
ORACLE®

ORACLE

Using DTrace for Leverage

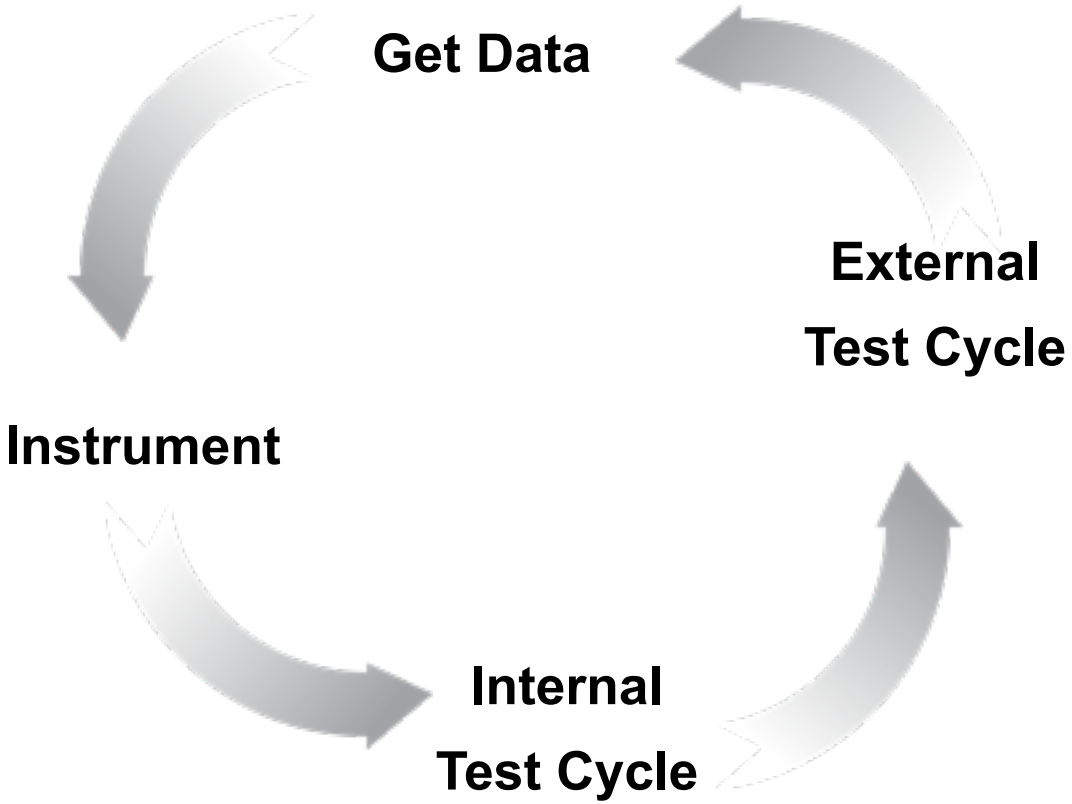
Thomas Haynes
Principal Software Engineer

The many faces of DTrace

- Solve customer escalations
- Debugging
  - Before integration
  - After integration
- Gross performance analysis

Section Customer Escalations

Getting debug code to customers



Solving Customer Escalations

- Gather data at a customer site
  - Raw analysis
  - Predictive confirmation
- Reproduce problem in-house
  - How do we know we have captured everything?
  - Can we see what the customer is seeing?
- Get a fix in place
  - How do we know the fix works?
  - How do we convince the customer?
- Provide the fix to the customer
  - Predictive confirmation
  - Emergency backup - what data to gather?

Preventing Customer Escalations

- Do not want to provide a kernel that causes problems
- Do not want to provide a kernel that stays in production
  - You fixed a specific issue
  - Customer wants to stay on that “release”
  - Your support department may not know how to handle that release

Preventing Customer Escalations

- Do want to predict what the customer will see
- Do want to confirm the fix
- Do want to get them on the path to a supported release

6882460 - mountd storms

- Customer has observed that simultaneously mounting shares via NFSv3 and immediately beginning to read data--with hundreds to near 1,000 Linux clients--causes some mounts to fail.
- NFSv4 does not exhibit the same issue
- waiting for a short time between mounting and reading also alleviates the errors.
- http://bugs.opensolaris.org/bugdatabase/view_bug.do?bug_id=6882460

Usual suspects

- Packet traces show either
  - Quick responses to mount requests
  - No response at all
- Name server is quick to respond
  - Putting names in /etc/hosts does not improve things
- No netgroups
- Bumping number of mountd threads improves things

Recreating a mountd storm

- Want to be able to blast N mount requests to a server
- Do not care about the replies
- Need to have both forward and reverse DNS entries
- Need to consider one request per IP versus many
  - Only first request should see a name lookup across the wire
  - Case appears to be insensitive to name lookups
- Use a Perl script to blast N UDP mount requests
  - Are serialized, but end up arriving quick enough
  - Easy enough to scale past customer needs
  - Might need to really scale in order to get meaningful data

Time to drill down

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

/*
 *      # ./mountd.d `pgrep -x mountd`
 */

dtrace:::BEGIN
{
        printf("Sampling... Hit Ctrl-C to end.\n");
}

pid$1::mount:entry
{
        self->timestamp[probefunc] = timestamp;
        @function_count[probefunc] = count();
        self->trace = 1;
}

pid$1::mount:return
{
        @function_quantize[probefunc] =
              quantize(timestamp - self->timestamp[probefunc]);
        self->timestamp[probefunc] = 0;
}
```

Real customer data

```
     mount
    449
mount
           value  ------------- Distribution ------------- count
         2097152 |                                         0
         4194304 |@@                                       26
         8388608 |@                                        14
        16777216 |@                                        13
        33554432 |@                                        9
        67108864 |@@                                       18
       134217728 |@@                                       28
       268435456 |@@@@@                                    59
       536870912 |@@@                                      31
      1073741824 |                                         5
      2147483648 |                                         4
      4294967296 |                                         2
      8589934592 |                                         2
     17179869184 |@@@                                      30
     34359738368 |@@@@@@@@@@@@@@@@@@@@                     208
     68719476736 |                                         0
```

Analysis

- 1000 mount requests
- 449 made it to mount(), which means 551 did not
- 30 mount requests took between 17-34 seconds
- 208 mount requests took between 34-68 seconds

Reproduction with 4000 requests

```
    washdc# ./mountd.d `pgrep -x mountd`
dtrace: script './mountd.d' matched 3 probes
CPU FUNCTION
  5 | :BEGIN                                    Sampling... Hit Ctrl-C
to end.

^C

  mount                                                             843
  mount
           value  ------------- Distribution ------------- count
          524288 |                                         0
         1048576 |@@@@@@@@@@                               214
         2097152 |                                         3
         4194304 |@                                        15
         8388608 |@                                        12
        16777216 |                                         0
        33554432 |                                         1
        67108864 |                                         3
       134217728 |@                                        25
       268435456 |@@@@@@@                                  147
       536870912 |@@@@@@@@@@@@@@@@@@@@@                    422
      1073741824 |                                         1
      2147483648 |                                         0
```

Comparing real vs synthetic

- Went to 4000 to get more data
  - Went up to 40k at some points
- Lab machine appears faster
  - Did not model NFS traffic, just mountd
  - Did do a NFS load later
- Mostly happy we simulated a gap
  - I.e., data appears representative of the general problem

Looking at the mount() code

- caller
  - Picked apart request
  - Determined which procedure to invoke
- do name lookup
- do access check
- reply to client
- log into BSM audit trail
- add to /etc/rmtab
- return

Auditing looked suspicious

- Fast replies once we got them
- Increasing threads appeared to alleviate problem
  - Actually provided a queue for starvation
- We were starved waiting on a global lock
- Applied a fix to add entries to BSM audit trail asynchronously

Asynch auditing with 4000 requests

```
    washdc# ./mountd.d `pgrep -x mountd`
dtrace: script './mountd.d' matched 5 probes
CPU FUNCTION
  0 | :BEGIN                                    Sampling... Hit Ctrl-C
to end.

^C

  mount                                                           2970
  mount
           value  ------------- Distribution ------------- count
           32768 |                                         0
           65536 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ 2051
          131072 |@@@@@@@@@@@@@                            902
          262144 |                                         15
          524288 |                                         1
         1048576 |                                         1
         2097152 |                                         0
```

Analysis of async auditing

- Service about 3 times as many mount requests
- Long pole is about 130 microseconds
  - versus 0.5 seconds
- We were also able to show it was independent of the number of mountd threads
  - Get the customer off of this customization!

Customer results with async logging

```
mount
           value  ------------- Distribution ------------- count
           16384 |                                         0
           32768 |                                         4
           65536 |@@@@@                                    123
          131072 |@@@@@@@@@@@@@@@@@@@@@@@@                 530
          262144 |@@@@@@@@@@                               232
          524288 |@@@                                      66
         1048576 |                                         7
         2097152 |                                         7
         4194304 |                                         0
         8388608 |                                         3
        16777216 |                                         0
```

Section Debugging

Debugging
Before integration

- Live system debugging
  - Save mdb for core dumps
- Want to avoid printf()
- Want to avoid recompilation
- What can existing function calls tell you?
- Where do you need to add a static probe?

Adding static probes

- Counters
- Error detection
  - Avoid logging to console
    - Certainly avoid spamming console/logs
- Allow targeted data gathering
  - Avoid ifdef
  - Avoid /etc/system
  - Allow dtrace to be your on switch

Debugging
After integration

- Help others understand your code
  - Provide scripts to allow them to debug problems
- Help you understand if more static probes needed
  - First external use
  - Customers will have same needs

Debugging spe

- spe assigns layouts
  - mds does not need to use it
  - Loads from
    - /etc/npools.spe
    - /etc/policies.spe
- Always being asked
  - Did spe work?
    - nfsstat -l can tell you that
  - How do I know which policy fired?
- A small script can tell all

spe.d part 1

```
    nfsv4:::spe-i-check_open
{
        printf("%d (%d) from %s is checking %s",
            (uid_t)arg0, (gid_t)arg1, stringof(arg3), stringof(arg2));
}

nfsv4:::spe-i-policy_eval
{
        printf("Policy %d %s with error %d from %s",
            (uint_t)arg0, (boolean_t)arg1 ? "matched" : "did not
match",
            (int)arg2, stringof(arg3));
}

::nfs41_spe_allocate:entry
{
        self->addr = (struct netbuf *)arg2;
        self->stripe_count = (count4 *)arg4;
        self->unit_size = (uint32_t *)arg5;
        self->mds_sids = (mds_sid **)arg6;

        self->loaded_sids = 0;
}
```

spe.d part 2

```
    ::nfs41_spe_allocate:return
/args[1] == 0/
{
        printf("Policy has %d stripes and %u block size",
            *self->stripe_count, *self->unit_size);
}

::nfs41_spe_allocate:return
/args[1] != 0/
{
        printf("No matching policy");
}

::mds_ds_path_to_mds_sid:entry
{
        self->ustring = (utf8string *)arg0;
        self->ss_name = stringof(self->ustring->utf8string_val);
        self->mds_sid = (struct mds_sid *)arg1;
}
```

spe.d part 3

```
::mds_ds_path_to_mds_sid:return
/args[1] == 0/
{
        ss_name = (char *)alloca(self->ustring->utf8string_len + 1);
        bcopy(self->ustring->utf8string_val, ss_name,
            self->ustring->utf8string_len);
        ss_name[self->ustring->utf8string_len + 1] = '\0';
        printf("mds_sid[%d] = %s", self->loaded_sids++,
            stringof(ss_name));
}

::mds_ds_path_to_mds_sid:return
/args[1] != 0/
{
        ss_name = (char *)alloca(self->ustring->utf8string_len + 1);
        bcopy(self->ustring->utf8string_val, ss_name,
            self->ustring->utf8string_len);
        ss_name[self->ustring->utf8string_len + 1] = '\0';
        printf("ERROR - could not find a matching pgi for %s",
            stringof(ss_name));
}
```

Policies not loaded

```
[root@pnfs-minipit2-5 ~]> ./spe.d
dtrace: script './spe.d' matched 8 probes
CPU     ID                    FUNCTION:NAME
 1  28427 nfs41_spe_allocate:spe-i-check_open 60001 (60001)
       from 10.1.235.62 is checking /diskpool/DS/P2/foo
 1  59258        nfs41_spe_allocate:return No matching policy
```

DS dataset does not match mds

```
    1 3420 nfs41_spe_allocate:spe-i-check_open 200096 (10) from
    10.1.233.191 is checking /diskpool/JUNK/TEST/P5/tomper
1 3419 nfs41_spe_allocate:spe-i-policy_eval Policy 101 did not match
with error 0 from 10.1.233.191
1 3419 nfs41_spe_allocate:spe-i-policy_eval Policy 102 did not match
with error 0 from 10.1.233.191
1 3419 nfs41_spe_allocate:spe-i-policy_eval Policy 103 did not match
with error 0 from 10.1.233.191
1 3419 nfs41_spe_allocate:spe-i-policy_eval Policy 104 did not match
with error 0 from 10.1.233.191
1 3419 nfs41_spe_allocate:spe-i-policy_eval Policy 111 matched with
error 0 from 10.1.233.191
1 63756 mds_ds_path_to_mds_sid:return mds_sid[0] =
    pnfs-minipit1-6:pNFSpool1/p1DS2
1 63756 mds_ds_path_to_mds_sid:return mds_sid[1] =
    pnfs-minipit1-6:pNFSpool2/p2DS2
1 63756 mds_ds_path_to_mds_sid:return mds_sid[2] =
    pnfs-minipit1-6:pNFSpool3/p3DS2
1 63756 mds_ds_path_to_mds_sid:return mds_sid[3] =
    pnfs-minipit1-7:pNFSpool1/p1DS1
1 63756 mds_ds_path_to_mds_sid:return
1 57043 nfs41_spe_allocate:return No matching policy
```

What went wrong?

- Policy stated that 5 datasets were needed
  - 111, 5, 32k, default, path == /diskpool/JUNK/TEST/P5
- Only 4 found
- DS has a dataset named:
  - pnfs-minipit1-7:pNFSpool2/p2DS1
- /etc/npools.spe has dataset named as
  - pnfs-minipit1-7:pNFSpool1/p2DS1

Section Gross Performance

Gross performance analysis

- What is the performance cost of a
  - New feature
  - Bug fix
    - What was the performance before?
- “Gross” means
  - A metric you can use as a developer
  - Gives you a rule of thumb approximation
  - Not something you might put in a formal report
- But you need to be able to answer the question about the impact

Gross Performance of referrals

- Referrals built on top of mirrormount framework
- As part of integration, asked ourselves
  - What is the performance of doing a referral?
- Note: we never asked ourselves that when we did the mirrormount work

Timing referrals

```
    [root@pnfs-4-11 ~]> more mms.d
#!/usr/sbin/dtrace -Fs

::nfs4_trigger_mount:entry
{
        self->timestamp[probefunc] = timestamp;
        @function_count[probefunc] = count();
}

::nfs4_trigger_mount:return
{
        @function_quantize[probefunc] = quantize(timestamp -
                self->timestamp[probefunc]);
        self->timestamp[probefunc] = 0;
}
```

- Kernel versus userland
- But pretty much the same as before

First look at the performance

```
    nfs4_trigger_mount
   4098
nfs4_trigger_mount
           value  ------------- Distribution ------------- count
            4096 |                                         0
            8192 |                                         19
           16384 |@@@@@@@@@@@@@@@@@@                       1749
           32768 |                                         27
           65536 |                                         2
          131072 |                                         0
          262144 |                                         1
          524288 |                                         1
         1048576 |                                         0
         2097152 |                                         0
         4194304 |                                         0
         8388608 |@@@@@@@                                  672
        16777216 |@@@@@@@@@@@@                             1216
        33554432 |@                                        95
        67108864 |@                                        102
       134217728 |@                                        97
       268435456 |@                                        74
       536870912 |                                         42
      1073741824 |                                         1
      2147483648 |                                         0
```

Okay, 1/2 second sounds gross

- About 750 referrals, all under one parent
- mirrormounts gave about the same numbers
- We did get a 68 second outlier

3099 mirrormounts under 1 real

```
    nfs4_trigger_mount
   3099
nfs4_trigger_mount
      value  ------------- Distribution ------------- count
    1048576 |                                         0
    2097152 |                                         1
    4194304 |                                         37
    8388608 |@@@                                      235
   16777216 |@@                                       147
   33554432 |@@                                       189
   67108864 |@@                                       150
  134217728 |@@@                                      211
  268435456 |@@@                                      234
  536870912 |@@@                                      224
 1073741824 |@@@@                                     329
 2147483648 |@@@@@                                    420
 4294967296 |@@@@@                                    375
 8589934592 |@@@                                      265
17179869184 |@@@                                      198
34359738368 |@                                        84
68719476736 |                                         0
```

2049 mirrormounts under 1051 real

```
    nfs4_trigger_mount
   2049
nfs4_trigger_mount
     value  ------------- Distribution ------------- count
   8388608 |                                         0
  16777216 |                                         9
  33554432 |@@@                                      132
  67108864 |@@@@@@@@@@@@@@@@                         789
 134217728 |@@@@@@@@@@@@@@@@@@                       874
 268435456 |@@@@                                     211
 536870912 |@                                        33
```

Analysis - Another locking issue

- We are holding a lock as we do a syscall to perform an ephemeral mount
  - I.e., mirrormount or referral
- Lock prevents tree from going away from other events
- With 1051 real mount points, we distribute the load
- With 1 real mount point, we see the strain
- Can model a reader/writer lock to alleviate the starvation

Section Take home message

Take it home with you

- DTrace allows you to avoid printf()
  - I.e., no recompiling
- DTrace allows you to gather data at a customer's site
  - They can even gather it. :->
- No instrumented kernels polluting a customer's site


ORACLE IS THE INFORMATION COMPANY