zones
  - SUNWdefault.xml – defaults for zonecfg
  - <zone>.xml – zone configuration file
  - index – state information for all zones

zoneadm -z <zonename> <cmd>

- `zoneadm(1M)` is used by the global zone administrator to
  - install a new root file system for a configured zone
  - list zones and optionally their state
  - verify whether the configuration of an installed zone is semantically complete and ready to be installed
  - boot or ready an installed zone
  - halt or reboot a running zone
  - uninstall the root file system of an installed zone

Accessing a local zone

- `zlogin(1)` is used to enter the zone
- Interactive mode
  - > Similar to `rlogin(1)` or `ssh(1)`

```
global# zlogin zone1
```

- Non-interactive mode
  - > Similar to `rsh(1M)` or `ssh(1)`

```
global# zlogin -l jpb zone1 ps -ef
```

- Zone pseudo-console available for each zone
  - > Mimics a hardware console
  - > Accessible via `zlogin -C`

```
global# zlogin -C zone1
```

Zones: Advantages

- Consolidates multiple applications
- Provides security perimeter between applications and underlying system
- Makes more effective use of hardware
- Simplifies administration
- Adds flexibility to resource management
- Fast and easy deployment of a virtual OS

Zones: Challenges

- Single kernel, one OS, one KU patch rev.
- One TCP/IP stack, one routing table, no per-zone routing table.
- Because of this, inter-zone communication is short-circuited.
- No IPFilter for non-global zone.
- Inter-zone traffic cannot be snooped, or audited.
- We are working on this aggressively.

Zones: Challenges

- Does not yet work with LiveUpgrade or flash.
- Depedent on the non-virtualized OS, cannot easily move zones.
- Backup and restore take special considerations.
- Cannot directly manipulate /etc/system.
- Cannot directly manipulate metadb.
- Presents a new paradigm for system management, although SunMC 3.6.1 has zone management module built in.

Predictive Self-Healing: Solaris Management Facility and Fault Management Architecture

Motivation for SMF

- How are programs or daemons started today?
- Is running a software program the same as delivering a service? What exactly is a service?
- Does there currently exist an OS framework for service support and management?
- Could these services be related? How are the relationships described?

Diagnostic Ability

- An application fails to start
  - > Missing configuration file?
  - > Failed to mount a filesystem?
  - > Database is late in starting up?
  - > Missing another component?
- Lack of knowledge of service boundary and service interdependencies limits error handling ability of system.
- Lacks common framework of error-handling
- Lacks restart capability

Service Management Today

- Thousands of different text files, arbitrarily grouped and managed with multiple administration techniques.
- Undeclared, often unknown dependencies; linear startup is a by-product.
- Lacks common interface, each service is started a different way.
- Does not address multiple instances of a service, for example, web servers.
- Does not address services that may span multiple hosts, for example, a grid.

Introducing SMF

- A consistent service model in a common framework: command interface, service description, property specification, status view, etc.
- A meaningful system view.
- Ability to state dependencies (by-product is parallel boot).
- Restart capabilities
- All data stored in persistent, transaction-based repository.
- Snapshots allow undo and rollback to a working configuration.

A Service is

- An abstract description of a long-lived software object.
- May describe object on a collection of machines (grid).
- May have multiple instances (httpd).
- Each instance of a service has a well-defined state and a well-defined error boundary [process contract].
- Each service defines methods: start, stop, refresh, etc.
- Each service defines dependencies: what it depends on, and may define dependents: what depends on it.

FMRI: Fault Mgtment. Resource ID

svc://localhost/network/login:rlogin

- Scheme: svc or lrc
- Location: localhost or hostname (future release)
- Functional Category:
  - > Application – traditional daemons
  - > Device – useful for dependencies
  - > Milestone – similar to run levels
  - > Network – converted services from inetd.conf
  - > Platform – platform-specific services
  - > System – platform-independent system services
  - > Site – reserved for local site use

FMRI: Fault Mgtment. Resource ID

svc://localhost/network/login:rlogin

- Service Description – related to method or RC script
- Instance – “default” is the default instance
- FMRI can be addressed by the shortest unique match
- Some common names have changed to a different FMRI, for example: syslog is now system-log

Service State

- 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, usually due to a dependency that has not been satisfied, or an error in the start method.
- disabled – the service instance is not enabled and is not running.
- maintenance – the service instance has encountered an error that must be resolved before it could be started again. There are many reasons why a service could be in this state.

Service State

- legacy_run – the legacy service is not managed by SMF, but the service can be observed.
  - Faults not handled by SMF, no automated restart
  - Administrative error undetected
  - Software or hardware error results in process death
- 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.

Service Dependency

- A dependency is a formal description of other services that are required to start a service.
- A service can be dependent on another service or files. When a dependency is not met, the service stays offline.

Service Dependency

- Whether a dependency is satisfied depends on its type:
  - > require_all – all services are running or all files are present.
  - > require_any – at least one is running or at least one file is present.
  - > optional_all – all are running, disabled, in maintenance, or not present. For files, this type is the same as require_all.
  - > exclude_all – all are disabled, in maintenance, or when files are not present.

SMF Manifest

- Description of the initial configuration of a service in XML.
- Loaded into the SMF repository at boot time.
- Sun-delivered services live in /var/svc/manifest.
- For ISVs, manifests should be placed in the appropriate subdirectory /var/svc/manifest.
- Manifests for anything that is specific to the customer's site only can go in /var/svc/manifest/site. This is reserved for local use.

SMF Manifest

- A manifest identifies, at the very least:
  - > the service name
  - > what the service is dependent on
  - > ways to start and stop the service
- Can contain other attributes, such as the delegated restarter, config files, log files and others.

Service Configuration Repository

- Located in /etc/svc.
- Distributed between local memory (volatile) and local file (repository.db).
- repository.db holds the persistent service description read from the manifests.
- volatile is a tmpfs allocated from swap containing transient data (lock files, init state, log files).
- The underlying DB engine is sqlite 2.8
- Zones have their own repository.
- Changes to services should be made against repository.

Profiles and Archives

- Profile – a set of service instances and their enabled or disabled state. Useful for copying service states between systems.
- Generated by svccfg extract
- Archive – a complete set of persistent data for all service instances. Useful for copying service definitions between systems.
- Generated by svccfg archive
- Output is an XML file similar to manifests.

Demo

- SMF Demo

Basic SMF Commands

- General commands
  - > `svcs(1)` service status listings
  - > `svcadm(1M)` administrative actions
  - > `svccfg(1M)` general property manipulation
  - > `svcprop(1)` property reporting
- inetd management commands
  - > `inetadm(1M)` administrative actions/property modification
  - > `inetconv(1M)` conversion of legacy inetd.conf entries

New boot process

- Instead of booting to run level, SMF introduces the concept of milestone.
- milestone – a service which specifies a collection of services which declare a specific state of system-readiness.
- You can boot to specific milestone, services not part of that milestone are temporarily disabled.
- The default milestone is the milestone you will transition to at every boot. It is usually milestone “all”.
- init S, boot -s and friends still work.

SMF: Advantages

- Common administration interface
- Persistent service states
- Portable profiles
- Codify service dependencies, leas to parallel boot, somewhat easier troubleshooting

SFM: Challanges

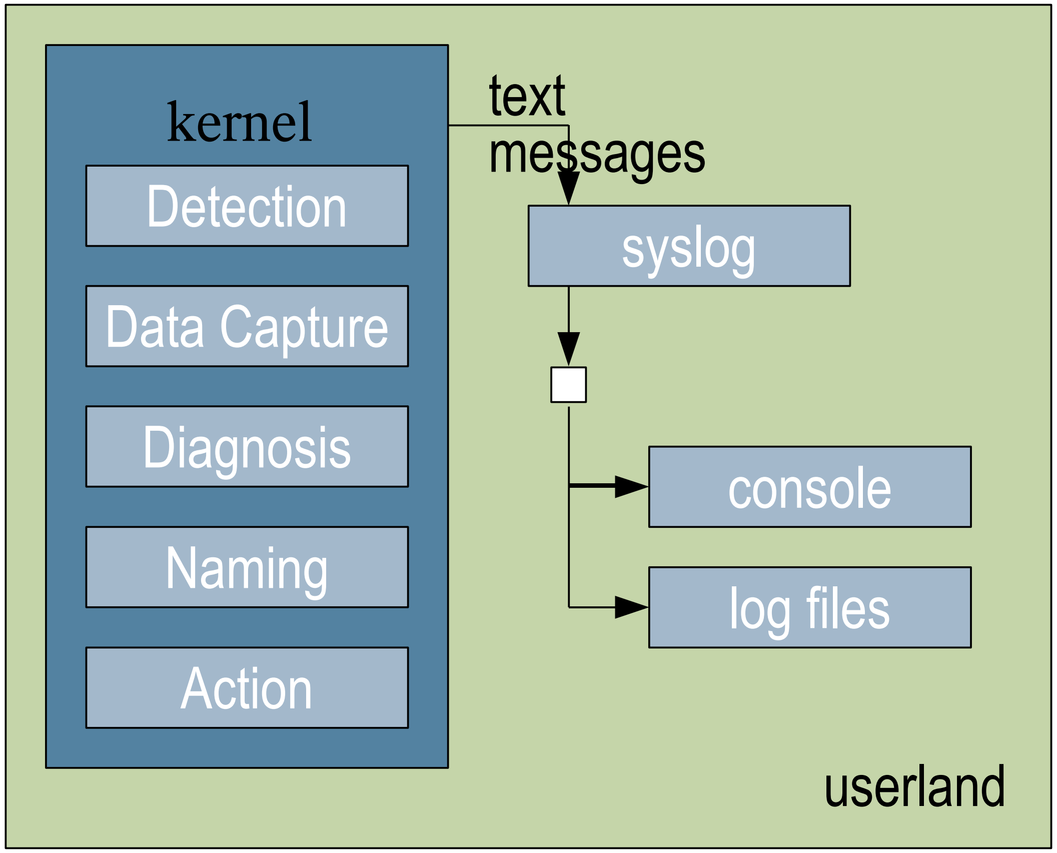
- Totally new and intimidating
- SMF manifest writing

Before FMA

- Previous to Solaris 10, everything having to do with fault diagnosis is done in the kernel.
- One driver does many things:
  - > Detect error condition
  - > Collect data on error
  - > Send data on error to logging facility
  - > Attempt to correct error
    (offline CPU, retire memory pages, etc.)

Before FMA

- Faults can propagates to other subsystems, with their own drivers, which attemps to do the same thing.
- Each driver attempts to reach a diagnosis without knowing what else is happening in other driver subsystems.



Before FMA

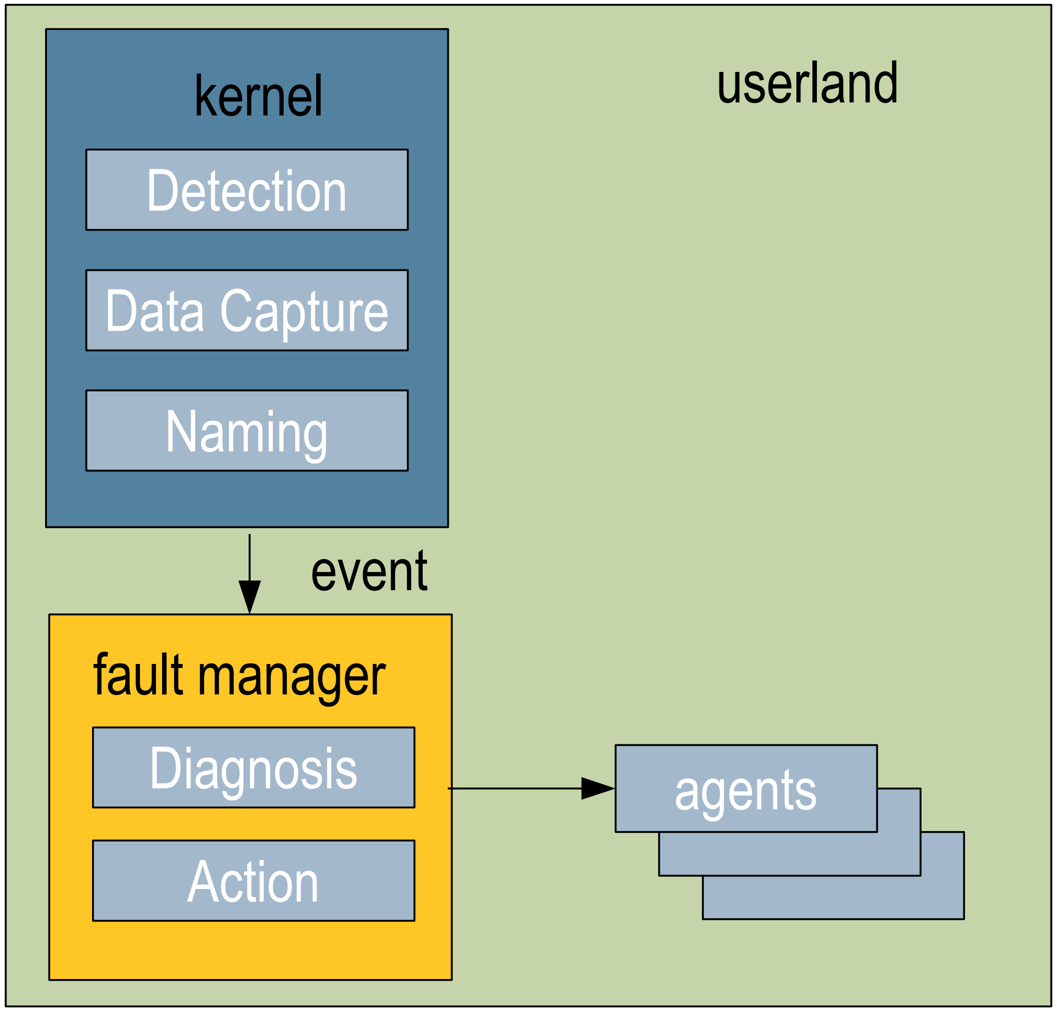
- The result is multiple drivers doing work in parallel to find perhaps just one cause.
- Many log messages spewing from different driver subsystems, saying different things.
- The task of determining the root cause of the problem is left to a human looking at these logs.
- How fast one can come to a root cause depends on the experience of the system administrator.

After FMA

- Solaris 10 provides error detectors, responsible for collection information about errors called ereport.
- ereports are forwarded to fmd, the fault management daemon.
- fmd manages many diagnosis engines, which provide the logic regarding relationships between the errors.
- The logic about how errors correlate is encoded in these engines.
- Error symptoms go to the diagnosis engine instead of various log files.

After FMA

- Humans are removed from the main role of fault determination.
- Diagnosis engine determines what's faulty and a list of “suspects” are sent to the agents.
- Note: syslog still works the same way.



Current practice for error handling

- Various methods depending on the subsystem, various locations for log files.
- Different error message string, depending on the coder
- Can panic system without doing anything to ameliorate.
- Most of the time pushed to System Management software based on SNMP. Good for multiple-system view, not so good for component view.

Current practice for fault diagnosis

- Usually based on log-scraping
- Solaris syslog messages are “unstable”
- Different subsystems throw out different error messages which require a human to correlate
- Incomplete view of a problem depending on who is looking at what
- Correct diagnosis depends on the human looking at the error reports

Terminology

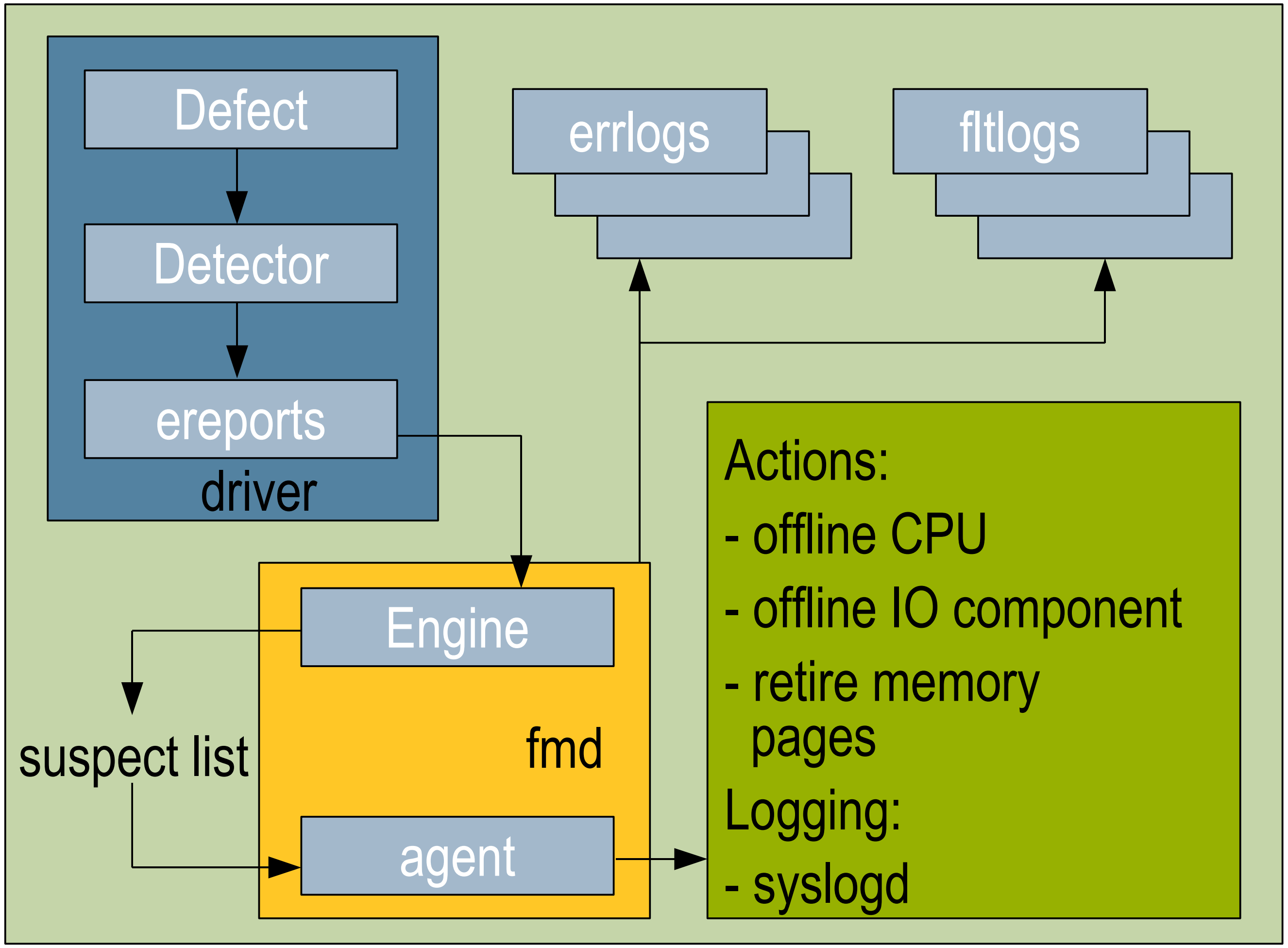
- Defect – a defect in the system may corrupt some signal or data. This condition is detected by a driver, which has a detector in place.
- Detector – collects all information it can about the error and create an ereport.
- fmd (fault management daemon) – acts as a central switchboard for events. Detectors send fmd ereports, fmd routes ereports to the correct diagnosis engine.

Diagnosis engine

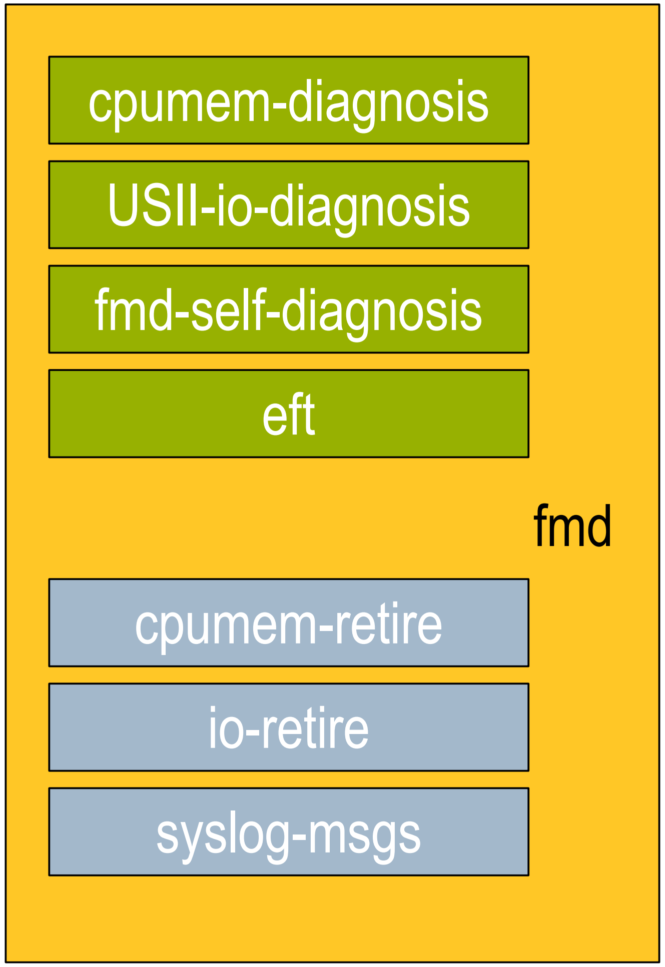
- The diagnosis engine compiles information about errors in the system which it knows about. Based on statistical or platform information, the DE may see enough ereports conclude which component is faulty.
- The DE then produces a suspect list which contains information about which parts the DE believes are faulty.
- This event is sent to fmd, which routes it to any agents that may be registered for this event.
- Agents take action in response, which may include offlining a CPU or retiring memory pages.

FMA components

- Detectors
- Diagnosis engine
- Agents
- Log outputs



Diagnosis engine and agent

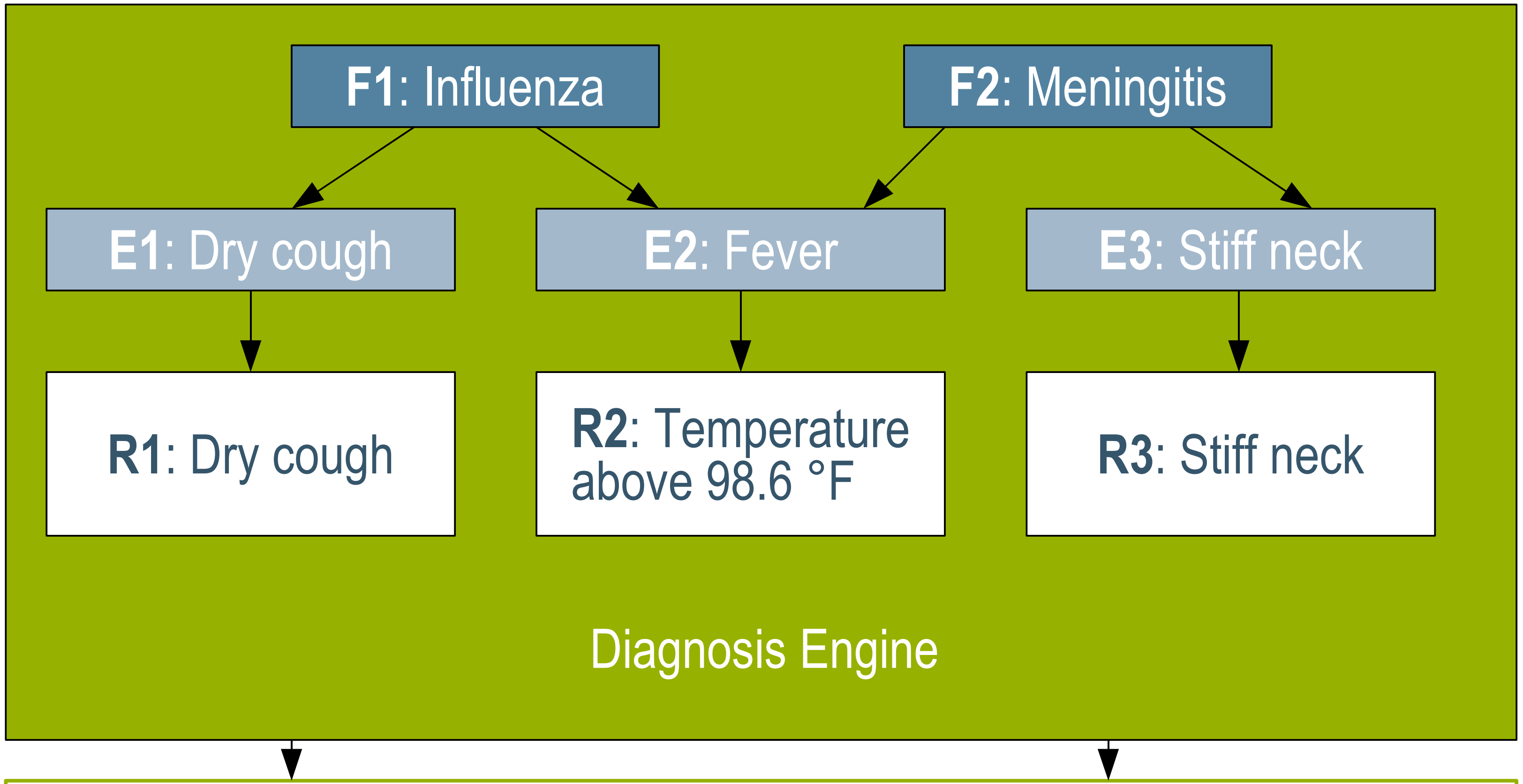


- fmd is central switchboard for DEs and agents
- Engines have fault tree and logic
- Retire agents have action to retire components
- In particular, syslog-msgs agent accept suspect list events, format info received and writes result to syslog, which sends it to messages file and/or console.

EFT engine

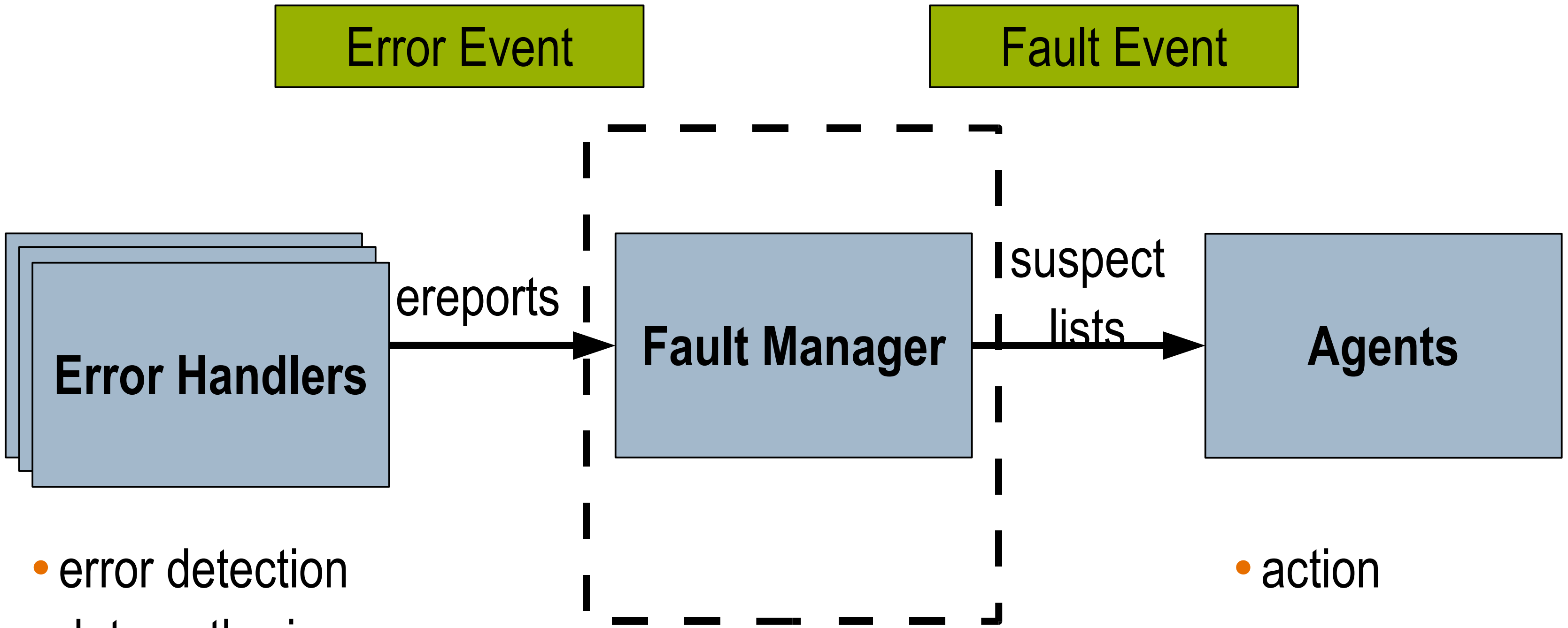
- The eft diagnosis engine is an engine which uses a special logic to determine the cause of faults.
- When a Sun product is designed, the entire system is mapped out in a fault tree that describes how errors may propagate through a system.
- When ereports are received by eft, it determines which part in the fault tree could be defective.
- After enough ereports are examined, eft send a suspect list with extremely detailed diagnosis information.
- This is an entirely different kind of diagnosis engine, unlike cpumem-diagnosis, for example

Fault Tree, DE, and suspect list



- Common cold
- Flu
- Avian Flu

Fault Management Architecture



- error detection
- data gathering
- error handling

- diagnosis
- event recording

- action

Message ID

```
SUNW-MSG-ID: SUN4U-8000-F2, TYPE: Fault, VER: 1, SEVERITY:
   Major
EVENT-TIME: Tue Nov  8 15:19:02 EST 2005
PLATFORM: SUNW,Sun-Fire-280R, CSN: -, HOSTNAME: atl-sewr-
   158-123
SOURCE: cpumem-diagnosis, REV: 1.3
EVENT-ID: ccf4d269-dd4e-e614-d6dd-82baaba6266d
DESC: The number of errors associated with this CPU has exceeded
   acceptable levels.  Refer to http://sun.com/msg/SUN4U-8000-F2
   for more information.
AUTO-RESPONSE: An attempt will be made to remove the affected
   CPU from service.
IMPACT: Performance of this system may be affected.
REC-ACTION: Schedule a repair procedure to replace the affected
   CPU.  Use fmdump -v -u <EVENT_ID> to identify the CPU.
```

http://sun.com/msg/SUN4U-8000-F2

Article for Message ID: SUN4U-8000-F2

CPU errors exceeded acceptable levels

Type	Fault
Severity	Major
Description	The number of errors associated with this CPU has exceeded acceptable levels.

Automated Response

The fault manager will attempt to remove the affected CPU from service.

Details

The Message ID: SUN4U-8000-F2 indicates diagnosis has determined that a CPU is faulty. The Solaris fault manager arranged an automated attempt to disable this CPU. The recommended action for the system administrator is to contact Sun support so a Sun service technician can replace the affected component.

Terminology

- FMRI - Fault Management Resource Identifier
  - > formal name for a resource for which Solaris
  - > example: cpu:///cpuid=0/serial=1132F212143
- UUID - Universal Unique Identifier
  - > unique identifier of a diagnosis for each resource
  - > Example: ccf4d269-dd4e-e614-d6dd-82baaba6266d
- Diagnosis engine will tag an FMRI with a state and a UUID, based on the suspect list.

Command overview

- fmd(1M) – fault manager daemon
- fmadm(1M) – administrative action
- fmdump(1M) – show logs
- fmstat(1M) – show module statistics
  - Diagnosis engines are loadable/unloadable modules.
  - They keep fault event statistics.
  - fmstat allows you to view these statistics and threshhold, more on this later.

(Some) Solaris 10 Security Features

Concept of least privileges

- Traditionally user root (UID 0) has unlimited privileges
- This leads to many applications running as UID 0, whether as root, or with suid bit, whether it needs all root privileges or not.
- Solaris 10 introduces Process Rights Management, privileges are broken up to 50 discrete privileges that can be granted to a process or revoked.
- To perform an operation (at the system call level) the process need certain privileges. This is a process attribute that is enforced by the kernel.

Privileges ppriv -l

contract_event	contract_observer	cpc_cpu
dtrace_kernel	dtrace_proc	dtrace_user
file_chown	file_chown_self	file_dac_execute
file_dac_read	file_dac_search	file_dac_write
file_link_any	file_owner	file_setid
ipc_dac_read	ipc_dac_write	ipc_owner
net_icmpaccess	net_privaddr	net_rawaccess
proc_audit	proc_chroot	proc_clock_highres
proc_exec	proc_fork	proc_info
proc_lock_memory	proc_owner	proc_priocntl
proc_session	proc_setid	proc_taskid
proc_zone	sys_acct	sys_admin
sys_audit	sys_config	sys_devices
sys_ipc_config	sys_linkdir	sys_mount
sys_net_config	sys_nfs	sys_res_config
sys_resource	sys_suser_compat	sys_time

(**BASIC set**; **global zone only**)

Examples

- file_dac_write and file_dac_read are privileges used to provide an otherwise unprivileged process with the ability to write or read any file on the system, regardless of its ownership or permissions.
- ppriv can be configured to attach to an existing process, or start a new one, with privilege debugging enabled using its -D option.
- This can get messy,so use privedebug, a Perl+DTrace program written to do this, get it here:

  http://www.opensolaris.org/os/community/security/files

Examples

- For entire listing of privileges

```
# ppriv -vl
```

- What privilege am I missing?

```
# ppriv -e -D touch /etc/acct/yearly
touch[8692]: missing privilege "file_dac_write" (euid = 23234,
syscall = 224) needed at ufs_iaccess+0xd2
touch: /etc/acct/yearly cannot create
#  ppriv -e -D cat /etc/shadow
cat[6286]: missing privilege "file_dac_read" (euid = 100, syscall =
225) needed at ufs_iaccess+0xd2
cat: cannot open /etc/shadow
```

Privilege debugging

- Using privdebug, you can follow and track all privileges used by an executable, a PID, or a zone.

```
# ./privdebug -n in.telnetd -f -v
STAT TIMESTAMP        PPID  PID   PRIV
      CMD
USED 1231183251139719 238   7115  sys_audit
      in.telnetd
USED 1231183251612259 238   7115  proc_fork
      in.telnetd
USED 1231183251974167 7115  7116  proc_exec
      in.telnetd
USED 1231183472328575 238   7115  proc_fork
      in.telnetd
```

Privilege debugging

- The idea is to start up an application
- Run privdebug and watch that pid or executable
- Put the application through its paces, watching what privileges are invoked using privdebug
- Then remove all the unrequired privileges so the process only has enough privilege to do its job.
- Setting SMF manifest and method can be tricky. See Glen Brunette's Blueprint called *Privilege Debugging*.

Process rights management

- This show the privilege of the shell.

```
# ppriv -S $$
```

- Remember the this is a process attribute and not a user attribute. When you run, for example

```
# usermod -K
  defaultpriv=basic,dtrace_kernel,dtrace_proc,
  dtrace_user someuser
```

- You are granting privilege to the user's shell, which is used to start up processes.
- SMF start services now, so modify privileges in SMF method directly.

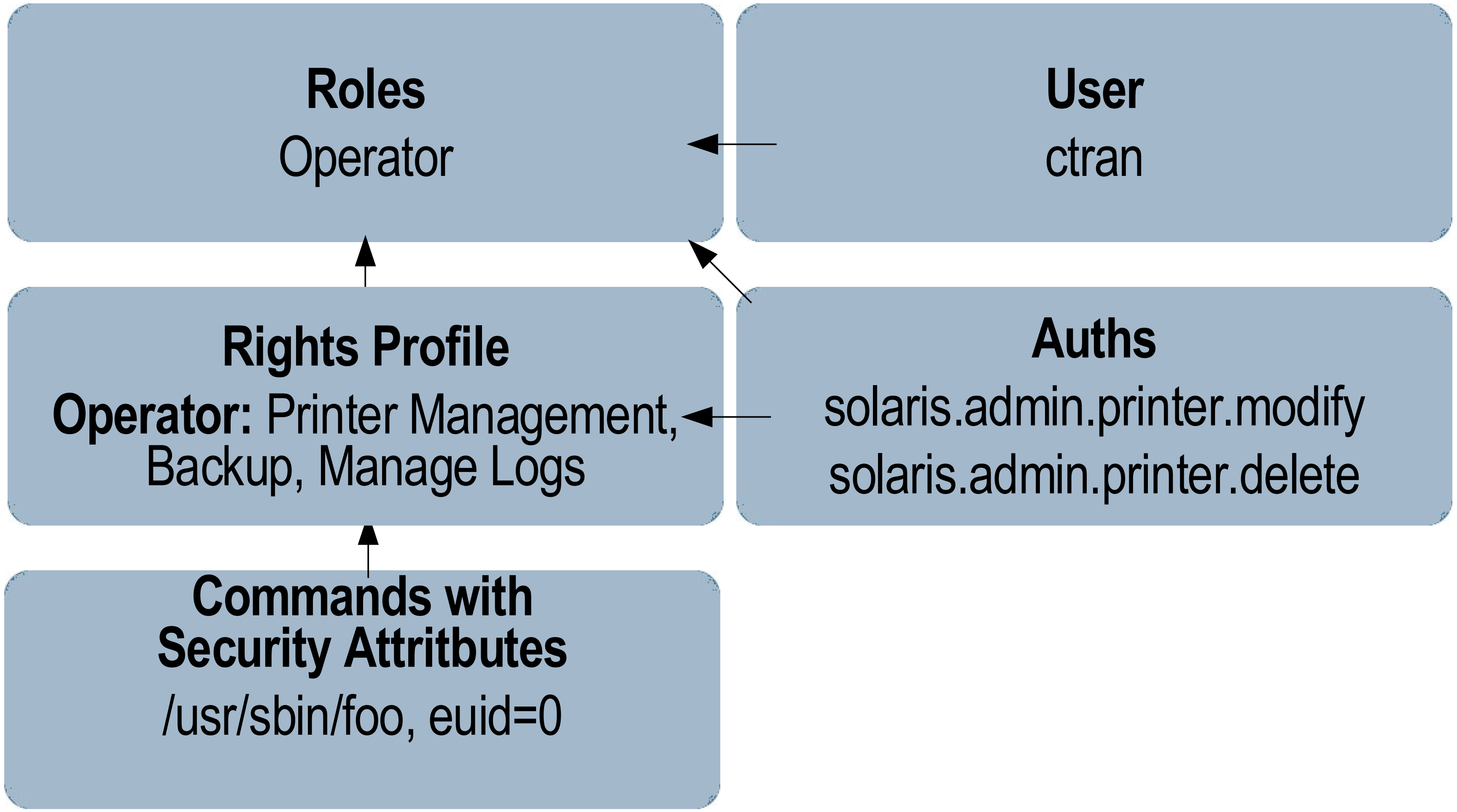
Role-Based Acces Control (RBAC)

- Introduced in Solaris 8, only getting more widely used in Solaris 10. Perhaps because of more integration?
- Doing everything as root is generally lazy and bad.
- Giving users root password is generally bad.
- Sudo was a popular tool to get around this problem, it was easy to use, portable, had logging mechanism, had some control granularity.
- RBAC does everything sudo does,and more!
- It is more complex, too!

Components of RBAC

- Users: a normal user
- Roles: like user, but cannot login. Users assume roles using “su”
- Authorizations: permission that can be assigned to a role or user to perform some action
- Rights: description privilege needed to run something as a privileged user, typically root
- Rights profile: collection of authorizations, commands, and other rights profiles

Component Interactions


Roles
Operator
User
ctran
Rights Profile
Operator: Printer Management, Backup, Manage Logs
Auths
solaris.admin.printer.modify
solaris.admin.printer.delete
Commands with Security Attritbutes
/usr/sbin/foo, euid=0

Location of files

- Users: /etc/passwd
- Roles: /etc/user_attr
- Authorizations: /etc/security/auth_attr
  - > Maps authorizations to descriptions
- Rights: /etc/security/exec_attr
  - > Maps rights to profiles
- Rights profile: /etc/security/prof_attr
  - > Maps profiles to authorizations and other profiles

RBAC Summary

- Lots of knobs, but once you see how they interact, it's not terribly hard.
- Users assume roles.
- Rights are defined, and then grouped into profiles.
- Profiles are assigned to roles.
- Authorizations are given to profiles, they can also be given directly to roles.

Basic Auditing and Reporting Tool

- A tool for tracking changes to files and filesystems over time
- Run once to establish a baseline.
- Run later to compare result with baseline.
- Can identify changes in attributes as well as content, and new or deleted files
- BART manifest (different from SMF manifest) is a catalogue of files, filesystems and their attributes taken at a specific time

Usage example

- As easy as
```
> bart create > control-manifest
```
- 6 months later,
```
> bart create > new-manifest
```
- Compare:
```
> bart compare control new
```
- Can insert rules file to ignore some expected changes, such as new patch, or directory timestamp
```
> Bart compare -r rules control new
```
- Used with zones, and reduced process privilege, can be a formidable defense.

ZFS

Existing Filesystems

- No defense against many modes of data corruption
- Lots of Limits: size, number of files, size of file
- Difficult to manage: fsck, /etc/fstab, partitions, volumes
- Too many things to tune
- Design Principles:
  - > End-to-End data integrity
  - > Huge capacity (128-bit)
  - > Simple to administer

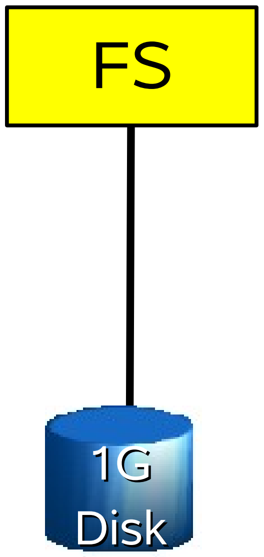
ZFS Design Goal

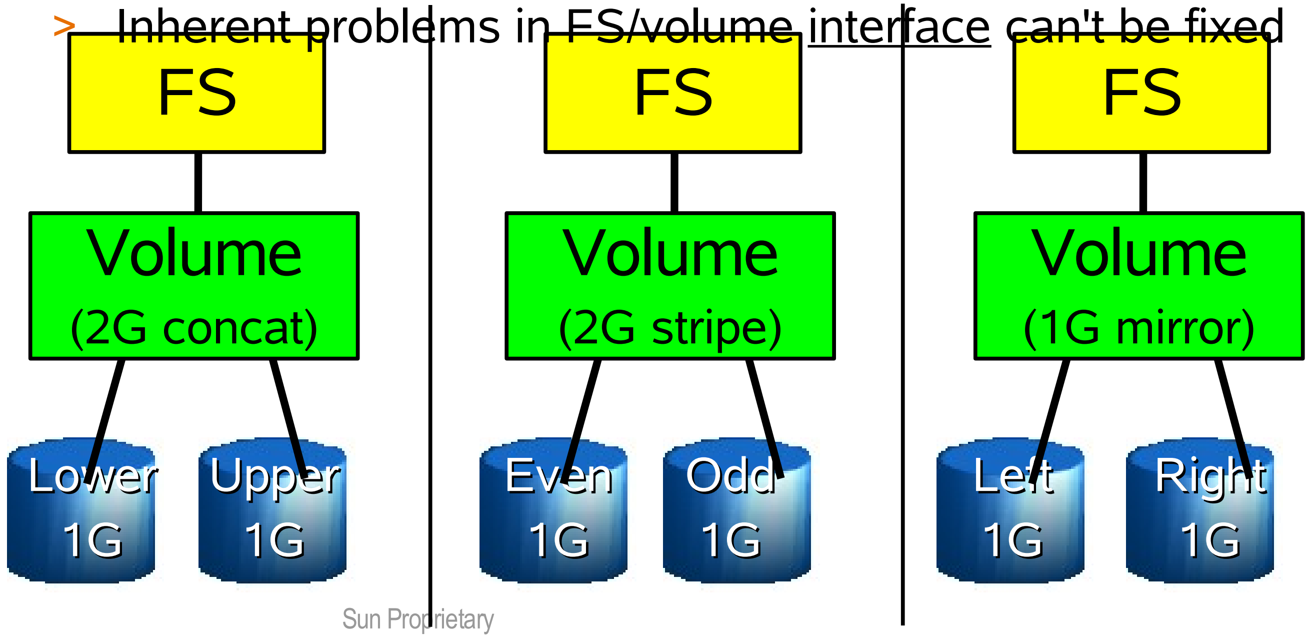
- Pooled storage
  - > Completely eliminates the antique notion of volumes
  - > Does for storage what VM did for memory
- End-to-end data integrity
  - > Historically considered “too expensive”
  - > Turns out, no it isn't
  - > And the alternative is unacceptable
- Transactional Operation
  - > Keeps things always consistent on disk
  - > Removes almost all constraints on I/O order
  - > Allows us to get huge performance wins

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

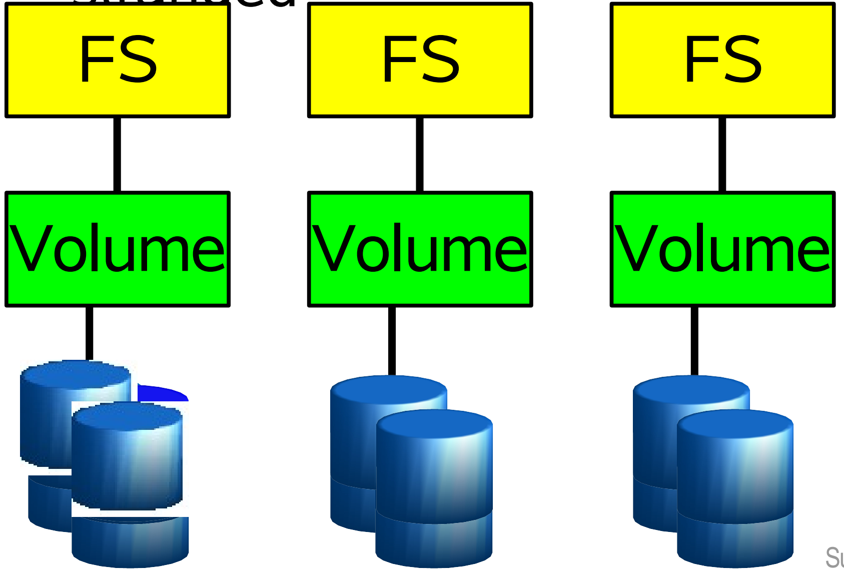




FS/Volume Model vs. ZFS

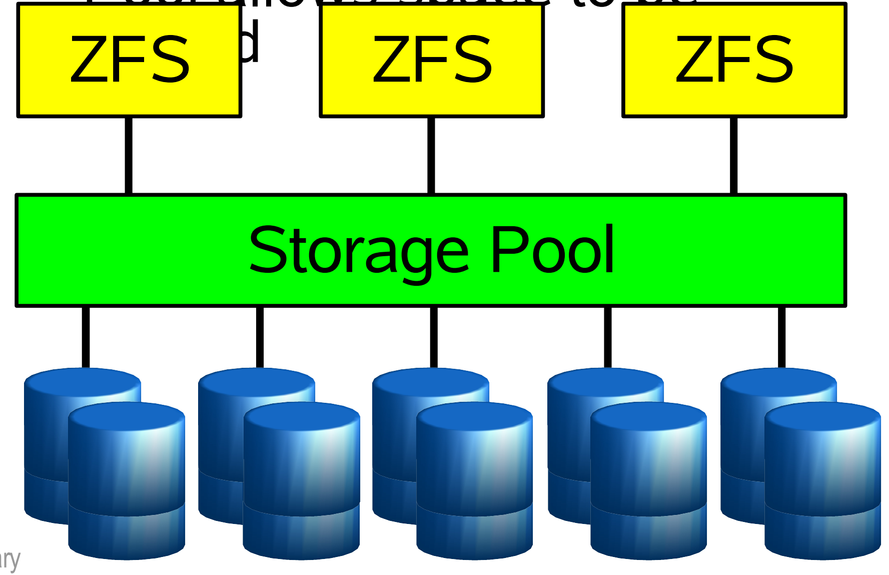
Traditional Volumes

- 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 [illegible]d



FS/Volume Model vs. ZFS

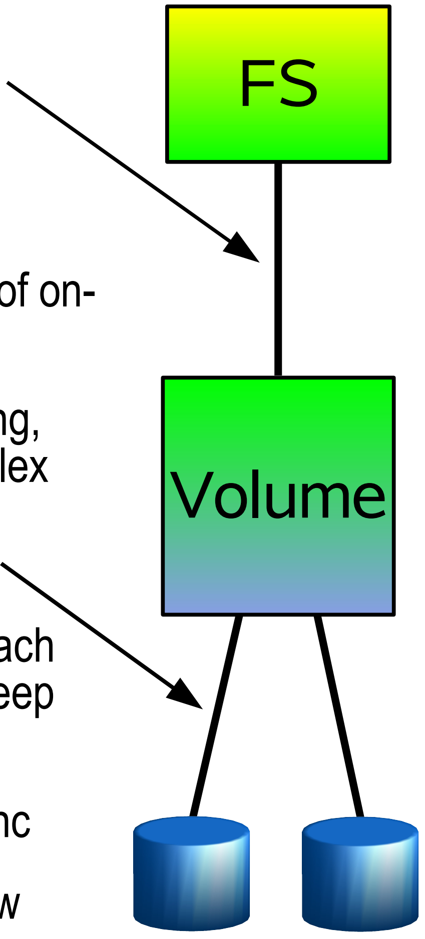
FS/Volume I/O Stack

Block Device Interface

- "Write this block, then that block, ..."
- Loss of power = loss of on-disk consistency
- Workaround: journaling, which is slow & complex

Block Device Interface

- Write each block to each disk immediately to keep mirrors in sync
- Loss of power = resync
- Synchronous and slow



ZFS I/O Stack

Object-Based Transactions

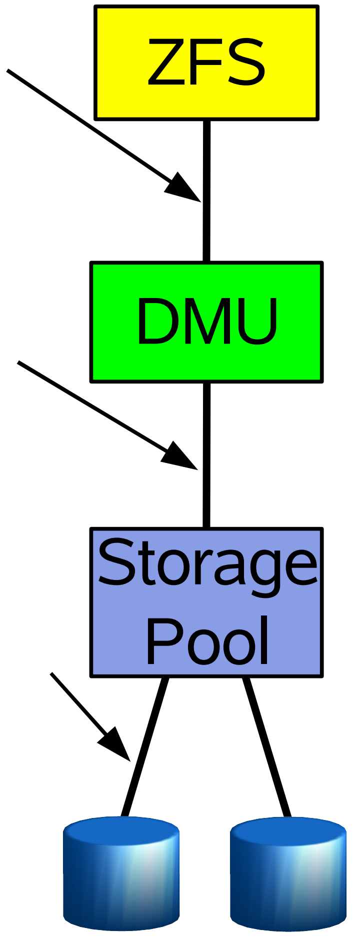
- "Make these 7 changes to these 3 objects"
- All-or-nothing

Transaction Group Commit

- Again, all-or-nothing
- Always consistent on disk
- No journal – not needed

Transaction Group Batch I/O

- Schedule, aggregate, and issue I/O at will
- No resync if power lost
- Runs at platter speed

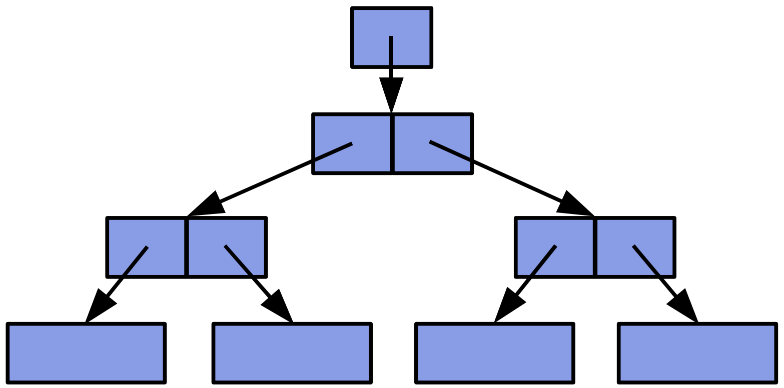


ZFS Data Integrity Model

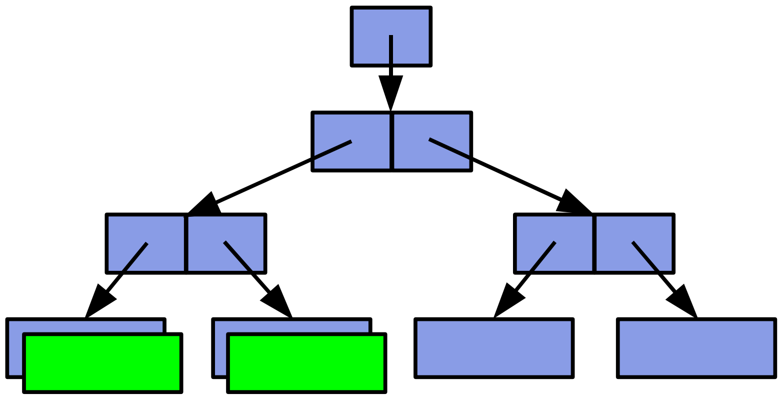
- All operations are copy-on-write
  - Never overwrite live data
  - On-disk state always valid – no “windows of vulnerability”
  - No need for fsck(1M)
- All operations are transactional
  - Related changes succeed or fail as a whole
  - No need for journaling
- All data is checksummed
  - No silent data corruption
  - No panics on bad metadata

Copy-On-Write Transactions

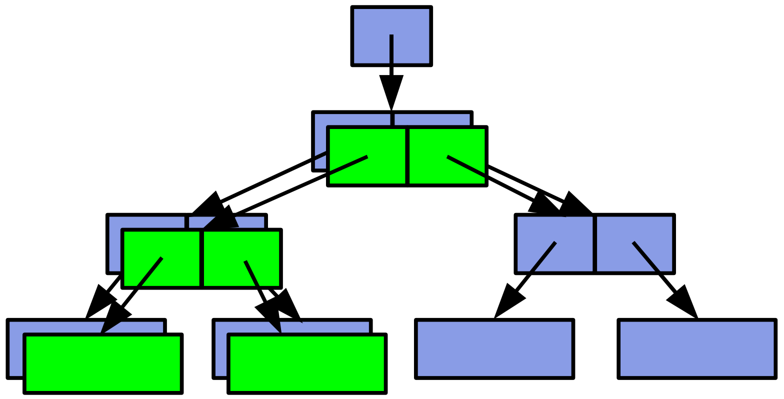
1. Initial block tree



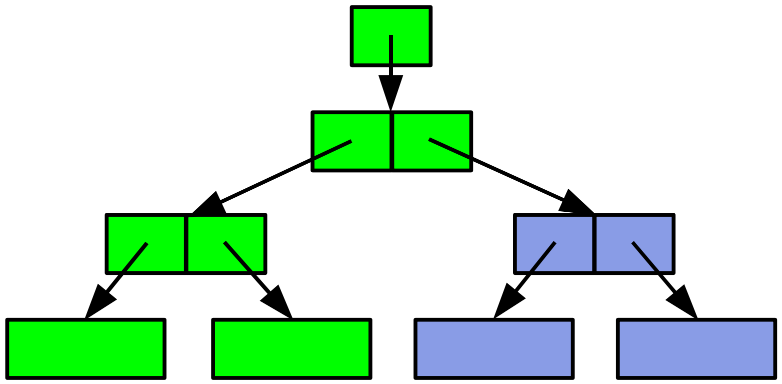
2. COW some blocks



3. COW indirect blocks

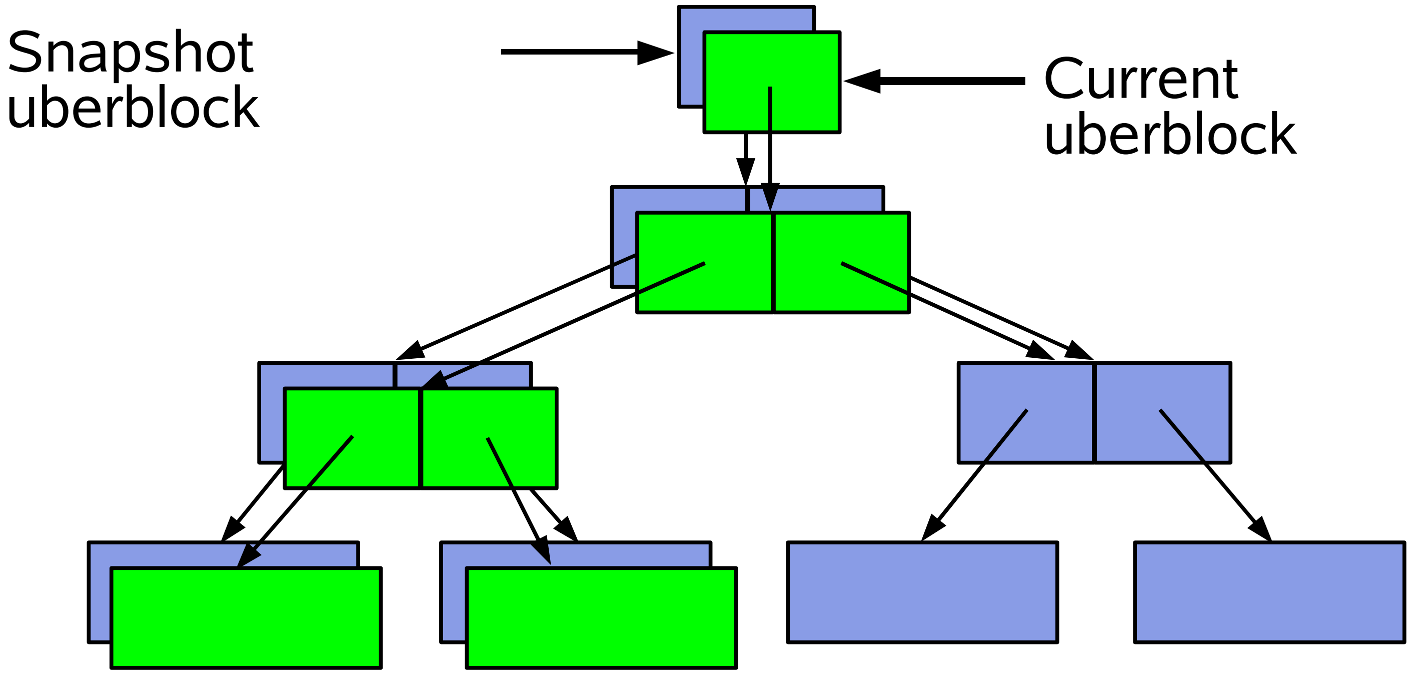


4. Rewrite uberblock (atomic)



Bonus: Constant-Time Snapshots

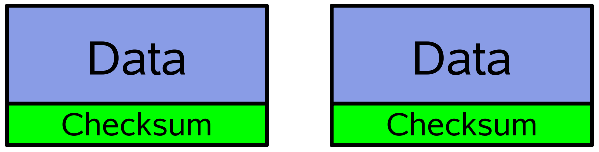
- At end of TX group, don't free COWed blocks
  - Actually cheaper to take a snapshot than not!



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

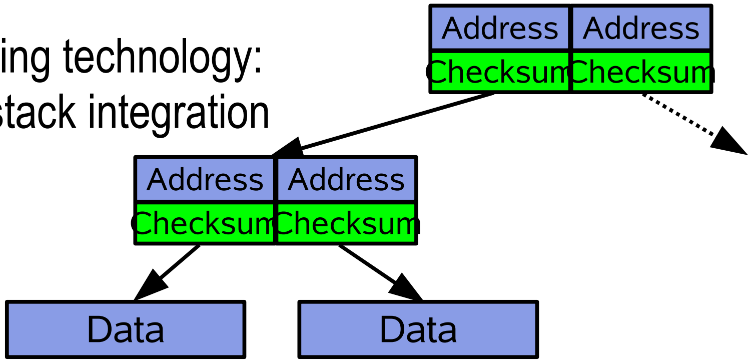


Only validates the media

✔ 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



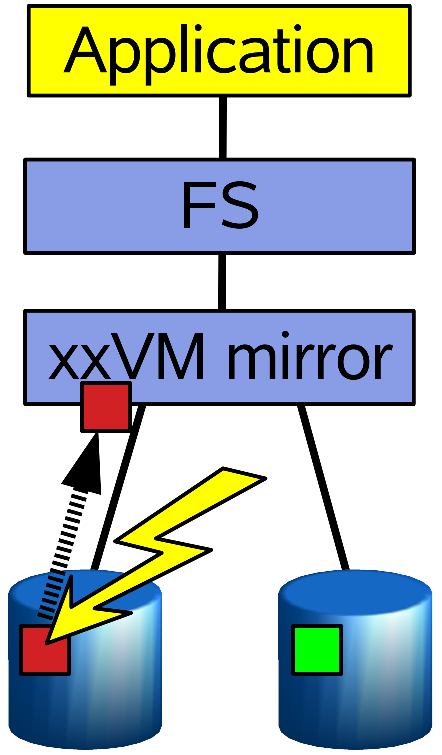
Validates the entire I/O path

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

Traditional Mirroring

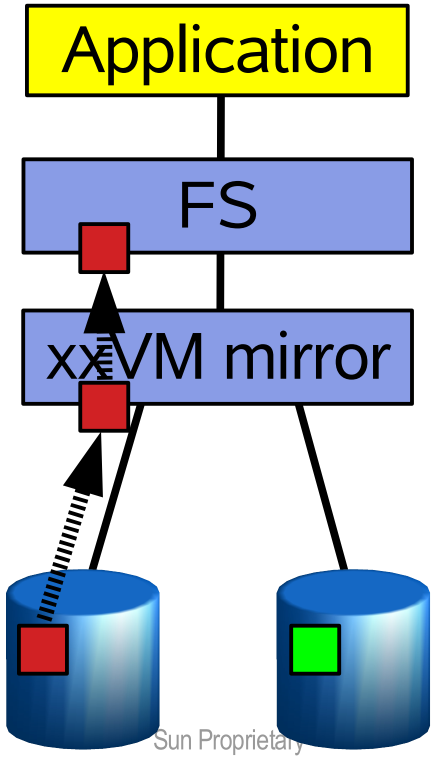
1. Application issues a read. Mirror reads the first disk, which has a corrupt block.

It can't tell.

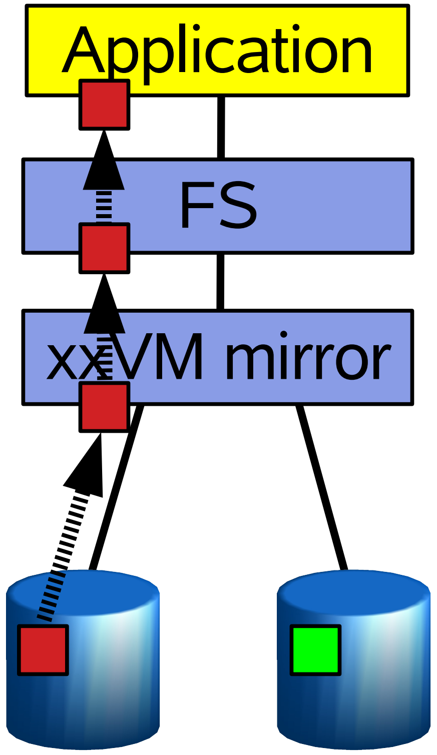


2. Volume manager passes bad block up to filesystem.

If it's a metadata block, the filesystem panics. If not...



3. Filesystem returns bad data to the application.

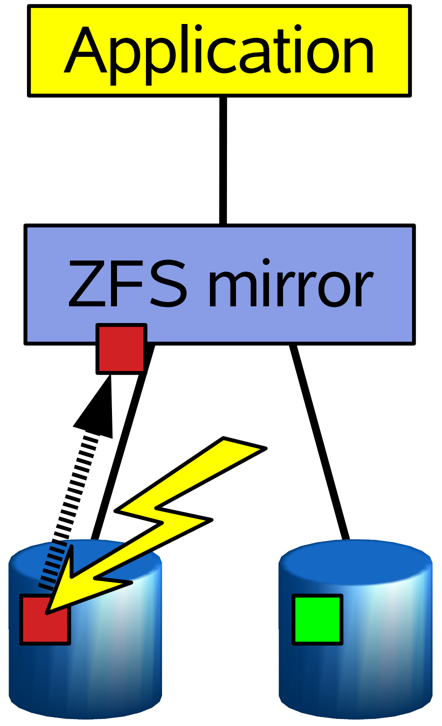


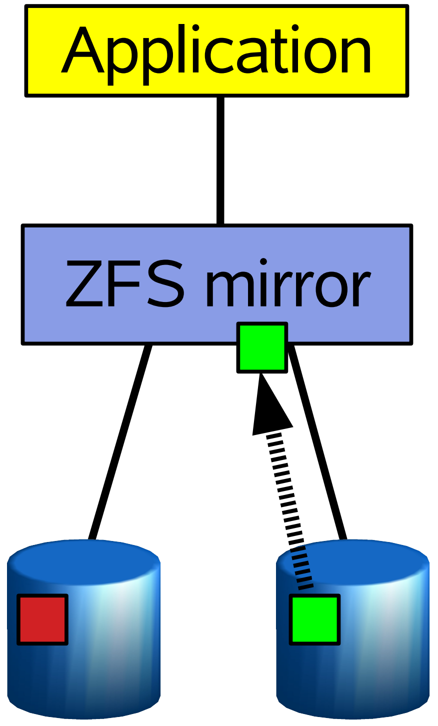
Self-Healing Data in ZFS

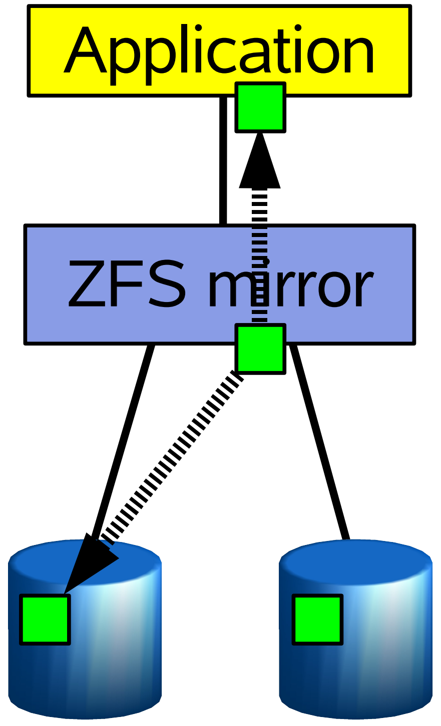
1. Application issues a read. ZFS mirror tries the first disk. Checksum reveals that the block is corrupt on disk.

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.







zpool(1M) – Manage storage pools

- create a storage pool
- add storage to an existing storage pool
- destroy a storage pool
- import a storage pool
- export a storage pool
- Observe a storage pool with status, iostat, list

zfs(1M) – Manage ZFS filesystems

- create a ZFS filesystem or virtual device
- create a snapshot of a ZFS filesystem or virtual device
- rollback a ZFS filesystem to a previous snapshot
- destroy a filesystem or snapshot
- set properties (mountpoint, quota, compression, etc) for a ZFS filesysem
- mount or unmount a ZFS filesystem

Demo

- ZFS demo

DTrace

Why Dynamic Tracing?

- Well-defined techniques exist for debugging fatal, non-reproducible failures:
- Obtain core file or crash dump
- Debug problem postmortem using mdb(1), dbx(1)
- Debugging transient failures is more difficult.
- Typical techniques push traditional tools (e.g. truss(1), mdb(1)) beyond their design centers
- Many transient problems cannot be debugged at all using existing techniques
- Where to go after truss ?

Debugging transient failure

- Historically, we have debugged transient failures using process-centric tools:
- truss(1), pstack(1), prstat(1), etc.
- These tools were not designed to debug systemic problems
- But the tools designed for systemic problems, (i.e., mdb(1)) are designed for postmortem analysis...

Post-mortem techniques

- One technique is to use postmortem analysis to debug transient problems by inducing fatal failure during period of transient failure.
- Better than nothing, but not by much:
  - Requires inducing fatal failure, which nearly always results in more downtime than the transient failure.
  - Requires a keen intuition to be able to identify a dynamic problem from a static snapshot of state

Invasive techniques

- If existing tools cannot root-cause transient failure, more invasive techniques must be used.
- Typically, custom instrumentation is developed for the failing program and/or the kernel.
- The customer reproduces the problem using the instrumented binaries.
- Requires either:
  - > running instrumented binaries in production or reproducing a transient problem in a development environment. Neither of these is desireable!
- Invasive techniques are slow, error prone, and often ineffective.

Dynamic Tracing

- Static probes are designed to collect a specific set of data
  - The questions are asked in advance of the problem
  - Extending the data collection would require a rebuild of the system
- Want to be able to dynamically modify the system to record arbitrary data.
- Must be able to do this on a production system.
- Must be completely safe - there should be no way to induce fatal failure

Introducing DTrace

- Solaris Dynamic Tracing (DTrace) framework introduced in Build 43 of Solaris 10 (Software Express).
- A typical system has more than 30,000 probes.
- Dynamically interpreted language allows for arbitrary actions and predicates.
- Can instrument at both user- and kernel-level.
- Powerful data management primitives eliminate need for most post-processing.
- Unwanted data is pruned as close to the source as possible.

Probes

- A probe is a point of instrumentation.
- A probe:
  - Is made available by a provider.
  - Identifies the module and function that it instruments.
  - Has a name.
  - Is assigned a integer identifier.
- A probe is uniquely identified by its provider:module:function:name

Providers

- A provider represents a methodology for instrumenting the system.
- Providers make probes available to the DTrace framework.
- DTrace informs providers when a probe is to be enabled.
- Providers transfer control to DTrace when an enabled probe is hit (fired).

Some providers examples

- The function boundary tracing (FBT) provider instruments every function entry and return in the kernel.
- The syscall provider can dynamically instrument the system call table
- The lockstat provider can dynamically instrument the kernel synchronization primitives (lockstat)
- The profile provider dynamically interrupts the system at a user-configurable rate

Some providers examples

- The vminfo provider can dynamically instrument the kernel “vm” statistics (vmstat)
- The sysinfo provider can dynamically instrument the kernel “sys” statistics (mpstat)
- The pid provider can dynamically instrument application code
  - > Function entry and return
  - > Instruction by instruction
- The io provider can dynamically instrument I/O events
- And more!

Listing probes

- Probes can be listed with the “-l” option to dtrace(1M)
- Can list probes
- from a specific provider with “-P provider”
- in a specific module with “-m module”
- in a specific function with “-f function”
- with a specific name with “-n name”
- A probe is defined as follows: provider:module:function:name

Actions

- Actions are taken when a probe fires.
- Actions are completely programmable.
- Most actions record some specified state in the system.
- Some actions change the state of the system system in a well-defined way.
- These are called destructive actions.
- Destructive actions are disabled by default.

D language

- D is a C-like language specific to DTrace, with some constructs similar to awk(1)
- Complete access to kernel C types, complete support for ANSI-C operators
- Global, thread local and clause local variables
- External variables (such as kernel symbols)
- Rich set of built-in variables
- Arrays, associative arrays, and aggregations
- Support for strings as first-class citizen

D script

- Complicated DTrace enablings become difficult to manage on the command line.
- dtrace -s will read commands from a script rather than stdin
- Alternatively, executable DTrace interpreter files may be created.
- Interpreter files always begin with:

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

Basic structure of a D script

```
probe description (provider:module:function:name)
/ predicate /
{
action statements
}
```

- The following script will trace the executable name upon entry into any system call:

```
#!/usr/sbin/dtrace -s
syscall::entry
{
trace(execname);
}
```

Predicates, or conditionals

- Predicates allow actions to only be taken when certain conditions are met.
- A predicate is a D expression.
- Actions will only be taken if the predicate expression evaluates to true.
- A predicate takes the form “/expression/” and is placed between the probe description and the action.

Predicate example

- Trace the pid of every process named “date” that performs an open(2):

```
#!/usr/sbin/dtrace -s
syscall::open:entry
/execname == “date”/
{
trace(pid);
}
```

Destructive actions

- Must specify “-w” option to DTrace.
- stop() stops the current process
- Use prun(1) to resume the stopped process
- raise() sends a specified signal to the current process
- breakpoint() triggers a kernel breakpoint and transfers control to the kernel debugger (kdb)
- panic() induces a kernel panic
- chill() spins for a specified number of nanoseconds

Aggregations

- Often a pattern in the data values is more useful than individual values themselves.
- Aggregation functions allow the observation of the trends and patterns in data values.
- Traditionally, one has had to use conventional tools (e.g. awk(1), perl(1)) and possibly create tables do be displayed by spreadsheet programs.
- DTrace provides a rich set of aggregation functions.

Aggregations

- Aggregations are stored in aggregation arrays
  - > Similar to associative arrays
  - > Denoted by @
  - > Does not have to be named
- By default, aggregation results are printed when dtrace(1M) exits.

DTrace in the real world

- Use the manual, it is excellent!
  http://www.sun.com/bigadmin/content/dtrace/d10_latest.pdf
- Examples in /usr/demo/dtrace
- DTrace toolkit:
  http://users.tpg.com.au/bdgcvb/dtrace.html
- DTrace one-liner
  http://users.tpg.com.au/bdgcvb/dtrace.html#OneLiners
- Documented scripts for use with typical problems, and exemplifying demos

Migrating to Solaris 10

Should I upgrade?

Yes!

How do I start?

- Download Solaris 10 – it's free!!!
  - Load on a spare system
  - Or set up a dual boot environment
  - Use Live Upgrade to preserve customizations
  - http://www.sun.com/downloads
- If you want to see where this may be going
  - Download Software Express for Solaris
  - It's free too!
  - http://www.sun.com/softwareexpress

Migration Process

- Inventory
- Certification
- Binary Compatibility
- Prioritizing
- Testing
- Pilot
- Deploy
- Re-iterate

Reason for migration

- Migration is a costly effort, consuming time, money, and staff. Before embarking on a migration effort, consider why you are making that effort.
- Consider cost:benefit ratio and ask if you are better staying where you are.
- Solaris 10 has many new features to help you improve your business, pick at least one and go in that direction.
- Adopting Solaris 10 follows the same path as any other operating system adoption or upgrade

What you need

- A Reason - A business need to be addressed by adoption
- A Sponsor – A decision-maker who can make adoption a priority activity for staff
- Assigned Staff – Needs to be someone's responsibility to make adoption happen
- Time – Certification & testing of any product take time. Be realistic with schedules.
- Training – Don't have to be expert before you begin, but baseline training will help remove obstacles and make the time used more effective

Define Goal and Scope

- Keep it small and focused, for example: all web servers for business XYZ excluding development.
- Get there with an iterative approach
- Start with a meaningful but small proof-of-concept
- Move to a production pilot
- Roll out to general production
- Testing and verification needed at each step
- Don't forget your DR environment

Why adopt a new OS?

- Technical and Operational Reasons
  - > Compelling new features

    How can the new OS improve my environment?
  - > Support for new hardware

    New hardware is not supported by old operating systems
  - > Vendor support

    Every OS follows a natural support roadmap that ends.
  - > Cleaner administration model

    New OS has old problems resolved.

    Many small bugs fixed but never issued as patches

Why adopt a new OS?

- Business Reasons
  - > "Cool" and "new" are not valid reasons for migration
  - > Start with a problem, not a solution.
  - > What business problem do you have that can be addressed with new features of the OS?
    - Consolidation?
    - Resource management?
    - Performance?
    - Data-center footprint?

Platform Inventory

- Hardware type
- Storage components and volume layout
- Failure recovery method
- Current operating environment and patch level
- System Administration practice: backup, system management, security, user administration, etc.
- Applications hosted on this platform
- Special notes for this platform

Target platform

- Decide whether to migrate to new hardware or re-use identical hardware
- If migrating to new hardware, have performance and capacity data ready to profile new hardware.

Application Inventory

- Application name and brief description
- Currently supported version
- Server where hosted and filesystem layout
- Application inter-connect (if a tiered application)
- Failure recovery method
- Application Administrators and tasks
- Special notes for this application

Application Certification

- Certification is the the risk-mitigating process of ensuring the application will work on Solaris 10 as it has on previous version of Solaris.
- Commercial-Off-The-Shelf (COTS) Application Certification:
  - > For COTS applications, the current version can be checked against an internal database that Sun maintains with ISV partners to determine if the application is ready for Solaris 10.

Application Certification

- Custom Internal Application
  - > Sun offers a Solaris Application Guarantee that says your existing application will run on Solaris 10 as it previously had on other versions of Solaris, even if it has not been recompiled for Solaris 10.
  - > There are some caveats, the application must have passed the binary compatibility test.
  - > Read more at

http://www.sun.com/software/solaris/guarantee.jsp

Solaris Application Guarantee

- Solaris Application Guarantee
  - > Solaris Binary Guarantee
    - applicable for binary when you move from one previous Solaris release to a new Solaris release
  - > Solaris Source Code Application Guatantee
    - applicable for source when you move from SPARC to x86/x64 or from x86/x64 to SPARC
- Binary compatibility is based upon a system's Application Binary Interface (ABI), which is the set of supported run-time interfaces available for applications to use.

Tools to check Binary compatibility

- Sun has developed two tools to check an application's compliance: *appcert* and *apptrace*. They are part of the OS since Solaris 8.
- Run Solaris Application Scanner (ABIScan) to detect if an application uses an interface or library that has been removed on Solaris 10. Get it at

  http://www.sun.com/software/solaris/faqs/abiscan.xml

Prioritizing

- The business need will be the prioritizing factor to determine which application will be first in the migration pipeline.
- Obvious choice might not be the best choice:
  - > Very old server running very old OS
  - > Probably because of application dependency
  - > Probably because very near EOL
  - > Probably because of low-impact to business
  - > This could be done at later iterations

Business as Usual Adoption

- Solaris allows you to maintain virtually everything in place.
- New features are either transparent (new device drivers, performance gain) or optional (zones, DTrace)
- Service Management Facility is the most obvious mandatory change
- Separate the ISV certification from operational function

Deploying a test environment

- Hardware platform
  - > New or current
  - > If new, then worry about sizing
  - > Firmware
- Operating Environment
  - > Upgrade or new install?
  - > Determine the OS release, package cluster
  - > Determine installation process (jumpstart, flars)
  - > Organizational security scrutiny and approval
- Application Installation
- Can you still do routine System Administration task?

Testing and Deployment

- Develop success criteria and procedure to validate your application.
- Functional test

  Does my application work correctly?
- Performance test

  Is my application running as well or better than before?
- Integration test (for tiered applications)

  Does my application behave well with other applications?
- Deploy a small pilot and then broaden the scope.

What next?

- Adopt a business-as-usual approach
- Define business driver and key Solaris 10 feature
- Identify the hardware and/or the application to migrate
- Start an inventory matrix
- Consider impacts to your operations processes, training requirements, etc.
- Let Sun help you!

opensolaris™

- Solaris is now open sourced, all development is open, under CDDL (Common Development and Distribution License).
- Communities at http://opensolaris.org/os
- Documentation, discussions and archives, and bleeding edge stuff are all there.
- Bugs and fixes are open, users can submit bug reports, contribute code to projects.
- Multiple opensolaris distros already released.

SOLARIS 10 BOOTCAMP

ADELPHI, MD - 2/27/2007

christine.tran@sun.com

Sun Microsystems