Paper 225-30

SAS®9 on Solaris™ 10: The Doctor is "In" (and Makes House Calls)

Maureen Chew, Sun Microsystems

Abstract : *As physicians and hospital staff in our healthcare system are stretched by thin budgets and personnel shortages, individuals are being required to take their own healthcare into their hands by becoming mini-experts in specific health problems, often suggesting their own courses of treatment and medications. A similar trend is seen among SAS® user communities faced with extremely constrained IT budgets and IT staff reductions. This paper targets both the Solaris novice and expert communities of SAS users who need to understand and diagnose application and system performance heuristics. SAS users new to or not yet familiar with Solaris/UNIX environments will learn basic tools for characterizing 75% of normal everyday performance bottlenecks. But the more experienced users will see the power of these unique new features of* ***Solaris 10*** *and how these leading edge technologies can be brought in house, without requiring years of training. We'll examine how these industry-heralded features such as* ***Zones, DTrace(Dynamic Tracing) and ZFS*** *can provide both vaccination-like immunities and/or rapid, full circle diagnostic through treatment capabilities to dreaded application and system performance pains. And as if that weren't enough--quickly and cheaply. You, the doctor, "is in"!*

The much anticipated release of Sun's Solaris 10 in February '05, represents a tremendous amount of innovation. Coupled with the Solaris open source release, there is no doubt that widespread adoption of Solaris 10 will follow. Solaris 10 is an excellent platform for deployment of SAS 9 applications.

We discuss several of the new Solaris 10 innovations but have to exclude many other important features such as the numerous performance enhancements, predictive self-healing framework and, a hot topic in today's compliance based regulations, security. A major new feature of Solaris 10 is the integration of the Process Rights Management model from Trusted Solaris as well as the enhancement of the cryptographic framework.

The goal of the paper is to give the reader of how to leverage some of these new features by giving you a cookbook recipe for trying them out yourself rather than leaving it as an exercise in your time constrained days.

Topics Covered:

- Standard Solaris tools (prstat, iostat, mpstat, vmstat)
- Solaris Containers*
- DTrace* (Dynamic Tracing)
- Service Management Facility*
- ZFS*
- Bonus: Tribal Wisdom artifacts on SAS 9 deployments

* New to Solaris 10

Standard Solaris Tools (prstat, iostat, mpstat, vmstat)

A few instruments in any physician's bag will be the stethoscope and knee reflex tool. Low tech but tried and true. Similarly, the utilities above are not new to Solaris 10, but comprise a basic set of command line tools that a Solaris novice can use to understand 80-90% of the basic system performance heuristics. With the exception of iostat(1), a sampling time interval (in seconds) is sufficient as the only argument for basic usage.

prstat(1) (Process STATus) gives a good snapshot of the top running processes and/or insight into an individual process.

```
bash-2.05b$ prstat 5
   PID USERNAME  SIZE   RSS STATE  PRI NICE      TIME  CPU PROCESS/NLWP
```

```
 23844 sasmau     103M   92M cpu17    0     0    0:02:01 4.1% sas/5
 23892 sasmau    4904K 4640K cpu9     59    0    0:00:00 0.0% prstat/1
  5907 root        90M   57M sleep    59    0    0:03:03 0.0% java/15
 20801 root        91M   24M sleep    59    0    0:04:59 0.0% poold/8
 22868 sasmau    2856K 2072K sleep    59    0    0:00:00 0.0% bash/1
   406 daemon    2392K 1792K sleep    60  -20    0:00:00 0.0% nfsd/2
   441 smmsp     5976K 1656K sleep    59    0    0:00:13 0.0% sendmail/1
```

To look at the above SAS process in more detail:

```
bash-2.05b$ prstat -Lm -p 23844
   PID USERNAME USR SYS TRP TFL DFL LCK SLP LAT VCX ICX SCL SIG PROCESS/LWPID
 23844 sasmau   100 0.0 0.0 0.0 0.0 0.0 0.0 0.0   0  28   0   0 sas/3
 23844 sasmau   0.0 0.0 0.0 0.0 0.0 0.0 100 0.0 152   0 174   0 sas/2
 23844 sasmau   0.0 0.0 0.0 0.0 0.0 100 0.0 0.0   0   0   0   0 sas/6
 23844 sasmau   0.0 0.0 0.0 0.0 0.0 100 0.0 0.0   0   0   0   0 sas/5
 23844 sasmau   0.0 0.0 0.0 0.0 0.0 100 0.0 0.0   0   0   0   0 sas/1
```

iostat(1) (I/O STATus) has a lot of functionality (in other words, it requires an obtuse set of arguments to be practically useful). A useful set is -cmnxz 5, but you may want to experiment with the different options. A mnemonic always helps so for this set, try: Cee My New Xtreme Zfs_file_system(-cmnxz).

```
bash-2.05b$ iostat -cmnxz 5
     cpu
 us sy wt id
  4  3  0 93
                    extended device statistics
    r/s    w/s   kr/s    kw/s wait actv wsvc_t asvc_t  %w  %b device
    0.0    0.4    0.0     3.4  0.0  0.0    0.0    7.3   0   0 c1t0d0
    0.0    0.2    0.0     0.1  0.0  0.0    0.0    8.5   0   0 c1t1d0
    0.0  175.0    0.0 143365.9  0.0 12.8    0.0   73.2   0 100 c3t216000C0FF804371d3
```

Columns to watch over an extended period of time: asvc_t(service times) and %b(usy).

mpstat(1) (Multi Processor STATus) reveals the individual CPU utilization on multi-processor systems. The output below shows plenty of system capacity as there is low idle time only for processors 8 and 529. Thus, the majority of the 24 processes (not all shown) are idle.

```
bash-2.05b$ mpstat 5
CPU minf mjf xcal  intr ithr  csw icsw migr smtx  srw syscl  usr sys  wt idl
  0    0   0   36   304  202  126    0    2    4    0   335    0   0   0 100
  1    0   0    2     4    2   89    0    2    0    0   298    0   0   0 100
  2    0   0    2     4    2  128    0    3    1    0    93    0   0   0 100
  3    0   0    2     4    2   56    0    3    0    0    48    0   0   0 100
  8   26   0    4    25    2   13   20    1    3    0  7598    2  43   0  55
 ....
512    1   0  208     4    1   58    0    1    2    0   346    1   0   0  99
513    0   0    1     4    1   11    0    1    0    0   186    1   0   0  99
514    0   0    1     5    3    4    0    1    0    0   186    1   0   0  99
515    0   0    1     4    1    5    0    1    0    0   184    1   0   0  99
.....
528    0   0    1     4    1   25    1    1    1    0   191    8   0   0  92
529    0   0    1     8    1    2    5    1    0    0   184   80   0   0  20
530    0   0    1     4    1    4    1    1    0    0   184    4   0   0  96
531    0   0   28     4    1   19    0    0    0    0   186    1   0   0  99
```

vmstat(1) is used to monitor the virtual memory subsystem and higher than normal paging activity can be indicative of improper resource utilization. Look for high scan rates (sr column); the -p option can give you a detailed breakdown.

```
bash-2.05b$ vmstat 5
 kthr      memory            page            disk          faults      cpu
 r b w   swap  free  re  mf pi po fr de sr s0 s1 s3 sd   in   sy   cs us sy id
 0 0 0 42868920 45628400 13 30 7 5 4  0  0  1  1  0  0  452 1561  396  1  0 99
 0 0 0 40590640 39650864 17620 14 0 0 0 0 0  1  0  0  0  599 9835  660  4  2 93
 0 0 0 40590640 39650872 17871 0 0 0 0 0 0  5  0  0  0  628 9935  675  4  2 93
 0 0 0 40590400 39650576 17852 59 0 2 2 0 0 0  0  0  0  599 12287 667  5  3 93
```

```
bash-2.05b$ vmstat -p 5
     memory           page          executable      anonymous      filesystem
   swap  free  re  mf  fr  de  sr  epi  epo  epf  api  apo  apf  fpi  fpo  fpf
```

```
42868904 45628360 13 30 4  0    0     0     0     0     0     0     0     6     5     4
40590616 39650848 17669 14 0 0 0     0     0     0     0     0     0     0     0     0
40590848 39650928 13147 1 78 0 0     0     0     0     0     0     0     0    78    78
40590376 39650688 17942 59 2 0 0     0     0     0     0     0     0     0     2     2
```

Solaris Containers

Think of the large medical centers usually located with major universities. If a patient needs both a cardiologist and pulmonary specialist, services are consolidated under the same system making it easier for the patient to find comprehensive and integrated treatment options. Similarly, Solaris Container technology allows for the virtualization of system resources to give the appearance of having multiple ***independent*** operating environments under a ***single*** Solaris instance! Solaris Containers functionality comprises of two main components, Solaris Zones and Resource Management. Zones enable the administrator to create separate environments for running applications, while the Resource Management framework provides for policy based allocations to resources such as CPUs and memory. Each Zone has its own network identity and ***appears***, for all intents and purposes, like a completely different instance of the Solaris Operating Environment.

Reasons to use Solaris Containers:

- A less secure application such as an external web server might run in one zone, while business critical intranet applications run in another.
- Separation of development, test and production environments
- Many applications such as SAS 9 may utilize a number of shared resources such as network ports. If you have development and test groups who all want to use port 8561 for the SAS Metadata Server, they can do so trivially in their own zone
- Non global zone reboots do not affect other zones.

All these reasons provide for much better resource utilization leading to greater efficiencies and lower costs. Solaris containers are ideal for a SAS 9 deployment.

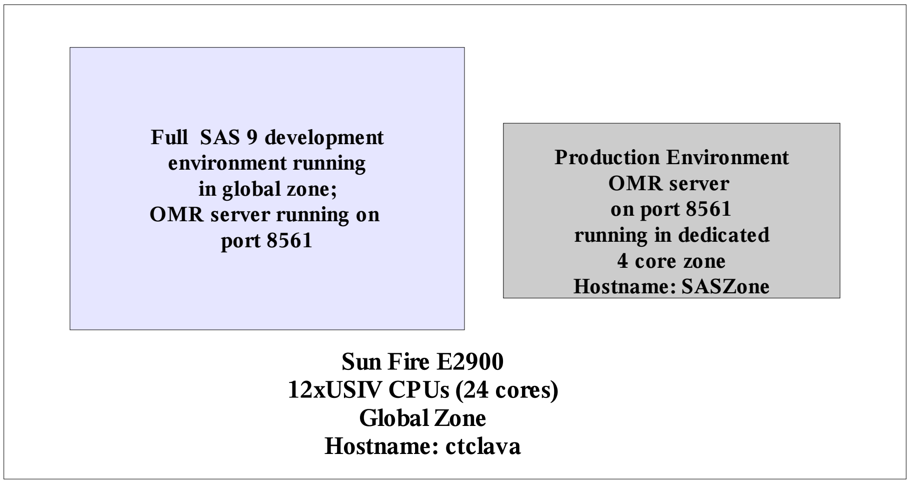
Full SAS 9 development environment running in global zone; OMR server running on port 8561
Production Environment OMR server on port 8561 running in dedicated 4 core zone Hostname: SASZone
Sun Fire E2900 12xUSIV CPUs (24 cores) Global Zone Hostname: ctclava

Fig 1: Example config for Metadata Server in dedicated 4 core zone

Only two major steps are needed to create the container – i) Create the resource pool and ii) Create the zone.
After creating the pool configuration file (/poolcfg/sas-metadata-pool.txt), only these 3 commands are necessary to create the resource pool.

- pooladm -e
- poolcfg -f /poolcfg/sas-metadata-pool.txt
- pooladm -c

Looking at this in further detail:

# Enable the pools facility
```
root@ctclava # pooladm -e
```

```
root@ctclava # cat /poolcfg/sas-metadata-pool.txt
create pset SASpset (uint pset.min = 1; uint pset.max = 4)
create pool SASpool (string pool.scheduler="TS")
associate pool SASpool (pset SASpset)
```

# Configure the pool
```
root@ctclava # poolcfg -f /poolcfg/sas-metadata-pool.txt
```

# Activate the configuration
```
root@ctclava # pooladm -c
```

# Display the newly modified configuration
```
root@ctclava # pooladm
system ctclava
        string  system.comment
        int     system.version 1
        boolean system.bind-default true
        int     system.poold.pid 20801
        pool pool_default
                int     pool.sys_id 0
 ...            pset    pset_default
        pool SASpool
                int     pool.sys_id 3
...             string  pool.comment
                pset    SASpset
        pset SASpset
                int     pset.sys_id 1
                boolean pset.default false
                uint    pset.min 1
                uint    pset.max 4
 ...            uint    pset.size 4
                cpu
 ...                    int     cpu.sys_id 1
                cpu
 ...                    int     cpu.sys_id 0
                cpu
 ...                    int     cpu.sys_id 3
                cpu
 ...                    int     cpu.sys_id 2
 ...
```
# Show the status of the default and newly created pool
```
root@ctclava # poolstat
                             pset
 id pool                 size used load
  0 pool_default           20 0.00 0.17
  3 SASpool                 4 0.00 0.00
```

Now we're ready for part ii) Creation of the zone(Hostname: SASzone). After creating the zone configuration file, SASZone-create.txt, and the proper destination directory for the zone root file system, and the zone system id file, 5 commands will create and boot the zone:

- zonecfg -z SASzone -f SASzone-create.txt
- zoneadm -z SASzone verify
- zoneadm -z SASzone install
- zoneadm -z SASzone ready
- zoneadm -z SASzone boot

Let's examine in further detail:
(Note: we use the root prompt root@ctclava # below to distinguish between commands in the global zone, hostname ctclava, and the root prompt bash-2.05 # to indicate root commands run in the newly created zone, hostname: SASzone)

Create a directory for the zone root, /d0/SASzone, a zone configuration file, /zonescfg/SASzone-create.txt, and system identification information /d0/SASzone/root/etc/sysidcfg.

```
root@ctclava # mkdir /d0/SASzone

root@ctclava # cat /zonescfg/SASzone-create.txt
create
set zonepath=/d0/SASzone
set pool=SASpool                 <<=== Assignment of resource pool, SASpool
set autoboot=false
add net
    set physical=ce0
    set address=192.168.30.23     <<==== Unique IP/Hostname required
end
add inherit-pkg-dir
    set dir=/opt
end
add fs
    set dir=/SASfs1                      <<=== Assign a filesystem, /SASfs1
    set type=lofs
    add options [rw,nodevices]
    set special=/A/test_r0_3/SASzonefs1    <<=== /SASfs1 maps from here
end
add fs
    set dir=/SASfs2                      <<=== Assign another fileystem, /SASfs2
    set type=lofs
    add options [rw,nodevices]
    set special=/A/priv_r0_1/SASzonefs2    <<=== /SASfs2 maps from here
end
verify
commit

root@ctclava # cat /d0/SASzone/root/etc/sysidcfg
system_locale=C
terminal=xterm
network_interface=primary {
                hostname=SASzone
                ip_address=192.168.30.23 }
security_policy=NONE
name_service=NONE
timezone=US/Eastern
root_password=XX_encrypted_passwd_XXX
```

# Create, verify and install the zone

```
root@ctclava # zonecfg -z SASzone -f SASzone-create

root@ctclava # zoneadm -z SASzone verify

root@ctclava # zoneadm -z SASzone install
Preparing to install zone <SASzone>.
Creating list of files to copy from the global zone.
Copying <3114> files to the zone.
Initializing zone product registry.
Determining zone package initialization order.
Preparing to initialize <1125> packages on the zone.
nitialized <1125> packages on zone.
Zone <SASzone> is initialized.
The file </d0/SASzone/root/var/sadm/system/logs/install_log> contains a log of the
zone installation.
```

# Ready the zone and boot it!

```
root@ctclava # zoneadm -z SASzone ready
root@ctclava # zoneadm -z SASzone boot
```

# Show the 2 zones

```
root@ctclava # zoneadm list -v
```

```
  ID NAME             STATUS        PATH
   0 global           running       /
   3 SASzone          running       /d0/SASzone
```

```
# Log into the console of the zone
root@ctclava # zlogin -C SASzone
SASzone console login: root
Password:
Feb  2 17:48:21 SASzone login: ROOT LOGIN /dev/console

# We now see the 2 file systems we configured, /SASfs1, /SASfs2
bash-2.05b# df -h
Filesystem             size   used  avail capacity  Mounted on
/                       64G    56G   7.4G    89%    /
/SASfs1                134G    17G   116G    13%    /SASfs1
/SASfs2                 67G   1.3G    65G     2%    /SASfs2
....
swap                    40G    32K    40G     1%    /var/run
swap                    40G     0K    40G     0%    /tmp

# We appear to have our own network interface; we do have a unique IP address
bash-2.05b# ifconfig -a
....
ce0:1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 192.168.30.23 netmask ffffff00 broadcast 192.168

# Voila!  We only see the 4 processors that were assigned to our zone  WOW!
bash-2.05b# psrinfo
0       on-line   since 12/17/2004 10:58:26
1       on-line   since 12/17/2004 10:58:26
2       on-line   since 12/17/2004 10:58:26
3       on-line   since 12/17/2004 10:58:26
```

At this point, we can install the SAS Metadata Server into the new zone, SASzone, and both can be running on the default port, 8561, with no conflict. Processes in SASzone can only see their own processes but the from the global zone, ctclava, we can see both instances of the Metadata server.

```
root@ctclava # ps -ef | grep sas
     root 19540 19536   0   Feb 04 ?         1:16 /SASinstall/SAS_9.1/sasexe/sas -log /
SAS/EGServers/Lev1/SASMain/MetadataServer/
     sas 29602 29598   0 10:28:38 ?         0:10 /d0/apps/sas/sas9-1201/SAS_9.1/sasexe/sas -log
/d0/apps/sas/sas9-1201/CTC4/Lev1
```

DTrace (Dynamic Tracing)

Continuing our medical analogies, DTrace is one of the most powerful, holistic, diagnostic tools available. The injection of radioactive dye and subsequent imaging technologies can give great insight into a particular body system. But what is this process lacking by the somewhat artificial constraints placed upon the process? A patient might have to fast for many hours, consume noxious amounts of enemas, go to the appointment only early in the morning and at a very set time? What happens if the problem occurs late in the day, only after eating and while exercising on the treadmill? And what if the problem is linked to another malfunction in a different subsystem other than the one being ***traced***? DTrace addresses all of those problems – it can be run anytime, is not invasive in terms of performance impact, does not require any instrumentation, at all, on the part of the application or OS and can simultaneously probe numerous areas of interest or speculation.

DTrace, a comprehensive dynamic tracing framework provides a powerful infrastructure to permit administrators, developers, and service personnel to concisely answer arbitrary questions about the behavior of the operating system and user programs. The *Solaris Dynamic Tracing Guide(http://docs.sun.com)* describes how to use DTrace to observe, debug and tune system behavior.

DTrace compiles and runs scripts written in 'D' . Numerous sample scripts are included in Solaris under /usr/demo/dtrace and documented in the above mentioned manual. For example, on a system wide basis, show me all failed open(2) system calls and error code (badopen.d), show me all signals sent along with the senders and recipients(sig.d), show me all write(2) system calls aggregated by command (writesbycmd.d), which processes are doing I/O (whoio.d). Starting with these last 2 scripts as a basis, we created a D script, which gives insight into the total amount of I/O for the system by summing the number of bytes field (b_bcount) for an I/O request via the I/O provider as well as counting all read(2), write(2) system calls and quantizing by the size of the request.

# Show total I/O by process name, along with detailed read(2)/write(2) data

```
bash-2.05b$ cat iototal.d
#pragma D option quiet
io:::start
{
        @[args[1]->dev_statname, execname,pid ] = sum(args[0]->b_bcount);
}
syscall::write:entry
{
        @writes[probefunc] = count();
        @sizes["write Sizes"] = quantize(arg2);
}
syscall::read:entry
{
        @reads[probefunc] = count();
        @sizes["read Sizes"] = quantize(arg2);
}
END
{
   printf("%10s %20s %10s %15s\n", "DEVICE", "APP", "PID", "BYTES");
   printa("%10s %20s %10d %15@d\n", @);
}
```

DTrace has a myriad of invocation alternatives, but to run this particular script, we use the –s option to pass in the name of the D script. We are not matching on a particular command name nor PID number, as many of the examples do, but rather, just watching the system as a whole.

To help you understand the output below, keep in mind the mapping of the device name (ie: ssd102) to file system mount points (/A/priv_r0_1 – SASWORK in our case). SAS was effectively running a data set copy of ~780 Mb from sd1 (/d0) to ssd102(/A/priv_r0_1).

# After desired time period, interrupt dtrace with <CNTRL-C>

```
bash-2.05b$ dtrace -s iototal.d
^C
    DEVICE                  APP        PID           BYTES
       sd1              fsflush          3            5120
       sd0                   vi      21042           11776
       sd0             cpudiagd        493           24576
       sd1                sched          0           54784
       sd0              fsflush          3          369152
       sd0                sched          0          698368
    ssd102                sched          0         1370624
    ssd102              fsflush          3         1936896
       sd1                  sas      21054       811515904
       sd0                  sas      21054       811735040
    ssd102                  sas      21054       880518144

  write                                                          99095
  write Sizes
           value  ------------- Distribution ------------- count
               0 |                                         0
               1 |                                         38
               2 |                                         3
               4 |                                         4
               8 |                                         10
              16 |                                         8
              32 |                                         0
              64 |                                         4
             128 |                                         0
             256 |                                         0
             512 |                                         11
            1024 |                                         0
            2048 |                                         0
            4096 |                                         1
            8192 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ 99016
           16384 |                                         0

  read Sizes
           value  ------------- Distribution ------------- count
               0 |                                         0
               1 |                                         15
               2 |                                         0
               4 |                                         5
               8 |                                         5
              16 |                                         8
              32 |                                         0
              64 |                                         16
             128 |                                         6
             256 |                                         0
             512 |                                         4
            1024 |                                         8
            2048 |                                         1
            4096 |                                         30
            8192 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ 99082
           16384 |                                         0

  read                                                          99180
```

What does this tell us? Just 8-10 lines of D code shows : **WOW!!!!**

- ~800 MB of I/O was done to sd1 and ssd102 by sas pid 21054
- We did ~100K read(2) sys calls and ~100K write(2) sys calls
- The read(2) and write(2) sys calls were almost exclusively 8K in size

To translate the ssd102 name to something more meaningful, use prtconf(1M).

```
bash-2.05b$ prtconf -v
......
             ssd, instance #102
                 Driver properties:
                     name='pm-hardware-state' type=string items=1 dev=none
.......
                 Device Minor Nodes:
                      dev=(118,816)
                           dev_path=/ssm@0,0/pci@19,700000/SUNW,qlc@2,1/fp@
0,0/ssd@w266000c0ffd04371,1:a              <== DEVICE NAME
....
```

The actual device name is now known and a search in /etc/vfstab will correlate the device name to the mount point (/A/priv_r0_1 as mentioned above).

```
bash-2.05b$ grep 266000c0ffd04371 /etc/vfstab
/dev/dsk/c10t266000C0FFD04371d1s6 /dev/rdsk/c10t266000C0FFD04371d1s6           /
A/priv_r0_1       ufs      2       yes      logging
```

DTrace requires root privilege in order to run. However, the astute reader will notice that all of the shell prompts above imply a normal user shell. Although we are not going to address the new Process Rights Management and the subject of least privileges, this is an example of how to allow users to run DTrace without giving them root access. For the user maureen, we simply put the following entry in the file, /etc/user_attr

```
maureen::::defaultpriv=basic,dtrace_proc,dtrace_user,dtrace_kernel
```

ZFS

Studies have shown that when medical interns have to pull long shifts, the likelihood for making critical mistakes and bad decisions increases dramatically. Similarly when medication regiments are extremely complicated, the chance of a wrong or fatal dosage increase. In both cases, less hours, less stress, less complexity lead to a much higher quality and safer care. And these goals, among others, exactly align with the benefits of ZFS.

ZFS* is a new, dynamic file system offering:

- Simple administration : automates and consolidates complicated storage administration concepts, thereby greatly reducing administrative overhead.
- Provable Data integrity: protects all data with 64-bit checksums that detect and correct silent data corruption.
- Unlimited scalability: As a 128-bit file system, ZFS offers 16 billion billion times the capacity of 32- or 64-bit systems.
- High performance: based on a transactional object model that removes most of the traditional constraints on the order of issuing I/Os, which results in huge performance gains.

*ZFS will not be available in the initial Solaris 10 release.

Imagine this scenario - a SAS user rushes to the IT ER. The cranky attending physician has never seen this new disease born of the Internet boom and barks a command to the new intern on the job for the 1st day: “Give this patient 3 new file systems - STAT! The intern finds 2 spare partitions in the dark corners in the back of the SAN cabinet. With ZFS, no problem. In less than 1 minute, the patient has a remedy and now has 3 new file systems after running the commands below... **WOW!**

```
# zpool create -fv SAS_WORK_AREA c5t266000C0FFD03FE0d0s6 c8t226000C0FF903FE0d1s6
# zfs create SAS_WORK_AREA/BLDG_R /SASWORK_A
# zfs create SAS_WORK_AREA/BLDG_S /SASWORK_B
# zfs create SAS_WORK_AREA/BLDG_T /SASWORK_C
```

What administrative headaches are missing? Creation of partition for volume manager metadata, potential re-partition via format command to accommodate this metadata partition, initialization of volume manager, creation of volume, complicated command to create file systems, creation of directory mount point, mounting of file system. If you've ever done this before, you realize that it takes much more than 4 commands and 1 minute. Lastly, if you want to grow or shrink your storage pools, you can do that dynamically without having to rebuild or quiesce the file system.

Let's take a closer look at what was just done.

Create a storage pool named SAS_WORK_AREA consisting of 3 partitions.

```
# zpool create -fv SAS_WORK_AREA c5t266000C0FFD03FE0d0s6 c8t226000C0FF903FE0d1s6
```

Show that the new mount points don't exist

```
# ls -ld /SASWORK_A /SASWORK_B /SASWORK_C
/SASWORK_A: No such file or directory
/SASWORK_B: No such file or directory
/SASWORK_C: No such file or directory
```

Note: In the ZFS documentation, the term data set is used for partitions. Since the term data set has specific context for SAS users, we will use the term partition to refer to the concept of a ZFS data set.
Partition the pool by buildings R,S, &T and then mount respectively on /SASWORK_A, /

SASWORK_B, /SASWORK_C

```
# zfs create SAS_WORK_AREA/BLDG_R /SASWORK_A
# zfs create SAS_WORK_AREA/BLDG_S /SASWORK_B
# zfs create SAS_WORK_AREA/BLDG_T /SASWORK_C
```

Sys admins will appreciate that the mount point directory will be created if it doesn't exist.

```
# ls -ld /SASWORK_A /SASWORK_B /SASWORK_C
drwxr-xr-x   2 root     sys            2 Feb  5 23:44 /SASWORK_A
drwxr-xr-x   2 root     sys            2 Feb  5 23:45 /SASWORK_B
drwxr-xr-x   2 root     sys            2 Feb  5 23:45 /SASWORK_C
```

mount(1M) shows that the 3 new file systems are now mounted

```
# mount | grep SASWORK
/SASWORK_A on SAS_WORK_AREA/BLDG_R read/write/setuid/devices/noatime/dev=55c0001 on Sat Feb
5 23:44:54 2005
/SASWORK_B on SAS_WORK_AREA/BLDG_S read/write/setuid/devices/noatime/dev=55c0002 on Sat Feb
5 23:45:11 2005
/SASWORK_C on SAS_WORK_AREA/BLDG_T read/write/setuid/devices/noatime/dev=55c0003 on Sat  Feb
5 23:45:30 2005
```

At this point, the available space for the storage pool is equivalent to the available space for any of the new file systems.

```
# zpool df
Pool                   size   used  avail capacity
-------------------- ------ ------ ------ --------
SAS_WORK_AREA          135G  20.8M   135G      1%
# cd /SASWORK_A
# df -k .
Filesystem            kbytes    used   avail      capacity  Mounted on
SAS_WORK_AREA/BLDG_R 138314040      75 138313965     1%     /SASWORK_A
```

2 other useful commands:
# Show the ZFS partitions

```
# zfs ls
SAS_WORK_AREA/BLDG_R
SAS_WORK_AREA/BLDG_S
SAS_WORK_AREA/BLDG_T
```

# Display I/O stats for ZFS pools. Below write I/O is going to /SASWORK_A but we can see that the I/O is striped across both devices behind the scenes.

```
# zpool iostat 5
 ....
                                    capacity   operations   bandwidth    errors
vdev   description                 used avail  read write  read write  cksum I/O
   1 ..c5t226000C0FF903FE0d0s6    1.34G 66.0G     0   411     0 51.5M      0   0
   2 ..c5t226000C0FF903FE0d1s6    1.35G 66.0G     0   415     0 51.4M      0   0
-----  ----------------------     ----- ----- ----- ----- ----- -----  ----- ---
Total  SAS_WORK_AREA              2.69G  132G     0   826     0  103M      0   0
```

Service Management Facility (SMF)

Services that are self-healing - need more be said? Another feature to enable services to be more robust, reliable and easier to manage. Preventative and pro-active health care in action.

SMF is a new unified model for services and service management on each Solaris system. It is a core part of the Predictive Self-Healing technology available in Solaris 10, which provides automatic recovery from software and hardware failures as well as administrative errors. Failed services are restarted in dependency order. The management of these services can be delegated to non-root users. SMF is a follow-on to the legacy method of starting/stopping services. Note that these /etc/rc scripts will continue to run just as before.

Deployment of SAS services such as the SAS Metadata Server, SAS Object Spawner, SAS OLAP Server, SAS SHARE Server, etc via SMF provides a much more consistent and robust environment. First, users can query Solaris with a simple command (svcs –a) to determine if the service is running at all instead of running their SAS program (such as the SAS Management Console) and wondering if the connection to the SAS Metadata Server will succeed. Additionally, critical services such as the SAS Metadata Server can be automatically restarted in the event of a problem where the process might have gone AWOL for whatever reason (someone inadvertently killed it, bug causing a core dump, etc).

After a SAS 9 installation, there can be a half dozen or more SAS servers to start. For each service, these are the logical steps that need to be done to incorporate these services into SMF:

1. Create a service manifest file, and shell script file to define the start, stop, restart methods for the service.
2. Validate and import the service manifest using svccfg(1M)
3. Enable or start the service using svcadm(1M)
4. Verify the service is running using svcs(1)

Our example will create 2 services, one for the SAS Metadata Server(OMR for short) which has to start first and then for the SAS Object Spawner which has a dependency on the OMR.

Step 1a) Create a directory and manifest file in /var/svc/manifest/application/sas. Note: You can locate the manifest and script method file in any directory. Solaris has a concept of an application subdirectory from these directories for applications such as SAS but the files don't have to be stored there. I chose the sas subdirectory since there is potential to have 6+ entries purely for manageability reasons.

/var/svc/manifest/application/sas/metadata.xml contains:

```
<?xml version="1.0"?>
<!DOCTYPE service_bundle SYSTEM "/usr/share/lib/xml/dtd/service_bundle.dtd.1">

<service_bundle type='manifest' name='SAS:Metadata'>
<service
    name='application/sas/metadata'
    type='service'
    version='1'>
        <create_default_instance enabled='false' />
        <single_instance />

        <dependency
                name='multi-user-server'
                grouping='optional_all'
                type='service'
                restart_on='none'>
                  <service_fmri value='svc:/milestone/multi-user-server' />
        </dependency>
        <exec_method
            type='method'
            name='start'
            exec='/lib/svc/method/sas/metadata %m'
            timeout_seconds='60'>
            <method_context>
                <method_credential user='sas' />
            </method_context>
         </exec_method>

        <exec_method
            type='method'
            name='restart'
            exec='/lib/svc/method/sas/metadata %m'
            timeout_seconds='60'>
            <method_context>
                <method_credential user='sas' />
            </method_context>
         </exec_method>

        <exec_method
            type='method'
            name='stop'
            exec='/lib/svc/method/sas/metadata %m'
            timeout_seconds='60' >
            <method_context>
                <method_credential user='sas' />
            </method_context>
        </exec_method>

        <property_group name='startd' type='framework'>
                <propval name='duration' type='astring' value='contract' />
        </property_group>

        <template>
                <common_name>
                        <loctext xml:lang='C'>
                                SAS Metadata Service
                        </loctext>
                </common_name>
                <documentation>
                <doc_link name='sas_metadata_overview'
                   uri='http://www.sas.com/technologies/bi/appdev/base/metadatasrv.html' />
                <doc_link name='sas_metadata_install'
                 uri='http://support.sas.com/rnd/eai/openmeta/v9/setup' />
                </documentation>
        </template>
</service>
</service_bundle>
```

Step 1b) Create the methods file in /lib/svc/method/sas/metadata

```
#!/sbin/sh
# Start/stop client SAS MetaData service
#
. /lib/svc/share/smf_include.sh
SASDIR=/d0/apps/sas/sas9-1201
SRVR=MetadataServer
CFG=$SASDIR/CTC4/Lev1/SASMain/"$SRVR".sh

case "$1" in
'start')
        $CFG start
        sleep 2
        ;;
'restart')
        $CFG restart
        sleep 2
        ;;
'stop')
        $CFG stop
        ;;
*)
        echo "Usage: $0 { start | stop }"
        exit 1
        ;;
esac
exit $SMF_EXIT_OK
```

That's the hardest part of setting up the service. Now all that's left:

```
# pwd
/var/svc/manifest/application/sas
```

Step 2) Validate and import the manifest into the Solaris service repository

```
# svccfg
svc:> validate /var/svc/manifest/application/sas/metadata.xml
svc:> import /var/svc/manifest/application/sas/metadata.xml
svc:> quit
```

Step 3) Enable the service, exclude the –t if you want this to be a permanent change to persist between reboots. We use it here for testing the service bring up.

```
# svcadm enable -t svc:/application/sas/metadata
```

Step 4) Verify that the service is online, and that the processes really are running

```
# svcs -a | grep sas
online          8:44:37 svc:/application/sas/metadata:default
```

```
# ps -ef | grep sas
....
     sas 26795 26791   0 08:44:36 ?          0:01 /d0/apps/sas/sas9-1201/SAS_9.1/sasexe/sas
-log /d0/apps/sas/sas9-1201/CTC4/Lev1
...
     sas 26791     1   0 08:44:36 ?          0:00 /bin/sh /d0/apps/sas/sas9-
1201/CTC4/Lev1/SASMain/MetadataServer/MetadataServer.sh
```

Now, let's add the ObjectSpawner service. The manifest looks very similar to the Metadata manifest but we would add a dependency such as:

```
        <dependency
                name='sas-metadata-server'
                grouping='optional_all'
                type='service'
                restart_on='none'>
                     <service_fmri value='svc:/application/sas/metadata' />
        </dependency>
```

After creating the manifest and correlating the /lib/svc/method/sas/objectspawner script, register and enable the new service in the same manner:

```
# svccfg
svc:> import /var/svc/manifest/application/sas/objectspawner.xml
svc:> quit

# svcadm enable -t /application/sas/objectspawner
# svcs -a | grep sas
online         10:28:39 svc:/application/sas/metadata:default
online         10:38:20 svc:/application/sas/objectspawner:default
```

We now see that both the Metadata and ObjectSpawner servers are indeed running.
Note: PID 26864 is the Metadata server process.

```
# ps -ef | grep sas
     sas 26860     1   0 18:18:47 ?        0:00 /bin/sh
/d0/apps/sas/sas9-1201/CTC4/Lev1/SASMain/MetadataServer/MetadataServer.sh
    sas 26864 26860    0 18:18:47 ?        0:01 /d0/apps/sas/sas9-1201/SAS_9.1/sasexe/sas
-log /d0/apps/sas/sas9-1201/CTC4/Lev1
     sas 26914     1   0 18:18:49 ?        0:00 /bin/sh
/d0/apps/sas/sas9-1201/CTC4/Lev1/SASMain/ObjectSpawner/ObjectSpawner.sh
     sas 26918 26914   0 18:18:49 ?        0:00
/d0/apps/sas/sas91201/SAS_9.1//utilities/bin/objspawn -sasSpawnerCn SASMain -
```

To show that the services will automatically restart themselves as configured, we kill off the current Metadata server (pid: 26864).

```
# kill 26864
```

ps(1) now shows that the previous Metadata PID of 26864 no longer exists and a new PID, 27035, is now running. All restarted without user intervention.

```
# ps -ef | grep sas     WOW!
     sas 27031     1   0 18:28:05 ?        0:00 /bin/sh
/d0/apps/sas/sas9-1201/CTC4/Lev1/SASMain/MetadataServer/MetadataServer.sh
     sas 27035 27031   0 18:28:05 ?        0:01 /d0/apps/sas/sas9-1201/SAS_9.1/sasexe/sas -log
/d0/apps/sas/sas9-1201/CTC4/Lev1
     sas 26914     1   0 18:18:49 ?        0:00 /bin/sh
/d0/apps/sas/sas9-1201/CTC4/Lev1/SASMain/ObjectSpawner/ObjectSpawner.sh
     sas 26918 26914   0 18:18:49 ?        0:00
/d0/apps/sas/sas9-1201/SAS_9.1//utilities/bin/objspawn -sasSpawnerCn SASMain -
```

Given that the Metadata Server is a critical component of the SAS 9 deployment, this feature greatly adds to the availability of the application environment.

Bonus: Tribal Wisdom artifacts on SAS 9 deployments

- CPUCOUNT – on large systems CPUCOUNT will default to the number of processors as reported in psrinfo(1M). This variable is used as an advisory for the number of LWPs to spawn for the multithreaded PROCs. Set it to 4 on systems with 4 or more CPUs.
- Metadata repository – When you initialize the SAS Metadata repository for the first time, you have to specify the directory in which it will reside. This data store could be potentially very update intensive. Choose a directory on a high performance storage subsystem.
- Consider trying to use a larger data set BUFSIZE(SAS option). In our DTrace example above, we saw I/O sizes almost exclusively of 8K. Increasing BUFSIZE to 64K should help on I/O intensive applications. Once you increase BUFSIZE, you might want to use the iototal.d example above to confirm that larger I/Os are being performed.
- If using SAS 9 with SAS 8.2 data sets, using PROC MIGRATE could help realize a 5-10% performance gain on I/O intensive jobs since the CEDA conversion would be avoided.
- The SAS 9 installation adds somewhat non descript entries into the Solaris package registry. Thus, if you wish to uninstall a SAS 9 installation, it's generally bad practice to 'rm -rf $SAS_INSTALL_DIR/*' as bogus entries will remain in the registry. While this, in of itself, is not a problem, it's possible that future installations could erroneously install into the incorrect directory (or worse, overwrite a previous, working installation) based on stale state info in the package registry. Consequently, if you wish to re-install the software or components of it, it's important to properly remove the existing software. Additionally, the SAS software navigator stores installation state in $HOME/vpd.properties and $HOME/.sasprefs. Check or remove these files if a complete re-installation is done.

  To compound matters, there is no master uninstall script and each software component must be individually removed. You can use a simple command such as
  ```
  # find . -type d -name \*uninst\* -print
  ```
  to find all the directories which contain uninstallation scripts usually stored in a component sub-directory with uninst in the name. For example, the SAS Management Console uninstall script would be .../_uninst/UninstSASMC.
  ```
   # pwd
  /d0/apps/sas/sas9-1201/SASManagementConsole/9.1/_uninst
  # ls
  UninstSASMC    uninstall.dat  uninstall.jar
  ```

- The SAS installation process consists of invoking individual component installation scripts. A number of the scripts use find(1) starting with the installation user's home directory. If the user, root (not generally recommended), is installing the software, and if many large file systems are mounted (both local and remote), traversal of each and every file system will *greatly* add to the time to do the installation.
- A number of the scripts to start/stop/restart the various SAS servers, ie: MetadataServer.sh, have an entry, SERVERUSER=root, who will be the effective user when starting the actual daemon process. If you change this to, SERVERUSER=sas, and then start the script as root, issuing the MedataServer.sh stop command kills only the parent shell and the actual server doesn't go away. Without manual intervention, you can't restart the server because the network communication port will be in use. Using SMF, you can leave SERVERUSER set to root, but specify the user sas for the credentials in the service manifest, as above, and the script works for startup and shutdown under the user id, sas. And if the Metadata Server dies for any reason, it will automatically be restarted by SMF.
  If you are not using SMF and start the servers via the legacy /etc/rc method, the trick to avoid this problem is to set SERVERUSER=sas and modify the MetdataServer.sh script to be similar to:

```
....
        if [ $root -eq 1 ]; then
           su - $SERVERUSER -c "$DIR/$0 start2 &"
        else
```

```
                $0 start2 &
            fi
            ;;
    start2)
            cd $DIR/..
                nohup $SASCMD -log $DIR/logs/MetadataServer_%Y.%m.%d.log  ...
....
```

The difference being that you "su – sas" prior to actually invoking the actual invocation of the SAS server. SAS Tech Support can give you specific details on the exact modifications.

Summary

In summary, we've taken a quick hands on tour of a few of the new and innovative features introduced in Solaris 10 and how these features can provide a comprehensive healthcare plan for IT departments who are ready to roll out SAS 9 deployments. Solaris 10 rocks!

On that note, feel free to email the author (address below) or sas-on-sun@sas.com if you have any comments, suggestions or questions. ☺

All testing was done on a Sun Fire E2900 with 12 UltraSPARC IV CPUs, 48GB RAM, StorEdge 3510s in SAN configuration running Solaris 10.

References

SAS 9 Runs Seamlessly Sun's New Solaris 10
http://www.sas.com/news/preleases/111504/news1.html

SAS 9.1.3 Metadata Server: Setup and Administration Guide
http://support.sas.com/rnd/eai/openmeta/v9/setup

10 Reasons to Move to Solaris 10 – http://www.sun.com/software/solaris/top10.jsp

Solaris 10 System Administration Collection – http://docs.sun.com/app/docs/coll/47.16

BigAdmin System Administration Portal - DTrace – http://www.sun.com/bigadmin/content/dtrace

Solaris Dynamic Tracing Guide – http://docs.sun.com/app/docs/doc/817-6223?q=dtrace

Spotlight on Solaris Zones – http://www.sun.com/bigadmin/features/articles/solaris_zones.html

BigAdmin System Administration Portal - Solaris Zones – http://www.sun.com/bigadmin/content/zones

BigAdmin System Administration Portal - Predictive Self-Healing
Solaris Service Management Facility - Quickstart Guide
http://www.sun.com/bigadmin/content/selfheal/smf-quickstart.html

Solaris Service Management Facility - Service Developer Introduction
http://www.sun.com/bigadmin/content/selfheal/sdev_intro.html

Solaris 10 System Administrator Collection >> System Administration Guide: Basic Administration >> 9. Managing Services (Overview)
http://docs.sun.com/app/docs/doc/817-1985/6mhm8o5n0?a=view

ZFS --the last word in file systems. - http://www.sun.com/2004-0914/feature

Sun BluePrints[tm] OnLine - Performance Oriented System Administration - Bob Larson
http://www.sun.com/solutions/blueprints/1202/817-1054.pdf

SAS Version 9.1 on Solaris 9 Performance, Monitoring & Optimization Tips (M. Chew - 2003)
http://www.sas.com/partners/directory/sun/v9on9.pdf

Performance Tuning & Sizing Guide for SAS Users and Sun System Administrators (T. Keefer / W. Kearns - 2003)
http://www.sas.com/partners/directory/sun/sugi28.pdf

Pushing the Envelope: SAS System Considerations for Solaris/UNIX in Threaded, 64 bit Environments (M. Chew - 2002)
http://www.sas.com/partners/directory/sun/64bit.pdf

Peace between SAS Users & Solaris/Unix System Administrators (M. Chew / L. Ihnen / T. Keefer – 1999)
http://www.sas.com/partners/directory/sun/performance/index.html

Sun Tunathon 2004 Showcases a Global Convergence of Engineering Talent
http://support.sas.com/news/feature/04oct/suntune.html

Acknowledgments

Many thanks to Phil Kinslow for his zones contributions and much needed support, advice, encouragement and Chien-Hua Yen for sharing his Solaris expertise.

Lastly, thanks to Tony, Rita, Michael and Kipp.

About the Author

Maureen Chew, Staff Engineer, has been with Sun Microsystems for 16+ years. She is a resident of Chapel Hill, NC and can be reached at maureen.chew@sun.com.

SUGI 30, Philadelphia, Apr 10-13
Wed, Apr 13, Rm: 203

http://www.sas.com/partners/directory/sun/sas9-on-solaris10-paper.pdf