Dailymotion

Quick introduction to Systemtap

Vincent Bernat

November 2013

The problem we are trying to solve

Source: styeb on Flickr
CC BY SA 2.0

Extract information from running systems

- Low overhead
- From kernel
- From userland processes
- Information unavailable by other ways

Existing solutions

- Classic tools: `top`, `sar`, `vmstat`, `mpstat`, `iostat`, `netstat`, `iotop`, `strace`, `gdb`, `cat`, …
- Tracing tools:
  - DTrace (Solaris and Linux)
  - LTTng (Linux)
  - Systemtap
  - ktap

How Systemtap works?

Source: Pete + Lynne on Flickr
CC BY ND 2.0

General view

- Write code to be **executed on events**
- Events can happen in **kernel** or in **userland**
- Use its own **C-like** language
- Translated to **C** code
- Compiled to a **kernel module**

Events

An event can be:

- A function entry: `kernel.function("vfs_read")`
- A function exit: `kernel.function("vfs_read").return`
- A statement: `kernel.statement("*@mm/quicklist.c:56")`
- A trace point: `kernel.mark("kfree_skb")`
- A timer: `timer.s(5)`
- A static probe: `kernel.mark("context_switch")`
- ...

More events

All the previous examples can be adapted for userland!

- A function entry:
  `process("php5").function("php_request_shutdown")`
- A function exit:
  `process("php5").function("php_request_shutdown").return`
- A statement:
  `process("php5").statement("*@zend/zend_alloc.c:56")`
- A static probe:
  `process("php5").provider("php").mark("request__shutdown")`
- ...

Simple example

```
# cat strace-open.stp
probe syscall.open
{
  printf ("%s(%d) open (%s)\n", execname(), pid(), argstr)
}
probe timer.ms(4000) # after 4 seconds
{
  exit ()
}

# stap strace-open.stp
awesome(2394) open ("/sys/class/net/docker0/carrier", O_RDONLY)
tmux(3219) open ("/proc/31978/cmdline", O_RDONLY)
tmux(3219) open ("/proc/31978/cmdline", O_RDONLY)
```

Another example

```
# cat socket-trace.stp
probe kernel.function("*@net/socket.c").call {
  printf ("%s -> %s\n", thread_indent(1), probefunc())
}
probe kernel.function("*@net/socket.c").return {
  printf ("%s <- %s\n", thread_indent(-1), probefunc())
}

# stap socket-trace.stp
     3 urxvtd(3216):  -> SYSC_recvfrom
     5 urxvtd(3216):   -> sockfd_lookup_light
     7 urxvtd(3216):   <- SYSC_recvfrom
     8 urxvtd(3216):   -> sock_recvmsg
```

Last example

```
# cat cookies.stp
global sizes;
probe process("/usr/lib/libapr-1.so.0").function("apr_table_addn") {
    if (user_string2($key, "") == "Cookie") {
      size = strlen(user_string2($val, ""));
      sizes <<< size;
    }
}
probe timer.s(1) { print(@hist_log(sizes)); }
# stap cookies.stp
value |-------------------------------------------------- count
   64 |@@@@@                                              10
  128 |@@@@                                                9
  256 |@@@@@@@@@@@@@@@@@@@@                               40
  512 |@@@@@@@@@@@@@@@@@@@@@@@@@@                         52
 1024 |@@@@@@@@@@@@@@@@@                                  34
 2048 |@@@                                                 7
```

Documentation

- Tutorial
- Beginners Guide
- Language reference
- Available tapsets (helper functions)

How to use it at Dailymotion?

Source: ericbegin on Flickr
CC BY NC ND 2.0

bcfg2

- Groups: web+debug, memcache+debug
- Bundle: debug

```
<Bundle name="debug">
  <Path name="/etc/apt/sources.list.d/debug.list"/>
  <Package name="systemtap"/>
  <Package name="linux-image-3.11.0-13-generic"/>
  <Package name="linux-image-3.11.0-13-generic-dbgsym"/>
  <Package name="gcc"/>
  <Package name="gdb"/>
  <Package name="python-jinja2"/>
</Bundle>
```

Debug packages

- They are needed for most tasks
- Need a **recent kernel** to get accurate debug symbols for kernel
- We mirror ddebs.ubuntu.com to get **debug packages** (end with `-dbgsym`)
- Currently, we don't have `-dbgsym` packages for our own packages. Need to build them by hand.

Cookbook

- Systemtap comes with a lot of examples (mostly kernel related)
- We wrap our own scripts into **Python scripts**: self-documenting, better argument handling, Jinja templating
- Currently available in github.com/vincentbernat/systemtap-cookbook
- Mostly PHP related, a bit of Apache, IO and TCP stuff (listening queue monitoring!)

Cookbook (continued)

```
usage: php time [-h] [--function FN] [--slow] [--step MS] [--log]
                [--interval INTERVAL] [--uri PREFIX] [--php PHP]

Distributions of response time for PHP requests.


optional arguments:
  -h, --help           show this help message and exit
  --function FN        profile FN instead of the whole request
  --slow               log slowest requests
  --step MS            each bucket represents MS milliseconds
  --log                display a logarithmic histogram
  --interval INTERVAL  delay between screen updates in milliseconds
  --uri PREFIX         restrict the profiling to URI prefixed by PREFIX
  --php PHP            path to PHP process or module
```

Cookbook (continued)

```
# ./php time --uri /video --slow --log
value |-------------------------------------------------- count
    1 |                                                   0
    2 |                                                   0
    4 |                                                   1
    8 |@@                                                 4
   16 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@            76
   32 |@@@@@@@@@@@@@@@                                   30
   64 |@@@@@@@@@@@@@@@@@@@@@@@@@                         51
  128 |@@@@@@@                                           14
  256 |@@@@@                                             11
  512 |                                                   0
 1024 |                                                   0

 - min:6ms avg:78ms max:478ms count:187
 - slowest requests:
     372ms:   GET /video/xs32g1_animekage-sayonara-zetsubou-sensei-ep-11-ro_shortfilms
     342ms:   GET /video/k49PDd47J2x7qG3IOIq
```

Dailymotion

Questions?