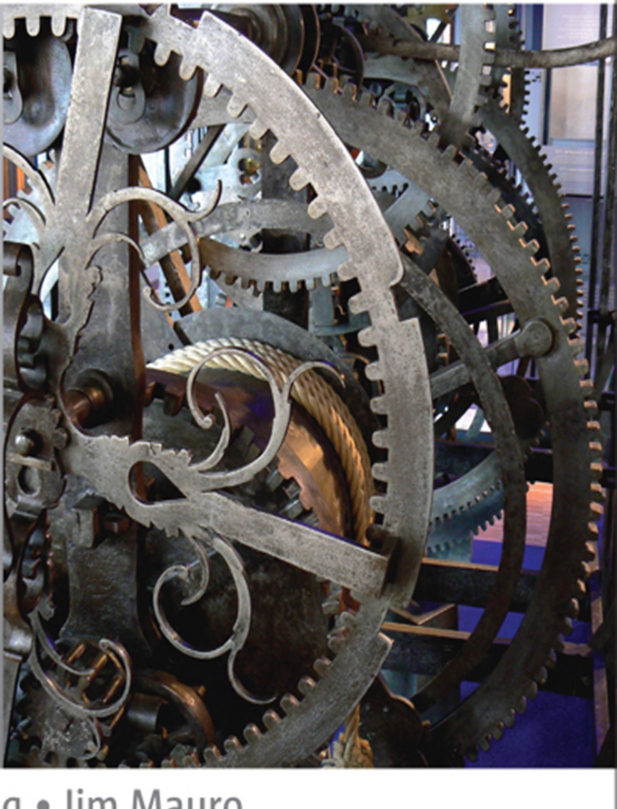
DTrace

Dynamic Tracing in Oracle® Solaris, Mac OS X, and FreeBSD



Brendan Gregg • Jim Mauro

Foreword by Bryan Cantrill

DTrace

```
  getpage                                                        1700
  getattr                                                        3221
  cmp                                                           48342
  putpage                                                       77557
  inactive                                                      80786
  lookup                                                        86059
```

The most frequent vnode operation was `lookup()`, called 86,059 times while this one-liner was tracing.

File System Calls by Mountpoint

The fsinfo provider has `fileinfo_t` as `args[0]`. Here the mountpoint is frequency counted by fsinfo probe call, to get a rough idea of how busy (by call count) file systems are as follows:

```
# dtrace -n 'fsinfo::: { @[args[0]->fi_mount] = count(); }'
dtrace: description 'fsinfo::: ' matched 44 probes
^C

  /home                                                             8
  /builds/bmc                                                       9
  /var/run                                                         11
  /builds/ahl                                                      24
  /home/brendan                                                    24
  /etc/svc/volatile                                                47
  /etc/svc                                                         50
  /var                                                             94
  /net/fw/export/install                                          176
  /ws                                                             252
  /lib/libc.so.1                                                  272
  /etc/mnttab                                                     388
  /ws/onnv-tools                                                 1759
  /builds/brendan                                               17017
  /tmp                                                         156487
  /                                                            580819
```

Even though I'm doing a source build in `/builds/brendan`, it's the root file system on / that has received the most file system calls.

Bytes Read by Filename

The fsinfo provider gives an abstracted file system view that isn't dependent on syscall variants such as `read()`, `pread()`, `pread64()`, and so on.

```
# dtrace -n 'fsinfo:::read { @[args[0]->fi_pathname] = sum(arg1); }'
dtrace: description 'fsinfo:::read ' matched 1 probe
^C

  /usr/bin/chmod                                                  317
  /home/brendan/.make.machines                                    572
```

continues

```
  /usr/bin/chown                                                  951
  <unknown>                                                      1176
  /usr/bin/chgrp                                                 1585
  /usr/bin/mv                                                    1585
[...output truncated...]
  /builds/brendan/ak-on-new/usr/src/uts/intel/Makefile.rules          325056
  /builds/brendan/ak-on-new/usr/src/uts/intel/Makefile.intel.shared          415752
  /builds/brendan/ak-on-new/usr/src/uts/intel/arn/.make.state          515044
  /builds/brendan/ak-on-new/usr/src/uts/Makefile.uts           538440
  /builds/brendan/ak-on-new/usr/src/Makefile.master            759744
  /builds/brendan/ak-on-new/usr/src/uts/intel/ata/.make.state          781904
  /builds/brendan/ak-on-new/usr/src/uts/common/Makefile.files          991896
  /builds/brendan/ak-on-new/usr/src/uts/common/Makefile.rules         1668528
  /builds/brendan/ak-on-new/usr/src/uts/intel/genunix/.make.state          5899453
```

The file being read the most is a `.make.state` file: During tracing, more than 5MB was read from the file. The fsinfo provider traces these reads to the file system: The file may have been entirely cached in DRAM or read from disk. To determine how the read was satisfied by the file system, we'll need to DTrace further down the I/O stack (see the "Scripts" section and Chapter 4, Disk I/O).

Bytes Written by Filename

During tracing, a `.make.state.tmp` file was written to the most, with more than 1MB of writes. As with reads, this is writing to the file system. This may not write to disk until sometime later, when the file system flushes dirty data.

```
# dtrace -n 'fsinfo:::write { @[args[0]->fi_pathname] = sum(arg1); }'
dtrace: description 'fsinfo:::write ' matched 1 probe
^C

  /tmp/DAA1RaGkd                                                   22
  /tmp/DAA5JaO6c                                                   22
[...truncated...]
  /tmp/iroptEAA.1524.dNaG.c                                    250588
  /tmp/acompBAA.1443.MGay0c                                    305541
  /tmp/iroptDAA.1443.OGay0c                                    331906
  /tmp/acompBAA.1524.aNaG.c                                    343015
  /tmp/iroptDAA.1524.cNaG.c                                    382413
  /builds/brendan/ak-on-new/usr/src/cmd/fs.d/.make.state.tmp          1318590
```

Read I/O Size Distribution by File System Mountpoint

This output shows a distribution plot of read size by file system. The `/builds/brendan` file system was usually read at between 1,024 and 131,072 bytes per read. The largest read was in the 1MB to 2MB range.

```
# dtrace -n 'fsinfo:::read { @[args[0]->fi_mount] = quantize(arg1); }'
dtrace: description 'fsinfo:::read ' matched 1 probe
^C
```

```
  /builds/bmc
           value  ------------- Distribution ------------- count
              -1 |                                         0
               0 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ 2
               1 |                                         0

[...output truncated...]

  /builds/brendan
           value  ------------- Distribution ------------- count
              -1 |                                         0
               0 |@                                        15
               1 |                                         0
               2 |                                         0
               4 |                                         0
               8 |                                         0
              16 |                                         0
              32 |                                         0
              64 |@@                                       28
             128 |                                         0
             256 |                                         0
             512 |@@                                       28
            1024 |@@@@@@@                                  93
            2048 |@@@@                                     52
            4096 |@@@@@@                                   87
            8192 |@@@@@@@                                  94
           16384 |@@@@@@@@                                 109
           32768 |@@                                       31
           65536 |@@                                       30
          131072 |                                         0
          262144 |                                         2
          524288 |                                         1
         1048576 |                                         1
         2097152 |                                         0
```

Write I/O Size Distribution by File System Mountpoint

During tracing, /tmp was written to the most (listed last), mostly with I/O sizes between 4KB and 8KB.

```
# dtrace -n 'fsinfo:::write { @[args[0]->fi_mount] = quantize(arg1); }'
dtrace: description 'fsinfo:::write ' matched 1 probe
^C

  /etc/svc/volatile
           value  ------------- Distribution ------------- count
             128 |                                         0
             256 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ 34
             512 |                                         0
[...]

  /tmp
           value  ------------- Distribution ------------- count
               2 |                                         0
               4 |                                         1
               8 |                                         4
              16 |@@@@                                     121
              32 |@@@@                                     133
              64 |@@                                       56
             128 |@@                                       51
```

continues

```
             256 |@                                        46
             512 |@                                        39
            1024 |@                                        32
            2048 |@@                                       52
            4096 |@@@@@@@@@@@@@@@@@@@@@@@@                 820
            8192 |                                         0
```

One-Liners: sdt Provider Examples

Who Is Reading from the ZFS ARC?

This shows who is performing reads to the ZFS ARC (the in-DRAM file system cache for ZFS) by counting the stack backtraces for all ARC accesses. It uses SDT probes, which have been in the ZFS ARC code for a while:

```
# dtrace -n 'sdt:::arc-hit,sdt:::arc-miss { @[stack()] = count(); }'
dtrace: description 'sdt:::arc-hit,sdt:::arc-miss ' matched 3 probes
^C
[...]

              zfs`arc_read+0x75
              zfs`dbuf_prefetch+0x131
              zfs`dmu_prefetch+0x8f
              zfs`zfs_readdir+0x4a2
              genunix`fop_readdir+0xab
              genunix`getdents64+0xbc
              unix`sys_syscall32+0x101
              245

              zfs`dbuf_hold_impl+0xea
              zfs`dbuf_hold+0x2e
              zfs`dmu_buf_hold_array_by_dnode+0x195
              zfs`dmu_buf_hold_array+0x73
              zfs`dmu_read_uio+0x4d
              zfs`zfs_read+0x19a
              genunix`fop_read+0x6b
              genunix`read+0x2b8
              genunix`read32+0x22
              unix`sys_syscall32+0x101
              457

              zfs`dbuf_hold_impl+0xea
              zfs`dbuf_hold+0x2e
              zfs`dmu_buf_hold+0x75
              zfs`zap_lockdir+0x67
              zfs`zap_cursor_retrieve+0x74
              zfs`zfs_readdir+0x29e
              genunix`fop_readdir+0xab
              genunix`getdents64+0xbc
              unix`sys_syscall32+0x101
             1004

              zfs`dbuf_hold_impl+0xea
              zfs`dbuf_hold+0x2e
              zfs`dmu_buf_hold+0x75
              zfs`zap_lockdir+0x67
              zfs`zap_lookup_norm+0x55
              zfs`zap_lookup+0x2d
```

```
                  zfs`zfs_match_find+0xfd
                  zfs`zfs_dirent_lock+0x3d1
                  zfs`zfs_dirlook+0xd9
                  zfs`zfs_lookup+0x104
                  genunix`fop_lookup+0xed
                  genunix`lookuppnvp+0x3a3
                  genunix`lookuppnat+0x12c
                  genunix`lookupnameat+0x91
                  genunix`cstatat_getvp+0x164
                  genunix`cstatat64_32+0x82
                  genunix`lstat64_32+0x31
                  unix`sys_syscall32+0x101
                 2907
```

This output is interesting because it demonstrates four different types of ZFS ARC read. Each stack is, in order, as follows.

1. `prefetch read`: ZFS performs prefetch before reading from the ARC. Some of the prefetch requests will actually just be cache hits; only the prefetch requests that miss the ARC will pull data from disk.
2. `syscall read`: Most likely a process reading from a file on ZFS.
3. `read dir`: Fetching directory contents.
4. `stat`: Fetching file information.

Scripts

Table 5-4 summarizes the scripts that follow and the providers they use.

Table 5-4 Script Summary

Script	Target	Description	Providers
`sysfs.d`	Syscalls	Shows reads and writes by process and mountpoint	syscall
`fsrwcount.d`	Syscalls	Counts read/write syscalls by file system and type	syscall
`fsrwtime.d`	Syscalls	Measures time in read/write syscalls by file system	syscall
`fsrtpk.d`	Syscalls	Measures file system read time per kilobyte	syscall
`rwsnoop`	Syscalls	Traces syscall read and writes, with FS details	syscall
`mmap.d`	Syscalls	Traces `mmap()` of files with details	syscall
`fserrors.d`	Syscalls	Shows file system syscall errors	syscall

continues

Table 5-4 Script Summary (*Continued*)

Script	Target	Description	Providers
`fswho.d`[1]	VFS	Summarizes processes and file read/writes	fsinfo
`readtype.d`[1]	VFS	Compares logical vs. physical file system reads	fsinfo, io
`writetype.d`[1]	VFS	Compares logical vs. physical file system writes	fsinfo, io
`fssnoop.d`	VFS	Traces file system calls using fsinfo	fsinfo
`solvfssnoop.d`	VFS	Traces file system calls using fbt on Solaris	fbt
`macvfssnoop.d`	VFS	Traces file system calls using fbt on Mac OS X	fbt
`vfssnoop.d`	VFS	Traces file system calls using vfs on FreeBSD	vfs
`sollife.d`	VFS	Shows file creation and deletion on Solaris	fbt
`maclife.d`	VFS	Shows file creation and deletion on Mac OS X	fbt
`vfslife.d`	VFS	Shows file creation and deletion on FreeBSD	vfs
`dnlcps.d`	VFS	Shows Directory Name Lookup Cache hits by process[2]	fbt
`fsflush_cpu.d`	VFS	Shows file system flush tracer CPU time[2]	fbt
`fsflush.d`	VFS	Shows file system flush statistics[2]	profile
`ufssnoop.d`	UFS	Traces UFS calls directly using fbt[2]	fbt
`ufsreadahead.d`	UFS	Shows UFS read-ahead rates for sequential I/O[2]	fbt
`ufsimiss.d`	UFS	Traces UFS inode cache misses with details[2]	fbt
`zfssnoop.d`	ZFS	Traces ZFS calls directly using fbt[2]	fbt
`zfsslower.d`	ZFS	Traces slow HFS+ read/writes[2]	fbt
`zioprint.d`	ZFS	Shows ZIO event dump[2]	fbt
`ziosnoop.d`	ZFS	Shows ZIO event tracing, detailed[2]	fbt
`ziotype.d`	ZFS	Shows ZIO type summary by pool[2]	fbt
`perturbation.d`	ZFS	Shows ZFS read/write time during given perturbation[2]	fbt
`spasync.d`	ZFS	Shows SPA sync tracing with details[2]	fbt
`hfssnoop.d`	HFS+	Traces HFS+ calls directly using fbt[3]	fbt
`hfsslower.d`	HFS+	Traces slow HFS+ read/writes[3]	fbt
`hfsfileread.d`	HFS+	Shows logical/physical reads by file[3]	fbt
`pcfsrw.d`	PCFS	Traces pcfs (FAT16/32) read/writes[2]	fbt
`cdrom.d`	HSFS	Traces CDROM insertion and mount[2]	fbt
`dvd.d`	UDFS	Traces DVD insertion and mount[2]	fbt
`nfswizard.d`	NFS	Summarizes NFS performance client-side[2]	io