



DTrace for Beginners

Michael Clarke
Sun Microsystems
March 23rd 2008



Overview

- Tracing of the kernel, user level programs and libraries.
- Developed by Bryan Cantrill, Adam Leventhal and Mike Shaprio.
- Perfect for performance investigation and troubleshooting.
- Over 50,000 points in the kernel that can be traced.
- System is dynamically modified to enable trace points – zero performance hit when disabled.
- Safe to use on production systems.

Terminology

consumer : “dtrace”

/usr/bin/dtrace is a consumer of the DTrace framework (via libdtrace).

probe : “syscall::exece:return”

Describes what points in the software to trace.

action : “{ trace(execname); }”

What happens when a probe is hit.

Consumers

dtrace – command line and scripting interface.

lockstat – kernel lock statistics.

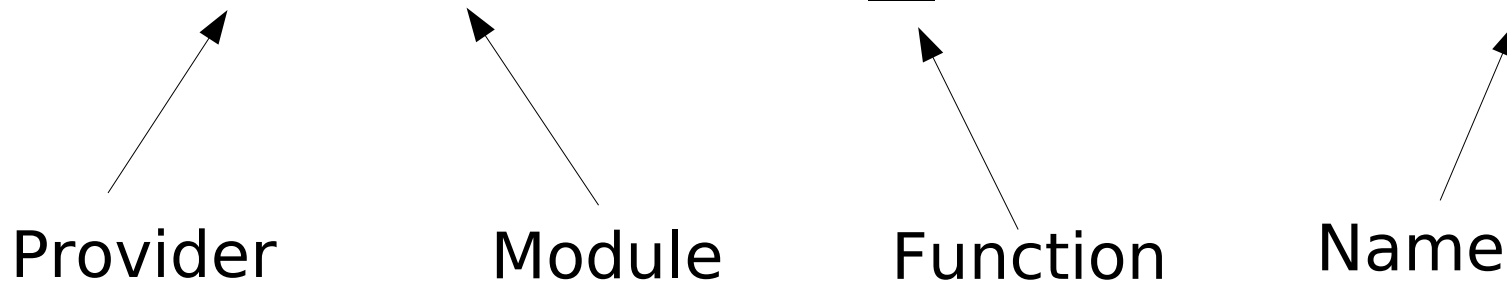
plockstat – user-level lock statistics.

intrstat – run-time interrupt statistics.

All are found in `/usr/sbin/` and you need to be root to execute them.

Probes

fbt : zfs : arc_read : entry



Many providers available:

- fbt Function Boundary Tracking (Kernel Functions)
- syscall Allows you to trace system calls.
- pid Process ID Tracking
- profile Profiling provider and many many more...

Actions

Actions are a set of instructions that are to be executed when a specific probe is fired.

This will print the executable name that caused the probe to fire.

```
{ trace(execname); }
```

This will print the name of the function that caused the probe to fire.

```
{ printf("%s", probefunc); }
```

And this will do both!

```
{  
    trace(execname);  
    printf("%s", probefunc);  
}
```

D Language

- Running on the command line isn't always powerful enough, and it's not reusable.
- The 'D' programming language allows you to create complex (and simple), reusable DTrace scripts.
- The syntax is similar to C and awk.

Representation of Data

- Raw data may not be useful without post-processing.
- Data can be aggregated (where single pieces of data are not useful) to allow patterns to be identified.
- Built in aggregation functions – count, sum, average etc.

DTrace Syntax

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

```
fbt:zfs:arc_read:entry,  
fbt:zfs:arc_write:entry  
/execname == "zpool" || execname == "zfs"/  
{  
    printf("%s caused %s to enter.", execname, probefunc);  
}
```

```
fbt:zfs:arc_read:return,  
fbt:zfs:arc_write:return  
/execname == "zpool" || execname == "zfs"/  
{  
    printf("%s caused %s to return.", execname, probefunc);  
}
```

And the rest..

- Dtrace has actions to record user stack traces within the kernel, user space programs, Java and Python.
- Additional providers out there:
 - > Java
 - > Python
 - > Ruby
 - > Perl
- As Solaris is Open Source, Dtrace has been implemented on Mac OS X 10.5 and FreeBSD (ongoing).

Where to find more information...

- To learn more:
 - > The Solaris Dynamic Tracing (DTrace) Guide:
<http://docs.sun.com/app/docs/doc/817-6223>
 - > DTrace Portal :
<http://www.sun.com/bigadmin/content/dtrace>
 - > Open Solaris:
<http://www.opensolaris.org/>
 - > Solaris Internals:
http://www.solarisinternals.com/wiki/index.php/DTrace_Topics



Michael Clarke
michael.clarke@sun.com

