

ORACLE[®]

DTrace on Linux

Elena Zannoni
Director, Languages and Tools
Linux Engineering

Kris Van Hees
Consulting Engineer
Linux Engineering

What is DTrace?

- Originally a Solaris tool, available since 2005
- Allows static tracing using instrumentation compiled into kernel and applications
- Allows dynamic tracing by defining probe points 'on the fly'
- Probes and actions at probe points are defined by scripts written in the 'D' language
- Many types of providers, main ones are
 - DTrace: BEGIN, END, ERROR probes
 - Syscall: entry and exit of each system call
 - Profile: fires at specific time intervals (dynamic probes)
 - sysinfo, vminfo, fpuinfo, sched, io, iscsi, etc.: Static tracing in kernel: probes at certain locations in subsystems
 - Pid: Static tracing in userspace applications: MySQL, Perl, Java
 - Pid: Dynamic tracing in userspace applications: can probe at every instruction in a running process
- Speculative tracing: filter events and data presented to user after probes fired (uses intermediate holding buffer)
- Solaris DTrace documentation:
 - <https://wikis.oracle.com/display/DTrace/DTrace>
 - <http://docs.oracle.com/cd/E19253-01/817-6223/index.html>

Why DTrace on Linux

- Plethora of tools on Linux with different usage cases, syntax, data format and outputs
- Lack of integrated user space tracing solution for Linux
- Want to offer compatibility with existing DTrace scripts for Solaris
- Expertise of Solaris user and administrators can be reused on Linux
- Customer demand

DTrace on Linux

- Integrated with Oracle Unbreakable Enterprise Kernel:
 - Version 0.2 currently available on UEK 2.6.39 (UEK2 GA)
 - Available as a separate technology preview kernel
- x86_64 only
- Kernel changes are GPL
- Kernel Module is CDDL
- Test suite ported

DTrace on Linux

- Initial release October 2011
- Current release is 0.2.5 -- April 2012
- Available on ULN channel: ol6_x86_64_Dtrace_BETA
- RPMs: Kernel, kernel modules, and userspace utilities
- Code posted on <http://oss.oracle.com/git/> (dtrace-0.2 branches)
 - linux-2.6-dtrace-modules-beta.git
 - linux-2.6-dtrace-unbreakable-beta.git
- Functionality currently available:
 - DTrace provider
 - syscall provider
 - SDT (statically defined tracing)
 - Profile provider (partial)
 - Proc provider

DTrace on Linux: Userspace

- `/usr/include/dtrace.h` *DTrace userspace consumer defs*
- `/usr/include/sys/ctf*` *Compact C type format definitions*
- `/usr/include/sys/dtrace*` *DTrace system definitions*
- `/usr/lib/libdtrace.so (& Co)` *DTrace userspace consumer library*
- `/usr/lib/dtrace` *DTrace userspace consumer D library*

- `/usr/sbin/dtrace` *DTrace userspace consumer*

DTrace on Linux: Providers

- Kernel modules:
 - dtrace – Dtrace core infrastructure
 - BEGIN, END, ERROR
 - profile – Timer based probes
 - tick-*n* (where *n* indicates frequency or time interval)
 - systrace – System call tracing (entry and return)
 - <syscall>:entry
 - <syscall>:return
 - sdt – Statically Defined Tracing (meta-provider)
 - <provider>::<function>:<probe>
 - E.g. proc::do_fork:create

DTrace on Linux: Kernel

- Kernel changes:
 - Writable system call table
 - Page fault die notifier
 - A few new fields in `task_struct`
 - Iterator over `kallsyms` info *[in development]*
- Kernel infrastructure that got added:
 - Enumeration of statically defined tracing probe points
 - Very minimal implementation of Solaris' `cyclics`
 - System call table manipulation code

DTrace on Linux: Evaluation?

- As reported on some blogs:
 - MacOS X: 154918 probes (many pre-created per process)
 - Delphix OS: 77320 probes
 - DTrace for Linux: 574 probes (release 0.1)
 - DTrace for Linux: 617 probes (release 0.2)
 - DTrace for Linux: 41916 probes (development)
- Utterly useless comparison
- FBT (Function Boundary Tracing): provides on average 2 probes per function in the kernel

DTrace userspace in general

- More straightforward porting
- Solaris-wide libraries needed porting (libproc, libctf).
- Most changes in OS specific code
- Maintain libdtrace API to the extent possible

CTF (Compact C Type Format)

- Developed for Solaris
- DTrace needs knowledge of user-defined C types for a lot of operations (e.g. structure members, function parameters local variables, understanding typedefs etc).
- Utility to convert kernel DWARF debugging info at compile time into CTF data in a new section of kernel and modules
- Removal of duplicate types occurring in both kernel and modules.
- DTrace userspace reads these sections from the modules directly

CTF, continued

- Limitation In Linux
 - Modular subsystems can be built into the kernel
 - Kallsyms info cannot distinguish the subsystems
 - DTrace scripts reference identifiers by module name
 - Choice to build as modules needs to be transparent
 - Need to keep track of lost namespace info
- Solution
 - Retain namespace info for subsystem
 - Export full identifier info via kallsyms equivalent

DTrace testsuite

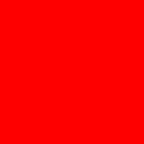
- Solaris Version:
 - Extensive suite of functionality tests
 - Driver script somewhat limited
 - No useful logging
 - Can only run the entire testsuite at once
 - Code duplication
 - No hang detection
- Linux version:
 - New testsuite engine
 - better logging
 - clearer output
 - automatic expected-result comparison

Performance Evaluation

- Run DTrace testsuite in parallel with OLT (Oracle Linux Testkit) testsuite
- Compare:
 - Lmbench on UEK2 (standard)
 - Lmbench on UEK2 with DTrace (no active probes)
 - Lmbench on UEK2 with DTrace (specific probes enabled)
- No performance difference at this point, with version 0.2 of DTrace

DTrace Next

- Complete the Profile provider
 - Add more SDT providers (more static probes instrumentation)
 - Function Boundary Tracing
 - CTF (Compact Type Format) support
 - Userspace application tracing (longer term)
-
- Oracle Forum available for DTrace for Linux
<https://forums.oracle.com/forums/forum.jspa?forumID=1398>



The preceding is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.