PRESENTED BY RED HAT

LEARN. NETWORK.
EXPERIENCE OPEN SOURCE.

www.theredhatsummit.com

Smiting Functional and Performance Problems with SystemTap

William Cohen, Jason Baron, and Dominic Duval

Red Hat
June 24, 2010

Outline

- Dynamic instrumentation
- SystemTap
- SystemTap “ready-to-run” examples
- Big Kernel Lock example
- Tracepoints and example
- User-space probing and example
- Common SystemTap techniques
- How to write your own scripts
- Where to get more information

Why Dynamic Instrumentation?

- Complicated systems:
  - Many possible reasons for a performance problem
  - Many components
  - Unexpected interaction between component
  - Unexpected use of components

Why Dynamic Instrumentation? (cont)

- Traditional debugging techniques unattractive:
  - Interrupt normal operation of system, e.g. gdb
  - Recompilation and re-installation of software
  - Tools only look at one executable or aspect of system at time, not the entire system

Dynamic instrumentation

- Allows instrumentation of running system
- Avoids interrupting already running processes
- Avoids rebuilding and re-installing software
- Reduces time to test out hypotheses

SystemTap

- Dynamic scriptable instrumentation tool
- Powerful scripting features:
  - Conditional constructs
  - Associative arrays
  - Statistics and histograms
- Number of safety checks to make ensure script benign:
  - Translator limit actions available
  - Run-time checks to limit overhead

How SystemTap Works

- SystemTap script contains:
  - Probe events
  - Probe handlers
- SystemTap steps to run a script:
  - Parses script
  - Elaborates (pulls in information from debuginfo/tapsets)
  - Translates
  - Compiles kernel module
  - Loads kernel module and collects data

Simple SystemTap Script

```
$ cat simple.stp
probe vfs.read {
  printf("read performed\n")
  exit()
}
$ stap simple.stp
read performed
$
```


SystemTap "Ready-to-Run" Scripts

- Building catalog of SystemTap scripts
- Catalog included in systemtap RPM, e.g.

  `/usr/share/doc/systemtap-1.1/examples`
- Have indices with short descriptions:

  `/usr/share/doc/systemtap-1.1/examples/index.html`

  `/usr/share/doc/systemtap-1.1/examples/index.txt`

Big Kernel Lock

- Big Kernel Lock (BKL) introduced in Linux 2.0
- Allowed for multiple processors
- Scaling problems (serialization of kernel code)
- Kernel developers working to replace BKL with finer-grain locking
- Some kernel sub-systems still use BKL:
  - NFS
  - SMB
  - TTY

Big Kernel Lock (BKL) Example

- `examples/locks/bkl.stp`
- One argument, number of threads waiting on BKL
- If number of waiting thread exceeded, print holding thread's:
  - Name
  - PID
  - Duration holding BKL

Tracepoints

- Callback located at strategic points in kernel
- Advantages:
  - Much faster than kprobes
  - Improved portability
- Incorporated in a number of subsystems: kvm, module, jb2, scsi, ext4, workqueue, skb, bkl, kmem, block, syscalls, lock, irq, signal, sched, wireless, gfs2, xfs

Tracepoints (cont)

- Increased use in upstream Linux kernel:
  - 2.6.28 - 12
  - 2.6.29 - 31
  - 2.6.30 - 45
  - 2.6.31 - 117
  - 2.6.32 - 209
  - 2.6.33 - 271
  - 2.6.34 - 282

Tracepoint example, schedtimes.stp

- `examples/process/schedtimes.stp`
- Can run system-wide
- Can focus on a single process (Optional)
- Traces amount of time process(es) spend:
  - Running
  - Sleeping
  - Waiting for IO
  - Queued
  - Total time

User-space probing

- SystemTap user-space probing uses the utrace mechanism
- Utrace mechanism designed to address issues with ptrace:
  - Traces changes to process (creation/exit/mmap)
  - Allows multiple engines to attach to a single process
- SystemTap requires kernel that includes utrace mechanism

User-space Probing Example

- `general/para-callgraph.stp`
- Traces program execution:
  - When function entered and parameters
  - When function exited with return value
- Flexibility in tracing:
  - Portion of code in some file
  - Trace all code in executable

Markers for User-Space Programs

- User-space Markers are similar to kernel tracepoints
  - Define interesting points in user-space code
  - Improve portability
- Fedora-13 and RHEL-6 packages with markers:
  - postgresql
  - java-1.6.0-openjdk
  - tcl
- See markers with:

  stap -L 'process(“executable_name”).mark(“*”)'

Common SystemTap Script Uses

- “Super strace”
- Determine whether particular function is called
- Get traceback to determine what is calling a function
- Examine arguments passed into or returned by a function
- Determine which process or thread is triggering an event
- Determine time between events

“Super Strace”

- Strace is a very useful tool
- strace limitations:
  - Only able to watch a single process
  - Limits on filtering (cannot filter on return values)
  - Can generate very verbose log
- Systemtap able to monitor syscalls system-wide
- Systemtap can have more flexible filtering, for example syscall return value < 0

Writing Your Own Systemtap Scripts

- Use existing examples as starting points
- Find possible probe points with "`-L`" option:

```
stap -L 'kernel.trace("*")'
stap -L 'process("a.out").function("*")'
```

- Systemtap man pages:

```
man -k 3stap
```

- Look through tapsets for probe points:

```
/usr/share/systemtap/tapset
```

- Look through the kernel sources

Navigating the Linux Kernel

- Linux kernel cross references (lxr)
- Red Hat Enterprise Linux kernels:
  `http://www.rhkernel.org`
- Upstream kernels
  `http://lxr.linux.no/linux/`

Where to get more information

- Red Hat SystemTap Beginner's Guide:
  - http://www.redhat.com/docs/manuals/enterprise/
- IBM Red Book:
  - http://www.redbooks.ibm.com/abstracts/redp4469.html
- SystemTap project page:
  - http://sourceware.org/systemtap/
- Forums for questions and help:
  - Email systemtap@sources.redhat.com
  - IRC #systemtap on irc.freenode.net

Get Your Script into the SystemTap Examples

- Submit scripts for the examples
- Get enthusiastic feedback on the script from SystemTap developers
- Make sure that script works on wide variety of environments, example scripts are run a part of testsuite
- More details about submitting examples in:

  `/usr/share/doc/systemtap-1.1/examples/README`

Questions?

FOLLOW US ON TWITTER

www.twitter.com/redhatsummit

TWEET ABOUT IT

#summitjbw

READ THE BLOG

http://summitblog.redhat.com/