

# *New Features in Solaris 10*

Jeff Muse

St. Louis Unix User's Group

June 8, 2005



# *What's New in Solaris 10?*

- Predictive self healing
  - Zones
  - Dtrace
  - Updated TCP/IP stack including ipfilter
  - Improved x86 support
  - NFS v4
  - Java Desktop System
  - Improved Open Source Support
- 
-

# *What Didn't Make It?*

- ZFS – new and improved file system
- Janus – Linux binary compatibility

# *What About Open Solaris?*

- Scheduled to be released Q2 2005
- Under CDDL
  - Common Development and Distribution License
  - Based on Mozilla Public License
  - Existing open-source projects retain their standard licenses (e.g. Perl, Apache, GNU software)

# *Tour of the the JDS*

- A GNOME by any other name...



# *Predictive Self-Healing*

- Hardware element
  - Find and take failing hardware offline
  - Available in new Sun hardware – firmware support required
- Software element
  - SMF (Service Management Facility)



# *SMF (Service Management Facility)*

- Restart failed services in dependency order
- Parallel service startup
- Monitors state of services



# *Changes Resulting From SMF*

- Quieter Boot Process
    - Use “boot -m verbose” at OB prompt
  - Failed services placed in maintenance mode – beware dependencies!
  - Services ended via kill(1) will restart
  - End services with svcadm instead of kill(1)
  - Many scripts removed from /etc/init.d, /etc/rc\*.d, /etc/inittab
  - 3rd-party startup scripts continue to run
  - “Milestones” replace concept of runlevels
  - Inetd applications are integrated into SMF
- 
-

# *SMF Services*

- Service instance – one configuration of one service
  - Services provide capabilities to applications or other services
  - May include network device or configuration, kernel configuration, init state, or daemons
  - View with “svcs -l”
- 
-

# *Service Identifiers*

- FMRI – Fault Management Resource Identifier
  - FMRI = service name + instance name
  - Service names may include:
    - application
    - device
    - milestone (init state)
    - network
    - platform
    - site
    - system
    - legacy init scripts
  - View with “svcs -a”
- 
-

# *SMF Components*

- `svc.startd` – restarter daemon
  - service manifest – xml file with properties for service or service instance – located in `/var/svc`
  - smf profile – xml file listing services started at boot – located in `/var/svc`
- 
-

# SVCS

- No flags - list available services
  - `-l service` – details for specified service
  - `-p service` – list processes associated with *service*
  - `-d service` – list services required by *service*
  - `-D service` - list services that depend on *service*
  - `-xv service` – list service state and dependent services
- 
-

# *svcadm*

- `svcadm -v enable -rs ssh` – enable ssh and all dependencies ( recursive flag ) and wait for service to come up ( `-s` )
- `svcadm -v enable ssh` – enable sshd
- `svcadm -v restart ssh` – restart sshd
- Log file: `/var/svc/log/*`

# *Milestones*

- Replacement for runlevels
- Logical group of services
- View associated services with “svcs -d milestone-name”
- “svcadm -d milestone milestone-name” sets default milestone



# *Inetd*

- `inetadm` command manipulates `inetd` services
- `-l service` – list properties of *service*
- `-d service` – disable *service*
- `-m service [property=setting]` – configure options (e.g. `tcp wrappers`)

# *Zones*

- Allow multiple virtual machines to run on physical machine
- Processes on one machine cannot see or affect with processes on another



# Zone Types

- Global -
    - Only bootable zone
    - Only system from which other zones can be administered
    - Only zone aware of all devices, file systems, processes
  - Non-global
    - Can be viewed and administered from global zone
    - Each non-global zone is invisible to other zones
    - By default /lib, /usr/, /platform, and /sbin are mounted as read-only loopback filesystems from global zone (sparse root zone)
- 
-

# *System Requirements*

- Minimum 100M of disk space per non-global zone, assuming complete install in global zone
- Can install entire OS into non-global-zone ( whole root zone)
- Can limit size via disk partitions or loopback filesystems

# *Zone Commands*

- `zonectfg` – configure a zone before installing
- `zoneadm` – install zone, boot zone, halt zone, list installed zones
- `zlogin` – log in to zone
- You need to boot zone before logging into it!

# Creating a Zone

```
# zonecfg -z zone-name
> create
> set zonepath=path
> set autoboot=true
> add net
    > set physical=interface name
    > set address=IP address
    > end
> remove inherit-pkg-dir dir=/usr
> verify
> exit
> zoneadm -z zone-name install
```

# *Running Zones*

- Boot virtual machine “zoneadm -z zone boot”
- Login to console of virtual machine “zlogin -C zone”
- sysidnet starts – allows configuration of IP/DNS/hostname/root password
- Default route picked up automatically
- Interfaces in non-global zone appear as virtual interfaces (eri0:1)

# *Zoneadmd*

- Daemon which manages virtual machine
  - Allocates zone devices
  - Manages file systems
  - Manages zone console device
  - zsched – zone scheduler – scheduler for zone processes

# *Managing Software*

- Packages can be:
  - Added/removed/queried in global zone only
  - Added/removed/queried in global zone and all non-global zones
  - Copied from global zone to all non-global zones
  - Added/removed/queried in specified non-global zone
- To add a package to non-global zone, filesystems must be mounted read-write!

# *Zone Restrictions*

- Non-global zones cannot be NFS servers
- Sharing external devices between zones can be a security risk
- Zones cannot access kernel space – no driver loading/unloading, accessing physical hardware, etc.
- The non-global zone directory in the global zone's file system should not be written to.
- Devices can not be created via mknod in non-global zones

# *Dtrace (Dynamic Tracing)*

- “Truss on steroids”
- Allows user to look inside specified process
- Requires knowledge of how application is supposed to work
- Scripted through language called “D”
- Can be used safely on live systems with no need for debugging symbols

# *Probes*

- Probes are sensors scattered throughout the operating system
  - Each probe has a unique integer ID and a human-readable name
  - Each probe, if called, will record the time it fires or take specified action
  - View with “dtrace -l”
  - First two probes are “BEGIN” and “END”
- 
-

# *Probe Names*

- There are four parts to a probe name, separated by colons
  - Provider:Module:Function:Name
    - Provider is usually a dtrace kernel module
    - Module is the name of kernel module or user library.  
Used only if probe corresponds to a specific program.
    - Function is used if the probe corresponds to a specific function in a specific program
    - Name is descriptive
  - Dtrace will expand any blank fields to include all matches based on what you've listed.
- 
-

# *Programming in D*

- A D script is a group of clauses
  - Each clause specifies probe(s) to enable
  - Each statement ends in a semicolon
  - Each probe may have a set of actions enclosed in braces (`{ }`). If there are no specified actions, the braces are still required!
  - When `“/usr/sbin/dtrace -s”` is called, it compiles your code and sends it to the kernel.
- 
-

# *Hello, World*

- Uses trace() to display a statement (“Hello world”) when the specified probe (BEGIN) fires.
- Output specifies CPU, probe ID, function, and name of probe.

# Counter

- Basic counting program
- Increment a variable
- To count in milliseconds, change the provider

# *Predicates*

- Predicates are what other languages would call conditionals
- Tests are enclosed in slashes
- Example: Countdown script

# *Tracing System Calls for a PID*

- Watch entry and return of system calls.
- We can specify a PID on the command line
- “PID” is a built-in variable in D



# Formatting

- D supports `printf()`, just like in C!
- We can specify quiet formatting on the command line
  - `/usr/sbin/dtrace -q -s script`
- Example: `trussrw.d`

# *Timing Syscalls*

- `rwtime.d` uses an array to track a variable
- `rwtime.d` also uses D's built-in nanosecond counter



# *What's the Kernel Doing?*

- kernel.d shows all activity visible to :genunix::

# URL's

- CDDL:
  - [http://www.sun.com/cddl/CDDL\\_why\\_summary.html](http://www.sun.com/cddl/CDDL_why_summary.html)
- -SMF:
  - <http://docs.sun.com/app/docs/doc/817-1985/6mhm8o5n>
- Zones:
  - <http://docs.sun.com/app/docs/doc/817-1592/6mhahuons>
- Third-party articles:
  - <http://learningsolaris.com>
- This presentation:
  - <http://sluug.org/~jmuse/presentations/solaris-10.pdf>

