Using SystemTap for Dynamic Tracing and Performance Analysis

Mike Mason
IBM Linux Technology Center
mmlnx@us.ibm.com

With assistance from:
Frank Ch. Eigler <fche@redhat.com>
Josh Stone <joshua.i.stone@intel.com>
Richard Moore <richardj_moore@uk.ibm.com>
Vara Prasad <varap@us.ibm.com>

Tutorial Objectives

- You will learn:
  - What SystemTap is and how it works
  - SystemTap basics: safety, scripts, commands, tapsets
  - How to use SystemTap, with examples on Live CD
  - How to set up SystemTap
  - How to contribute to SystemTap
  - How to get more information

SystemTap Live CD

- Fedora 7 x86
- SystemTap and it's prerequisites
- Slides and examples in /usr/share/doc/stap_tutorial-1.0/
- How to boot:
  - Insert CD in drive
  - Restart system
  - Make sure BIOS is set to boot from CD first
  - Be patient :-) Booting a live CD can take a while

What is SystemTap?

- A kprobes-based debugging and performance analysis tool for Linux – *makes using kprobes easy*
- A scripting language translator
- A flexible and extendable framework for *dynamic* instrumentation and creating new tools
- An open source project with contributors from Red Hat, IBM, Intel, Hitachi, Oracle and other community members

SystemTap Target Audience

- Kernel Developer: I wish I could add debug statements easily without going through the insert/build/reboot cycle.
- Technical Support: How can I get additional data out of a customer's kernel easily and safely?
- Application Developer: How can I improve the performance of my application on Linux?
- System Admin: Occasionally jobs take significantly longer than usual to complete, or do not complete. Why?
- Researcher: How would a proposed OS/hardware change affect system performance?
- Student: How can I learn more about the call flow of a kernel subsystem?

Kprobes

- An in-kernel interface for dynamic probing at any kernel code address
- Kprobes requires that you:
  - Write a kernel module
  - Specify an address and handler for each probe point
  - *Be careful!* Mistakes can crash the system.
- Powerful, but cumbersome to use “on-the-fly”

SystemTap + Kprobes

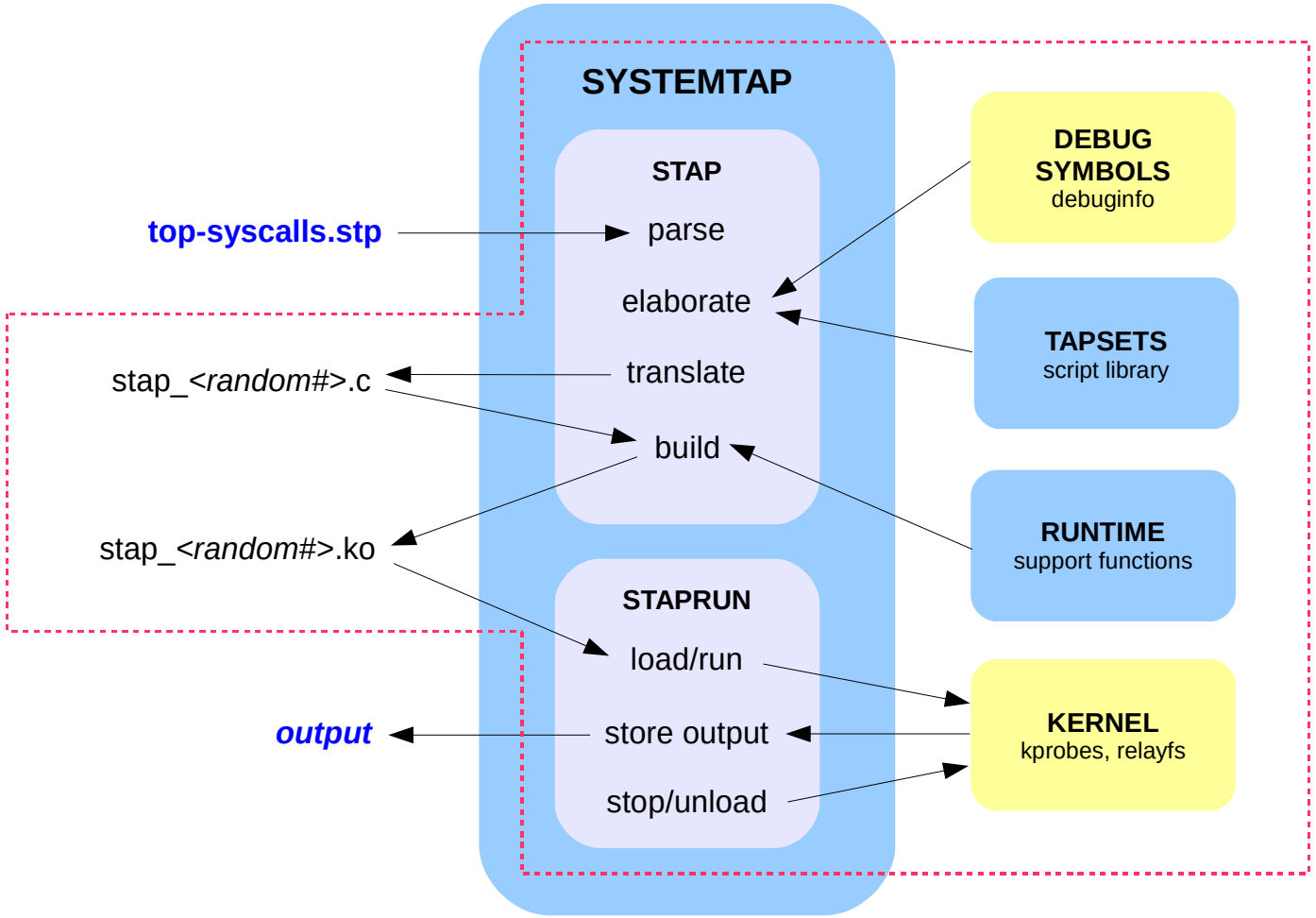
- No module writing required. Create and insert probes quickly and easily using a simple scripting language.
- No kprobes knowledge required
- No kernel addresses required. Automates gathering of symbol information.
- Enhances kprobes safety
- Provides pre-written probes for common kernel areas
- Growing set of pre-written scripts
- Powerful and simple to use

Example Script

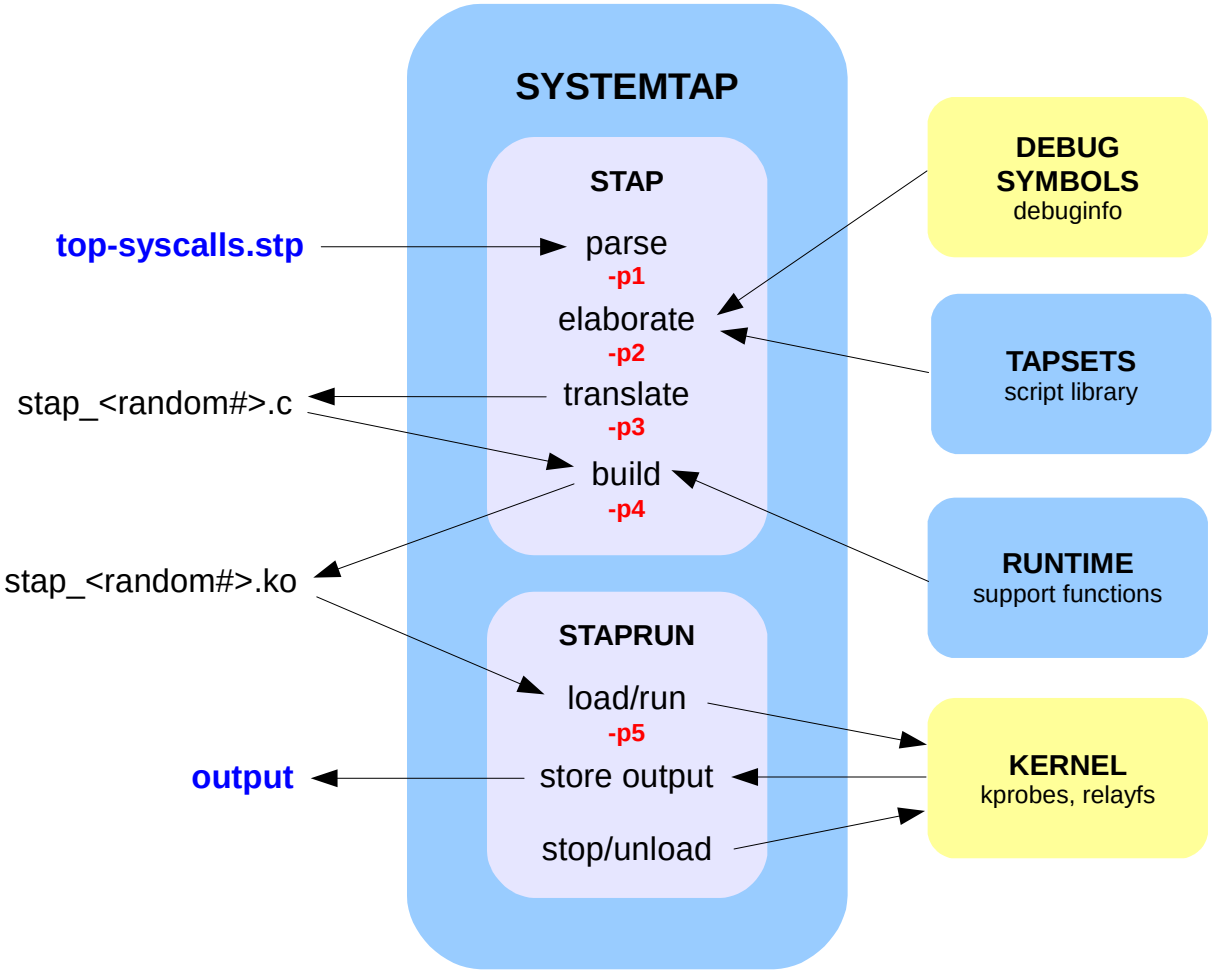
- top-syscalls.stp

```
global syscalls

probe begin {
        printf("Collecting data...\n")
}

probe syscall.* {
        syscalls[name]++
}

probe end {
        foreach (name in syscalls- limit 20)
                printf("%10d %s\n", syscalls[name], name)
}
```

How SystemTap Works



How SystemTap Works



SystemTap Basics

- Safety
- Commands
- Scripts
- Tapsets

Safety

- Built-in safety checks
  - Infinite loops and recursion
  - Excessive CPU overhead
  - Invalid variable access
  - Division by zero
  - Restricted access to kernel memory
  - Array bound checks
  - Version compatibility checks
  - Sensitive kernel functions blocked from probing via blacklist and __kprobes macro, primarily for locking reasons
- Language safety features
  - No dynamic memory allocation
  - Types and type conversions limited
  - Limited and protected pointer operations
  - Probes run with interrupts disabled (except begin/end probes) & preemption disabled
- Limits are configurable
  - e.g. MAXACTION limits statements run during probe hit, default 1000
  - See stap(5) for list

Stap Command

- **`stap [options] script.stp`**
- Example options

`script.stp`	Run this script
`-v`	Increase verbosity
`-g`	Guru mode, embedded C allowed
`-k`	Keep temporary directory
`-m MODULE`	Set probe module name
`-x PID`	Sets target() to PID
`-c CMD`	Start probes, run CMD, exit when it finishes
`-r RELEASE`	Cross-compile to kernel RELEASE
`-D NAME=VALUE`	Override limits

- See stap(5) man page for complete list and details

Scripting Language

- Probes & Probe Aliases – function entry & exit, source line #, kernel address, timer, begin/end
- Wildcarding
- Functions
- Types – string, 64-bit long, associative array, aggregation
- Comparison – *if else &* ternary operators
- Looping - *while, for, foreach*
- Usual binary & numeric operators
- String manipulation – *sprint, sprintf, . &* .= operators
- Output – *log, print, printf*
- Target variables – accessible with '$' prefix
- Embedded C – raw C code, not covered by safety checks

What is a Tapset?

- Probe set that encapsulates kernel subsystem knowledge – defines probes, data, auxiliary functions
- Abstracts away subsystem implementation details
- Isolates user scripts from kernel variations
- Probes are usable and extendable by other scripts
- Tested and packaged with SystemTap
- Located in either:
  - /usr/local/share/systemtap/tapset if installed from source
  - /usr/share/systemtap/tapset if installed from rpm
- See stapprobes(5) & stapprobes.*(5) man pages

Installation & Setup

- Three basic requirements
  - SystemTap
  - Kernel development environment – everything needed to build a module
  - Kernel debug symbols
- How you meet these requirements is distro dependent

Distro Availability

- Distributions
  - RHEL 4 U2+, RHEL 5
  - SLES 10 via maintenance web & SP1
  - Fedora 5, 6 & 7
  - Ubuntu 6.10 & 7.04, Debian GNU/Linux 4.0
  - Gentoo
- Architectures
  - x86
  - x86_64
  - ppc64
  - ia64
  - S390x (SLES 10 SP1 & RHEL 5)

Example: Installation on Fedora 7

- Install the required packages, temporarily enabling the debuginfo channels:

```
# yum --enablerepo=fedora-debuginfo --enablerepo=updates-debuginfo install kernel-debuginfo kernel-devel systemtap
```

- Verify your installation

```
# stap –e ‘probe begin { printf(“Hello World!\n”) }’
```

- That's it!
- See SystemTap wiki for other distros

Installation & Setup Issues

- Kernel debug symbols not always available
  - Debuginfo rpms (vmlinux+modules) available for Fedora 5/6/7, RHEL 4/5 and SLES 10, but not shipped on media. Must be downloaded.
  - Ubuntu has linux-image-debug (vmlinux only) available for download, vmlinux-dbg-<ver> must be renamed or linked
  - Debian & Gentoo – must build your own kernel with debug symbols
- Space requirements for debug files
  - e.g., debug files on live CD ~466MB
- Debug files must be in SystemTap's search path
- May not want debuginfo and/or kernel development environment on target machine – cross compilation helps with this

The Future

- User space probes
- Static marker probes
- More and better tapsets
- Allow non-root users to run scripts
- SystemTap Toolkit
  - Set of useful user scripts. Contributions encouraged!
  - See http://sourceware.org/systemtap/wiki/ScriptsTools
- SystemTap GUI
  - Eclipse-based IDE and graphing
  - Much improved graphing in progress
  - See http://sourceforge.net/projects/stapgui/

How to Contribute to SystemTap

- Use it! Let others know if you find it useful.
- Write scripts and post on wiki and mailing list
- Write tapsets in your areas of expertise
- Evaluate current tapsets
- Test in your environment and post results
- Submit and/or fix bugs
- Participate in discussions:
  - Mailing list: sign up at http://sourceware.org/systemtap/getinvolved.html
  - IRC: #systemtap on irc.freenode.net

How to Get More Information

- SystemTap Home Page
  - http://sourceware.org/systemtap
- Community Wiki (scripts, presentations/papers, etc.)
  - http://sourceware.org/systemtap/wiki
- Tutorial with language focus
  - HTML: http://sourceware.org/systemtap/tutorial/
  - PDF: http://sourceware.org/systemtap/tutorial.pdf
- Man pages (use “man –k stap” for a listing)
- Language Reference (coming soon, check wiki)
- Other documents
  - http://sourceware.org/systemtap/documentation.html

Legal Statement

This work represents the views of the author and does not necessarily reflect the views of IBM Corporation.

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States and/or other countries: IBM (logo), A full list of U.S. trademarks owned by IBM may be found at http://www.ibm.com/legal/copytrade.shtml.

Linux is a registered trademark of Linus Torvalds.

Other company, product, and service names may be trademarks or service marks of others.

Questions?

Backup Slides

Current Tapsets

tapset name (function entry, exit probe counts)

- ioblock (2,0)
- ioscheduler (3,1)
- lket (188,120)
- memory (7,1)
- network (2,0)
- nfs (111,111)
- process (5,1)
- rpc (23,23)
- scsi (4,0)
- signal (15,12)
- socket (8,10)
- syscall (395,400)
- tcp (3,3)
- udp (3,3)
- vfs (21,21)

Build Kernel with SystemTap Support

- Build with these options enabled:
  - CONFIG_DEBUG_INFO
  - CONFIG_KPROBES
  - CONFIG_RELAY
  - CONFIG_DEBUG_FS
  - CONFIG_MODULES
  - CONFIG_MODULE_UNLOAD
- Install kernel and reboot
- Make sure *unstripped* vmlinux and modules are in SystemTap's debug info search path

Debug Info Search Path

- vmlinux
  - /boot/vmlinux-`uname -r`
  - /usr/lib/debug/lib/modules/`uname -r`/vmlinux
  - /lib/modules/`uname -r`/vmlinux
- Modules
  - /usr/lib/debug/lib/modules/`uname -r`
  - /lib/modules/`uname -r`
- Build directory
  - /lib/modules/`uname -r`/build

Build SystemTap from Latest Source

- Get latest elfutils source
  - ftp://sources.redhat.com/pub/systemtap/elfutils/elfutils-NNNN.tar.gz
    (latest NNNN is 0.128)
  - ftp://sources.redhat.com/pub/systemtap/elfutils/elfutils-portability.patch
  - Untar and apply patch
- Download SystemTap source (weekly snapshot or CVS)
  - ftp://sources.redhat.com/pub/systemtap/snapshots/latest.tar.bz2 ***or****
  - cvs -d :pserver:anoncvs@sources.redhat.com:/cvs/systemtap login
    # enter "anoncvs" as the password
    cvs -d :pserver:anoncvs@sources.redhat.com:/cvs/systemtap co src
- Build and install it
  - cd src
  - ./configure –with-elfutils=*PATCHED-ELFUTILS-DIR*
  - make
  - make install