
ORACLE®

ORACLE®

How To Build Better Applications With Oracle Solaris DTrace

Brendan Gregg
Principal Software Engineer

Jim Mauro
Senior Principal Software Engineer

Agenda - Oracle Solaris Dynamic Tracing

- What is DTrace
- Using DTrace
- Application Analysis
- USDT Probes

What is DTrace?

- An observability and analysis tool
  - /usr/sbin/dtrace + kernel stuff
- … that instruments all software
  - kernel, user-land applications
- … using thousands of probes
  - dynamic or static instrumentation points
- … each providing data
  - timestamps, process info, function arguments, ...
- … and the D language to process it
  - powerful summaries, or just printf()

Where is DTrace?

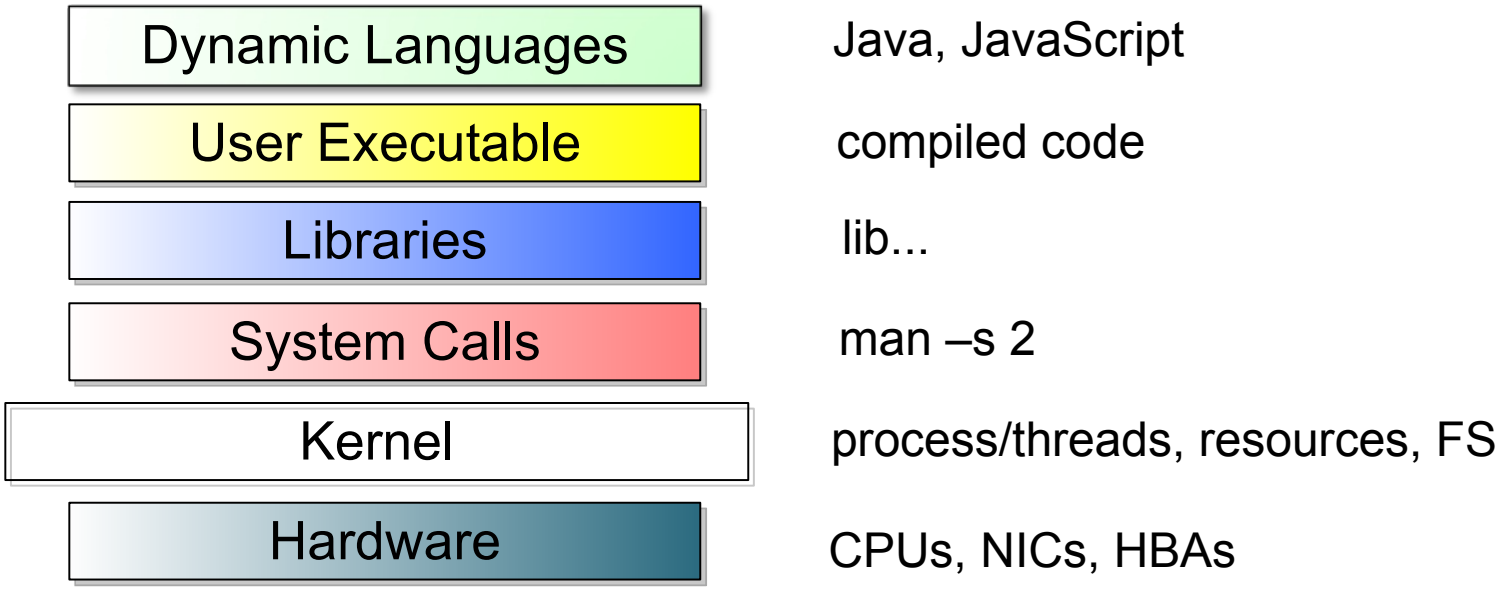
- Oracle Solaris 10+, Oracle Solaris Express
- Mac OS X 10.5+
- FreeBSD 7.1+
- Not yet in Linux, but see:
  http://www.crisp.demon.co.uk/blog/

What Is DTrace For?

- Troubleshooting performance problems
  - Profile applications (and the kernel)
  - Latency measurements
- Troubleshooting software bugs
  - Proving what the problem is, and is not
  - Measuring the magnitude of the problem
- Detailed observability
  - Hardware resources – CPU, disk I/O, network I/O
  - Unmodified application software
  - *The entire software stack*

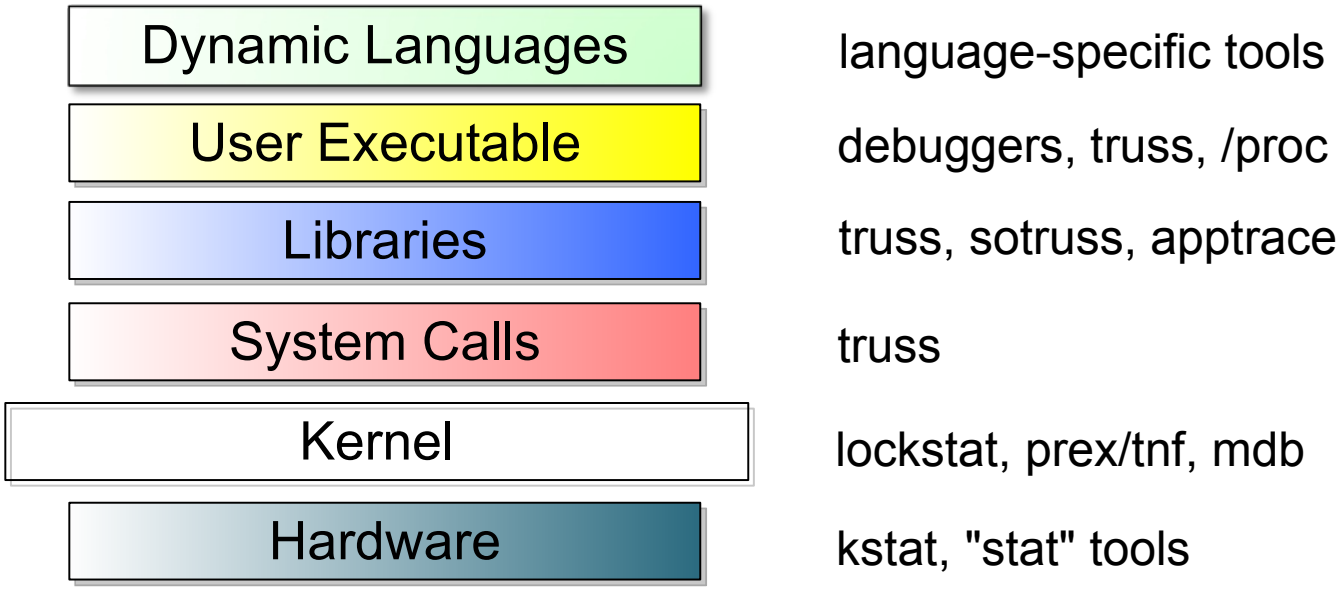
The Entire Software Stack

- How did you analyze these?



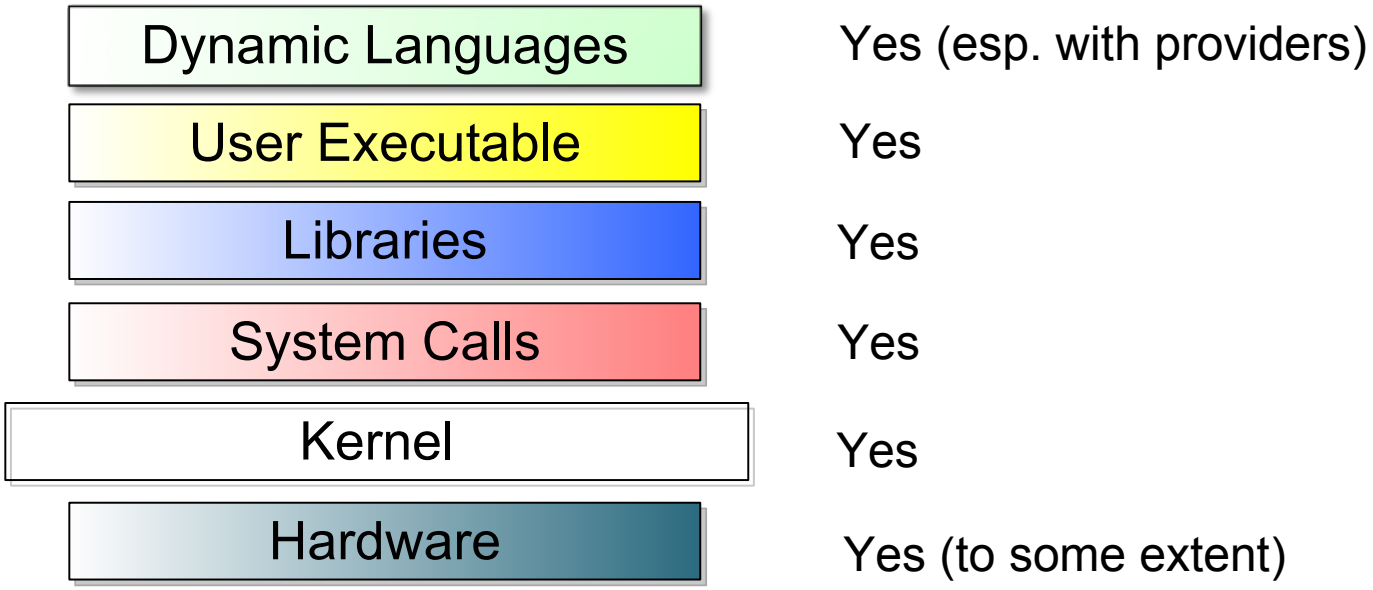
The Entire Software Stack

- It was possible, but difficult (e.g., Solaris):



The Entire Software Stack

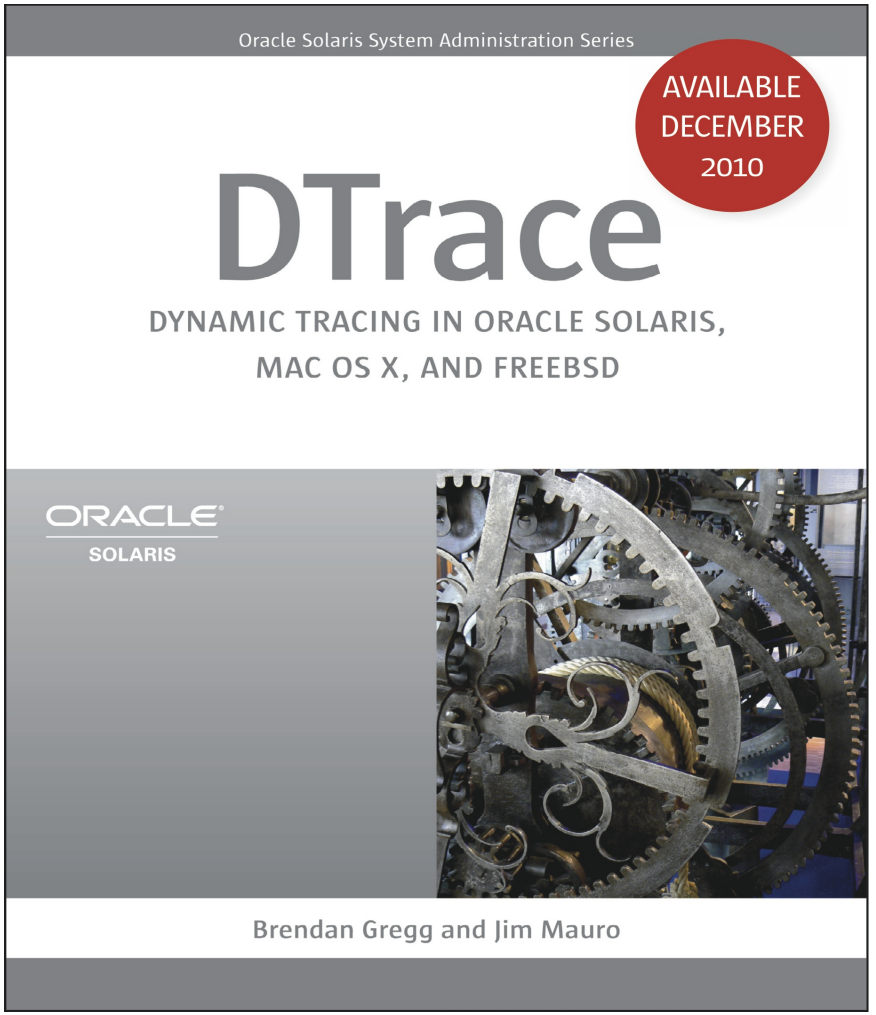
- DTrace is all seeing



The Entire Software Stack – Seeing is Believing

- DTrace cookbook:
  - 230+ scripts
  - 270+ one-liners
  - Shows entire software stack, *by example*
  - Topics include:

CPU, Memory, Disk, Network, SCSI, SATA, IDE, SAS, VFS, UFS, ZFS, HFS+, sockets, IP, TCP, UDP, ICMP, Ethernet, NFS, CIFS, HTTP, DNS, C, Java, JavaScript, Perl, PHP, Python, Ruby, Mysql, Kernel, Apps, ...



A Quick Peek at DTrace…

```
# dtrace -n 'syscall:::entry /execname == "Xorg"/ {
    @[probefunc, ustack()] = count(); }'
dtrace: description 'syscall:::entry ' matched 233 probes
^C
[...]
setitimer
    libc.so.1`setitimer+0xa
    Xorg`WaitForSomething+0x526
    Xorg`Dispatch+0x12d
    Xorg`main+0x681
    Xorg`0x46ed8c
    10
read
    libc.so.1`__read+0xa
    Xorg`_XSERVTransSocketRead+0xf
    Xorg`ReadRequestFromClient+0x284
    Xorg`Dispatch+0x7c4
    Xorg`main+0x681
    Xorg`0x46ed8c
    14
```

DTrace – Unified Observability

- Software analysis tools often constrain view. e.g.:
  - Language specific
  - Software stack layer specific
  - Tied to a development framework
  - Non-production use
- System analysis tools also constrain view:
  - “stat” tools for resource utilization
  - Procfs tools for process/thread views

DTrace – Unified Observability

- DTrace can see all:
  - Locate bugs, *no matter where they are*
  - Locate latency, *no matter where it is*
- Connecting the dots:
  - Show system activity in application context
  - Correlate system resources to who and how they are used

DTrace Features

- Dynamic instrumentation
- Static instrumentation
- Unified interface
- Arbitrary-context kernel instrumentation
- Data integrity
- Arbitrary actions
- Predicates
- High-level control language
- Scalable data aggregation
- Speculative tracing
- Virtualized consumers

Agenda

- *What is DTrace*
- Using DTrace
- Application Analysis
- USDT Probes

DTrace Components (and terminology)

- Consumers
  - DTrace user commands that call into the framework:

    `dtrace(1M), lockstat(1), plockstat(1), intrstat(1)`
  - `dtrace(1M)` is for one-liners and scripting
- Probes
  - A point of instrumentation and data generation
  - User-defined actions can be executed when probes "fire"
- Providers
  - Libraries of probes
  - Static: `io:::start, proc:::exec-success, perl*:::sub-entry, php*:::error, …`
  - Dynamic: `pid816:libc:strlen:entry, fbt:zfs:arc_read:entry, …`

D Program Structure

```
probe, ...
/optional predicate/
{
   optional action;
   ...
}
```

D Script Example

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

syscall::read:entry
/execname == "java"/
{
    @reads[pid, fds[arg0].fi_pathname] = count();
}
```

D Command Line Example

```
# dtrace -n 'syscall::read:entry /execname == "java"/ {
        @reads[pid, fds[arg0].fi_pathname] = count(); }'
```

DTrace Probes (more terminology)

- Have a unique four-tuple name:

  provider:module:function:name

provider	The DTrace provider that manages the probe
module	kernel module or user library location
function	kernel or user function location
name	meaningful probe name

```
fbt:zfs:zfs_read:entry
io:::start
ip:::send
...
```

DTrace Probe Types

- Anchored Probes
  - Instrument a specific location in code

```
fbt:ufs:ufs_read:entry
pid1234:libc:malloc:entry
vminfo:::pgin
perl*:::sub-entry
```

- Unanchored Probes
  - Fire independently of code; eg, time-based

```
profile:::profile-997hz
cpc:::PAPI_l2_tcm-user-10000
```

DTrace Provider Types (more terminology)

- Statically Defined Tracing (SDT)
  - Static (hard-coded) kernel instrumentation
  - Build meaningful and easy to use providers and probes

```
io:::start
proc:::exec-success
```

- User-land Statically Defined Tracing (USDT)
  - Static (hard-coded) application instrumentation
  - Build meaningful and easy to use providers and probes

```
plockstat*:::mutex-acquire
javascript*:::function-entry
mysql*:::query-start
```

DTrace Provider Stability (essential terminology)

- Stable Providers
  - The provider interface (probes & args) won't change
  - Same across OS versions, different OSes
  - Usually achieved with SDT or USDT instrumentation

```
io:::start
ruby*:::function-entry
```

- Unstable Providers
  - May change between software and OS versions
  - D scripts using these may only run in one location

```
pid1234:a.out:do_record:entry
fbt:zfs:arc_read:entry
```

DTrace Predicates

- User-defined conditional statements
  - DTrace's if/then/else
  - Evaluated when probes fire
  - Data pruning at the source

```
/execname == "httpd"/
/pid == 123/
/errno != 0/
```

DTrace Actions

- Action statements
  - Grouped in a clause { … }
  - What to do when a probe fires
  - Data to gather, timestamps for profiling, etc.

```
printf("uid: %d", uid);
self->delta = timestamp - self->start;
ustack();
```

The D Language

- Similar to C and awk(1)
  - ANSI C operators: arithmetic, relational, logical, etc
- Built-in variables
  - execname, pid, tid, uid, cpu, probefunc, curthread, …
- User-defined variables
  - Thread-local, clause-local, global
  - ints, strings, arrays, associative arrays, etc
- Built-in functions
  - printf(), stack(), ustack(), copyin(), copyinstr(), raise(), ...
- Aggregations
  - A new *scaleable* data type to summarize values

DTrace Aggregations

- New data type; assigned using:

```
@name[key] = aggfunc(arg);
```

- Scaleable
  - maintained per-CPU and later combined
- Summarize data in powerful ways:
  - count occurrences `count()`
  - average, sum, min, max `avg() sum() min() max()`
  - distribution plot (2-power) `quantize()`
  - distribution plot (linear) `lquantize()`
- … and, all of the above by one or more keys
  - instant frequency count – no perl/awk post processing

```
@who[execname] = count();
```

Agenda

- *What is DTrace*
- *Using DTrace*
- Application Analysis
- USDT Probes

Applications Analysis With DTrace

ORACLE

Observing and Profiling Applications

- DTrace can observe unmodified applications
  - Using DTrace providers such as:
    - syscall – which system calls are being executed
    - profile/tick – time based profiles
    - sched – scheduling activity
    - pid – dynamic probes in the user-land code
    - vminfo, sysinfo – kernel and virtual memory statistics
    - proc – process/thread events
- Or, you can modify your application and add custom DTrace providers (USDT)

Strategy for Using DTrace to Analyze Applications

1. Canned tools:
   - System: `vmstat(1M)`, `iostat(1M)`
   - Process: `prstat(1)`, `top(1)`
   - DTrace: DTraceToolkit one-liners, scripts
2. Application familiarization:
   - functional diagrams: targets for DTrace
   - existing logs and statistics: take further with DTrace
   - get/write load generation tools: sanity test DTrace scripts
3. DTrace programs:
   - check for application USDT providers
   - system calls
   - system resources: CPU, disk, net/sockets, memory
4. Application internals:
   - USDT language provider, if available: perl, python, java
   - pid provider

Strategy Examples: 1. Canned Tools

- System tools; eg, `vmstat(1M)`

```
solaris# vmstat 1
 kthr      memory            page            disk          faults      cpu
 r b w   swap  free  re  mf pi po fr de sr m1 m1 m1 m2   in   sy   cs us sy id
 0 0 0 148132968 72293376 3 11 2 0 0  0  0 14 14 14  1 13671 30379 23848 2 1 96
 0 2 0 138621040 59492152 9 97 0 0 0  0  0 161 160 161 0 243611 503049 436315 46 25 28
 4 1 0 138620616 59491792 0 30 0 0 0  0  0 189 189 190 0 241980 508303 434517 47 25 27
 4 1 0 138620616 59491776 0 0 0 0  0  0  0 142 142 141 0 239497 507814 431138 46 25 29
[...]
```

- Memory looks ok (high free, no sr)
- High number of syscalls
  - by who? which types? what args? any errors? … and why?
- CPU load: user and system time
  - who? what functions/modules? what codepaths?
- … answer using DTrace

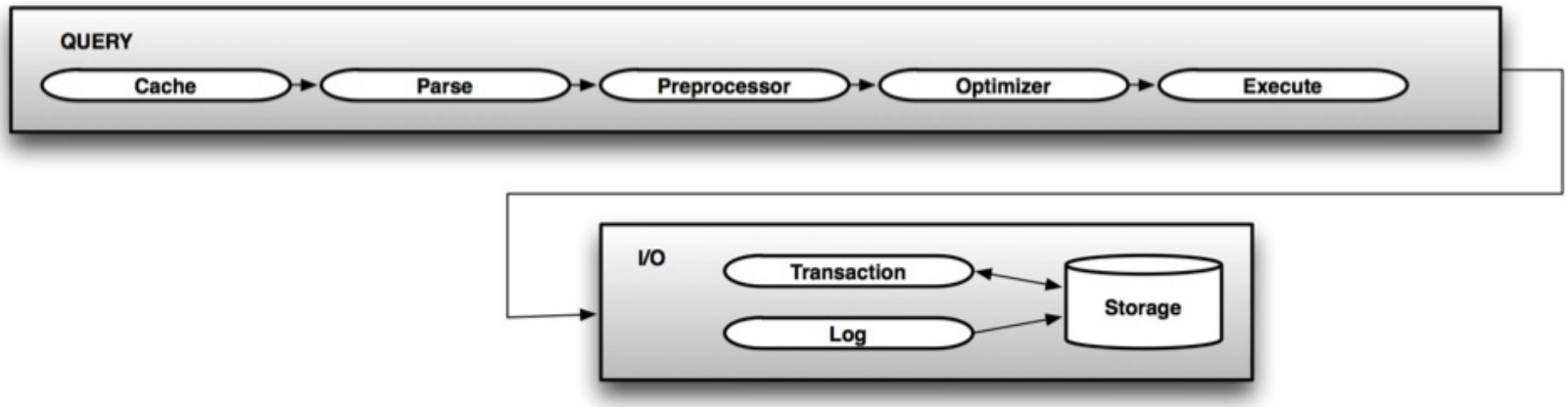
Strategy Examples: 1. Canned Tools

- ... DTrace one-liner for syscalls by pid:

```
solaris# dtrace -n 'syscall:::entry { @[pid, probefunc] = count(); }'
dtrace: description 'syscall:::entry ' matched 236 probes
^C
[...]
       47  read                                                  4781
       47  llseek                                                6307
     1025  close                                                 6717
     1025  fsat                                                  7907
     1025  fstat64                                               8759
     1025  doorfs                                               10804
     1025  getpid                                               11782
     1025  portfs                                               12513
     1025  read                                                 23655
       47  doorfs                                               28956
     1016  ioctl                                                35097
     1025  llseek                                               86897
     1025  ioctl                                               233267
```

Strategy Examples: 2. Application Familiarization

- For example, a functional diagram such as:



- … provides many targets for DTrace analysis

Strategy Examples: 3. DTrace Programs

- DEMO

Strategy Examples: 3. DTrace Programs

- Check for application USDT providers; eg, mysql:

```
solaris# dtrace -n 'mysql*:::query-start { @[copyinstr(arg0)] = count(); }'
dtrace: description 'mysql*:::query-start ' matched 1 probe
^C
[...]
  SELECT /* MediaWikiBagOStuff::_doquery 192.168.1.109 */ value,exptime
FROM `objectcache` WHERE keyname='wikidb:messageslock'                4
  SELECT /* MessageCache::loadFromDB 192.168.1.109 */  page_title  FROM
`page`  WHERE (page_len > 10000) AND page_is_redirect = '0' AND
page_namespace = '8'                  4
  SELECT /* MessageCache::loadFromDB 192.168.1.109 */  page_title,old_text,
old_flags  FROM `page`,`revision`,`text`  WHERE page_is_redirect = '0' AND
page_namespace = '8' AND (page_latest=rev_id) AND (rev_text_id=old_id) AND
(page_len <= 10000)                   4
  SET /* Database::open 192.168.1.109 */ sql_mode = ''                 4
  COMMIT                                                             10
  BEGIN                                                              11
```

Strategy Examples: 3. DTrace Programs

- Examine syscalls; eg, by user-land stack:

```
solaris# dtrace -n 'syscall::write*:entry /execname == "httpd"/ {
@[ustack()] = count(); }'
dtrace: description 'syscall::write*:entry ' matched 2 probes
[...]
              libc.so.1`__writev+0x7
              libapr-1.so.0.3.9`apr_socket_sendv+0x8d
              httpd`writev_it_all+0x55
              httpd`ap_core_output_filter+0x72f
              mod_ssl.so`bio_filter_out_flush+0xc3
              mod_ssl.so`ssl_io_filter_output+0x281
              httpd`ap_http_header_filter+0x7c8
              httpd`ap_content_length_filter+0x1d3
              mod_deflate.so`deflate_out_filter+0xaf3
              httpd`end_output_stream+0x4e
              httpd`ap_process_request+0x93
              httpd`ap_process_http_connection+0x5b
              httpd`ap_run_process_connection+0x28
              httpd`child_main+0x3d8
              httpd`make_child+0x86
              httpd`ap_mpm_run+0x410
              httpd`main+0x812
              httpd`_start+0x7d
               12
```

Strategy Examples: 3. DTrace Programs

- Examine resources; eg, CPU profiling:

```
solaris# dtrace -n 'profile-101 /execname == "httpd"/ {
@[ustack(5)] = count(); }'
dtrace: description 'profile-101 ' matched 1 probe
[...]

              libc.so.1`__pollsys+0x7
              libc.so.1`poll+0x4c
              libapr-1.so.0.3.9`apr_wait_for_io_or_timeout+0x9e
              libapr-1.so.0.3.9`apr_socket_recv+0x95
              libaprutil-1.so.0.3.9`socket_bucket_read+0x5b
               17

              libc.so.1`memcpy+0x1b
              mod_php5.2.so`zend_do_push_object+0x2a
              mod_php5.2.so`zendparse+0x1292
              mod_php5.2.so`compile_file+0x102
              mod_php5.2.so`dtrace_compile_file+0x2e
               19
```

Strategy Examples: 4. Application Internals

- DEMO

Strategy Examples: 4. Application Internals

- DTrace one-liner to count user-land functions:

```
solaris# dtrace -n 'pid$target:mysqld::entry { @[probefunc] = count(); }'
-p `pgrep -n mysqld`
dtrace: description 'pid$target:mysqld::entry ' matched 13313 probes
[...]
  handle_one_connection                                             1
[...]
  __1cJPROFILINGPstart_new_query6Mpkc_v_                           28
  __1cJPROFILINGUfinish_current_query6M_v_                         28
[...]
  __1cJItem_funcEtype6kM_nEItemEType__                            475
  net_store_length                                                514
  my_mb_wc_latin1                                                 515
  __1cJsql_alloc6FI_pv_                                           534
  my_strnncollsp_8bit_bin                                         549
  __1cEItemLused_tables6kM_X_                                     565
  my_strcasecmp_utf8                                              688
  my_charset_same                                                 976
  ha_key_cmp                                                     1094
  alloc_root                                                     1143
  _mi_get_binary_pack_key                                        1210
  my_wc_mb_latin1                                                2934
  my_utf8_uni                                                    3457
```

Strategy Examples: 4. Application Internals

- DTrace one-liner to trace user-land function flow:

```
solaris# dtrace -Fn 'pid$target:mysqld::entry,pid$target:mysqld::return {
trace(timestamp); }' -p `pgrep -n mysqld`
dtrace: description 'pid$target:mysqld::entry,pid$target:mysqld::return '
matched 26611 probes
CPU FUNCTION
  4  <- os_thread_sleep                          3210680878965941
  4  -> log_get_lsn                              3210680878984505
  4    -> mutex_enter_func                       3210680878993052
  4      -> os_fast_mutex_trylock                3210680879000119
  4      <- os_fast_mutex_trylock                3210680879005680
  4    <- mutex_enter_func                       3210680879011652
  4    -> mutex_exit                             3210680879018121
  4      -> os_fast_mutex_unlock                 3210680879022560
  4      <- os_fast_mutex_unlock                 3210680879024998
  4    <- mutex_exit                             3210680879031610
  4  <- log_get_lsn                              3210680879037329
  4  -> ut_dulint_cmp                            3210680879047672
  4  <- ut_dulint_cmp                            3210680879053409
  4  -> sync_arr_wake_threads_if_sema_free       3210680879074676
  4    -> os_mutex_enter                         3210680879084298
[...]
```

Checklist for Using DTrace to Analyze Applications

- On-CPU Time
  - High %CPU
  - Determine where the time is being spent
- Off-CPU Time
  - Typically sleeping, waiting for I/O (disk, network), waiting for locks, or waiting for a CPU
- Volume
  - Hot function/code path
- Locks
  - Spin and/or blocking on mutex or R/W locks
- Memory Allocation
  - malloc (heap) management
- Errors
  - User functions, libraries, system calls encountering errors that are not being handled properly, or very high error rates inducing poor performance

Agenda

- *What is DTrace*
- *Using DTrace*
- *Application Analysis*
- USDT Probes

DTrace Custom Probes in Applications

- User Statically Defined Tracing (USDT) probes
  - Add custom probes to source code
  - Non-enabled probe effect is still negligible (extra NOPs)
  - Requires running instrumented binaries in production
  - Abstract what the software does to meaningful probe names
    - e.g. transaction-start, log-write, client-request, etc
  - Determine what data should be made available to the probes
  - Enhances application observability and debugging

Adding USDT Probes to Programs

- Adding USDT probes is easy
- Deciding what to add is hard
  - Useful probes with intuitive names
  - Necessary arguments
  - And *committing* to the interface

    (if you want a stable provider)

USDT – Provider and Probe Example

- Create a .d file with the provider and probes
- Optionally include stability level definitions
- For example, `probes.d`:

```
provider world {
    probe loop(int);
}

#pragma D attributes Evolving/Evolving/Common provider world provider
#pragma D attributes Private/Private/Common provider world module

[…]
```

Add USDT Probe To The Source File

Original Source File

```
#include <stdio.h>
#include <unistd.h>

int
main(int ac, char **av)
{
     int i;
     for (i = 0; i < 5; i++) {
          printf("Hello World\n");
          sleep(2);
     }
}
```

DTrace USDT Probe Added

```
#include <stdio.h>
#include <unistd.h>
#include <sys/sdt.h>

int
main(int ac, char **av)
{
     int i;
     for (i = 0; i < 5; i++) {
          DTRACE_PROBE1(world, loop, i);
          printf("Hello World\n");
          sleep(2);
     }
}
```

USDT – Compile the Code, Take it For a Spin

```
solaris# cc -c helloworld.c
solaris# dtrace -G -s probes.d helloworld.o
solaris# cc -o helloworld -ldtrace probes.o helloworld.o
```

```
solaris# dtrace -q -c ./helloworld -n 'world$target:::loop {
    printf("%s:%s loop = %d\n", probemod, probefunc, arg0);
}'
helloworld:main loop = 0
Hello World
helloworld:main loop = 1
Hello World
helloworld:main loop = 2
Hello World
helloworld:main loop = 3
Hello World
helloworld:main loop = 4
```

USDT headergen

- -h option to dtrace(1)
- Generates header file with externs and macros
- PROVIDER_PROBE() macros instead of generic DTRACE_PROBEn()

USDT is-enabled

- USDT overhead is very low, but…
- Setting up arguments can be expensive
- PROVIDER_ENABLED() macros introduced
  - Conditional statement preceeds probe macro

```
if (PROVIDER_PROBE_ENABLED()) }
  PROVIDER_PROBE(args, …);
}
```

- is-enabled has slightly higher probe effect
- Use of is-enabled or not should be considered on a case-by-case basis
  - If the cost of non is-enabed probes is high, use is-enabled

http://www.opensolaris.org/jive/thread.jspa?messageID=31314

USDT is-enabled / headergen example

- Create the header file:

```
solaris# dtrace -h -s probes.d
```

- Include the header file, conditional statement, and probe:

```
#include <stdio.h>
#include <unistd.h>
#include <sys/sdt.h>
#include "probes.h"

int main(int ac, char **av) {
    int i;
    for (i = 0; i < 5; i++) {
        if (WORLD_LOOP_ENABLED()) {
            /* Lots of stuff that takes time */
            WORLD_LOOP(i);
        }
    printf("Hello World\n");
    sleep(2);
    }
}
```

DTrace USDT Probes

- References, Examples

http://blogs.sun.com/barts/entry/putting_user_defined_dtrace_probe

http://blogs.sun.com/ahl/entry/user_land_tracing_gets_better

http://blogs.sun.com/dap/entry/writing_a_dtrace_usdt_provider

http://blogs.sun.com/tdd/entry/a_simple_dtrace_usdt_example

http://www.opensolaris.org/jive/thread.jspa?messageID=23815

Summary

- DTrace enables observing, profiling and analyzing applications through dynamic and static instrumentation and use of available providers
- Every aspect of application behavior can be measured and understood
- DTrace USDT probes further enhance observability, enabling developers to instrument their code in meaningful and useful ways

DTrace Resources

Websites:

- http://wikis.sun.com/display/DTrace/Documentation
- http://dtrace.org
- http://solarisinternals.com
- http://hub.opensolaris.org/bin/view/Community+Group+dtrace/WebHome

Books:

- DTrace Guide (see 1st URL)
- Solaris Performance and Tools: DTrace and MDB Techniques for Solaris 10 and OpenSolaris
- DTrace: Dynamic Tracing in Oracle Solaris, Mac OS X and FreeBSD (Dec 2010)

Question Time

SOFTWARE. HARDWARE. COMPLETE.


ORACLE®

Supplemental Example Slides

Using DTrace to measure read and write throughput

```
ds2# cat rw.d
#!/usr/sbin/dtrace -qs

#pragma D option switchrate=333hz
#pragma D option dynvarsize=4m

syscall::read:entry,
syscall::write:entry
/pid == 1656/
{
      self->flag[probefunc] = 1;
      self->fs[probefunc] = fds[arg0].fi_fs;
}
syscall::read:return,
syscall::write:return
/arg0 > 0 && self->flag[probefunc]/
{
      @bytes[probefunc,self->fs[probefunc]] = sum(arg0);
      self->flag[probefunc] = 0;
      self->fs[probefunc] = 0;
}
tick-1sec
{
      printf("%-8s %-12s %-12s\n","SYSCALL","FILE SYS","BYTES-PER-SEC");
      printa("%-8s %-12s %-@12d\n",@bytes);
      trunc(@bytes);

      printf("\n\n");
}
tick-10sec
{
      exit(0);
}
```

Running the rw.d Script

```
ds2# ./rw.d
[...]

SYSCALL  FILE SYS     BYTES-PER-SEC
read     fifofs       19957
write    fifofs       19974
write    sockfs       647859
read     sockfs       728080
write    zfs          2553721


SYSCALL  FILE SYS     BYTES-PER-SEC
read     fifofs       20122
write    fifofs       20122
write    sockfs       652797
read     sockfs       733298
write    zfs          2553698

[...]
```

Which ZFS file?

```
ds2# cat zfs.d
#!/usr/sbin/dtrace -qs

#pragma D option switchrate=10hz
#pragma D option dynvarsize=4m

syscall::read:entry,
syscall::write:entry
/pid == 1656 && fds[arg0].fi_fs == "zfs"/
{
      self->flag[probefunc] = 1;
      self->file[probefunc] = fds[arg0].fi_pathname;
}
syscall::read:return,
syscall::write:return
/arg0 > 0 && self->flag[probefunc]/
{
      @bytes[probefunc,self->file[probefunc]] = sum(arg0);
      self->flag[probefunc] = 0;
      self->file[probefunc] = 0;
}
tick-1sec
{
      printf("%-8s %-32s %-12s\n","SYSCALL","ZFS FILE","BYTES-PER-SEC");
      printa("%-8s %-32s %-@12d\n",@bytes);
      trunc(@bytes);

      printf("\n\n");
}
tick-10sec
{
      exit(0);
}
```

Running zfs.d

```
ds2# ./zfs.d

SYSCALL  ZFS FILE                          BYTES-PER-SEC
write    /logs/info/ds1-100/access         2684598

SYSCALL  ZFS FILE                          BYTES-PER-SEC
write    /logs/info/ds1-100/access         2750060

SYSCALL  ZFS FILE                          BYTES-PER-SEC
write    /logs/info/ds1-100/access         2618873

SYSCALL  ZFS FILE                          BYTES-PER-SEC
write    /logs/info/ds1-100/access         2881368

SYSCALL  ZFS FILE                          BYTES-PER-SEC
write    /logs/info/ds1-100/access         2684255
```

Measuring ZFS file write latency

```
ds2# cat zfst.d
#!/usr/sbin/dtrace -qs

#pragma D option switchrate=333hz
#pragma D option dynvarsize=4m

syscall::write:entry
/pid == 1656 && fds[arg0].fi_fs == "zfs"/
{
    self->st = timestamp;
}
syscall::write:return
/self->st/
{
    @t = quantize(timestamp - self->st);
    self->st = 0;
}
tick-10sec
{
    printa(@t);
    exit(0);
}
```

Measuring ZFS file write latency

```
ds2 # ./zfst.d

           value  ------------- Distribution ------------- count
            2048 |                                         0
            4096 |                                         3
            8192 |                                         1
           16384 |                                         1
           32768 |                                         0
           65536 |@                                        5
          131072 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ 219
          262144 |@@@@@@@@@@@@                             91
          524288 |                                         0
```

Time-based user-mode profile

```
ds2 # dtrace -qn 'profile-997hz /arg1 && pid == 1656/ { @[ufunc(arg1)] = count(); }
          tick-10sec { trunc(@,20); printa(@); exit(0); }'

  libdb.so.5`__bam_cmp                                              507
  libslapd.so.1`PL_HashTableRawLookup_const                         528
  libslapd.so.1`slapi_attr_basetype                                 533
  libslapd.so.1`slapi_pblock_get                                    541
  libc.so.1`mutex_lock_impl                                         553
  libc.so.1`clear_lockbyte                                          564
  libdb.so.5`__db_pthread_mutex_lock                                582
  libslapd.so.1`value_done                                          596
  libslapd.so.1`attrlist_free                                       601
  libc.so.1`memset                                                  637
  libc.so.1`fast_process_lock                                       662
  libdb.so.5`__memp_fget                                            831
  libsmartheap_smp64.so`shi_test_and_clear                          901
  libdb.so.5`__env_alloc                                           1685
  libdb.so.5`__env_size_insert                                     1936
  libsmartheap_smp64.so`shi_allocSmall2                            2219
  libc.so.1`mutex_unlock                                           3300
  libc.so.1`mutex_trylock                                         41905
  libc.so.1`set_lock_byte64                                       51661
  libc.so.1`atomic_cas_64                                        127256
```

User stack trace for mutex_trylock() call

```
ds2 # dtrace -qn 'pid1656:libc:mutex_trylock:entry { @[ustack()] =
count(); }'
[…]

              libc.so.1`mutex_trylock
              libdb.so.5`__db_pthread_mutex_lock+0x78
              libdb.so.5`__memp_alloc+0xa83
              libdb.so.5`__memp_fget+0x13dc
              libdb.so.5`__bam_search+0xa86
              libdb.so.5`__bamc_search+0x6f2
              libdb.so.5`__bamc_get+0x146
              libdb.so.5`__dbc_get+0x4cf
              libdb.so.5`__db_get+0xd2
              libdb.so.5`__db_get_pp+0x1d6
              libback-ldbm.so`idl_fetch_one+0x79
              libback-ldbm.so`idl_old_fetch+0x7e
              libback-ldbm.so`index_read_ext+0x326
              libback-ldbm.so`index_read+0xf
              libback-ldbm.so`keys2idl+0xce
              libback-ldbm.so`ava_candidates+0x4cc
              libback-ldbm.so`filter_candidates+0x313
              libback-ldbm.so`list_candidates+0x42e
              libback-ldbm.so`filter_candidates+0x178
              libback-ldbm.so`subtree_candidates+0x76
            3788020
```